Lösen des Mehrklassenproblems der Sekundärstrukturvorhersage von Proteinen

Dissertation zur Erlangung des akademischen Grades des

Doktors der Naturwissenschaften (Dr. rer. nat.)

eingereicht im Fachbereich Biologie, Chemie, Pharmazie

der Freien Universität Berlin

vorgelegt von

Dawid Rasiński

aus

Bydgoszcz, Polen

2011

Diese Arbeit wurde in dem Zeitraum von 23. November 2006 bis 21. März 2011 unter der Leitung von Prof. Dr. Ernst Walter Knapp im Institut für Chemie und Biochemie angefertigt.

Erstgutachter:	Prof. Dr. Ernst-Walter Knapp
Zweitgutachter:	Prof. Dr. Alexander Bockmayr

Disputation am 30. Juni 2011

Inhaltsverzeichnis

1. Einleitung

Seit den sechziger Jahren des vergangenen Jahrhunderts, als der Aufbau von Proteinen und seine Bedeutung für deren Funktion immer besser verstanden wurde, wuchs das Interesse der Wissenschaft an der Struktur von Proteinen. Sehr schnell wurde klar, dass die räumliche Struktur eines Proteins für seine Funktion bestimmend ist, was dazu führte, dass strukturelle Analysen für das Verständnis der Funktion von Proteinen an Wichtigkeit gewannen. Zudem zeigte sich schnell, dass die Sequenz eines Proteins seine räumliche Struktur bestimmt. Auf dieser Grundlage und der damals geringen Anzahl an bekannten Proteinstrukturen wurde schnell die Annahme aufgestellt, dass die Struktur eineindeutig von der Sequenz bestimmt ist, also dass die Sequenz eine Art Fingerabdruck der räumlichen Struktur eines Proteins darstellt. Dies ließ die Idee aufkommen, dass eine Methode zur Bestimmung der Struktur eines Proteins einer gegebenen Sequenz auf einfache Weise konzipiert werden könnte. Mit zunehmender Anzahl bekannter Proteinstrukturen, die durch Methoden wie Röntgenkristallographie oder NMR-Spektroskopie bestimmt wurden, zeigte sich, dass das Problem nicht so einfach war wie anfangs gedacht. Aus den Daten ging hervor, dass die Struktur eines Proteins zwar von dessen Sequenz, jedoch keineswegs eineindeutig abhängt. Tatsächlich erwies sich das Problem der Proteinfaltung, also das Problem der Bestimmung der räumlichen Struktur eines Proteins aus seiner Sequenz allein, als ein derart komplexes Problem, dass bis zum heutigen Tag keine exakte Lösung existiert.

Die Stabilität und damit auch die Bildung von Proteinstrukturen hängt von einigen wenigen bekannten Kräften ab, die sich in einem empfindlichen, metastabilen Gleichgewicht befinden. Dadurch liegt es einerseits nahe, dass ein mathematisches Modell für die Bildung von Proteinstrukturen möglich sein sollte. Andererseits spricht die Metastabilität und das Fehlen der Kenntnis um das genaue Zusammenspiel der einzelnen Kräfte dafür, dass die Aufstellung eines solchen Modells keine einfache Aufgabe darstellt. Als Computer und Methoden aus Informatik und Informationstheorie immer mehr in der Wissenschaft Fuß fassten, wurde klar, dass ihre Möglichkeiten zur Berechnung geeigneter Modelle der Proteinfaltung genutzt werden kann. Bereits früh wurde dabei erkennbar, dass die Aufgabe der Bestimmung der räumlichen Struktur (also der Tertiärstruktur) anhand der Sequenz deutlich vereinfacht werden kann, wenn die Verteilung der Sekundärstruktur (also lokal auftretender Muster im räumlichen Verlauf des Proteins) entlang der Sequenz bekannt ist. Dies

führte zur Definition des Problems der Sekundärstrukturvorhersage von Proteinen, das bis heute eine wichtige Problemstellung der Molekularbiologie und Biochemie ist.

Methoden zur Sekundärstrukturvorhersage von Proteinen haben das Ziel, zu jeder Position innerhalb der Sequenz den Sekundärstrukturtyp zu bestimmen, dem diese angehört. Erste Methoden verwendeten dabei einfache statistische Modelle für das Auftreten der einzelnen Aminosäuren in bestimmten Sekundärstrukturtypen. Die Genauigkeit dieser Methoden sprach jedoch für die Entwicklung verbesserter Methoden zur Vorhersage der Sekundärstruktur. Mit der zunehmenden Rechenleistung von Computern, der stetig steigenden Menge an bekannten Proteinstrukturen und den Fortschritten im Bereich der Informatik, insbesondere der künstlichen Intelligenz und des maschinellen Lernens, wurden mit der Zeit immer bessere Methoden zur Sekundärstrukturvorhersage entwickelt. Moderne Methoden, wie das an der Freien Universität Berlin von Francesco Bettella entwickelte Programm SPARROW [1], basieren vollständig auf Konzepten aus dem maschinellen Lernen und der diesem zugrunde liegenden Lerntheorie und erreichen bei ihren Vorhersagen Genauigkeiten im Bereich von 80%. Diese Zahl zeugt einerseits von der Güte heutiger Methoden, andererseits aber auch davon, dass weitere Verbesserungen auf dem Gebiet der Sekundärstrukturvorhersage von Proteinen möglich sind.

Das Ziel der vorliegenden Arbeit ist die Entwicklung eines neuen Programms zur Vorhersage der Sekundärstruktur von Proteinen, das als verbesserter Nachfolger SPARROWs dienen soll. Zur Umsetzung dieses Programms mit dem Namen *SPARROW soll ein vollkommen neues Lernverfahren entwickelt werden, dass für das konkrete Problem der Sekundärstrukturvorhersage, wie auch allgemein zur Lösung ähnlicher Probleme verwendet werden kann. Dabei soll die allgemeine Fähigkeit dieses neuartigen Lernverfahrens Mehrklassenprobleme, wie das der Sekundärstrukturvorhersage eines ist, anhand der Anwendung in *SPARROW überprüft werden. Zudem soll ein Vergleich zwischen *SPARROW und anderen Verfahren zur Sekundärstrukturvorhersage von Proteinen, darunter SPARROW, gezogen werden, um die Güte dieses neuen Vorhersageprogramms mit bereits existierenden, vergleichbaren Programmen in Beziehung zu setzen.

1.1. Gliederung dieser Arbeit

Aufgrund der Tatsache, dass diese Arbeit dem interdisziplinären Feld der Bioinformatik zuzuordnen ist und sowohl ein konkretes biologisches Problem löst als auch eine neuartige

Lernmaschine präsentiert, wurde sie in einer Weise konzipiert, dass sie sowohl einem Biologen oder Chemiker, der in den Methoden des maschinellen Lernens nicht bewandert ist, als auch einem Informatiker ohne Vorkenntnisse der zugrunde liegenden Biochemie zugänglich ist. Aus diesem Grund beschreiben die ersten beiden Kapitel dieser Arbeit sowohl die biologischen Grundlagen (Kapitel 2) als auch die Grundzüge der verwendeten Lerntheorie (Kapitel 3) in relativ ausführlicher Weise. Das darauf folgende Kapitel 4 beschreibt das neuartige Lernverfahren sowie den Aufbau von *SPARROW, das mit Hilfe dieses Verfahrens umgesetzt wurde. Den Ergebnissen, die in den im Rahmen dieser Arbeit durchgeführten Versuchen mit *SPARROW erzielt wurden, ist Kapitel 5 gewidmet. Dieses enthält sowohl Ergebnisse zur Untersuchung des allgemeinen Verhaltens von *SPARROW, wie auch einen Vergleich mit anderen Vorhersageprogrammen. Schließlich enthält Kapitel 6 eine weiterführende Diskussion der Ergebnisse während Kapitel 7 einige Möglichkeiten zur Verbesserung *SPARROWs, wie auch des neuen Lernverfahrens, beinhaltet. Kapitel 8 bildet eine abschließende Zusammenfassung dieser Arbeit.

2. Biologische Grundlagen

Proteine sind biologische Makromoleküle, die in allen lebenden Organismen auf der Erde vorkommen und in der Funktionsweise der einzelnen Organismen eine Vielfalt von Aufgaben übernehmen. Proteine wie beispielsweise *Keratine* und *Kollagene* haben wichtige strukturelle Aufgaben im Körper von Lebewesen. Keratine bilden Federn, Haare, Fingernägel/Klauen, Hörner, Schuppen und Schnäbel höherer Lebewesen und sind somit maßgeblich an den Schutz- und Verteidigungsmaßnahmen des Organismus gegen äußere Einflüsse und Feinde beteiligt. Kollagene bestimmen die Struktur von Haut, Bindegewebe und Knochen im Körper. Sie sind für den Aufbau von Zellen und Gewebe von zentraler Bedeutung und sind damit bestimmend für den gesamten Körperbau des jeweiligen Individuums. Auch Bewegungen werden mit Hilfe von Proteinen ausgeführt. *Myosine* beispielsweise sind so genannte *Motorproteine*, die im Muskelgewebe von Lebewesen vorkommen und durch Veränderungen ihrer Form die Kontraktionen des Muskelgewebes hervorrufen. *Enzyme* wiederum sind körpereigene Katalysatoren, die die biochemischen Vorgänge im Körper ermöglichen beziehungsweise kontrollieren. Ein Großteil der für die Verdauung notwendigen Prozesse, also die Aufspaltung der aufgenommenen Nahrung in Nährstoffe und die Umwandlung dieser Nährstoffe in vom Körper verwertbare Formen, wird von Enzymen durchgeführt. Proteine sind auch in Form von *Kanalproteinen* an der Signalweiterleitung im Nervensystem beteiligt, indem sie den Austausch von Ladungsträgern zwischen dem Inneren und dem Äußeren einer Nervenzelle beeinflussen und somit die elektrische Spannung über der Zellmembran regulieren. Dies ist jedoch nicht die einzige Form von Informationstransport im Körper, an der Proteine beteiligt sind: Zellen besitzen an ihrer Membran so genannte *Rezeptoren.* Diese Proteine erkennen bestimmte Stoffe außerhalb der Zelle und können diese binden. Das Binden führt zu einer Veränderung der dreidimensionalen Struktur des Rezeptors, was von der Zelle als Signal interpretiert werden und diese zu bestimmten Reaktionen veranlassen kann. Rezeptoren sind stoffspezifisch, was bedeutet, dass ein bestimmter Rezeptor in der Regel nur einen bestimmten Stoff bindet beziehungsweise erkennt. Man spricht in diesem Fall vom sogenannten *Schlüssel-Schloss-Prinzip*: Stellt man sich den Rezeptor als ein Schloss vor und die potentiell zu bindenden Stoffe als Schlüssel, dann passt immer nur ein bestimmter Schlüssel in ein bestimmtes Schloss. Dieses einfache Prinzip ermöglicht dem Körper beispielsweise die Steuerung der Vorgänge in einer Zelle mit Hilfe von *Hormonen*, biochemischen Boten-

stoffen, was wiederum zu einer Regulierung des gesamten Körperhaushalts beiträgt. Hormone werden vom Blut durch den Körper transportiert und können somit überall im Körper intrazelluläre Prozesse wie Stoffproduktion oder -abbau beeinflussen. Dabei ist die Konzentration eines bestimmten Hormons im Blut proportional zu dem von ihm hervorgerufenen Effekt. Eines der bekanntesten Hormone ist das *Insulin*, das für die Regulierung des Blutzuckerspiegels zuständig ist und durch seine Rolle bei der Diabeteserkrankung in das Bewusstsein der Öffentlichkeit getreten ist. Gleichzeitig ist das Insulin ein Beispiel für ein Protein, welches als Hormon fungiert, denn nicht alle Hormone sind zwangsläufig auch Proteine. Proteine sind also auch als Hormone an der Regulierung und am Informationstransport in Organismen beteiligt. Eine weitere Funktionsgruppe von Proteinen sind die sogenannten *Transportproteine*, die für den Transfer von chemischen Verbindungen zuständig sind. Ein prominentes Beispiel hierfür ist das *Hämoglobin,* welches den Sauerstofftransport im Blut übernimmt. Zu guter Letzt soll an dieser Stelle das Immunsystem Erwähnung finden, an dessen Funktionsweise Proteine in Form von *Immunoglobulinen,* auch als *Antikörpern* bezeichnet, maßgeblich beteiligt sind. Antikörper sind Gegenmaßnahmen des Körpers gegen körperfremde Schadstoffe, so genannte *Antigene*. Antikörper funktionieren ähnlich den Rezeptoren nach dem Schlüssel-Schloss-Prinzip, das heißt, dass jeder Antikörper ein spezifisches Antigen binden und damit in Kooperation mit anderen Bestandteilen des Immunsystems unschädlich machen kann.

Proteine sind somit für das Funktionieren von Organismen unabdingbar und werden nicht umsonst als Bausteine des Lebens bezeichnet. Es ist also nachvollziehbar, dass die Wissenschaft ein hohes Interesse am Verstehen der Funktionsweise von Proteinen hat. Denn das Verständnis der Funktionsweise von Proteinen trägt zum Verständnis des Lebens selbst und seiner Ausprägungen bei. Hinzu kommt, dass ein detaillierteres Verständnis der Vorgänge in Organismen die Entwicklung effizienterer Medikamente und Behandlungsmöglichkeiten für bisher unheilbare Krankheiten ermöglichen könnte. Viele Medikamente wirken beispielsweise über die bereits erwähnten Rezeptoren, indem die enthaltenen Wirkstoffe ins Blut gelangen und entweder an die Rezeptoren körpereigener Zellen oder der Krankheitserreger binden und deren Funktion beeinflussen. So können Körperzellen dazu angeregt werden bestimmte Stoffe auszusenden oder deren Produktion einzustellen, was den Genesungsprozess des Körpers beschleunigen oder die Symptome der Erkrankung hemmen kann. Das Binden an den Krankheitserreger wiederum kann diesen entweder

davon abhalten Schadstoffe auszuschütten, ihn ganz neutralisieren oder für das Immunsystem leichter bekämpfbar machen. Die Fragen um die Funktion von Proteinen sind also ein wichtiges Thema moderner Forschung. Bevor jedoch die Funktion von Proteinen verstanden werden kann, muss erst ihr Aufbau verstanden werden. Das folgende Kapitel soll einen kleinen Einblick in das bisher gesammelte Wissen geben. Bevor jedoch direkt auf den Aufbau von Proteinen eingegangen werden kann, sollen zunächst die *Aminosäuren* angesprochen werden, aus denen sich Proteine zusammensetzen.

2.1. Aminosäuren

Aminosäuren (engl. *amino acids*) sind organische Moleküle bestehend aus mindestens einer Carboxylgruppe (COOH) und (bis auf *Prolin*) einer primären Aminogruppe (NH_2). In den meisten Fällen sind diese beiden Gruppen an einem gemeinsamen Kohlenstoffatom gebunden, welches als C_α-Atom bezeichnet wird. Die entsprechenden Aminosäuren werden als α-Aminosäuren bezeichnet [2].

Die grundlegende Struktur von Aminosäuren ist in der folgenden Abbildung 1 dargestellt:

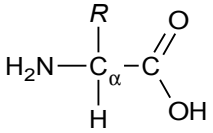


Abbildung 1: *Grundstruktur von Aminosäuren*

In der Abbildung repräsentiert *R* die so genannte *Seitenkette* (engl. *side chain*), welche von Aminosäure zu Aminosäuren unterschiedlich ist. Die Komplexität der Seitenkette kann zum Teil stark variieren und reicht von einem einfachen Wasserstoffatom (im Fall von *Glycin*) bis hin zu komplexeren Strukturen wie aromatischen Ringen (im Fall von *Phenylalanin*). Dadurch unterscheiden sich die Aminosäuren in ihren Eigenschaften, wie zum Beispiel ihrer elektrischen Ladung oder ihrer Wasserlöslichkeit.

Untersuchungen natürlich vorkommender Proteine haben ergeben, dass diese sich aus einer Menge von 20 Standardaminosäuren zusammensetzen. Zwar existieren Derivate dieser Standardaminosäuren, die auch in Proteinen beobachtet wurden, jedoch werden diese Derivate in der Regel nach dem Aufbau des jeweiligen Proteins vom Körper nachträglich aus der ursprünglichen Standardaminosäure erzeugt [2]. Der Vorgang kann als eine Art Nachbearbeitungsschritt nach der Fertigung angesehen werden. Die Strukturformeln der 20

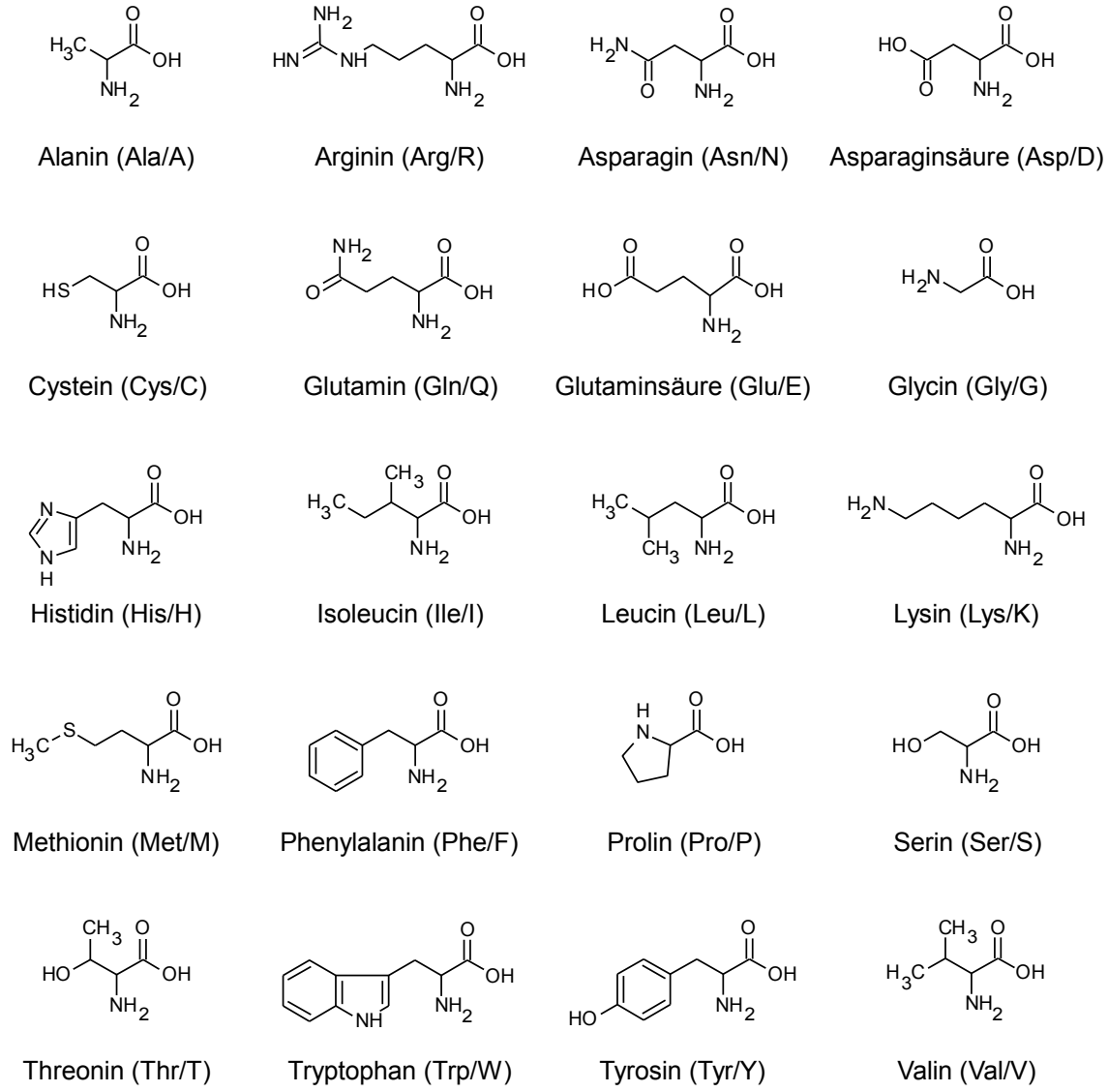


Abbildung 2: *Die formalen zweidimensionalen Strukturen der 20 Standardaminosäuren*

Standardaminosäuren, die das Rohmaterial für Proteine darstellen, sind in Abbildung 2 gezeigt. Unter den Strukturen stehen neben dem jeweiligen Namen auch die abkürzende Bezeichnungen der einzelnen Aminosäuren. Dabei gibt es zwei Konventionen, den *Dreibuchstabencode* und den *Einbuchstabencode*, die beide in Klammern hinter dem entsprechenden Aminosäurenamen stehen. Im Dreibuchstabencode wird jeder Aminosäure, wie der Name schon sagt, ein eindeutiger Code aus drei Buchstaben zugewiesen, der als Abkürzung für die Aminosäure verwendet werden kann. Der Einbuchstabencode hingegen weist jeder Aminosäure einen einzigen Buchstaben zu, der stellvertretend für den Namen der Aminosäure verwendet werden kann. So hat beispielsweise die Aminosäure Glycin den Dreibuchstabencode „Gly“ und den Einbuchstabencode „G“, während Tryptophan entspre-

chend „Trp“ und „W“ als Abkürzungen besitzt.

Die in Abbildung 2 dargestellten Strukturformeln sind die formalen zweidimensionalen Strukturen von Aminosäuren. Je nachdem, ob Aminosäuren als Feststoffe oder in Lösung vorhanden sind, können die im Allgemeinen neutralen Amino- und Carboxylgruppen unter Umständen eine positive oder negative Ladung besitzen (also protoniert beziehungsweise deprotoniert sein). In welchem Ausmaß dies der Fall ist wird über den *pH-Wert* des Mediums und die *pK_s-Werte* der jeweiligen protonierbaren molekularen Gruppen bestimmt.

2.1.1. Der pH-Wert und seine Auswirkung auf Aminosäuren

Der pH-Wert einer Lösung bestimmt sich aus dem negativen Logarithmus der Konzentration in ihr enthaltener überschüssiger Protonen:

$$\mathrm{pH} = -\log[\mathrm{H}^+]$$

Diese Formel geht auf Søren Sørensen zurück, der sie im Jahre 1909 im Rahmen einer Studie über die Bedeutung der Konzentration von Wasserstoffionen für enzymatische Prozesse [3] aufstellte. Heute wird der pH-Wert vor allem im Zusammenhang mit der *Acidität* beziehungsweise *Basizität* von Lösungen verwendet. Diese beiden Attribute beschreiben die Eigenschaft einer Lösung, als Säure beziehungsweise Base zu fungieren. Zur weiteren Begriffserklärung muss jedoch zunächst die Definition einer Base und einer Säure gegeben werden. Hierzu soll die von Johannes Brønsted und Thomas Lowry 1923 aufgestellte Definition herangezogen werden. Diese besagt, dass eine Säure eine Substanz ist, die als *Protonendonor* fungieren kann, also die Fähigkeit besitzt Protonen (beziehungsweise Wasserstoffionen) abzugeben. Eine Base hingegen ist ein Stoff, der Protonen aufnehmen kann, also als *Protonenakzeptor* fungieren kann. Mit dieser Definition ergibt sich, dass Stoffe mit Säureeigenschaft einen niedrigen pH-Wert haben müssen: In Säurelösungen müssen viele freie Protonen vorhanden sein, da diese von den Säuremolekülen in ihrer Eigenschaft als Protonendonor an die Lösung abgegeben werden. Im umgekehrten Fall enthalten basische Lösungen eine niedrige Konzentration an Protonen, da Basenmoleküle als Protonenakzeptoren möglichst viele Protonen aus der Lösung permanent binden. Damit haben Basen einen hohen pH-Wert.

Reines Wasser, beziehungsweise eine neutrale Lösung, hat einen pH-Wert von 7 und ist damit weder sauer noch basisch. Säuren haben einen pH-Wert, der kleiner ist als 7, wobei

ein geringerer pH-Wert für stärkere Säureeigenschaften spricht. Ein kleinerer pH-Wert einer Lösung bedeutet also, dass diese Lösung viele freie Protonen enthält. Basische Lösungen, mit pH-Werten oberhalb von 7, neigen umso eher dazu Protonen aufzunehmen, je höher ihr pH-Wert ist.

Das Lösen einer Säure in wässriger Lösung führt in der Regel (je nach pH-Wert der Lösung) zur Abgabe von Protonen seitens der Säure. Dadurch verliert diese ihre Eigenschaft als Säure, kann aber dafür als Protonenakzeptor fungieren, was ihr wiederum basische Eigenschaften verleiht. Die so entstandene Base kann ihrerseits Protonen aufnehmen und somit wieder das Verhalten einer Säure als Protonendonor annehmen. Dies ist eine sehr vereinfachte Darstellung einer solchen Lösung. In der Realität verändern einzelne Moleküle der Lösung ständig ihr Verhalten von sauer zu basisch und umgekehrt. Dies kann mittels der folgenden Gleichgewichtsreaktion dargestellt werden:

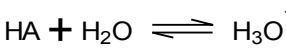


Abbildung 3: *Gleichgewichtsreaktion in wässriger Lösung*

Hierbei reagiert auf der linken Seite der Gleichung die Säure (HA) mit dem Wasser, das in diesem Fall die Rolle der Base übernimmt, zu einer *konjugierten Säure* (H_3O^+) und einer *konjugierten Base* (A^-). Diese können ihrerseits zu der ursprünglichen Paarung von Säure und Base reagieren. Je nach pH-Wert der Lösung, in der diese Reaktionen stattfinden, und der beteiligten Säure beziehungsweise Base, kann das Gleichgewicht zu Gunsten einer der beiden Seiten der Gleichung verschoben sein. Der pH-Wert, bei dem sich ein Gleichgewicht zwischen beiden Seiten der Gleichung, also zwischen den Konzentrationen der Säure und der Base und denen von konjugierter Säure und konjugierter Base, einstellt wird als pK_s-Wert bezeichnet. Dieser ist definiert als:

$$pK_s = -\log\left(\frac{[H_3O^+][A^-]}{[HA]}\right)$$

Der pK_s-Wert, auch als *Säurekonstante* bezeichnet, ist stoffspezifisch und gibt Aufschluss darüber, in welchem Maße der betreffende Stoff in der Gleichgewichtsreaktion mit Wasser unter *Protolyse* (also einer chemischen Reaktion unter Austausch eines Protons zwischen beiden Reaktionspartnern) reagiert. Säuren in einer wässrigen Lösung, deren pH-Wert über dem pK_s-Wert der Säure liegt, liegen eher in Form ihrer konjugierten Base vor. Im Fall von

Basen begünstigt ein pH-Wert unterhalb des pK_s-Wertes der Base eine hohe Konzentration der entsprechenden konjugierten Säure in der Lösung.

Die Aminogruppe der Aminosäuren ist eine basische funktionale Gruppe mit einem pK_s-Wert von ca. 9,4. Dieser Wert ist ein Mittelwert über die 20 Standardaminosäuren [2]. Für die Carboxylgruppe, die eine saure funktionale Gruppe ist, gilt entsprechend ein mittlerer pK_s-Wert von ca. 2,2. Die physiologisch vorkommenden Flüssigkeiten, in denen sich die Aminosäuren im Organismus befinden, variieren in ihrem pH-Wert zwischen 3,5 und 8,0. Dies führt dazu, dass die Amino- und Carboxylgruppen von Aminosäuren im Organismus größtenteils ionisiert sind, also die Form ihrer konjugierten Säure beziehungsweise Base in der Aminosäure annehmen. Damit haben Aminosäuren in physiologischer Umgebung die folgende Struktur:

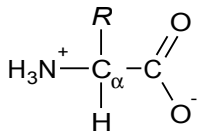


Abbildung 4: *Struktur einer Aminosäure in physiologischer Umgebung*

Die in Abbildung 4 dargestellte Struktur unterscheidet sich von der Grundstruktur (Abbildung 1 auf Seite 7) insofern, als dass die Aminogruppe positiv und die Carboxylgruppe negativ geladen sind. Aufgrund dieser unterschiedlichen Ionisation beider Gruppen werden Aminosäuren auch als *Zwitterionen* oder *Dipolionen* bezeichnet. Diese Eigenschaft hat zur Folge, dass Aminosäuren in physiologischer Umgebung sowohl als Protonendonor wie auch -akzeptor wirken können und somit sowohl sauer wie auch basisch sein können. Stoffe mit dieser Eigenschaft werden als *Ampholyte* (zusammengesetzt aus dem Griechischen Wort αμφίς, gelesen *amphis*, das „beidseitig“ oder „auf beiden Seiten“ bedeutet und λύσίς, gelesen *lysis*, also „Auflösung“) bezeichnet.

Aminosäuren lassen sich aufgrund ihres Dipolcharakters besser in polaren Lösungsmitteln lösen, weshalb die meisten zwar wasserlöslich sind, sich jedoch nicht in organischen Lösungsmitteln lösen lassen.

2.1.2. Klassifikation von Aminosäuren

Neben der Amino- und Carboxylgruppe haben auch die Seitenketten Einfluss auf die Eigenschaften von Aminosäuren. Diese können je nach Aufbau einerseits *polar* oder *unpolar*

sein. Polarität bedeutet hierbei, dass innerhalb der Seitenkette eine asymmetrische Verteilung von Elektronen, vor allem eine Verschiebung der an Atombindungen beteiligten Elektronenpaare, existiert. Dies hat zur Folge, dass polare Seitenketten sich wie elektrische Dipole verhalten. Zusätzlich wird bei polaren Seitenketten zwischen geladenen und ungeladenen Seitenketten unterschieden. Ungeladene polare Seitenketten besitzen trotz ihres Dipolcharakters keine Ladung, sie sind elektrisch neutral. Bei geladenen polaren Seitenketten existiert neben dem Dipolcharakter noch ein geladenes Atom innerhalb der Kette, welches eine positive oder eine negative Ladung trägt. Je nach Ladung kann eine Seitenkette somit nach der Definition in Abschnitt 2.1.1 als basisch oder sauer klassifiziert werden. Polare Seitenketten gehen im Allgemeinen mit der Eigenschaft der *Hydrophilie* (griechisch: ὕδωρ, gelesen *hydōr*, „Wasser“ und φιλος, gelesen *phílos*, „liebend“, engl. *hydrophility*) einher, also der Eigenschaft eines Stoffes, Wasser anzuziehen und wasserlöslich zu sein.

Unpolare Seitenketten hingegen besitzen weder Ladung noch Dipolcharakter. Sie neigen dazu *hydrophob* (griechisch: ὕδωρ, gelesen *hydōr*, „Wasser“ und φόβος, gelesen *phóbos*, „Furcht“, engl. *hydrophobic*), also wasserabstoßend zu sein. Hydrophobie führt weiterhin dazu, dass der betreffende Stoff, wie beispielsweise Öl, dazu neigt, sich nicht mit Wasser zu vermischen und somit auch nicht im Wasser gelöst werden kann. Die Hydrophobizität ist eine der treibenden Kräfte in der Entstehung von Proteinstrukturen [4] und der Interaktion zwischen Proteinen und soll in Abschnitt 2.4.1.3 nochmals angesprochen werden.

Auf der folgenden Seite befindet sich eine tabellarische Darstellung der genannten Klassifikation von Aminosäuren. Tabelle 1 enthält die 20 Standardaminosäuren mit Namen, Dreibuchstabencode, Einbuchstabencode, sowie den pK_s-Werten für Amino- und Carboxylgruppe. Die einzelnen Gruppen sind farblich markiert: Aminosäuren mit unpolaren Seitenketten sind grau unterlegt, mit polaren ungeladenen Seitenketten gelb, mit sauren Seitenketten blau und mit basischen Seitenketten rot. Die pK_s-Werte der einzelnen Amino- und Carboxylgruppen sowie die Klassifikation der Seitenketten sind [2] entnommen.

2.1.3. Die Peptidbindung

Zwischen zwei Aminosäuren kann eine Bindung entstehen, die als *Peptidbindung* bezeichnet wird. Vereinfacht dargestellt, entsteht die Bindung zwischen der Aminogruppe der einen und der Carboxylgruppe der anderen Aminosäure, wobei Wasser als Nebenprodukt

		Name	Dreibuchstabencode	Einbuchstabencode	pK_s Carboxylgruppe	pK_s Aminogruppe
unpolar		Alanin	Ala	A	2,35	9,87
		Glycin	Gly	G	2,35	9,78
		Isoleucin	Ile	I	2,32	9,76
		Leucin	Leu	L	2,33	9,74
		Methionin	Met	M	2,13	9,28
		Prolin	Pro	P	1,95	10,64
		Phenylalanin	Phe	F	2,2	9,31
		Tryptophan	Trp	W	2,46	9,41
		Valin	Val	V	2,29	9,74
polar	ungeladen	Asparagin	Asn	N	2,14	8,72
		Cystein	Cys	C	1,92	10,7
		Glutamin	Gln	Q	2,17	9,13
		Serin	Ser	S	2,19	9,21
		Threonin	Thr	T	2,09	9,1
		Tyrosin	Tyr	Y	2,2	9,21
	geladen basisch	Arginin	Arg	R	1,82	8,99
		Histidin	His	H	1,8	9,33
		Lysin	Lys	K	2,16	9,06
	geladen sauer	Asparaginsäure	Asp	D	1,99	9,9
		Glutaminsäure	Glu	E	2,1	9,47

Tabelle 1: *Klassifikation von Aminosäuren entsprechend der Eigenschaften ihrer Seitenketten*

der Reaktion entsteht. Die unten stehende Abbildung zeigt die Reaktionsgleichung des beschriebenen Prozesses.

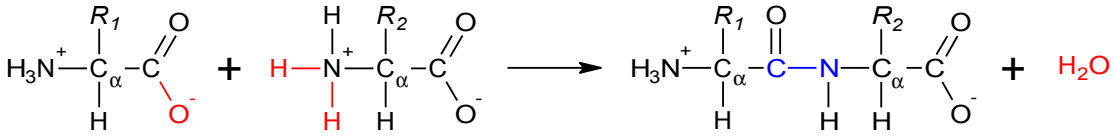


Abbildung 5: *Reaktionsgleichung zur Entstehung der Peptidbindung*

In der Reaktionsgleichung ist das entstehende Wasser, sowohl die zum Wassermolekül reagierenden Atome der Edukte auf der linken Seite, als auch das Produkt auf der rechten Seite, rot markiert. Die entstandene Peptidbindung selbst ist auf der rechten Seite blau gekennzeichnet. Das Produkt der Reaktion wird als *Peptid* bezeichnet und besitzt seinerseits jeweils eine freie Amino- und Carboxylgruppe, an die weitere Aminosäuren mittels Peptidbindung gebunden werden können. Nach diesem Schema können also theoretisch beliebig lange Ketten von Aminosäuren gebildet werden. Dabei ergeben die Peptidbindungen der einzelnen Aminosäuren in einer solchen Kette das sogenannte *Rückgrat* (engl. *backbone*) des Peptids.

Entsprechend ihrer Länge, also der Anzahl an enthaltenen Aminosäuren, oder *Residuen*,

erhalten die Bezeichnungen der entsprechenden Peptide einen griechischen Präfix. So wird das in Abbildung 5 dargestellte Peptid, da es aus zwei Residuen besteht, als *Dipeptid* bezeichnet, während ein Peptid aus drei Residuen *Tripeptid* genannt wird. Längere Peptide werden der Einfachheit halber als *Polypeptide* bezeichnet. Eine fertige Polypeptidkette besitzt die positiv geladene Aminogruppe an einem Ende der Kette und die negativ geladene Carboxylgruppe am anderen Ende. Die Enden (auch *Termini* bezeichnet) werden diesen beiden Gruppen entsprechend als *C-Terminus* (Carboxylgruppe) und *N-Terminus* (Aminogruppe) bezeichnet. Beide Termini sind durch das Rückgrat miteinander verbunden.

Proteine sind Moleküle, die aus einer oder mehreren Polypeptidketten bestehen. Die einzelnen Ketten können dabei zwischen 40 und 4000 Residuen enthalten. Bereits bei der Betrachtung kleinerer Proteine, die aus einer einzelnen Polypeptidkette aus 100 Residuen bestehen, ergeben sich rechnerisch 20^{100} verschiedene Möglichkeiten für Proteine dieser Länge. Diese Zahl entspricht umgerechnet einem Wert von $1{,}27 \cdot 10^{130}$, welcher die geschätzte Anzahl an Atomen im Universum von $9 \cdot 10^{78}$ übersteigt. Dies bedeutet, dass nicht alle möglichen Proteine in der Natur vorkommen oder erzeugt werden können, jedoch ein Spektrum an möglichen Proteinen existiert, welches der Evolution und der Erforschung proteinbasierter künstlicher Wirkstoffe einen großen Spielraum verleiht.

2.2. Primärstruktur von Proteinen

Die Abfolge von Aminosäuren in einer Polypeptidkette wird als *Aminosäuresequenz*, oder kurz *Sequenz*, des Polypeptids bezeichnet. Dabei wird die Reihenfolge der Aminosäuren im Peptid vom N-Terminus zum C-Terminus hin abgelesen. Für kurze Peptide kann die Sequenz mittels des Dreibuchstabencodes angegeben werden. So würde beispielsweise ein Tripeptid bestehend aus drei Alaninen die Sequenz *Ala-Ala-Ala* besitzen, wobei das Alanin ganz links den N-Terminus des Peptids darstellt. Wie unschwer erkennbar ist, ist diese Darstellung für lange Peptide unübersichtlich, weshalb sich zur Darstellung der Sequenz der Einbuchstabencode durchgesetzt hat. Das obige Tripeptid hätte somit die kompaktere Darstellung *AAA*. Die Sequenz aller in einem Protein vorhandenen Peptidketten zusammengenommen wird als dessen *Primärstruktur* (engl. *primary structure*) bezeichnet.

Die erste Bestimmung der Primärstruktur eines Proteins führte der englischen Biochemiker Frederick Sanger im Jahre 1955 [5] durch. Dabei handelte es sich um die vollständige Sequenz des Insulins des Rinds, welches aus zwei Polypeptidketten besteht. Damit gelang

ihm nicht nur die Bestimmung der ersten vollständigen Primärstruktur eines Proteins, sondern auch die Entwicklung einer Methode zur Bestimmung der Primärstruktur von Proteinen, die in Teilen noch heute Anwendung findet. Für diese Leistung erhielt Sanger im Jahre 1958 den Nobelpreis für Chemie.

Bei der von Sanger verwendeten Methode wird das Protein zunächst in die einzelnen Polypeptide, oder *Untereinheiten*, zerlegt, aus denen es besteht. Daraufhin werden diese Untereinheiten voneinander getrennt und wiederum in kleinere Peptidketten zerlegt, die kurz genug sind um deren Sequenz bestimmen zu können. Die Fragmentierung der Untereinheiten in kleinere Peptidketten wird auf mehrere unterschiedliche Weisen durchgeführt und die Ergebnisse der unterschiedlichen Fragmentierungen werden verglichen, um die vollständige Sequenz der einzelnen Untereinheiten zu bestimmen. Die Bestimmung der Sequenz einzelner Untereinheiten kann heutzutage automatisiert durchgeführt werden, wobei der von Pehr Edman 1950 beschriebene Edman-Abbau Verwendung findet [6]. Dabei wird das Residuum am N-Terminus des Peptids abgetrennt und die entsprechende Aminosäure identifiziert. Dieser Schritt kann mit dem Rest des Peptids wiederholt werden, sodass die Sequenz schrittweise vom N-Terminus zum C-Terminus hin bestimmt werden kann. Damit kann für ein unbekanntes Protein die Primärstruktur mit relativ geringem Aufwand bestimmt werden.

Der Vergleich der Primärstrukturen analoger Proteine eines Individuums oder verwandter Spezies kann Einblick in die Funktionsweise von Proteinen und den Verwandtheitsgrad der verglichenen Spezies liefern. So können beispielsweise die Sequenzen daraufhin untersucht werden, ob Residuen an bestimmten Positionen konserviert werden, also in beiden verglichenen Proteinen erhalten bleiben. Ist dies der Fall, spricht das für die Wichtigkeit des betreffenden Residuums für die Funktion des Proteins. Ein Vergleich des Cytochrom C mehrerer Organismen beispielsweise zeigt, dass sich die einzelnen Proteine in 23 von 105 Residuen nicht unterscheiden. Da diese Proteine in allen Organismen die gleiche Aufgabe erfüllen, wird angenommen, dass diese 23 Residuen im Wesentlichen die Funktion des Cytochrom C bestimmen.

Weiterhin deutet eine stärkere Ähnlichkeit der Sequenzen analoger Proteine zweier Spezies darauf hin, dass beide Spezies miteinander verwandt sind und zu irgendeinem Zeitpunkt ihrer Evolution einen gemeinsamen Vorfahren gehabt haben müssen, welcher das ur-

sprüngliche Protein besaß.

Die Funktion eines Proteins im Organismus ist von dessen dreidimensionaler Struktur abhängig. Das Beispiel des Cytochrom C zeigt, dass es bestimmte Residuen in einem Protein zu geben scheint, die für die Funktion und damit auch die Struktur des Proteins ausschlaggebend sind. Diese werden während der Evolution einer Spezies konserviert, da Mutationen, die die Funktion eines Proteins zum Nachteil des jeweiligen Organismus ändern in der Regel automatisch aus dem Genpool entfernt werden. Wie stark der Zusammenhang von Sequenz, Struktur und Funktion ist zeigte Linus Pauling im Jahre 1949 mit seiner Annahme, dass Sichelzellenanämie eine durch Mutation des Hämoglobins hervorgerufene Krankheit ist [7]. Bei der Sichelzellenanämie nehmen die roten Blutkörperchen unter Sauerstoffmangel die Form einer Sichel an, wodurch die Flexibilität der Blutkörperchen gestört wird. Somit kann es zu Durchblutungsstörungen in Kapillarregionen des Blutkreislaufs kommen, was teilweise zu einer Blutunterversorgung bis hin zur Beschädigung von Organen führen kann. Dies resultiert in einer verminderten Lebenserwartung des betroffenen Individuums. Mittlerweile ist bekannt, dass diese strukturelle Verformung durch die Mutation eines einzelnen Residuums hervorgerufen wird. Das Hämoglobin besteht aus vier Untereinheiten, von denen jeweils zwei sich gleichen: zwei so genannte *α-Untereinheiten* bestehend aus 141 Aminosäuren und zwei *β-Untereinheiten* mit je 146 Aminosäuren. Ursache der Sichelzellenanämie ist der Austausch der Glutaminsäure an Position 6 der β-Untereinheiten durch Valin, also eine *Punktmutation* der β-Untereinheit. Dieses Beispiel zeigt also, dass bereits die Veränderung einer einzelnen Aminosäure die Struktur eines Proteins soweit verändern kann, dass es seine Funkion nur bedingt erfüllen kann. Auf der anderen Seite zeigt das Beispiel von Cytochrom C, dass an einem Protein auch ein großer Teil der Aminosäuren verändert werden kann und dass das Protein Struktur und Funktion beibehält, solange bestimmte Residuen unangetastet bleiben. Dies legt den Schluss nahe, dass die räumliche Struktur eines Proteins direkt von seiner Primärstruktur abhängt, jedoch stärker als diese wird konserviert.

2.3. Sekundärstruktur von Proteinen

Die nächste Hierarchieebene auf dem Weg von der Primärstruktur eines Proteins hin zu seiner dreidimensionalen Struktur stellt die *Sekundärstruktur* (engl. *secondary structure*) dar. Diese wird beschrieben durch die relative räumliche Position der einzelnen Aminosäu-

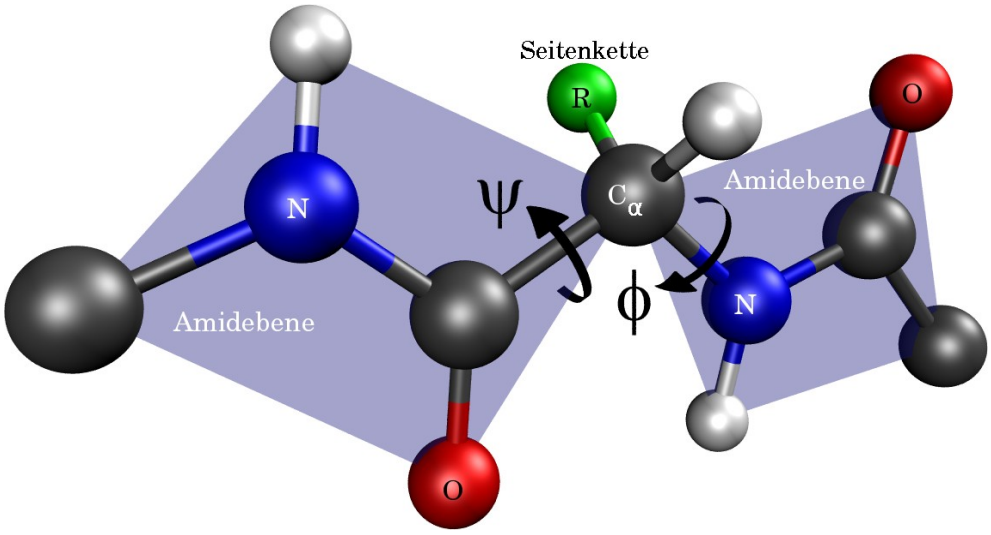


Abbildung 6: *Die Definition der Torsionswinkel ϕ und ψ*

ren entlang des Proteinrückgrats zueinander. Es zeigt sich, dass sich in der räumlichen Anordnung benachbarter Residuen regelmäßige Muster ergeben, die sich mittels der so genannten *Torsionswinkel* und dem Vorhandensein von *Wasserstoffbrückenbindungen* beschreiben lassen.

2.3.1. Die Torsionswinkel

Abbildung 6 zeigt einen Ausschnitt aus dem Rückgrat eines Proteins als Kugel-Stab-Modell, in dem Atome als Kugeln und Bindungen als Zylinder dargestellt sind. Die einzelnen Atomarten unterscheiden sich in ihrer Färbung. So sind Kohlenstoffe, wie Abbildung 6 entnommen werden kann, dunkelgrau dargestellt, Stickstoffe blau, Wasserstoffe weiß und Sauerstoffe rot. Die Seitenkette, welche an das C_α-Atom gebunden ist, wird stellvertretend durch eine grüne Kugel repräsentiert. Es zeigt sich, dass bei natürlich vorkommenden Proteinen die Atome C_α-O-H- C_α eine Ebene, die so genannte *Amidebene* (hellblaues Viereck) bilden. Wie Abbildung 6 entnommen werden kann, grenzen somit an einem C_α-Atom zwei Amidebenen aneinander. Die Orientierung dieser Ebenen zueinander, die den räumlichen Verlauf des Rückgrats maßgeblich beeinflusst, kann sich je nach Verdrehung der C_α-C- beziehungsweise der C_α-N-Bindung ändern und somit über diese beschrieben werden. Die Verdrehung der beiden Bindungen wird über die Torsionswinkel (engl. *torsion angles*) ϕ und ψ beschrieben, wobei ϕ die Verdrehung der C_α-N-Bindung und ψ die der C_α-C-Bindung definiert. Beide Winkel sind in der Abbildung als runde Pfeile dargestellt, wobei die Richtung des Pfeils die Richtung der Zunahme des jeweiligen Winkels darstellt. Sind

beide Winkel 180°, so liegen beide Amidebenen in einer Ebene.

Die Werte, welche die Torsionswinkel ϕ und ψ annehmen können, sind nicht unabhängig voneinander. Logischerweise dürfen weder die Atome der Seitenketten entlang des Proteinrückgrats noch die des Rückgrats selbst sich zu nahe kommen oder gar den gleichen Raum einnehmen. Mit der Frage, welche Paarungen von Torsionswinkeln zugelassen sind, beschäftigten sich der indische Wissenschaftler Gopalasamudram N. Ramachandran und seine Mitarbeiter in den frühen 1960er Jahren [8]. Das Resultat ihrer Forschungen sind die so genannten *Ramachandran-Diagramme*, die die erlaubten Paarungen von Torsionswinkeln als Flächen in einem zweidimensionalen Koordinatensystem darstellen. Die Achsen dieses Koordinatensystems repräsentieren die beiden Torsionswinkel, jeweils im Intervall $[-\pi, \pi[$. Unerlaubte Bereiche für Torsionswinkel sind definiert als diejenigen Werte der Torsionswinkel, für welche der Abstand zweier Residuen kleiner wird als der Van-der-Waals-Abstand, also wenn beide Residuen überlappen würden. Somit hängt die Ausprägung der unerlaubten Bereiche nicht nur von den Torsionswinkeln selbst, sondern auch von den beiden Residuen ab für die das Diagramm erstellt wird: Ihre räumliche Größe bestimmt, wann ein Überlappen stattfindet. Die erlaubten Bereiche zeichnen sich als isolierte Inseln unterschiedlicher Größen innerhalb der unerlaubten Bereiche ab und können den einzelnen Sekundärstrukturtypen (siehe Abschnitt 2.3.4) zugeordnet werden.

2.3.2. Wasserstoffbrücken

Neben den Torsionswinkeln spielen, wie bereits erwähnt, Wasserstoffbrückenbindungen (engl. *hydrogen bonds*) eine wichtige Rolle für die unterschiedlichen Ausprägungen von Sekundärstrukturen. Bei Wasserstoffbrücken interagieren zwei funktionelle Gruppen über ein Wasserstoffatom. Dabei ist das Wasserstoffatom in einer der beiden Gruppen über eine Atombindung an ein Atom mit einer hohen *Elektronegativität* (Donor) gebunden. Stark elektronegative Bindungspartner neigen dazu, das bindende Elektronenpaar stärker zu sich zu ziehen, wodurch beide Bindungspartner eine Teilladung erhalten. Das Wasserstoffatom erhält in diesem Fall eine positive Teilladung, während sein elektronegativer Bindungspartner eine negative Teilladung erhält. Die positive Teilladung des Wasserstoffs ermöglicht eine elektrostatische Interaktion des Wasserstoffs mit einem beliebigen anderen elektronegativen Atom (Akzeptor), das freie Elektronenpaare besitzt. Diese elektrostatische Interaktion macht die Wasserstoffbrückenbindung aus. Abbildung 7 zeigt schematisch eine

Wasserstoffbindung zwischen einem Donor (-X-H) und einem Akzeptor (Y-) als punktierte Linie zwischen beiden Gruppen.

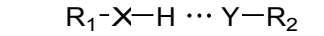


Abbildung 7: *Eine Wasserstoffbrücke zwischen einem Donor (-X-H) und einem Akzeptor (Y-)*

Wasserstoffbrückenbindungen sind in der Regel schwächer als Atom- oder Ionenbindungen, bilden jedoch die strukturelle Basis von Proteinstrukturen und sind für die Stabilität verschiedener Sekundärstrukturtypen essentiell (siehe Abschnitt 2.3.4).

2.3.3. Bestimmung der Sekundärstruktur

Bevor auf die einzelnen Typen von Sekundärstrukturen eingegangen wird, soll zunächst die Methodik zu deren Bestimmung beschrieben werden. Um die Sekundärstruktur eines Proteins bestimmen zu können, muss zunächst seine Primärstruktur bekannt sein. Wie bereits in Abschnitt 2.2 beschrieben, existieren automatisierte Verfahren mittels derer die Primärstruktur unter geringem Aufwand bestimmt werden kann. Die Primärstruktur ist für den nächsten Schritt in der Bestimmung der Sekundärstruktur vonnöten: Der Bestimmung der räumlichen Koordinaten der Atome innerhalb des Proteins. Hierfür gibt es im wesentlichen zwei Verfahren, die sich durchgesetzt haben: einerseits die *Röntgenkristallographie*, andererseits die *Kernspinresonanzspektroskopie* (kurz. NMR-Spektroskopie, aus dem Englischen *nuclear magnetic resonance spectroscopy*). Eine detaillierte Beschreibung beider Verfahren übersteigt den Rahmen dieser Arbeit, sodass an dieser Stelle nur kurz auf sie eingegangen werden soll, um einen Überblick zu verschaffen.

2.3.3.1. Röntgenkristallographie

Die Röntgenkristallographie nutzt die Tatsache, dass elektromagnetische Strahlung, in diesem Fall Röntgenstrahlung, an Kristallgittern gebeugt wird. Licht besitzt minimale Wellenlängen von 4000 Å, wodurch es sich nach den Gesetzen der Optik zum Betrachten solch kleiner Gebilde wie Molekülen oder gar Atomen nicht eignet. Röntgenstrahlen jedoch besitzen Wellenlängen im Bereich von ca. 1,5 Å, was in der Größenordnung von Bindungslängen liegt. Jedoch ist es mit Röntgenstrahlen nicht möglich einzelne Moleküle mittels eines Röntgenmikroskops zu betrachten, da ein solches mangels von Röntgenlinsen nicht existiert. Aus diesem Grund wird der Umweg über die Beugungsmuster monochro-

matischer Röntgenstrahlen an einem Kristallgitter genommen. Die einfallenden Röntgenstrahlen werden am Kristallgitter eines Proteinkristalls gebeugt und die Verteilung der Beugungsmaxima wird gemessen. So ergibt sich ein zweidimensionales Muster von Beugungsmaxima, das mittels zumeist computergestützter mathematischer Verfahren in eine räumliche Information umgerechnet werden kann. Da Röntgenstrahlen fast ausschließlich mit der Elektronenhülle der einzelnen Atome und nicht mit den Atomkernen selbst interagieren, stellt diese Information eine räumliche Elektronendichte dar. Mit der Kenntnis der Primärstruktur des untersuchten Moleküls, und damit auch der Kenntnis über Auftreten und Gruppierung von Atomen, kann aus der Elektronendichte wiederum auf die Position der einzelnen Atome relativ zueinander geschlossen werden [2].

Für die Durchführung der Methode ist es jedoch notwendig, dass das Protein, für welches eine kristallographische Messung durchgeführt werden soll, als Kristall vorliegt. Der Kristallisationsprozess kann je nach Protein unterschiedlich lange dauern. So sind Zeitdauern von einigen Tagen bis hin zu einigen Monaten möglich. Wichtig ist, dass der Kristall möglichst rein und möglichst regelmäßig ist, um das bestmögliche Ergebnis bei der Messung zu erzielen. Proteinkristalle besitzen einen Wassergehalt von 40%-60%, und werden zumeist in Lösung gezüchtet [2]. Dabei wird das Protein zunächst gelöst und daraufhin seine Löslichkeit langsam vermindert, bis sich eine kleine Kristallstruktur (in der Regel ungefähr 100 Moleküle) bildet. Diese wird dann im weiteren Verlauf vergrößert, bis sich ein Kristall der gewünschten Größe ergibt. Der Kristall selbst hat die Konsistenz eines Gels, was eine der Schwierigkeiten der Kristallographie von Proteinen darstellt: Aufgrund ihrer Konsistenz besitzen Proteinkristalle geringfügige Abweichungen in der Regelmäßigkeit ihres Kristallgitters. Diese führen zu Halos um die Braggreflexe und damit zu einer Ungenauigkeit der Intensität der Beugungsmuster, die wiederum zu Ungenauigkeiten in der Berechnung der Elektronendichte führen. Als Konsequenz können strukturelle Details nicht erkannt werden [2]. Aus diesem Grund ist die Kenntnis der Primärstruktur bei der Bestimmung der Atompositionen notwendig. Die Elektronendichten können trotz der erwähnten Ungenauigkeiten verwendet werden, um den Verlauf des Rückgrats und die Position und Orientierung der Seitenketten zu bestimmen. Jedoch ist es zuweilen nicht möglich, Seitenketten vergleichbarer Größe und Form, wie beispielsweise bei Leucin, Isoleucin, Threonin und Valin, zu unterscheiden. Weiterhin sind Wasserstoffatome, die lediglich ein Elektron in ihrer Hülle aufweisen, in kristallographischen Aufnahmen nicht sicht-

bar, was eine sichere Bestimmung der Atomkoordinaten behindern kann [2].

Die Verwendung eines Kristalls zur Bestimmung der Atompositionen innerhalb eines Proteins lässt die Frage aufkommen, inwiefern kristallographische Messungen die realen Atompositionen, also die reale biologische Struktur eines Proteins in Lösung unter physiologischen Bedingungen, abbilden können. Durch die Anordnung der einzelnen Moleküle in einem Gitter können an Stellen, an denen diese sich nahe kommen, Interaktionen zwischen den betroffenen Atomen entstehen. Diese können beispielsweise zu Verschiebungen dieser Atome und einer Verfälschung der Struktur führen. Es zeigt sich jedoch, dass kristallisierte Proteine näherungsweise die gleiche Struktur haben, wie biologisch vorkommende (*native*) Proteine. Hierfür existieren mehrere Hinweise [2]:

- **Wassergehalt des Kristalls:** Der Wassergehalt eines Proteinkristalls ist mit 40%-60% dem Wassergehalt im Inneren einer Zelle ähnlich. Die Oberfläche der einzelnen Proteine im Kristallgitter ist im Wesentlichen von Flüssigkeit umgeben, bis auf wenige kleine Abschnitte, in denen Kontakt zu benachbarten Proteinmolekülen besteht.
- **Übereinstimmung von Kristallographie und NMR-Spektroskopie:** In Fällen, in denen für ein Protein sowohl Strukturen aus der Kristallographie wie auch aus der NMR-Spektroskopie (Abschnitt 2.3.3.2) vorliegen, sind beide Strukturen innerhalb des experimentellen Fehlers fast identisch. Da die NMR-Spektroskopie mit wässrigen Lösungen von Proteinen arbeitet, ist dies ein Indiz dafür, dass Proteine im Kristall wie auch in wässriger Lösung, die eher dem physiologischen Medium entspricht, die gleiche Struktur besitzen.
- **Funktionalität kristallisierter Enzyme:** Die katalytische Aktivität von Enzymen ist sehr stark von der relativen Orientierung der einzelnen Seitenketten abhängig, die an Katalyse und Bindungen zu anderen Stoffen beteiligt sind. Viele Enzyme entfalten jedoch auch im Kristall ihre katalytische Wirkung, was dafür spricht, dass die angenommene Struktur der in physiologischer Lösung entspricht.

Aufgrund der genannten Beobachtungen wird angenommen, dass die aus der Kristallographie gewonnenen Daten, trotz der unnatürlichen Bedingungen, die ein Proteinkristall darstellt, die biologische Realität im Großen und Ganzen widerspiegeln. Somit kann die Verwendung dieser Daten zur Bestimmung der Struktur, vor allem der Sekundärstruktur, von

Proteinen als gerechtfertigt angesehen werden.

2.3.3.2. NMR-Spektroskopie

NMR-Spektroskopie ermöglicht es, die Struktur von Proteinen in wässriger Lösung direkt zu bestimmen. Dabei wird die Tatsache ausgenutzt, dass bestimmte Atomkerne eine intrinsische Magnetisierung, die *Spin* genannt wird, besitzen. Der Spin entspricht einem magnetischen Moment, welches innerhalb eines von außen angelegten Magnetfeldes eine von zwei Orientierungen, oder Zuständen, annehmen kann. Diese werden als *α-Zustand* und *β-Zustand* bezeichnet [9]. Die Energiedifferenz beider Zustände ist in diesem Fall proportional zur Feldstärke des angelegten Magnetfeldes. Der α-Zustand ist energetisch etwas günstiger und dadurch geringfügig stärker besetzt als der β-Zustand. Elektromagnetische Strahlung, deren Frequenz der Energiedifferenz der beiden Zustände entspricht, kann Kerne vom α-Zustand in den β-Zustand überführen. Dieser Effekt wird in diesem Zusammenhang als *Resonanz* bezeichnet. Durch Variation des angelegten Magnetfeldes bei konstanter Frequenz der elektromagnetischen Strahlung, beziehungsweise im umgekehrten Fall durch Variation ebendieser Frequenz bei konstanter Magnetfeldstärke, kann ein Resonanzspektrum für ein Molekül erzeugt werden.

Diese Tatsache wird ausgenutzt, um die chemische Umgebung von Wasserstoffkernen im Molekül zu untersuchen. Elektronenbewegungen um einen magnetischen Atomkern erzeugen schwache magnetische Felder, die dem äußeren Magnetfeld entgegenwirken. Die Ausprägung dieser lokalen Magnetfelder ist abhängig von der lokalen Elektronendichte um die Kerne. Somit zeigen Wasserstoffatome in unterschiedlichen Umgebungen unterschiedliche Frequenzen der elektromagnetischen Strahlung oder unterschiedliche Magnetfeldstärken, bei denen Resonanz auftritt. Dies verleiht ihnen lokal unterschiedliche Absorptionseigenschaften bezüglich der elektromagnetischen Strahlung, welche gemessen werden können. Dieses Verfahren, auch als *eindimensionale NMR* bezeichnet, erlaubt es bereits die Abstände von Protonen im Proteinmolekül zu bestimmen. Als Erweiterung des Verfahrens berücksichtigt die *zweidimensionale NMR* neben den Absorptionseigenschaften der einzelnen Protonen auch deren Interaktion untereinander. Grundlage hierfür ist der *Kern-Overhauser-Effekt*, benannt nach Albert Overhauser, der ihn im Jahre 1953 beschrieb ([10], [11]). Der Effekt stellte eine Interaktion zwischen verschiedenen Atomkernen dar, die mit der sechsten Potenz ihres Abstands abnimmt. Dieser Effekt wurde von F. A .L.

Anet und A. J. R. Bourn 1965 in den Kontext der NMR-Spektroskopie gestellt [12]: Mit Hilfe der Tatsache, dass die Magnetisierung eines Atomkerns sich auf seine Nachbarn überträgt, wenn ihr Abstand unter 5 Å liegt, können die relativen Positionen von Wasserstoffkernen innerhalb eines Proteinmoleküls bestimmt werden.

Die NMR-Spektroskopie liefert also für ein Protein einen Satz von Protonenabständen, mit deren Hilfe die Atomkoordinaten des Proteins berechnet werden können. Hierzu müssen jedoch weitere, bekannte Größen wie Primärstruktur, Van-der-Waals-Radien, Bindungsabstände und Bindungswinkel mit einbezogen werden. Diese Berechnungen sind in der Regel nicht vollkommen eindeutig, sodass die NMR-Spektroskopie einen Satz von Strukturen liefert, die kleine Unterschiede zueinander aufweisen.

Bei der NMR-Spektroskopie werden die Messungen mit einer hochgradig reinen und konzentrierten Lösung des zu untersuchenden Proteins durchgeführt, was eher dessen natürlichem Zustand entspricht. Unterschiede zwischen kristallographischen Messungen und NMR-Messungen sind ein Resultat der in Abschnitt 2.3.3.1 beschriebenen Randeffekte am Kristallgitter. Der Nachteil der NMR-Spektroskopie liegt jedoch in ihrer Mehrdeutigkeit, weshalb sich das Verwenden von NMR-Strukturen als Referenz für Strukturanalysen nur bedingt eignet.

2.3.3.3. Die Bestimmung der Sekundärstruktur aus den Atomkoordinaten

Gegeben die Koordinaten der einzelnen Atome, kann nun für jedes Residuum im Protein bestimmt werden, zu welchem Sekundärstrukturtyp es gehört. Dies kann manuell oder aber auch automatisiert durchgeführt werden. Im ersten Fall entscheidet ein Kristallograph anhand einer geeigneten Darstellung der Atomkoordinaten, welches Residuum welcher Sekundärstruktur zuzuordnen sei. Eine solche Darstellung kann beispielsweise ein virtuelles Kugel-Stab-Modell sein, welches ein Computer aus den Atomkoordinaten generieren kann und welches in Echtzeit rotiert und vergrößert werden kann. Ein Beispiel eines solchen Kugel-Stab-Modells wurde bereits in Abbildung 6 auf Seite 17 präsentiert. Zusätzlich kann ein Computer mittels eines Kugel-Stab-Modells auch den Verlauf des Proteinrückgrats im dreidimensionalen Raum visualisieren, was die Bestimmung der Sekundärstruktur eines Proteins unter Umständen vereinfachen kann. Ein Beispiel beider Darstellungen eines Neurotoxin aus dem Gift der Königskobra (Protein-Data-Bank-Eintrag 3HH7) zeigt Abbildung 8. Dargestellt ist die als Kette A beschriebene Untereinheit des Proteins als Kugel-Stab-

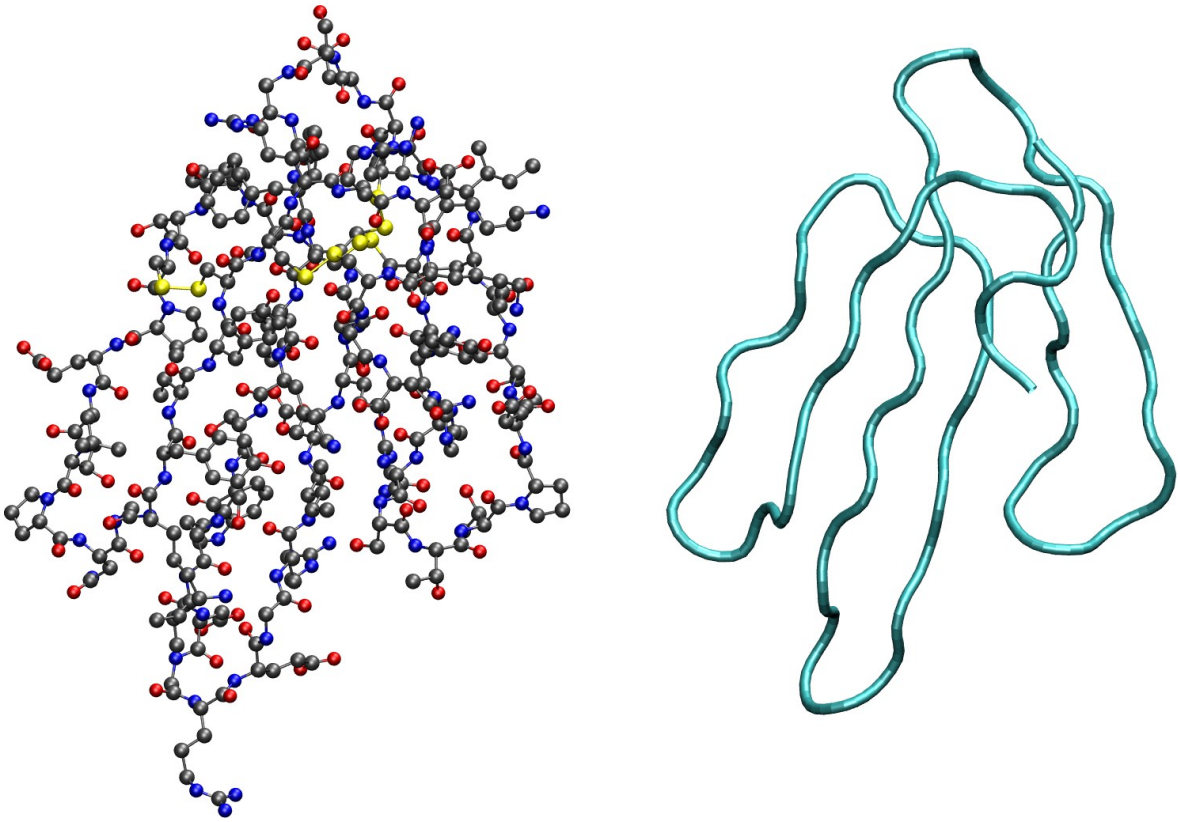


Abbildung 8: *Kette A des Neurotoxins der Königskobra (PDB: 3HH7) als Kugel-Stab-Modell (links) und der Verlauf des Proteinrückgrats (rechts)*

Modell auf der linken Seite und der Verlauf des Proteinrückgrats als Band auf der rechten. Da das Modell auf kristallographischen Daten beruht, sind Wasserstoffatome in ihm nicht enthalten.

Die manuelle Methode hat jedoch den Nachteil, dass es je nach Größe des Proteins mitunter relativ zeitintensiv sein kann, die Sekundärstruktur eines Proteins auf diese Weise zu bestimmen. Zudem hängt die resultierende Zuweisung auch von den Fähigkeiten des Kristallographen ab, sodass die Ergebnisse unterschiedlich ausfallen können. Aus diesem Grund wurden bereits recht früh automatisierte Verfahren zur Bestimmung der Sekundärstruktur aus Atomkoordinaten entwickelt.

Wie bereits in Abschnitt 2.3.2 erwähnt wurde, stellen Wasserstoffbrücken die Basis von der Proteinstruktur dar. Im Jahr 1983 veröffentlichten Wolfgang Kabsch und Christian Sander das Programm DSSP [13], welches aus den Mustern von Wasserstoffbrückenbindungen im Proteinmolekül dessen Sekundärstruktur berechnet. Die Existenz einer Wasserstoffbrücke wird über eine elektrostatische Energiefunktion bestimmt, die auf Abstände von Atomen, die sich in potentiellen Donor- und Akzeptorgruppen befinden, beruht. Dabei

werden den Atomen die Teilladungen $q_1 = 0{,}42\,e$ und $q_2 = 0{,}2\,e$, wobei e der Elementarladung $1{,}602176487 \cdot 10^{-19}\,\mathrm{C}$ entspricht, zugewiesen. Im Modell erhalten Kohlenstoff und Wasserstoff eine positive Teilladung ($+q_1$ und $+q_2$), während Sauerstoff und Stickstoff mit negativen Teilladungen ($-q_1$ und $-q_2$) behaftet sind. Die verwendete Energiefunktion hat die Form:

$$E = q_1 q_2 \left(\frac{1}{r_{ON}} + \frac{1}{r_{CH}} - \frac{1}{r_{OH}} - \frac{1}{r_{CN}} \right) \cdot 332 \frac{\mathrm{kcal}}{\mathrm{mol}}$$

Dabei beschreibt r_{AB} den Abstand zwischen Atom *A* und Atom *B* in Ångström. Eine Wasserstoffbrücke zwischen C=O eines Residuums und N-H eines anderen Residuums wird erkannt, wenn die Energiefunktion unter Einsetzen der entsprechenden Atomabstände einen Wert ergibt, der kleiner als $-0{,}5\,\mathrm{kcal/mol}$ ist. Sind alle Wasserstoffbrücken dem Modell entsprechend erkannt, so kann die Zuweisung der einzelnen Residuen zu einem der verschiedenen Sekundärstrukturtypen anhand von Mustern in der Verteilung der Wasserstoffbrücken durchgeführt werden. DSSP unterscheidet dabei acht unterschiedliche Typen von Sekundärstrukturen, auf welche in Abschnitt 2.3.4 näher eingegangen wird.

Ein weiteres weit verbreitetes Programm zur Bestimmung der Sekundärstruktur aus Atomkoordinaten ist STRIDE, das 1995 von Dmitrij Frishman und Patrick Argos entwickelt wurde [14]. Dieses Programm verwendet wie DSSP auch Wasserstoffbrücken als Kriterium zur Bestimmung der Sekundärstruktur. STRIDE verwendet jedoch ein anderes Verfahren, bei dem eine empirische Energiefunktion zur Beschreibung von Wasserstoffbrücken genutzt wird. Neben Wasserstoffbrücken werden auch die Werte der Torsionswinkel berücksichtigt, die wie bereits in Abschnitt 2.3.1 angedeutet, an der Bildung von Sekundärstrukturen beteiligt sind. Mithilfe beider Größen werden die Residuen eines Proteins den einzelnen Sekundärstrukturtypen zugeordnet, wobei STRIDE zwischen sieben der auch in DSSP verwendeten Sekundärstrukturtypen unterscheidet. Beide Programme haben hohe Übereinstimmungen mit von Kristallographen gegebenen Zuweisungen von Sekundärstrukturtypen, jedoch ist das ältere DSSP weiter verbreitet als STRIDE, weshalb die meisten Angaben von errechneten Sekundärstrukturen auf DSSP beruhen.

2.3.4. Die Sekundärstrukturtypen

Da die Unterscheidung verschiedener Sekundärstrukturtypen auf Regelmäßigkeiten in der

relativen Position benachbarter Residuen basiert, gibt es zunächst einen großen Spielraum für die Definition der einzelnen Sekundärstrukturtypen. Mit der Einführung von DSSP wurde dieser Spielraum auf die acht am häufigsten auftretenden Sekundärstrukturen reduziert, auch wenn bei weitem mehr (wie beispielsweise Ω-Schleifen [2]) denkbar sind. Auf die von DSSP definierten Sekundärstrukturtypen soll in den folgenden Abschnitten eingegangen werden.

2.3.4.1. Die α-Helix

Wie alle helikalen Strukturen zeichnet sich die *α-Helix* durch eine gleichmäßige Verdrehung der Polypeptidkette aus. Die α-Helix ist die stabilste helikale Struktur, da sich die Torsionswinkel der beteiligten Residuen im erlaubten Bereich der Ramachandran-Diagramme befinden und die Struktur durch eine vorteilhafte Verteilung von Wasserstoffbrücken stabilisiert wird. Die Torsionswinkel betragen ungefähr $\phi = -57°$ und $\psi = -47°$ (es handelt sich hierbei um mittlere Werte, von denen es leichte Abweichungen geben kann) und verleihen der α-Helix das Aussehen einer rechtshändig gedrehten Spirale. Dies ist im mittleren Teil von Abbildung 9 auf der folgenden Seite anhand einer künstlich erzeugten α-Helix mit den obigen Torsionswinkeln dargestellt. In Abbildung 9b ist der Verlauf des Proteinrückgrats in Form eines Strangs abgebildet. Dabei sind benachbarte Residuen jeweils abwechselnd durch eine schwarze beziehungsweise weiße Färbung des Strangs gekennzeichnet. Die Atome der α-Helix sind als transparentes Kugel-Stab-Modell mit dem Rückgrat überlagert, wobei die Seitenketten der einzelnen Aminosäuren durch grüne Kugeln als Platzhalter repräsentiert werden. Wasserstoffbrückenbindungen sind durch gestrichelte schwarze Linien zwischen den einzelnen Bindungspartnern dargestellt. Abbildung 9e zeigt eine symbolische Darstellung (*Cartoon-Darstellung*) der α-Helix, wie sie oft in Darstellungen von Proteinen verwendet wird. Es gibt keine einheitliche Färbung für diese Darstellung, jedoch sollen α-Helices innerhalb dieser Arbeit rot dargestellt werden.

Eine Windung der α-Helix enthält 3,6 Residuen und der Abstand zweier Windungen entlang der Achse, um die sich die α-Helix windet, beträgt 5,4 Å. Diese Größe wird auch als *Ganghöhe* bezeichnet und kann Abbildung 9b entnommen werden. Die Seitenketten der einzelnen Aminosäuren innerhalb der α-Helix zeigen von der Achse weg nach außen, wodurch sich Bindungspartner zur Bildung von Wasserstoffbrückenbindungen nahe genug

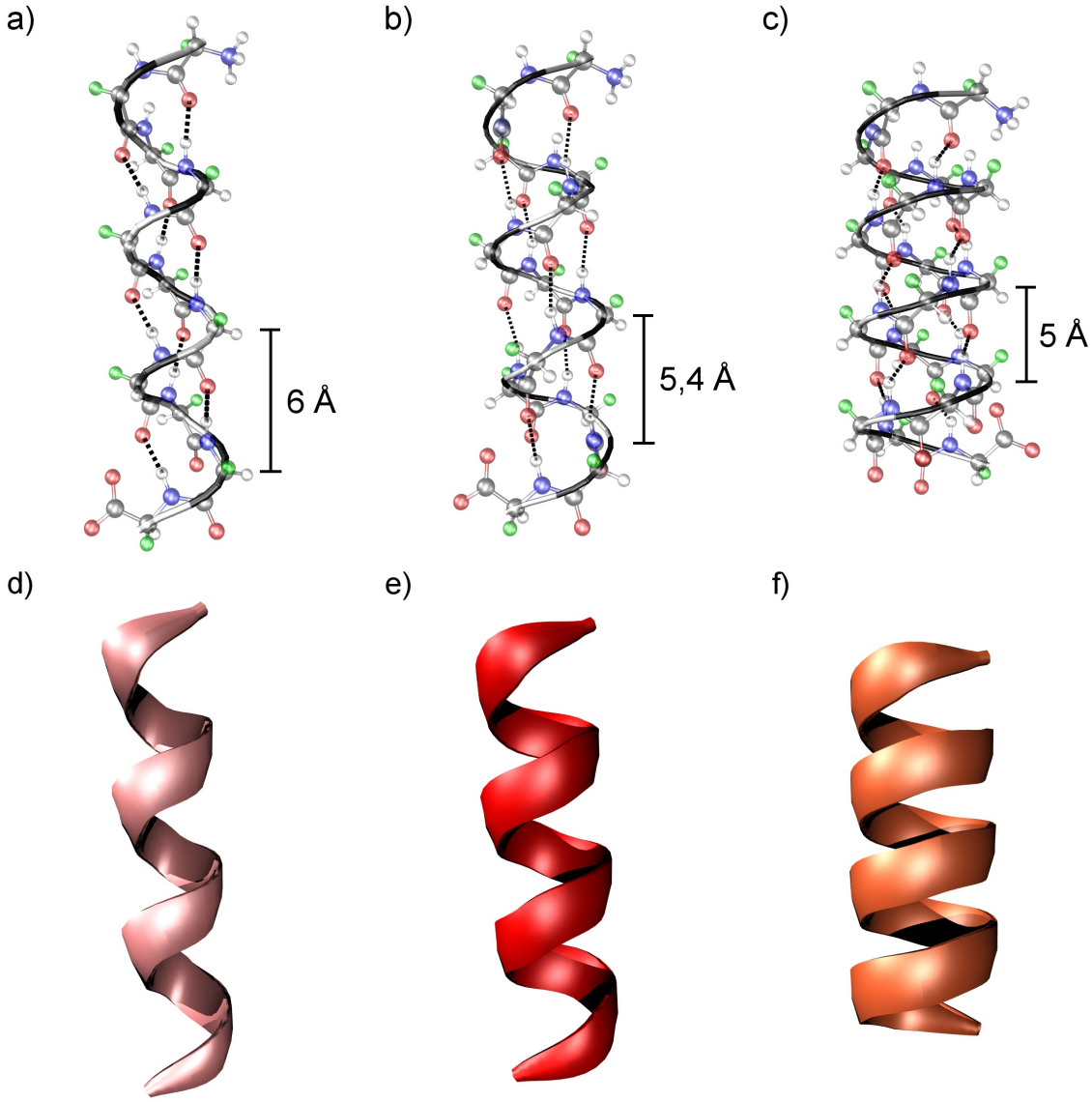


Abbildung 9: *Die drei helikalen Sekundärstrukturtypen als Kugel-Stab-Modell mit Verlauf des Rückgrats (oben) und Cartoon-Darstellung (unten): 3_{10}-Helix (a +d), α-Helix (b + e) und π-Helix (c + f)*

kommen. Dadurch bilden sich starke Wasserstoffbrücken zwischen dem *i*-ten und dem *i*+4-ten Residuum, bei denen der Abstand zwischen Donor und Akzeptor fast den optimalen Wert besitzt. Abbildung 9b zeigt, dass sich dabei ein Muster von Wasserstoffbrücken bildet, das die α-Helix entlang ihrer Achse stabilisiert.

Dank ihrer Stabilität sind α-Helices ein häufig auftretender Sekundärstrukturtyp. Weiterhin verleihen sie der α-Helix in Proteinen eine ähnliche Funktion, die das Skelett bei höheren Organismen erfüllt: Sie stabilisieren die Struktur des Proteins. Die Stabilität einer α-Helix kann durch Prolin negativ beeinflusst werden, da diese Aminosäure die regelmäßige Struk-

tur der α-Helix stört und somit deren Stabilität senkt. Aus diesem Grund befindet sich Prolin in der Regel nicht in α-Helices und wird oftmals als *„Helixbrecher“* bezeichnet.

2.3.4.2. Die 3_{10}-Helix

Die *3_{10}-Helix* ist wie die α-Helix eine rechtshändig gedrehte Spiralstruktur, jedoch ist sie gegenüber der α-Helix kleiner im Durchmesser. So enthält eine Windung drei Aminosäuren und die Ganghöhe beträgt 6 Å. Wie auch bei der α-Helix sind Wasserstoffbrücken für die Stabilisierung der Struktur verantwortlich, werden jedoch im Falle der 3_{10}-Helix zwischen dem *i*-ten und dem *i*+3-ten Residuum gebildet. Die Torsionswinkel der einzelnen Aminosäuren liegen bei $\phi = -49°$ und $\psi = -26°$ und befinden sich teilweise leicht im unerlaubten Bereich der Ramachandran-Diagramme. Dadurch ist die 3_{10}-Helix weniger stabil als die α-Helix und kommt entsprechend seltener vor. Zumeist kommen 3_{10}-Helices in Übergängen zwischen α-Helices und dem Rest der Polypeptidkette vor. Hierbei handelt es sich zumeist um einzelne Windungen, die als Übergänge dienen. Jedoch kommen vereinzelt auch eigenständige 3_{10}-Helices vor.

Der linke Teil von Abbildung 9 zeigt eine künstlich erzeugte 3_{10}-Helix, welche mithilfe der idealen Werte der Torsionswinkel für 3_{10}-Helices erzeugt wurde. Wie auch schon in Abbildung 9b für α-Helices ist in Abbildung 9a der Verlauf des Proteinrückgrats als Strang dargestellt, entlang dessen benachbarte Aminosäuren abwechselnd schwarz und weiß gefärbt sind. Die Atome der 3_{10}-Helix sind abermals als transparentes Kugel-Stab-Modell mit dem Proteinrückgrat überlagert, wobei Seitenketten stellvertretend durch grüne Kugeln dargestellt sind. Wasserstoffbrücken sind durch gestrichelte schwarze Linien zwischen den beteiligten Bindungspartnern dargestellt. In Abbildung 9d befindet sich die Cartoon-Darstellung der 3_{10}-Helix, für die eine rosa Färbung gewählt wurde. Die dargestellte 3_{10}-Helix umfasst 11 Residuen, was für eine real existierende 3_{10}-Helix eine ungewöhnliche Länge darstellt. Die meisten 3_{10}-Helices sind, wie bereits erwähnt, eher kurz.

Der Name der 3_{10}-Helix beinhaltet zwei Zahlen, mit denen sich helikale Strukturen generell beschreiben lassen: Jede Helix kann als A_B-Helix bezeichnet werden, wobei A die Anzahl von Aminosäuren innerhalb einer Windung der Helix beschreibt und B der Anzahl der Atome entspricht, die sich innerhalb des Rings befinden, der von beiden Bindungspartnern einer Wasserstoffbrückenbindung begrenzt wird. Bei der α-Helix wäre die entsprechende Notation $3{,}6_{13}$-Helix, da sie 3,6 Aminosäuren pro Windung besitzt und zwischen Donor

und Akzeptor einer Wasserstoffbrücke 13 Atome liegen.

2.3.4.3. Die π-Helix

Die *π-Helix* stellt den dritten helikalen Sekundärstrukturtyp dar, der auch als $4{,}4_{16}$-Helix bezeichnet werden kann. Die rechtshändig gedrehten Windungen beinhalten 4,4 Residuen, was den Durchmesser der π-Helix gegenüber der α-Helix vergrößert. Die Ganghöhe beträgt 5 Å, wodurch die π-Helix das Aussehen einer leicht gestauchten α-Helix besitzt. Die für helikale Strukturen typischen Wasserstoffbrücken werden bei der π-Helix zwischen dem *i*-ten und dem *i*+5-ten Residuum ausgebildet. Die π-Helix ist, wie auch die 3_{10}-Helix, instabiler als die α-Helix und kommt deshalb auch seltener vor. Grund dafür sind auch hier die Torsionswinkel, die die Werte $\phi = -55\,°$ und $\psi = -70\,°$ annehmen und teilweise im unerlaubten Bereich der Ramachandran-Diagramme liegen. Der rechte Teil von Abbildung 9 zeigt das Beispiel einer künstlich erzeugten π-Helix, wobei in Abbildung 9c abermals der Verlauf des Proteinrückgrats mit transparenten Atomen und Wasserstoffbrücken dargestellt ist. In Abbildung 9f wiederum ist die Cartoon-Darstellung der π-Helix in orange abgebildet.

π-Helices bilden in der Regel Übergänge zwischen einer α-Helix und dem Rest der Polypeptidkette, die wie bei der 3_{10}-Helix aus einer Windung bestehen. Die in Abbildung 9 dargestellte π-Helix besitzt somit mit einer Länge von 17 Residuen eine unnatürliche Länge. Derart lange π-Helices kommen in der Natur nicht vor.

2.3.4.4. Das β-Faltblatt

Das *β-Faltblatt* (englisch: *β-sheet*) stellt die zweite Form regelmäßiger Sekundärstrukturtypen neben den helikalen Typen dar. β-Faltblätter zeichnen sich durch ein periodisches Auftreten von Torsionswinkeln und ein ausgeprägtes Muster von Wasserstoffbrückenbindungen aus. Ein β-Faltblatt besteht aus zwei oder mehr sogenannten *β-Strängen* (engl. *β-strands*), die Abschnitten einer oder mehrerer Polypeptidketten entsprechen, welche an der Bildung des β-Faltblatts beteiligt sind. Jedem β-Strang kann eine Richtung zugewiesen werden, die vom N-terminalen zum C-terminalen Ende des β-Strangs verläuft. Auf dieser Grundlage werden zwei Arten von β-Faltblättern unterschieden: Parallele β-Faltblätter, bei denen die β-Stränge in die gleiche Richtung verlaufen, und antiparallele β-Faltblätter, bei welchen die β-Stränge entgegengesetzt gerichtet sind.

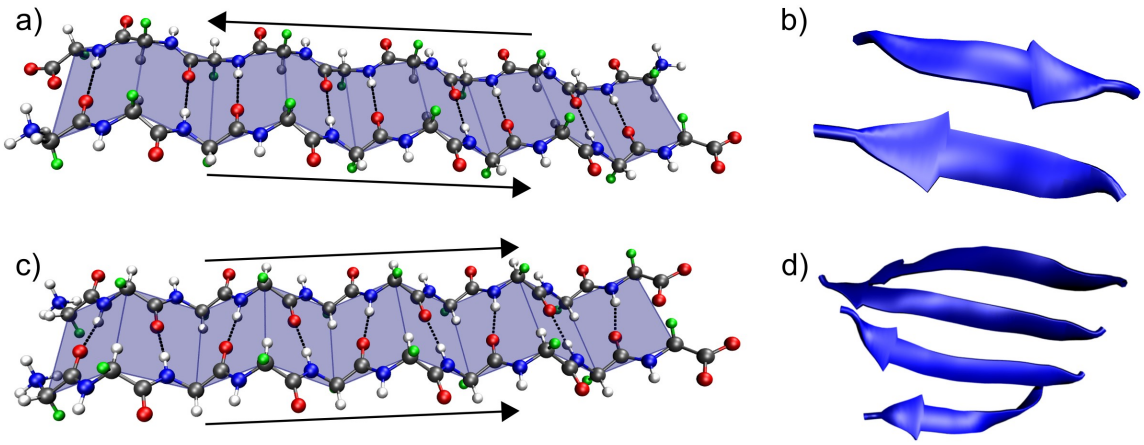


Abbildung 10: *Antiparalleles (oben) und paralleles (unten) β-Faltblatt*

Abbildung 10a zeigt das Kugel-Stab-Modell eines künstlich erzeugten antiparallelen β-Faltblatts besehend aus zwei β-Strängen. Die Torsionswinkel entlang der Stränge entsprechen den idealen Werten von $\phi = -139\,°$ und $\psi = +135\,°$ [15] und die Seitenketten der einzelnen Aminosäuren sind stellvertretend als grüne Kugeln dargestellt. Die Richtungen der beiden Stränge sind jeweils durch einen schwarzen Pfeil gekennzeichnet, der von ihrem N- zu ihrem C-Terminus verläuft. Zwischen den Strängen bilden sich lateral Wasserstoffbrücken aus, welche beim antiparallelen β-Faltblatt in einer Ebene liegen und in der Abbildung durch gestrichelte schwarze Linien gekennzeichnet sind. Dieses Muster von Wasserstoffbrücken ist vorteilhaft und trägt wesentlich zur Stabilisierung des antiparallelen β-Faltblatts bei. Weiterhin kann der Abbildung die Eigenschaft entnommen werden, die den β-Faltblättern ihren Namen verleiht: Zwischen den sich gegenüberliegenden C_α-Atomen beider Stränge (beziehungsweise den benachbarten C_α-Atomen eines Strangs) können gedachte Verbindungslinien gezogen werden, die in der Abbildung als blaue Linien dargestellt sind. Wie zu sehen ist, schließen diese Linien viereckige Flächen im Raum ein, die in der Abbildung hellblau dargestellt sind. Diese Flächen wiederum bilden zusammen eine Struktur, die an ein mehrfach gefaltetes Blatt Papier erinnert. Der Abstand zwischen dem *i*-ten und dem *i+2*-ten C_α-Atom eines Stranges, also den Eckpunkten eines Knicks im Faltblatt, bestehend aus zwei der viereckigen Flächen, beträgt 7 Å.

Parallele β-Faltblätter besitzen eine geringere Stabilität als antiparallele β-Faltblätter, da die Wasserstoffbrücken zwischen den einzelnen Strängen nicht in einer Ebene liegen. Dies ist in Abbildung 10c anhand eines künstlich erzeugten β-Faltblatts dargestellt. In diesem

Fall besitzen die beiden Stränge gegenüber denen des antiparallelen β-Faltblatts abweichende Torsionswinkel, nämlich $\phi = -119\,°$ und $\psi = +113\,°$. Nichtsdestotrotz besitzen auch parallele β-Faltblätter das Aussehen eines mehrfach gefalteten Papierblattes, welches für β-Faltblätter typisch ist.

In dreidimensionalen Darstellungen von Proteinen werden β-Stränge in der Regel symbolisch in Form von Pfeilen in Richtung des C-Terminus dargestellt. Anhand von Gruppierungen von β-Strängen können die einzelnen Formen von β-Faltblättern unterschieden werden. In Abbildung 10b ist das Beispiel eines antiparallelen β-Faltblatts in Cartoon-Darstellung abgebildet. Dabei handelt es sich um einen Ausschnitt aus der Struktur der Maltosephosphorylase des *Lactobacillus brevis* (PDB: 1H54), der lediglich auf die in der Struktur enthaltenen β-Stränge beschränkt ist, die das antiparallele β-Faltblatt bilden. Das Faltblatt besteht, wie aus der Abbildung hervorgeht, aus zwei relativ kurzen β-Strängen bestehend aus jeweils drei Residuen (Residuen 170-172 und 239-241). Wie aus der Nummerierung der Residuen hervorgeht, liegen die beiden Stränge sequentiell nicht in unmittelbarer Nachbarschaft. Dies ist nicht immer der Fall, da oft gerade bei antiparallelen β-Faltblättern die einzelnen Stränge über kurze Stücke andersartiger Struktur miteinander verbunden und somit auch sequentiell nahe beieinander liegen können. In Abbildung 10d ist weiterhin ein Beispiel für ein paralleles β-Faltblatt enthalten. Dieses besteht aus vier parallel verlaufenden β-Strängen, die der Struktur der menschlichen Ferrochelatase (PDB: 1HRK) entnommen wurden. Die Abbildung zeigt auch, dass β-Faltblätter durchaus gewölbt sein können und nicht zwangsläufig eine Ebene bilden müssen. Die blaue Färbung in der Cartoon-Darstellung beider Arten von β-Faltblättern soll innerhalb dieser Arbeit für die Repräsentation von β-Strängen und -Faltblättern verwendet werden.

Wie bereits erwähnt wurde, müssen die einzelnen Stränge, die ein Faltblatt bilden, in der Sequenz des betreffenden Proteins nicht zwangsläufig benachbart sein. Durch den Verlauf der Polypeptidkette im Raum ergeben sich oftmals räumliche Nachbarschaften von sequentiell weit entfernten Residuen, die an der Bildung von β-Faltblättern beteiligt sein können. Tendenziell ist es jedoch bei antiparallelen β-Faltblättern eher wahrscheinlich, dass die beteiligten Stränge auch sequentiell nahe sind, da sie sich untereinander durch relativ kurze Stücke der Polypeptidkette verbinden lassen. Dies ist in Abbildung 11 angedeutet. Die Abbildung zeigt weiterhin, dass bei einem parallelen β-Faltblatt eine sequentielle Nachbarschaft der einzelnen Stränge unwahrscheinlicher wird, da die Polypeptidkette zumindest

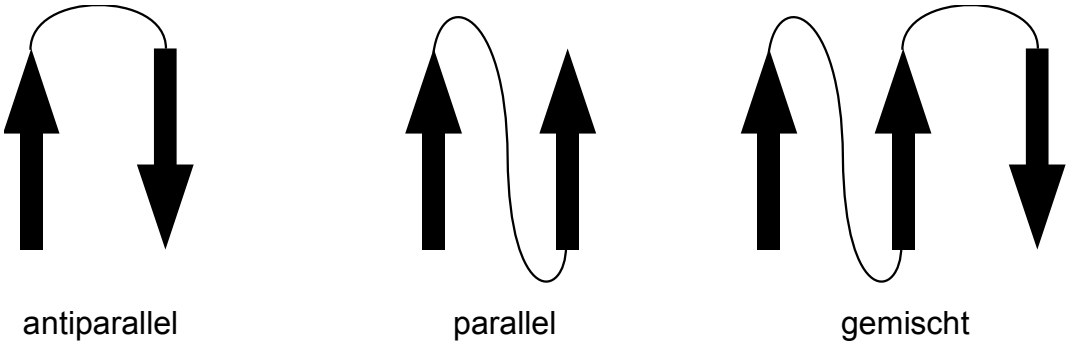


Abbildung 11: *Schematische Darstellung der unterschiedlichen Arten von β-Faltblättern*

die in der Abbildung dargestellte Wende nehmen müsste, um die beiden dargestellten Stränge miteinander zu verbinden. Diese ist im Allgemeinen länger als die direkte Verbindung zweier antiparalleler Stränge und wird in der Regel umso länger, je länger die Stränge werden, die verbunden werden müssen. Somit bilden sich parallele β-Faltblätter zumeist zwischen β-Strängen aus, die entlang der Polypeptidkette weit auseinander liegen.

β-Stränge müssen nicht zwangsläufig ausschließlich parallel oder antiparallelen angeordnet sein. Es kann durchaus passieren, dass ein β-Faltblatt aus drei oder mehr β-Strängen gebildet wird, die teilweise parallel, teilweise aber auch antiparallel angeordnet sind. Ein solches gemischtes β-Faltblatt ist in Abbildung 11 rechts dargestellt.

2.3.4.5. Die isolierte β-Brücke

Eine *isolierte β-Brücke* (engl. *isolated β-bridge*) besteht aus zwei Residuen, welche zwar innerhalb der Sequenz weit auseinanderliegen, jedoch räumlich nahe beieinander sind. Die räumliche Nähe beider Residuen hat zur Folge, dass sich zwischen ihnen eine Wasserstoffbrücke ausbildet, die diesen Sekundärstrukturtyp charakterisiert. Isolierte β-Brücken besitzen in der Regel keine vordefinierten Torsionswinkel und zeichnen sich lediglich durch die Existenz der besagten Wasserstoffbrücke aus. Wie der Name bereits sagt, sind die beiden Residuen, die diese Brücke bilden, isoliert. Das bedeutet, dass sie nicht in irgendeiner Form an einem Muster von Wasserstoffbrücken beteiligt sind, das zur Ausbildung von helikalen Strukturen oder β-Faltblättern beiträgt. Zudem sind beide Residuen zumeist die einzigen in ihrer unmittelbaren sequenziellen Nachbarschaft, die diesem Sekundärstrukturtyp zugeordnet werden können. Mehrere sequentiell aufeinander folgende Residuen, die zu isolierten β-Brücken gehören, sind sehr selten und zumeist sind die Bindungspartner dieser Residuen ihrerseits nicht benachbart.

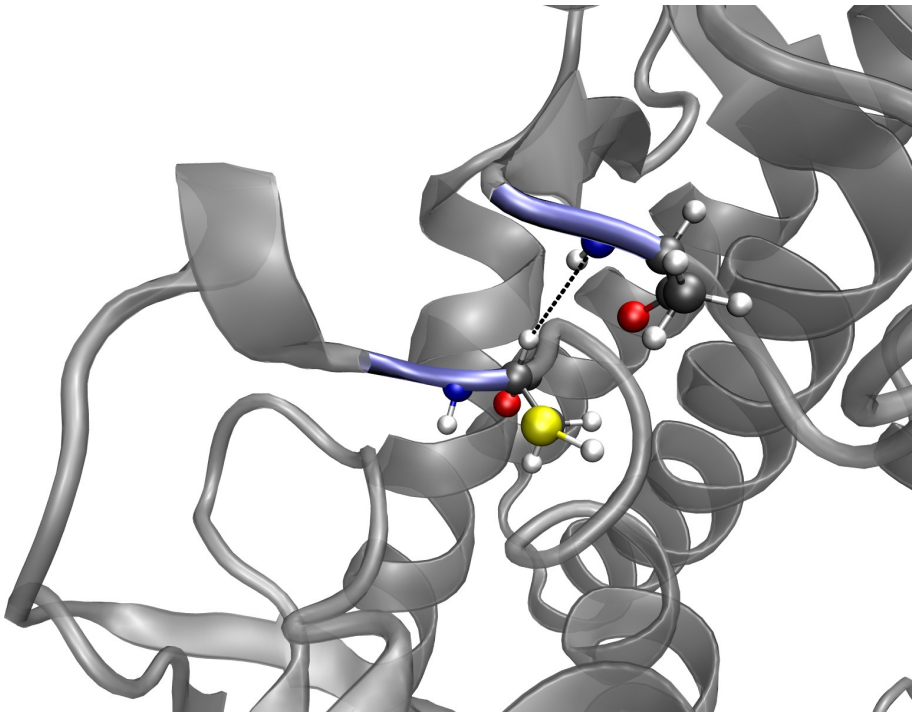


Abbildung 12: *Die isolierte β-Brücke*

In Abbildung 12 ist ein Beispiel einer isolierten β-Brücke dargestellt. Die Abbildung zeigt einen Ausschnitt der symbolischen Darstellung der Chitobiose Phosphorylase des Bakteriums *Vibrion proteolyticus* (PDB: 1V7W). Der Teil der Struktur, der nicht zu der isolierten β-Brücke gehört, ist transparent und grau dargestellt. Die beiden an der isolierten β-Brücke beteiligten Residuen (Residuen 424 und 488) sind in der Cartoon-Darstellung als violett gefärbte Stränge skizziert und mit ihrem Kugel-Stab-Modell überlagert. Weiterhin ist die Wasserstoffbrücke, die zwischen beiden Residuen besteht, als gestrichelte schwarze Linie dargestellt. Wie der Abbildung selbst, als auch der Nummerierung der verwendeten Residuen entnommen werden kann, sind beide Residuen innerhalb der Sequenz relativ weit entfernt. Weiterhin sind es die einzigen beiden Residuen in ihrer sequentiellen Nachbarschaft, die einer isolierten β-Brücke zugeordnet sind. Die hellblaue Färbung beider Residuen soll innerhalb dieser Arbeit verwendet werden, um isolierte β-Brücken farblich von anderen Sekundärstrukturen abzuheben.

DSSP verwendet β-Brücken zur Erkennung von β-Faltblättern. Sequentiell weit entfernte Residuen, die eine Wasserstoffbrücke ausbilden, werden von DSSP zunächst als β-Brücken erkannt. Das Programm weist daraufhin mehrere aufeinander folgende Residuen, die als β-Brücken identifiziert sind, einem potentiellen β-Strang (in [13] als *β-ladder* bezeich-

net), im folgenden β-Leiter genannt, zu. Sofern die Bindungspartner der innerhalb einer solchen β-Leiter enthaltenen Residuen ihrerseits eine β-Leiter bilden, werden die beiden β-Leitern als vollwertige β-Stränge erkannt und zu einem β-Faltblatt zusammengefasst. Kann einer β-Leiter auf diese Weise keine zweite β-Leiter zugeordnet werden oder ist diese Zuordnung unvollständig in dem Sinne, dass nicht allen Residuen einer der beiden β-Leitern ein Residuum in der jeweils anderen zugeordnet werden kann, so werden die jeweiligen Residuen aus den β-Leitern entfernt und als isolierte β-Brücken klassifiziert. Dies kann im Extremfall zu einem vollständigen Zerfall der betreffenden β-Leitern zu einer Folge von isolierten β-Brücken führen. Dies geschieht ebenfalls mit β-Strängen, die zu kurz sind, also weniger als zwei Residuen enthalten. Da DSSP β-Brücken als elementare Sekundärstruktureinheiten zur Erkennung von β-Strängen (und damit auch β-Faltblättern) verwendet, können isolierte β-Brücken nach der Definition von DSSP als fehlgeschlagene, zu kurze β-Stränge betrachtet werden.

2.3.4.6. Die wasserstoffbrückengebundene Windung

Wasserstoffbrückengebundene Windungen (engl. *hydrogen bonded turn*) sind eine Form der Sekundärstruktur, die sich durch eine relative Nähe der beteiligten Residuen und eine Ausprägung von Wasserstoffbrücken kennzeichnet. Im Gegensatz zu isolierten β-Brücken ist in diesem Falle nicht nur der räumliche Abstand der Residuen innerhalb einer Windung gemeint, sondern auch ihr sequentieller Abstand. Windungen entstehen, wenn der räumliche Abstand sequentiell benachbarter Residuen durch eine räumliche Verbiegung des Rückgrats eines Proteins auf unter 7 Å sinkt. Dabei ist der Abstand der C_α-Atome gemeint. Es entsteht eine Art Schlaufe, die durch die Entstehung von Wasserstoffbrücken stabilisiert wird. Man unterscheidet unterschiedliche Arten von Windungen anhand des sequentiellen Abstands der Residuen, zwischen denen Wasserstoffbrücken gebildet werden. So existieren *α-Windungen* (engl. *α-turns* oder *4-turns*), bei denen die Wasserstoffbrücke zwischen dem *i*-ten und dem *i+4*-ten Residuum gebildet wird. Bei *β-Windungen* (engl. *β-turns* oder *3-turns*) liegen zwischen Donor und Akzeptor der Wasserstoffbrückenbindung drei Residuen, während dieser Abstand bei *π-Windungen* (engl. *π-turn* oder *5-turns*) fünf Residuen beträgt.

Die obige Definition der einzelnen Windungstypen ähnelt augenscheinlich der Definition der einzelnen helikalen Sekundärstrukturtypen. Dies ist kein Zufall, denn grob kann eine

Windung, ihrem Namen entsprechend, als eine Windung innerhalb einer Helix des entsprechenden Typs angesehen werden. Diese Eigenschaft macht sich DSSP bei der Erkennung von Helices zunutze. Zunächst werden die einzelnen Windungstypen anhand des sequentiellen Abstands der Residuen, die an einer Wasserstoffbrücke beteiligt sind, bestimmt. Dabei werden Donor und Akzeptor jeweils dem entsprechenden Windungstyp zugewiesen. Ergeben sich auf diese Weise mehrere aufeinander folgende Residuen eines bestimmten Windungstyps, so wird dieser Abschnitt der Sequenz dem entsprechenden Helixtyp zugeordnet. Die einzelnen Helixtypen haben in diesem Fall eine minimale Länge, die genau einer Windung der entsprechenden Helix entspricht. Dies wären bei 3_{10}-Helices drei, bei α-Helices vier und bei π-Helices fünf Residuen. Windungen eines Windungstyps, die die entsprechende Mindestlänge unterschreiten, werden einem generischen Windungstyp zugewiesen, der eine allgemeine wasserstoffbrückengebundene Windung darstellt. So kann es passieren, dass ein längerer Abschnitt der Sequenz eines Proteins zunächst unterschiedlichen Windungstypen zugewiesen wird. Die einzelnen Abschnitte, die einem bestimmten Windungstyp zugewiesen sind, können jedoch zu kurz sein, um den entsprechenden Helixtyp auszubilden, sodass die jeweiligen Residuen der allgemeinen Windung zugewiesen werden. Somit kann es durchaus vorkommen, dass ein langer Abschnitt eines Proteins aus aufeinander folgenden Residuen besteht, die wasserstoffbrückengebundenen Windungen zugewiesen sind. Entgegen der intuitiven Idee, diesen Abschnitt einem helikalen Sekundärstrukturtyp zuordnen zu wollen, ist die Zuweisung eines längeren Abschnitts wasserstoffbrückengebundener Windungen jedoch nach der obigen Argumentation sinnvoll.

Aus der Definition, die in DSSP verwendet wird, ergibt sich eine gewisse konzeptionelle Ähnlichkeit zwischen wasserstoffbrückengebundenen Windungen und isolierten β-Brücken. Beide Sekundärstrukturtypen entspringen elementaren Sekundärstruktureinheiten, die zum Auffinden größerer und komplexerer Sekundärstrukturtypen verwendet werden. Weiterhin stellen beide im gewissen Sinne Fehlschläge in der Ausbildung der entsprechenden Sekundärstrukturtypen dar: Während eine isolierte β-Brücke einen fehlgeschlagenen β-Strang darstellt, kann eine wasserstoffbrückengebundene Windung als eine fehlgeschlagene Helix interpretiert werden. Der Unterschied zwischen den beiden liegt jedoch in den Torsionswinkeln: Während für isolierte β-Brücken keine vordefinierten Torsionswinkel existieren, können für die einzelnen Windungstypen durchaus Torsionswinkel definiert werden. Diese sind jedoch nicht so einheitlich wie im Falle helikaler Strukturen oder

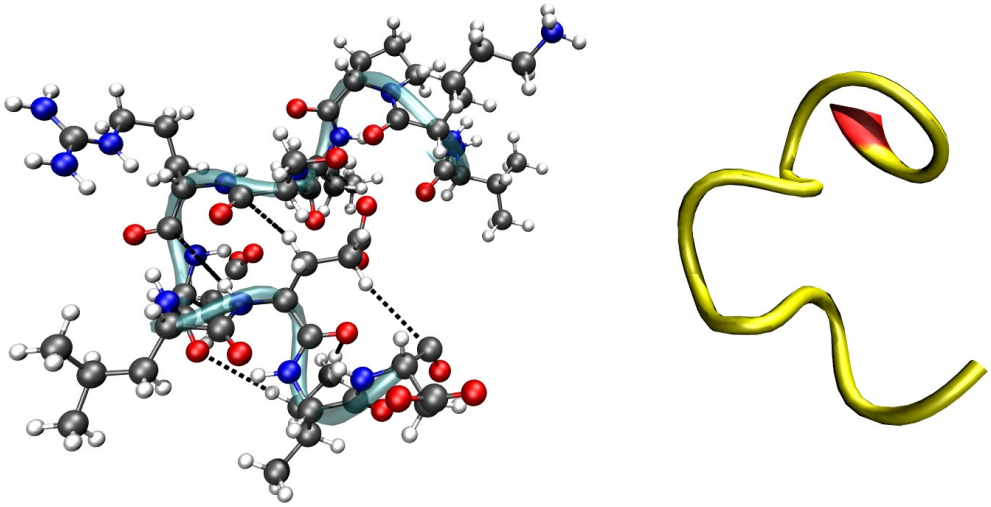


Abbildung 13: *Die wasserstoffbrückengebundene Windung*

β-Stränge. Die einzelnen Windungstypen können ihren Torsionswinkeln entsprechend in mehrere Unterkategorien unterteilt werden. Auf diese Weise lassen sich zum Beispiel im Falle der β-Windung acht unterschiedliche Kategorien unterscheiden.

Zur Veranschaulichung der geometrischen Gegebenheiten ist in Abbildung 13 das Beispiel einer wasserstoffbrückengebundenen Windung dargestellt. Die Abbildung zeigt einen Ausschnitt aus der Struktur des Proteins MTH396 aus dem Bakterium *Methanothermobacter termautotrophicus* (PDB: 1NXH), der einen längeren Abschnitt aus Residuen darstellt, die einer wasserstoffbrückengebundene Windungen zugeordnet sind. Links im Bild ist das Kugel-Stab-Modell des entsprechenden Abschnitts überlagert mit dem Verlauf des Proteinrückrats (dargestellt als blaues, transparentes Band). Wasserstoffbrücken sind, wie in vorherigen Abbildungen auch, als schwarze gestrichelte Linien dargestellt. Die räumliche Nähe im Verlauf des Proteinrückrats und die Ausprägung der Wasserstoffbrücken sind gut sichtbar. Rechts ist die symbolische Darstellung desselben Abschnitts als gelber Strang dargestellt, dessen Färbung in dieser Arbeit zum Hervorheben wasserstoffbrückengebundener Windungen verwendet werden soll. Diese Darstellung verdeutlicht, dass sich eine Ähnlichkeit zu helikalen Sekundärstrukturtypen ergibt, betrachtet man den räumlichen Verlauf des Proteinrückrats. Jedoch scheint es, dass die Richtung, in die sich das Proteinrückrat windet, entlang der Sequenz mehrfach wechselt, sodass keine regelmäßige Struktur entsteht. Weiterhin ist an einem Ende der dargestellten Struktur der Übergang in einen helikalen Sekundärstrukturtyp (rote Verbreiterung des Strangs) angedeutet.

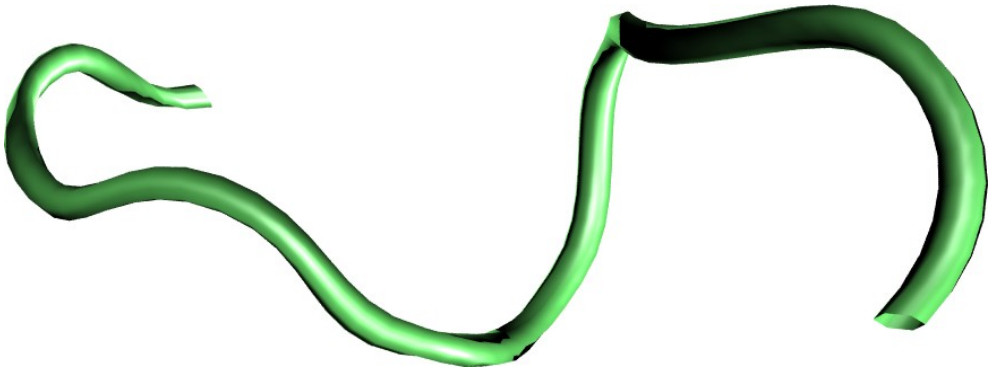


Abbildung 14: *Die Biegung*

2.3.4.7. Die Biegung

Im Gegensatz zu den bisher vorgestellten Sekundärstrukturtypen sind *Biegungen* (engl. *bend*) ausschließlich über geometrische Kriterien definiert, sodass die Existenz von Wasserstoffbrücken für diesen Sekundärstrukturtyp keine Rolle spielt. Tatsächlich besitzen Biegungen eine gewisse Ähnlichkeit zu wasserstoffbrückengebundenen Windungen insofern, dass sie eine räumliche Verbiegung des Proteinrückrats darstellen, die eine enge Kurve im Raum beschreibt und sequentiell benachbarte Residuen auch räumlich näher bringt. Zur Erkennung von Biegungen verwendet DSSP jedoch ausschließlich Verbindungsvektoren zwischen den C_α-Atomen der Polypeptidkette. Ein Residuum *i* wird unter Verwendung dieser Vektoren als Biegung erkannt, wenn der Winkel zwischen dem Verbindungsvektor der C_α-Atome von Residuum *i* und *i+2* und dem Verbindungsvektor von Residuum *i* und *i-2* mehr als 70° beträgt. Diese Definition zur Erkennung von stark gebogenen Abschnitten des Proteinrückgrats geht auf Rose und Seltzer [16] zurück, wobei Kabsch und Sander in ihrem Artikel [13] anmerken, dass der Schwellenwert für den Winkel von ihnen anhand visueller Analysen existierender Proteinstrukturen festgelegt wurde. Die Biegung ist der einzige Sekundärstrukturtyp, der von DSSP ohne Betrachtung von Wasserstoffbrückenmustern erkannt wird.

Abbildung 14 zeigt das Beispiel einer Biegung, welches der Struktur eines menschlichen Proteins (PDB: 1DDF) entnommen wurde. Die Abbildung skizziert einen Ausschnitt der Struktur (Residuen 317-325) in Cartoon-Darstellung, der den stark gebogenen Charakter von Biegungen zeigt. Die in der Abbildung gewählte Darstellung als olivgrüner Strang soll innerhalb dieser Arbeit zur Repräsentation von Biegungen verwendet werden.

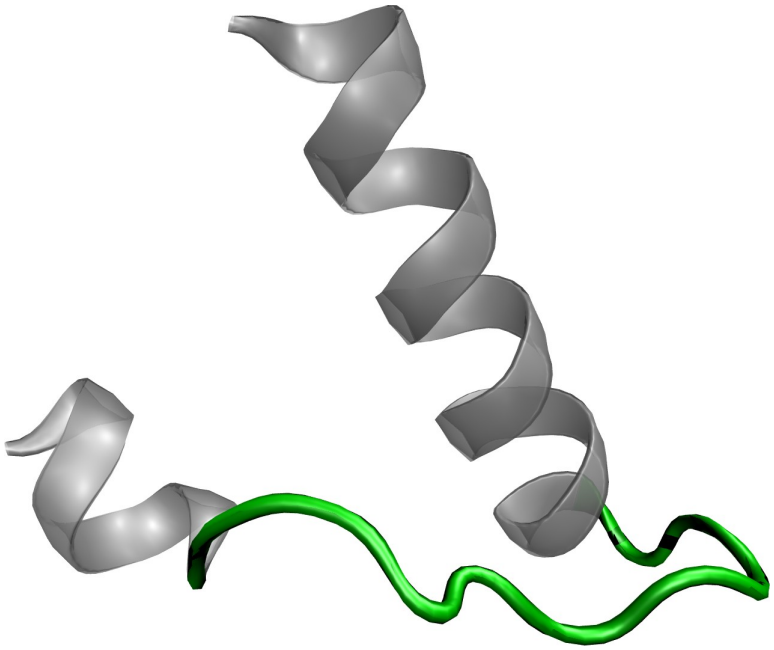


Abbildung 15: *Eine strukturlose Region als Verbindung zweier Helices*

2.3.4.8. Strukturlose Regionen

Der letzte Sekundärstrukturtyp, der behandelt werden soll, ist keine Sekundärstruktur im eigentlichen Sinne. Vielmehr handelt es sich, wie der Name bereits sagt, um Regionen, die keine regelmäßige Struktur besitzen, also laut der von DSSP verwendeten Kriterien keinem der Sekundärstrukturtypen zugewiesen werden können, die in den Abschnitten 2.3.4.1 bis 2.3.4.7 beschrieben wurden. Diese *strukturlosen Regionen*, im Englischen *random coils* genannt, stellen Verbindungsabschnitte zwischen den einzelnen strukturierten Regionen eines Proteins dar. So zeigt Abbildung 15 eine strukturlose Region, welche zwei Helices miteinander verbindet. Die beiden Helices sind hier aufgrund ihrer untergeordneten Rolle in transparentem Grau dargestellt, während die im Vordergrund stehende strukturlose Region als ein grüner, zwischen den Helices verlaufender Strang hervorgehoben ist. Der abgebildete Strukturausschnitt entstammt dem Hämoglobins des Pantoffeltierchens (*Paramecium caudatum*, PDB: 1DLW) und ist in symbolischer Darstellung gezeigt. Innerhalb der strukturlosen Region lassen sich weder Muster von Wasserstoffbrücken erkennen, noch greift das geometrische Kriterium, das Biegungen in der Struktur definiert. Strukturlose Regionen, wie auch Biegungen, Windungen und β-Brücken, können zusammengenommen als Verbindungsabschnitte zwischen den regelmäßigen Sekundärstrukturtypen, die sich aus den helikalen Strukturen und β-Strängen/-Faltblättern zusammensetzten, betrachtet wer-

den. Die Darstellung strukturloser Regionen als grünes Band soll innerhalb dieser Arbeit beibehalten werden.

2.4. Tertiärstruktur von Proteinen

Die einzelnen beschriebenen Sekundärstrukturtypen bilden, wie in Abschnitt 2.3 beschrieben, lediglich lokale Verformungen des Proteinrückgrats im dreidimensionalen Raum. Sie basieren einzig auf den relativen Positionen benachbarter Atome zueinander. Die eigentliche dreidimensionale Struktur ist jedoch eine globale Eigenschaft des Proteins und ergibt sich aus dem Verlauf der gesamten Polypeptidkette im dreidimensionalen Raum. Diese globale dreidimensionale Struktur eines Proteins wird als dessen *Tertiärstruktur* bezeichnet und ist im hierarchischen Aufbau der Proteinstruktur der Sekundärstruktur übergeordnet.

Die Ausbildung der Tertiärstruktur wird von der Verteilung von Sekundärstrukturtypen über die Polypeptidkette, deren Ausprägung und Wechselwirkung untereinander beeinflusst. Das Vorhandensein eines bestimmten Sekundärstrukturtyps an einer bestimmten Stelle der Sequenz erzeugt eine lokale Verformung der Polypeptidkette und ändert somit ihren Verlauf im dreidimensionalen Raum. Die Seitenketten der einzelnen Sekundärstrukturen können untereinander Wasserstoffbrücken bilden oder aber anziehende oder abstoßende Wechselwirkungen aufweisen, was den weiteren Verlauf des Proteinrückrats durch den Raum beeinflusst. Die Tertiärstruktur eines Proteins, so wie auch deren Entstehung, wird auch als die *Faltung* des Proteins bezeichnet. Die Faltung der meisten Proteine ähnelt dem Knäuel eines Fadens oder einer mehr oder weniger ausgeprägten Kugel und weist eine kompakte Struktur auf, bei der sich die Polypeptidkette im Raum mehrere Male selbst passieren kann. Proteine mit einer solchen Faltung werden als *globuläre Proteine* oder auch *Sphäroproteine* bezeichnet. Bei diesen Proteinen lässt sich zumeist feststellen, dass die Seitenketten unpolarer Aminosäuren sich im Inneren einer solchen Sphäre befinden, während geladene polare Seitenketten an ihrer Oberfläche zu finden sind. Dies führt dazu, dass globuläre Proteine wasserlöslich sind. Die Gruppe der *fibrillären Proteine* bildet hingegen Strukturen, die langen Fasern oder Fäden entsprechen. Somit befinden sich sowohl polare wie auch unpolare Aminosäuren an der Oberfläche des Proteins, weshalb die meisten dieser Proteine nicht wasserlöslich sind. Fibrilläre Proteine, wie beispielsweise das Keratin, sind zumeist Strukturproteine die Stütz- oder Gerüstfunktionen besitzen.

In Abbildung 16 ist die Tertiärstruktur zweier globulärer Proteine dargestellt. Das Protein

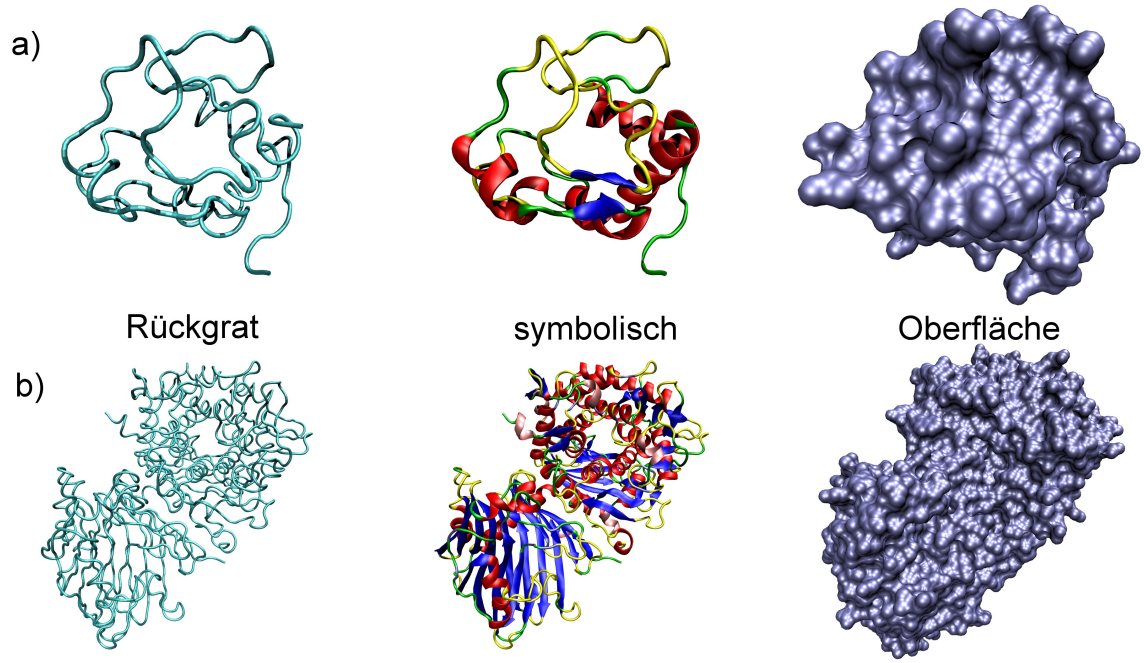


Abbildung 16: *Die Tertiärstruktur zweier globulärer Proteine*

in Abbildung 16a stellt das Cytochrom C der Hefe dar (PDB: 1YCC), das für den Elektronentransport bei der Energiegewinnung in den Mitochondrien zuständig ist. Dieses Protein wurde aufgrund seiner geringen Größe und Komplexität als Beispiel für die Tertiärstruktur eines kleinen Proteins ausgewählt. Der Verlauf des aus 108 Aminosäuren gebildeten Rückgrats der Polypeptidkette ist als hellblauer Strang im dreidimensionalen Raum im linken Bild dargestellt. Das mittlere Bild zeigt abermals den räumlichen Verlauf der Polypeptidkette, diesmal jedoch mit Hilfe der Cartoon-Darstellung der Sekundärstruktur. Hierbei sind die einzelnen Sekundärstrukturtypen auch farblich voneinander abgehoben. Auf diese Weise lässt sich der Beitrag der einzelnen Sekundärstrukturtypen zur Tertiärstruktur erkennen. Schließlich zeigt das rechte Bild die Oberfläche des Proteins, die aus der Tertiärstruktur resultiert. Diese Oberfläche geht in ihrer Definition auf Lee und Richards [17] zurück, die sie 1971 als die einem Lösungsmittel zugängliche Fläche (engl. *solvent accessible surface area*) beschrieben. Sie ergibt sich aus dem minimalen Abstand, den ein Wassermolekül zu den Atomen der einzelnen Aminosäuren haben kann. Zur Berechnung der Oberfläche wird ein Algorithmus verwendet, der auf Shrake und Rupley [18] zurückgeht und von idealisierten Wassermolekülen in Form von Kugeln mit einem Radius von 1,4 Å ausgeht. Die Oberfläche ergibt sich so als die Abrollfläche dieser Kugel über das Protein, die aufgrund des Radius um die einzelnen Atome, von dem Wassermolekül nicht

durchdrungen werden kann. Bei dem Protein in Abbildung 16b handelt es sich um die Chitobiose Phosphorylase von *Vibrio proteolyticus*, die bereits in Abschnitt 2.3.4.5 ausschnittsweise als Beispiel für isolierte β-Brücken erwähnt wurde (Abbildung 12 auf Seite 33). Dieses Protein zeigt mit seinen 807 Residuen die Komplexität der Tertiärstruktur größerer Proteine. Abermals sind von links nach rechts der Verlauf des Proteinrückgrats, die Cartoon-Darstellung der Sekundärstruktur und die Oberfläche des Proteins dargestellt. In diesem Fall liegt die Assoziation des Verlaufs des Proteinrückgrats mit einem Fadenknäuel näher. Beide Proteine zeigen anhand ihrer Oberfläche, dass diese durch eine verzerrte Sphäre approximiert werden könnte.

2.4.1. Stabilität von Proteinstrukturen

Die Komplexität von Tertiärstrukturen wirft die Frage auf, was für die Stabilität dieser komplexen Strukturen verantwortlich ist. Proteine sind sehr instabile Gebilde, die nur unter bestimmten Außenbedingungen ihre Struktur beibehalten [2]. Die Struktur eines Proteinmoleküls ist das Ergebnis eines empfindlichen Kräftegleichgewichts zwischen unterschiedlichen, gegeneinander wirkenden Kräften. Bereits eine geringfügige Störung dieses Gleichgewichts kann zu starken Veränderungen der Struktur führen. Im Folgenden sollen die einzelnen wechselwirkenden Kräfte und deren Beitrag zur Stabilität der Proteinstruktur angesprochen werden.

2.4.1.1. Elektrostatische Kräfte

Proteinmoleküle können ungeladene wie geladene polare Aminosäuren (Abschnitt 2.1.2) enthalten, die aufgrund des Dipolmoments oder der Ladung ihrer Seitenketten in elektrostatischer Wechselwirkung zueinander stehen können. Paare entgegengesetzt geladener Seitenketten bilden Ionenbindungen aus, die so genannten *Salzbrücken*. In der Regel befinden sich Salzbrücken an der Oberfläche von Proteinen und sind deshalb in wässriger Lösung gelöst. Zudem zeigt der Vergleich von Proteinen ähnlicher Struktur, dass Salzbrücken nicht in all diesen Proteinen zur Erhaltung der Struktur notwendig sind. Somit haben Salzbrücken eher geringen Einfluss auf die Stabilität der Proteinstruktur. Dipolmomente andererseits, wie sie von ungeladenen polaren Seitenketten erzeugt werden, haben eine starke stabilisierende Wirkung. Die Wechselwirkung zwischen den einzelnen Dipolen in einem Protein, seien diese induziert oder permanent, ist schwach. Jedoch führt die relative Nähe und die hohe Dichte von Dipolen innerhalb eines Moleküls zu einer Vielzahl von

Dipol-Dipol-Interaktionen. Je nach Ausrichtung der Dipole zueinander können sich diese Wechselwirkungen beispielsweise durch Addition der einzelnen Dipolmomente merklich verstärken. Aus diesem Grund haben Dipolmomente einen starken Einfluss auf die Stabilität von Proteinstrukturen.

2.4.1.2. Wasserstoffbrücken

In den Abschnitten 2.3.4.1, 2.3.4.2, 2.3.4.3, 2.3.4.4 und 2.3.4.6 wurde die Wichtigkeit von Wasserstoffbrücken für die Bildung von Sekundärstrukturen angesprochen. Die regelmäßigen Muster von Wasserstoffbrücken in Helices und β-Faltblättern tragen wesentlich zum Entstehen dieser Strukturen bei. Es zeigt sich jedoch, dass es in einem Protein keine idealen Wasserstoffbrücken zwischen einem Donor und einem Akzeptor gibt. Vielmehr ist jede Wasserstoffbrücke Teil eines Netzwerks, in dem jeder Donor Wasserstoffbrückenbindungen mit mehreren Akzeptoren eingeht und jeder Akzeptor Wasserstoffbrücken zu mehreren Donoren ausbildet. Effektiv ordnen sich die einzelnen Gruppen, die Wasserstoffbrücken ausbilden können, innerhalb eines Proteins so zueinander an, dass fast alle Wasserstoffbrücken, die möglich sind, gebildet werden. Dies ist ein weiteres Indiz für den Einfluss von Wasserstoffbrücken auf die Proteinstruktur.

Nicht gefaltete Proteine bilden Wasserstoffbrücken zu den sie umgebenden Wassermolekülen. Energetische Vergleiche der Wasserstoffbrücken des gefalteten und des nicht gefalteten Proteins zeigen, dass Wasserstoffbrücken trotz ihres starken Einflusses bei der Bildung von Proteinstrukturen nicht signifikant zur Stabilisierung dieser Strukturen beitragen müssen und diese im Gegenteil sogar destabilisieren können [2]. Nichtsdestotrotz stellen Wasserstoffbrücken das Gerüst dar, um das die Struktur von Proteinen gebaut wird. Wird ein Protein auf eine Weise gefaltet, sodass nicht alle Wasserstoffbrücken der nativen Struktur ausgebildet werden können, so ist die so entstandene Struktur des Proteins weniger stabil.

2.4.1.3. Hydrophobe Kräfte

Hydrophobe Kräfte bewirken, dass unpolare Gruppen (siehe Abschnitt 2.1.2) den Kontakt mit Wasser minimieren. Dies führt beispielsweise bei Seifen zur Bildung von *Mizellen*, also kugelförmigen Anordnungen von Molekülen, bei denen die Kugeloberfläche von wasserlöslichen (polaren) Teilen der Moleküle gebildet wird, während sich die hydrophoben

(unpolaren) Teile im Kugelinneren befinden. Globuläre Proteine besitzen im wässrigen physiologischen Medium einen ähnlichen Aufbau, bei dem polare Gruppen an der Oberfläche zu finden sind und die hydrophoben unpolaren Gruppen sich, von dieser Oberfläche umschlossen, im Inneren des Proteins befinden. Somit stellen Proteine in gewissem Sinne Mizellen dar. Diese Tatsache spricht sehr für den Einfluss des hydrophoben Effekts auf die Bildung von Proteinstrukturen. Es kann angenommen werden, dass eine Polypeptidkette sich im Wasser immer so falten wird, dass die unpolaren Aminosäuren möglichst schnell durch die polaren Aminosäuren vom Wasser isoliert werden. Weiterhin sorgt die Hydrophobie dafür, dass die isolierten unpolaren Aminosäuren ihre Isolierung aufrechterhalten und somit die einmal entstandene Struktur stabilisiert wird. Aufgrund dessen kann der hydrophobe Effekt als eine der treibenden Kräfte der Proteinfaltung und als wichtiger Faktor in der Stabilisierung von Proteinstrukturen angesehen werden [4].

2.4.1.4. Disulfidbrücken

Disulfidbrücken sind eine besondere Form von Bindungen innerhalb von Proteinen, die zwischen den Schwefelatomen der Seitenketten zweier Cysteine gebildet werden. Die Reaktion, die zur Entstehung dieser Bindungen führt, kann vereinfacht durch die folgende Reaktionsgleichung beschrieben werden:

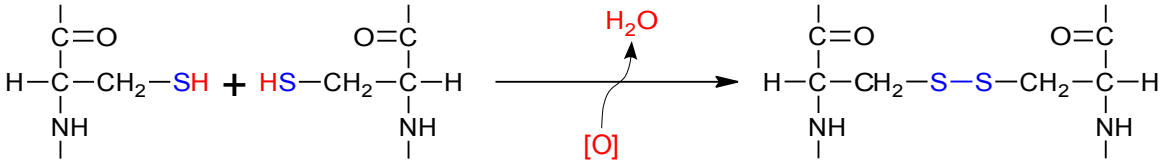


Abbildung 17: *Reaktionsgleichung zur Entstehung einer Disulfidbindung*

Die Reaktion läuft unter Zufuhr von Sauerstoff ab. Die an die Schwefelatome der Cysteine gebundenen Wasserstoffatome werden von diesen gelöst und bilden zusammen mit dem Sauerstoff Wasser. Dies ermöglicht es den Schwefelatomen ihrerseits die Bindung einzugehen, die die Disulfidbrücke ausmacht. In Abbildung 17 sind Edukte und Produkte der Bildung des Wassers rot hervorgehoben, während die Schwefelatome und die Disulfidbrücke, die sie bilden, blau gekennzeichnet sind.

Disulfidbrücken werden bei der Faltung von Proteinen gebildet und stabilisieren ihre Struktur. Die Stärke dieser Bindungen, wie auch ihr stabilisierender Einfluss, ist im intrazellulären Medium schwächer als im extrazellulären Medium. Letzteres ist gegenüber Zell-

flüssigkeit weniger kontrolliert, was pH-Werte und Temperaturen angeht, und kann somit Proteinen gegenüber als feindseliger angesehen werden. Aus diesem Grund müssen Proteine, welche im extrazellulären Medium zum Einsatz kommen, robuster sein, um in dieser Umgebung eine stabile Struktur aufrecht erhalten zu können. Bei vielen dieser Proteine wird diese Robustheit durch das bilden von Disulfidbindungen erreicht.

2.4.2. Denaturierung

Das empfindliche Gleichgewicht der Kräfte, das die Stabilität von Proteinen bestimmt, kann sehr schnell gestört werden. Eine solche Störung kann dazu führen, dass sich das entsprechende Protein strukturell verändert und eine andere Sekundärstruktur beziehungsweise Tertiärstruktur annimmt. Eine solche Änderung der Struktur hat wiederum zur Folge, dass das betroffene Protein die von der Natur vorgesehene Funktion nicht mehr erfüllen kann. Dieser Vorgang wird deshalb als *Denaturierung* bezeichnet, weil er das Protein aus seinem natürlichen Zustand in einen unnatürlichen Zustand überführt. Dabei kann diese Zustandsänderung sowohl reversibel als auch irreversibel sein.

Es existiert eine Vielzahl von Faktoren, die Denaturierung begünstigen. Einer davon ist die Temperatur. Proteine weisen eine starke Änderung ihrer Eigenschaften über einen kleinen Temperaturbereich auf. Der Grund hierfür ist, dass die Erwärmung eines Proteins ab einem bestimmten Punkt zu einer teilweisen Entfaltung von Bereichen des Proteins führen kann, welche ihrerseits eine weitere Destabilisierung des gefalteten Bereichs erzeugt. Dies führt dazu, dass sich weitere gefaltete Bereiche entfalten, was eine starke Veränderung der gesamten Struktur zur Folge hat. Man spricht in diesem Fall von einer kooperativen Entfaltung. Ein einfaches Beispiel für eine temperaturbedingte Denaturierung stellt das Erhitzen eines Hühnereis dar: Das Eiweiß des erhitzten Eis verändert bei hohen Temperaturen seine Struktur und geht von der klaren Flüssigkeit, die es normalerweise darstellt, in eine feste weiße Masse über, weil die ursprünglich isoliert vorliegenden globulären Proteinmoleküle sich entfalten und untereinander ein räumliches Geflecht ausbilden. Dies ist weiterhin ein Beispiel für eine irreversible Denaturierung, da das Eiweiß sich nach der Erhitzung nicht mehr in seine ursprüngliche Form überführen lässt.

Auch der pH-Wert des Mediums in dem sich das Protein befindet kann zur Denaturierung führen. Der pH-Wert kann die Ionisation von Seitenketten und somit auch die Ladungsverteilung im Protein verändern und so den Einfluss von elektrostatischen Kräften und Was-

serstoffbrücken verändern, was zu einer Veränderung der Struktur führen kann. Detergenzien und hohe Konzentrationen organischer Substanzen wiederum binden sich an hydrophobe Gruppen und können auf diese Weise die Auswirkung der die Struktur der Proteinmoleküle stabilisierenden hydrophoben Kräfte beeinflussen. Auch Salze können eine Denaturierung hervorrufen, da sie gelöst in Form von Ionen auftreten, die eine Auswirkung auf die elektrostatischen Kräfte haben können, die Proteinstrukturen stabilisieren. Zudem können Salze Proteine gegenüber Veränderungen der Temperatur empfindlicher machen, sodass ein Protein bei Zuführung von Salzen einer geringeren Erhitzung bedarf, um zu denaturieren.

2.5. Quartärstruktur von Proteinen

Die letzte Ebene, die in der hierarchischen Unterteilung von Proteinstrukturen existiert, ist die *Quartärstruktur*. In der Natur kommen Proteine vor, die aus mehreren Polypeptidketten bestehen. Jede dieser Polypeptidketten besitzt eine Tertiärstruktur, die ein Volumen im Raum einnimmt und eine Untereinheit des Proteins bildet. Die räumliche Anordnung der einzelnen Untereinheiten zu einem größeren Proteinkomplex wird als Quartärstruktur bezeichnet. Proteine, die aus nur einer Polypeptidkette bestehen, besitzen im allgemeinen keine Quartärstruktur, da es keine der Tertiärstruktur übergeordnete Strukturebene gibt.

Die Schnittstellen, an denen die einzelnen Untereinheiten eines Proteinkomplexes aneinander gebunden sind, müssen so aufgebaut sein, dass sie exakt zusammenpassen, um durch Bindungen eine Haftung der Untereinheiten aneinander zu ermöglichen. Der Aufbau dieser Schnittstellen, was polare und unpolare, geladene und ungeladene Seitenketten angeht, muss so strukturiert sein, dass die entsprechenden Schnittstellen sich in irgendeiner Weise miteinander verbinden können. Dadurch hat Denaturierung, obwohl sie zunächst Einfluss auf Tertiär- und Sekundärstruktur hat, auch Einfluss auf die Bildung von Quartärstrukturen: Eine Änderung von auch nur einer der Schnittstellen führt dazu, dass der Proteinkomplex nicht mehr aufgebaut werden kann. Viele Proteinkomplexe können ihre Funktion jedoch erst aufgrund ihrer Zusammensetzung aus Untereinheiten ausüben, wobei jede Untereinheit eine Teilaufgabe in der Funktion des Proteinkomplexes übernimmt. Ist dieses Zusammenspiel der einzelnen Untereinheiten nicht gewährleistet, so kann der Proteinkomplex seine Aufgabe nicht mehr zufrieden stellend ausführen.

2.6. Zusammenhang zwischen Sequenz und Struktur

In den vorherigen Abschnitten wurde bereits erwähnt, dass die Funktion eines Proteins von dessen Struktur abhängig ist. Aus diesem Grund ist die Untersuchung von Proteinstrukturen und deren Bildung wichtig für das Verständnis der Funktionsweise von Proteinen. Im Jahr 1973 stellte Christian B. Anfinsen die Behauptung auf, dass die Struktur eines Proteins eineindeutig von seiner Sequenz bestimmt wird [19]. Als Ausgangspunkt für seine Hypothese diente Anfinsen die Tatsache, dass Polypeptidketten gleicher Sequenz unter gleichen Bedingungen (Druck, Temperatur, Lösungsmittel) die gleiche Struktur bilden, was dafür spricht, dass die Struktur von Proteinen in ihrer Sequenz verschlüsselt ist. Anfinsen nahm an, dass daraus im Umkehrschluss folgen musste, dass Proteine gleicher Struktur auch die gleiche Sequenz haben müssten. Zum Zeitpunkt ihrer Entstehung wurde diese Hypothese von den relativ wenigen Proteinstrukturen, die zur damaligen Zeit bekannt waren, gestützt. In der Zwischenzeit hat die Menge bekannter Proteinstrukturen beträchtlich zugenommen und die gesammelten Daten stellen den von Anfinsens durchgeführten Umkehrschluss zunächst in Frage. Tatsächlich zeigt das bereits erwähnte Beispiel des Cytochrom C, dass Proteine gleicher Struktur unterschiedliche Sequenzen besitzen können. Heutzutage ist bekannt, dass eine Anzahl von möglichen Faltungstypen existiert, von denen jeder von einer Vielzahl durchaus sehr unterschiedlicher Sequenzen angenommen wird, was einem eineindeutigen Sequenzcode, der zu einer bestimmten Proteinstruktur gehört, widerspricht [20]. Die Unterschiede zwischen Proteinen, die die gleiche Faltung annehmen, können sogar soweit gehen, dass zwei Proteine gleicher Struktur lediglich in 1-2% ihrer Sequenzen übereinstimmen [21]. All diese Tatsachen erschweren es, eine Korrelation zwischen der Sequenz eines Proteins und seiner Struktur zu finden. Wünschenswert wäre die Aufstellung einer Funktion oder die Konzipierung eines Computerprogramms, um die Faltung einer gegebenen, unbekannten Proteinsequenz vorausberechnen zu können. Die Existenz eines solchen Modells würde eine effiziente Bestimmung neuer Proteinstrukturen ermöglichen, ohne dabei sofort auf aufwändige Methoden wie Röntgenkristallographie (Abschnitt 2.3.3.1) oder NMR-Spektroskopie (Abschnitt 2.3.3.2) zurückgreifen zu müssen. Dies wiederum würde die Untersuchung unbekannter Proteinsequenzen, vor allem in der Entwicklung neuer Wirkstoffe für Arzneimittel sowie künstlicher Enzyme, erheblich erleichtern. In Abbildung 18 sind die beiden Alternativen (Experiment und Theorie) zur Bestimmung von Struktur und Funktion eines Proteins in Form eines Flussdiagramms

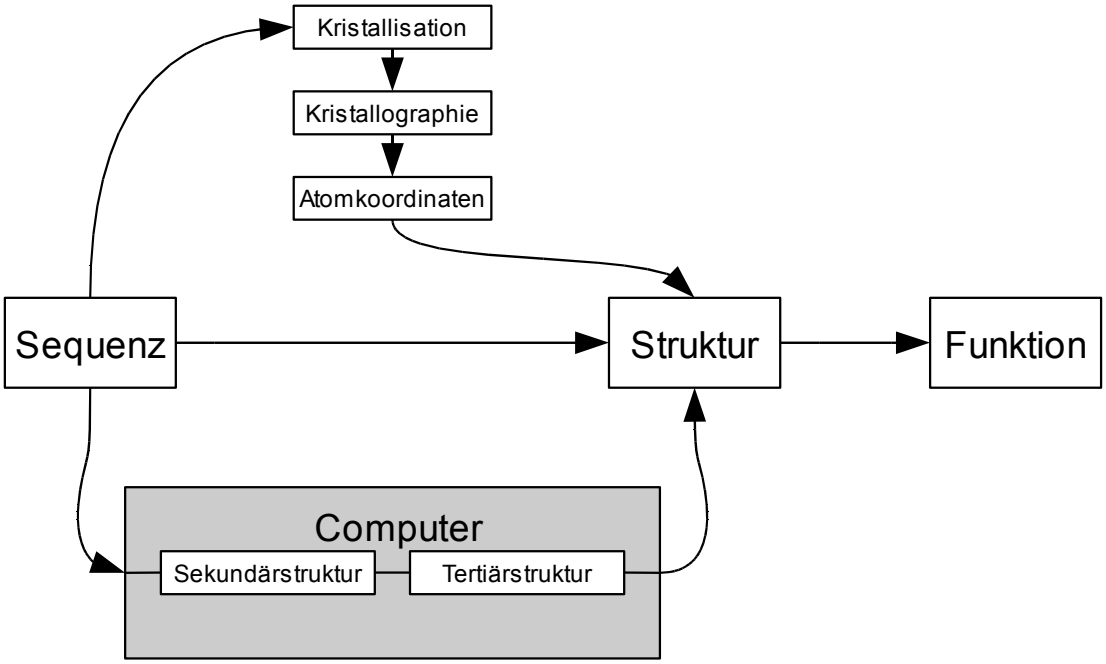


Abbildung 18: *Computergestützte Strukturvorhersage als alternative zur Strukturbestimmung*

dargestellt. Der horizontale Pfad in der Mitte der Abbildung steht für den Zusammenhang Sequenz-Struktur-Funktion, der bereits angesprochen wurde. Der obere Pfad im Flussdiagramm stellt den gängigen Weg zur Bestimmung der Proteinstruktur dar, der in diesem Fall aus der Kristallisation des betreffenden Proteins, der Röntgenkristallographie und der Bestimmung der Struktur aus den Atomkoordinaten besteht. Demgegenüber steht der untere Pfad, auf dem mittels eines Computers und eines geeigneten Modells der Proteinfaltung die Struktur direkt aus der Sequenz berechnet werden kann.

Da die einzelnen Kräfte, die zur Proteinfaltung beitragen, bekannt und berechenbar sind, scheint ein solches Modell der Proteinfaltung im Bereich des Möglichen zu liegen. Jedoch sind gerade das Fehlen einer eindeutigen Zuordnung von Strukturen zu Sequenzen und die Komplexität des Gleichgewichtes zwischen den Kräften, die die Faltung von Proteinen und deren Stabilität bewirken (Abschnitt 2.4.1), Faktoren, welche das Aufstellen eines expliziten mathematischen Modells der Proteinfaltung stark erschweren. Die Suche nach einem exakten Modell für die Faltung von Proteinen ist somit bis zum heutigen Tage ein ungelöstes Problem. Jedoch existieren computergestützte Verfahren, die dieses Problem, oder zumindest einen Teil davon, durch Approximationen zu lösen versuchen. Ein solches mögliches Verfahren ist in Abbildung 18 innerhalb des unteren Pfades dargestellt. Der als grauer Kasten dargestellte Computer berechnet aus der Sequenz zunächst die Sekundär-

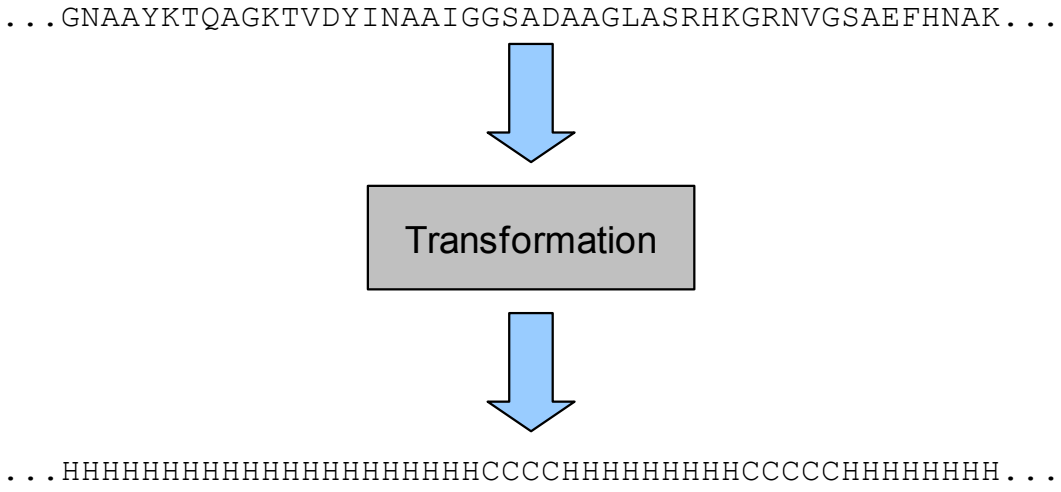


Abbildung 19: *Das Problem der Sekundärstrukturvorhersage*

struktur eines Proteins und verwendet daraufhin das Ergebnis dieser Rechnung zur Bestimmung der Tertiärstruktur. Dieser Zwischenschritt wird als die *Sekundärstrukturvorhersage von Proteinen* bezeichnet (engl. *protein secondary structure prediction*) und gilt als ein wichtiger Schritt bei der Bestimmung der dreidimensionalen Struktur von Proteinen [22]. Die Suche nach einer neuen Methode zur Sekundärstrukturvorhersage ist Gegenstand dieser Arbeit.

2.7. Sekundärstrukturvorhersage von Proteinen

Wie in Abschnitt 2.4 bereits erwähnt, ist die Tertiärstruktur eines Proteins von der Verteilung von Sekundärstrukturtypen entlang der Polypeptidkette abhängig. Die Sekundärstruktur wiederum wird von der Sequenz des Proteins bestimmt. Hierbei kann die gleiche Argumentation wie im vorherigen Abschnitt verwendet werden: Proteine mit gleicher Sequenz bilden unter gleichen Bedingungen die gleiche Verteilung von Sekundärstrukturtypen aus, weshalb die Sekundärstruktur auch in der Sequenz codiert sein muss. Jedoch ist auch in diesem Fall die Korrelation von Sequenz und Sekundärstruktur nicht bekannt. Abbildung 19 zeigt eine schematische Darstellung des Problems der Sekundärstrukturvorhersage. Dieses kann vereinfacht als die Suche nach einer Transformation angesehen werden, die eine Zeichenkette in eine andere überführt, nämlich die Primärstruktur in die Sekundärstruktur. Erstere ist in Abbildung 19 als Zeichenkette im Einbuchstabencode im oberen Bereich der Abbildung dargestellt. Letztere wird von der Zeichenkette im unteren Bereich der Abbildung repräsentiert. Diese enthält zu jedem Residuum in der Sequenz des Proteins einen

Buchstaben, der aussagt, zu welchem Sekundärstrukturtyp dieses Residuum gehört. Das Problem der Sekundärstrukturvorhersage hat somit einen Vorteil gegenüber der Suche nach einem Modell für die Proteinfaltung: Im letzten Fall ist ein Modell gesucht, dass für alle in einer Proteinsequenz vorhandenen Aminosäuren deren Koordinaten im dreidimensionalen Raum berechnet, also eine eindimensionale Größe in eine dreidimensionale Größe überführt. Im ersten Fall muss lediglich eine Zeichenkette in eine andere umgewandelt werden, was das Problem insgesamt eindimensional macht.

Einer der ersten Ansätze zur Sekundärstrukturvorhersage ist die *Chou-Fasman-Methode* ([23], [24], [25]), benannt nach ihren Erfindern Chou und Fasman, die sie in den 1970ern entwickelten. Sie versuchten anhand von relativen Häufigkeiten von Aminosäuren in den Sekundärstrukturen der damals bekannten Proteine ein Modell zu bestimmen, das für eine gegebene Sequenz die Wahrscheinlichkeit ergab, dass diese Sequenz zu einem bestimmten Sekundärstrukturtyp gehört. Bei dem von ihnen entwickelte einfachen Modell wurden 50-60% der Sekundärstrukturen korrekt erkannt. Die im Jahr 1978 entwickelte *GOR-Methode* [26], benannt nach ihren Erfindern Garnier, Osguthorpe und Robson, verwendet Bayes'sche Statistik um die Sekundärstruktur mittels eines Modells bedingter Wahrscheinlichkeiten zu berechnen. Dieses Modell verbessert die Korrektheit der Sekundärstrukturvorhersage gegenüber der Chou-Fasman-Methode auf 65%. Beide Methoden sind rein empirisch und benutzen einfache, manuell erstellte Modelle zur Berechnung der Sekundärstruktur eines Proteins. Moderne Methoden ([27], [28], [29], [30], [31], [32], [1]) besitzen eine gewisse Ähnlichkeit zu diesen ersten Ansätzen: Auch sie funktionieren empirisch, indem sie bekannte Daten von Sekundärstrukturen auswerten. Jedoch liegt der Unterschied in der Verwendung von Computern und Ansätzen aus der *statistischen Lerntheorie* (engl. *statistical learning theory*). Die Korrektheit der Vorhersagen dieser Verfahren liegt im Bereich von 80%. Diese hohe Verbesserung resultiert zum größten Teil aus der Verwendung evolutionärer Daten (sogenannter *Sequenzprofile*) und großer Datenmengen, deren effektive Handhabung durch die Verwendung der Methoden der statistischen Lerntheorie ermöglicht wird. Diese Tatsache zeigt, dass es durchaus von Vorteil ist, die Prinzipien der Lerntheorie auf das Problem der Sekundärstrukturvorhersage anzuwenden. Auf diese Prinzipien soll zunächst im nächsten Kapitel eingegangen werden, bevor eine Beschreibung moderner Methoden angegangen und letztendlich die Entwicklung einer eigenen Methode zur Sekundärstrukturvorhersage von Proteinen vorgestellt werden kann.

3. Lerntheorie

In einer Welt, die durch Naturgesetze bestimmt ist, kann im Prinzip der Ausgang eines jeden Experimentes vorhergesagt werden. Weiterhin kann jede Beobachtung, die im Laufe eines solchen Experimentes gemacht wird, erklärt und begründet werden. Die einzig notwendige Bedingung hierfür ist, dass die dem Experiment zugrunde liegenden Naturgesetze bekannt sind. Diese Bedingung ist jedoch nicht hinreichend, denn die Kenntnis der physikalischen Gegebenheiten eines Prozesses ermöglicht nicht immer eine exakte Vorhersage oder Beschreibung seines Ausgangs oder seines zeitlichen Verlaufs. Neben der Kenntnis der Naturgesetze, denen ein Prozess unterliegt, gibt es weitere Faktoren, die es erschweren den genauen Zusammenhang zwischen einem Experiment und dessen beobachtetem Ausgang zu finden. Einerseits wäre da die Komplexität des Systems, mit dem experimentiert wird. Aus der Chaostheorie ist bekannt, dass ein vollkommen deterministisches System, also ein System, das definierten Gesetzen genügt, bei ausreichend hoher Systemkomplexität chaotisch reagieren kann. Das System neigt trotz der Tatsache, dass es durch bekannte Regeln oder Gesetze beschrieben wird, zu einer gewissen Unvorhersagbarkeit. Diese Eigenschaft eines Systems bezeichnet man als *deterministisches Chaos*. Deterministisches Chaos tritt bei nichtlinearen dynamischen Systemen auf und zeichnet sich dadurch aus, dass der Ausgang eines Experiments mit einem solchen System empfindlich von den Anfangsbedingungen des Experimentes abhängt. Das wohl bekannteste Beispiel für ein chaotisches System ist die von dem Meteorologen Edward N. Lorenz im Jahre 1963 versuchte Computersimulation zur Entstehung von Zyklonen und Antizyklonen in der Atmosphäre, die zur Vorhersage des Wetters genutzt werden sollte [33]. Lorenz benutzte dabei ein vereinfachtes Modell zur Beschreibung von Flüssigkeiten und Gasen bei deren Erhitzung, welches aus einem System dreier miteinander verbundener Differentialgleichungen bestand, mit deren Hilfe die Berechnungen durchgeführt wurden. Die Berechnungen lieferten das zeitliche Verhalten des Modellsystems, indem eine numerische Integration über die Zeit durchgeführt wurde. Das Verfahren berechnete auf iterative Weise den Zustand des Modellsystems zu diskreten Zeitpunkten, indem es den augenblicklichen Zustand des Systems verwendete, um den nächsten Systemzustand zu berechnen, der um ein festgelegtes Zeitinkrement in der Zukunft lag. Lorenz stellte schnell fest, dass geringfügige Änderung der Ausgangsgrößen der Berechnung, also des initialen Zustands des Systems, zu stark divergierenden zeitlichen Verläufen führen konnten. In einem späteren Artikel [34] erwähnt

Lorenz, dass bereits eine beliebige Änderung in der vierten Nachkommastelle zu diesem chaotischen Verhalten führt und beschreibt eine Reihe von Experimenten zur Untersuchung dieses Effekts. Bei diesen Versuchen wurde jeweils eine der Ausgangsgrößen um einen kleinen Wert gegenüber einer existierenden Berechnung verändert. Daraufhin wurde eine zweite Berechnung mit den veränderten Anfangswerten vorgenommen und das zeitliche Verhalten beider Berechnungen miteinander verglichen. Es stellte sich heraus, dass sich das System bei beiden Berechnungen anfangs noch relativ ähnlich verhielt, aber mit zunehmender Zeit ein sich immer stärker unterscheidendes Verhalten zeigte. Die Abweichung der Anfangsgrößen zur Berechnung des zeitlichen Verlaufs der Bildung von Luftströmungen war dabei in der Größenordnung eines Lufthauchs, der durch den Flügelschlag einer Möwe erzeugt wird, was Lorenz im selben Artikel zur folgenden Aussage verleitete:

„One meteorologist remarked that if the theory were correct, one flap of a sea gull's wings would be enough to alter the course of the weather forever. The controversy has not yet been settled, but the most recent evidence seems to favor the sea gulls.“

Trotz der von Lorenz verwendeten Analogie mit dem Flügelschlag einer Möwe, wurde der von ihm untersuchte Effekt später unter einem anderen Namen bekannt: *Schmetterlingseffekt* (englisch: *Butterfly Effect*). Um die Entstehung dieser Namensgebung ranken sich viele Gerüchte, jedoch ist klar, dass diese Bezeichnung durch einen von Lorenz im Jahre 1972 gehaltenen Vortrag mit dem Titel *„Predictability: Does a flap of a butterfly's wing in Brazil set off a tornado in Texas?“* ins Leben gerufen wurde [35].

Anschaulich beschreibt der Schmetterlingseffekt die Problematik komplexer Systeme: Selbst wenn ein komplexes System sich durch eine Reihe einfacher oder mathematisch einfach beschreibbarer Regeln definieren lässt, kann eine kleine Abweichung in einem den Systemzustand beschreibenden Wert (*Zustandsvariable*) zu unvorhergesehenen Konsequenzen im zeitlichen Verhalten des Systems führen. Man mag nun anmerken, dass der von Lorenz beschriebene Effekt nur bedingt einen Bezug zur Realität habe, da der beobachtete Effekt ein Artefakt sein könnte. Die verwendete Numerik und die Tatsache, dass alle Berechnungen auf Computern durchgeführt wurden, deren Genauigkeit begrenzt ist, würden diesen Schluss nahe legen. Lorenz schreibt jedoch unter Berufung auf Birkhoff [36], dass die von ihm erzielten Ergebnisse empirisch sind und nicht bloße Theorie. Weiterhin wird der von Lorenz beschriebene Effekt bereits in einem Artikel von W. S. Franklin

aus dem Jahre 1898 [37] erwähnt, wobei dort der Flügelschlag einer Heuschrecke als Analogie verwendet wird.

Der Effekt um die Abhängigkeit des Verhaltens komplexer Systeme von kleinen Änderungen der Anfangsgrößen ist also wohl bekannt und zeigt, dass die Vorhersagbarkeit eines solchen Systems durch diese stark beeinträchtigt werden kann, auch wenn alle physikalischen Größen und deren Zusammenspiel bekannt sind. Diese Tatsache ist einer der Faktoren, die es erschweren den Ausgang eines Experiments vorherzusagen. Der andere Faktor ist das gänzliche Fehlen eines mathematischen Modells für die Beschreibung des Zusammenspiels der in einem System wirkenden Naturgesetze. Das Wissen um die wirkenden Kräfte, die einen Effekt hervorrufen können, trägt zwar durchaus zu dem Verständnis und zur Erklärung dieses Effektes bei, ermöglicht es jedoch nicht, das Resultat eines leicht abgeänderten Experiments vorherzusagen. Diese Änderung kann nämlich ein anderes Resultat als das vorher beobachtete ergeben. Als anschauliches Beispiel sei die Faltung von Proteinen genannt (siehe Abschnitt 2.4), die den Ausgangspunkt des in dieser Arbeit behandelten Problems der Sekundärstrukturvorhersage darstellt (Abschnitt 2.7). Die Faltung, die ein Protein annimmt, kann im Nachhinein analysiert und in der Regel auch erklärt werden, da die Kenntnis der Struktur Anhaltspunkte dafür liefert, wie die Kräfte zusammenwirken, die zu ihrer Entstehung und Stabilisierung beitragen (siehe Abschnitt 2.4.1). Kleine Änderungen der Sequenz des Proteins oder aber der Umgebungsbedingungen (Druck, pH-Wert, Temperatur) können, wie die Beispiele der Sichelzellenanämie (Punktmutation, siehe Abschnitt 2.2) und der Denaturierung (Änderung der Umgebung, siehe Abschnitt 2.4.2) zeigen, bereits große Änderungen der Struktur herbeiführen. Diese Änderungen können im Allgemeinen aufgrund der Komplexität der Wechselwirkung der einzelnen Kräfte, die bei der Proteinfaltung wirken, nicht vorhergesagt werden. So kann es sein, dass zwei dieser Kräfte sich in einem Fall gegenseitig kompensieren und in einem anderen Fall verstärken. Oder dass ein bestimmter Faktor in einem Fall keine Rolle spielt, obwohl er bei anderer Gelegenheit eine ausschlaggebende Rolle gespielt hat. Die die Faltung bedingenden Faktoren gehen also nichtlinear in das System ein. Wie man aus diesem Beispiel ableiten kann, wird es mit steigender Komplexität immer schwieriger ein geeignetes Modell für das Verhalten eines solchen Systems aufzustellen. Wie das deterministische Chaos ist auch das Aufstellen eines Modells zur Beschreibung eines System von dessen Komplexität abhängig.

Bei komplexen Systemen, wie beispielsweise Proteinen, ist es offensichtlich schwierig einen Zusammenhang zwischen den Anfangsbedingungen und dem beobachteten Ausgang eines Experiments zu finden, der das Verhalten dieses Systems verlässlich vorhersagt. Es stellt sich nun die Frage, inwiefern das Verhalten eines Systems mithilfe eines geeigneten Modells derart approximiert werden kann, dass das Modell in einem möglichst großen Prozentsatz an Fällen bei gegebenen Anfangsbedingungen (oder *Eingangsdaten*) die Beobachtungen, beziehungsweise den Ausgang eines Experiments (oder *Ausgangsdaten*), korrekt vorhersagen kann. Lorenz erwähnt in einem seiner Artikel die Möglichkeit mit Hilfe von Methoden aus der Statistik das zeitliche Verhalten nichtlinearer dynamischer Systeme vorherzusagen [34]. Die beschriebene Methode nimmt einen linearen Zusammenhang zwischen den Zustandsvariablen und dem nächsten Systemzustand an, wobei die gewichtete Summe der aktuellen Zustandsvariablen verwendet wird, um eine der Zustandsvariablen des nächsten Zustands zu berechnen. Jeder Summand hat ein eigenes Gewicht und jedes dieser Gewichte ist ein freier Parameter des Modellsystems. Zur Berechnung dieser freien Parameter werden Statistiken verwendet, die von realen Beobachtungen des gegebenen Systems stammen.

Zusammengefasst versucht das Verfahren das Verhalten eines unvorhersagbaren oder unbekannten Systems anhand eines begrenzten Datensatzes und einer einfachen Modellfunktion nachzubilden. Es ist ersichtlich, dass ein solches Verfahren in der Regel lediglich über eine endliche Genauigkeit verfügt. Jedoch ist eine derartige Approximation in vielen Fällen die einzige Möglichkeit den gesuchten Zusammenhang zwischen Eingangs- und Ausgangsdaten nachzubilden, da bei vielen Prozessen der reale Zusammenhang entweder nicht bekannt ist oder aber zu komplex ist, um ihn sinnvoll nachbilden zu können. Heutzutage haben sich auf vielen Gebieten solche approximativen Verfahren durchgesetzt, da fortschrittliche Approximationsmethoden eine durchaus akzeptable Genauigkeit ermöglichen. Die Anwendung solcher Verfahren ist nicht ausschließlich auf zeitliche Verläufe unbekannter Systeme beschränkt. So kann beispielsweise, wie im Fall der Faltung von Proteinen, lediglich der Endzustand eines solchen zeitlichen Verlaufs bei gegebenen Anfangsbedingungen von Interesse sein. Ein weiteres Anwendungsgebiet wäre zum Beispiel die Bilderkennung. Dabei ist das Ziel, auf digitalen Bildern bestimmte Merkmale, wie Gesichter oder Personen, automatisiert erkennen zu können. Bei diesem Problem ist es nicht möglich, einen exakten Zusammenhang zwischen dem digitalen Bild und dem gesuchten Merk-

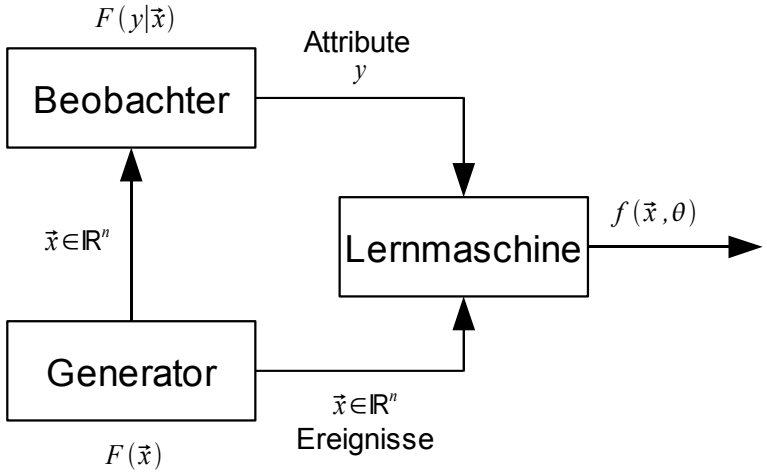


Abbildung 20: *Schematische Darstellung eines Lernprozesses*

mal in einen einfachen Satz von Regeln zusammenzufassen, die für einen Computer verständlich wären. Auch hier haben sich approximative Methoden bewährt, die anhand eines endlichen Satzes an Beispielen versuchen, den gewünschten Zusammenhang möglichst genau nachzubilden. Solche Methoden werden als *Lernverfahren* bezeichnet, aufgrund ihrer Natur, aus Beispielen eine Verknüpfung von Ausgangsdaten und Eingangsdaten zu *erlernen*. Die automatisierte Umsetzung solcher Lernverfahren mit Hilfe von Computerprogrammen oder auch in seltenen Fällen Hardware wird als *maschinelles Lernen* bezeichnet und ist ein Teilgebiet der Künstlichen Intelligenz.

3.1. Das Lernproblem

Die mathematischen Grundlagen von Lernverfahren sind bereits länger bekannt. Sowohl der auf dem Satz von Bayes (benannt nach Thomas Bayes, der einen Spezialfall dieses Satzes in seinem Essay aus dem Jahre 1763 beschrieb [38]) basierende *Bayes-Klassifikator* ([39], [40]) wie auch die von Fisher im Jahre 1936 beschriebene *Diskriminanzanalyse* [41] sind Beispiele für Lernverfahren. Die mathematische Formalisierung von Lernverfahren wurde jedoch erst in den Jahren 1960–1990 von Vladimir Vapnik und Alexey Chervonenkis aufgestellt. Im Zentrum der von ihnen begründeten statistischen Lerntheorie steht das sogenannte *Lernproblem*. In seinem Buch [42] beschreibt Vapnik das Lernproblem als das Problem, ausschließlich auf der Grundlage einer begrenzten Anzahl von Beobachtungen eine gesuchte Abhängigkeit zu finden. Das in der Lerntheorie verwendete Modell eines Lernprozesses besteht aus drei Komponenten, wie sie in Abbildung 20 schematisch dargestellt sind. Die einzelnen Komponenten sind wie folgt definiert:

1. **Generator/Umgebung:** Erzeugt zufällige Eingangsvektoren (Ereignisse) $\vec{x} \in \mathbb{R}^n$ mit definiter aber unbekannter Verteilungsfunktion $F(\vec{x})$.
2. **Beobachter/Überwacher:** Weist jedem Eingangsvektor (Ereignis) $\vec{x}$ einen Ausgangswert (Attribut) y mit fester aber unbekannter Verteilungsfunktion $F(y|\vec{x})$ zu. Hierin ist auch der Spezialfall einer eindeutigen Zuordnung mittels einer Funktion $y = f(\vec{x})$ enthalten.
3. **Lernmaschine/Lernverfahren:** Implementiert einen Satz von Funktionen $f(\vec{x}, \theta)$ mit $\theta \in \Lambda$, mit deren Hilfe die Zuordnung der Eingangsvektoren (Ereignisse) $\vec{x}$ zu den dazugehörigen Ausgangswerten (Attributen) y approximiert werden kann. Hierbei stellt Λ einen abstrakten Parameterraum dar.

Mit dieser Definition eines Lernprozesses lässt sich das Lernproblem mathematisch beschreiben als das Suchen einer Funktion $f(\vec{x}, \theta)$ mit $\theta \in \Lambda$, welche den Beobachter am besten approximiert. Dabei wird davon ausgegangen, dass die vorhandene Menge an Beispielen (im weiteren Text als *Lerndatensatz* bezeichnet) aus einer beschränkten Menge unabhängiger Beobachtungen $(\vec{x}, y)$ mit der Verteilung $F(\vec{x}, y) = F(\vec{x}) F(y|\vec{x})$ besteht.

3.1.1. Das Prinzip der Risikominimierung

Mit dieser Definition des Lernproblems kann nun ein mathematischer Ansatz zu dessen Lösung angegangen werden. Hierzu wird zunächst eine sogenannte Kostenfunktion (engl. *loss function*) $L(y, f(\vec{x}, \theta))$ definiert, die ein Maß für die Abweichung des wahren Ausgabewertes y von der Ausgabe der Lernmaschine $f(\vec{x}, \theta)$ darstellt. In der Regel wird $L(y, f(\vec{x}, \theta))$ so gewählt, dass eine größere Abweichung der Lernmaschine vom wahren Ausgabewert zu einem größeren Funktionswert von $L(y, f(\vec{x}, \theta))$ führt. Mit Hilfe von $L(y, f(\vec{x}, \theta))$ kann nun ein sogenanntes Risikofunktional (engl. *risk functional*) mit folgender Form definiert werden:

$$R(\theta) = \int L(y, f(\vec{x}, \theta)) dF(\vec{x}, y)$$

Das Risikofunktional stellt den Erwartungswert der Abweichung der Lernmaschine von den erwarteten Ausgangswerten dar. Dabei wird von der Grundgesamtheit aller Daten ausgegangen, die für das zu lösende Lernproblem beziehungsweise den gesuchten Zusammenhang relevant sind. Für die Lösung eines Lernproblems ist die Minimierung von $R(\theta)$ hinsichtlich $f(\vec{x}, \theta)$ notwendig. Dies ist jedoch aufgrund mangelnder Kenntnis der Ver-

teilung $F(\vec{x}, y)$ nicht direkt möglich, was der beschriebenen *Risikominimierung* (engl. *risk minimization*) einen rein theoretischen Stellenwert verleiht. In der Praxis wird die *empirische Risikominimierung* verwendet, die in Abschnitt 3.1.3 näher beschrieben wird.

3.1.2. Beispiele von Kostenfunktionen

Vapnik erwähnt drei grundlegende Lernprobleme, für die Beispiele von Kostenfunktionen genannt werden, die bei der Risikominimierung verwendet werden können. Auf diese soll an dieser Stelle kurz eingegangen werden, um die Verwendung von Kostenfunktionen zu veranschaulichen und Anwendungsgebiete für Lernmaschinen zu zeigen.

3.1.2.1. Kostenfunktion für Klassifikationsprobleme

Beim einfachsten Klassifikationsproblem wird eine Funktion $f(\vec{x}, \theta)$ gesucht, die die gegebenen Daten möglichst genau zwei gegebenen Klassen zuordnen kann. Damit sind die Ausgaben, die der Beobachter erzeugt, auf die Menge $\{0,1\}$ beschränkt, welche die Klassenindices repräsentiert. Zur Lösung dieses Lernproblems kann nun die folgende Kostenfunktion verwendet werden:

$$L(y, f(\vec{x}, \theta)) = \begin{cases} 0 & \text{, wenn } y = f(\vec{x}, \theta) \\ 1 & \text{, wenn } y \neq f(\vec{x}, \theta) \end{cases}$$

Dies ist eine recht einfache Kostenfunktion, welche aber das Klassifikationsproblem durchaus löst. Die Funktion bestraft Fehlklassifikationen und begünstigt richtige Klassifikationen. Unter Verwendung dieser Kostenfunktion stellt das Risikofunktional die Wahrscheinlichkeit der Fehlklassifikation für eine Funktion $f(\vec{x}, \theta)$ dar.

Das am Anfang von Kapitel 3 erwähnte Problem der Mustererkennung auf Bildern ist ein solches Klassifikationsproblem. Ist beispielsweise eine Funktion gesucht, die auf Bildern mit der Höhe H und der Breite B Gesichter erkennen soll, so können die Werte der Rot-, Grün- und Blaukomponenten der einzelnen Pixel als Eingabewerte genommen werden. Für die Vektoren $\vec{x}$ ergibt sich also $\vec{x} \in \mathbb{R}^{3 \cdot H \cdot B}$. Die Ausgabewerte $y \in \{0,1\}$ entsprechen dann den beiden Klassen „Gesicht“ (Wert 1) und „kein Gesicht“ (Wert 0). Klassifikationsprobleme spielen im Rahmen dieser Arbeit eine zentrale Rolle, weshalb in Abschnitt 3.2 nochmals genauer auf sie eingegangen wird.

3.1.2.2. Kostenfunktion für Regressionsprobleme

Beim Regressionsproblem entsprechen die Ausgabewerte y den Werten einer unbekannten reellwertigen Funktion $y=g(\vec{x})$. Ziel ist es nun eine Modellfunktion $f(\vec{x},\theta)$ zu finden, für die folgendes gilt:

$$\forall \vec{x}: |f(\vec{x},\theta)-g(\vec{x})|<\epsilon,\ \ 0<\epsilon$$

Die Variable ϵ ist dabei eine positive reelle Zahl, die im Idealfall möglichst klein sein sollte. In Worte gefasst soll der Verlauf der Modellfunktion $f(\vec{x},\theta)$ möglichst nahe am Verlauf der realen Funktion $g(\vec{x})$ liegen. Hieraus ergibt sich schnell die Kostenfunktion als der quadratische Fehler:

$$L(y,f(\vec{x},\theta))=(y-f(\vec{x},\theta))^2$$

Ein Beispiel für ein Regressionsproblem das Problem der Vorhersage des zeitlichen Verlaufs von Aktienkursen an der Börse [43]. In diesem Fall würde der Ausgabewert y dem vorhergesagten Börsenkurs entsprechen, während die Eingabewerte $\vec{x}$ den Faktoren entsprechen würden, für die angenommen wird, dass sie Einfluss auf die Aktienkurse haben.

3.1.2.3. Kostenfunktion zur Schätzung von Dichtefunktionen

Das Problem der Dichteschätzung ist mit dem Problem der Regression eng verwandt. Im Falle der Dichteschätzung wird für die Funktion $f(\vec{x},\theta)$ angenommen, dass sie eine Dichtefunktion $p(\vec{x},\theta)$ darstellt. Gesucht ist also eine Funktion, die eine unbekannte Dichtefunktion anhand gegebener Daten am besten approximiert. Laut Vapnik ist in diesem speziellen Fall die Kostenfunktion $L(p(\vec{x},\theta))=-\log(p(\vec{x},\theta))$ zu wählen, um das Risikofunktional minimieren zu können. Diese Kostenfunktion entspricht dem Informationsgehalt des Ereignisses $\vec{x}$ [44].

3.1.3. Das Prinzip der Empirischen Risikominimierung

Wie bereits in Abschnitt 3.1.1 erwähnt, wird bei der Risikominimierung von der Grundgesamtheit der Daten ausgegangen. Das heißt mit anderen Worten, es wird angenommen, dass die für das vorliegende Lernproblem relevanten Daten in all ihren Ausprägungen bekannt sind. Zudem wird eine definite Verteilung der Daten $F(\vec{x},y)$ als gegeben vorausgesetzt. In der Praxis ist diese Verteilung jedoch nicht bekannt und die zur Verfügung stehenden Daten sind lediglich ein begrenzter Ausschnitt der Grundgesamtheit. Dies wirft die

Frage auf, inwiefern das Prinzip der Risikominimierung überhaupt einen praktischen Bezug hat.

Zur Klärung dieser Frage kann das aus der Statistik bekannte Gesetz der großen Zahlen herangezogen werden. Gegeben sei eine Folge von unabhängigen und identisch verteilten Zufallsvariablen $X_1, X_2, \ldots X_N$ mit endlichem Erwartungswert, d.h.:

$$\langle X_1 \rangle = \langle X_2 \rangle = \ldots = \langle X_N \rangle = \mu < \infty$$

Für diese Folge ist der Mittelwert definiert als: $\bar{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i$

Das Gesetz der großen Zahlen sagt nun aus, dass der obige Mittelwert mit steigender Länge N der Folge gegen den Erwartungswert konvergiert, so dass gilt:

$$\lim_{N \to \infty} \bar{X}_N = \mu$$

Das Gesetz der großen Zahlen liegt in zwei Versionen vor, deren Kern durch die obige Aussage beschrieben wird. Das schwache Gesetz der großen Zahlen sagt aus, dass der Mittelwert stochastisch gegen den Erwartungswert konvergiert, also dass die Gleichung

$$\lim_{N \to \infty} P\left(|\bar{X}_N - \mu| < \epsilon\right) = 1$$

mit beliebigem $\epsilon > 0$ erfüllt ist. Dabei entspricht $P\left(|\bar{X}_N - \mu| < \epsilon\right)$ der Wahrscheinlichkeit, dass die Bedingung $|\bar{X}_N - \mu| < \epsilon$ zutrifft. Das starke Gesetz der großen Zahlen hingegen sagt aus, dass der Mittelwert sicher gegen den Erwartungswert konvergiert. Dies wird durch die Gleichung

$$P\left(\lim_{N \to \infty} \bar{X}_N = \mu\right) = 1$$

ausgedrückt.

Im Falle des Lernproblems ist aus Abschnitt 3.1.1 bereits bekannt, dass das Risikofunktional für eine gegebene Kostenfunktion $L(y, f(\vec{x}, \theta))$ deren Erwartungswert darstellt. Es sei nun ein *empirisches Risikofunktional* (engl. *empirical risk functional*) R_{emp} folgendermaßen definiert:

$$R_{emp}(\theta) = \frac{1}{N^{lern}} \sum_{i=1}^{N^{lern}} L(y_i, f(\vec{x}_i, \theta))$$

Dieses stellt den Mittelwert der Kostenfunktion $L(y, f(\vec{x}, \theta))$ unter Verwendung eines begrenzten Datensatzes $D^{lern} = \{(\vec{x}_i, y_i) | 1 \leq i \leq N^{lern}\}$ dar und so kann davon ausgegangen werden, dass es nach dem Gesetz der großen Zahlen mit steigender Datenmenge gegen das Risikofunktional konvergiert. Das empirische Risikofunktional kann nun, wie auch das Risikofunktional selbst, hinsichtlich der Parameter θ minimiert werden. Sei nun $f(\vec{x}, \theta_{opt})$ diejenige Funktion, die das Risikofunktional minimiert, während $f(\vec{x}, \hat{\theta})$ die Funktion sei, die das empirische Risikofunktional minimiert, so kann angenommen werden, dass $f(\vec{x}, \hat{\theta})$ bei genügend großer Datenmenge eine beliebig gute Approximation von $f(\vec{x}, \theta_{opt})$ darstellt. Das empirische Risikofunktional bietet also eine Möglichkeit zur approximativen Lösung des Lernproblems, wie sie bereits am Anfang von Kapitel 3 angedeutet wurde. Das Lösungsverfahren unter Verwendung des empirischen Risikofunktionals wird als *empirische Risikominimierung* (engl. *empirical risk minimization*), abgekürzt *ERM*, bezeichnet.

Im Allgemeinen ist bei Anwendung der ERM nicht sichergestellt, dass der Parametersatz $\hat{\theta}$ dem Paramtersatz θ_{opt} entspricht. Ein gefundener Parametersatz $\hat{\theta}$, der das empirische Riskofunktional minimiert, muss also nicht zwangsläufig auch das Risikofunktional selbst minimieren. Dies wirft zwei Fragen auf:

1. Enthält ein gegebener Lerndatensatz genug Informationen, um den gewünschten Zusammenhang geeignet lernen zu können?
2. Besitzt die gewählte Lernmaschine ausreichend Kapazität, um auf Grundlage eines gegebenen Datensatzes den bestmöglichen Lerneffekt zu erreichen.

Beides soll im folgenden Abschnitt betrachtet werden.

3.1.4. Generalisierungsfähigkeit von Lernmaschinen

In Zusammenhang mit Lernmaschinen wird häufig der Begriff *Generalisierung* beziehungsweise *Generalisierungsfähigkeit* verwendet. Generalisierungsfähigkeit beschreibt die Fähigkeit einer Lernmaschine, nach dem Lernen auf einem Lerndatensatz auch auf einem zweiten, vom Lerndatensatz disjunkten Datensatz gute Ergebnisse zu liefern. Dieser zweite Datensatz wird *Testdatensatz* genannt. Ein gutes Verhalten einer trainierten Lernmaschine auf einem Testdatensatz ist nicht notwendigerweise gewährleistet.

Für alle nachfolgenden Abschnitte soll davon ausgegangen werden, dass die Parameter der Funktion $f(\vec{x},\theta)$, wie auch die aus der ERM gewonnenen Parameter, reelle Zahlen sind, die in einem Vektor reeller Zahlen zusammengefasst werden können. Es soll also $\theta \in \mathbb{R}^K$ und $\hat{\theta} \in \mathbb{R}^K$ gelten, wobei K der Anzahl der freien Parameter der Funktion $f(\vec{x},\theta)$ entspricht. Dies ist gängige Praxis bei der im Rahmen dieser Arbeit entstandenen Software, wie auch bei den meisten bekannten Verfahren zur Lösung des Lernproblems. Zur Verdeutlichung der Generalisierungsfähigkeit von Lernmaschinen sei nun angenommen, dass eine Lernmaschine, die die nichtlineare Funktion $f(\vec{x},\theta)$ implementiert, mit Hilfe des Lerndatensatzes $D^{lern} = \{\vec{d}_i^{lern} = (\vec{x}_i, y_i) | 1 \le i \le N\}$ die Eigenschaften einer unbekannten Funktion $y = g(\vec{x})$ lernen soll. Nach dem Lernen soll das Verhalten der Lernmaschine auf einem Testdatensatz $D^{test} = \{\vec{d}_j^{test} = (\vec{x}_j, y_j) | 1 \le j \le M\}$ mit $D^{lern} \cap D^{test} = \emptyset$ getestet werden.

Eine Lernmaschine besitzt optimale Generalisierungsfähigkeit, wenn gilt:

$$\hat{\theta} = \arg\min_{\theta} R_{emp}(\theta) \Rightarrow \hat{\theta} = \arg\min_{\theta} R(\theta)$$

In diesem Idealfall würde die aus dem begrenzten Lerndatensatz D^{lern} gewonnene Information ausreichen, um der Lernmaschine eine optimale Vorhersage beliebiger Daten aus der Verteilung $F(\vec{x}, y)$ zu ermöglichen. Die Funktion $g(\vec{x})$ würde in diesem Fall optimal nachgebildet werden. Dies ist jedoch nicht immer der Fall, wie bereits in Abschnitt 3.1.3 angemerkt wurde.

Das Prinzip der empirischen Risikominimierung weist Parallelen zum induktiven Denken auf, welches von Spezialfällen auf das Allgemeine zu schließen versucht. Vom induktiven Denken ist jedoch bekannt, dass es zuweilen zu Trugschlüssen führen kann. Somit können auch Lernmaschinen prinzipiell falsche Schlüsse aus den gegebenen Lerndaten ziehen. Nun stellt sich die Frage, ob es möglich ist die Generalisierungsfähigkeit einer Lernmaschine anhand begrenzter Daten abzuschätzen. Hierfür bietet sich das Verfahren der Kreuzvalidierung an. Die Idee hinter der Kreuzvalidierung ist recht einfach: Zunächst sei die Menge B die Menge aller bekannten Beobachtungen $\vec{b}_i = (\vec{x}_i, y_i)$ aus der Verteilung $F(\vec{x}, y)$. Der erste Schritt der Kreuzvalidierung unterteilt B in β gleichgroße Teilmengen B_i, die folgende Bedingungen erfüllen:

1. $B = B_1 \cup B_2 \cup \ldots \cup B_\beta$
2. $B_i \cap B_j = \emptyset, \forall i, j \in [1 \ldots \beta], i \neq j$
3. $|B_i| = \frac{|B|}{\beta}, \forall i \in [1 \ldots \beta]$

Dabei gilt für β die Ungleichung $2 \leq \beta \leq |B|$. Im nächsten Schritt können β Paare aus einem Testdatensatz $D_\alpha^{test} = B_\alpha$ und einem dazugehörigen Lerndatensatz $D_\alpha^{lern} = B \setminus D_\alpha^{test}$ definiert werden, wobei $1 \leq \alpha \leq \beta$. Für jedes $1 \leq \alpha \leq \beta$ kann nun ein Lernvorgang mittels ERM durchgeführt werden, wobei die Lösung $\hat{\theta}_\alpha$ das empirische Risikofunktional für den Lerndatensatz D_α^{lern} minimiert. Anschließend wird die Güte der gelernten Funktion $f(\vec{x}, \hat{\theta}_\alpha)$ für D_α^{lern} und D_α^{test} mittels eines geeigneten Qualitätskriteriums gemessen. Für dieses Beispiel soll das empirische Risikofunktional als Indikator für die Güte von $f(\vec{x}, \hat{\theta}_\alpha)$ verwendet werden. Dies führt zu der Definition der beiden Gütemaße

- $R_\alpha^{test} = \frac{1}{|D_\alpha^{test}|} \sum_{(x_j, y_j) \in T_\alpha^{test}} L\left(y_j, f\left(\vec{x}_j, \hat{\theta}_\alpha\right)\right)$ als mittlere Diskrepanz auf D_α^{test}
- $R_\alpha^{lern} = \frac{1}{|D_\alpha^{lern}|} \sum_{(x_i, y_i) \in D_\alpha^{lern}} L\left(y_i, f\left(\vec{x}_i, \hat{\theta}_\alpha\right)\right)$ als mittlere Diskrepanz auf D_α^{lern}

Zur Abschätzung der Generalisierungsfähigkeit einer Lernmaschine wird der Mittelwert beider Größen über alle α berechnet, sodass sich eine Schätzung der Diskrepanz auf bekannten und unbekannten Daten ergibt. Man erhält also am Ende die beiden Größen

$$\bar{R}^{lern} = \frac{1}{\beta} \sum_{i=1}^{\beta} R_\alpha^{lern} \text{ und } \bar{R}^{test} = \frac{1}{\beta} \sum_{i=1}^{\beta} R_\alpha^{test},$$

wobei $\bar{R}^{lern}$ die mittlere Fähigkeit der Lernmaschine beschreibt, gelernte Daten zu reproduzieren und $\bar{R}^{test}$ angibt, wie gut das gelernte im Mittel auf unbekannte Daten angewendet wird.

Im idealen Fall sollte sowohl $\bar{R}^{lern}$ als auch $\bar{R}^{test}$ möglichst niedrig sein. Es lassen sich jedoch zwei extreme Sonderfälle konstruieren, bei denen diese Bedingung nicht zutrifft und eine Generalisierung nicht gewährleistet ist. Dies wäre zunächst der triviale Fall, in dem schon der Verlauf von $g(\vec{x})$ auf den Lerndaten von der Lernmaschine nicht rekonstruiert werden kann. Dieser Fall tritt auf, wenn K zu klein gewählt ist und äußert sich in jeweils

einem hohen Wert von $\bar{R}^{lern}$ und $\bar{R}^{test}$. Die Lernmaschine besitzt in diesem Fall nicht die Kapazität, die in den Lerndaten steckende Information sinnvoll zu verarbeiten, sodass der Verlauf der Funktion $g(\vec{x})$ nicht nachgebildet werden kann. Diesen Effekt nennt man *Unteranpassung* (engl. *underfitting).* Als ein Beispiel für Unteranpassung kann man sich für den einfachen Fall skalarer Eingangsdaten $x \in \mathbb{R}$ vorstellen, dass die Funktion $g(x)$ ein Polynom höheren Grades ist, während die Funktion $f(x,\theta)$ eine Parabel der Form $f(x,\theta) = a \cdot x^2 + b \cdot x + c$ ist. Dabei ist $\theta = (a,b,c)$. Diese Funktion kann ein Polynom höheren Grades im Allgemeinen nicht zuverlässig abbilden, da ihr die Terme höheren Grades fehlen, die den Verlauf der Funktion $g(x)$ maßgeblich beeinflussen. Auch wenn dieses Beispiel vielleicht etwas trivial erscheinen mag, so ist es doch valide, da das Lernproblem als ein Problem der mathematischen Funktionsapproximation angesehen werden kann ([42], [45]). Der Spezialfall einer Lernmaschine, die eine quadratische Approximation eines Polynoms erzeugen soll, lässt sich beliebig verallgemeinern. Das Beispiel zeigt jedoch, dass die Wahl der Lernmaschine, beziehungsweise der Funktion $f(\vec{x},\theta)$, die Lösung des Lernproblems maßgeblich beeinflusst und dass die Kapazität einer solchen Maschine von der Anzahl der frei wählbaren Parameter abhängt.

Diese Erkenntnis führt zum zweiten Sonderfall, nämlich zu dem Fall, dass die Lernmaschine zu viele Parameter und somit eine zu hohe Kapazität besitzt. Für diesen Sonderfall kann das einfache Beispiel konstruiert werden, in dem die Lernmaschine exakt so viele Parameter besitzt wie der verwendete Lerndatensatz Elemente enthält. In diesem Fall kann man sich vorstellen, dass jeder Parameter den Zusammenhang zwischen Ein- und Ausgangsdaten für jeweils ein Element des Lerndatensatzes speichert. Dies reduziert die Lernmaschine im Prinzip zu einer Tabelle, in der für jeden Eingangsvektor $\vec{x}_i$ aus dem Lerndatensatz der entsprechende Wert y_i ausgelesen werden kann, ohne dass der wahre Zusammenhang $g(\vec{x})$ auch nur ansatzweise abgebildet wird. Dies hat zur Folge, dass für alle $\vec{d}_{\alpha i}^{lern} = (\vec{x}_i, y_i)$ aus dem Lerndatensatz D_α^{lern} zwar die Gleichung $y_i = f(\vec{x}_i, \hat{\theta}_\alpha)$ gilt, die Funktion $f(\vec{x}, \hat{\theta}_\alpha)$ bei Daten die nicht in D_α^{lern} enthalten sind in der Regel jedoch stark vom wahren Verlauf von $g(\vec{x})$ abweicht. Dies ist in Abbildung 21 für das einfache Beispiel skalarer Eingangsdaten $x \in \mathbb{R}$ skizziert, wobei die schwarzen Punkte den Lerndatensatz symbolisieren, die schwarze Kurve den wahren Verlauf von $g(x)$ und die rote Kurve den durch eine Funktion $f(x,\hat{\theta})$ approximierten Verlauf darstellt. Wie ersichtlich ist,

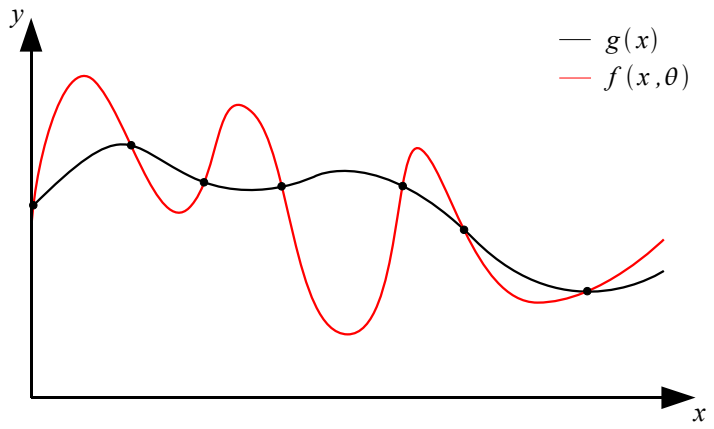


Abbildung 21: *Ein Beispiel für Überanpassung*

wird in diesem Beispiel die Generalisierung nicht gewährleistet. Würde das menschliche Lernen als Analogie herangezogen, so würde die Lernmaschine hier auswendig lernen, beziehungsweise lernen die Antworten zu geben, die der Beobachter verlangt, ohne den wahren Zusammenhang hinter dem Gelernten zu verstehen. Somit würde auch jedwede Fähigkeit fehlen, das Gelernte auf neue, unbekannte Probleme anzuwenden. Dieser Effekt wird im maschinellen Lernen wie auch in der statistischen Regression als *Überanpassung* (engl. *overfitting*) bezeichnet. Überanpassung tritt auf, wenn das verwendete Modell, in diesem Falle also die Lernmaschine, zu viele freie Parameter enthält. Was „zu viele“ im Einzelnen heißt ist von Fall zu Fall unterschiedlich, da es vom gewählten Modell und der Größe des gewählten Lerndatensatzes abhängt. Für neuronale Netze (Abschnitt 3.5.4) beispielsweise tritt Überanpassung bei einem Lerndatensatz D^{lern} auf, wenn gilt [45]:

$$\frac{\left|D^{lern}\right|}{K} < 30$$

Hier entspricht K der Anzahl der Parameter des neuronalen Netzes.

In Abbildung 22 ist der Verlauf von $\bar{R}^{lern}$ und $\bar{R}^{test}$ in Abhängigkeit von K skizziert, wobei die schwarze Kurve $\bar{R}^{lern}$ entspricht, die blaue Kurve $\bar{R}^{test}$. Im linken Teil des Diagramms ist der Bereich dargestellt, in dem Unteranpassung stattfindet. Dort besitzen beide Größen einen hohen Wert, was auf eine hohe Abweichung der Lernmaschine vom realen Zusammenhang hinweist. Wie zu sehen ist, sind beide Kurven, sowohl $\bar{R}^{lern}$ als auch $\bar{R}^{test}$, mit steigendem K fallend. Dieses Verhalten zeigen sie bis zu einem bestimmten Wert von K, der als der optimale Wert für das untersuchte Problem angesehen werden

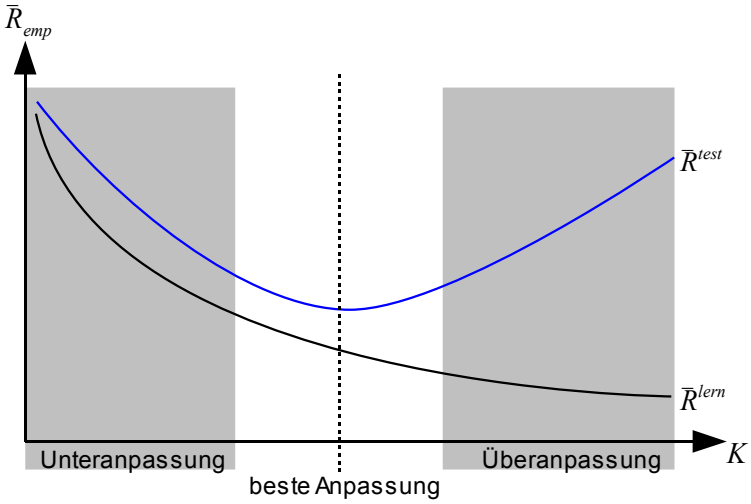


Abbildung 22: *Diskrepanz einer Lernmaschine in Abhängigkeit von der Parameterzahl*

kann. Für diesen Wert hat $\bar{R}^{test}$ sein Minimum, was dafür spricht, dass für dieses K die bestmögliche Generalisierung für die verwendete Funktion $f(\vec{x},\theta)$ erreicht wird. Ab diesem Punkt setzt langsam die Überanpassung ein, wie am Verlauf der Kurven sichtbar ist: Während $\bar{R}^{lern}$ weiter abnimmt, steigt $\bar{R}^{test}$ mit wachsendem K. In dem vorher erwähnten extremen Beispiel für Überanpassung müsste $\bar{R}^{lern}$ den Wert Null besitzen, während $\bar{R}^{test}$ einen sehr hohen Wert annehmen würde, was in Abbildung 22 nicht dargestellt ist.

Aus der in diesem Abschnitt erörterten Generalisierungsfähigkeit geht somit hervor, dass die Güte einer Lernmaschine, also ihr Vermögen das für einen bestimmten Sachverhalt Gelernte zu verallgemeinern, von der Anzahl der Parameter der implementierten Funktion $f(\vec{x},\theta)$ im Verhältnis zur Größe des verwendeten Lerndatensatzes abhängig ist. Es zeigt sich, dass es eine optimale Anzahl an Parametern für jedes Lernproblem und jede Funktion $f(\vec{x},\theta)$ gibt, die ein Maximum an Generalisierung gewährleistet. Jedoch ist es in der Praxis nicht so einfach, dieses Optimum zu finden, weshalb unterschiedliche Methoden zur Eingrenzung von Überanpassung verwendet werden.

3.2. Klassifikationsprobleme

Wie bereits erwähnt, spielen Klassifikationsprobleme für diese Arbeit eine wichtige Rolle, weshalb an dieser Stelle nochmals ausführlich auf sie eingegangen werden soll. Gegeben sei eine Menge von Vektoren $\vec{x}_i$ aus dem Vektorraum V_e. Dabei kann V_e ein beliebiger abstrakter Vektorraum sein, zum Beispiel der Raum aller alphabetischen Zeichenketten ei-

ner festen Länge oder der Raum aller Vektoren mit reellwertigen Komponenten und fester Länge. Weiterhin seien $N^{Klassen}$ disjunkte Mengen ζ_j definiert, sodass für jedes $\vec{x}_i$ eine eindeutige Zuweisung zu einer der Mengen ζ_j existiert. Die Mengen ζ_j werden in diesem Zusammenhang als *Klassen* bezeichnet, während die Zuweisung eines Vektors zu einer Klasse als *Klassifikation* bezeichnet wird. Mit diesem Hintergrund kann das Klassifikationsproblem definiert werden als die Suche nach einer Funktion $f(\vec{x},\theta)$ mit den Parametern θ, für die die Beziehung $\vec{x}_i \in \zeta_\alpha \Leftrightarrow f(\vec{x}_i,\theta)=\alpha$ (siehe auch Abschnitt 3.1.2.1) gilt. Das bedeutet, dass diejenige Funktion gesucht wird, die die Klassifikation der einzelnen Vektoren $\vec{x}_i$ möglichst genau nachbildet. Diese Funktion wird als *Klassifikator* bezeichnet.

3.2.1. Das Klassifikationsproblem als Regressionsproblem

In Abschnitt 3.1.2.1 wurde bereits eine Kostenfunktion vorgestellt, mit deren Hilfe das Klassifikationsproblem unter Verwendung der ERM gelöst werden kann. Diese besitzt jedoch den Nachteil, dass sie einen diskreten Wertebereich besitzt und damit nicht differenzierbar ist. Somit ist die Minimierung des Risikofunktionals unter Verwendung dieser Kostenfunktion mittels mathematischer Methoden nicht möglich. Es ist jedoch möglich, das Klassifikationsproblem auf ein Regressionsproblem zu übertragen, bei dem die Eingangsdaten $\vec{x}_i$ durch die Funktion $f(\vec{x},\theta)$ auf diskrete Werte y_i abgebildet werden. Diese repräsentieren die Indizes der einzelnen Klassen und können sowohl Skalare als auch Vektoren beliebiger aber fester Dimension sein, solange jede Klasse einen eindeutigen Index besitzt. Für diese Abwandlung der Definition des Klassifikationsproblems kann die quadratische Abweichung als Kostenfunktion (Abschnitt 3.1.2.2) für die ERM benutzt werden. Diese hat den Vorteil, dass die Kostenfunktion unter der Annahme dass $f(\vec{x},\theta)$ differenzierbar ist, selbst auch differenzierbar ist, wodurch die ERM mittels bekannter Optimierungsverfahren aus der Mathematik durchgeführt werden kann. Das Verfahren birgt jedoch zugleich alle Nachteile, die ein Regressionsverfahren mit sich bringen kann. So passiert es in der Regel, dass nach der Optimierung $f(\vec{x}_i,\theta)=y_i+\eta$ gilt, wobei η einen Rauschterm repräsentiert. $f(\vec{x},\theta)$ hat also in der Regel eine Abweichung vom Sollwert, was die Klassifikation über solch einfache Entscheidungsregeln wie $\vec{x}_i \in \zeta_j \Leftrightarrow f(\vec{x}_i,\theta)=j$ erschwert. Dieses Problem kann jedoch durch die Wahl geeigneter Entscheidungsregeln umgangen werden (siehe Abschnitt 3.4.2).

In der Praxis hat es sich durchgesetzt, die quadratische Abweichung als Kostenfunktion in der ERM zu verwenden. Viele bekannte Lernverfahren, wie beispielsweise neuronale Netze [45], verwenden die mittlere quadratische Abweichung als zu minimierende Funktion beim Lernvorgang.

3.2.2. Maximum Likelihood

Neben der bereits gegebenen Begründung für den mittleren quadratischen Fehler als zu optimierender Größe bei der Lösung des Klassifikationsproblems, gibt es noch eine zweite mögliche Begründung, die auf der aus der Statistik bekannten *Maximum-Likelihood-Methode* aufbaut. Ausgehend von einem Datensatz $D^{lern}=\{(\vec{x}_i, y_i)|1\leq i\leq N\}$ und einer parametrisierten Funktion $f(\vec{x},\theta)$, die den Zusammenhang $y_i=g(\vec{x}_i)$ nachbilden soll, sucht die Maximum-Likelihood-Methode nach demjenigen Parametersatz $\hat{\theta}$ der die folgende Funktion maximiert:

$$L(\theta)=\prod_{i=1}^{N} p(y_i|\vec{x}_i,\theta)$$

Diese Funktion wird als *Likelihood-Funktion* bezeichnet. Die Funktion $p(y|\vec{x},\theta)$ stellt die Wahrscheinlichkeit für die Beobachtung y, gegeben die Eingangsdaten $\vec{x}$ und den Parametersatz θ, dar. Die Methode geht im Prinzip davon aus, dass der Datensatz D^{lern} von der Funktion $f(\vec{x},\theta)$ mit unbekanntem Parametersatz θ_{opt} erzeugt wurde. Das Maximieren der Likelihood-Funktion hinsichtlich θ entspricht der Suche nach denjenigen Parametern $\hat{\theta}$, die die Existenz von D^{lern} am wahrscheinlichsten erscheinen lassen, für die also mit anderen Worten die einzelnen $p(y_i|\vec{x}_i,\theta)$ für alle $1\leq i\leq N$ möglichst nahe bei der Zahl Eins liegen. In der Regel wird statt der Maximierung von $L(\theta)$ die Minimierung der sogenannten *log-Likelihood-Funktion* vorgenommen, die den negativen natürlichen Logarithmus von $L(\theta)$ darstellt. Das Problem wird also in die Minimierung der folgenden Funktion hinsichtlich θ umgewandelt:

$$l(\theta)=-\ln\big(L(\theta)\big)=-\ln\left(\prod_{i=1}^{N} p(y_i|\vec{x}_i,\theta)\right)=-\sum_{i=1}^{N}\ln\big(p(y_i|\vec{x}_i,\theta)\big)$$

Die Maximum-Likelihood-Methode wird hauptsächlich in der statistischen Regression verwendet. Da sich das Klassifikationproblem auf ein Regressionsproblem übertragen lässt, kann die Methode auch in diesem Fall verwendet werden. Nun sei angenommen, dass die

Modellfunktion $f(\vec{x},\theta)$ mit Rauschen behaftet ist, also dass der wahre Zusammenhang $g(\vec{x})$ nicht exakt nachgebildet werden kann. Ausgehend von einem gaußförmigen Rauschen, kann die folgende Wahrscheinlichkeit definiert werden:

$$p(y|\vec{x},\theta)=\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(y-f(\vec{x},\theta))^2}{2\sigma^2}}$$

Dabei ist σ^2 die Varianz des Rauschens. Wird diese Wahrscheinlichkeit in $l(\theta)$ eingesetzt, so erhält man den folgenden Ausdruck:

$$l(\theta)=-\sum_{i=1}^{N}\ln\left(\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(y_i-f(\vec{x}_i,\theta))^2}{2\sigma^2}}\right)=\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-f(\vec{x}_i,\theta))^2+\ln\left(\sqrt{2\pi}\sigma\right)$$

Die Minimierung von $l(\theta)$ hinsichtlich θ entspricht der Minimierung der Funktion

$$E_1(\theta)=\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-f(\vec{x}_i,\theta))^2,$$

da der Term $\ln\left(\sqrt{2\pi}\sigma\right)$ nicht von θ abhängt und somit nicht berücksichtigt werden muss. Die Minimierung von $E_1(\theta)$ ist äquivalent zu der Minimierung von

$$E(\theta)=\frac{1}{2\mathrm{N}}\sum_{i=1}^{N}(y_i-f(\vec{x}_i,\theta))^2,$$

da die Konstante vor der Summe für das Ergebnis der Minimierung keine Rolle spielt. Die zu minimierende Funktion $E(\theta)$, die prinzipiell $R_{emp}(\theta)$ bei der ERM entspricht, soll im weiteren Text als *Fehlerfunktion* bezeichnet werden.

Die Interpretation des Klassifikationsproblems als Regressionsproblem und das Maximum-Likelihood-Verfahren legen die Minimierung des mittleren quadratischen Fehlers als Lösungsansatz nahe. Dieser Ausdruck hat gegenüber dem im Abschnitt 3.1.2.1 vorgestellten Ansatz den Vorteil differenzierbar zu sein, was seine Minimierung sehr erleichtert.

3.3. Das *Zweiklassenproblem*

Die einfachsten Klassifikationsprobleme sind *Zweiklassenprobleme*. Hierbei soll eine Menge von Eingangsdaten korrekt auf zwei Klassen aufgeteilt werden. Ausgehend von Eingangsdaten $\vec{x}_i\in\mathbb{R}^I$ kann die Funktion $f(\vec{x},\theta)$ in diesem Fall als *I*-dimensionale Trennfläche zwischen den Mitgliedern beider Klassen im Raum aufgefasst werden. Diese

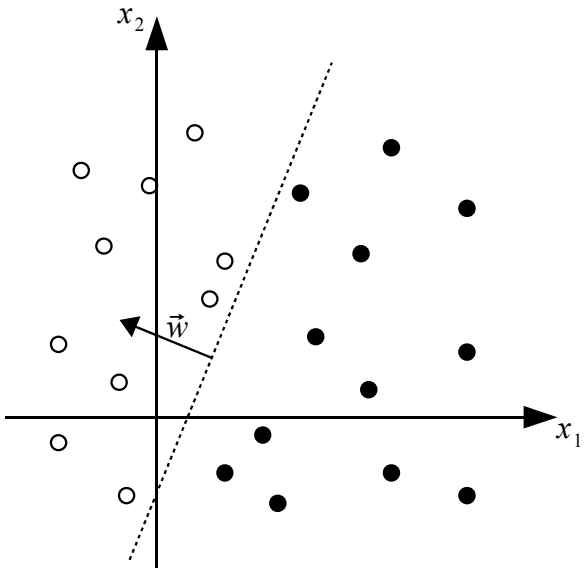


Abbildung 23: *Lineare Trennbarkeit zweier Klassen im zweidimensionalen Raum*

Fläche kann dabei theoretisch jede beliebige Form annehmen. Gesucht ist nun diejenige Trennfläche, die die Daten räumlich so in zwei Teilmengen teilt, dass diese räumlichen Mengen möglichst den ursprünglichen zwei Klassen entsprechen.

Wie gut die Daten sich im Einzelnen durch einen Klassifikator trennen lassen, wird durch die Eigenschaft der Trennbarkeit beschrieben. Diese ist eine Eigenschaft der Daten, die aussagt, wie leicht oder schwer sich die beiden in den Daten enthaltenen Klassen voneinander trennen lassen, was durch den Grad der Komplexität des notwendigen Klassifikators wiedergegeben wird. Beim Zweiklassenproblem werden im Allgemeinen zwei Fälle der Trennbarkeit unterschieden (*lineare* und *nichtlineare Trennbarkeit*), auf die im Folgenden kurz eingegangen werden soll.

3.3.1. Lineare Trennbarkeit

Lineare Trennbarkeit der Daten stellt den günstigsten Fall dar, der auftreten kann. Die Daten $\vec{x}_i \in \mathbb{R}^I$ sind derart im Raum verteilt, dass die Trennung beider Klassen mittels einer Hyperebene vorgenommen werden kann. In Abbildung 23 ist ein Beispiel linearer Trennbarkeit für den Fall $I=2$ dargestellt. Für die Klassifikation der Daten wird ein linearer Klassifikator der Form $f(\vec{x},\theta)=\vec{w}\cdot\vec{x}+b$ verwendet, wobei $\theta=(\vec{w},b)$, $\vec{w}\in\mathbb{R}^2$ und $b\in\mathbb{R}$. Die gestrichelte Linie in Abbildung 23 symbolisiert diejenige Gerade, an der die beiden Klassen voneinander getrennt werden, die sogenannte *Klassifikationsgrenze* (engl.

classification border). Zur Berechnung der Parameter eines solchen linearen Klassifikators für das Zweiklassenproblem mittels ERM benutzt man üblicherweise den mittleren quadratischen Fehler als Fehlerfunktion, wie es in den Abschnitten 3.2.1 und 3.2.2 beschrieben wurde. Dabei wird jeder der beiden Klassen ein bestimmter Zahlenwert zugewiesen, der dann für die erwarteten Ausgangsdaten y_i eingesetzt wird. Oftmals benutzt man für diese Klassenindizes die Werte $y=1$ und $y=-1$ oder $y=1$ und $y=0$.

Bei Verwendung der Klassenindices $y=1$ und $y=-1$ kann lässt sich die Klassifikationsgrenze in Abbildung 23, wie auch in höherdimensionalen Vektorräumen, als eine Ebene mit dem Normalenvektor $\vec{w}$ und dem Abstand b vom Koordinatenursprung interpretieren. Diese Interpretation lässt sich aus der Normalform einer Ebene $E:[\vec{x}-\vec{a}]\cdot\vec{w}=0$ mit Normalenvektor $\vec{w}$ und Aufpunkt $\vec{a}$ herleiten. Durch Ausmultiplizieren erhält man $[\vec{x}-\vec{a}]\cdot\vec{w}=\vec{x}\cdot\vec{w}-\vec{a}\cdot\vec{w}=0$, was durch das Ersetzen von $\vec{a}\cdot\vec{w}=-b$ zu $\vec{w}\cdot\vec{x}+b=0$ als Gleichung der Klassifikationsgrenze führt. Damit lässt sich die Gleichung des linearen Klassifikators $f(\vec{x},\theta)=\vec{w}\cdot\vec{x}+b$, für den Fall dass $|\vec{w}|=1$ gilt, als der Abstand eines Punktes $\vec{x}$ von der Klassifikationsgrenze interpretieren. Im Falle $|\vec{w}|\neq 1$ ist der Funktionswert von $f(\vec{x},\theta)$ proportional zum Abstand zwischen $\vec{x}$ und der Klassifikationsgrenze. In beiden Fällen gibt das Vorzeichen von $f(\vec{x},\theta)$ Aufschluss darüber, auf welcher Seite der Klassifikationsgrenze $\vec{x}$ liegt. Somit kann die Klassifikation von $\vec{x}$ über das Vorzeichen von $f(\vec{x},\theta)$ durchgeführt werden. Im Allgemeinen kann bei dieser Art der Klassifikation angenommen werden, dass der Betrag von $f(\vec{x},\theta)$ proportional zu der Wahrscheinlichkeit ist, dass $\vec{x}$ der bestimmten Klasse zugehörig ist. Da $f(\vec{x},\theta)$ proportional zum Abstand von $\vec{x}$ zur Klassifikationsgrenze ist, bedeutet ein größerer Betrag von $f(\vec{x},\theta)$ einen größeren Abstand zur Klassifikationsgrenze und somit potentiell eine sicherere Klassifikation.

3.3.2. Nichtlineare Trennbarkeit

Im Allgemeinen kann in der Praxis nicht davon ausgegangen werden, dass vorliegende Daten linear trennbar sind. Wie bereits erwähnt, stellt lineare Trennbarkeit den günstigsten der möglichen Fälle dar. Jedoch tritt es weitaus häufiger auf, dass die in realen Daten enthaltenen Klassen ausschließlich *nichtlinear trennbar* sind. Bei nichtlinearer Trennbarkeit besitzen die Daten also Eigenschaften, die eine Trennung nur mit Hilfe einer nichtlinearen Funktion ermöglichen. Ein solcher Fall ist für zweidimensionale Eingangsdaten in

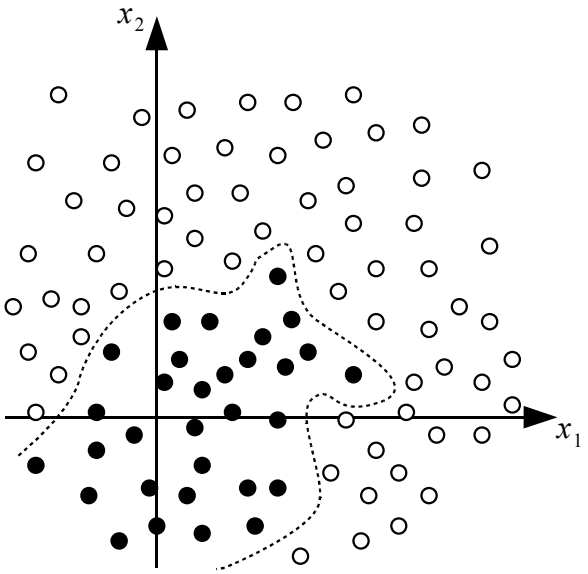


Abbildung 24: *Nichtlineare Trennbarkeit zweier Klassen im zweidimensionalen Raum*

Abbildung 24 dargestellt, in der wie in Abbildung 23 auf Seite 69 die Einteilung der Eingangsdaten in zwei Klassen vorhanden ist. Allerdings sind die Daten diesmal nicht mehr, wie in Abbildung 23 gezeigt, mit einer einfachen Geraden zu trennen. Die als gestrichelte Linie dargestellte Klassifikationsgrenze stellt die Niveaulinie $f(\vec{x},\theta)=0$ eines nichtlinearen Klassifikators dar. Wie auch im Fall der linearen Trennbarkeit kann hier für die Klassenindices $y=1$ und $y=-1$ die Klassifikation über das Vorzeichen von $f(\vec{x},\theta)$ vorgenommen werden, wobei der Betrag der Funktion grob als proportional zur Wahrscheinlichkeit, dass ein Vektor $\vec{x}$ zu einer bestimmten Klasse gehört, angesehen werden kann. Die Parameter eines nichtlinearen Klassifikators können wie auch schon beim linearen Klassifikator mittels ERM und des mittleren quadratischen Fehlers als Fehlerfunktion berechnet werden. Auch hier werden für die Klassenindizes oftmals die Werte $y=1$ und $y=-1$ oder $y=1$ und $y=0$ verwendet.

In der Regel ist die Chance zwei Klassen mit einem nichtlinearen Klassifikator zu trennen höher als es bei einem linearen Klassifikator der Fall ist. Weiterhin kann generell angenommen werden, dass die Trennung der Daten besser gelingt, wenn ein komplexerer der Klassifikator verwendet wird. Dabei wird die Komplexität von $f(\vec{x},\theta)$ mit der Anzahl ihrer Parameter korrelieren, wenn mit steigender Parameterzahl mehr nichtlineare Zusammenhänge zwischen den Eingangsdaten und dem Klassenindex berücksichtigt werden. Größere Komplexität von $f(\vec{x},\theta)$ kann aber auch, wie in Abschnitt 3.1.4 beschrieben,

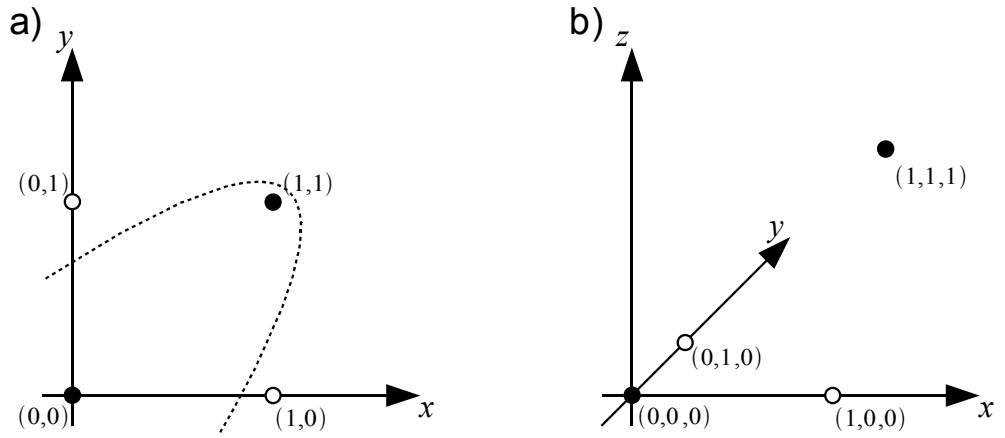


Abbildung 25: *Das XOR-Problem erfordert einen nichtlinearen Klassifikator (links), kann jedoch durch geeignete Transformation der Eingangsdaten in einen Raum höherer Dimension (rechts) auch mit einem linearen Klassifikator gelöst werden*

leicht zu einer Überanpassung führen.

Nichtlineare Trennbarkeit kann bereits bei einer geringen Datenmenge auftreten. Ein berühmtes Beispiel hierfür ist das sogenannte *Exklusiv-Oder-Problem* (*XOR-Problem* [45]). Obwohl dieses Problem recht einfach mit Hilfe der Aussagelogik gelöst werden kann, zeigt das Anwenden von Lernverfahren auf dieses Problem einerseits einen einfachen Fall der nichtlinearen Trennbarkeit, andererseits einen Fall, in dem das Problem bei einer geeigneten Darstellung der Eingangsdaten linear trennbar gemacht werden kann.

Der Klassifikation beim XOR-Problem liegt das Exklusiv-Oder aus der Aussagelogik zugrunde, also die Verknüpfung zweier Aussagen X_1 und X_2 über den Ausdruck $A(X_{1,}X_2)=\neg X_1\wedge X_2\vee X_1\wedge\neg X_2$. Der Ausdruck A ist somit genau dann wahr, wenn genau eine der beiden Aussagen X_1 oder X_2 wahr ist. Mit Hilfe des Exklusiv-Oders kann nun eine Klassifikation von Vektoren im zweidimensionalen Raum vorgenommen werden, die einen Vektor $\vec{x}=(x_{1,}x_2)$ mit $x_{1,}x_2\in\{0,1\}$ einer Klasse abhängig davon zuweist, ob der Ausdruck $A(x_{1,}x_2)=(x_1\neq 1)\wedge(x_2=1)\vee(x_1=1)\wedge(x_2\neq 1)$ wahr oder falsch ist. Die beiden Klassen entsprechen damit den Mengen $\zeta_1=\left\{\vec{x}\in B^2|x_1=x_2\right\}$ und $\zeta_2=\left\{\vec{x}\in B^2|x_1\neq x_2\right\}$ mit $B=\{0,1\}$. Eine solche Klasseneinteilung ist in Abbildung 25a dargestellt. Die beiden gefüllten Kreise symbolisieren die Klasse ζ_1, während die Mitglieder von ζ_2 durch leere Kreise dargestellt sind.

Abbildung 25a zeigt deutlich, dass es unmöglich ist eine Gerade in das Koordinatensystem

einzuzeichnen, die die beiden Klassen voneinander trennen würde. Es zeigt sich jedoch, dass bereits ein quadratischer Klassifikator ausreicht um eine korrekte Klassifikation vornehmen zu können. Dieser erzeugt eine paraboloide Klassifikationsgrenze, die durch die gestrichelte Linie dargestellt ist. Dies führt jedoch zu der bereits erwähnten Erhöhung der Parameterzahl gegenüber einem linearen Klassifikator und somit zu einem höheren Risiko zur Überanpassung.

Es existiert jedoch ein zweiter Ansatz zur Lösung des XOR-Problems, bei dem ein linearer Klassifikator Verwendung findet. Bei diesem Ansatz wird das Problem in ein dreidimensionales Koordinatensystem übertragen, wobei jeder Vektor zusätzlich zu den beiden Koordinaten x_1 und x_2 eine dritte Koordinate $x_3 = x_1 \cdot x_2$ besitzt. Dabei ist es wichtig, dass die zusätzliche Koordinate eine nichtlineare Kombination der ursprünglichen Koordinaten ist. Die daraus entstehende Verteilung der Datenvektoren ist in Abbildung 25b skizziert. Die so erweiterten Vektoren lassen sich durch die Ebene $x_3 = 0{,}5\,x_1 + 0{,}5\,x_2 - 0{,}25$ (diese entspricht der Klassifikationsgrenze des Klassifikators $f(\vec{x}, \theta) = -2x_1 - 2x_2 + 4x_3 + 1$) räumlich voneinander trennen, was das Problem zu einem linear trennbaren Problem macht. Im Allgemeinen kann davon ausgegangen werden, dass nichtlinear trennbare Probleme sich durch geeignete Vergrößerung der Dimension der Eingangsdaten auf linear trennbare Probleme reduzieren lassen. Jedoch führt eine Erhöhung der Dimension der Eingangsdaten zu zwei Problemen. Zum einen erhöht sich in der Regel die Anzahl der Parameter des Klassifikators mit steigender Dimension. Zwar ist diese Zunahme zumeist geringer als die Zunahme der Parameterzahl, die die Verwendung einer Funktion mit höherer Nichtlinearität nach sich ziehen würde, jedoch kann auch dies zu Überanpassung führen. Gleichzeitig führt eine Dimensionserhöhung aber auch zum so genannten *Fluch der Dimensionalität* (engl. *curse of dimensionality*) [46]. Dieser hängt damit zusammen, dass bei Erhöhung der Dimension eines Vektorraums das aufgespannte Volumen stark zunimmt, da es exponentiell von der Dimension abhängig ist. Somit wird es schwieriger statistische Zusammenhänge innerhalb der Daten herzuleiten, da in der Regel mehr Datenpunkte benötigt werden, um den gleichen Informationsgehalt für ein Volumen, das eine bestimmte Klasse einnimmt, zu erhalten, als es in einem Vektorraum kleinerer Dimension der Fall wäre.

Zusammenfassend kann man sagen, dass die Lösbarkeit nichtlinear trennbarer Probleme einerseits von dem gewählten Klassifikator, andererseits von der gewählten Darstellung

der Eingangsdaten abhängig ist. Dies schließt sowohl die hier beschriebene Dimension der Eingangsdaten als auch die Wahl eines geeigneten Vektorraumes, aus dem die Eingangsdaten gewonnen werden, ein. Letzteres entspricht einer geeigneten Auswahl an Messwerten oder Zustandsvariablen des beobachteten Systems, sodass schon bei kleiner Anzahl an Zustandsvariablen der Informationsgehalt der Eingangsdaten möglichst groß ist. Beide Möglichkeiten, also die Wahl eines geeigneten Klassifikators und eine geeignete Darstellung der Eingangsdaten, lassen sich kombinieren, jedoch sind dabei die Gefahr der Überanpassung und der Fluch der Dimensionalität begrenzende Faktoren für die Generalisierungsfähigkeit des verwendeten Klassifikators.

3.4. Das Mehrklassenproblem

Das Zweiklassenproblem stellt lediglich einen Sonderfall des allgemeinen Klassifikationsproblems dar. Es hat jedoch durchaus einen praktischen Bezug, da viele Klassifikationsprobleme Zweiklassenprobleme sind. Beispielsweise sind alle Probleme, bei denen sich die Frage stellt, ob reale Gebilde eine bestimmte Eigenschaft besitzen oder nicht, Zweiklassenprobleme. Beispiele hierfür wären:

- Können zwei gegebene molekulare Gruppen chemische eine Bindung eingehen?
- Ist ein auf einem Foto dargestelltes Gesicht männlich oder weiblich?
- Befindet sich innerhalb von akustischen Daten Gesang?

Weiterhin hat das Zweiklassenproblem den Vorteil, dass sich für dieses viele Konzepte und Probleme bei der Lösung von Klassifikationsproblemen einfach und anschaulich beschreiben lassen.

Im allgemeinen Fall des Klassifikationsproblems kann eine größere Anzahl von Klassen innerhalb der Daten vertreten sein, wie es in Abbildung 26 auf der folgenden Seite am Beispiel von vier Klassen dargestellt ist. Solche Klassifikationsprobleme mit mehr als zwei Klassen werden als *Mehrklassenprobleme* bezeichnet. Es stellt sich nun die Frage, wie sich ein solches Mehrklassenproblem lösen lässt.

Hierfür gibt es prinzipiell zwei Ansätze. Der erste wäre, eine Lernmaschine zu entwickeln, der es möglich ist Klassifikationsprobleme mit einer beliebigen Anzahl an Klassen zu lösen. Dazu müsste das Konzept der Lernmaschine derart allgemein gefasst sein, dass es auf

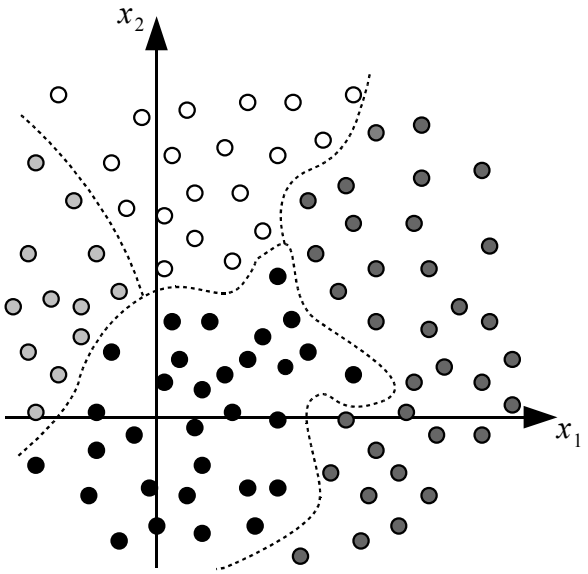


Abbildung 26: *Ein Vierklassenproblem als Beispiel für ein Mehrklassenproblem*

eine beliebige Anzahl an Klassen anwendbar ist. Dies ist zum Beispiel bei neuronalen Netzen der Fall, auf die in Abschnitt 3.5.4 eingegangen werden soll. Der zweite Ansatz wäre das Prinzip von „divide et impera“ (lat. für „teile und herrsche“), das ein sehr beliebtes Prinzip zur Problemlösung in der Informatik ist. Die Idee ist, ein komplexes Problem in eine Menge von Teilproblemen aufzuteilen, die alle für sich genommen mit einfachen Mitteln lösbar sind. Sobald alle Teilprobleme gelöst sind, kann daraus eine Lösung des ursprünglichen Problems konstruiert werden. Die Anwendung dieses Prinzips auf Mehrklassenprobleme würde bedeuten, dass das Mehrklassenproblem in eine Reihe von Zweiklassenproblemen unterteilt werden würde, welche jeweils einzeln gelöst werden würden. Für die Lösung von Zweiklassenproblemen stehen, wie in den vorhergehenden Abschnitten beschrieben, einfache Klassifikatoren zur Verfügung, die auch mit einfachen Mitteln berechnet werden können. Auf die Lösung des Mehrklassenproblems mit Hilfe von Klassifikatoren zweier Klassen wird im folgenden Abschnitt eingegangen.

3.4.1. Verwendung von Klassifikatoren zweier Klassen

Zunächst stellt sich die Frage, wie sich ein Mehrklassenproblem in eine Anzahl von Zweiklassenproblemen unterteilen lässt. Ausgehend von einem Datensatz $D^{lern} = \{\vec{x}_i | 1 \le i \le N\}$ und einer Menge an Klassen $Z = \{\zeta_j | 1 \le j \le N^{Klassen}\}$ mit einer eindeutigen und vollständigen Verteilung aller $\vec{x}_i$ auf alle ζ_α bestehen zwei Möglichkeiten, wie mit Hilfe der Men-

ge Z Zweiklassenprobleme gebildet werden können [47]. Dies wäre zunächst der Fall, in dem versucht wird jeweils zwei Klassen $\zeta_\alpha \in Z$ und $\zeta_\beta \in Z$ mit $\alpha \neq \beta$ voneinander zu trennen. Das heißt, die Klassifikatoren sind jeweils Funktionen $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$, welche die Klassen ζ_α und ζ_β voneinander trennen, beziehungsweise die eine Information darüber liefern sollen, ob $\vec{x}_i \in \zeta_\alpha$ oder $\vec{x}_i \in \zeta_\beta$ ist. Nun stellt sich die Frage, wie viele solcher Funktionen $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$ berechnet werden müssen, um alle möglichen Kombinationen von α und β abzudecken. Eine Antwort auf diese Frage liefert die Kombinatorik: Die Fragestellung entspricht der Suche nach der Anzahl an Kombinationen beim Ziehen zweier Elemente aus der Menge Z, wobei die Reihenfolge der Elemente außer Acht gelassen wird und ein Element nach dem Ziehen nicht zurückgelegt wird $(\alpha \neq \beta)$. Somit ergibt sich die Anzahl P der möglichen Paarungen von Klassen ζ_α und ζ_β, und damit auch die Anzahl der nötigen Funktionen $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$, als Kombination von zwei aus $N^{Klassen}$ Elementen wie folgt:

$$P = \binom{N^{Klassen}}{2} = \frac{N^{Klassen}!}{2! \cdot (N^{Klassen} - 2)!} = \frac{N^{Klassen} \cdot (N^{Klassen} - 1)}{2}$$

Die Anzahl der Zwei-Klassen-Klassifikatoren $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$ steigt also quadratisch mit der Anzahl der Klassen $N^{Klassen}$. Die beschriebene Zerlegung wird als *1-1-Ansatz* bezeichnet ([47], aus dem Englischen: *one-against-one*).

Die zweite Möglichkeit besteht darin, die Zweiklassenprobleme derart zu definieren, dass nun Klassifikatoren $f_\alpha(\vec{x}, \theta_\alpha)$ gesucht werden, die eine bestimmte Klasse $\zeta_\alpha \in Z$ vom Rest aller vorhandenen Klassen $\zeta_j \in Z \setminus \zeta_\alpha$ trennen. Die Fragestellung für ein gegebenes $\vec{x}_i \in D^{lern}$ ist in diesem Fall also, ob $\vec{x}_i \in \zeta_\alpha$ oder $\vec{x}_i \notin \zeta_\alpha$ ist. Mit der Menge Z lassen sich auf diese Weise insgesamt $N^{Klassen}$ Zweiklassenprobleme definieren, was auch der Anzahl der zu berechnenden Klassifikatoren $f_\alpha(\vec{x}, \theta_\alpha)$ entspricht. In diesem Fall nimmt die Anzahl der Klassifikatoren also linear mit der Anzahl an Klassen zu. Diese Art der Zerlegung wird als *1-r-Ansatz* ([47], aus dem Englischen: *one-against-rest*) bezeichnet.

In Abbildung 27 auf der folgenden Seite sind beide Ansätze für den Fall $\vec{x}_i \in \mathbb{R}^2$ und $N^{Klassen} = 3$ dargestellt. Die zweidimensionalen Vektoren $\vec{x}_i \in D^{lern}$ sind ihrer Zugehörigkeit zu den Klassen ζ_1, ζ_2 und ζ_3 entsprechend verschiedenfarbig dargestellt. Abbildung 27a zeigt den 1-1-Ansatz, mit den Klassifikationsgrenzen der Funktionen

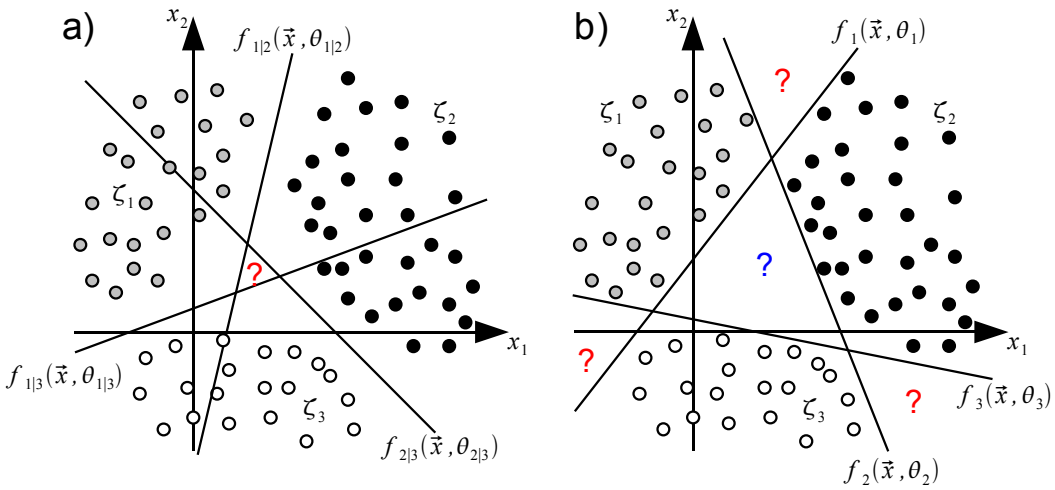


Abbildung 27: *Zerlegung eines Dreiklassenproblems in verschiedene Zweiklassenpronbleme: Links ist der 1-1-Ansatz dargestellt, nach dem jeder Klassifikator jeweils zwei Klassen voneinander trennt, rechts der 1-r-Ansatz, bei dem jeder Klassifikator eine Klasse von allen anderen trennt*

$f_{1|2}(\vec{x},\theta_{1|2})$, $f_{1|3}(\vec{x},\theta_{1|3})$ und $f_{2|3}(\vec{x},\theta_{2|3})$. Der Anschaulichkeit halber werden lineare Klassifikatoren angenommen, was zu den als durchgezogene Geraden dargestellten Klassifikationsgrenzen führt. Diese schließen in der Mitte der Abbildung ein Dreieck ein, das durch ein rotes Fragezeichen markiert ist. Für $\vec{x}$ aus diesem Dreieck würden die Klassifikatoren die Aussage $\vec{x}\in\zeta_1\wedge\vec{x}\in\zeta_2\wedge\vec{x}\in\zeta_3$ liefern, was bedeutet, dass der 1-1-Ansatz für diese $\vec{x}$ zu einem Konfliktfall führt. Abbildung 27b skizziert den 1-r-Ansatz, abermals unter Verwendung linearer Klassifikatoren $f_1(\vec{x},\theta_1)$, $f_2(\vec{x},\theta_2)$ und $f_3(\vec{x},\theta_3)$. Auch dieser Ansatz führt zu Konfliktfällen, die wieder mittels roter Fragezeichen markiert sind. In diesen Bereichen des Koordinatensystems liefern die Klassifikatoren Aussagen der Form $\vec{x}\in\zeta_\alpha\wedge\vec{x}\in\zeta_\beta$ mit $\alpha\neq\beta$, die keinen eindeutigen Schluss zulassen. Zusätzlich treten beim 1-r-Ansatz Fälle von Unbestimmtheit auf. Dies trifft in der Abbildung für das von den Klassifikationsgrenzen gebildete Dreieck zu, das durch ein blaues Fragezeichen gekennzeichnet ist. Für alle Vektoren $\vec{x}$ aus diesem Bereich des Koordinatensystems führen die Klassifikatoren zu der Aussage $\vec{x}\notin\zeta_1\wedge\vec{x}\notin\zeta_2\wedge\vec{x}\notin\zeta_3$. Während also in den Konfliktfällen (rote Fragezeichen) die Klassifikatoren noch die Zuteilung eines Vektors zu einer von zwei möglichen Klassen ermöglichen, ist im Falle der Unbestimmtheit keine Information vorhanden, die die Zuweisung eines Vektors zu einer bestimmten Klasse erleichtert.

Das Bemerkenswerte am dargestellten Dreiklassenproblem ist, dass sowohl der 1-1-Ansatz als auch der 1-r-Ansatz mit drei Klassifikatoren auskommt. Beide Ansätze besitzen in die-

sem Fall die gleiche Komplexität, was die Anzahl der notwendigen Zweiklassen-Klassifikatoren angeht. Das Beispiel zeigt auch, dass beide Ansätze eine Reihe von Klassifikatoren liefern, von denen jeder eine binäre Entscheidung darstellt. Diese Entscheidungen zusammengenommen charakterisieren das ursprüngliche Mehrklassenproblem und müssen lediglich in geeigneter Weise verarbeitet werden, um dieses lösen zu können. Wie jedoch bereits gezeigt wurde, treten Konfliktfälle und Fälle von Unbestimmtheit auf, die eine eindeutige Entscheidung erschweren. Auch dieses Problem kann mittels geeigneter Entscheidungsregeln behandelt werden. Auf die Wahl passender Entscheidungsregeln soll im folgenden Abschnitt eingegangen werden.

3.4.2. Entscheidungsregeln

Nachdem nun klar ist, wie man das Mehrklassenproblem in für sich lösbare Teilprobleme aufteilt, stellt sich die Frage, wie man die Lösungen dieser Teilprobleme verwenden kann um das ursprüngliche Problem zu lösen. Beim Zweiklassenproblem, wie es in Abschnitt 3.3 beschrieben worden ist, kann das Vorzeichen einer Klassifikatorfunktion $f(\vec{x},\theta)$ als Kriterium für die Zuweisung eines Vektors $\vec{x}$ zu einer der beiden Klassen verwendet werden. Zusätzlich kann der Betrag der Funktion als ein Indiz dafür benutzt werden, wie wahrscheinlich die Korrektheit der Zuweisung ist. Diese Tatsache kann auch beim Mehrklassenproblem Verwendung finden. Nach der Lösung der im Mehrklassenproblem eingebetteten Zweiklassenprobleme, wie sie in Abschnitt 3.4.1 beschrieben wurde, steht ein Satz von Klassifikatoren zur Verfügung, die jeweils ein Zweiklassenproblem lösen. Für jeden dieser Klassifikatoren kann die Klassifikatorfunktion mit Vorzeichen und Betrag berechnet werden, was eine Information darüber liefert, in welche der vorhandenen Klassen der betreffende Klassifikator einen gegebenen Vektor einteilt und mit welcher Wahrscheinlichkeit diese Einteilung korrekt ist. Dies ergibt zwei Möglichkeiten der Entscheidungsfindung.

Zunächst kann die Klassifikation ausschließlich mit Hilfe Vorzeichen der einzelnen Klassifikatoren ermittelt werden. Dies führt zu einer Art Abstimmungsverfahren, bei dem ein Vektor $\vec{x}$ derjenigen Klasse zugewiesen wird, die die meisten Stimmen erhält. Hierbei hat jeder Klassifikator eine Stimme, die vom Vorzeichen der jeweiligen Klassifikatorfunktion abhängig ist. Wie diese Stimme ausgewertet wird, hängt von dem jeweiligen Ansatz ab, mit dessen Hilfe die Klassifikatoren erzeugt wurden. Beim 1-1-Ansatz erhielte beispiels-

weise die Klasse ζ_α eine Stimme, wenn $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta}) > 0$, und die Klasse ζ_β eine Stimme, wenn $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta}) < 0$. Der Fall $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta}) = 0$ kann entweder als Grenzfall gesondert betrachtet werden (in dem Fall würde keine der beiden Klassen eine Stimme bekommen), oder aber mit einem der beiden vorher genannten Fälle vereint werden. Für den 1-r-Ansatz muss diese Entscheidungsregel etwas abgewandelt werden, da die einzelnen Klassifikatoren nicht zwei Klassen voneinander trennen, sondern eine Klasse von allen übrigen. Die Regel wäre in diesem Fall also, dass eine Klasse ζ_α eine Stimme bekommt, sofern $f_\alpha(\vec{x}, \theta_\alpha) > 0$ ist. Somit kann jede Klasse maximal eine Stimme bekommen, während beim 1-1-Ansatz maximal $N^{Klassen} - 1$ Stimmen möglich sind.

Die zweite Möglichkeit der Entscheidungsfindung benutzt hauptsächlich den Betrag der einzelnen Klassifikatorfunktionen, berücksichtigt dabei aber auch ihr Vorzeichen. Die Idee ist, einen Vektor $\vec{x}$ derjenigen Klasse zuzuordnen für die es am wahrscheinlichsten ist, dass diese Zuordnung korrekt ist. Am einfachsten ist dies im Falle des 1-r-Ansatzes möglich. Hier wird der Vektor $\vec{x}$ der Klasse ζ_α zugewiesen, wenn gilt:

$$\alpha = \arg\max_j f_j(\vec{x}, \theta_j)$$

In diesem Fall wird angenommen, dass $f_\alpha(\vec{x}, \theta_\alpha) > 0$ gleichbedeutend ist mit $\vec{x} \in \zeta_\alpha$. Somit ist ein möglichst großer Wert von $f_\alpha(\vec{x}, \theta_\alpha)$ mit der Wahrscheinlichkeit gleichzusetzen, dass der Vektor $\vec{x}$ der Klasse ζ_α angehört. Gilt nun für mehrere Klassen, dass die betreffende Klassifikatorfunktion größer Null ist, so würde die vielversprechendste Klassifikation diejenige sein, bei der die Wahrscheinlichkeit der Korrektheit am höchsten ist. Somit kommt die oben beschriebene Entscheidungsregel zustande, die im Folgenden als die *Regel der maximalen Konfidenz* (im Englischen oft auch als *majority rule*) bezeichnet werden soll. Für den 1-1-Ansatz sieht dieser Zusammenhang nun nicht mehr so einfach aus. Hier unterscheidet eine Klassifikatorfunktion $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$ zwischen den Klassen ζ_α und ζ_β. Unter der Annahme, dass $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta}) > 0$ mit $\vec{x} \in \zeta_\alpha$ gleichzusetzen ist und $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta}) < 0$ dem Fall $\vec{x} \in \zeta_\beta$ entspricht, kann man folgern, dass ein möglichst großer positiver Wert von $f_{\alpha|\beta}(\vec{x}, \theta_{\alpha|\beta})$ einer hohen Wahrscheinlichkeit von $\vec{x} \in \zeta_\alpha$ entspricht und ein möglichst kleiner negativer Wert einer hohen Wahrscheinlichkeit für $\vec{x} \in \zeta_\beta$ entspricht. Nun existieren aber jeweils $N^{Klassen} - 1$ Klassifikatoren, die eine Aussage darüber liefern, ob ein Vektor zu einer der Klassen ζ_α oder ζ_β gehören könnte. Um eine Aussage

für eine bestimmte Klasse treffen zu können müssen alle $N^{Klassen}-1$ Klassifikatoren berücksichtigt werden, was auf unterschiedliche Weise passieren kann. Eine Möglichkeit wäre es, die Summen der Beträge der Funktionswerte aller Klassifikatoren zu bilden, unter der Bedingung, dass das Vorzeichen der Klassifikatorfunktion die betreffende Klasse begünstigt. Damit lässt sich für jede Klasse ein Wert s_α definieren, der folgender Formel entspricht:

$$s_\alpha = \sum_{\substack{k=1 \\ k\neq\alpha}}^{N^{Klassen}} s_{\alpha k}, \quad s_{\alpha k} = \begin{cases} f_{\alpha|k}(\vec{x}, \theta_{\alpha|k}), & \exists f_{\alpha|k}(\vec{x}, \theta_{\alpha|k}) \wedge f_{\alpha|k}(\vec{x}, \theta_{\alpha|k}) > 0 \\ -f_{k|\alpha}(\vec{x}, \theta_{k|\alpha}), & \exists f_{k|\alpha}(\vec{x}, \theta_{k|\alpha}) \wedge f_{k|\alpha}(\vec{x}, \theta_{k|\alpha}) < 0 \\ 0, & \text{sonst} \end{cases}$$

Nach dieser Definition würde ein Vektor $\vec{x}$ der Klasse ζ_α zugewiesen werden, wenn gilt:

$$\alpha = \arg\max_j s_j$$

Die obigen Ausführungen zeigen, dass die Umsetzung eines Abstimmungsverfahrens sich eher für den 1-1-Ansatz anbietet, während der 1-r-Ansatz einfacher mit der Regel der maximalen Konfidenz kombinierbar ist. Die Regel der maximalen Konfidenz geht zudem davon aus, dass die einzelnen Klassifikatoren vergleichbar sind, also dass eine Art Normierung der Werte der einzelnen Klassifikatorfunktionen existiert, die diese untereinander vergleichbar macht. Weiterhin stellt sich die Frage, wie mit Konfliktfällen bzw. Fällen von Unbestimmtheit umgegangen werden soll. Konfliktfälle treten bei beiden Entscheidungsregeln auf und äußern sich beim Abstimmungsverfahren darin, dass mehrere Klassen die gleiche Anzahl an Stimmen haben, welche zugleich der maximalen Anzahl an Stimmen entspricht. Bei der Regel der maximalen Konfidenz äußern sich Konfliktfälle in dem recht unwahrscheinlichen Fall, dass die Klassifikatorfunktionen zweier oder mehrerer Klassifikatoren exakt den gleichen Wert besitzen, beziehungsweise, dass die s_k für mehrere Klassen den gleichen Wert haben, der dem maximalen s_k entspricht. In diesem Fall kann der betreffende Vektor zufällig einer der am Konflikt beteiligten Klassen zugewiesen werden [47]. Die Wahrscheinlichkeit der Zuweisung zu einer bestimmten Klasse kann dabei sowohl für alle Klassen gleich, als auch bei entsprechendem Vorwissen zu Gunsten bestimmter Klassen gewählt werden. Das Problem der Ungewissheit gestaltet sich etwas schwieriger. Es tritt bei beiden Entscheidungsregeln auf, jedoch ausschließlich im Falle des 1-r-Ansatzes, wenn alle $f_\alpha(\vec{x}, \theta_\alpha) < 0$ sind. Dies ist gleichbedeutend damit, dass $\vec{x}$ mittels der existierenden Klassifikatoren zu keiner der Klassen zugeordnet werden kann. Das würde

wiederum bedeuten, dass für den entsprechenden Vektor das Klassifikationsproblem nicht einmal ansatzweise gelöst wurde. Auch in diesem Fall bietet sich eine zufällige Klassifikation des betreffenden Vektors an, jedoch gibt es keine Beschränkung der in Frage kommenden Klassen auf einige wenige, sondern alle Klassen sind mögliche Kandidaten. Diese Tatsache erhöht die Wahrscheinlichkeit einer Fehlklassifikation immens und benötigt ein geschickteres Vorgehen, um diese zu minimieren.

3.5. Gängige Lernverfahren

Bevor in den folgenden Kapiteln auf die konkrete Anwendung von Lerntheorie und Lernverfahren eingegangen wird, sollen zunächst noch einige Beispiele gängiger Lernverfahren Erwähnung finden. Dies soll zeigen, welche Optionen zur Verfügung stehen um das in dieser Arbeit behandelte Problem der Sekundärstrukturvorhersage zu lösen. Zudem soll eine Unterscheidung von Lernverfahren diskutiert werden, auf die bisher nicht weiter eingegangen wurde. Diese erfolgt im folgenden Abschnitt.

3.5.1. Überwachtes und unüberwachtes Lernen

Im Allgemeinen gibt es eine Unterscheidung von Lernverfahren in zwei Kategorien, die der bisher beschriebenen Kategorisierung in Klassifikationsprobleme, Regressionsprobleme und Dichteschätzungsprobleme übergeordnet ist. Die bisherigen Spezifikation von Lernproblemen ging davon aus, dass für die Lösung eines Lernproblems ein Datensatz $D^{lern} = \{(\vec{x}_i, y_i) | 1 \leq i \leq N\}$ zur Verfügung steht, für den ein unbekannter Zusammenhang $y_i = g(\vec{x}_i)$ angenommen wird. Weiterhin existiert im Beobachter, wie in Abbildung 20 auf Seite 55 dargestellt, eine Instanz, der zu jedem $\vec{x}_i$ das entsprechende y_i bekannt ist und die damit Verlauf und Güte des Lernvorganges überwachen kann. Lernverfahren bei denen ein solcher Beobachter vorhanden ist werden deshalb als *überwachte Lernverfahren* bezeichnet.

Es existieren jedoch Problemstellungen, für die lediglich die Eingangsdaten bekannt sind, der Datensatz D^{lern} also die Gestalt $D^{lern} = \{\vec{x}_i | 1 \leq i \leq N\}$ besitzt. Solche Probleme wären beispielsweise Klassifikationsprobleme, bei denen die Klassenzugehörigkeit der einzelnen $\vec{x}_i \in D^{lern}$ nicht bekannt ist, jedoch von der Annahme ausgegangen wird, dass eine solche Klasseneinteilung möglich ist. Lernverfahren die eine derartige Problemstellung lösen wer-

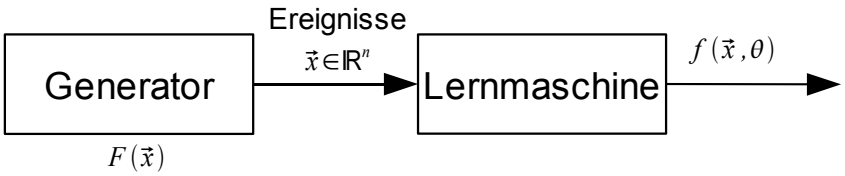


Abbildung 28: *Schematische Darstellung des unüberwachten Lernens*

den *unüberwachte Lernverfahren* genannt. Bei unüberwachten Lernverfahren existiert entsprechend kein Beobachter, der den Lernvorgang überwacht und bewertet. Die Lernmaschine extrahiert vielmehr selbstständig statistische Regelmäßigkeiten aus den Daten, die dann entsprechend der Aufgabenstellung verwertet werden können. Eine schematische Darstellung des unüberwachten Lernens ist in Abbildung 28 skizziert. Das jeweilige Gütemaß, das bei unüberwachten Lernverfahren optimiert wird, ist nicht problemspezifisch und hängt in der Regel nur vom verwendeten Lernverfahren ab [45].

Ein Beispiel für unüberwachtes Lernen stellen *Clusteralgorithmen* dar. Bei diesen wird davon ausgegangen, dass die Daten in eine Zahl unbekannter Klassen unterteilt werden können, die in diesem Fall als Cluster bezeichnet werden. Der Lernvorgang hat zum Ziel, die Zuweisung der einzelnen Daten zu den Clustern einzig Anhand statistischer Ähnlichkeiten der Eingangsdaten zu erlernen und für jeden Cluster einen Prototypen zu generieren. Dieser ermöglicht eine spätere Zuweisung unbekannter Daten zu den einzelnen Clustern. Je nach Verfahren kann die Anzahl der Cluster während des Lernverfahrens ein fest vorgegebener Wert (beispielsweise beim k-means-Clustering) oder eine variable Größe sein, die während des Lernverfahrens optimiert wird.

Im Rahmen dieser Arbeit wird nicht weiter auf unüberwachte Lernverfahren eingegangen, da sich bei der gegebenen Problemstellung überwachte Verfahren anbieten. Das Problem der Sekundärstrukturvorhersage geht von definierten Eingangsdaten (Primärstruktur) und dazugehörigen definierten Ausgangsdaten (Sekundärstruktur) aus. Es existiert also bereits eine Menge an Daten $D^{lern} = \{(\vec{x}_i, y_i) | 1 \leq i \leq N\}$, welche für ein überwachtes Lernverfahren verwendet werden kann. Aus diesem Grund liegt das Hauptaugenmerk dieser Arbeit auf den überwachten Lernverfahren.

3.5.2. Lineare Diskriminanzanalyse

Die lineare Diskriminanzanalyse wurde erstmals von Ronald A. Fisher in seinem Artikel von 1936 [41] erwähnt, wenn auch nicht unter diesem Namen. In diesem Artikel behandelt Fisher die Frage, welche Bedingung eine Linearkombination von Messwerten, die für zwei verschiedene Populationen vorhanden sind, erfüllen muss um Vertreter beider Populationen voneinander trennen zu können. Dabei wird das Beispiel von Schwertlilien genommen, welche zusammen in einer Kolonie wachsen. Für diese beiden Populationen wurden vier Messwerte x_1, x_2, x_3 und x_4 gesammelt, sodass ein Datensatz von Vektoren $\vec{x} \in \mathbb{R}^4$ vorhanden ist. Nun ergibt sich die Fragestellung, wie anhand dieser Messwerte Individuen beider Populationen, beziehungsweise Klassen, korrekt identifiziert werden können. Um dieses Problem zu lösen schlägt Fisher in [41] eine lineare Funktion der Form $f(\vec{x}) = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 = \vec{\lambda}^T \cdot \vec{x}$ vor. Die grafische Interpretation dieser Formel ist neben der bereits genannten Ebenengleichung (Abschnitt 3.3.1) die, dass das vierdimensionale Problem mittels einer geeigneten Projektion auf ein eindimensionales Problem reduziert wird, welches einfacher zu lösen ist. Die einzelnen Vektoren $\vec{x}$ werden somit auf eine Gerade durch den Koordinatenursprung projiziert, die den Richtungsvektor $\vec{\lambda}$ besitzt. Nun stellt sich die Frage, wie der Vektor $\vec{\lambda}$ gewählt werden muss, damit Individuen zuverlässig den einzelnen Populationen zugewiesen werden können. Dazu müssen zunächst aus den vorhandenen Messwerten einige Größen berechnet werden. Dies wären zunächst die Mittelwerte der Messdaten für die beiden Populationen. Daraus ergibt sich für die Population ζ_1 ein Vektor mittlerer Messwerte $\vec{\mu}_1$ und für die Population ζ_2 der Mittelwert $\vec{\mu}_2$. Zudem werden noch die Kovarianzmatrizen Σ_1 und Σ_2 für die Messwerte beider Populationen gemessen. Beides resultiert aus der Annahme, dass beide Populationen normalverteilt sind. Im nächsten Schritt definiert Fisher zwei Größen, nämlich zum einen

$$D = \vec{\lambda}^T \cdot (\vec{\mu}_1 - \vec{\mu}_2) = \sum_{i=1}^{4} \lambda_i (\mu_{1i} - \mu_{2i}) = \sum_{i=1}^{4} \lambda_i d_i = \vec{\lambda} \cdot \vec{d},$$

was dem Abstand der Projektionen der Mittelwerte für beide Populationen entspricht, zum anderen die Größe

$$S = \vec{\lambda}^T \cdot \Sigma \cdot \vec{\lambda} = \sum_{i=1}^{4} \sum_{j=1}^{4} \lambda_i \lambda_j \Sigma_{ij} \text{ mit } \Sigma = \Sigma_1 + \Sigma_2,$$

die zur Gesamtvarianz der projizierten Daten proportional ist. Mit diesen Definitionen

kann D nachrichtentheoretisch als Nutzsignal angesehen werden, welches maximiert werden sollte, da es im Interesse der Problemlösung ist in der Projektion einen möglichst großen Abstand zwischen die beiden Populationen zu bringen. Weiterhin kann S als ein Rauschsignal interpretiert werden, das es zu minimieren gilt, da ein großer Wert von S die Chance erhöht, dass beide Populationen in der Projektion überlappen. Somit kann also das Signal-Rausch-Verhältnis, welches es zu maximieren gilt, folgendermaßen definiert werden:

$$\frac{D^2}{S} = \frac{\left(\sum_{i=1}^{4} \lambda_i d_i\right)^2}{\sum_{i=1}^{4}\sum_{j=1}^{4} \lambda_i \lambda_j \Sigma_{ij}}$$

Nun kann für jedes λ_i die partielle Ableitung der oberen Größe berechnet und für die Maximierung gleich Null gesetzt werden, was zur folgenden Gleichung führt:

$$\frac{\partial}{\partial \lambda_i}\frac{D^2}{S} = \frac{2\mathrm{D}\cdot\frac{\partial D}{\partial \lambda_i}\cdot S - D\cdot\frac{\partial S}{\partial \lambda_i}}{S^2} = \frac{D}{S^2}\left[2\mathrm{S}\cdot\frac{\partial D}{\partial \lambda_i} - D\cdot\frac{\partial S}{\partial \lambda_i}\right] = 0$$

Dies ergibt nach Umformung die Gleichung

$$\frac{S}{D}\cdot\frac{\partial D}{\partial \lambda_i} = \frac{1}{2}\frac{\partial S}{\partial \lambda_i} \quad \text{bzw.} \quad \frac{S}{D}\cdot d_i = \frac{1}{2}\cdot\sum_{j=1}^{4} \lambda_j \Sigma_{ij}$$

Für die Berechnung von $\vec{\lambda}_i$ ergibt sich also ein Gleichungssystem der Form

$$\frac{D}{2\mathrm{S}}\Sigma\vec{\lambda} = \vec{d}$$

Dieses kann mit $a = \frac{D}{2\mathrm{S}}$, welches hinsichtlich der Parameter $\vec{\lambda}$ konstant ist und somit bei der Maximierung keine Rolle spielt, und $\vec{w} = a\vec{\lambda}$ auf die vereinfachte Form $\Sigma\vec{w} = \vec{d} \Leftrightarrow \vec{w} = \Sigma^{-1}\vec{d}$ reduziert werden. Dieses Gleichungssystem liefert also einen Satz von Parametern $\vec{w}$ für eine lineare Funktion $f(\vec{x}) = \vec{w}^T\vec{x}$. Mit Hilfe dieser Funktion können nun Individuen der beiden Populationen identifiziert oder klassifiziert werden, indem Individuen mit $f(\vec{x}) > 0$ als der Population ζ_1 zugehörig klassifiziert werden, während $f(\vec{x}) < 0$ für die Population ζ_2 steht.

Somit stellt die sogenannte *Fisher'sche Diskriminanzfunktion* $f(\vec{x}) = \vec{w}^T\vec{x}$ einen einfachen linearen Klassifikator für zwei Klassen dar, wie er bereits in Abschnitt 3.3.1 vorgestellt

wurde. Bemerkenswert dabei ist der nachrichtentheoretische Ansatz zur Berechnung der Parameter $\vec{w}$. Weiterhin sei an dieser Stelle gesagt, dass Fishers Berechnungen zwar auf vierdimensionalen Eingangsdaten beruhen, wie aber leicht ersichtlich ist, auf eine beliebige Dimension der Eingangsdaten anwendbar sind. Lediglich die Komplexität des linearen Gleichungssystems nimmt zu.

3.5.3. Der naive Bayes-Klassifikator

Der *naive Bayes-Klassifikator* ist ein Klassifikator, welcher auf einfachen Prinzipien der Wahrscheinlichkeitsrechnung beruht. Ausgegangen wird von einem Datensatz von Beobachtungen $B=\left\{\vec{x}_i|\vec{x}_i=\left(x_{i1},x_{i2},\ldots,x_{in}\right)^T\right\}$, wobei x_{ij} als Merkmal bezeichnet wird. Jedes Merkmal x_{ij} stammt aus einer eigenen Grundmenge G_j, welche sowohl diskret (z. B. Farben, Blutgruppen, Geschlechter oder Berufsgruppen) wie auch kontinuierlich (z. B. Körpergrößen, Gewichte, Geschwindigkeiten oder Zeitdauern) sein kann. Somit ergibt sich aus den Merkmalen $x_{ij}\in G_j$ die Beobachtung als Vektor $\vec{x}_i\in G_1\times G_2\times\ldots\times G_n$. Der Bayes-Klassifikator ist in der Lage eine solche Beobachtung $\vec{x}_i$ einer von beliebig vielen Klassen ζ_k zuzuordnen, indem für jede Klasse eine Wahrscheinlichkeit $p(\zeta_k|\vec{x}_i)$ berechnet wird, die aussagt wie wahrscheinlich die entsprechende Klasse erscheint wenn die betreffende Beobachtung gegeben ist. Die Beobachtung wird derjenigen Klasse zugeordnet für die diese Wahrscheinlichkeit maximal ist.

Die Berechnung von $p(\zeta_k|\vec{x}_i)$ geschieht über den Satz von Bayes, der dem Klassifikator seinen Namen verleiht. Zur Erklärung dieses Satzes soll zunächst vom Klassifikationsproblem abstrahiert werden. Gegeben sei eine Hypothese H (z. B. die Zugehörigkeit zu einer Klasse ζ_k) und eine Beobachtung X (in diesem Fall eine der Beobachtungen $\vec{x}_i$). Es soll untersucht werden, inwiefern die Hypothese H als korrekt angesehen werden kann, wenn die Beobachtung X in Betracht gezogen wird. Nun sei zu jeder Hypothese die Wahrscheinlichkeit $p(H)$ gegeben, die aussagt, wie wahrscheinlich die Hypothese H unter den existierenden Hypothesen ist, ohne dass weitere Betrachtungen durchgeführt werden oder die Bedeutung der Beobachtung X für die Richtigkeit der Hypothese berücksichtigt wird. Diese Wahrscheinlichkeit wird als *A-priori-Wahrscheinlichkeit* (engl. *prior*) bezeichnet. Eine Beobachtung X hat ihrerseits eine A-priori-Wahrscheinlichkeit $p(X)$, welche im Englischen als *evidence*, also Beweis oder Beleg, bezeichnet wird. Schließlich sei be-

kannt, wie wahrscheinlich das Auftreten der Beobachtung X laut der Hypothese H ist. Diese Wahrscheinlichkeit $p(X|H)$ wird im Englischen als *likelihood* bezeichnet. Mit diesen drei Wahrscheinlichkeiten kann nun die Wahrscheinlichkeit der Korrektheit der Hypothese H mit Hilfe des Satzes von Bayes berechnet werden zu:

$$p(H|X) = \frac{p(H) \cdot p(X|H)}{p(X)}$$

In diesem Zusammenhang wird $p(H|X)$ als *A-posteriori-Wahrscheinlichkeit* (engl. *posterior*) bezeichnet, welche charakterisiert wie sinnvoll die Hypothese in Anbetracht der Beobachtung X gewählt ist. Die Übertragung des Satzes von Bayes auf das Klassifikationsproblem liefert zu jeder Klasse eine A-posteriori-Wahrscheinlichkeit:

$$p(\zeta_k|\vec{x}_i) = \frac{p(\zeta_k) \cdot p(\vec{x}_i|\zeta_k)}{p(\vec{x}_i)}$$

Die Entscheidungsregel, welcher Klasse $\vec{x}_i$ zugewiesen wird, wird als *Maximum-A-Posteriori* bezeichnet, da die Beobachtung der Klasse mit der maximalen A-posteriori-Wahrscheinlichkeit zugeordnet wird. Diese Entscheidungsregel entspricht der Regel der maximalen Konfidenz, welche in Abschnitt 3.4.2 bereits erwähnt wurde.

Die obige Gleichung zur Berechnung der A-posteriori-Wahrscheinlichkeiten ist noch nicht vollständig, da ihr noch die naive Annahme fehlt, welche dem naiven Bayes-Klassifikator seinen Namen gibt. Diese Annahme besagt, dass die einzelnen Merkmale (engl. *features*) einer Beobachtung statistisch voneinander unabhängig sind, also dass gilt:

$$p(\vec{x}_i|\zeta_k) = \prod_{j=1}^{n} p(x_{ij}|\zeta_k)$$

Mit dieser Annahme lässt sich die Gleichung der A-posteriori-Wahrscheinlichkeit umformen zu:

$$p(\zeta_k|\vec{x}_i) = \frac{p(\zeta_k) \cdot \prod_{j=1}^{n} p(x_{ij}|\zeta_k)}{p(\vec{x}_i)}$$

Der Lernvorgang für einen Bayes-Klassifikator besteht in der Berechnung der einzelnen in der Gleichung für die $p(\zeta_k|\vec{x}_i)$ auftretenden Wahrscheinlichkeiten. Im Falle diskreter Merkmale kann dies über die relativen Häufigkeiten der einzelnen Werte, die die Merk-

male auf dem Lerndatensatz annehmen, durchgeführt werden. Diese Berechnung wird durch die Annahme der statistischen Unabhängigkeit der einzelnen Merkmale stark vereinfacht. Im Falle kontinuierlicher Merkmale können diese entweder künstlich diskretisiert werden oder über ein geeignetes Modell für eine Wahrscheinlichkeitsverteilung beschrieben werden. Die Wahrscheinlichkeit $p(\vec{x}_i)$ wird in der Regel vernachlässigt, da dieser Term für alle $p(\zeta_k|\vec{x}_i)$ gleich ist und somit für den Vergleich der A-posteriori-Wahrscheinlichkeiten untereinander keine Rolle spielt.

Die Annahme, dass die einzelnen Merkmale der zu klassifizierenden Objekte voneinander statistisch unabhängig sind, besitzt in der Praxis nur bedingt Gültigkeit. In der Regel sind die einzelnen Merkmale mehr oder weniger stark korreliert, was die Berechnung der A-posteriori-Wahrscheinlichkeit nach der obigen Methode ungenau macht. Trotz allem zeigt sich, dass naive Bayes-Klassifikatoren sich gut auf viele real auftretende Probleme anwenden lassen. So werden naive Bayes-Klassifikatoren erfolgreich zur Lösung des Problems der Dokumentklassifikation, insbesondere der Filterung von Spammails ([48], [49]), eingesetzt. Weiterhin konnte gezeigt werden, dass der naive Bayes-Klassifikator auf Probleme angewendet werden kann, bei denen eine funktionale Abhängigkeit der Merkmale untereinander existiert [50]. Hanson et al. [51] konnten zeigen, dass sich das Konzept naiver Bayes-Klassifikatoren auch auf unüberwachte Lernprobleme anwenden lässt, bei denen weder die Anzahl der in den Daten vertretenen Klassen noch die Zugehörigkeit der Daten zu diesen bekannt ist. Somit finden naive Bayes-Klassifikatoren trotz ihrer Einfachheit heute in vielen Bereichen Anwendung und sind auch aktuell noch Gegenstand weiterer Forschung.

3.5.4. Künstliche neuronale Netze

Künstliche neuronale Netze (kurz KNN) sind Lernmaschinen, die vom Aufbau des Nervensystems, vor allem des Gehirns, von Lebewesen inspiriert sind. Das Gehirn ist eine komplexe Struktur, bestehend aus in ihrer Funktion relativ einfach operierenden Nervenzellen, die durch eine hochgradige Vernetzung untereinander und durch die Fähigkeit der Adaption die unangefochtene Überlegenheit des Gehirns gegenüber künstlich hergestellten Geräten in vielen Bereichen begründet. So kann beispielsweise das aufrechte Gehen auf zwei Beinen als ein komplexer dynamischer Regelungsvorgang [52] angesehen werden, welcher ein hohes Maß an Rechenleistung erfordert und welcher im Gehirn als Prozess im Hinter-

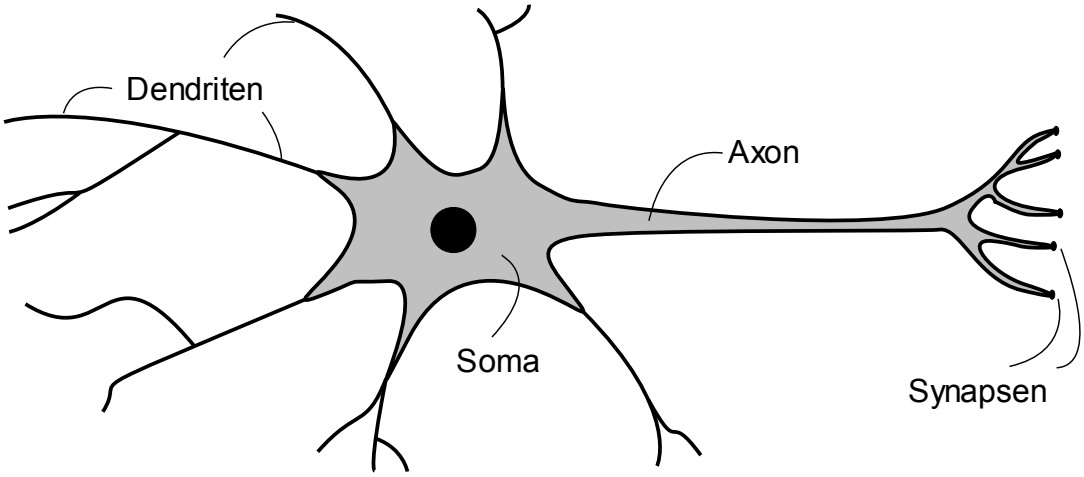


Abbildung 29: *Schematischer Aufbau eines biologischen Neurons*

grund abläuft, ohne dass das betreffende Lebewesen bewusst Ressourcen in dessen Steuerung investieren muss. Ein weiteres Beispiel ist die Mustererkennung, für die das menschliche Gehirn prädestiniert ist, welche aber auch Gegenstand langjähriger Forschungen im Bereich der Informatik und des maschinellen Lernens ist. Die Leistungsfähigkeit des Gehirns bewog Frank Rosenblatt im Jahre 1958 zu der Entwicklung eines elementaren Modells (Des so genannten *Perzeptrons* [53]) für die Informationsverarbeitung in Nervenzellen, auf dessen Grundlage sich in den folgenden Jahren die Theorie neuronaler Netze entwickelte.

3.5.4.1. Das Perzeptron

Das Perzeptron orientiert sich in Aufbau und Funktionsweise an einer einzelnen Nervenzelle. Eine Nervenzelle kann grob in drei Teile unterteilt werden, welche die drei Phasen der Informationsverarbeitung innerhalb der Nervenzelle repräsentieren. Diese Phasen sind das Sammeln von Informationen, die eigentliche Informationsverarbeitung und die Weiterleitung der verarbeiteten Information. Der schematische Aufbau einer Nervenzelle ist in Abbildung 29 dargestellt. Für das Sammeln von Informationen sind die *Dendriten* zuständig. Diese wurzelartigen Fortsätze bilden die eingehende Verbindung einer Nervenzelle zu anderen Nervenzellen. Über die Dendriten empfängt eine Nervenzelle Informationen von anderen Nervenzellen in Form kurzer Spannungsimpulse, wobei angenommen wird, dass die Information unter anderem in der Impulsrate kodiert ist [54]. Die über die Dendriten eintreffenden Spannungsimpulse werden im Zellkörper, dem *Soma,* verarbeitet und schließlich über das so genannte *Axon* weitergeleitet. Die Übertragung von Nervenimpul-

sen zwischen dem Axon einer Nervenzelle und den Dendriten einer anderen Nervenzelle findet über die *Synapsen* statt.

Die Verarbeitung der eintreffenden Information im Soma erfolgt durch zeitliche Integration der eintreffenden Spannungsimpulse. Die Membran des Soma ist isolierend, enthält jedoch Ionenkanäle und Natrium-Kalium-Pumpen, die einen Teilchen- und Ladungsaustausch zwischen Zellflüssigkeit und Außenmedium ermöglichen. Ionenkanäle sind ionenspezifisch und transportieren Ionen einer bestimmten Sorte mittels Diffusion durch die Membran. Sie tragen also zu einem Ausgleich des Konzentrationsgefälles der einzelnen Ionen bei. Dies führt auch zu einem Abbau des Ladungsunterschiedes zwischen Zellflüssigkeit und Außenmedium und damit zu einer kontinuierlichen Abnahme des über der Membran anliegenden Potentials. Die Natrium-Kalium-Pumpen hingegen, stellen eine aktive Form des Stofftransports entgegen dem Konzentrationsgefälle dar. Unter Aufwendung von Energie werden dabei Ladungsträger in das Zellinnere transportiert, was zu einer Erhöhung des Membranpotentials beiträgt. Die für den Transport notwendige Energie kommt indirekt von den in den Dendriten eintreffenden Spannungsimpulsen. Die Natrium-Kalium-Pumpen sind ligandengesteuert, was bedeutet, dass bestimmte Substanzen an Rezeptoren an diesen Pumpen andocken müssen, um den Stofftransport zu ermöglichen. Diese Substanzen werden in kleinen Mengen in den Synapsen freigesetzt, sobald ein Spannungsimpuls in diesen ankommt, und beim Stofftransport durch die Pumpen verbraucht. Somit erzeugt jeder Spannungsimpuls, der in den Dendriten empfangen wird, eine sowohl zeitlich wie auch vom Betrag her begrenzte Zunahme des Membranpotentials. Es ist jedoch ersichtlich, dass die Frequenz der eintreffenden Spannungsimpulse proportional zum zeitlichen Anstieg des Membranpotentials ist. Das Soma integriert also, wie bereits erwähnt, die eintreffenden Spannungsimpulse über die Zeit. Erreicht das Membranpotential einen bestimmten Schwellenwert, so wird ein sogenanntes *Aktionspotential* erzeugt, das heißt das Membranpotential steigt innerhalb kürzester Zeit stark an und wird daraufhin in ebenso kurzer Zeit abgebaut. Dies führt zu der Erzeugung eines Spannungsimpulses, welcher über das Axon an andere Nervenzellen weitergeleitet wird. Nach einem Aktionspotential nimmt das Membranpotential seinen Ausgangswert, das *Ruhepotential,* an, welcher abermals durch eintreffende Spannungsimpulse erhöht werden kann. An dieser Stelle soll angemerkt werden, dass die Vorgänge in der Membran und der zeitliche Verlauf des Membranpotentials in diesem Abschnitt lediglich skizziert wurden. Die realen Prozesse inner-

halb der Membran sind wesentlich komplexer und würden den Rahmen dieser Arbeit sprengen.

Im Wesentlichen hängt die Frequenz, mit der Aktionspotentiale erzeugt werden, von drei Größen ab. Dies wären die Frequenz, mit der Spannungsimpulse in den Dendriten eintreffen, die Größe des Schwellenwertes, den das Membranpotential überschreiten muss bevor ein Aktionspotential entsteht, und schließlich die synaptische Kopplung. Die letzte Größe beschreibt, wie stark das Membranpotential verändert wird, wenn ein Impuls über die entsprechende Synapse in die Dendriten gelangt. Dabei sei erwähnt, dass die bisherige Beschreibung des Einflusses einer Synapse auf das Membranpotential die sogenannte *exzitatorische Kopplung* ist. Bei dieser Art der synaptischen Kopplung erzeugt die Synapse für jeden eintreffenden Spannungsimpuls eine geringfügige Zunahme des Membranpotentials. Es existiert jedoch eine zweite Art der synaptischen Kopplung, welche als *inhibitorische Kopplung* bezeichnet wird. Im Falle einer inhibitorischen Kopplung trägt ein von der betreffenden Synapse empfangener Spannungsimpuls zu einer Senkung des Membranpotentials und somit zur Verhinderung oder Verzögerung eines Aktionspotentials bei. Das von Rosenblatt geschaffene Perzeptron stellt ein sehr vereinfachtes Modell der in einer Nervenzelle stattfindenden Vorgänge dar, welches die Funktion einer Nervenzelle auf die drei Größen Impulsrate, Schwellenpotential und synaptische Kopplung und deren Zusammenspiel reduziert.

In Abbildung 30 auf der folgenden Seite ist der schematische Aufbau des Perzeptrons im Vergleich zur Nervenzelle dargestellt. Die in das Perzeptron eingehenden reellen Werte x_i im linken Bereich der Abbildung können mit den Aktivitäten anderer Neuronen gleichgesetzt werden. Dabei muss es sich für die Anwendung nicht zwangsläufig um positive Zahlenwerte handeln, welche direkt in Raten oder Frequenzen umsetzbar sein müssen. Jeder dieser Eingangswerte x_i ist mit einem reellen Gewicht w_i versehen, das die synaptische Kopplung zu der betreffenden Aktivität x_i modelliert. Im Allgemeinen kann man bei positiven w_i von exzitatorischer, bei negativen w_i von inhibitorischer Kopplung sprechen. Somit bilden x_i und w_i die Dendriten der Nervenzelle und die eingehenden synaptischen Verbindungen zu anderen Nervenzellen nach. Das Soma wird durch zwei aufeinanderfolgende Operationen nachgebildet. Zunächst wird die Summe der eingehenden Aktivitäten gewichtet mit der synaptischen Kopplung gebildet, was den Vorgang der zeitlichen Inte-

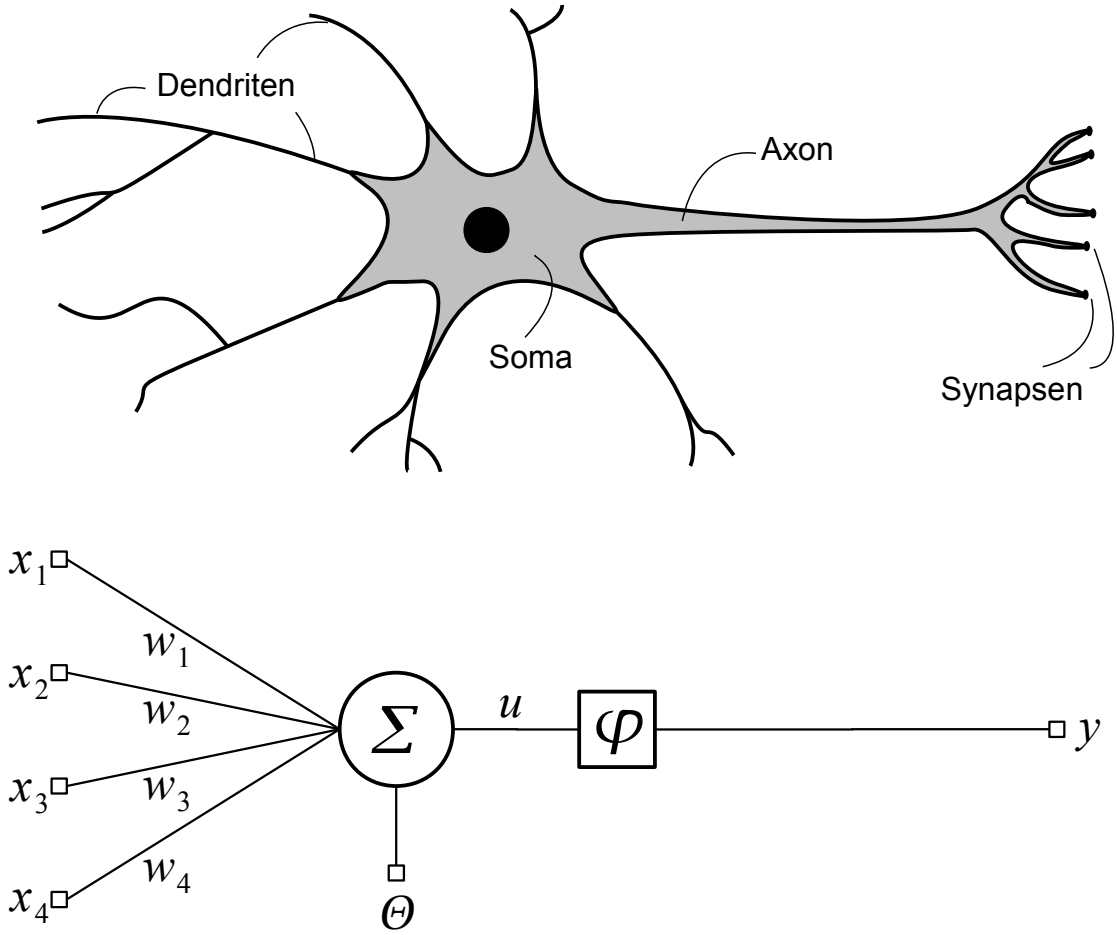


Abbildung 30: *Biologisches (oben) und künstliches Neuron (unten) im Vergleich*

gration der eingehenden Aktivitäten modelliert. Von der so entstanden gewichteten Summe wird ein konstanter Wert Θ subtrahiert, sodass sich für den Wert u folgendes ergibt:

$$u = \sum_{i=0}^{I} w_i \cdot x_i - \Theta = \vec{w} \cdot \vec{x} - \Theta$$

Dabei entspricht $I \in \mathbb{N}$ der Anzahl der Eingangswerte, welche in der Abbildung auf $I = 4$ gesetzt wurde. Die zweite Operation entspricht der Bildung des ausgehenden Spannungsimpulses und wird durch eine einfache Stufenfunktion der Form

$$\varphi(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases}$$

nachgebildet. Dabei würde eine 1 Aktivität des Perzeptrons bedeuten, also das weiterleiten eines Aktionspotentials über das Axon, während eine 0 keine Aktivität bedeutet. Die Funktion $\varphi(u)$ wird als *Aktivierungsfunktion* bezeichnet. Die Bedeutung des Wertes Θ bei

der Bildung von u wird aus der Definition von $\varphi(u)$ ersichtlich. Θ entspricht dem Schwellenwert des Potentials, der erreicht werden muss, bevor ein Aktionspotential aufgebaut werden kann, da die Aussage $u>0$ gleichbedeutend ist mit $\vec{w}\cdot\vec{x}>\Theta$. Schließlich wird das Axon in seiner Funktionsweise über den binären Ausgangswert $y=f(\vec{x})=\varphi(u)$ modelliert.

Das Lernen erfolgt beim Perzeptron über ein iteratives Lernverfahren, bei dem für jedes $(\vec{x}_k, y_k)$ aus dem Lerndatensatz $D^{lern}=\{(\vec{x}_k, y_k)|1\le k\le N\}$ eine Anpassung des Gewichtsvektors $\vec{w}$ mittels der *Perzeptronlernregel*

$$\vec{w}_{neu}=\vec{w}_{alt}+\alpha\left(y_k-f(\vec{x}_k)\right)\vec{x}_k \text{ und } \Theta_{neu}=\Theta_{alt}+\alpha\left(y_k-f(\vec{x}_k)\right)$$

vorgenommen wird. Hierbei ist $0<\alpha\le 1$ die so genannte *Lernkonstante*, über welche das Ausmaß eines Anpassungsschrittes, und damit die Lerngeschwindigkeit, variiert werden kann.

Das Perzeptron ist ein einfacher Klassifikator, der mittels einer Hyperebene im Raum und einer Entscheidungsregel in Form der Aktivierungsfunktion linear trennbare Zweiklassenprobleme lösen kann. Damit können unter Verwendung mehrerer Perzeptrons auch einfache Mehrklassenprobleme, wie in Abschnitt 3.4 beschrieben, gelöst werden, solange die im Mehrklassenproblem enthaltenen Zweiklassenprobleme linear trennbar sind. An dieser Stelle sei angemerkt, dass in der Praxis neben der hier erwähnten Stufenfunktion auch beliebige sigmoide Funktionen, wie zum Beispiel der hyperbolische Tangens, oder aber auch eine lineare Aktivierungsfunktion $\varphi(u)=u$ Verwendung findet. Je nach gewählter Aktivierungsfunktion nimmt die Perzeptronlernregel eine andere Form an. Welche Aktivierungsfunktion im Einzelnen verwendet werden sollte, hängt von der Art des Lernproblems ab.

3.5.4.2. Das Multilagenperzeptron

Wie eingangs erwähnt, rührt die Überlegenheit des Gehirns, was Lernfähigkeit und die Fähigkeit Probleme zu lösen angeht, von der starken Vernetzung der Nervenzellen untereinander. Wie Rosenblatts Perzeptron zeigt, können einzelne Nervenzellen als einfache Klassifikatoren zweier Klassen angesehen werden, die nur auf eine begrenzte Auswahl von Problemen anwendbar sind. Eine Ansammlung einzelner Nervenzellen kann bereits als einfacher Klassifikator mehrerer Klassen angesehen werden, doch auch in diesem Fall

müssen die Probleme relativ einfacher Natur sein. Die Informationsverarbeitung im Gehirn durchläuft jedoch mehrere Ebenen, oder Schichten, von Nervenzellen, bevor der letzte Verarbeitungsschritt durchgeführt wird. Es liegt also nahe, künstliche Neuronen zu größeren Netzen zusammenzuschalten, um weiterentwickelte Lernmaschinen zu konstruieren. Die so entstandenen Netzwerke werden als künstliche neuronale Netze bezeichnet. Es existieren viele unterschiedliche Architekturen für künstliche neuronale Netze, doch die bekanntesten und am meisten verbreiteten bilden die so genannten *Multilagenperzeptrons* (engl. *multilayer perceptron*), abgekürzt MLPs.

MLPs sind Netzwerke bestehend aus künstlichen Neuronen, die in mehreren Schichten organisiert sind. Während Neuronen, die sich in derselben Schicht befinden, keine Verbindungen untereinander besitzen, sind zwei aufeinander folgende Schichten miteinander verbunden. Der Informationsfluss in einem MLP ist unidirektional, was sich darin äußert, dass die Ausgaben einer Schicht als Eingaben der nachfolgenden Schicht verwendet werden. In der Regel sind zwei aufeinander folgende Schichten vollständig verbunden, das heißt jedes Neuron einer Schicht ist mit allen Neuronen der vorhergehenden Schicht verbunden. Jedes Neuron in einem MLP entspricht dem von Rosenblatt präsentierten Perzeptron, jedoch wird in der Regel der hyperbolische Tangens als Aktivierungsfunktion verwendet. Weiterhin besitzt jedes Neuron einen eigenen Gewichtsvektor und eine eigene Schwelle, die unabhängig von den anderen Neuronen im MLP angepasst werden können, was den MLPs ihre Mächtigkeit verleiht.

Ein Beispiel für ein MLP mit fünf Eingangswerten und drei Ausgangswerten ist in Abbildung 31 auf der folgenden Seite dargestellt. MLPs werden üblicherweise als ungerichtete Graphen dargestellt, die zwei Arten von Knoten enthalten. Zum einen wären das Knoten die Werte wie Ein- und Ausgaben repräsentieren (in der Regel durch Quadrate symbolisiert), zum anderen Knoten, die für ganze Neuronen stehen (in der Regel durch Kreise dargestellt). Die einzelnen Knoten sind durch Kanten verbunden, welche synaptische Kopplungen zwischen den entsprechenden Knoten darstellen. Somit steht jede Kante auch für ein Gewicht im MLP.

Man unterscheidet bei MLPs zwischen drei grundsätzlichen Arten von Schichten. Die erste Schicht ist die *Eingabeschicht* (engl. *input layer*), welche die *Eingabeknoten* (engl. *input nodes*) enthält. Jeder Eingabeknoten repräsentiert dabei einen Eingangswert, der in das

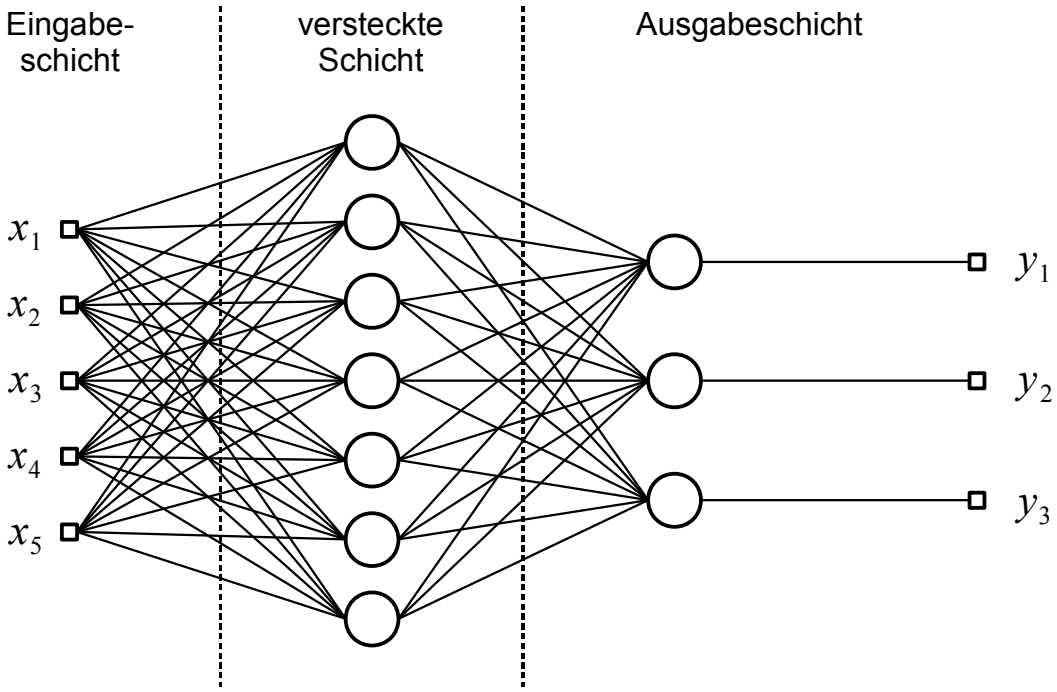


Abbildung 31: *Beispiel eines Multilagenperzeptrons mit einer versteckten Schicht*

neuronale Netz gespeist wird. Die zweite Schicht ist die *Ausgabeschicht* (engl. *output layer*), welche die Ausgaben des MLPs und die sie erzeugenden Neurone enthält. Zwischen Eingabe- und Ausgabeschicht kann eine beliebige Anzahl so genannter *versteckter Schichten* (engl. *hidden layers*) angeordnet werden, wobei mindestens eine versteckte Schicht vorhanden sein muss. Diese enthalten Neuronen, deren Aufgabe die der Datenvorverarbeitung und Merkmalsextraktion für die Ausgabeschicht darstellt. In der Regel enthalten MLPs in der Anwendung eine bis zwei versteckte Schichten. Es hat sich gezeigt, dass die Verwendung von mehr als zwei versteckten Schichten im Allgemeinen keinen zusätzlichen Gewinn bringt [45]. Die Anzahl der Neuronen in einer versteckten Schicht ist generell beliebig wählbar, jedoch sollte diese nicht zu groß gewählt werden um Überanpassung zu vermeiden. Bereits das Hinzufügen eines einzelnen Neurons kann, je nach Größe der vorherigen und nachfolgenden Schicht, zu einem erheblichen Anstieg der Parameterzahl des MLPs führen.

MLPs werden zur Lösung vieler Probleme verwendet, vorzugsweise jedoch für Mehrklassen- oder nichtlinear trennbare Zweiklassenprobleme. Bei Mehrklassenproblemen ist die variable Dimensionierung der Ausgabeschicht ein großer Vorteil. In der Praxis werden genauso viele Ausgangsknoten verwendet wie Klassen vorhanden sind, wobei jeder Aus-

gangswert als proportional zur Wahrscheinlichkeit angesehen werden kann, dass ein Eingangsvektor der entsprechenden Klasse zugehörig ist. In diesem Fall wird ein Eingangsvektor also derjenigen Klasse zugeteilt, für die der entsprechende Ausgangswert maximal ist, was der Regel der maximalen Konfidenz bzw. Maximum-A-Posteriori entspricht. Bei nichtlinear trennbaren Zweiklassenproblemen besteht die Ausgabeschicht aus einem einzelnen Neuron, mit dessen Hilfe die zwei Klassen einfach anhand des Vorzeichens des Ausgangswertes unterschieden werden können.

Durch eine versteckte Schicht können die Eingangsdaten in einen anderen Vektorraum transformiert werden, welcher das Problem linear trennbar macht, bevor sie die Ausgabeschicht erreichen. So kann das XOR-Problem bereits durch ein einfaches MLP gelöst werden, welches eine versteckte Schicht mit zwei Neuronen und ein einzelnes Ausgabeneuron besitzt.

Lernverfahren für MLPs sind logischerweise komplexer als Lernverfahren für einzelne Neuronen, das Perzeptron oder eine Ansammlung von Neuronen. Das bekannteste Lernverfahren ist der *Back-Propagation-Algorithmus*, auf welchen im nächsten Abschnitt eingegangen werden soll.

3.5.4.3. Der Back-Propagation-Algorithmus

Ziel des Back-Propagation-Algorithmus ist die Minimierung des mittleren quadratischen Fehlers auf einem gegebenen Lerndatensatz

$$D^{lern} = \left\{ (\vec{x}_k, \vec{y}_k) | \vec{x}_k \in \mathbb{R}^I, \ \vec{y}_k \in \mathbb{R}^O, \ 1 \leq k \leq N \right\},$$

wobei I die Anzahl der Eingangswerte und O die Anzahl der Ausgangswerten ist. Dabei wird davon ausgegangen, dass die Minimierung der einzelnen quadratischen Fehler auf den Daten zu einer Minimierung des mittleren quadratischen Fehlers führt. Der Algorithmus ist ein iteratives gradientenbasiertes Verfahren, bei welchem dem MLP die Daten aus dem Lerndatensatz in einer zufälligen Reihenfolge präsentiert werden. Für jedes einzelne Beispiel wird die Ausgabe des MLP berechnet und anhand der Abweichung zwischen dieser und der erwarteten Ausgabe werden die Gewichte des MLPs angepasst. Jedes Beispiel kann dem MLP nur einmal präsentiert werden, solange der Datensatz nicht vollständig bearbeitet wurde. Ist dies geschehen, so steht abermals der vollständige Datensatz zur Verfügung und kann ein weiteres Mal in zufälliger Reihenfolge bearbeitet werden. In der Regel

werden die Gewichte des MLPs mit kleinen Zufallszahlen initialisiert.

Die Ausgabe eines Neurons j in einer Schicht l des MLPs kann wie beim Perzeptron über eine gewichtete Summe wie folgt berechnet werden:

$$\hat{y}_j^{(l)} = \varphi_j\left(u_j^{(l)}\right) \text{ mit } u_j^{(l)} = \sum_{i=0}^{m} w_{ji}^{(l)} \cdot \hat{y}_i^{(l-1)}$$

m entspricht dabei der Anzahl der Neurone in der vorhergehenden Schicht $l-1$. Die Schwelle Θ wurde zur Vereinfachung als Gewicht $w_{j0}^{(l)}$ in die Summe aufgenommen und der Wert $\hat{y}_0^{(l-1)} = 1$ so gesetzt, dass sich für jedes Neuron die gewichtete Summe ergibt, die das Perzeptron verwendet. Für die erste versteckte Schicht mit $l = 1$ wird $\hat{y}_i^{(l-1)} = x_i$ gesetzt, während für die Ausgabeschicht mit $l = L$ ein Ausgangsvektor

$$\vec{o} = (o_{1,} \ldots o_O)^T = \left(\hat{y}_1^{(L)}, \ldots, \hat{y}_O^{(L)}\right)^T$$

definiert werden kann, wobei L der Anzahl der Schichten des MLPs entspricht. Die Berechnung dieses Vektors für ein Beispiel aus dem Lerndatensatz stellt den ersten Teil einer Lerniteration dar, bei welchem das MLP in seinem normalen Betriebszustand, der sogenannten *Vorwärtspropagation,* betrieben wird. Nach der Berechnung dieses Vektors kann zusammen mit der erwarteten Ausgabe $\vec{y}$ die quadratische Abweichung der Ausgabe des MLPs vom Sollwert mittels

$$E = \frac{1}{2} \sum_{i=1}^{O} (y_i - o_i)^2 = \frac{1}{2} \sum_{i=1}^{O} e_j^2$$

berechnet werden. Diese Größe gilt es hinsichtlich der Parameter des MLPs zu minimieren, was mit Hilfe eines Gradientenabstiegs des quadratischen Fehlers durchgeführt wird. Es kann somit für das Gewicht $w_{ij}^{(L)}$ eines Ausgabeneurons j ein Inkrement

$$\Delta w_{ij}^{(L)} = -\eta \frac{\partial E}{\partial w_{ij}^{(L)}} = -\eta \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial o_j} \frac{\partial o_j}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ij}^{(L)}}$$

definiert werden, wobei η, die Lernkonstante, ein Parameter zur Regulierung der Lernrate ist. Die einzelnen in der Gleichung vorkommenden partiellen Ableitungen ergeben sich zu

$$\frac{\partial E}{\partial e_j} = e_j, \quad \frac{\partial e_j}{\partial o_j} = -1, \quad \frac{\partial o_j}{\partial u_j^{(L)}} = \varphi_j'\left(u_j^{(L)}\right) \text{ und } \frac{\partial u_j^{(L)}}{\partial w_{ij}^{(L)}} = \hat{y}_i^{(L-1)}.$$

Damit ergibt sich für das Inkrement

$$\Delta w_{ij}^{(L)} = -\eta \frac{\partial E}{\partial w_{ij}^{(L)}} = \eta e_j \varphi_j'(u_j) \hat{y}_i^{(L-1)} = \eta \delta_j^{(L)} \hat{y}_i^{(L-1)}$$

mit $\delta_j^{(L)}$ als dem so genannten *lokalen Gradienten*, welcher sich aus der Gleichung

$$\delta_j^{(L)} = -\frac{\partial E}{\partial u_j^{(L)}} = -\frac{\partial E}{\partial e_j}\frac{\partial e_j}{\partial o_j}\frac{\partial o_j}{\partial u_j^{(L)}} = e_j \varphi_j'\left(u_j^{(L)}\right)$$

folgt.

Bei einem Neuron j, das sich in einer versteckten Schicht $l = L-1$ befindet, sieht der Sachverhalt weniger einfach aus, da für ein verstecktes Neuron kein Sollwert für seine Ausgabe $\hat{y}_j^{(l)}$ vorhanden ist. Jedoch lässt sich für hier analog ein lokaler Gradient zu

$$\delta_j^{(l)} = -\frac{\partial E}{\partial \hat{y}_j^{(l)}}\frac{\partial \hat{y}_j^{(l)}}{\partial u_j^{(l)}} = -\frac{\partial E}{\partial \hat{y}_j^{(l)}} \varphi_j'\left(u_j^{(l)}\right)$$

definieren. Aus der Definition von E ergibt sich die Ableitung nach $\hat{y}_j^{(l)}$ zu

$$\frac{\partial E}{\partial \hat{y}_j^{(l)}} = \sum_{i=1}^{O} e_i \frac{\partial e_i}{\partial \hat{y}_j^{(l)}} = \sum_{i=1}^{O} e_i \frac{\partial e_i}{\partial u_i^{(L)}}\frac{\partial u_i^{(L)}}{\partial \hat{y}_j^{(l)}} = \sum_{i=1}^{O} e_i \left(-\varphi_i'\left(u_i^{(L)}\right)\right) w_{ij}^{(L)} = -\sum_{i=1}^{O} e_i \varphi_i'\left(u_i^{(L)}\right) w_{ij}^{(L)} \quad .$$

Mit der Definition von $\delta_j^{(L)}$ lässt sich der obige Ausdruck vereinfachen zu

$$\frac{\partial E}{\partial \hat{y}_j^{(l)}} = -\sum_{i=1}^{O} \delta_i^{(L)} w_{ij}^{(L)} \quad .$$

Einsetzen in die Gleichung für $\delta_j^{(l)}$ liefert den lokalen Gradienten für ein verstecktes Neuron. Da diese Gleichung allgemeingültig ist, also für jede beliebige versteckte Schicht gilt, nicht nur für $l = L-1$, kann gleich der allgemeine Ausdruck für den lokalen Gradienten angegeben werden:

$$\delta_j^{(l)} = \varphi_j'\left(u_j^{(l)}\right) \sum_{i=1}^{n} \delta_i^{(l+1)} w_{ij}^{(l+1)}$$

Hier entspricht n der Anzahl der Neuronen in der nächsthöheren Schicht.

Die Definition von $\delta_j^{(l)}$ verdeutlicht den zweiten Betriebszustand des MLPs, der nur beim Lernvorgang Anwendung findet: Die *Rückwärtspropagation,* die dem Algorithmus seinen Namen gibt. Zur Berechnung des lokalen Gradienten für ein bestimmtes verstecktes Neu-

ron müssen zunächst die lokalen Gradienten aller Neurone in der nächsthöheren Schicht bekannt sein. Diese werden über die entsprechenden Gewichte des MLPs an die aktuelle Schicht weitergeleitet, was einer Umkehrung des normalen Informationsflusses in einem MLP entspricht. Mittels des lokalen Gradienten kann ein Gewicht $w_{ji}^{(l)}$ eines Neurons j in der Schicht l mittels des Ausdrucks $w_{ji,neu}^{(l)} = w_{ji,alt}^{(l)} + \eta\, \delta_j^{(l)}\, \hat{y}_i^{(l-1)}$ angepasst werden. Der Algorithmus wiederholt den oben beschriebenen Vorgang, bis ein vorgegebenes Abbruchkriterium erfüllt ist, welches beispielsweise eine feste Anzahl von Iterationen oder das Unterschreiten einer gegebenen Schwelle in der Änderung des mittleren quadratischen Fehlers über dem Lerndatensatz sein kann.

Die beschriebene Form des Back-Propagation-Algorithmus ist die grundlegendste und einfachste Form des Algorithmus. Die Trajektorie, die der Algorithmus durch den Parameterraum (also durch den Raum der Gewichte) des MLPs beschreibt, ist eine Approximation der Trajektorie, welche das Gradientenabstiegsverfahren durch diesen Raum beschreiben würde. Diese Trajektorie wird umso glatter, je kleiner der Wert von η gewählt wird, jedoch mit dem Nachteil einer langsameren Konvergenz und damit eines langsameren Lernens. Ein großer Wert von η führt zwar zu einer schnelleren Konvergenz, jedoch kann dies zu Oszillationen oder Instabilitäten des Lernvorgangs führen. Um diesem vorzubeugen kann als Erweiterung des Algorithmus ein so genannter *Trägheitsterm* zur Glättung der Trajektorie in das Inkrement zur Anpassung der Gewichte einbezogen werden. Das resultierende Inkrement wäre damit:

$$\Delta w_{ji,t}^{(l)} = \alpha\, \Delta w_{ji,t-1}^{(l)} + \eta\, \delta_j^{(l)}\, \hat{y}_i^{(l)}, \quad 0 \le |\alpha| < 1$$

Dabei ist t die Nummer der aktuellen Iteration. Mit dieser Erweiterung werden alle vorhergehenden Iterationen in das aktuelle Inkrement mit einbezogen, was zu einer Glättung der Trajektorie durch den Parameterraum führt. Die Konstante $0 \le |\alpha| < 1$ gibt dabei an, wie stark der Einfluss des Trägheitsterms sein soll. Die obige Gleichung für das Inkrement $\Delta w_{ji,t}$ kann damit auch als eine Summe aller vorherigen Inkremente dargestellt werden, was zu folgendem Ausdruck führt:

$$\Delta w_{ji,t} = -\eta \sum_{s=0}^{t} \alpha^{t-s} \frac{\partial E_s}{\partial w_{ji,s}}$$

Ein gleich bleibendes Vorzeichen aufeinander folgender Summanden in der obigen Summe

deutet auf eine kontinuierliche Verringerung des quadratischen Fehlers hin. In diesem Fall nimmt der Betrag der Summe und damit das Inkrement zu, was diese Verringerung in den nächsten Iterationen weiter beschleunigt. Unterscheiden sich aufeinander folgende Summanden jedoch im Vorzeichen, würde das auf Oszillationen hindeutet und der Betrag der Summe wird kleiner, was das Ausmaß der Oszillationen mit der Zeit verringert [45].

Es existieren weitere Erweiterungen des Algorithmus, die ihn beispielsweise auf das Lernen zeitlicher Verläufe anwendbar machen. Diese Erweiterungen sollen im Rahmen dieser Arbeit jedoch keine Erwähnung finden, da sie für das Thema nicht relevant sind.

3.5.4.4. Andere Netzarchitekturen

Neben MLPs existiert eine Reihe weiterer Netzarchitekturen, welche zum Teil auf bestimmte Problemstellungen zugeschnitten sind. So gibt es als Erweiterung der MLPs *zeitverzögerte Neuronale Netze* [55] (engl. *time delayed neural network*, kurz *TDNN*). Diese entsprechen einem MLP, das zeitlich aufeinander folgende Eingaben in Form eines Zeitfensters erhält. Das besondere an TDNNs ist, dass versteckte Schichten und Ausgabeschicht in mehreren Replikationen vorhanden sind, wobei jede der Replikationen den gleichen Satz an Gewichten auf einen eigenen Ausschnitt dieses Zeitfensters anwendet.

Eine weitere verbreitete Netzarchitektur sind die sogenannten *Kohonenkarten* (engl. *Kohonen map*), auch als *Kohonennetze* (engl. *Kohonen network*) oder *selbstorganisierende Karten* (engl. *self-organizing maps*, kurz *SOM*) bezeichnet. Sie wurden von dem namensgebenden finnischen Ingenieur Teuvo Kohonen im Jahre 1982 entwickelt [56]. Eine Kohonenkarte besteht aus einer Schicht von Neuronen, die in einem zweidimensionalen Gitter angeordnet sind. Dadurch ergibt sich eine Nachbarschaftsbeziehung zwischen den Neuronen, welche beim Lernprozess berücksichtigt wird. Der Lernprozess selbst stellt ein iteratives unüberwachtes Lernferfahren dar, das auf den euklidischen Distanzen zwischen den Gewichtsvektoren der einzelnen Neuronen und den Lerndaten basiert. Dies bedeutet genauer, dass in einem Lernschritt die Gewichte desjenigen Neurons am stärksten angepasst werden, bei dem die Differenz des Gewichtsvektors zum aktuellen Lernbeispiel am geringsten ist. Dieses Neuron wird als Siegerneuron bezeichnet. Alle anderen Neuronen werden entsprechend einer Nachbarschaftsfunktion in geringerem Maße angepasst, wobei die Größenordnung der Anpassung mit dem Abstand eines Neurons zum Siegerneuron abnimmt. Kohonenkarten erzeugen eine Art topografischer Karte der gezeigten Daten. Dabei kann

jedem Datenpunkt ein Neuron im zweidimensionalen Gitter zugewiesen werden und Daten, die Ähnlichkeiten aufweisen, in der Regel dem gleichen Neuron oder Neuronen, die nahe beieinander liegen, zugewiesen werden. Somit können Ähnlichkeiten in den Daten festgestellt werden, die unter Umständen vorher nicht bekannt waren. Weiterhin sind Kohonenkarten eine vereinfachte diskrete Approximation des Eingangsdatenraums, da sie eine große Anzahl von Eingangsdaten auf eine kleine Anzahl prototypischer Gewichtsvektoren abbilden, was die Gemeinsamkeit mit Clusteringalgorithmen zeigt. Kohonenkarten werden in der Regel als Vorverarbeitungsschritt verwendet, um die Daten für ein anderes Lernverfahren aufzubereiten und mit diesen dann möglicherweise ein besseres Ergebnis zu erzielen.

Eine weitere Art neuronaler Netze sind die *Hopfield-Netze* (engl. *Hopfield network*, [57], [58]), gelegentlich auch *Attraktornetze* (engl. *attractor network*) genannt, welche rekursive neuronale Netze darstellen, die als Assoziativspeicher genutzt werden können. In ein solches Netzwerk kann eine endliche Anzahl von Vektoren $\vec{x}_i \in R^n$ eingespeichert werden, welche als *Attraktoren* bezeichnet werden. Der Grund für die Namensgebung ergibt sich aus der Betriebsweise des Netzes: Zunächst wird ein initialer Zustand $\vec{x}_s \in R^n$ als Eingangsvektor an das Netz angelegt. Das Netz berechnet daraus eine Ausgabe, welche im nächsten Schritt als neuer Eingangsvektor oder Zustand verwendet wird. Das Netz berechnet also iterativ seinen aktuellen Zustand aus dem Zustand aus der vorherigen Iteration. Es zeigt sich, dass mit steigender Anzahl an Iterationen der Zustand des Netzes gegen einen der Attraktoren strebt und dass nach dessen Erreichen keine weitere Änderung des Zustands des Netzes mehr erfolgt. Die Attraktoren stellen somit stabile Zustände für das Hopfield-Netz dar. Beobachtet man die Trajektorien des Netzes durch den Zustandsraum ausgehend von unterschiedlichen Anfangszuständen, so zeigt sich, dass jedem Attraktor ein ihn einschließendes Volumen im Zustandsraum zugeordnet werden kann. Anfangszustände, die aus einem solchen Volumen gewählt werden, erzeugen Trajektorien, welche immer im entsprechenden Attraktor selbst enden, als wäre der Attraktor Quelle eines anziehenden Kraftfeldes, das ausschließlich auf das ihn einschließende Volumen begrenzt ist. Dieser Tatsache verdanken die Attraktoren ihren Namen. Die Interpretation der Attraktoren als gespeicherte Daten offenbart die Funktionsweise des Hopfield-Netzes als Assoziativspeicher: Eingangsdaten, die den Attraktoren ähnlich sind, weil sie ähnliche Werte aufweisen oder auf irgendeine Weise unvollständig sind, führen dazu dass das Netz denjeni-

gen Zustand erreicht, den es unter den eingespeicherten Zuständen am ehesten mit den Eingangsdaten assoziiert, nämlich den entsprechenden Attraktor. Neben ihrer Nutzung als assoziative Speichereinheiten dienen Attraktornetze auch als anschauliches Modell zur Funktionsweise des Gehirns. So zeigt sich, dass viele Regionen des Gehirns in ihrer Funktionsweise der eines Attraktonetzes entsprechen [59].

Schließlich soll noch kurz auf eine weit verbreitete Form neuronaler Netze eingegangen werden, welche von Vapnik entwickelt wurde und auf dessen Beiträgen zur statistischen Lerntheorie beruht. Diese spezielle Art neuronaler Netze wird als *Support Vector Machine*, kurz *SVM,* bezeichnet ([60], [42]). SVMs sind lineare Klassifikatoren, die versuchen die optimale Klassifikationsgrenze in Form einer Hyperebene zu finden. Optimal bedeutet in dieser Hinsicht, dass der Abstand derjenigen Datenpunkte aus dem Lerndatensatz, die der Hyperebene am nächsten sind, maximiert wird. Hierzu wird auf beiden Seiten der Klassifikationsgrenze ein Rand definiert ‚innerhalb dessen sich keine Datenpunkte befinden dürfen. Der Lernvorgang besteht in der Suche derjenigen Hyperebene, die diesen Rand maximiert. Es zeigt sich, dass die Lage der Klassifikationsgrenze sich lediglich über die ihr am nächsten gelegenen Datenpunkte definiert, also diejenigen Punkte, die genau auf dem Rand liegen. Diese Datenpunkte werden als *Stützvektoren* bezeichnet und geben der SVM ihren Namen. Mit Hilfe dieser Stützvektoren lässt sich die Klassifikationsgrenze vollständig beschreiben. Die Annahme einer Hyperebene als Klassifikationsgrenze impliziert die Annahme der linearen Trennbarkeit der Lerndaten, was jedoch im Allgemeinen nicht gewährleistet ist. Um dies zu umgehen, bedienen sich SVMs einer Transformation in einen höherdimensionalen Raum, den so genannten *Merkmalsraum*. Die Idee, die hinter diesem Vorgehen steckt, ist, dass jede beliebige Menge von Vektoren, welche in einem Vektorraum nichtlinear trennbar ist in einem höherdimensionalen Vektorraum mit genügend hoher Dimension linear trennbar wird (siehe Abschnitt 3.3.2). Mittels einer geeigneten Transformationsfunktion wird ein Eingangsvektor aus dem Eingangsraum in den Merkmalsraum transformiert. Daraufhin werden die Ähnlichkeiten der Stützvektoren zum transformierten Eingangsvektor bestimmt. Beide Schritte werden mit Hilfe eines *Kernels* durchgeführt, der einem Skalarprodukt ähnlich die Abbildung zweier Vektoren aus dem Eingaberaum auf eine reelle Zahl darstellen. Der Vorteil eines Kernels ist, dass die Transformation eines Vektors aus dem Eingangsraum in den Merkmalsraum von ihm implizit durchgeführt wird, sodass die explizite Transformation der ursprünglichen Vektoren nicht notwendig ist. Ker-

nels werden im Falle von SVMs verwendet, um die Ähnlichkeiten eines gegebenen Eingangsvektors zu den einzelnen Stützvektoren zu berechnen, da der vom Kernel gelieferte Wert als Ähnlichkeitsmaß im Merkmalsraum interpretiert werden kann. Da die Abbildung der Stützvektoren die Lage der trennenden Hyperebene im Merkmalsraum vollständig beschreibt kann über Ähnlichkeitsmaße zu den Stützvektoren die Lage eines Datenpunktes relativ zur Trennebene bestimmt werden. Dieser Vorgang und somit auch die eigentliche Klassifikation wird mit Hilfe eines linearen Ausgabeneurons ausgeführt, welches die Ähnlichkeitsmaße zu den Stützvektoren als Eingabe erhält.

Der Rand um die Trennebene hat den Zweck, die Generalisierungsfähigkeit der SVM zu maximieren, indem der Abstand der Ebene zu den Vertretern beider Klassen maximiert wird. Somit besteht eine hohe Chance, dass neue Daten, die unter Umständen näher an der Trennebene liegen, korrekt klassifiziert werden. Jedoch ist es nicht immer möglich, einen genügend großen Rand um die Trennebene zu legen, weshalb eine Weiterführung des SVM-Ansatzes eine Verletzung des Rands erlaubt. Bei der Optimierung wird toleriert, dass eine bestimmte Anzahl an Datenpunkten innerhalb des Rands oder gar auf der falschen Seite der Klassifikationsgrenze liegen darf, was zu einem weichen Rand (engl. *soft margin*) führt.

4. Methode

Mit den Werkzeugen aus der statistischen Lerntheorie kann nun ein geeignetes Verfahren zur Lösung des Problems der Sekundärstruktursvorhersage von Proteinen angegangen werden. Bevor jedoch ein Ansatz zur Lösung dieses Problems entwickelt werden kann, müssen zunächst einige Vorbetrachtungen gemacht werden. Einerseits ist es wichtig bereits existierende Methoden zu analysieren, um einen Überblick über mögliche Methoden zur Lösung des Problems zu erhalten. Andererseits muss das Problem derart definiert werden, dass die Lerntheorie zu dessen Lösung angewendet werden kann. Dementsprechend muss auch eine geeignete Repräsentation der verwendeten Daten gefunden werden, um dann geeignete lerntheoretische Methoden zur Vorhersage der Sekundärstruktur von Proteinen einzusetzen.

4.1. Vorbetrachtung

Das Problem der Sekundärstrukturvorhersage von Proteinen ist bereits seit den 70er Jahren des vergangenen Jahrhunderts Gegenstand der Forschung. Abschnitt 2.7 erwähnte bereits zwei der ersten Methoden, deren Ziel es war die Sekundärstruktur unbekannter Proteine aus deren Sequenz zu berechnen: Die Chou-Fasman-Methode und die GOR-Methode. Beide Methoden basieren auf der Informationstheorie und versuchen anhand eines geeigneten Modells von Wahrscheinlichkeiten einen Zusammenhang zwischen zwei Alphabeten, nämlich dem der Primärstruktur (Einbuchstabencode der Aminosäuresequenz, Abschnitt 2.2) und dem der Sekundärstruktur, herzustellen. Von beiden Methoden findet gegenwärtig nur noch die GOR-Methode Anwendung ([61], [62], [63]). Seit der Entstehung dieser ersten Methoden entstand eine Vielzahl teils sehr unterschiedlicher Ansätze, das Problem der Sekundärstrukturvorhersage von Proteinen zu lösen. Einige dieser Methoden soll der folgende Abschnitt behandeln.

4.1.1. Vorhandene Programme zur Sekundärstrukturvorhersage

In den letzten Jahrzehnten wurde eine Reihe von Methoden zur Vorhersage von Sekundärstrukturen entwickelt und eingesetzt, die zum großen Teil auf Lernverfahren, wie etwa künstlichen neuronalen Netzen, aber auch auf anderen Ansätzen basieren. Da das Ziel dieser Arbeit die Entwicklung einer auf Lernverfahren basierenden Methode zur Sekundärstrukturvorhersage ist, sollen zunächst diejenigen Ansätze betrachtet werden, die maschi-

nelles Lernen als Ausgangspunkt verwenden. Die ersten Verfahren dieser Art erschienen in den späten 1990er Jahren. Seit dieser Zeit fanden neuronale Netze immer mehr Anwendung in realen Problemen. Innerhalb der zweiten Hälfte der Neunzigerjahre hielten neuronale Netze auch in die Sekundärstrukturvorhersage Einzug. Eines der ersten Programme dieser Art ist PHD [27], das 1996 von Burkhard Rost entwickelt wurde. Dieses verwendet zwei MLPs (Multilagenperzeptrons, Abschnitt 3.5.4.2) zur Vorhersage der Sekundärstruktur. Das erste MLP stellt die Korrelation zwischen Sequenz und lokaler Sekundärstruktur her, während das zweite MLP die Vorhersage des ersten verwendet um eine Struktur-Struktur-Korrelation herzustellen, also eine Korrelation zwischen der von PHDs erstem MLP vorhergesagten lokalen Sekundärstruktur auf benachbarten Positionen. Die endgültige Vorhersage ist somit eine nachbearbeitete Version der Vorhersage des ersten MLPs. Dieser stufenweise Aufbau wird von vielen anderen Vorhersageprogrammen verwendet. So besteht das im Jahre 1999 von David Jones entwickelte PSIPRED [30] ebenfalls aus zwei hintereinander geschalteten MLPs, die in ihrer Funktionsweise denen von PHD ähneln. Der Hauptunterschied beider Programme liegt in der Repräsentation der Eingangsdaten, auf die in Abschnitt 4.1.3 eingegangen werden soll. Im selben Jahr, in dem PSIPRED entwickelt wurde, entstand auch JPred [64], welches aus einer Anzahl künstlicher neuronaler Netze besteht, die auf unterschiedlichen Darstellungen der Eingangsdaten trainiert wurden. Die Vorhersage der Sekundärstruktur ergibt sich aus der Kombination der Vorhersagen der einzelnen neuronalen Netze. Weitere auf ähnlich aufgebauten, unidirektionalen neuronalen Netzen basierende Programme sind JUFO [65] und SABLE [31]. Daneben existiert mit SSpro [29], beziehungsweise dessen Nachfolger Porter [28], ein Ansatz, bei dem ein rekursives neuronales Netz zur Sekundärstrukturvorhersage Verwendung findet. Hierbei wird zur Vorhersage der Sekundärstruktur eines Residuums auch die Sekundärstruktur seiner Nachbarschaft verwendet, welche vom Netz rekursiv berechnet wird. PMSVM [66] wiederum ist in seinem zweistufigen Aufbau den bereits vorgestellten Programmen ähnlich, verwendet jedoch statt MLPs SVMs (Abschnitt 3.5.4.4) zur Vorhersage. Den bisher vorgestellten Verfahren, welche alle lerntheoretische Ansätze verwenden, soll noch SPARROW [1] hinzugefügt werden, das von Francesco Bettella im Rahmen seiner Doktorarbeit an der Freien Universität Berlin in der gleichen Arbeitsgruppe, in der auch die vorliegende Arbeit entstand, entwickelt wurde. Dieses Verfahren basiert auf der Verwendung eines zweistufigen quadratischen Klassifikators, bei dem die einzelnen Stufen der

von Fisher (Abschnitt 3.5.2) vorgestellten linearen Diskriminanzfunktion ähnlich sind, jedoch auch gewichtete Multiplikationen der einzelnen Komponenten eines Eingangsvektors berücksichtigen. Jede der beiden Stufen enthält eine Reihe solcher Klassifikatoren, die gemäß Abschnitt 3.4.1 einzelne Zweiklassenprobleme lösen. Dabei werden sowohl Klassifikatoren nach dem 1-1-Ansatz als auch nach dem 1-r-Ansatz verwendet. Die endgültige Vorhersage der Sekundärstruktur wird von einem MLP durchgeführt, das hierzu die Vorhersage des zweistufigen quadratischen Klassifikators benutzt.

Neben der Lerntheorie bieten auch noch einige andere Konzepte die Möglichkeit, die Sekundärstruktur von Proteinen vorherzusagen. So existieren Programme auf der Grundlage von Hidden Markov Models (IPSSP [67], BSMPSSP [68]), Threading (PROSPECT [69]) und Sequenzvergleichen (PREDATOR [70]). Zudem existieren auch Mischverfahren, die beispielsweise die GOR-Methode mit einem neuronalen Netz kombinieren (PROF [32]), Hidden Markov Models mit neuronalen Netzen kombinieren (YASPIN [71]), auf Hidden Markov Models basierendes Homology Modelling (SAM [72]), sowie die Kombination eines neuronalen Netzes mit einem Clusteralgorithmus (MUPRED [73], verwendet fuzzy k-nearest-neighbor-Clustering [74]). Diese Verfahren liegen jedoch aufgrund der verwendeten Ansätze nicht im Zentrum der vorliegenden Arbeit, welche sich mit einem neuen Lernverfahren befasst. Aus diesem Grund wird nicht detaillierter auf diese Programme eingegangen.

Die Vielzahl an unterschiedlichen Ansätzen, welche neuronale Netze verwenden zeigt, dass Lernverfahren durchaus auf die Sekundärstrukturvorhersage von Proteinen anwendbar sind. Jedoch zeigt sie auch, dass wenig Spielraum für neue Entwicklungen ausschließlich auf Grundlage neuronaler Netze existiert. Viele der vorgestellten Verfahren unterscheiden sich zum Teil lediglich in der gewählten Netzarchitektur und in der Repräsentation der Daten, sodass jede neue Entwicklung dieser Art lediglich eine Variation existierender Verfahren wäre. SPARROW, welches in dieser Hinsicht einen recht individuellen Weg geht, zeigt jedoch, dass das Gebiet Möglichkeiten für neue Ansätze bietet und dass ein solcher Ansatz durchaus mit anderen Verfahren vergleichbar sein kann. Jedoch zeigen die detaillierten Untersuchungen, die in [1] durchgeführt wurden auch, dass für die Verwendung von Klassifikatoren zweier Klassen wenig Spielraum für weitere Entwicklungen bleibt.

Ziel dieser Arbeit ist es nun, ein weiteres neuartiges Lernverfahren zu entwickeln, das

einerseits eine zuverlässige Vorhersage von Sekundärstrukturen ermöglichen soll und andererseits ein neuartiges Werkzeug bieten soll, mit dessen Hilfe Mehrklassenprobleme gelöst werden können. Hierzu muss zunächst die Verknüpfung zwischen der Sekundärstrukturvorhersage von Proteinen und Mehrklassenproblemen hergestellt werden, indem das Problem der Sekundärstrukturvorhersage als Klassifikationsproblem definiert wird. Diesem Zweck soll der folgende Abschnitt dienen.

4.1.2. Sekundärstrukturvorhersage als Klassifikationsproblem

Zunächst stellt sich die Frage, wie das Problem selbst definiert werden sollte, um die Verwendung einer Lernmaschine zu dessen Lösung zu ermöglichen. In Abschnitt 3.1.2 sind einige grundlegende Probleme aufgeführt, die mit Hilfe von Lernverfahren gelöst werden können. Von diesen eignet sich das Klassifikationsproblem am ehesten als Betrachtung der vorliegenden Fragestellung: Um das Problem der Sekundärstrukturvorhersage von Proteinen zu lösen, ist es erforderlich einen Mechanismus zu finden, der jedes Residuum innerhalb eines Proteins dem korrekten Sekundärstrukturtyp zuweist. Damit stellt die Vorhersage der Sekundärstruktur eine Klassifikation der Residuen eines Proteins und der erwähnte Mechanismus einen Klassifikator dar. Diese Sichtweise auf das Problem wird von den auf neuronalen Netzen basierenden Methoden, wie auch SPARROW, geteilt.

Diese Anschauung wirft jedoch die Frage auf, wann eine Klassifikation als richtig anzusehen ist. Existierende Verfahren zur Sekundärstrukturvorhersage orientieren sich hierfür an Proteinen mit bekannter Sekundärstruktur. Dabei variiert die Anzahl der Sekundärstrukturtypen und damit auch die Anzahl der Klassen, die von den einzelnen Methoden unterschieden werden, von drei bei den meisten Verfahren, wie PHD, PSIPRED oder SPARROW, bis hin zu acht Klassen bei einer Variante von SSpro, die SSpro8 heißt. Jedoch handelt es sich bei allen Verfahren stets um ein Mehrklassenproblem (Abschnitt 3.4) und soll auch in dieser Arbeit als solches behandelt werden.

Die große Anzahl an bekannten Proteinstrukturen, die in der Protein Data Bank (PDB, [75]) vorhanden ist, bietet eine genügend hohe Datenmenge um einen Klassifikator mittels der ERM (Abschnitt 3.1.3) zu trainieren. Zudem bietet es sich an, das Lernproblem als überwachtes Lernproblem gemäß Abschnitt 3.5.1 zu sehen, da ja dank der Existenz bekannter Proteinstrukturen eine erwartete Klassifikation gegeben ist.

In welchem Bezug steht nun das in diesem Abschnitt dargestellte Klassifikationsproblem zu der in Abschnitt 2.7 beschriebenen Transformation von Zeichenketten, als die die Sekundärstrukturvorhersage von Proteinen angesehen werden kann? Angenommen, jede Klasse, der ein Residuum zugeordnet werden kann, wird durch ein eindeutiges Zeichen repräsentiert. Damit kann durch die Klassifikation jedem Residuum ein Zeichen zugeordnet werden, das der Klasse, also auch dem Sekundärstrukturtyp entspricht, zu dem das Residuum gehört. Aus den Klassifikationen aller Residuen kann so eine Zeichenkette erzeugt werden, die der Sekundärstruktur des Proteins entspricht und bereits in Abbildung 19 auf Seite 48 gezeigt wurde. Klassifikation und Transformation der Primärstrukturzeichenkette in eine Sekundärstrukturzeichenkette sind damit äquivalent.

4.1.3. Datenrepräsentation

In der Praxis wird die Klassifikation eines Residuums nicht nur anhand des Residuums selbst, sondern auch anhand der in seiner Nachbarschaft befindlichen Residuen durchgeführt. Dies bietet sich an, da die Entstehung der meisten Sekundärstrukturen aus dem Zusammenspiel benachbarter Residuen resultiert (Abschnitt 2.3). Gegen die Verwendung der gesamten Primärstruktur als Eingabe spricht hingegen die Tatsache, dass ein solches Vorgehen die Anzahl der Parameter des Klassifikators in die Höhe treiben würde und auf diese Weise ein Garant für die Überanpassung des Klassifikators wäre (Abschnitt 3.1.4). Gängige Verfahren zur Sekundärstrukturvorhersage verwenden aus diesem Grund ein Fenster auf der Primärstruktur zur Klassifikation eines Residuums, wobei die Größe des Fensters von Methode zu Methode variiert. Das Fenster beinhaltet das Residuum das klassifiziert werden soll, sowie eine Anzahl von Nachbarn in beide Richtungen entlang der Sequenz. In der Regel werden hierfür symmetrische Fenster verwendet, was bedeutet, dass die Zahl der Nachbarn in beide Richtungen gleich ist und sich das zu klassifizierende Residuum in der Mitte des Fensters befindet. Das Beispiel eines solchen Fensters, sowie des beschriebenen Klassifikationsverfahrens ist in Abbildung 32 auf der folgenden Seite dargestellt. Für die Klassifikation des Fett hervorgehobenen Alanins wird in diesem Fall das durch das schwarze Rechteck repräsentierte Fenster auf der Sequenz, die durch die obere Zeichenkette dargestellt ist, berücksichtigt. Der Klassifikator weist auf dieser Grundlage dem Alanin einen Sekundärstrukturtyp zu, der durch das markierte C in der unteren Zeichenkette, die die Sekundärstruktur darstellt, symbolisiert wird. Eine schrittweise Verschiebung

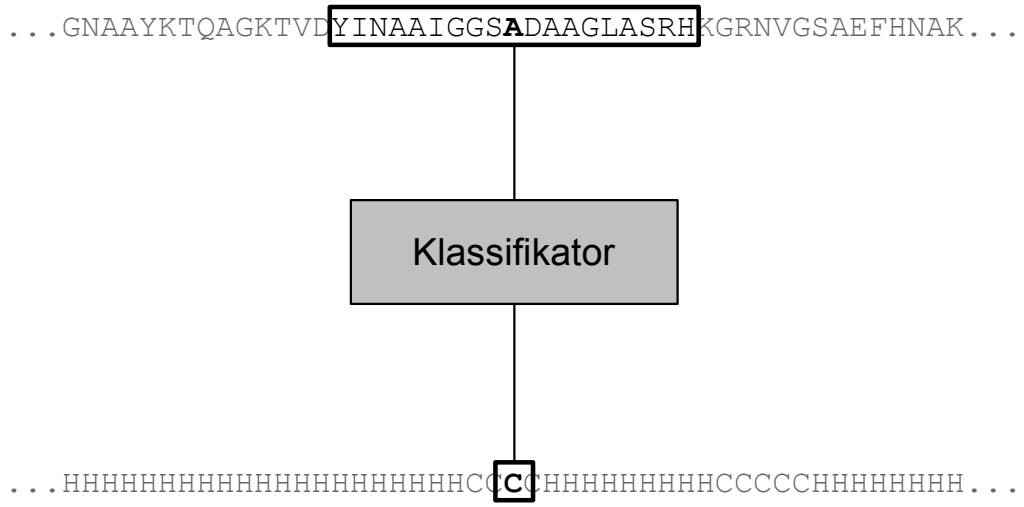


Abbildung 32: *Fensterbasierte Klassifikation von Residuen*

des Fensters entlang der Sequenz erlaubt es, die gesamte Sekundärstruktur des Proteins sukzessiv vorherzusagen. Bei einem symmetrischen Fenster spielt es dabei keine Rolle, ob das Fenster vom C-Terminus zum N-Terminus hin verschoben wird oder andersherum. Dies würde nur bei einem asymmetrischen Fenster eine Rolle spielen, in dem das zu klassifizierende Residuum nicht im Zentrum liegt. Tests mit solchen asymmetrischen Fenstern zeigen jedoch, dass diese keinen Vorteil gegenüber symmetrischen Fenstern bieten [1]. Aus diesem Grund verwendet der in dieser Arbeit beschriebene Klassifikator ebenfalls symmetrische Sequenzfenster zur Vorhersage der Sekundärstruktur eines Residuums.

4.1.3.1. Repräsentation des Sequenzfensters

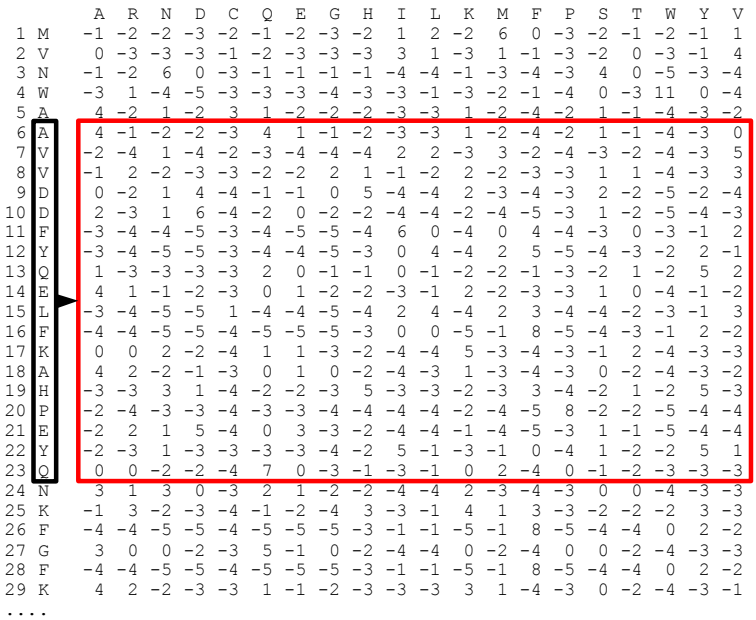
Da auf der Lerntheorie basierende Klassifikatoren mathematischen Funktionen entsprechen, müssen die Daten, die in diese Funktionen einfließen, auf eine mathematische Weise repräsentiert werden. Hierzu existieren unterschiedliche Ansätze wie beispielsweise orthogonale, binäre Vektoren mit 20 Einträgen, von denen jeder für eine der Standardaminosäuren (Abschnitt 2.1) steht. Befindet sich an einer bestimmten Position innerhalb der Sequenz eine bestimmte Aminosäure, so kann diese Position durch einen Vektor beschrieben werden, bei dem der Eintrag, der für diese Aminosäure steht, eine 1 enthält und alle anderen Einträge auf 0 gesetzt sind. Ein solcher Vektor lässt sich wie folgt beschreiben:

$$\vec{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{20} \end{pmatrix}, a_t \in \{0,1\}$$

Ein Sequenzfenster mit $2n+1$ Residuen kann auf diese Weise durch einen Vektor mit $20\cdot(2n+1)$ Komponenten beschrieben werden, wobei n die Anzahl der Nachbarn beiderseits des zentralen Residuums ist. Ein solcher Vektor für ein Residuum i ergibt sich zu:

$$\vec{x}_i = \left(\vec{a_{i-n}} \cdots \vec{a}_i \cdots \vec{a_{i+n}}\right)^T$$

Am Beispiel der frühen Entwicklung von SPARROW zeigt sich jedoch, dass eine so einfache Codierung der Sequenz, und damit auch von Sequenzfenstern, suboptimal ist [1]. Vielmehr erscheint es, dass der Verwendung von *positionsspezifischen Sequenzprofilen* Vorzug zu geben ist. Diese wurden durch Rost und Sander (PHD [76], [77]) als Repräsentation der Eingangsdaten für die Sekundärstrukturvorhersage eingeführt und werden in abgewandelter Form auch von PSIPRED [30] verwendet. Diese Profile aus einem *multiplen Sequenzalignment* erzeugt, was im Fall von PSIPRED mittels des Programms PSI-BLAST ([78], [79]) geschieht. Das Programm sucht innerhalb einer großen Datenbank von Proteinsequenzen nach Proteinen, die einer gegebenen Sequenz ähnlich sind. Auf der Grundlage der so gefundenen Proteinfamilie wird das Profil als eine Art mittlerer Sequenz erzeugt. Das Programm funktioniert iterativ, sodass mit dem gewonnenen Profil eine neue Anfrage durchgeführt werden kann, die ihrerseits ein neues Profil liefert. Wie oft dieser Vorgang wiederholt werden soll, kann vom Benutzer spezifiziert werden. Das Profil eines Proteins mit der Länge l ist eine Matrix der Größe $l\times 20$, wobei jedes Element dieser Matrix eine ganze Zahl darstellt. Jeder Sequenzposition innerhalb des Proteins kann somit eine Zeile des Profils zugeordnet werden, die ihrerseits zwanzig Einträge enthält. Zur Verdeutlichung soll Abbildung 33 auf der folgenden Seite herangezogen werden, in der das Beispiel des Ausschnitts eines Sequenzprofils in Tabellenform dargestellt ist. Die ersten beiden Spalten dieser Tabelle repräsentieren die Sequenz des Proteins, für welche das Profil erstellt wurde. Die erste Spalte stellt hierbei die fortlaufende Sequenzposition dar, während die zweite Spalte die eigentliche Sequenz in Form des Einbuchstabencodes enthält. Die zwanzig verbliebenen Spalten bilden das Profil, also die bereits erwähnte Matrix. Jede Spalte des Profils kann einer der zwanzig Standardaminosäuren zugeordnet werden. Diese Zuordnung ergibt sich aus der ersten Zeile der Tabelle, in der die Einbuchstabencodes der Standardaminosäuren verzeichnet sind. Die Elemente des Profils werden als proportional zur Wahrscheinlichkeit dafür interpretiert, dass eine Aminosäure an einer bestimmten Position innerhalb der Sequenz durch eine andere Aminosäure ersetzt werden könnte. Je positiver ein



		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	M	-1	-2	-2	-3	-2	-1	-2	-3	-2	1	2	-2	6	0	-3	-2	-1	-2	-1	1
2	V	0	-3	-3	-3	-1	-2	-3	-3	-3	3	1	-3	1	-1	-3	-2	0	-3	-1	4
3	N	-1	-2	6	0	-3	-1	-1	-1	-1	-4	-4	-1	-3	-4	-3	4	0	-5	-3	-4
4	W	-3	1	-4	-5	-3	-3	-3	-4	-3	-3	-1	-3	-2	-1	-4	0	-3	11	0	-4
5	A	4	-2	1	-2	3	1	-2	-2	-2	-3	-3	1	-2	-4	-2	1	-1	-4	-3	-2
6	A	4	-1	-2	-2	-3	4	1	-1	-2	-3	-3	1	-2	-4	-2	1	-1	-4	-3	0
7	V	-2	-4	1	-4	-2	-3	-4	-4	-4	2	2	-3	3	-2	-4	-3	-2	-4	-3	5
8	V	-1	2	-2	-3	-3	-2	-2	2	1	-1	-2	2	-2	-3	-3	1	1	-4	-3	3
9	D	0	-2	1	4	-4	-1	-1	0	5	-4	-4	2	-3	-4	-3	2	-2	-5	-2	-4
10	D	2	-3	1	6	-4	-2	0	-2	-2	-4	-4	-2	-4	-5	-3	1	-2	-5	-4	-3
11	F	-3	-4	-4	-5	-3	-4	-5	-5	-4	6	0	-4	0	4	-4	-3	0	-3	-1	2
12	Y	-3	-4	-5	-5	-3	-4	-4	-5	-3	0	4	-4	2	5	-5	-4	-3	-2	2	-1
13	Q	1	-3	-3	-3	-3	2	0	-1	-1	0	-1	-2	-2	-1	-3	-2	1	-2	5	2
14	E	4	1	-1	-2	-3	0	1	-2	-2	-3	-1	2	-2	-3	-3	1	0	-4	-1	-2
15	L	-3	-4	-5	-5	1	-4	-4	-5	-4	2	4	-4	2	3	-4	-4	-2	-3	-1	3
16	F	-4	-4	-5	-5	-4	-5	-5	-5	-3	0	0	-5	-1	8	-5	-4	-3	-1	2	-2
17	K	0	0	2	-2	-4	1	1	-3	-2	-4	-4	5	-3	-4	-3	-1	2	-4	-3	-3
18	A	4	2	-2	-1	-3	0	1	0	-2	-4	-3	1	-3	-4	-3	0	-2	-4	-3	-2
19	H	-3	-3	3	1	-4	-2	-2	-3	5	-3	-3	-2	-3	3	-4	-2	1	-2	5	-3
20	P	-2	-4	-3	-3	-4	-3	-3	-4	-4	-4	-4	-2	-4	-5	8	-2	-2	-5	-4	-4
21	E	-2	2	1	5	-4	0	3	-3	-2	-4	-4	-1	-4	-5	-3	1	-1	-5	-4	-4
22	Y	-2	-3	1	-3	-3	-3	-3	-4	-2	5	-1	-3	-1	0	-4	1	-2	-2	5	1
23	Q	0	0	-2	-2	-4	7	0	-3	-1	-3	-1	0	2	-4	0	-1	-2	-3	-3	-3
24	N	3	1	3	0	-3	2	1	-2	-2	-4	-4	2	-3	-4	-3	0	0	-4	-3	-3
25	K	-1	3	-2	-3	-4	-1	-2	-4	3	-3	-1	4	1	3	-3	-2	-2	-2	3	-3
26	F	-4	-4	-5	-5	-4	-5	-5	-5	-3	-1	-1	-5	-1	8	-5	-4	-4	0	2	-2
27	G	3	0	0	-2	-3	5	-1	0	-2	-4	-4	0	-2	-4	0	0	-2	-4	-3	-3
28	F	-4	-4	-5	-5	-4	-5	-5	-5	-3	-1	-1	-5	-1	8	-5	-4	-4	0	2	-2
29	K	4	2	-2	-3	-3	1	-1	-2	-3	-3	-3	3	1	-4	-3	0	-2	-4	-3	-1
....																					

Abbildung 33: *Darstellung eines Sequenzfensters als Profil*

Profilelement ist, desto wahrscheinlicher ist es, dass eine solche Ersetzung stattfinden kann. Je negativer der Wert, desto unwahrscheinlicher wird sie. So kann das Methionin in der ersten Sequenzposition am wahrscheinlichsten durch ein Methionin ersetzt werden, gefolgt vom Leucin. Am unwahrscheinlichsten wäre eine Ersetzung des Methionins durch Asparaginsäure, Glycin oder Prolin. Die einzelnen Profilzeilen sind, wie bereits erwähnt, positionsspezifisch, sodass einer bestimmten Aminosäure an einer Sequenzposition eine andere Profilzeile zugeordnet werden kann, als es für dieselbe Aminosäure an einer anderen Position der Fall wäre. So besitzen beispielsweise die Alanine an den Positionen 5, 6 und 18 jeweils unterschiedliche Profile. Entsprechendes gilt für das Valin in Position 7 und 8.

Da Sequenzprofile unter Berücksichtigung einer Vielzahl von Proteinen erzeugt werden, die eine hohe Sequenzähnlichkeit besitzen, enthalten sie auch indirekt Informationen über diese Proteine, vor allem, ob innerhalb der gefundenen Proteinfamilie bestimmte Residuen konserviert werden. Konservierte Residuen könnten wiederum laut Abschnitt 2.2 für die Struktur des Proteins essentiell sein. Weiterhin können Profile Informationen darüber enthalten, ob in bestimmten Teilen der Sequenz bestimmte Klassen von Aminosäuren (beispielsweise polare oder unpolare) bevorzugt werden, was Rückschlüsse auf die Struktur eines Proteins erlauben könnte. Somit enthalten Profile evolutionäre Informationen, die für

die Sekundärstrukturvorhersage von Proteinen von Vorteil sein können. Aus diesem Grund, wie auch aufgrund der Erfahrungen, die die Entwickler von PSIPRED und SPARROW mit PSI-BLAST-Profilen gemacht haben, werden auch für den in dieser Arbeit präsentierten Klassifikator besagte PSI-BLAST-Profile zur Repräsentation der Eingangsdaten verwendet. Ein Sequenzfenster (in Abbildung 33 als schwarzer Kasten um einen Ausschnitt der Sequenz symbolisiert) wird somit durch einen Vektor $\vec{x} \in \mathbb{Z}^{20 \cdot r}$ repräsentiert, der sich aus dem entsprechenden Ausschnitt der Profilmatrix zusammensetzt (roter Rahmen in der Abbildung). Hierbei ist $r = 2n + 1$ die Breite des Sequenzfensters in Residuen.

4.1.3.2. Definition der Sekundärstrukturklassen

In Abschnitt 4.1.2 wurde bereits angedeutet, dass die Daten bereits bekannter Proteinstrukturen aus der PDB zur Erzeugung einer Referenzklassifikation für die Sekundärstrukturvorhersage verwendet werden können. Diese Strukturdaten sind in Form von Atomkoordinaten vorhanden, wie sie aus der Röntgenkristallographie (Abschnitt 2.3.3.1) oder der NMR-Spektroskopie (Abschnitt 2.3.3.2) gewonnen werden. Bei der Sekundärstrukturvorhersage wird in der Regel die von DSSP (Abschnitt 2.3.3.3) aus den PDB-Daten berechnete Sekundärstruktur der einzelnen Residuen als Referenzklassifikation verwendet, wobei NMR-Strukturen wegen ihrer Mehrdeutigkeit zumeist nicht verwendet werden. Somit steht mit DSSP eine Klassifikation in acht Klassen bereit, die den in Abschnitt 2.3.4 vorgestellten Sekundärstrukturtypen entsprechen. Jedem dieser Sekundärstrukturtypen weist DSSP ein Zeichen zu, dass in der erzeugten Zeichenkette zur Repräsentation der Sekundärstruktur für den entsprechenden Sekundärstrukturtyp steht. Dabei erhält jeder Sekundärstrukturtyp einen Buchstaben, während strukturlose Regionen durch ein Leerzeichen gekennzeichnet sind. Dieser DSSP-Code soll innerhalb dieser Arbeit zur einfachen Codierung der acht DSSP-Klassen verwendet werden, wobei jedoch strukturlose Regionen durch ein C statt eines Leerzeichens codiert werden sollen. Der Übersichtlichkeit halber sind in Tabelle 2 beide Codierungen für die acht Sekundärstrukturtypen aufgeführt.

	3_{10}-Helix	α-Helix	π-Helix	β-Strang	β-Brücke	Windung	Biegung	strukturlos
DSSP-Code	'G'	'H'	'I'	'E'	'B'	'T'	'S'	' '
Klassencode	G	H	I	E	B	T	S	C

Tabelle 2: *Codierung der acht DSSP-Klassen*

Da DSSP keine ganzen β-Faltblätter, sondern lediglich die Stränge (Abschnitt 2.3.4.4) er-

kennt, aus denen sie gebildet werden, ist in der Tabelle der β-Strang als Klasse aufgeführt. Weiterhin werden isolierte β-Brücken (Abschnitt 2.3.4.5) der Kürze halber als β-Brücken aufgeführt, wie wasserstoffbrückengebundene Windungen (Abschnitt 2.3.4.6) entsprechend nur als Windungen bezeichnet sind. In der Zeile „DSSP-Code“ ist die Codierung der Sekundärstrukturtypen nach DSSP dargestellt. Darunter ist die für diese Arbeit verwendete Codierung der entsprechenden Klassen als Buchstaben, wie auch als farbliche Codierung, die bereits in Abschnitt 2.3.4 in den Darstellungen der einzelnen Sekundärstrukturtypen verwendet wurde, abgebildet.

Zur direkten Nachbildung des DSSP-Codes mit Hilfe eines Klassifikators ist es notwendig, das entsprechende Achtklassenproblem zu lösen. Diese durchaus komplexe Problemstellung wird jedoch von relativ wenigen Programmen zur Sekundärstrukturvorhersage, wie etwa SSpro8 als Variante von SSpro [29], angegangen. Vielmehr sagen die meisten Programme lediglich drei Klassen voraus, nämlich Helices, β-Stränge und strukturlose Regionen. Das so entstandene vereinfachte Dreiklassenproblem ist leichter lösbar als das Achtklassenproblem, erfordert jedoch eine geeignete Reduktion der acht DSSP-Klassen auf drei Klassen. Dies wird durch Vereinigung bestimmter Klassen bewerkstelligt, wobei unterschiedliche Möglichkeiten der Vereinigung existieren. Welche Möglichkeit gewählt wird, hängt davon ab, als was die drei resultierenden Klassen betrachtet werden. Soll die helikale Klasse beispielsweise ausschließlich α-Helices enthalten, also eine reine α-Helixklasse sein, und die β-Strangklasse auch nur ausschließlich β-Stränge beinhalten, so werden die anderen sechs Klassen vereinigt und bilden somit die neue strukturlose Klasse. Aufgrund der strengen Handhabung der acht DSSP-Klassen soll diese Klasseneinteilung als *strenge Klasseneinteilung* bezeichnet werden. Das angeführte Beispiel stellt lediglich eine von mehreren sinnvollen Möglichkeiten dar, wie die acht DSSP-Klassen zu drei Klassen vereinigt werden können, welche beispielsweise als eine mögliche Klasseneinteilung für SSpro in [29] getestet wurde. Diese Vielfalt an Möglichkeiten würde es erschweren, einzelne Programme zur Sekundärstrukturvorhersage bei Verwendung unterschiedlicher Klasseneinteilungen untereinander zu vergleichen, da für jedes Programm eine andere Klassifikation ein und desselben Residuums als richtig angesehen werden würde. Um einen Schritt hin zur automatischen Vergleichbarkeit von Methoden zu machen wurde deshalb das EVA-Projekt ins Leben gerufen. EVA (*Evaluation of automatic structure prediction servers*, [80], [81], [82]) sollte die einzelnen Methoden zur Sekundärstrukturvorhersage, die Anfra-

gen über Webserver entgegennehmen, auf automatische Weise vergleichen. Hierzu sollten automatisch Anfragen mit Proteinen durchgeführt werden, deren Struktur zum Zeitpunkt der Anfrage neu war, also erst kürzlich bestimmt wurde. Die Vorhersagen für diese Proteinstrukturen sollten dazu verwendet werden, die verschiedenen Methoden miteinander zu vergleichen. Zu diesem Zweck wurde eine Klasseneinteilung definiert, anhand derer die Korrektheit der einzelnen Vorhersagen einheitlich bestimmt werden sollte. Diese behandelt die helikale Klasse als die Vereinigung aller helikalen DSSP-Klassen. Die β-Strangklasse wird aus den β-Strängen selbst, sowie den β-Brücken, die als extrem kurze oder fehlgeschlagene β-Stränge interpretiert werden können (Abschnitt 2.3.4.5), gebildet. Schließlich setzt sich die strukturlose Klasse aus Biegungen, Windungen und der eigentlichen strukturlosen DSSP-Klasse zusammen. Zur Zeit der Entstehung dieser Arbeit schien es, dass das EVA-Projekt bereits vor einiger Zeit eingestellt worden war. Jedoch wird die vom EVA-Projekt konzipierte Klasseneinteilung noch heute teilweise verwendet (beispielsweise bei SPARROW und SSpro) und soll auch im Rahmen dieser Arbeit verwendet werden. Diese Klasseneinteilung soll als *lockere Klasseneinteilung* bezeichnet werden, da sie die DSSP-Klassen im Kontrast zur strengen Klasseneinteilung nach lockeren Maßstäben vereinigt. Tabelle 3 liefert einen Überblick über die bisher erwähnten Klasseneinteilungen.

	3_{10}-Helix	α-Helix	π-Helix	β-Strang	β-Brücke	Windung	Biegung	strukturlos
vollständig	G	H	I	E	B	T	S	C
locker	H	H	H	E	E	C	C	C
streng	H	C	C	E	C	C	C	C

Tabelle 3: *Klasseneinteilungen im Überblick*

Damit ergeben sich also drei Klasseneinteilungen, von denen jede für sich genommen ein eigenes Mehrklassenproblem darstellt. Die als vollständig bezeichnete Klasseneinteilung ist eine Klassifikation, welche die DSSP-Klassen unverändert übernimmt und damit das Achtklassenproblem darstellt. Die lockere und die strenge Klasseneinteilung bilden jeweils ein Dreiklassenproblem. Im Fall des Dreiklassenproblems wird, wie der Tabelle entnommen werden kann, die helikale Klasse durch den Buchstaben 'H' und Rot als Symbolfarbe repräsentiert. Entsprechend sollen für die β-Strangklasse der Buchstabe 'E' sowie die Farbe Blau verwendet werden und für die strukturlosen Regionen 'C' beziehungsweise Grün. Mittels des in dieser Arbeit entwickelten Klassifikators, der im nächsten Abschnitt vorgestellt wird, soll jedes der drei Klassifikationsprobleme angegangen werden.

4.2. Der vektorwertige Klassifikator

Um ein Mehrklassenproblem wie das der Sekundärstrukturvorhersage zu lösen gibt es prinzipiell zwei Herangehensweisen: Entweder die Lösung von Teilproblemen in Form von Zweiklassenproblemen (Abschnitt 3.4.1) oder aber eine direkte Lösung des Mehrklassenproblems mit einem entsprechenden Klassifikator, wie einem MLP (Abschnitt 3.5.4.2) oder einem Bayes-Klassifikator (Abschnitt 3.5.3). Wie bereits in Abschnitt 4.1.1 angemerkt, bietet der erste Weg nicht mehr viel Potential für neue Entwicklungen, die sich von bereits existierenden Methoden unterscheiden. Aus diesem Grund geht der in dieser Arbeit vorgestellte Klassifikator den zweiten Weg und löst das Mehrklassenproblem als solches. Hierzu wurde ein neuartiges Verfahren verwendet, das auf der Lerntheorie und einigen geometrischen Überlegungen basiert. Dieses Verfahren wird in den folgenden Abschnitten beschrieben.

4.2.1. Einführung

Normalerweise reduzieren Verfahren, welche ein Mehrklassenproblem lösen, die eigentliche Klassifikation auf einen Vergleich von einigen wenigen Werten. Bei einem Mehrklassenproblem mit $N^{Klassen}$ Klassen aus einer Menge $Z=\{\zeta_j | 1 \le j \le N^{Klassen}\}$ wird die Entscheidung mit Hilfe von $N^{Klassen}$ Werten durchgeführt, die über entsprechende Entscheidungsregeln miteinander Verknüpft werden (Abschnitt 3.4.2). Bei Zweiklassenproblemen wiederum genügt ein Wert, um eine Klassifikation vornehmen zu können (Abschnitt 3.3). Es stellt sich nun die Frage, ob ein prinzipieller Unterschied zwischen einem Mehrklassenproblem und einem Zweiklassenproblem besteht. Grundsätzlich könnte das Zweiklassenproblem als ein Sonderfall des Mehrklassenproblems gesehen werden, bei dem die Anzahl der Klassen $N^{Klassen}=2$ ist. Auf diese Weise bestünde kein konzeptioneller Unterschied zwischen Zwei- und Mehrklassenproblemen. Jedoch ergibt sich eine gewisse Inkonsistenz, da Mehrklassenprobleme $N^{Klassen}$ Werte brauchen, um eine Klassifikation durchzuführen, während es bei einem Zweiklassenproblem genügt $N^{Klassen}-1$, also einen Wert, in Betracht zu ziehen. Zwar ist es durchaus möglich eine Klassifikation in zwei Klassen durchzuführen, indem beispielsweise die beiden Wahrscheinlichkeiten dafür verglichen werden, dass das zu klassifizierende Objekt zu einer der beiden Klassen gehört. So eine Herangehensweise wird jedoch, außer im Fall des Bayes-Klassifikators, selten verwendet. Einfache lineare Klassifikatoren, wie Fishers lineare Diskriminanzfunktion (Abschnitt 3.5.2) oder

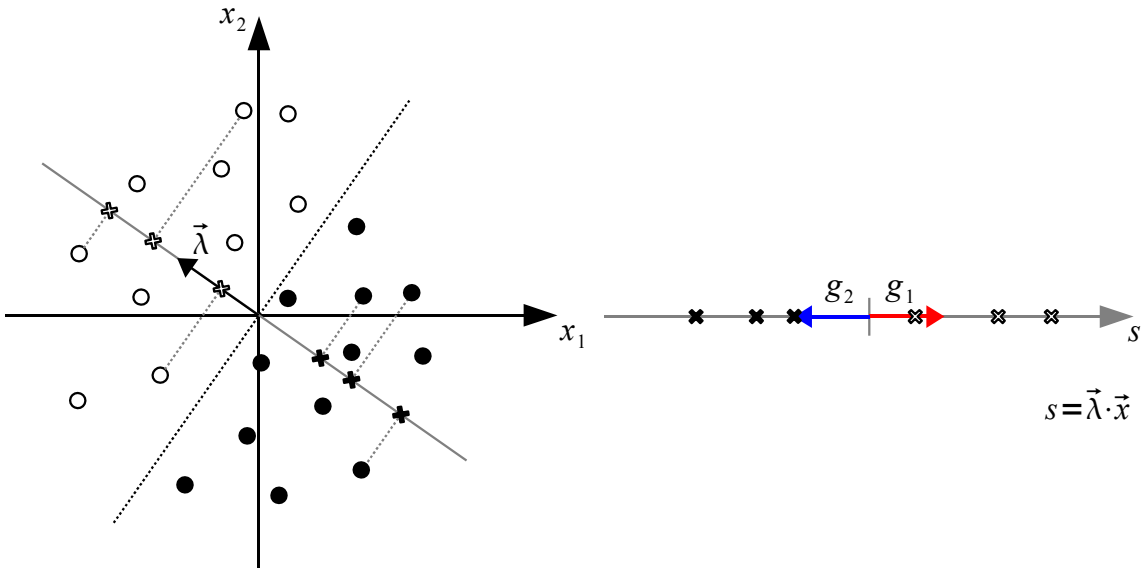


Abbildung 34: *Reduktion eines zweidimensionalen Zweiklassenproblems (links) auf ein eindimensionales Problem (rechts) durch Projektion auf eine Gerade mit dem Richtungsvektor* $\vec{\lambda}$ *(grau)*

Rosenblatts Perzeptron (Abschnitt 3.5.4.1), wie auch weiterentwickelte Klassifikatoren, wie Vapniks SVMs (Abschnitt 3.5.4.4), kommen mit einem Wert aus, anhand dessen eine Klassifikation vorgenommen werden kann. Diese Tatsache wirft nun eine interessante Frage auf: Wenn das Zweiklassenproblem ein besonderer Fall des Mehrklassenproblems mit $N^{Klassen}=2$ Klassen ist, das in einem Raum mit $N^{Klassen}-1=1$ Dimensionen gelöst werden kann, lässt sich dann ein beliebiges Mehrklassenproblem mit $N^{Klassen}$ Klassen in einem Raum mit $N^{Klassen}-1$ Dimensionen lösen? Zudem stellt sich als zweite Frage, wie ein Klassifikator aussehen könnte, der ein Mehrklassenproblem auf die beschriebene Weise löst.

Um einer Antwort auf die zweite Frage näher zu kommen, soll zunächst Fishers lineare Diskriminanzfunktion betrachtet werden. Wie Abschnitt 3.5.2 entnommen werden kann, ist eine der graphischen Interpretationen des Verfahrens, das Fisher verwendet hat, dass die zu klassifizierenden Daten auf eine Gerade im Raum, also in einen eindimensionalen Raum, der durch die Gerade beschrieben wird, projiziert werden. Dies ist in Abbildung 34 für das Beispiel eines Zweiklassenproblem im zweidimensionalen Raum visualisiert, wobei links die Verteilung der Daten im zweidimensionalen Raum samt Klassifikationsgrenze (gestrichelte Linie), dem dazugehörigen Normalvektor $\vec{\lambda}$ und der daraus resultierenden Geraden, auf die die Daten projiziert werden (graue Linie), dargestellt ist. Auf der rechten Seite ist

der dazugehörige eindimensionale Raum dargestellt, der sich durch die Projektion auf die Gerade ergibt und durch den grauen Zahlenstrahl repräsentiert wird. Projektionen der Daten auf die Gerade sind in Abbildung 34 für einige ausgewählte Beispiele durch Kreuze symbolisiert, deren Färbung der Klassenzugehörigkeit entspricht. Die Klassifikation dieser Daten wird durch die relative Lage der projizierten Daten hinsichtlich eines definierten Bezugspunkts oder Nullpunkts beschrieben (in Abbildung 34 auf der rechten Seite an einer senkrechte Markierung am Zahlenstrahl erkennbar). In einem solchen eindimensionalen Raum kann die Lage eines Punktes durch einen Skalar beschrieben werden, welcher prinzipiell als ein „eindimensionaler Vektor“ angesehen werden könnte. Dieser Skalar ergibt sich aus dem Skalarprodukt $s = \vec{\lambda} \cdot \vec{x}$ der Eingangsdaten $\vec{x} \in \mathbb{R}^2$ mit dem Vektor $\vec{\lambda}$, der die Richtung der Geraden im zweidimensionalen Raum angibt (siehe Abbildung 34 links). Die Klassifikation wird anhand des Vorzeichens von s vorgenommen. Betrachtet man diesen Skalar als eindimensionalem Vektor, so kann diese Entscheidung folgendermaßen interpretiert werden: Im eindimensionalen Raum gibt es zwei „Einheitsvektoren“ die durch die Skalare $g_1 = 1$ und $g_2 = -1$ (in Abbildung 34 durch einen roten beziehungsweise blauen Pfeil gekennzeichnet) definiert sind. Diese Vektoren haben den maximalen Abstand, den zwei Einheitsvektoren im eindimensionalen Raum zueinander haben können, nämlich 2. Die Klassifikation mittels des Vorzeichens von s entspricht dem Vergleich der Produkte $s \cdot g_1$ und $s \cdot g_2$, wobei der durch s repräsentierte Vektor $\vec{x}$ einer der beiden Klassen in Abhängigkeit davon zugewiesen wird, welches der beiden Produkte größer ist.

Allgemein sei nun die Projektion $s = f(\vec{x})$ eines Punktes $\vec{x} \in \mathbb{R}^n$ aus dem n-dimensionalen in den eindimensionalen Raum durch die Klassifikatorfunktion $f(\vec{x})$ gegeben. Die Klassifikation dieses Punktes kann unter Berücksichtigung der Produkte von s mit den Einheitsvektoren g_1 und g_2 folgendermaßen vorgenommen werden:

$$\vec{x} \in \zeta_\alpha \Leftrightarrow \alpha = \underset{j}{\operatorname{argmax}} \left(s \cdot g_j \right)$$

Die beiden Einheitsvektoren g_1 und g_2 bestimmen die jeweiligen Klassen in diesem Fall vollständig und sollen deshalb im weiteren Text als *Klassenvektoren* bezeichnet werden.

Mit Hilfe dieser für das Zweiklassenproblem recht komplizierten Darstellung kann nun ein allgemeiner Ansatz zur Lösung von Klassifikationsproblemen mit einer beliebigen Anzahl von Klassen definiert werden. Ausgegangen werden soll von einem Satz von Daten, die

durch die Vektoren $\vec{x}_i \in \mathbb{R}^n$ mit $1 \le i \le N$ repräsentiert werden, von denen jeder eindeutig jeweils einer Klasse aus der Menge $Z = \{\zeta_j | 1 \le j \le N^{Klassen}\}$ zugeordnet werden kann. Analog zur obigen Betrachtung der linearen Diskriminanzfunktion kann nun angenommen werden, dass dieses Klassifikationsproblem mit einer geeigneten Projektion in einen Vektorraum mit $N^{Klassen} - 1$ Dimensionen gelöst werden kann. Diese Projektion soll mittels einer vektorwertigen Funktion der folgenden Form durchgeführt werden:

$$\vec{f} : \mathbb{R}^n \to \mathbb{R}^{N^{Klassen}-1}$$

Diese Funktion stellt eine allgemeine Funktion dar, die von $\mathbb{R}^n$ nach $\mathbb{R}^{N^{Klassen}-1}$ abbildet, und muss nicht zwangsläufig eine lineare Projektionsfunktion sein. Um die Klassifikation eines Vektors $\vec{x}$ vornehmen zu können soll das Skalarprodukt mit allen Klassenvektoren gebildet werden und $\vec{x}$ nach folgender Regel klassifiziert werden:

$$\vec{x} \in \zeta_\alpha \Leftrightarrow \alpha = \underset{j}{\operatorname{argmax}} (\vec{s} \cdot \vec{g}_j) \text{ , mit } \vec{s} = \vec{f}(\vec{x})$$

Dies ist die allgemeine vektorielle Darstellung der Entscheidungsregel, die in diesem Abschnitt bereits für das Zweiklassenproblem beschrieben wurde. Der vorgestellte Ansatz ist somit ein Lösungsansatz für Klassifikationsprobleme mit einer beliebigen Anzahl an Klassen, wobei auch das Zweiklassenproblem eingeschlossen ist. Das Setzen von $N^{Klassen} = 2$ ergibt die weiter oben beschriebene Projektion in einen eindimensionalen Raum. An diesem Punkt ist jedoch weder klar, wie die Klassifikatorfunktion $\vec{f}(\vec{x})$ bestimmt ist, noch welche Eigenschaften die Klassenvektoren $\vec{g}_k$ besitzen sollen. Diesen Fragen soll in den nächsten beiden Abschnitten nachgegangen werden.

4.2.2. Klassenvektoren

Zunächst wäre die Frage zu klären, wie die Klassenvektoren bestimmt sein sollen. Aus der Analogie im eindimensionalen Raum können die Eigenschaften der Klassenvektoren für den allgemeinen Fall abgeleitet werden. Diese Eigenschaften wären:

- Für ein Klassifikationsproblem mit $N^{Klassen}$ Klassen gibt es $N^{Klassen}$ Klassenvektoren $\vec{g}_k$
- Für alle Klassenvektoren $\vec{g}_k$ soll $\vec{g}_k \in \mathbb{R}^{N^{Klassen}-1}$ gelten
- Die Klassenvektoren sollen Einheitsvektoren sein

- Die Klassenvektoren sollen die maximal möglichen Abstände zueinander haben

Der erste Punkt ist an sich trivial und soll nur der Vollständigkeit halber erwähnt werden. Da jede Klasse durch einen eindeutigen Klassenvektor beschrieben werden soll, ist es natürlich auch notwendig, dass es genauso viele unterschiedliche Klassenvektoren gibt wie unterschiedliche Klassen im betreffenden Klassifikationsproblem existieren. Der zweite Punkt ergibt sich aus der Definition der Klassifikatorfunktion und muss aus diesem Grund nicht weiter diskutiert werden. Der dritte Punkt ist keine Notwendigkeit, soll jedoch der Einfachheit halber realisiert werden. Tatsächlich ist es nur wichtig, dass alle Klassenvektoren die gleiche Länge besitzen, sodass das Skalarprodukt, als Ähnlichkeitsmaß zu den einzelnen Klassenvektoren, aussagekräftig ist. Während die ersten drei Punkte ohne weiteres realisierbar sind, ist es der letzte Punkt, der weitere Betrachtungen erfordert. Im eindimensionalen Raum ist, wie bereits erwähnt, der maximale Abstand beider Klassenvektoren dadurch gewährleistet, dass beide Vektoren gegensätzliche Richtungen (in diesem Fall Vorzeichen) haben. Ausgehend von einem Dreiklassenproblem entsteht jedoch ein zweidimensionaler Raum, also eine Ebene, in die die zu klassifizierenden Daten projiziert werden. In diesem Raum müssen nun die drei Klassenvektoren derart angeordnet werden, dass jeder Vektor den maximalen Abstand zu den beiden verbliebenen Vektoren besitzt. Als Lösung dieses Problems bieten sich die Eckpunkte eines gleichseitigen Dreiecks an, welches einem Kreis um den Koordinatenursprung einbeschrieben ist. Auf diese Weise ist gewährleistet, dass alle drei Vektoren den gleichen Abstand zueinander haben. Zudem ist dies der maximal mögliche Abstand zwischen den Vektoren, da es nicht möglich ist, den Abstand zweier Vektoren zu vergrößern, ohne dass einer der beiden Vektoren dem verbliebenen Vektor näher kommt oder die Länge der Vektoren verändert werden muss. Eine mögliche Anordnung der drei Klassenvektoren im zweidimensionalen Raum ist in Abbildung 35 auf der folgenden Seite dargestellt. Die Klassenvektoren sind unterschiedlich gefärbt, wobei die Färbung der einzelnen Vektoren den drei Sekundärstrukturklassen der lockeren oder strengen Klasseneinteilung (Abschnitt 4.1.3.2) entsprechend gewählt wurde. Weiterhin ist das beschriebene Dreieck, zu dessen Eckpunkten die Klassenvektoren zeigen, als gestrichelte Linie in der Abbildung eingezeichnet. Der Kreis, dem das Dreieck einbeschrieben ist, stellt in diesem Fall einen Einheitskreis dar, wodurch die hier dargestellten Klassenvektoren Einheitsvektoren sind und somit zusammen mit ihren anderen Eigenschaften (Dimension, Anzahl und Abstand zueinander) alle Anforderungen erfüllen, die

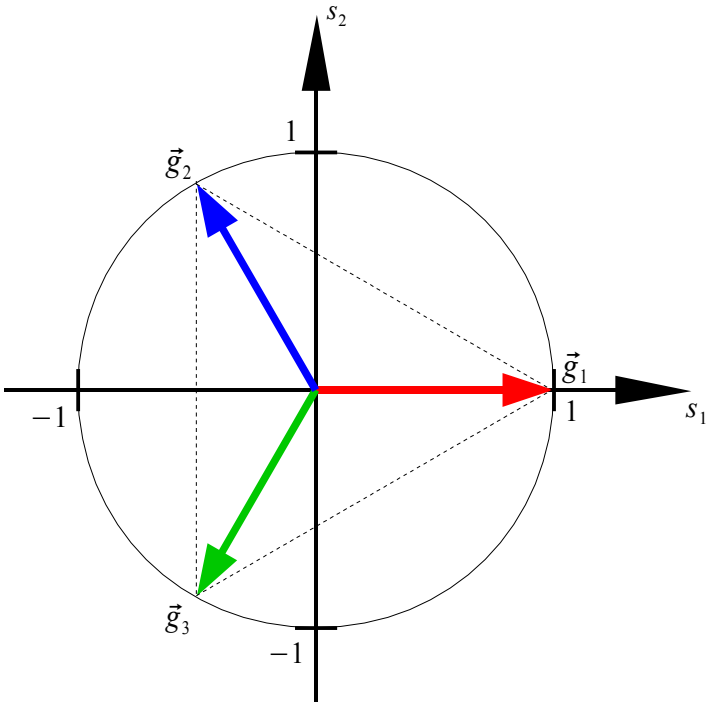


Abbildung 35: *Die Klassenvektoren für das Dreiklassenproblem*

am Anfang dieses Abschnitts an sie gestellt wurden.

Der nächste Schritt, das Vierklassenproblem, ergibt nach ähnlichen Überlegungen wie beim Dreiklassenproblem, dass in diesem Fall die Klassenvektoren derart angeordnet sein müssen, dass sie die Eckpunkte eines regelmäßigen Tetraeders bilden, das einer Einheitskugel um den Koordinatenursprung einbeschrieben ist. Auch in diesem Fall haben alle Vektoren den gleichen Abstand zueinander. Es lässt sich eine Verbindung zwischen den drei bisher beschriebenen geometrischen Figuren ziehen, die zur Beschreibung der Klassenvektoren verwendet werden. Diese Verbindung stellt das so genannte *regelmäßige Simplex* dar, das einer bestimmten Art des *regelmäßigen Polytops*, also eines regelmäßigen mehrdimensionalen Körpers, entspricht [83] und eine Verallgemeinerung des regelmäßigen Tetraeders auf Räume beliebiger Dimension ist: Eine Gerade ist ein 1-Simplex, ein gleichseitiges Dreieck entspricht einem regelmäßigen 2-Simplex und ein regelmäßiges Tetraeder schließlich einem regelmäßigen 3-Simplex. Die Zahl gibt in diesem Fall die Zahl der Dimensionen des Raums an, in dem sich das Simplex befindet. Für die Klassenvektoren würde das nun bedeuten, dass eine weitere Erhöhung der Dimension (und damit der Anzahl der Klassen) aus dem regelmäßigen Tetraeder ein regelmäßiges 4-Simplex und aus der Einheitskugel eine Einheitshyperkugel macht. Im allgemeinen Fall mit $N^{Klassen}$ Klas-

sen würden die Klassenvektoren zu den Eckpunkten eines regelmäßigen $(N^{Klassen}-1)$-Simplex zeigen, dessen Zentrum im Koordinatenursprung liegt. Somit ist eine allgemeine Verteilung der Klassenvektoren für beliebige Klassifikationsprobleme beschrieben, sodass als nächstes ein Algorithmus zur Berechnung dieser Klassenvektoren für beliebige Klassifikationsprobleme beschrieben werden kann.

Eine interessante Eigenschaft der Klassenvektoren zur Charakterisierung der Klassenzugehörigkeit ist, dass sie es dem Klassifikator auch ermöglichen nur eine eingeschränkte Klassifikation vorzunehmen. Wenn beispielsweise im Fall eines Vierklassenproblems, bei dem die Klassenvektoren die Eckpunkte eines regelmäßigen Tetraeders bilden, der Klassifikator eine der vier Klassen ausschließen kann, so zeigt die Klassifikatorfunktion in das Zentrum der entsprechenden Tetraederfläche, die dem Eckpunkt gegenüberliegt, der diese Klasse beschreibt. Analog führt der Ausschluss zweier Klassen dazu, dass die Klassifikatorfunktion in die Mitte der Tetraederkante zeigt, die von den Klassenvektoren der beiden übrigen Klassen aufgespannt wird. Entsprechende Verallgemeinerungen dieser Betrachtungen gelten für beliebige Mehrklassenprobleme.

4.2.2.1. Berechnung der Klassenvektoren

An dieser Stelle soll ein Algorithmus gezeigt werden, mit dessen Hilfe die Klassenvektoren, wie sie in Abschnitt 4.2.2 beschrieben wurden, für beliebige Klassifikationsprobleme mit $N^{Klassen}$ Klassen berechnet werden können. Der Algorithmus geht davon aus, dass das Zentrum des regelmäßigen d-Simplex mit $d = N^{Klassen}-1$, zu dessen Eckpunkten die einzelnen Klassenvektoren $\vec{g}_k$ zeigen, im Ursprung des $\mathbb{R}^d$ mit der kartesischen Basis $\left\{\vec{e}_j \in \mathbb{R}^d \middle| 1 \le j \le d\right\}$ mit $\vec{e}_i \cdot \vec{e}_j = \delta_{i,j}$ ruht. Daraus folgend ergibt sich, dass die gesuchten Klassenvektoren $\left\{\vec{g}_k \in \mathbb{R}^d \middle| 1 \le k \le d+1\right\}$, die Abschnitt 4.2.2 entsprechend Einheitsvektoren sind, folgenden Bedingungen bezüglich des Skalarprodukts genügen müssen:

I. $\forall\, \rho : \vec{g}_\rho \cdot \vec{g}_\rho = 1$

II. $\forall\, \rho \neq \sigma : \vec{g}_\rho \cdot \vec{g}_\sigma = -\frac{1}{d}$

Das Skalarprodukt in Bedingung II., welches dem Kosinus des Winkels zwischen zwei Klassenvektoren entspricht, ergibt sich aus der Geometrie des regelmäßigen d-Simplex [84]. Der erste Klassenvektor $\vec{g}_1$ kann mit der Einschränkung, dass er Bedingung I. erfül-

len muss, prinzipiell frei gewählt werden und soll für die folgende Berechnung zu $\vec{g}_1 = \vec{e}_1$ gesetzt werden. Daraus ergibt sich, dass für $\vec{g}_2$ gelten muss:

$$\vec{g}_2 = -\frac{1}{d}\vec{g}_1 + a_2\vec{e}_2$$

Aus $\vec{g}_2 \cdot \vec{g}_2 = 1$ ergibt sich $a_2 = \frac{\sqrt{d^2-1}}{d}$ und damit ist auch $\vec{g}_1 \cdot \vec{g}_2 = -1/d$ erfüllt, das heißt $\vec{g}_1$ und $\vec{g}_2$ erfüllen Bedingung I. und II. Als nächstes ergibt sich $\vec{g}_3$ aus:

$$\vec{g}_3 = -\frac{1}{d-1}\left(\vec{g}_1 + \vec{g}_2\right) + a_3\vec{e}_3$$

Eine einfache Überprüfung zeigt, dass $\vec{g}_3$ Bedingung II. erfüllt:

$$\vec{g}_3 \cdot \vec{g}_i = -\frac{1}{d-1}\left(1 - \frac{1}{d}\right) = -\frac{1}{d} \text{ für } i = 1,2$$

Die reelle Zahl a_3 ergibt sich aus der für Bedingung I. notwendigen Normierung zu:

$$\vec{g}_3 \cdot \vec{g}_3 = \frac{2}{(d-1)^2}\left(1 - \frac{1}{d}\right) + a_3{}^2 = 1 \;\Rightarrow\; a_3 = \sqrt{1 - \frac{2}{d(d-1)}}$$

Die bisher präsentierten Berechnungen zeigen die iterative Natur des Algorithmus: Sind k Klassenvektoren berechnet, für die die Bedingungen I. und II. gelten, so wird daraus der nächste Klassenvektor $\vec{g}_{k+1}$ konstruiert, der seinerseits beide Bedingungen erfüllt. Der allgemeine Ausdruck zur Berechnung von $\vec{g}_{k+1}$, der sich auf diese Weise ergibt, ist:

$$\vec{g}_{k+1} = -\frac{1}{d-k+1}\sum_{j=1}^{k}\vec{g}_j + a_{k+1}\vec{e}_{k+1}$$

Dieser Ausdruck erfüllt Bedingung II., wie die folgende Rechnung zeigt:

$$\vec{g}_{k+1} \cdot \vec{g}_i = -\frac{1}{d-k+1}\left(1 - \frac{k-1}{d}\right) = -\frac{1}{d}, \; 1 \le i \le k$$

Zur Berechnung von a_{k+1} muss die Normierung von $\vec{g}_{k+1}$ berücksichtigt werden (Bedingung I.). Auf diese Weise ergibt sich für a_{k+1}:

$$\vec{g}_{k+1} \cdot \vec{g}_{k+1} = \frac{1}{(d-k+1)^2}\left(k - \frac{k(k-1)}{d}\right) = 1 \Rightarrow a_{k+1} = \sqrt{1 - \frac{k}{d(d-k+1)}}$$

Da die Klassenvektoren aus $\mathbb{R}^d$ stammen, in dem kein Basisvektor $\vec{e}_{d+1}$ existiert, ist der allgemeine Ausdruck für $\vec{g}_{k+1}$ für $k = d$ nicht definiert. Aus diesem Grund kann der letzte Klassenvektor $\vec{g}_{d+1}$ nicht ohne weiteres berechnet werden. Da im $\mathbb{R}^d$ jedoch maximal d

linear unabhängige Vektoren gefunden werden können und die Klassenvektoren $\vec{g}_k$ mit $1 \le k \le d$ aufgrund ihrer Konstruktion bereits linear unabhängig sind, muss $\vec{g}_{d+1}$ von den übrigen Klassenvektoren linear abhängig sein. Die geometrischen Eigenschaften des regelmäßigen d-Simplex legen nahe, dass für $\vec{g}_{d+1}$ folgendes gelten muss:

$$\vec{g}_{d+1} = -\sum_{j=1}^{d} \vec{g}_j$$

Wie die Berechnung des Skalarprodukts $\vec{g}_{d+1} \cdot \vec{g}_i$ zeigt, erfüllt $\vec{g}_{d+1}$ Bedingung II.:

$$\vec{g}_{d+1} \cdot \vec{g}_i = -\left(1 - \frac{d-1}{d}\right) = -\frac{1}{d} \text{ , mit } 1 \le i \le k$$

Die Berechnung des Betrags von $\vec{g}_{d+1}$ ergibt, dass auch Bedingung I. erfüllt ist:

$$\vec{g}_{d+1} \cdot \vec{g}_{d+1} = d - d(d-1) \cdot \frac{1}{d} = 1$$

Damit erfüllen alle $\vec{g}_k$ mit $1 \le k \le d+1$ die geforderten Bedingungen I. und II.. Der vorgestellte Algorithmus eignet sich somit zur Berechnung der Eckpunkte eines regelmäßigen d-Simples in $\mathbb{R}^d$ mit $d = N^{Klassen} - 1$ und $N^{Klassen} > 1$ und kann verkürzt durch die folgenden Schritte beschrieben werden:

1. Initialisierung: $\vec{g}_1 = \vec{e}_1$
2. Iteration: $\vec{g}_{k+1} = -\frac{1}{d-k+1} \sum_{j=1}^{k} \vec{g}_j + \sqrt{1 - \frac{k}{d(d-k+1)}}\, \vec{e}_{k+1}$, für alle $1 \le k \le d-1$
3. Abschluss: $\vec{g}_{d+1} = -\sum_{j=1}^{d} \vec{g}_j$

Dieser Algorithmus stellt nur eine von zahlreichen Möglichkeiten dar, die gesuchten Klassenvektoren zu berechnen. Das Problem der Berechnung der Eckpunkte eines regelmäßigen $(N^{Klassen} - 1)$-Simplex besitzt unendlich viele mögliche Lösungen, da jede beliebige Rotation einer gefundenen Lösung um den Koordinatenursprung ebenfalls eine gültige Lösung liefert. Eine Veranschaulichung der Klassenvektoren, die der verwendete Algorithmus berechnet, ist in Abbildung 36 auf der folgenden Seite zu sehen. Dort sind die darstellbaren Fälle $2 \le N^{Klassen} \le 4$ auf einem Zahlenstrahl $(N^{Klassen} = 2)$, in der Ebene $(N^{Klassen} = 3)$ und im dreidimensionalen Raum $(N^{Klassen} = 4)$ dargestellt. Die einzelnen Klassenvektoren sind entsprechend der Legende, die sich im unteren linken Bereich der

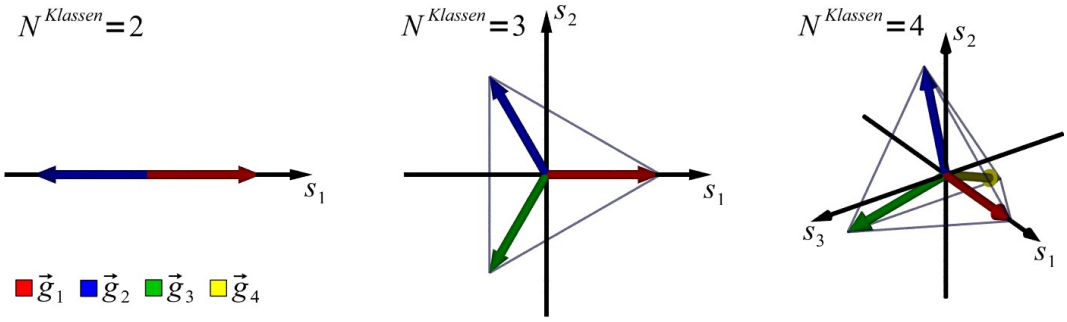


Abbildung 36: *Darstellung von Klassenvektoren für verschiedene Klassifikationsprobleme*

Abbildung befindet, farblich gekennzeichnet.

Mit Hilfe der so berechneten Klassenvektoren kann nun mittels der Klassifikatorfunktion $\vec{f}(\vec{x})$ die Klassifikation eines Punktes $\vec{x}$ zu einer Klasse ζ_α entsprechend Abschnitt 4.2.1 vorgenommen werden:

$$\vec{x} \in \zeta_\alpha \Leftrightarrow \alpha = \underset{j}{\operatorname{argmax}} \left(\vec{f}(\vec{x}) \cdot \vec{g}_j \right)$$

Diese Entscheidungsregel stellt eine Form der Regel der maximalem Konfidenz (siehe Abschnitt 3.4.2) dar. Ausgehend vom Skalarprodukt als Ähnlichkeitsmaß zu den Klassenvektoren, sollte die Ähnlichkeit von $\vec{f}(\vec{x})$ mit $\vec{g}_j$ proportional zur Wahrscheinlichkeit sein, dass $\vec{x}$ der Klasse ζ_j angehört. Die bisherigen Betrachtungen lassen jedoch noch immer die Frage offen, wie die Klassifikatorfunktion $\vec{f}(\vec{x})$ beschaffen ist. Mit dieser Fragestellung soll sich der folgende Abschnitt befassen.

4.2.3. Klassifikatorfunktion

Aus Abschnitt 4.2.1 ist bereits bekannt, dass die Klassifikatorfunktion $\vec{f}(\vec{x})$ folgendermaßen definiert sein muss:

$$\vec{f} : \mathbb{R}^n \rightarrow \mathbb{R}^{N^{Klassen}-1}$$

Hierbei ist $\mathbb{R}^n$ der Raum aus dem die Eingangsdaten bezogen werden. Nun gibt es prinzipiell viele Möglichkeiten, eine vektorwertige Funktion zu definieren, die den obigen Ansprüchen genügt. Die einfachste dieser Möglichkeiten stellt die Definition von $\vec{f}(\vec{x})$ als einen Vektor von $N^{Klassen}-1$ skalaren *Komponentenfunktionen* $f_k(\vec{x})$ mit $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$ dar. Diese Komponentenfunktionen werden unabhängig voneinander auf die Eingangsdaten $\vec{x}$ angewendet und ergeben so die einzelnen Komponenten der Projektion von $\vec{x}$ in

den $(N^{Klasen}-1)$ -dimensionalen Raum. Auf diese Weise sei $\vec{f}(\vec{x})$ also definiert als:

$$\vec{f}(\vec{x})=\begin{pmatrix} f_1(\vec{x}) \\ \vdots \\ f_{N^{Klassen}-1}(\vec{x}) \end{pmatrix}$$

Für den Fall $N^{Klassen}=2$ ergibt sich daraus der einfache skalare Klassifikator $f(\vec{x})=f_1(\vec{x})$. Die obige Definition von $\vec{f}(\vec{x})$ lässt nun jedoch eine weitere Frage offen, nämlich die nach der Definition der einzelnen Komponentenfunktionen. Grundsätzlich kommt als Komponentenfunktion jede beliebige Funktion in Frage, für die $f_k:\mathbb{R}^n\to\mathbb{R}$ gilt. Zusätzlich spräche nichts dagegen, dass die einzelnen Komponentenfunktionen unterschiedlichen Funktionstypen zugehörig sind, sodass es ein breites Spektrum von Funktionen gibt, aus denen $\vec{f}(\vec{x})$ zusammengestellt werden kann. Um das System jedoch nicht unnötig komplex zu machen, soll für diese Arbeit festgelegt werden, dass die Komponentenfunktionen alle dem gleichen Funktionstyp entstammen. Dieser Funktionstyp soll eine quadratische Funktion sein, die folgendermaßen bestimmt ist:

$$f_k(\vec{x})=f_k(\vec{x},\vec{\theta}_k)=\vec{x}^T\,\underline{A}_k\,\vec{x}+\vec{w}_k{}^T\vec{x}+b_k,\quad \vec{\theta}_k=\left(\underline{A}_k,\vec{w}_k,b_k\right)$$

Hierbei enthält der Vektor $\vec{\theta}_k$ die anpassbaren Parameter von $f_k(\vec{x})$. Die Matrix $\underline{A}_k\in\mathbb{R}^{n\times n}$ beschreibt den Einfluss, den die Produkte der Eigenschaften $x_i x_j$ auf den Wert von $f_k(\vec{x})$ und damit auf die k-te Komponente der Klassifikatorfunktion $\vec{f}(\vec{x})$ haben. Der Vektor $\vec{w}_k\in\mathbb{R}^n$ wiederum gewichtet den direkten Einfluss der Komponenten x_i des Eingangsvektors $\vec{x}$ auf $f_k(\vec{x})$. Schließlich erlaubt der Skalar $b_k\in\mathbb{R}$ eine von $\vec{x}$ unabhängige Anpassung des Wertes von $f_k(\vec{x})$. Mit a_{kij} als Komponente von $\underline{A}_k$ (Eintrag in der i-ten Zeile und j-ten Spalte von $\underline{A}_k$) und w_{ki} als i-te Komponente von $\vec{w}_k$ lässt sich $f_k(\vec{x},\vec{\theta}_k)$ ausschreiben als:

$$f_k(\vec{x},\vec{\theta}_k)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{kij}\,x_i\,x_j+\sum_{i=1}^{n}w_{ki}\,x_i+b_k$$

Die Wahl quadratischer Funktionen für die Komponenten der Klassifikatorfunktion $\vec{f}(\vec{x})$ geht einerseits davon aus, dass die Eingangsdaten, insbesondere bei der Sekundärstrukturvorhersage von Proteinen, nur nichtlinear trennbar sind, sodass auch eine entsprechend komplexe Projektion der Eingangsdaten verwendet werden muss (siehe Abschnitt 3.3.2,

XOR-Problem). Die quadratische Funktion stellt die einfachste nichtlineare Funktion dar, sodass ihre Wahl in diesem Fall den am wenigsten komplexen Lösungsansatz darstellt. Zudem ergibt sich ein potentieller Vorteil dieser Wahl für die Sekundärstrukturvorhersage: Die quadratische Funktion verknüpft mittels der Multiplikation $x_i x_j$ verschiedene Positionen eines Sequenzfensters miteinander, was insbesondere im Fall helikaler Sekundärstrukturen von Vorteil sein kann. Bei diesen Sekundärstrukturen existiert ein Zusammenhang zwischen nahe gelegenen Residuen, wie zum Beispiel zwischen dem i-ten und dem $i+3$-ten bei 3_{10}-Helices (Abschnitt 2.3.4.2). Diese Zusammenhänge können potentiell mit einer quadratischen Funktion modelliert werden. Zudem wurden von Francesco Bettella in der Entwicklung von SPARROW, welche parallel zur Entwicklung des hier vorgestellten Ansatzes stattfand, gute Ergebnisse mit quadratischen Funktionen erzielt [1]. Diese wurden in SPARROW verwendet, um die im Mehrklassenproblem eingebetteten Zweiklassenprobleme zu lösen. Der hier vorgestellte Ansatz würde im Fall $N^{Klassen}=2$ ein ähnliches Verfahren zur Lösung von Zweiklassenproblemen liefern: In diesem Fall wird, wie bei SPARROW auch, eine skalare quadratische Funktion zur Lösung des Zweiklassenproblems verwendet.

4.2.4. Optimierung der Parameter

Die Parameter $\vec{\theta}_k$ der einzelnen Komponentenfunktionen $f_k(\vec{x},\vec{\theta}_k)$ können zu einem globalen Parametersatz $\vec{\theta}=(\vec{\theta}_1,\ldots,\vec{\theta}_{N^{Klassen}-1})$ zusammengefasst werden, der alle anpassbaren Parameter der Klassifikatorfunktion $\vec{f}(\vec{x})$ enthält. Auf diese Weise kann die Klassifikatorfunktion auch geschrieben werden als:

$$\vec{f}(\vec{x})=\vec{f}(\vec{x},\vec{\theta})=\begin{pmatrix} f_1(\vec{x},\vec{\theta}_1) \\ \vdots \\ f_{N^{Klassen}-1}(\vec{x},\vec{\theta}_{N^{Klassen}-1}) \end{pmatrix}$$

Nun stellt sich die Frage, wie der Parametersatz $\vec{\theta}$ bestimmt sein muss, damit die Klassifikatorfunktion ihre Aufgabe als Klassifikator möglichst zuverlässig erfüllen kann. Die Antwort auf diese Frage liefert die Lerntheorie, insbesondere die empirische Risikominimierung (Abschnitt 3.1.3) und die Maximum-Likelihood-Methode (Abschnitt 3.2.2). Das präsentierte Lernverfahren verwendet einen begrenzten Lerndatensatz, um die Parameter $\vec{\theta}$ der Klassifikatorfunktion $\vec{f}(\vec{x},\vec{\theta})$ so zu berechnen, dass eine gegebene Fehlerfunktion

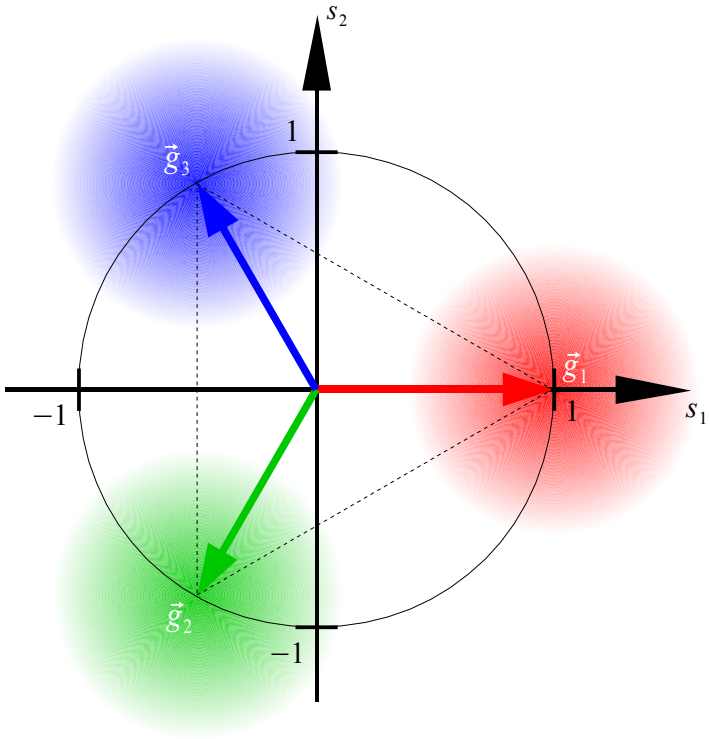


Abbildung 37: *Beispiel einer rauschbehafteten Projektion der Eingangsdaten beim Dreiklassenproblem*

minimiert wird, sodass die Klassifikatorfunktion die vorgegebene Klassifikation des Lerndatensatzes möglichst gut nachbildet. Unter der Annahme einer guten Generalisierungsfähigkeit des Klassifikators (Abschnitt 3.1.4) sollte so auch die Klassifikation beliebiger anderer Daten, die nicht im Lerndatensatz enthalten sind, nachgebildet werden können.

Im optimalen Fall sollte die Klassifikatorfunktion die Lerndaten so transformieren, dass ein Eingangsvektor $\vec{x} \in \mathbb{R}^n$ derart auf $\mathbb{R}^{N^{Klassen}-1}$ projiziert wird, dass seine Projektion mit dem Klassenvektor der Klasse, zu der $\vec{x}$ gehört, identisch ist. Im optimalen Fall soll also $\vec{x} \in \zeta_\alpha \Leftrightarrow \vec{f}(\vec{x}) = \vec{g}_\alpha$ gelten. Dadurch wäre gewährleistet, dass jeder Vektor $\vec{x}$ der Klasse zugewiesen wird, zu der er gehört. In der realen Anwendung kann jedoch nicht von diesem optimalen Fall ausgegangen werden. Vielmehr muss angenommen werden, dass die Klassifikatorfunktion $\vec{f}(\vec{x})$ mit einem Rauschen $\vec{\eta}$ behaftet ist, sodass sich das Verhältnis $\vec{x} \in \zeta_\alpha \Leftrightarrow \vec{f}(\vec{x}) = \vec{g}_\alpha + \vec{\eta}$ ergibt. Damit sind die Projektionen der Vertreter der einzelnen Klassen mit einer unbekannten Streuung um die Klassenvektoren verteilt. Es sei angenommen, dass die entsprechende Verteilung einer Normalverteilung wie in Abbildung 37 dargestellt entspricht. In diesem Fall beschreibt der Klassenvektor den Erwartungswert der

Projektionen aller Eingangsvektoren einer Klasse, die mit einer bestimmten Varianz um den Klassenvektor verteilt sind. Sei nun $D^{lern}=\{(\vec{x}_i,\vec{y}_i)|1\le i\le N\}$ der Lerndatensatz, wobei $\vec{x}_i\in\zeta_\alpha\Leftrightarrow\vec{y}_i=\vec{g}_\alpha$. Auf diese Weise kann gemäß Abschnitt 3.2.2 eine Wahrscheinlichkeit $p(\vec{y}_i|\vec{x}_i,\vec{\theta})$ angegeben werden. Für diese ergibt sich:

$$p(\vec{y}_i|\vec{x}_i,\vec{\theta})=\frac{1}{\sqrt{2\pi|\underline{\underline{\Sigma}}|}}e^{\frac{-(\vec{y}_i-\vec{f}(\vec{x}_i,\vec{\theta}))^T\underline{\underline{\Sigma}}^{-1}(\vec{y}_i-\vec{f}(\vec{x}_i,\vec{\theta}))}{2}}$$

Hierbei ist $\underline{\underline{\Sigma}}\in\mathbb{R}^{(N^{Klassen}-1)\times(N^{Klassen}-1)}$ die Kovarianzmatrix, die die Streuung der Daten um den Mittelwert $\vec{y}_i$ repräsentiert, $\underline{\underline{\Sigma}}^{-1}$ ihre inverse Matrix und $|\underline{\underline{\Sigma}}|$ ihre Determinante. Mit Hilfe dieser Wahrscheinlichkeit kann eine Likelihood-Funktion aufgestellt werden und eine ähnliche Rechnung durchgeführt werden, wie sie Bereits in Abschnitt 3.2.2 gezeigt wurde. Daraus ergibt sich im Fall eines Gaußschen Rauschens der projizierten Daten der mittlere quadratische Fehler als zu minimierende Fehlerfunktion:

$$E(\vec{\theta})=\frac{1}{2\mathrm{N}}\sum_{i=1}^{N}(\vec{y}_i-\vec{f}(\vec{x}_i,\vec{\theta}))^T(\vec{y}_i-\vec{f}(\vec{x}_i,\vec{\theta}))=\frac{1}{2\mathrm{N}}\sum_{i=1}^{N}\sum_{k=1}^{N^{Klassen}-1}(y_{ik}-f_k(\vec{x}_i,\vec{\theta}_k))^2$$

Als notwendige Bedingung für ein Minimum der Fehlerfunktion ergibt sich, dass ihre Ableitung nach den Parametern $\vec{\theta}$ gleich dem Nullvektor sein muss, also $\nabla E(\vec{\theta})=\vec{0}$. Um die Berechnung dieser Ableitung zu vereinfachen soll zunächst die Klassifikatorfunktion umgeschrieben werden. Als *Gewichte* sollen im Folgenden diejenigen Parameter einer Komponente der Klassifikatorfunktion $\vec{f}(\vec{x},\vec{\theta})$ bezeichnet werden, die mit dem Eingangsvektor $\vec{x}\in\mathbb{R}^n$ verknüpft sind. Damit sind also alle Elemente von $\underline{\underline{A}}_k$ und $\vec{w}_k$ einer Komponentenfunktion $f_k(\vec{x},\vec{\theta}_k)$ gemeint. Somit berechnet sich die Anzahl der Gewichte $\widetilde{P}$ aus der Gleichung $\widetilde{P}=n^2+n$ (die Anzahl aller anpassbaren Parameter von $f_k(\vec{x},\vec{\theta}_k)$ ergibt sich aus $P=\widetilde{P}+1$, da in diesem Fall auch die Konstante b_k berücksichtigt wird). Aus den Gewichten von $f_k(\vec{x},\vec{\theta}_k)$ lässt sich nun ein Gewichtsvektor konstruieren. Dieser Gewichtsvektor $\vec{W}_k\in\mathbb{R}^{\widetilde{P}}$ soll zunächst alle Elemente von $\underline{\underline{A}}_k$ enthalten, gefolgt von allen Elementen des Vektors $\vec{w}_k$. Damit ergibt sich $\vec{W}_k$ als:

$$\vec{W}_k=\left(a_{k11},a_{k12},\ldots,a_{k1(N^{Klassen}-1)},a_{k21},\ldots,a_{knn},w_{k1},\ldots,w_{kn}\right)^T$$

Entsprechend kann mit Hilfe des linearen Eingangsvektors $\vec{x}$ ein quadratischer Merkmals-

vektor $\vec{X} \in \mathbb{R}^{\tilde{P}}$ definiert werden, dessen Komponenten X_p diejenigen Größen enthalten, die in $f_k(\vec{x}, \vec{\theta}_k)$ mit dem Gewicht W_{kp} aus dem Gewichtsvektor $\vec{W}_k$ verknüpft sind. Der resultierende Vektor ist dann folgendermaßen definiert:

$$\vec{X} = \left(x_1 x_1, x_1 x_2, \ldots, x_1 x_{N^{Klassen}-1}, x_2 x_1, \ldots, x_n x_n, x_1, \ldots, x_n\right)^T$$

Mit diesen beiden Größen kann nun die Funktion $f_k(\vec{x}, \vec{\theta})$ umgeschrieben werden als:

$$f_k(\vec{x}, \vec{\theta}_k) = \vec{x}^T A_k \vec{x} + \vec{w}_k{}^T \vec{x} + b_k = \vec{W}_k^T \vec{X} + b_k$$

Die Komponentenfunktion $f_k(\vec{x}, \vec{\theta})$ ist damit nicht nur im Parameterraum sondern auch im Merkmalsraum, aus dem die Merkmalsvektoren $\vec{X}$ stammen, eine lineare Funktion. Hierbei weist die Definition des Merkmalsvektors Ähnlichkeit zu der für das XOR-Problem verwendeten Transformation in einen höherdimensionalen Merkmalsraum (Abschnitt 3.3.2) auf.

Mit Hilfe der Darstellung von $f_k(\vec{x}, \vec{\theta}_k)$, die aus der besagten Transformation folgt, kann nun die Fehlerfunktion $E(\vec{\theta})$ auf einfache Weise minimiert werden. Hierzu soll zunächst die Gleichung $\partial E(\theta)/\partial b_k = 0$ für ein beliebiges k gelöst werden. Die Berechnung der partiellen Ableitung nach b_k ergibt:

$$\frac{\partial E(\vec{\theta})}{\partial b_k} = \frac{1}{N} \sum_{i=1}^{N} \left(y_{ik} - f_k(\vec{x}_i, \vec{\theta}_k)\right) \cdot \left(-\frac{\partial f_k(\vec{x}_i, \vec{\theta}_k)}{\partial b_k}\right)$$

Daraus folgt mit $-\partial f_k(\vec{x}_i, \vec{\theta}_k)/\partial b_k = -1$ und $\partial E(\vec{\theta})/\partial b_k = 0$:

$$\frac{1}{N} \sum_{i=1}^{N} \left(f_k(\vec{x}_i, \vec{\theta}_k) - y_{ik}\right) = 0$$

Nach Einsetzen von $f_k(\vec{x}, \vec{\theta}_k) = \vec{W}_k^T \vec{X} + b_k$ ergibt sich daraus:

$$b_k = \frac{1}{N} \sum_{i=1}^{N} \left(y_{ik} - \vec{W}_k^T \vec{X}_i\right)$$

Mit Hilfe dieser Gleichung kann b_k unter der Voraussetzung berechnet werden, dass $\vec{W}_k^T$ bereits bekannt ist. Für die Berechnung einer beliebigen Komponente W_{kp} dieses Gewichtsvektors muss die Gleichung $\partial E(\theta)/\partial W_{kp} = 0$ gelöst werden. Für diese partielle Ableitung ergibt sich:

$$\frac{\partial E(\vec{\theta})}{\partial W_{kp}} = \frac{1}{N}\sum_{i=1}^{N}\left(y_{ik} - f_k(\vec{x}_i, \vec{\theta}_k)\right)\cdot\left(-\frac{\partial f_k(\vec{x}_i, \vec{\theta}_k)}{\partial W_{kp}}\right)$$

Die partielle Ableitung $\partial f_k(\vec{x}_i, \vec{\theta}_k)/\partial W_{kp}$ ergibt sich zu $-\partial f_k(\vec{x}_i, \vec{\theta}_k)/\partial W_{kp} = -X_{ip}$. Mit $\partial E(\vec{\theta})/\partial W_{kp} = 0$ kann die obige Gleichung geschrieben werden als:

$$\frac{1}{N}\sum_{i=1}^{N}\left(f_k(\vec{x}_i, \vec{\theta}_k) - y_{ik}\right)\cdot X_{ip} = 0$$

Nun kann abermals $f_k(\vec{x}, \vec{\theta}_k) = \vec{W}_k^T \vec{X} + b_k$ eingesetzt werden, womit sich nach einigen einfachen Umformungen folgendes ergibt:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\vec{W}_k^T \vec{X}_i - y_{ik}\right)\cdot X_{ip} + \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot b_k = 0$$

Als nächstes kann der vormals berechnete Ausdruck für b_k eingesetzt werden, wobei jedoch der Laufindex i in der Summe für b_k durch j ersetzt werden muss, um Verwechslungen mit den Laufindizes in der obigen Gleichung zu vermeiden. Auf diese Weise ergibt sich:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\vec{W}_k^T \vec{X}_i - y_{ik}\right)\cdot X_{ip} + \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N}\left(y_{jk} - \vec{W}_k^T \vec{X}_j\right)\right) = 0$$

Ausmultiplizieren und anschließende Umformung der obigen Gleichung, sodass alle Terme, die $\vec{W}_k$ enthalten, auf der linken Seite stehen und alle anderen Terme auf der rechten, liefert die Gleichung:

$$\left(\frac{1}{N}\sum_{i=1}^{N}\vec{X}_i X_{ip} - \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N}\vec{X}_j\right)\right)\cdot\vec{W}_k^T = \frac{1}{N}\sum_{i=1}^{N} y_{ik} X_{ip} - \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N} y_{jk}\right)$$

Diese Gleichung hat die Form $\vec{m}_p^T \cdot \vec{W}_k = v_{kp}$, wobei der Vektor $\vec{m}_p$ sich aus

$$\vec{m}_p = \frac{1}{N}\sum_{i=1}^{N}\vec{X}_i X_{ip} - \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N}\vec{X}_j\right) = \langle \vec{X}\, X_p\rangle - \langle X_p\rangle\langle \vec{X}\rangle$$

als Kovarianz der Eingangsdaten $\vec{X}$ und einer Komponente X_p dieser Daten ergibt. Da k nicht in der Gleichung für $\vec{m}_p$ auftaucht, ist $\vec{m}_p$ unabhängig davon, welche der Komponentenfunktionen $f_k(\vec{x}, \vec{\theta}_k)$ betrachtet wird. Tatsächlich ergeben sich für unterschied-

liche $f_k(\vec{x},\vec{\theta}_k)$ für die entsprechenden W_{kp} die gleichen $\vec{m}_p$. Anders sieht dies bei der skalaren Größe v_{kp} auf der rechten Seite der Gleichung aus. Diese ergibt sich als Kovarianz zwischen X_p und y_k aus:

$$v_{kp}=\frac{1}{N}\sum_{i=1}^{N} y_{ik}X_{ip}-\left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N} y_{jk}\right)=\langle y_k X_p\rangle-\langle X_p\rangle\langle y_k\rangle$$

Aus dieser Gleichung geht hervor, dass v_{kp} sowohl davon abhängt, welches $f_k(\vec{x},\vec{\theta}_k)$ als auch welches der Gewichte W_{kp} betrachtet wird, was sich darin äußert, dass sowohl k als auch p in der Gleichung vorkommen.

Die gezeigten Berechnungen können für alle Gewichte W_{kp} einer Komponentenfunktion $f_k(\vec{x},\theta_k)$ durchgeführt werden. Dabei ergibt sich für jedes W_{kp} eine Gleichung der Form $\vec{m}_p^T\cdot\vec{W}_k=v_{kp}$. All diese Gleichungen können zu einem linearen Gleichungssystem zusammengefasst werden, das die Form $\underline{\underline{M}}\cdot\vec{W}_k=\vec{v}_k$ besitzt. Hierbei ist die Matrix $\underline{\underline{M}}\in\mathbb{R}^{\tilde{P}\times\tilde{P}}$ die Koeffizientenmatrix des Gleichungssystems, die alle $\vec{m}_p$ als Zeilenvektoren besitzt. Damit entspricht $\underline{\underline{M}}=\langle\vec{X}\,\vec{X}^T\rangle-\langle\vec{X}\rangle\langle\vec{X}^T\rangle$ der Kovarianz der Eingangsdaten. Der Vektor $\vec{v}_k\in\mathbb{R}^{\tilde{P}}$, der die rechten Seiten des Gleichungssystems zusammenfasst, enthält die einzelnen v_{kp} als Komponenten und entspricht damit einer Zeile $\vec{v}_k=\langle y_k\vec{X}\rangle-\langle y_k\rangle\langle\vec{X}\rangle$ der Kovarianzmatrix $\underline{\underline{V}}\in\mathbb{R}^{\left(N^{Klassen}-1\right)\times\tilde{P}}$ der Eingangsdaten mit den Ausgangsdaten, die durch $\underline{\underline{V}}=\langle\vec{y}\,\vec{X}^T\rangle-\langle\vec{y}\rangle\langle\vec{X}^T\rangle$ bestimmt ist.

Der Lernvorgang, also die Berechnung der optimalen Parameter $\hat{\vec{\theta}}$, die den mittleren quadratischen Fehler minimieren, besteht somit in der Lösung einer Reihe linearer Gleichungssysteme, die sich in ihren Koeffizientenmatrizen gleichen. Dabei liefert jedes Gleichungssystem die optimalen Parameter $\hat{\vec{\theta}}_k$ der k-ten Komponentenfunktion $f_k(\vec{x},\vec{\theta}_k)$ der Klassifikatorfunktion $\vec{f}(\vec{x},\vec{\theta})$. Dies stellt einen Kontrast zu den meisten gängigen Lernverfahren dar, die iterativ vorgehen um das Minimum einer Fehlerfunktion zu bestimmen. An dieser Stelle sei der Back-Propagation-Algorithmus (Abschnitt 3.5.4.3) als Beispiel genannt. Der vorgestellte vektorwertige Klassifikator hingegen lernt, indem seine Parameter direkt aus dem Lerndatensatz berechnet werden. Insofern besitzt er eher Gemeinsamkeiten mit der linearen Diskriminanzanalyse (Abschnitt 3.5.2) oder dem Bayes-Klassi-

fikator (Abschnitt 3.5.3). Auch diese beiden Verfahren berechnen die Parameter des jeweiligen Klassifikators, wie es auch beim vektorwertigen Klassifikator der Fall ist, aus Statistiken, die aus dem Lerndatensatz gewonnen werden. Dabei ist die Ähnlichkeit zu Fishers linearer Diskriminanzanalyse größer. Tatsächlich ergibt der vektorwertige Klassifikator für $N^{Klassen}=2$ und $f(\vec{x},\vec{\theta})=\vec{W}_k^T\vec{X}+b$ einen skalaren Klassifikator der Fishers Ansatz $\vec{\lambda}\vec{x}$ (siehe Abschnitt 3.5.2) sehr ähnlich ist, sich jedoch in der Anzahl, Bedeutung und Optimierung der Parameter unterscheidet.

4.2.4.1. Regularisierung der Fehlerfunktion

Wie bei anderen Lernverfahren auch, ist es bei der Optimierung der Parameter des vektorwertigen Klassifikators, wie sie in Abschnitt 4.2.4 beschrieben wurde, möglich, dass Überanpassung nach Abschnitt 3.1.4 auftritt wenn der verwendete Lerndatensatz zu klein ist. Um die Auswirkungen dieses Effekts einzudämmen, ist es analog zu anderen Lernverfahren möglich, eine *Regularisierung* der Fehlerfunktion vorzunehmen. Hierfür wird eine erweiterte Fehlerfunktion verwendet, die sich aus der folgenden Gleichung ergibt:

$$\tilde{E}(\vec{\theta})=E(\vec{\theta})+\sum_{k=1}^{N^{Klassen}-1}\left(\frac{\lambda_{quad}}{2}\sum_{i=1}^{\tilde{P}-n}W_{ki}^2+\frac{\lambda_{lin}}{2}\sum_{i=\tilde{P}-n+1}^{\tilde{P}}W_{ki}^2\right)$$

Diese Erweiterung basiert auf einem Regularisierungsprinzip, das auch von neuronalen Netzen bekannt ist und als *Weight Decay* bezeichnet wird [45]. Dabei wird die Summe der Quadrate der Gewichte mit einem Faktor versehen und auf die Fehlerfunktion addiert, um die Gewichte so möglichst klein zu halten und auf diese Weise Überanpassung vorzubeugen, die sich meist durch große Zahlenwerte der Gewichte äußert. Die Faktoren $\lambda_{quad},\lambda_{quad}\in\mathbb{R}$ sind frei wählbare Parameter des Lernverfahrens. λ_{quad} bestimmt den Einfluss der Regularisierung auf die Gewichte der quadratischen Terme, also auf die Berechnung der Elemente der Matrizen $\underline{\underline{A}}_k$, während λ_{lin} den Einfluss auf die linearen Gewichte $\vec{w}_k$ der einzelnen Komponentenfunktionen $f_k(\vec{x},\vec{\theta}_k)$ des vektorwertigen Klassifikators regelt. Die Ableitung von $\tilde{E}(\vec{\theta})$ nach einem Gewicht W_{kp} liefert:

$$\frac{\partial\tilde{E}(\vec{\theta})}{\partial W_{kp}}=\frac{\partial E(\vec{\theta})}{\partial W_{kp}}+\lambda_p W_{kp}$$

Dabei beschreibt λ_p den zu W_{kp} gehörigen Faktor, also λ_{quad}, wenn W_{kp} eines der

quadratischen Gewichte ist, und λ_{lin} im anderen Fall. Mit $\partial \tilde{E}(\vec{\theta})/\partial W_{kp}=0$ und $\partial E(\vec{\theta})/\partial W_{kp}$ erhält man analog zu Abschnitt 4.2.4 für jedes W_{kp} eine lineare Gleichung:

$$\vec{m}_p^T \cdot \vec{W}_k + \lambda_p W_{kp} = v_{kp}$$

Das Ausschreiben des Skalarproduktes auf der linken Seite in Summenform ergibt:

$$\sum_{i=1}^{\tilde{P}} m_{pi} \cdot W_{ki} + \lambda_p W_{kp} = \sum_{i=1, i\neq p}^{\tilde{P}} m_{pi} \cdot W_{ki} + (m_{pp} + \lambda_p) \cdot W_{kp}$$

Daraus ergibt sich, dass die Regularisierung zu einer Addition von λ_p zum Diagonalelement m_{pp} der Koeffizientenmatrix führt. Die so entstandenen linearen Gleichungssysteme für die einzelnen $\vec{W}_k$ haben damit die Form:

$$(\underline{\underline{M}} + \underline{\underline{\Lambda}}) \cdot \vec{W}_k = \vec{v}_k$$

Dabei ist die Matrix $\underline{\underline{\Lambda}} \in \mathbb{R}^{\tilde{P}\times\tilde{P}}$ eine Matrix, die in der Diagonalen die einzelnen λ_p und sonst Nullen enthält.

4.2.5. Theoretische Zusammenfassung des Lernvorgangs

Mit Hilfe der Betrachtungen aus Abschnitt 4.2.4 kann nun eine grobe Übersicht des Lernvorgangs für den vektorwertigen Klassifikator aufgestellt werden. Diese fasst nur die für den Lernvorgang benötigten theoretischen Schritte zusammen, ohne auf die praktische Implementierung des Lernverfahrens einzugehen. Dabei wird von der bisher verwendeten Darstellung der Komponentenfunktionen $f_k(\vec{x}, \vec{\theta}_k) = \vec{W}_k^T \vec{X} + b_k$ als Skalarprodukt eines Gewichtsvektors $\vec{W}_k$ mit einem Merkmalsvektor $\vec{X}$ ausgegangen. Die beschriebenen Schritte des Lernverfahrens gehen weiterhin davon aus, dass die Eingangsdaten $\vec{x}_i$ des Lerndatensatzes bereits in ihre Darstellung als Merkmalsvektoren $\vec{X}_i$ überführt wurden. Daraus ergeben sich die folgenden Schritte für den Lernvorgang:

1. Berechnung der Koeffizientenmatrix $\underline{\underline{M}}$ der linearen Gleichungssysteme als Kovarianz der Eingangsdaten des Lerndatensatzes: $\underline{\underline{M}} = \langle \vec{X}\,\vec{X}^T \rangle - \langle \vec{X} \rangle \langle \vec{X}^T \rangle$
2. Berechnung der linken Seiten $\vec{v}_k$ der linearen Gleichungssysteme als Spaltenvektoren der Kovarianzmatrix $\underline{\underline{V}}$ von Ausgangs- und Eingangsdaten des Lerndatensates: $\underline{\underline{V}} = \langle \vec{y}\,\vec{X}^T \rangle - \langle \vec{y} \rangle \langle \vec{X}^T \rangle$

3. Berechnung von $\vec{W}_k$ und b_k für alle $1 \le k \le N^{Klassen} - 1$:

   3.1. Berechnen von $\vec{W}_k$ als Lösung des linearen Gleichungssystems $(\underline{\underline{M}} + \underline{\underline{\Delta}}) \cdot \vec{W}_k = \vec{v}_k$

   3.2. Berechnen von b_k aus $b_k = \frac{1}{N} \sum_{i=1}^{N} \left(y_{ik} - \vec{W}_k^T \vec{X}_i \right)$

Somit sind die Parameter $\hat{\vec{\theta}}$ des vektorwertigen Klassifikators $\vec{f}(\vec{x}, \vec{\theta})$ bestimmt, die den mittleren quadratischen Fehler über dem Lerndatensatz minimieren. Weiterhin ist damit auch der theoretische Rahmen des vektorwertigen Klassifikators und des dazugehörigen Lernverfahrens gesetzt. Mit Hilfe dieses theoretischen Rahmens kann im nächsten Abschnitt die Realisierung eines Programms zur Sekundärstrukturvorhersage von Proteinen beschrieben werden.

4.3. Realisierung des Vorhersageprogramms

Mit dem im Abschnitt 4.2 vorgestellten vektorwertigen Klassifikator als Basis kann nun das eigentliche Programm zur Vorhersage der Sekundärstruktur von Proteinen realisiert werden. Wie bereits im Abschnitt 4.1.1 beschrieben, besitzen viele Programme, die für diesen Zweck entwickelt wurden, einen mehrstufigen Aufbau. Es ist einleuchtend, dass ein solcher Aufbau von Vorteil ist, denn jede Stufe eines solchen Programms stellt eine Form der Nachverarbeitung der Vorhersage der vorhergehenden Stufe dar und sollte diese prinzipiell innerhalb eines gewissen Rahmens verbessern. Die Berücksichtigung der Vorhersagen benachbarter Residuen in einer zweiten Stufe, sollte beispielsweise eine Glättung der Vorhersage der ersten Stufe ermöglichen, also das Entfernen vereinzelter Residuen, die einer Klasse zugeordnet sind und sich innerhalb eines größeren Abschnitts der Sequenz befinden, der einer anderen Klasse zugeordnet ist. Ausgehend von der Annahme, dass ein großer, einheitlich klassifizierter Abschnitt der Sequenz eher richtig klassifiziert sein sollte, können Ausreißer in solchen Sequenzabschnitten potentiell falsch klassifiziert sein. Daher kann ihre Entfernung ein Vorteil sein, der sich in der Korrektheit der erzeugten Sekundärstrukturvorhersage niederschlägt. Aus diesem Grund besteht das in dieser Arbeit präsentierte Vorhersageprogramm auch aus mehreren Stufen, die in Abbildung 38 dargestellt sind.

Das Programm besitzt, in Anlehnung an SPARROW, drei Stufen. Die ersten beiden Stufen

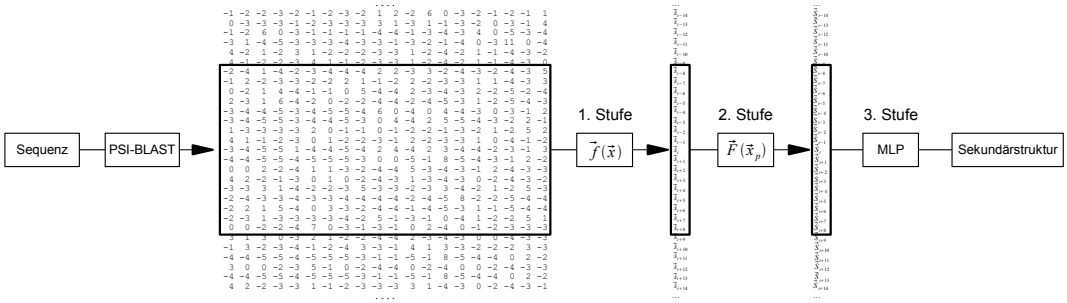


Abbildung 38: *Flussdiagramm der Informationsverarbeitung in *SPARROW*

sind mit Hilfe vektorwertiger Klassifikatoren umgesetzt, während die letzte Stufe aus einem MLP (Multilagenperzeptron, Abschnitt 3.5.4.2) besteht. Wie Abbildung 38 entnommen werden kann, erzeugt jede Stufe für jedes Residuum i eines Proteins eine Ausgabe. Dabei werden die Ausgaben einer höheren Stufe erzeugt, indem die Ausgaben der vorhergehenden Stufe für Residuen innerhalb eines symmetrischen Fensters um jedes Residuum i zu einem Eingangsvektor zusammengefasst und für die Vorhersage verwendet werden. Jede der Stufen des Programms kann prinzipiell unterschiedlich viele Klassen erkennen, im Fall der ersten Stufe $N_1^{Klassen}$ Klassen, der zweiten Stufe $N_2^{Klassen}$ Klassen und der letzten Stufe $N_3^{Klassen}$ Klassen. Dabei soll lediglich die Einschränkung gemacht werden, dass $N_1^{Klassen} \geq N_2^{Klassen} \geq N_3^{Klassen}$, sodass die Klassen, die eine Stufe unterscheidet, entweder der Klasseneinteilung der vorhergehenden Stufe entsprechen oder sich aus Vereinigungen von Klassen ergeben, auf denen die vorhergehende Stufe operiert. Grund hierfür ist, dass es kein Problem darstellt mehrere Klassen zu vereinigen, jedoch kann eine einmal vorgenommene Vereinigung nicht ohne weiteres wieder in die ursprünglichen Klassen zerlegt werden. Nach diesem Prinzip ist es also möglich, zum Beispiel zunächst in der ersten Stufe alle acht DSSP-Klassen zu berücksichtigen, um dann in der dritten Stufe schließlich eine Vorhersage mittels der lockeren Klasseneinteilung zu machen.

Aufgrund der Verwandtschaft des vorgestellten Vorhersageprogramms zu SPARROW (der vektorwertige Klassifikator ergibt für das Zweiklassenproblem im Prinzip den gleichen Ansatz, wie er in SPARROW zur Lösung von Zweiklassenproblemen verwendet wird) und der engen Zusammenarbeit mit Francesco Bettella in der Entwicklung beider Ansätze, wurde das in dieser Arbeit entwickelte Programm *SPARROW genannt. Der Stern entstammt hierbei der C-/C++-Syntax und wird für Zeiger, also mehrdimensionale Datengebilde, Felder oder Vektoren verwendet. Der Name SPARROW, den beide Programme ge-

meinsam haben, stellt eine Abkürzung dar und steht für ***S**econdary structure **P**redicting **ARR**ays of **O**ptimized **W**eights*, also in etwa „optimierte Gewichtsvektoren zur Sekundärstrukturvorhersage“. Auch *SPARROW basiert auf optimierten Gewichtsvektoren, jedoch in diesem Fall auf einem Feld von Gewichtsvektoren, was die Benennung des Programms weiter rechtfertigt. Zur weiteren Erläuterung des Aufbaus von *SPARROW soll in den folgenden Abschnitten auf die Umsetzung der einzelnen Stufe eingegangen werden.

4.3.1. Umsetzung der ersten Stufe

Die erste Stufe *SPARROWs stellt eine Sequenz-Struktur-Korrelation dar, die mittels des vektorwertigen Klassifikators $\vec{f}:\mathbb{R}^{20\cdot r_1}\rightarrow\mathbb{R}^{N_1^{Klassen}-1}$ (siehe Abschnitt 4.2) realisiert ist. Dabei ist $r_1=2n_1+1$ die Größe des symmetrisch um das jeweils vorhergesagte Residuum angeordneten Sequenzfensters und n_1 die Anzahl der Nachbarresiduen in beide Richtungen entlang der Sequenz, die in diesem Fenster enthalten sind. Die Parameter $\vec{\theta}^{(f)}=(\vec{\theta}_1^{(f)},\ldots,\vec{\theta}_{N_1^{Klassen}-1}^{(f)})^T$ der Klassifikatorfunktion $\vec{f}(\vec{x},\vec{\theta}^{(f)})$, mit $\vec{\theta}_k^{(f)}=(\underline{\underline{A}}_k^{(f)},\vec{w}_k^{(f)},b_k^{(f)})$ als Parameter der k-ten Komponentenfunktion $f_k(\vec{x},\vec{\theta}_k^{(f)})$, werden unter Verwendung des in Abschnitt 4.2.5 beschriebenen Lernverfahrens für den vektorwertigen Klassifikator und eines geeigneten Lerndatensatzes berechnet. Die Anzahl der anpassbaren Parameter, die dabei für jede Komponentenfunktion $f_k(\vec{x},\vec{\theta}_k^{(f)})$ in der ersten Stufe *SPARROWs berechnet werden müssen, ergibt sich aus:

$$P^{(f_k)}=\left(20\cdot r_1\right)^2+20\cdot r_1+1$$

Aufgrund der Definition der Komponentenfunktionen des vektorwertigen Klassifikators (siehe Abschnitt 4.2.3) folgt, dass die Matrizen $\underline{\underline{A}}_k^{(f)}$ symmetrisch sein müssen. Dadurch reduziert sich die Anzahl der unabhängigen, anpassbaren Parameter $P_u^{(f_k)}$ von $f_k(\vec{x},\vec{\theta}_k^{(f)})$ gegenüber $P^{(f_k)}$ zu:

$$P_u^{(f_k)}=\frac{\left(20\cdot r_1\right)\cdot\left(20\cdot r_1+1\right)}{2}+20\cdot r_1+1$$

Bei der Vorhersage berechnet die erste Stufe für jedes Residuum i, dass durch einen Profilvektor $\vec{x}_i\in\mathbb{R}^{20\cdot r_1}$ (siehe Abschnitt 4.1.3.1) repräsentiert wird, eine Ausgabe $\vec{s}_i$ mit $\vec{s}_i=\vec{f}(\vec{x}_i,\vec{\theta}^{(f)})$. Diese Ausgaben werden von der zweiten Stufe, die im nächsten Abschnitt

beschrieben wird, weiterverwendet.

4.3.2. Umsetzung der zweiten Stufe

Während es die Aufgabe der ersten Stufe ist, eine Korrelation zwischen Sequenz und Struktur herzustellen, soll die zweite Stufe ausgehend von der Vorhersage der ersten Stufe eine Struktur-Struktur-Korrelation herstellen. Hierzu wird in der zweiten Stufe ein vektorwertiger Klassifikator verwendet, der sich analog zur ersten Stufe als vektorwertige Funktion $\vec{F}:\mathbb{R}^{(N_1^{Klassen}-1)\cdot r_2}\rightarrow\mathbb{R}^{N_2^{Klassen}-1}$ ergibt. Dabei ist $r_2=2\cdot n_2+1$ die Größe des symmetrisch um die vorherzusagenden Residuen angeordneten Sequenzfensters, die für die Vorhersage in die Klassifikatorfunktion der zweiten Stufe eingehen. Ein solches Fenster wird für ein Residuum i durch den Vektor $\vec{x}_i^{(s)}\in\mathbb{R}^{(N_1^{Klassen}-1)\cdot r_2}$ repräsentiert, der sich aus den Ausgaben der ersten Stufe ergibt:

$$\vec{x}_i^{(s)}=\left(\vec{s}_{i-n_2},\ldots,\vec{s}_i,\ldots,\vec{s}_{i+n_2}\right)^T$$

Aus diesem Vektor berechnet die zweite Stufe eine Ausgabe $\vec{S}_i$ als $\vec{S}_i=\vec{F}(\vec{x}_i^{(s)},\vec{\theta}^{(F)})$. Die Berechnung der Parameter $\vec{\theta}_k^{(F)}=(\vec{\theta}_1^{(F)},\ldots,\vec{\theta}_{N_2^{Klassen}-1}^{(F)})$, mit $\vec{\theta}_k^{(F)}=(\underline{A}_k^{(F)},\vec{w}_k^{(F)},b_k^{(F)})$ als Parameter der Komponentenfunktionen $F_k(\vec{x}^{(s)},\vec{\theta}_k^{(F)})$ von $\vec{F}(\vec{x}^{(s)},\vec{\theta}^{(F)})$, wird analog zur ersten Stufe mittels des in Abschnitt 4.2.5 beschriebenen Lernverfahrens durchgeführt. Der Unterschied liegt bei der zweiten Stufe lediglich in der Verwendung der Vektoren $\vec{x}^{(s)}$ als Eingangsdaten. Abgesehen von diesem Unterschied verlaufen das Aufstellen der Statistiken und das Lösen der linearen Gleichungssysteme nach dem gleichen Prinzip, wie im Fall der ersten Stufe. Die Anzahl der anpaßbaren Parameter einer Komponentenfunktion $F_k(\vec{x}^{(s)},\vec{\theta}_k^{(F)})$ ergibt sich in diesem Fall aus:

$$P^{(F_k)}=\left(\left(N_1^{Klassen}-1\right)\cdot r_2\right)^2+\left(N_1^{Klassen}-1\right)\cdot r_2+1$$

Wie auch bei der ersten Stufe, ergibt sich daraus aufgrund der Symmetrie der Matrizen $\underline{A}_k^{(F)}$ die Anzahl der unabhängigen anpassbaren Parameter von $F_k(\vec{x}^{(s)},\vec{\theta}_k^{(F)})$ aus:

$$P_u^{(F_k)}=\frac{\left(N_1^{Klassen}-1\right)\cdot r_2\cdot\left(\left(N_1^{Klassen}-1\right)\cdot r_2+1\right)}{2}+\left(N_1^{Klassen}-1\right)\cdot r_2+1$$

4.3.3. Umsetzung der dritten Stufe

Eine weitere Struktur-Struktur-Korrelation stellt die dritte Stufe dar. Diese ist durch ein MLP, wie es in Abschnitt 3.5.4.2 beschrieben ist, realisiert. Durch die Aktivierungsfunktion ihrer Neuronen und den mehrschichtigen Aufbau besitzen MLPs einen höheren Grad an Nichtlinearität als die vektorwertigen Klassifikatoren. Aus diesem Grund kann ein MLP als dritte Stufe unter Umständen Korrelationen innerhalb der Daten aufdecken, die den vorhergehenden Stufen entgangen sind. Zudem bietet ein MLP die Möglichkeit einer intelligenten Entscheidungsregel, die einem Residuum anhand der Vorhersagen der zweiten Stufe die wahrscheinlichste Sekundärstruktur zuweist. Da im Allgemeinen nicht davon ausgegangen werden kann, dass eine vektorwertige Funktion Eingangsdaten genau auf die jeweiligen Klassenvektoren projiziert, kann eine gewissen Verteilung oder Verschiebung von Vertretern einer Klasse um den entsprechenden Klassenvektor angenommen werden (Abschnitt 4.2.4). Da das MLP für eine Sequenz die Vorhersageergebnisse für alle Klassen berücksichtigen und gegeneinander abwägen kann, könnten Fehlklassifikationen, die aufgrund eines solchen Rauschens um die Klassenvektoren entstehen können, vom MLP korrigiert werden. Die Entscheidungsregel für vektorwertige Klassifikatoren, wie sie in Abschnitt 4.2.1 vorgestellt wurde, kann dies nicht gewährleisten.

Das MLP in der dritten Stufe von *SPARROW besteht aus jeweils einer Ein- und Ausgabeschicht, die durch eine einzelne versteckte Schicht miteinander verbunden sind. Die Größe der einzelnen Schichten ist anpassbar, ergibt sich jedoch teilweise aus dem jeweiligen Problem. Die Anzahl der Eingabeknoten beispielsweise wird einerseits durch $N_2^{Klassen}$, andererseits aber auch durch die Größe des symmetrischen Fensters benachbarter Residuen um das Residuum, für welches das MLP eine Vorhersage machen soll, vorgegeben. Mit $r_3 = 2 \cdot n_3 + 1$ als Größe dieses Fensters und n_3 als Anzahl der Nachbarn, die beiderseits des zentralen Residuums berücksichtigt werden, ergibt sich die Anzahl der Eingabeknoten als $I_{MLP} = (N_2^{Klassen} - 1) \cdot r_3 + 2$, was auch der Größe der Eingangsvektoren des MLPs entspricht. Ein solcher Vektoren ergibt sich für ein Residuum i aus den Ausgaben der zweiten Stufe zu:

$$\vec{x}_i^{(S)} = \left(\vec{S}_{i-n_3}, \ldots, \vec{S}_i, \ldots, \vec{S}_{i+n_3}, T_i^{(C)}, T_i^{(N)} \right)^T$$

Die zwei zusätzlichen Werte $T_i^{(C)}$ beziehungsweise $T_i^{(N)}$ geben an, ob das Residuum i,

nahe am C- beziehungsweise N-Terminus ist. Ist dies nicht der Fall, so sind die Werte der jeweiligen Knoten auf -1 gesetzt (entspricht der Inaktivität des Knotens). Andernfalls erhält der betreffende Knoten den Wert 1 um zu signalisieren, dass der jeweilige Terminus in der Nähe liegt. Dieses Schema entspricht der Vorgehensweise, die bei PSIPRED für terminale Residuen verwendet wird [30]. Die Ausgabeschicht des MLP ist durch $N_3^{Klassen}$ bestimmt, da die Anzahl der Ausgabeknoten in einem MLP der Anzahl der Klassen entsprechen muss, die dieses MLP zu unterscheiden hat. Damit ist die Größe der Ausgabeschicht durch $O_{MLP}=N_3^{Klassen}$ bestimmt. Die versteckte Schicht ist die einzige Schicht, deren Größe H_{MLP} vollkommen frei dimensioniert werden kann. Somit ist die Anzahl der Parameter des MLP

$$P_{MLP}=(I_{MLP}+1)\cdot H_{MLP}+(H_{MLP}+1)\cdot O_{MLP}=((N_2^{Klassen}-1)\cdot r_3+3)\cdot H_{MLP}+(H_{MLP}+1)\cdot N_3^{Klassen}$$

Hierbei ist die Addition einer 1 zu I_{MLP} und H_{MLP} notwendig, um die Schwellenwerte der einzelnen Neurone in die Berechnung einfließen zu lassen.

Der Lernvorgang des MLP wird mittels des Back-Propagation-Algorithmus (Abschnitt 3.5.4.3) durchgeführt. Um das Konvergenzverhalten des Algorithmus zu verbessern, wurde der hyperbolische Tangens mit den optimalen Literaturwerten [45] für die Aktivierungsfunktion der Neurone im MLP verwendet. Diese ergibt sich somit als:

$$\varphi(u)=1{,}7159\cdot\tanh\left(\frac{2}{3}\cdot u\right)$$

Die Sollwerte der einzelnen Ausgabeknoten werden auf 1 bei Aktivität des entsprechenden Neurons (wenn das jeweilige Datenelement derjenigen Klasse zugehörig ist, die von dem Neuron repräsentiert wird) und -1 bei dessen Inaktivität (wenn das Datenelement nicht zur Klasse gehört, die das Neuron repräsentiert) gesetzt. Wie in Abschnitt 3.5.4.3 beschrieben, werden die Parameter des MLP mit Zufallszahlen zwischen -10^{-6} und 10^{-6} initialisiert und die Daten des Lerndatensatzes innerhalb einer Iteration des Lernalgorithmus in zufälliger Reihenfolge für das MLP präsentiert. Als Abbruchkriterium wurde eine feste Anzahl von Iterationen gewählt, da die Datensätze, die sich aus Strukturdaten bekannter Proteine ergeben, groß genug sind, sodass keine Überanpassung des MLP auftreten sollte. In solchen Fällen wird davon ausgegangen, dass das Ergebnis eines Lernvorgangs umso besser wird, je mehr Iterationen das Verfahren durchläuft. Die Klassifikation eines Residuums

im Protein wird von der dritten Stufe anhand der Regel der maximalen Konfidenz vorgenommen, indem das Residuum der Klasse zugewiesen wird, die durch den Ausgabeknoten mit der höchsten Ausgabe repräsentiert wird.

4.3.3.1. Einführung von Wahrscheinlichkeitstermen für die dritte Stufe

Die Ausgaben, die von der zweiten Stufe von *SPARROW erzeugt werden und der dritten Stufe als Eingabe dienen, sind schwerer zu interpretieren als beispielsweise Wahrscheinlichkeiten für die einzelnen Klassen. Die Vektoren $\vec{S}_i$ stellen eine räumliche Verteilung der einzelnen Klassen dar, die zunächst dechiffriert werden muss um verstanden zu werden. Zudem kann es sein, dass die Klassen so im Raum verteilt sind, dass sie ohne Verwendung von hochgradig nichtlinearen Klassifikationsgrenzen nur suboptimal unterschieden werden können. Bei Klassenwahrscheinlichkeiten hingegen ist lediglich das Verhältnis der einzelnen Größen zueinander ausschlaggebend und zumeist geht die korrekte Klassifikation mit der maximalen Klassenwahrscheinlichkeit einher. Damit sollte ein solcher Zusammenhang einfacher festzustellen sein, als der Zusammenhang zwischen einer räumlichen Verteilung und der Klassifikation.

An und für sich sollte es für ein MLP kein Problem sein, eine räumliche Verteilung von Klassen zu separieren, da MLPs prinzipiell zur Lösung solcher Probleme konzipiert wurden. Es besteht jedoch die Möglichkeit, dass das Verhalten des MLPs durch eine geeignete Anpassung seiner Eingangsdaten verbessert werden könnte. Wenn diese Daten so verteilt sind, dass die Nichtlinearität des MLPs nicht ausreicht um eine optimale Klassifikation zu erlauben, so könnte es nach Abschnitt 3.3.2 von Vorteil sein, die Dimension der Eingangsdaten zu erhöhen. Hierbei sollten die zusätzlichen Dimensionen in nichtlinearer Weise von den ursprünglichen Dimensionen abhängig sein. Aus diesem Grund soll in diesem Abschnitt eine Funktion beschrieben werden, mit der es möglich ist in nichtlinearer Weise Klassenwahrscheinlichkeiten aus den Vektoren $\vec{S}_i$ zu berechnen, die einerseits dazu verwendet werden sollen die Dimension der Eingangsdaten auf die beschriebene Weise zu erhöhen und andererseits dem MLP auch leichter interpretierbare Größen bieten sollen, als es die Vektoren $\vec{S}_i$ sind.

Zur Berechnung der Wahrscheinlichkeit, dass ein von einem Vektor $\vec{S} \in \mathbb{R}^{N_2^{Klassen}-1}$ repräsentiertes Residuum einer Klasse ζ_α mit $1 \le \alpha \le N_2^{Klassen}$ angehört, soll $\vec{S}$ zunächst nor-

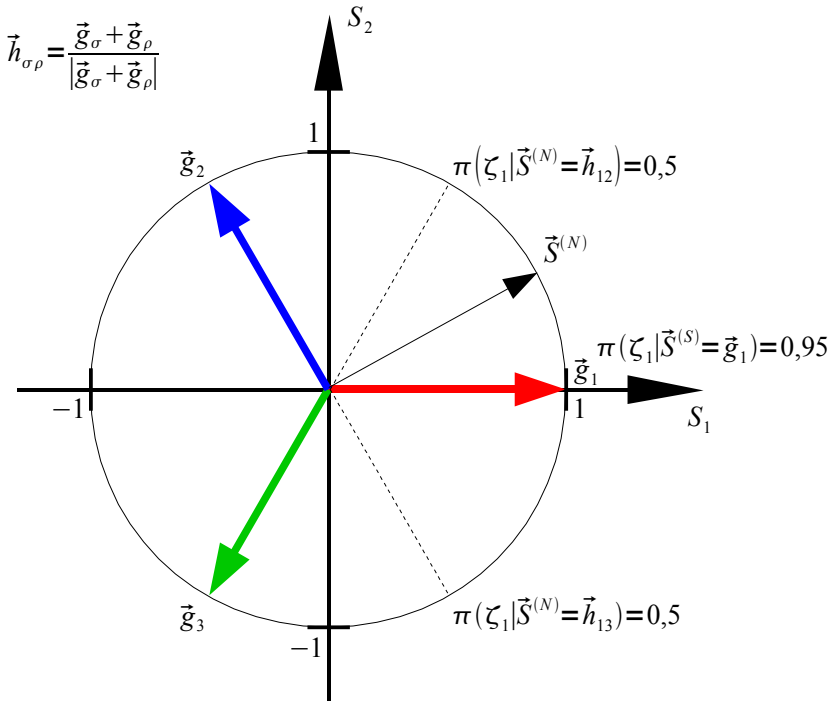


Abbildung 39: *Visualisierung der Funktionsweise der Wahrscheinlichkeitsfunktion*

miert werden. Dadurch ergibt sich der normierte Projektionsvektor $\vec{S}^{(N)}$ aus:

$$\vec{S}^{(N)}=\frac{\vec{S}}{|\vec{S}|}$$

Der Vektor $\vec{S}^{(N)}$ zeigt damit auf einen Punkt auf der Oberfläche der Einheitshyperkugel, der das $(N_2^{Klassen}-1)$-Simplex mit den Klassenvektoren $\vec{g}_\sigma$ als Eckpunkten einbeschrieben ist (siehe Abschnitt 4.2.2). Mit Hilfe von $\vec{S}^{(N)}$ soll die nichtnormierte Wahrscheinlichkeit $\pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)$ unter Verwendung einer Funktion berechnet werden, die der Fermi-Verteilung ähnelt. Diese Funktion sei folgendermaßen definiert:

$$\pi\left(\zeta_\sigma|\vec{S}\right)=\frac{1}{1+e^{-a(\vec{S}^{(N)}\cdot\vec{g}_\sigma-b)}}$$

Die obige Wahrscheinlichkeitsfunktion verwendet, wie auch die Entscheidungsregel für den vektorwertigen Klassifikator (siehe Abschnitt 4.2.1), das Skalarprodukt als Ähnlichkeitsmaß zwischen $\vec{S}^{(N)}$ und $\vec{g}_\sigma$. Da beide Vektoren in diesem Fall Einheitsvektoren sind, ergibt das Skalarprodukt den Kosinus des Winkels den diese Vektoren einschließen. Abbildung 39 zeigt das Beispiel eines normierten Projektionsvektors $\vec{S}^{(N)}$ im Fall eines Dreiklassenproblems, gekennzeichnet durch einen schwarzen Pfeil. Aus der Abbildung ist

ersichtlich, dass die Skalarprodukte mit den Klassenvektoren $\vec{g}_\sigma$ (farbige Pfeile), welche per Definitionem den Projektionen von $\vec{S}^{(N)}$ auf die einzelnen $\vec{g}_\sigma$ entsprechen, lediglich vom Winkel zwischen $\vec{S}^{(N)}$ und den $\vec{g}_\sigma$ abhängen.

Die Parameter a und b der Wahrscheinlichkeitsfunktion sollen diese derart an das jeweilige Klassifikationsproblem anpassen, dass die folgenden, bereits in Abbildung 39 angedeuteten, Bedingungen erfüllt werden:

1. Für die Projektion auf die Winkelhalbierende $\vec{h}_{\sigma\rho}$ zwischen zwei Klassenvektoren $\vec{g}_\sigma$ und $\vec{g}_\rho$ soll gelten: $\pi\left(\zeta_\sigma|\vec{S}^{(N)}=\vec{h}_{\sigma\rho}\right)=0{,}5$
2. Für die Projektion auf einen Klassenvektor soll gelten: $\pi\left(\zeta_\sigma|\vec{S}^{(N)}=\vec{g}_\sigma\right)=0{,}95$

Die erste Bedingung ergibt sich aus der Tatsache, dass die Winkelhalbierenden zwischen zwei Klassenvektoren (in Abbildung 39 als gestrichelte Linien dargestellt) in diesem Fall die Klassifikationsgrenzen für die betreffenden Klassen bilden. Damit bedeutet eine Projektion auf die Winkelhalbierende $\vec{h}_{12}$ beispielsweise, dass das betreffende Residuum gleichermaßen der Klasse ζ_1 wie der Klasse ζ_2 zugewiesen werden könnte. Da beide Klassen damit gleich Wahrscheinlich sind, soll die Wahrscheinlichkeitsfunktion in diesem Fall den Wert 0,5 besitzen. Dieser Wert wird von $\pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)$ genau dann angenommen, wenn das Argument der Exponentialfunktion im Nenner verschwindet. Daraus ergibt sich für b sofort der Ausdruck $b=\vec{g}_\sigma\cdot\vec{h}_{\sigma\rho}$, der dem Kosinus der Hälfte des Winkels zwischen $\vec{g}_\sigma$ und $\vec{g}_\rho$ entspricht.

Die zweite Bedingung folgt aus der Überlegung, dass beispielsweise bei der Betrachtung der Klasse ζ_1, der Wert von $\pi\left(\zeta_1|\vec{S}^{(N)}\right)$ umso näher am Wert 1 liegen sollte, je kleiner der Winkel zwischen $\vec{S}^{(N)}$ und $\vec{g}_1$ ist. Da die Wahrscheinlichkeitsfunktion selbst jedoch diesen Wert aufgrund ihrer Definition nicht erreichen kann, muss für den Fall $\vec{S}^{(N)}=\vec{g}_1$ ein Wert gewählt werden, der möglichst nahe bei 1 liegt. Dieser Wert wurde zu 0,95 gewählt, was zwei Gründe hat: Einerseits soll dadurch die Möglichkeit in Betracht gezogen werden, dass der vektorwertige Klassifikator trotz $\vec{S}^{(N)}=\vec{g}_1$ mit einer gewissen Wahrscheinlichkeit eine Fehlklassifikation vornimmt (diese Wahrscheinlichkeit wird hier zu 0,05 angenommen). Andererseits sollte $\pi(\zeta_1|\vec{S}^{(N)})$ für $\vec{S}^{(N)}=\vec{g}_1$ nicht zu nahe bei 1 liegen, da dadurch

die Ähnlichkeit der Wahrscheinlichkeitsfunktion mit einer Stufenfunktion erhöht werden würde, was einer fein aufgelösten Verteilung der Werte der Wahrscheinlichkeitsfunktion entlang des Einheitskreises entgegenwirken würde. Mit Hilfe dieser Bedingung kann der Parameter a durch das Setzen von $\pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)=0{,}95$ und einfache Umformungen bestimmt werden. Auf diese Weise ergibt sich der folgende Wert für den Parameter a:

$$0{,}95=\frac{1}{1+e^{-a\left(\vec{S}^{(N)}\cdot\vec{g}_\sigma-b\right)}}\Leftrightarrow a=\frac{\ln(0{,}95)-\ln(0{,}05)}{1-b}=\frac{\ln(19)}{1-b}$$

Das Einsetzen von a und b, sowie auch des jeweiligen $\vec{g}_\sigma$ ergibt für jede Klasse eine separate Wahrscheinlichkeitsfunktion, mit deren Hilfe die nicht normierte Wahrscheinlichkeit berechnet werden kann, dass $\vec{S}^{(N)}$ einen Vertreter dieser Klasse repräsentiert. Unter Verwendung dieser Wahrscheinlichkeiten kann eine normierte Klassenwahrscheinlichkeit $\Pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)$ mit Hilfe der folgenden, einfachen Normierung berechnet werden:

$$\Pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)=\frac{\pi\left(\zeta_\sigma|\vec{S}^{(N)}\right)}{\sum_{\forall\rho}\pi\left(\zeta_\rho|\vec{S}^{(N)}\right)}$$

Unter Verwendung dieser normierten Wahrscheinlichkeiten kann für ein Residuum i, dass durch den Vektor $\vec{S}_i^{(N)}$ repräsentiert wird, ein Wahrscheinlichkeitsvektor $\vec{\Pi}_i\in\mathbb{R}^{N^{Klassen}}$ mit $\vec{\Pi}_i=(\Pi(\zeta_1|\vec{S}_i^{(N)}),\ldots,\Pi(\zeta_{N^{Klassen}}|\vec{S}_i^{(N)}))^T$ berechnet werden. Aus $\vec{S}_i$ und $\vec{\Pi}_i$ kann wiederum ein erweiterter Ausgangsvektor $\vec{S}_i^{(E)}\in\mathbb{R}^{2N^{Klassen}-1}$ der Form $\vec{S}_i^{(E)}=(\vec{S}_i,\vec{\Pi}_i)^T$ für die zweite Stufe konstruiert werden, der als alternative zu $\vec{S}_i$ zur Repräsentation des Residuums i verwendet werden kann. Auf diese Weise kann alternativ zu $\vec{x}_i^{(S)}$ als Eingangsvektor für das MLP auch der erweiterte Eingangsvektor $\vec{x}_i^{(SE)}$ verwendet werden. Dieser ergibt sich aus:

$$\vec{x}_i^{(SE)}=\left(\vec{S}_{i-n_3}^{(E)},\ldots,\vec{S}_i^{(E)},\ldots,\vec{S}_{i+n_3}^{(E)},T_i^{(C)},T_i^{(N)}\right)^T$$

Die vergrößerte Eingabeschicht, die sich bei Verwendung von $\vec{x}_i^{(SE)}$ anstelle von $\vec{x}_i^{(S)}$ ergibt, zieht eine Erhöhung der Anzahl der Parameter der MLP nach sich. Die Parameterzahl, die sich auf diese Weise ergibt berechnet sich aus:

$$P_{MLP}^{(E)}=((2\cdot N_2^{Klassen}-1)\cdot r_3+3)\cdot H_{MLP}+(H_{MLP}+1)\cdot N_3^{Klassen}$$

Die dritte Stufe *SPARROWs lässt beide Alternativen zur Repräsentation ihrer Eingangsdaten zu.

4.3.4. Qualitätsmaße

Um die Vorhersagequalität von *SPARROW auf einem beliebigen Datensatz messen zu können müssen einige Qualitätsmaße definiert werden, mit deren Hilfe die Vorhersagen *SPARROWs mit denen anderer Methoden in Beziehung gebracht werden können. Zur Bestimmung der Vorhersagequalität müssen sowohl die Eingangsvektoren als auch deren erwartete Klassifikation bekannt sein, da zur Berechnung der Vorhersagequalität die Information notwendig ist, ob eine Vorhersage richtig oder falsch ist. Hierbei wird von einem Datensatz der Form $D = \{(\vec{x}_i, \vec{y}_i) | 1 \le i \le M\}$ ausgegangen. In diesem Fall ist D ein beliebiger Datensatz, auf dem die Klassifikationen durchgeführt werden. Die Vorhersage selbst wird anhand des Skalarprodukts von $\vec{f}(\vec{x}, \vec{\theta})$ mit den einzelnen Klassenvektoren durchgeführt, wie bereits in Abschnitt 4.2.1 beschrieben wurde. Daraus ergibt sich, dass die Vorhersage eines $\vec{x}_i \in D$ richtig ist, wenn die folgende Bedingung gilt:

$$\vec{x} \in \zeta_\sigma \wedge \sigma = \underset{j}{\operatorname{argmax}} \left(\vec{f}(\vec{x}, \vec{\theta}) \cdot \vec{g}_j \right), \quad 1 \le \rho \le N^{Klassen}$$

Daraus ergibt sich der $Q_{N^{Klassen}}$-Index als einfaches Maß für die Vorhersagequalität auf dem Datensatz D. Der $Q_{N^{Klassen}}$-Index, oder auch *Korrektklassifikationsrate* (engl. *accuracy*), setzt die Zahl der korrekten Vorhersagen innerhalb eines Datensatzes zur Gesamtzahl der Vorhersagen in Verhältnis. Die Zahl, die sich auf diese Weise ergibt, ist 0 wenn alle gemachten Vorhersagen falsch sind und 1 wenn alle Vorhersagen korrekt sind. Damit ergibt sich der $Q_{N^{Klassen}}$-Index für den Datensatz D aus:

$$Q_{N^{Klassen}} = \frac{\sum_{\sigma=1}^{N^{Klassen}} \left| \left\{ \vec{x}_i \in D \middle| \vec{x}_i \in \zeta_\sigma \wedge \sigma = \underset{j}{\operatorname{argmax}} \left(\vec{f}(\vec{x}_i) \cdot \vec{g}_j \right) \right\} \right|}{M}$$

Auf ähnliche Weise kann für jede Klasse ζ_σ der Anteil der korrekt vorhergesagten Vertreter dieser Klasse über D berechnet werden. Diese Werte q_σ ergeben sich aus:

$$q_\sigma = \frac{\left| \left\{ \vec{x}_i \middle| \vec{x}_i \in D \in \zeta_\sigma \wedge \sigma = \underset{j}{\operatorname{argmax}} \left(\vec{f}(\vec{x}_i) \cdot \vec{g}_j \right) \right\} \right|}{\left| \left\{ \vec{x}_i | \vec{x}_i \in \zeta_\sigma \right\} \right|}$$

Eine Multiplikation von $Q_{N^{Klassen}}$ beziehungsweise q_σ mit 100 ergibt den Prozentsatz korrekt vorhergesagter Eingangsdaten im Datensatz D beziehungsweise in der Klasse ζ_σ.

Eine komplexeres Maß für die Vorhersagequalität stellt der *Pearsonsche Korrelationskoeffizient* dar, der auf Mehrklassenprobleme angewendet werden kann [85]. Diese Größe ergibt sich aus den Werten der *Konfusionsmatrix* (engl. *confusion matrix*) $\Omega \in \mathbb{N}^{N^{Klassen} \times N^{Klassen}}$. Ein Element dieser Matrix für D ist definiert durch:

$$\omega_{\sigma\rho} = \left| \left\{ \vec{x}_i \in D \,\middle|\, \vec{x}_i \in \zeta_\rho \wedge \sigma = \underset{j}{\operatorname{argmax}} \left(\vec{f}(\vec{x}_i) \cdot \vec{g}_j \right) \right\} \right|$$

Die einzelnen Elemente $\omega_{\sigma\rho}$ sagen damit aus, wie viele der Eingangsdaten $\vec{x}_i$, die eigentlich der Klasse ζ_ρ angehören, vom Klassifikator der Klasse ζ_σ zugewiesen werden. Die Matrix gibt also insgesamt an, wie stark der Klassifikator die einzelnen Klassen durcheinander bringt. Dabei stellen die Diagonalelemente korrekte Klassifikationen dar. Im Fall eines idealen Klassifikators wären die Diagonalelemente $\omega_{\sigma\sigma}$ also gleich der Anzahl der Vertreter der Klasse ζ_σ in D, und alle anderen Elemente gleich Null. Aus den Werten der Konfusionsmatrix kann nun laut [85] ein Koeffizient berechnet werden, der aussagt, wie stark die von einem Klassifikator erzeugte Vorhersage mit der realen Klassifikation korreliert. Dieser auf dem Pearsonschen Korrelationskoeffizienten basierende Korrelationskoeffizient ergibt sich aus:

$$R_{N^{Klassen}} = \frac{\sum_{\alpha=1}^{N^{Klassen}} \sum_{\beta=1}^{N^{Klassen}} \sum_{\gamma=1}^{N^{Klassen}} \left(\omega_{\alpha\alpha} \omega_{\beta\gamma} - \omega_{\alpha\beta} \omega_{\gamma\alpha} \right)}{\sqrt{\sum_{\alpha=1}^{N^{Klassen}} \left(\sum_{\beta=1}^{N^{Klassen}} \omega_{\alpha\beta} \sum_{\gamma=1}^{N^{Klassen}} \sum_{\delta=1, \gamma \neq \alpha}^{N^{Klassen}} \omega_{\delta\gamma} \right)} \sqrt{\sum_{\alpha=1}^{N^{Klassen}} \left(\sum_{\beta=1}^{N^{Klassen}} \omega_{\beta\alpha} \sum_{\gamma=1}^{N^{Klassen}} \sum_{\delta=1, \gamma \neq \alpha}^{N^{Klassen}} \omega_{\gamma\delta} \right)}}$$

Für $R_{N^{Klassen}}$ gilt $-1 \leq R_{N^{Klassen}} \leq 1$, wobei der Wert 1 vollständige Korrelation der Vorhersage des Klassifikators mit der realen Klassifikation bedeutet, der Wert -1 Antikorrelation beider Klassifikationen und der Wert 0 das Fehlen jeglicher Korrelation.

5. Ergebnisse

Dieses Kapitel soll die Ergebnisse zusammenfassen, die mit *SPARROW erzielt wurden. Die Tests mit dem Programm wurden in drei Phasen durchgeführt, deren Ergebnisse in chronologischer Reihenfolge präsentiert werden sollen. Diese Phasen spiegeln die einzelnen Entwicklungsstadien von *SPARROW wider: Vorbetrachtungen, Validierung und Anwendung. Die Vorbetrachtungen beinhalten Festlegen wie Bestimmen von Randbedingungen der einzelnen Stufen, wie Fenstergrößen, Anzahl an Klassen, verwendete Klasseneinteilungen und dergleichen. In der Validierungsphase wiederum wurden verschiedene Herangehensweisen an das Problem der Sekundärstrukturvorhersage mit Hilfe von *SPARROW untersucht und verglichen. Schließlich wurde in der Anwendungsphase der Vergleich zwischen *SPARROW und existierenden Programmen zur Sekundärstrukturvorhersage gezogen. Zunächst müssen jedoch, bevor auf die Untersuchungen und Ergebnisse aus den einzelnen Phasen eingegangen werden kann, die Daten beschrieben werden, auf denen diese Untersuchungen durchgeführt wurden.

5.1. Verwendete Daten

Die Daten, die in allen drei Testphasen verwendet wurden, entstammen der SCOP-Datenbank (Abkürzung von ***S**tructural **C**lassification **o**f **P**roteins*, [86], [87], [88], [89]). Dabei handelt es sich um eine Datenbank zur sequentiellen und strukturellen Analyse von Proteinen, in der die einzelnen Proteine entsprechend ihrer Verwandtschaft bezüglich Sequenz oder Struktur gruppiert sind. Aus dieser Datenbank wurde eine Teilmenge des auf *Domänen* (ähnlich wie Untereinheiten eines Proteins, siehe Abschnitt 2.5) basierenden ASTRAL-Datensatzes ([90], [91], [92]) für die Tests ausgewählt, nämlich der ASTRAL40-Datensatz. Dieser Datensatz wurde aufgrund seiner Besonderheit ausgesucht, dass alle in ihm enthaltenen Domänen untereinander in maximal 40% ihrer Sequenzen übereinstimmen. Dadurch treten nur wenige Ähnlichkeiten in der Sequenz der einzelnen Lernbeispiele auf, wodurch das Programm beim Lernen eine hohe und ausgewogene Vielfalt an Lernbeispielen erhält. Von den zur Verfügung stehenden Versionen wurde die Version 1.71 (erschienen im Januar 2007) ausgewählt, um als Lerndatensatz zu fungieren. Zusätzlich wurde der von Version 1.71 disjunkte Teil der Version 1.73 (erschienen im Februar 2008) als Testdatensatz für die Anwendungstests verwendet. Zur Zeit der Entwicklung von *SPARROW waren dies die beiden aktuellsten Versionen der ASTRAL40, weshalb

	Teilmenge von ASTRAL40 1.71		*Teilmenge von ASTRAL40 1.73*	
Anzahl an Domänen	7212		9472	
minimale Länge	20		21	
mittlere Länge	178,42		174,76	
maximale Länge	1422		1422	
	absolute Häufigkeit	relative Häufigkeit[%]	absolute Häufigkeit	relative Häufigkeit[%]
α-Helices	420429	32,72	546418	33,06
3_{10}-Helices	47565	3,7	60083	3,64
π-Helices	219	0,02	294	0,02
β-Stränge	271270	21,11	348991	21,12
β-Brücken	13887	1,08	17470	1,06
Windungen	146311	11,39	148580	8,99
Biegungen	116372	9,06	186196	11,27
strukturlos	268732	20,92	344571	20,85
Residuen insgesamt	1284785		1652603	

Tabelle 4: *Details der verwendeten Datensätze*

sie zur Durchführung der Tests verwendet wurden. Die später, im Juni 2009, erschienene Version 1.75 wurde nicht in Betracht gezogen. Tabelle 4 fasst einige Daten beider Datensätze zusammen, wobei für den Fall von ASTRAL40 1.73 die Daten des vollständigen Datensatzes aufgelistet sind. Wie bereits erwähnt, wird lediglich eine disjunkte Teilmenge dieser Daten als Testdatensatz verwendet, auf deren Erzeugung in Abschnitt 5.1.2 eingegangen wird.

Die Daten der einzelnen Domänen im ASTRAL40-Datensatz sind in Form von Atomkoordinaten vorhanden, wie sie auch aus der PDB heruntergeladen werden können, und entstammen kristallographischen Strukturen (Abschnitt 2.3.3.1). Aus den Daten wurden nur die Domänen verwendet, für die sowohl DSSP eine Referenzklassifikation (Abschnitt 4.1.3.2), als auch PSI-BLAST (Abschnitt 4.1.3.1) die als Eingangsdaten verwendeten Profile erzeugen konnte. Dies ist in Tabelle 4 bereits berücksichtigt. Die ursprünglichen ASTRAL40-Datensätze enthalten 7897 Domänen für Version 1.71 und 9536 Domänen für Version 1.73. Die Profile wurden mit PSI-BLAST unter Verwendung der *nr*-Datenbank generiert, einer nichtredundanten Datenbank von Proteinsequenzen, die aus unterschiedlichen Proteindatenbanken zusammengestellt wurde. Dabei wurde PSI-BLAST mit drei Iterationen und einem Schwellenwert von 0,001 für den e-Wert ausgeführt. Je geringer dieser Schwellenwert ist, desto mehr Signifikanz muss ein Fund in der Datenbank besitzen, um bei der Berechnung des Profils eines Proteins berücksichtigt zu werden ([78], [79]).

Datensatz		3_{10}-Helices	α-Helices	π-Helices	β-Stränge	β-Brücken	Windungen	Biegungen	strukturlos	insgesamt
KV_1	D_1^{lern}	41722	368739	174	237222	12209	128420	101994	235962	1126442
	D_1^{test}	4588	40622	28	26109	1278	14056	11365	26183	124229
KV_2	D_2^{lern}	41496	367123	197	237015	12031	127663	101799	235383	1122707
	D_2^{test}	4814	42238	5	26316	1456	14813	11560	26762	127964
KV_3	D_3^{lern}	41706	368329	192	237077	12165	128030	102254	236169	1125922
	D_3^{test}	4604	41032	10	26254	1322	14446	11105	25976	124749
KV_4	D_4^{lern}	41863	371038	172	236853	12116	128480	102082	236066	1128670
	D_4^{test}	4447	38323	30	26478	1371	13996	11277	26079	122001
KV_5	D_5^{lern}	41568	366922	187	237709	12039	128035	101966	235535	1123961
	D_5^{test}	4742	42439	15	25622	1448	14441	11393	26610	126710
KV_6	D_6^{lern}	41742	369617	182	237325	12052	128323	102322	236129	1127692
	D_6^{test}	4568	39744	20	26006	1435	14153	11037	26016	122979
KV_7	D_7^{lern}	42236	371471	186	237891	12332	129388	102558	237411	1133473
	D_7^{test}	4074	37890	16	25440	1155	13088	10801	24734	117198
KV_8	D_8^{lern}	41560	366363	180	236377	12172	127882	101909	235807	1122250
	D_8^{test}	4750	42998	22	26954	1315	14594	11450	26338	128421
KV_9	D_9^{lern}	41635	368546	171	237143	12138	128190	101599	235628	1125050
	D_9^{test}	4675	40815	31	26188	1349	14286	11760	26517	125621
KV_{10}	D_{10}^{lern}	41262	366101	177	235367	12129	127873	101748	235215	1119872
	D_{10}^{test}	5048	43260	25	27964	1358	14603	11611	26930	130799

Tabelle 5: *Kreuzvalidierungsdaten für die Validierungstests*

5.1.1. Daten für die Validierungsphase

Für die Validierungsphase wurde ausschließlich die Version 1.71 des ASTRAL40-Datensatzes verwendet. Da es Ziel dieser Phase war, unterschiedliche Ansätze zur Realisierung des endgültigen Vorhersageprogramms hinsichtlich Vorhersagequalität und Generalisierungsfähigkeit miteinander zu vergleichen, wurde in dieser Phase eine Kreuzvalidierung durchgeführt, wie sie in Abschnitt 3.1.4 beschrieben ist. Da auch in der Entwicklung von SPARROW solche Tests gemacht wurden, stand für die im Rahmen dieser Arbeit durchgeführten Tests eine Einteilung des ASTRAL40-Datensatzes in zehn Paarungen von Lern- und Testdaten zur Verfügung, wie sie in [1] beschrieben wurden. Diese Paarungen wurden erzeugt, indem aus den ursprünglichen Daten des ASTRAL40-Datensatzes zunächst alle Membranproteine entfernt wurden, sodass von den ursprünglichen 7212 Domänen 7040 für die Kreuzvalidierung übrig blieben. Mit $\beta = 10$ (siehe Abschnitt 3.1.4) ergaben sich daraus zehn Kombinationen von Lern- und Testdatensätzen, die für die Kreuzvalidierung verwendet wurden und in Tabelle 5 mit KV_α bezeichnet sind. Dabei beinhaltete jeder Lerndatensatz 6336 Domänen, während jeder Testdatensatz aus 704 Domänen bestand. Tabelle 5 enthält eine Auflistung aller zehn KV_α, die für die Kreuzvalidierung verwendet wurden, mitsamt der Verteilung der acht DSSP-Klassen und der Anzahl der Residuen in

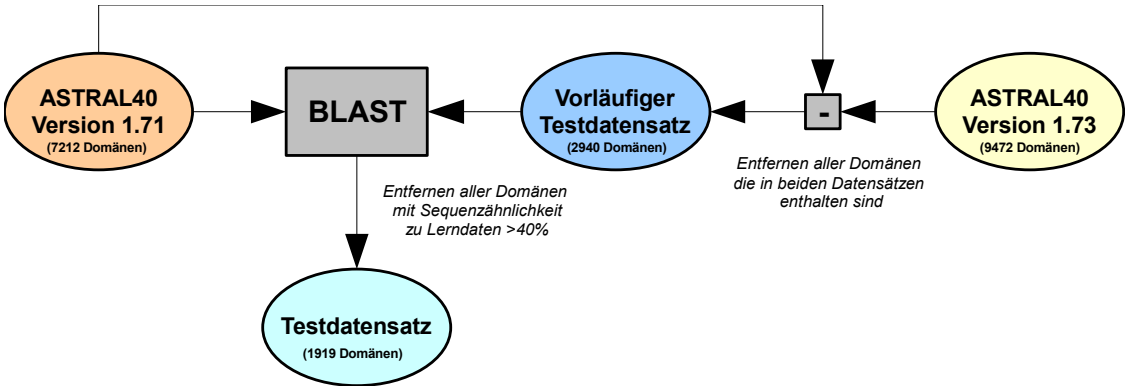


Abbildung 40: *Herstellung des Testdatensatzes für Vergleichstests*

den einzelnen Lerndatensätzen D_{α}^{lern} und Testdatensätzen D_{α}^{test}.

5.1.2. Daten für die Anwendungsphase

In der Anwendungsphase sollte der Vergleich zwischen dem endgültigen Vorhersageprogramm und anderen Programmen dieser Art gezogen werden. Die hierfür verwendete Methode, nebst verwendeter Daten, ist die gleiche, wie sie auch im Fall von SPARROW in [1] verwendet wurde: Der gesamte ASTRAL40-Datensatz in der Version 1.71, also alle in Tabelle 4 aufgeführten 7212 Domänen, diente als Lerndatensatz. Der Vergleich selbst wurde anhand des in [1] verwendeten Testdatensatzes vorgenommen. Dieser ist so konzipiert, dass keine der in ihm enthaltenen Domänen eine zu große Ähnlichkeit zu irgendeiner der Domänen des Lerndatensatzes besitzt. Um dies zu gewährleisten wurde in [1] die Version 1.73 des ASTRAL40-Datensatzes gefiltert. Hierzu wurden zunächst nur Domänen in Betracht gezogen, die nicht direkt im Lerndatensatz enthalten waren, sodass sich ein vorläufiger Testdatensatz von 2940 Domänen ergab. Die überraschend hohe Zahl ergibt sich daraus, dass einer neuen Version des ASTRAL40-Datensatzes nicht nur neue Domänen hinzugefügt werden, die in der älteren Version nicht vorhanden waren, sondern auch Domänen entfernt werden können, die in der älteren Version vorhanden waren. Um die bereits erwähnte geringe Ähnlichkeit zu den Lerndaten zu gewährleisten, wurde der vorläufige Testdatensatz weiter gefiltert. In diesem zweiten Schritt wurde BLAST [78] verwendet, um paarweise Alignments zwischen den Domänen des Lern- und denen des Testdatensatzes zu erzeugen. Daraufhin wurden alle Domänen aus dem vorläufigen Testdatensatz entfernt, die eine Sequenzähnlichkeit von mehr als 40% zu irgendeiner der Domänen des Lerndatensatzes besaßen. Die so übrig gebliebenen 1919 Domänen bilden den endgültigen

	Testdatensatz	
Anzahl an Domänen	1919	
minimale Länge	21	
mittlere Länge	171,17	
maximale Länge	1146	
	absolute Häufigkeit	relative Häufigkeit[%]
α-Helices	112147	34,14
3_{10}-Helices	11038	3,36
π-Helices	64	0,02
β-Stränge	68758	20,93
β-Brücken	3290	1,00
Windungen	35780	10,89
Biegungen	29255	8,91
strukturlos	68143	20,75
Residuen insgesamt	328475	

Tabelle 6: *Details des Testdatensatzes*

Testdatensatz. Der Vorgang, bei dem dieser Testdatensatz erzeugt wurde ist in Abbildung 40 in Form eines Flussdiagramms skizziert. Zusätzlich enthält Tabelle 6 einige Details des Testdatensatzes, wie Residuenzahl, Klassenverteilung und Sequenzlängen der Domänen.

5.2. Vorbetrachtungen

Das Ziel der Vorbetrachtungen, als erste Phase der Entwicklung von *SPARROW, war die Festlegung bestimmter Randbedingungen der einzelnen Stufen und Lernverfahren, die das Programm ausmachen. Hierzu gehörte einerseits die Wahl einiger Randbedingungen, ohne das hierfür Tests gemacht wurden, andererseits aber auch die Bestimmung von Parametern der Lernverfahren aus kleinen einfachen Tests heraus. Aufbauend auf den Ergebnissen dieser Phase sollte das weitere Vorgehen in den späteren Testphasen bestimmt sein. In diesem Abschnitt sollen zunächst die Randbedingungen beschrieben werden, die frei gewählt wurden. Daraufhin sollen die Tests für den vektorwertigen Klassifikator in der ersten und zweiten Stufe von *SPARROW behandelt werden, die zur Bestimmung der Regularisierungsparameter durchgeführt wurden.

5.2.1. Wahl von Randbedingungen

Zunächst stellt sich die Frage nach der zu verwendenden Größe für das Sequenzfenster in der ersten Stufe von *SPARROW. Francesco Bettella arbeitete bei SPARROW [1] mit einem Fenster von 15 Residuen und zeigte zudem, dass eine weiter Erhöhung der Fenstergröße eine geringfügige, jedoch noch messbare Verbesserung der Vorhersagequalität zur

Folge hat. Aus diesem Grund wurde für die erste Stufe von *SPARROW ein Fenster von 17 Residuen als Eingangsgröße gewählt. Dieselbe Fenstergröße wurde auch für die zweite Stufe von *SPARROW gewählt, sodass der vektorwertige Klassifikator dieser Stufe die Projektionsvektoren von 17 benachbarten Residuen für seine eigene Vorhersage verwendet.

Das MLP, welches in der dritten Stufe von *SPARROW Verwendung findet, wurde zur Verwendung sowohl für SPARROW wie auch *SPARROW entwickelt. In [1] fand dieses MLP bereits eine erste erfolgreiche Anwendung. Aufgrund der Erfahrungen mit SPARROWs MLP wurde das MLP für *SPARROW so dimensioniert, dass es hinsichtlich der Anzahl versteckter Neuronen (siehe Abschnitt 4.3.3), der Fenstergröße und Wahl der Lernkonstante für den Back-Propagations-Algorithmus (siehe Abschnitt 3.5.4.3) mit dem MLP in [1] identisch ist. Entsprechend [1] sind die jeweiligen Werte deshalb auf $H_{MLP}=100$, $r_3=15$ und $\eta=0{,}001$ gesetzt worden.

Die Anzahl der Klassen $N_1^{Klassen}$, $N_2^{Klassen}$ und $N_3^{Klassen}$, die die einzelnen Stufen von *SPARROW unterscheiden, wurden entsprechend den Vorgaben in Abschnitt 4.3 gewählt. Hierbei wurden mehrere Möglichkeiten untersucht, die verkürzt jeweils als $N_1^{Klassen}$ - $N_2^{Klassen}$ - $N_3^{Klassen}$ -Vorhersageschema, beziehungsweise als $N_1^{Klassen}$ - $N_2^{Klassen}(\pi)$ - $N_3^{Klassen}$ - Vorhersageschema bei Verwendung der in Abschnitt 4.3.3.1 beschriebenen Wahrscheinlichkeitsterme als zusätzliche Eingaben für die dritte Stufe, bezeichnet werden sollen. Diese Möglichkeiten sollen zunächst für das Dreiklassenproblem das 8-8-3-, das 8-8(π)-3- und das 3-3-3-Vorhersageschema sein. Für das Achtklassenproblem existieren entsprechend das 8-8-8- und das 8-8(π)-8-Vorhersageschema als Möglichkeiten zur Vorhersage der acht DSSP-Klassen.

5.2.2. Bestimmung der Regularisierungsparameter

Nach der Wahl der in Abschnitt 5.2.1 beschriebenen Randbedingungen bleibt lediglich die Frage nach den Werten von λ_{quad} und λ_{lin} (siehe Abschnitt 4.2.4.1) für die ersten beiden Stufen *SPARROWs offen. Beide Werte sollten so gewählt werden, dass das System mit möglichst wenig Überanpassung behaftet ist, also der Abstand zwischen dem Q_3-Index (siehe Abschnitt 4.3.4) auf den Lerndaten (im weiteren Text mit dem englischen Wort *Recall* bezeichnet) und dem auf den Testdaten (mit dem englischen Wort *Prediction* bezeich-

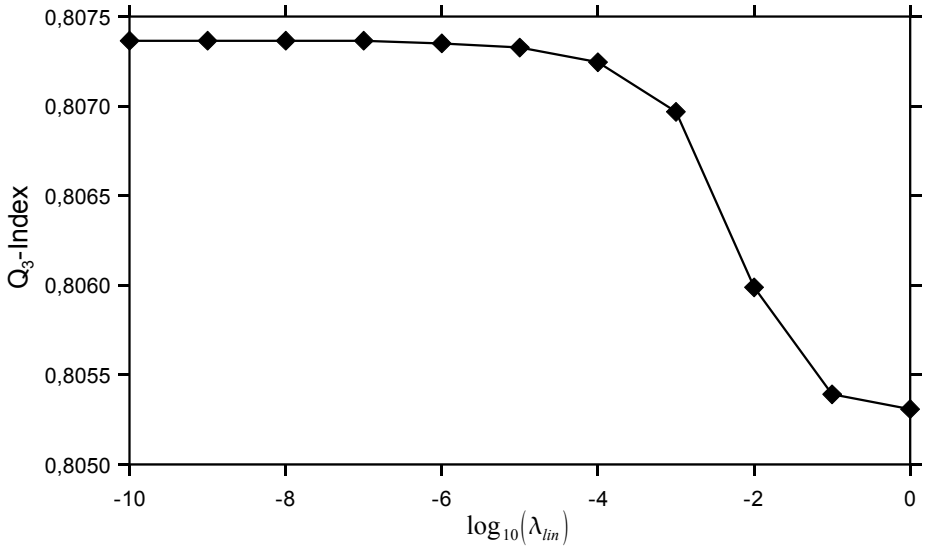


Abbildung 41: *Q_3-Index der Prediction in Abhängigkeit von λ_{lin}*

net) möglichst klein ist (siehe Abschnitt 3.1.4). Für die Bestimmung beider Regularisierungsparameter gibt es kein automatisiertes Optimierungsverfahren, sodass beide manuell durch ausprobieren einiger Werte bestimmt werden müssen. Laut [1] liegt der optimale Wert von λ_{quad} für die erste Stufe von SPARROW bei $\lambda_{quad}=10^{-4}$, weshalb dieser Wert für die erste Stufe von *SPARROW übernommen wurde. Zur Bestimmung von λ_{lin} für *SPARROW wurde ein einfacher Test durchgeführt: Zunächst wurden elf Werte von λ_{lin} ausgesucht, für die ein Lernvorgang mittels des vektorwertigen Klassifikators der ersten Stufe durchgeführt werden sollte. Diese Werte stellen die zwischen 10^{-10} und 1 liegenden negativen Zehnerpotenzen dar. Mit jedem dieser elf Werte von λ_{lin} sowie $\lambda_{quad}=10^{-4}$ wurde jeweils ein vektorwertiger Klassifikator für die erste Stufe unter Verwendung der strengen Klasseneinteilung (siehe Tabelle 3 auf Seite 113) trainiert. Hierzu wurde die in Tabelle 5 mit KV_1 gekennzeichnete Paarung von Lern- und Testdaten herangezogen, die auch schon in [1] zur Optimierung von λ_{lin} verwendet wurde. Das Lernen auf D_1^{lern} wurde zur Ersparnis von Speicherplatz (für die Speicherung der Koeffizientenmatrix des linearen Gleichungssystems) und Zeit (bei dessen Lösung) mit einem Fenster von elf Residuen durchgeführt. Dann wurde für jeden der Klassifikatoren der Q_3 der *Prediction* mit Hilfe von D_1^{test} gemessen. Laut [1] sollte dieser bis zum optimalen Wert von λ_{lin} hin steigen und danach stark abfallen. Die Ergebnisse des beschriebenen Tests sind in Abbildung 41 gezeigt. Zu sehen ist der Q_3-Index als Funktion von λ_{lin}. Aus dem Graphen geht hervor,

dass im untersuchten Bereich ein möglichst kleines λ_{lin} gewählt werden sollte, da eine Erhöhung des Wertes zu $\lambda_{lin}=1$ hin zu einer Senkung des Q_3-Index beiträgt. Aus diesem Grund wurde für die erste Stufe von *SPARROW der Wert $\lambda_{lin}=10^{-10}$ verwendet. Dieser Wert ist kleiner als der in [1] verwendete Wert von $\lambda_{lin}=10^{-5}$.

Da die Größe des Sequenzfensters die optimalen Werte der Regularisierungsparameter beeinflusst, muss überprüft werden, ob die bestimmten Werte auch für ein Sequenzfenster von 17 Residuen verwendet werden können. Hierzu wurde ein Lernvorgang für den vektorwertigen Klassifikator der ersten Stufe mit $\lambda_{lin}=10^{-10}$, $\lambda_{lin}=10^{-4}$, $r_1=17$ und D_1^{lern} als Lerndatensatz durchgeführt, wobei auch hier die strenge Klasseneinteilung verwendet wurde. Die Werte für Q_3 ergeben sich in diesem Fall zu 0,82 für den *Recall* und 0,8211 für die *Prediction*. Die Tatsache, dass die *Prediction* größer ist als der *Recall*, kann sich als statistische Fluktuation aus dem Größenverhältnis von Lern- und Testdatensatz ergeben. Prinzipiell sollte der *Recall* stets über der *Prediction* liegen, da die Diskrepanz einer Lernmaschine auf den Lerndaten stets kleiner sein sollte als auf den Testdaten (siehe Abschnitt 3.1.4, Abbildung 22). Da die Regularisierungsparameter jedoch in diesem Fall wie auch in [1] unter Zuhilfenahme von D_1^{test} optimiert wurden, müssen *Recall* und *Prediction* für KV_1 nahe beieinander liegen. Durch das Größenverhältnis von Lern- und Testdatensatz kann es jedoch zu einer geringfügigen Abweichung der *Prediction* kommen, was zu leicht kleineren oder, wie in diesem Fall, leicht größeren Werten führen kann. Nichtsdestotrotz zeigen die Werte für *Prediction* und *Recall*, dass die bestimmten Werte für λ_{quad} und λ_{lin} tatsächlich den erwarteten Effekt haben.

Für die zweite Stufe kann [1] entnommen werden, dass beide Regularisierungsparameter möglichst klein sein sollten, da jede Erhöhung dieser Parameter sich negativ auf den Q_3-Index auswirkt. Aus diesem Grund wurden die Werte $\lambda_{quad}=10^{-10}$ und $\lambda_{lin}=10^{-10}$ gewählt. Der vektorwertige Klassifikator der zweiten Stufe *SPARROWs wurde mit diesen Regularisierungsparametern trainiert ($r_2=17$, strenge Klasseneinteilung, Lerndatensatz D_1^{lern}) und die Q_3-Indices für D_1^{lern} und D_1^{test} gemessen. Aus den Werten 0,8244 für *Recall* und 0,8248 für *Prediction* lässt sich folgern, dass das Maß an Überanpassung im Übergang von der ersten zur zweiten Stufe nicht merklich zugenommen hat, wodurch eine weitere Anpassung der Regularisierungsparameter der zweite Stufe nicht notwendig erscheint.

5.3. Validierung und Methodenauswahl

Nachdem in der Vorbetrachtungsphase die Randbedingungen für *SPARROW gesetzt wurden, soll die Validierungsphase darüber Aufschluss geben, wie sich das Programm unter Verwendung dieser Randbedingungen verhält. Die hierfür durchgeführten Untersuchungen besitzen zwei Zielsetzungen: Einerseits muss, wie bereits in Abschnitt 5.1.1 erwähnt, die Generalisierungsfähigkeit der einzelnen Vorhersageschemata abgeschätzt und verglichen werden. Andererseits soll mit Hilfe dieser Tests aus den zur Verfügung stehenden Vorhersageschemata (Abschnitt 5.2.1) dasjenige bestimmt werden, das in der endgültigen Version von *SPARROW Anwendung finden soll.

Bei den durchgeführten Tests wurde dem Dreiklassenproblem die meiste Aufmerksamkeit geschenkt, da die meisten Programme zur Sekundärstrukturvorhersage dieses Problem betrachten. Zusätzlich wurden jedoch auch einige Untersuchungen des Achtklassenproblems durchgeführt, um die Möglichkeit der Verwendung von *SPARROW als Programm zur Vorhersage der acht DSSP-Klassen auszuloten. Sowohl für das Dreiklassenproblem wie auch für das Achtklassenproblem wurde, wie in Abschnitt 5.1.1 beschrieben, eine Kreuzvalidierung durchgeführt. Anschließend wurde die Abschätzung der Generalisierungsfähigkeit wie auch der Vergleich der Vorhersageschemata, die als Lösung des jeweiligen Klassifikationsproblems in Frage kamen, anhand der Mittelwerte und Standardabweichungen der aus der Kreuzvalidierung gewonnenen Qualitätsmaße $Q_{N_3^{Klassen}}$, $R_{N_3^{Klassen}}$ und q_σ (Definition in Abschnitt 4.3.4) durchgeführt. Dabei entspricht $N_3^{Klassen}$, wie aus Abschnitt 4.3.3 hervorgeht, der Anzahl der Klassen im zu lösenden Klassifikationsproblem. Die Werte der einzelnen Qualitätsmaße für jede Paarung von Lern- und Testdatensatz KV_α (Tabelle 5 auf Seite 147) wurden bestimmt, indem *SPARROW zunächst mit dem jeweiligen D_α^{lern} trainiert wurde und anschließend die Qualitätsmaße für eine Vorhersage von D_α^{lern} (*Recall*) und dem dazugehörigen D_α^{test} (*Prediction*) für jede Stufe berechnet wurden. Für Vorhersageschemata, bei denen die Anzahl an Klassen in der ersten ($N_1^{Klassen}$, siehe Abschnitt 4.3.1) oder der zweiten Stufe ($N_2^{Klassen}$, siehe Abschnitt 4.3.2) größer als $N_3^{Klassen}$ war, wie es zum Beispiel beim 8-8-3-Vorhersageschema der Fall ist, musste die Berechnung der Qualitätsmaße auf $N_3^{Klassen}$ Klassen zurückgeführt werden, da diese Qualitätsmaße nur $N_3^{Klassen}$ Klassen berücksichtigen. Die gesuchten Qualitätsmaße lassen sich jedoch beispielsweise

für die zweite Stufen berechnen, indem die entsprechende $N_2^{Klassen} \times N_2^{Klassen}$-Konfusionsmatrix (siehe Abschnitt 4.3.4) zu einer $N_3^{Klassen} \times N_3^{Klassen}$-Konfusionsmatrix reduziert wird. Durch die Vereinigung von Klassen, die für den Übergang von $N_2^{Klassen}$ zu $N_3^{Klassen}$ Klassen notwendig ist, ergibt sich ein einfaches Schema, nach dem Einträge der ursprünglichen $N_2^{Klassen} \times N_2^{Klassen}$-Konfusionsmatrix addiert werden müssen, um einen Eintrag der reduzierten Konfusionsmatrix zu ergeben. Mit dieser reduzierten Konfusionsmatrix wiederum können die gesuchten Qualitätsmaße einfach berechnet werden.

Weiterhin ist bei der dritten Stufe eine gesonderte Betrachtung bei der Berechnung der Qualitätsmaße notwendig: Da das Lernverfahren des MLP in der dritten Stufe aufgrund der zufälligen Reihenfolge bei der Präsentation der Lernbeispiele nichtdeterministisch ist, ergibt mehrmaliges Lernen auf dem gleichen Lerndatensatz in der Regel unterschiedliche Parametersätze. Um das Verhalten der dritten Stufe besser Abschätzen zu können, wurde der Lernvorgang für das MLP mit D_α^{lern} zehn Mal ausgeführt und die Qualitätsmaße für D_α^{lern} und D_α^{test} für jeden dieser zehn Lernvorgänge ermittelt. Die Qualitätsmaße für die dritte Stufe für KV_α ergaben sich so als Mittelwerte über die zehn Lernvorgänge, während sie bei den ersten beiden Stufen direkt aus der Vorhersage der entsprechenden Datensätze ergaben.

Nach der obigen Vorgehensweise standen für alle Stufen eines untersuchten Vorhersageschemas und alle KV_α Werte der für *Recall* und *Prediction* verwendeten Qualitätsmaße zur Verfügung, mit deren Hilfe Mittelwerte und Standardabweichungen für die einzelnen Stufen berechnet werden konnten. Bei der Mittelwertbildung wurde KV_1 außer acht gelassen, da D_1^{test} zur Optimierung der Regularisierungsparameter verwendet wurde und somit im Fall von KV_1 sowohl D_1^{lern} auf direktem Wege, wie auch D_1^{test} indirekt, in den Lernvorgang einfließen. Deshalb ist die *Prediction* auf D_1^{test} deutlich besser als es bei den anderen KV_α der Fall ist. Zusätzlich sagt die *Prediction* auf D_1^{test} wegen dessen Einfluss auf das Lernverfahren nichts über die Generalisierungsfähigkeit des jeweiligen Vorhersageschemas unter Verwendung von KV_1 aus. KV_1 würde aus diesen beiden Gründen die berechneten Mittelwerte für die einzelnen Vorhersageschemata verfälschen, weshalb es bei der Berechnung der Mittelwerte und Standardabweichungen nicht berücksichtigt wurde.

	1.Stufe		*2.Stufe*		*3. Stufe*			
	8-Vorhersageschema		8-8-Vorhersageschema		8-8-8-Vorhersageschema		8-8(π)-8-Vorhersageschema	
	Recall	Prediction	Recall	Prediction	Recall	Prediction	Recall	Prediction
Q_3	0,6718 ± 0,0005	0,6630 ± 0,0041	0,6924 ± 0,0006	0,6774 ± 0,0045	0,6975 ± 0,0006	0,6806 ± 0,0049	0,7002 ± 0,0007	0,6828 ± 0,0050
R_3	0,5712 ± 0,0007	0,5593 ± 0,0053	0,5990 ± 0,0007	0,5789 ± 0,0057	0,6079 ± 0,0009	0,5857 ± 0,0061	0,6116 ± 0,0008	0,5887 ± 0,0062
q_G	0,0288 ± 0,0008	0,0239 ± 0,0035	0,1592 ± 0,0013	0,1308 ± 0,0077	0,2484 ± 0,0085	0,2048 ± 0,0132	0,2701 ± 0,0114	0,2217 ± 0,0128
q_H	0,9242 ± 0,0004	0,9205 ± 0,0043	0,9272 ± 0,0004	0,9223 ± 0,0048	0,9051 ± 0,0018	0,8988 ± 0,0059	0,9064 ± 0,0025	0,9000 ± 0,0065
q_I	0,0000 ± 0,0000	0,0000 ± 0,0000	0,1898 ± 0,0563	0,0654 ± 0,0995	0,2927 ± 0,0422	0,0654 ± 0,0995	0,3778 ± 0,2348	0,0654 ± 0,0995
q_E	0,8097 ± 0,0007	0,8012 ± 0,0062	0,8171 ± 0,0009	0,8058 ± 0,0053	0,8226 ± 0,0088	0,8107 ± 0,0120	0,8274 ± 0,0049	0,8163 ± 0,0061
q_B	0,0011 ± 0,0003	0,0008 ± 0,0008	0,0925 ± 0,0028	0,0606 ± 0,0066	0,1344 ± 0,0039	0,0886 ± 0,0071	0,1449 ± 0,0114	0,0952 ± 0,0124
q_T	0,4432 ± 0,0012	0,4293 ± 0,0099	0,4570 ± 0,0014	0,4349 ± 0,0099	0,4910 ± 0,0093	0,4663 ± 0,0108	0,4947 ± 0,0051	0,4694 ± 0,0118
q_S	0,1119 ± 0,0010	0,0983 ± 0,0046	0,2245 ± 0,0009	0,1951 ± 0,0049	0,2771 ± 0,0095	0,2430 ± 0,0105	0,2892 ± 0,0127	0,2540 ± 0,0156
q_C	0,6541 ± 0,0011	0,6406 ± 0,0045	0,6562 ± 0,0009	0,6349 ± 0,0047	0,6505 ± 0,0111	0,6289 ± 0,0113	0,6449 ± 0,0107	0,6230 ± 0,0137

Tabelle 7: *Ergebnisse der Kreuzvalidierung für die Lösung des Achtklassenproblems*

5.3.1. Das Achtklassenproblem

Intuitiv wäre die naheliegendste Herangehensweise an das Problem der Sekundärstrukturvorhersage von Proteinen die Verwendung der acht Sekundärstrukturklassen wie sie von DSSP vorgegeben werden, da so die Ausgaben von DSSP direkt als Referenz verwendet werden können. Die direkte Vorhersage der acht DSSP-Klassen erfordert die Lösung eines Achtklassenproblems. Hierfür kommen prinzipiell nur zwei Vorhersageschemata in Frage, nämlich das 8-8-8-Vorhersageschema und das 8-8(π)-8-Vorhersageschema, das sich aus der Modifikation des 8-8-8-Vorhersageschemas nach der in Abschnitt 4.3.3.1 beschriebenen Methode ergibt.

Tabelle 7 präsentiert die Ergebnisse, die mit beiden Vorhersageschemata in der Kreuzvalidierung erreicht wurden. Die Tabelle enthält für jede Stufe beider Vorhersageschemata die Mittelwerte und Standardabweichungen von Q_8 und R_8 als auch für die q_σ (siehe Abschnitt 4.3.4) der einzelnen Klassen. Da beide Vorhersageschemata, was ihre ersten beiden Stufen angeht, identisch sind, enthält Tabelle 7 die Ergebnisse für diese beiden Stufen jeweils nur einmal. Die Bezeichnungen beider Stufen ergeben sich, ähnlich wie bei den vollständigen Vorhersageschemata, aus $N_1^{Klassen}$-Vorhersageschema für die erste Stufe und $N_1^{Klassen}$ - $N_2^{Klassen}$-Vorhersageschema für die zweite Stufe. Die dritten Stufen, in denen sich die beiden Vorhersageschemata unterscheiden, sind in Form ihrer Ergebnisse separat in Tabelle 7 aufgeführt und repräsentieren die Ergebnisse, die mit dem jeweiligen vollständigen Vorhersageschema erreicht wurden.

Wie Tabelle 7 zunächst entnommen werden kann, nehmen die Werte von Q_8 und R_8 mit jeder weiteren verwendeten Stufe sowohl im *Recall* wie auch in der *Prediction* zu. Dabei fällt auf, dass die Verbesserung in Q_8 und R_8 im Übergang von der ersten zur zweiten Stufe größer ist, als die Verbesserung, die die jeweilige dritte Stufe gegenüber der zweiten erreicht. So ist der Wert von Q_8 in der zweiten Stufe gegenüber der ersten um 0,0206 im *Recall* und 0,0144 in der *Prediction* erhöht. Im Übergang zur dritten Stufe ist für Q_8 beim 8-8-8-Vorhersageschema jedoch lediglich eine Zunahme von 0,0051 im *Recall* und 0,0032 in der *Prediction*, beziehungsweise 0,0078 und 0,0054 beim 8-8(π)-8-Vorhersageschema, zu verzeichnen. Dies zeigt auch, dass das 8-8(π)-8-Vorhersageschema gegenüber dem 8-8-8-Vorhersageschema zu einer, wenn auch geringen, Erhöhung des Q_8-Index in *Recall* wie in *Prediction* führt. Was die Standardabweichung von Q_8 angeht, so fällt diese in der *Prediction* bei allen Stufen höher aus als im *Recall*, was sich jedoch als Konsequenz aus dem Größenverhältnis der einzelnen Lern- und Testdatensätze ergibt (siehe Tabelle 5 auf Seite 147). Im Übergang von einer Stufe zur nächsten scheinen die Standardabweichungen von *Recall* und *Prediction*, bis auf eine geringfügige Erhöhung, im Wesentlichen unverändert. Ein ähnliches Verhalten, wie es für die Mittelwerte und Standardabweichungen von Q_8 festgestellt wurde, weisen auch die Mittelwerte und Standardabweichungen von R_8 auf, wie sich an Tabelle 7 ablesen lässt.

Eine Betrachtung der Vorhersage der einzelnen Klassen anhand der jeweiligen q_σ der ersten Stufe in Tabelle 7 offenbart unter Berücksichtigung von Tabelle 5 auf Seite 147, dass das Vorhersageverhalten der einzelnen Klassen mit dem Auftreten dieser Klassen innerhalb der Kreuzvalidierungsdaten korreliert. So werden α-Helices, die die Klasse mit den meisten Vertretern bilden, in *Recall* und *Prediction* am besten vorhergesagt, während π-Helices als kleinste Klasse die schlechteste Vorhersage aufweisen. Dabei fällt auf, dass die Werte von q_σ für die kleinsten Klassen, also π-Helices und isolierte β-Brücken vollkommen beziehungsweise annähernd verschwinden. Durch die Verwendung der zweiten Stufe kann, wie Tabelle 7 zeigt, eine Verbesserung der Vorhersage aller Klassen erreicht werden, was sich in erhöhten Mittelwerten der jeweiligen q_σ äußert und insbesondere im Fall der kleinen Klassen bemerkenswert ist. So steigt q_I von einem Wert von 0, sowohl in *Recall* wie in *Prediction*, auf 0,1898 im *Recall* und 0,0654 in der *Prediction*. Die hohe Standardabweichung von q_I weist jedoch darauf hin, dass die Vorhersage der π-Helices nicht in

allen Fällen so sicher ist, wie die Tabelle es vermuten ließe. Dies trifft insbesondere für die *Prediction* zu, die eine Standardabweichung von 0,0995 aufweist, welche den Mittelwert von q_I in diesem Fall übersteigt. Im Fall von q_B findet eine Verbesserung von dem ursprünglichen Wert von 0,0011 (*Recall*) beziehungsweise 0,0008 (*Prediction*) in der ersten Stufe auf einen Wert von 0,0925 beziehungsweise 0,0606 in der zweiten Stufe statt, was eine beträchtliche Verbesserung in der Vorhersage dieser Klasse darstellt. Zudem zeugen die entsprechenden Standardabweichungen von der Stabilität der Vorhersage der isolierten β-Brücken hinsichtlich der Werte von q_B auf den einzelnen Kreuzvalidierungsdaten.

Neben den beiden erwähnten Klassen erfahren auch die übrigen Klassen durch die zweite Stufe eine Verbesserung, die jedoch mit der Größe der jeweiligen Klasse abzunehmen scheint. Lediglich die strukturlosen Regionen stellen in diesem Zusammenhang eine Ausnahme dar, da der Wert von q_C in der *Prediction* der zweiten Stufe geringer ausfällt als es in der ersten Stufe der Fall war. Da für die *Prediction* diese Abnahme von q_C dessen Standardabweichung übersteigt, kann ein statistischer Effekt ausgeschlossen werden. Es scheint jedoch in Anbetracht der Tatsache, dass die zweite Stufe gegenüber der ersten erhöhte Werte von Q_8 und R_8 aufweist, dass die Abnahme von q_C in der *Prediction* durch die Verbesserung der Vorhersage der übrigen Klassen mehr als kompensiert wird. Im *Recall* ist hingegen eine, wenn auch geringe, Verbesserung der Vorhersage strukturloser Regionen in der zweiten Stufe zu verzeichnen.

Die Vorhersage der einzelnen Klassen wird durch die dritte Stufe in fast allen Fällen weiter verbessert, wobei wie bereits bei Q_8 und R_8 zwei Aussagen gemacht werden können: Einerseits ist das Ausmaß der Verbesserung, die durch beide Vorhersageschemata für die dritte Stufe erzielt wird, im Allgemeinen kleiner als die Verbesserung, die die zweite Stufe gegenüber der ersten Stufe aufweist. Andererseits ist die Verbesserung, die das 8-8(π)-8-Vorhersageschema gegenüber der zweiten Stufe erzielt, im Vergleich zum 8-8-8-Vorhersageschema höher. Beide Aussagen gelten sowohl für *Recall* wie für *Prediction*. Für beide Aussagen gibt es jedoch zwei Ausnahmen. Diese sind, wie bereits im Fall der zweiten Stufe, die strukturlosen Regionen und α-Helices. Für die strukturlosen Regionen stellt die dritte Stufe gegenüber der zweiten für beide Vorhersageschemata eine weitere Abnahme von q_C dar, die im Fall des 8-8(π)-8-Vorhersageschemas höher ausfällt als im Fall des 8-8-8-Vorhersageschemas. Auch im Fall von q_H ist für beide Vorhersageschemata eine Ab-

	locker		Q_3	R_3	q_H	q_E	q_C
1.Stufe	8-VS	*Recall*	0,7866 ± 0,0005	0,6748 ± 0,0007	0,8792 ± 0,0005	0,7851 ± 0,0008	0,7058 ± 0,0010
		Prediction	0,7803 ± 0,0033	0,6653 ± 0,0052	0,8755 ± 0,0042	0,7769 ± 0,0054	0,6984 ± 0,0056
	3-VS	*Recall*	0,8018 ± 0,0005	0,6920 ± 0,0007	0,8306 ± 0,0006	0,6960 ± 0,0010	0,8331 ± 0,0007
		Prediction	0,7953 ± 0,0037	0,6817 ± 0,0058	0,8255 ± 0,0059	0,6868 ± 0,0064	0,8267 ± 0,0046
2. Stufe	8-8-VS	*Recall*	0,8015 ± 0,0004	0,6967 ± 0,0007	0,8884 ± 0,0004	0,7955 ± 0,0009	0,7281 ± 0,0008
		Prediction	0,7918 ± 0,0033	0,6819 ± 0,0051	0,8822 ± 0,0043	0,7834 ± 0,0047	0,7166 ± 0,0050
	3-3-VS	*Recall*	0,8049 ± 0,0005	0,6968 ± 0,0007	0,8340 ± 0,0006	0,6875 ± 0,0007	0,8420 ± 0,0004
		Prediction	0,7982 ± 0,0038	0,6863 ± 0,0059	0,8289 ± 0,0059	0,6780 ± 0,0064	0,8354 ± 0,0049
3. Stufe	8-8-3-VS	*Recall*	0,8177 ± 0,0006	0,7175 ± 0,0009	0,8513 ± 0,0033	0,7655 ± 0,0100	0,8160 ± 0,0060
		Prediction	0,8071 ± 0,0036	0,7011 ± 0,0056	0,8426 ± 0,0060	0,7519 ± 0,0118	0,8053 ± 0,0082
	8-8(π)-3-VS	*Recall*	0,8194 ± 0,0004	0,7203 ± 0,0007	0,8559 ± 0,0045	0,7697 ± 0,0074	0,8139 ± 0,0068
		Prediction	0,8087 ± 0,0034	0,7036 ± 0,0054	0,8470 ± 0,0067	0,7562 ± 0,0066	0,8031 ± 0,0087
	3-3-3-VS	*Recall*	0,8087 ± 0,0006	0,7034 ± 0,0010	0,8414 ± 0,0046	0,7475 ± 0,0073	0,8128 ± 0,0051
		Prediction	0,8016 ± 0,0036	0,6923 ± 0,0057	0,8359 ± 0,0073	0,7382 ± 0,0077	0,8053 ± 0,0077

Tabelle 8: *Verhalten der einzelnen Stufen der untersuchten Vorhersageschemata (abgekürzt VS) für die lockere Klasseneinteilung*

nahme in *Recall* und *Prediction* zu verzeichnen, jedoch sind die Unterschiede in den Werten von q_H zwischen den beiden Vorhersageschemata geringer die entsprechenden Unterschiede für q_C.

5.3.2. Das Dreiklassenproblem

Nach Abschnitt 4.1.3.2 kommen für das Dreiklassenproblem der Sekundärstrukturvorhersage zwei mögliche Klasseneinteilungen in Frage, nämlich die strenge und die lockere Klasseneinteilung, die in Tabelle 3 auf Seite 113 dargestellt sind. Die Vorhersageschemata, die zur Lösung dieser beiden, durch die jeweilige Klasseneinteilung definierten Dreiklassenprobleme verwendetet wurden, beschränkten sich zunächst auf das 3-3-3-Vorhersageschema sowie das 8-8-3-Vorhersageschema und dessen Erweiterung in Form des 8-8(π)-3-Vorhersageschemas. Die Ergebnisse der Kreuzvalidierung mit diesen drei Vorhersageschemata sind für die lockere Klasseneinteilung in Form der Qualitätsmaße Q_3, R_3, q_H, q_E und q_C in Tabelle 8 dargestellt. Dabei werden, wie auch schon in Tabelle 7 für das Achtklassenproblem, die ersten Stufen der einzelnen Vorhersageschemata, abgekürzt VS, jeweils als $N_1^{Klassen}$-VS, die zweiten Stufen als $N_1^{Klassen}$ - $N_2^{Klassen}$-VS und die dritten Stufen mit der vollständigen Bezeichnung des jeweiligen Vorhersageschemas aufgeführt. Da das 8-8-3- und das 8-8(π)-3-Vorhersageschema in ihren ersten beiden Stufen identisch sind, enthält Tabelle 8 diese jeweils nur einmal.

Eine erste Betrachtung der Mittelwerte von Q_3 und R_3 in Tabelle 8 zeigt, dass jede weitere Stufe eines Vorhersageschemas zu einer Verbesserung von Q_3 und R_3 für dieses Vorhersageschema, sowohl in *Recall* wie auch in *Prediction*, führt. Dabei ist festzustellen, dass die ersten beiden Stufen des 3-3-3-Vorhersageschemas höhere Werte von Q_3 und R_3 aufweisen als die entsprechenden Stufen des 8-8-3- beziehungsweise 8-8(π)-3-Vorhersageschemas. Jedoch zeigt das Verhalten der dritten Stufen des 8-8-3- und des 8-8(π)-3-Vorhersageschemas, dass die Vorhersagen der zweiten Stufe dieser beiden Vorhersageschemata durchaus zur Vorhersage dreier Klassen mittels eines entsprechend optimierten MLPs verwendet werden können. So zeigt sich, dass in der dritten Stufe das 8-8(π)-3-Vorhersageschema in *Recall* und *Prediction* die höchsten Werte von Q_3 und R_3 aufweist, gefolgt von dem 8-8-3- und schließlich dem 3-3-3-Vorhersageschema. Dabei übersteigt der Unterschied, der sich zwischen dem 3-3-3- und dem 8-8-3-Vorhersageschema auf Q_3 beziehungsweise R_3 ergibt, in *Recall* und *Prediction* sowohl für Q_3 als auch für R_3 die Standardabweichung des jeweiligen Wertes. Dadurch kann ein statistischer Effekt als Ursache der beobachteten Abweichungen ausgeschlossen werden. Weiterhin fällt auf, dass die Differenz der Werte von Q_3 beziehungsweise R_3, die zwischen dem 8-8-8- und dem 8-8(π)-3-Vorhersageschema festgestellt werden kann, in *Recall* und *Prediction* jeweils eine ähnliche Größenordnung besitzt.

Die Betrachtung von q_H, q_E und q_C fördert einen weiteren interessanten Unterschied zwischen den einzelnen Vorhersageschemata zu Tage. So scheint es, dass in den ersten zwei Stufen des 3-3-3-Vorhersageschemas das Vorhersageverhalten der drei Klassen mit ihrem Auftreten in den verwendeten Datensätzen korreliert. Aus Tabelle 5 auf Seite 147 lässt sich berechnen, dass die Kreuzvalidierungsdaten im Fall der lockeren Klasseneinteilung im Mittel 36,45% Helices, 22,13% β-Stränge und 41,42% strukturlose Regionen enthalten. Entsprechend zeigt Tabelle 8, dass die ersten beiden Stufen des 3-3-3-Vorhersageschemas die strukturlosen Regionen, als größte der drei Klassen, am besten vorhersagen, während die β-Stränge, welche die kleinste Klasse bilden, am schlechtesten vorhergesagt werden. Bei den ersten beiden Stufen des 8-8-3- und 8-8(π)-3-Vorhersageschemas hingegen weist die helikale Klasse das beste Vorhersageverhalten auf, während strukturlose Regionen das schlechteste Vorhersageverhalten besitzen. Zudem ist der Unterschied zwischen den einzelnen q_σ stärker ausgeprägt als bei den ersten beiden Stufen des 3-3-3-Vorhersa-

streng			Q_3	R_3	q_H	q_E	q_C
1.Stufe	8-VS	*Recall*	0,7883 ± 0,0005	0,6836 ± 0,0008	0,9242 ± 0,0004	0,8097 ± 0,0007	0,6824 ± 0,0010
		Prediction	0,7822 ± 0,0036	0,6744 ± 0,0056	0,9205 ± 0,0043	0,8012 ± 0,0062	0,6756 ± 0,0059
	3-VS	*Recall*	0,8205 ± 0,0005	0,7148 ± 0,0008	0,8345 ± 0,0007	0,6889 ± 0,0011	0,8706 ± 0,0006
		Prediction	0,8148 ± 0,0042	0,7056 ± 0,0065	0,8293 ± 0,0068	0,6801 ± 0,0072	0,8660 ± 0,0042
2. Stufe	8-8-VS	*Recall*	0,8067 ± 0,0005	0,7083 ± 0,0007	0,9272 ± 0,0004	0,8171 ± 0,0009	0,7167 ± 0,0008
		Prediction	0,7977 ± 0,0039	0,6948 ± 0,0060	0,9223 ± 0,0048	0,8058 ± 0,0053	0,7058 ± 0,0056
	3-3-VS	*Recall*	0,8251 ± 0,0005	0,7226 ± 0,0008	0,8703 ± 0,0006	0,7128 ± 0,0008	0,8441 ± 0,0004
		Prediction	0,8189 ± 0,0039	0,7128 ± 0,0061	0,8655 ± 0,0048	0,7032 ± 0,0072	0,8387 ± 0,0049
3. Stufe	8-8-3-VS	*Recall*	0,8360 ± 0,0004	0,7407 ± 0,0007	0,8613 ± 0,0039	0,7668 ± 0,0081	0,8491 ± 0,0043
		Prediction	0,8268 ± 0,0041	0,7263 ± 0,0064	0,8519 ± 0,0091	0,7558 ± 0,0078	0,8413 ± 0,0079
	8-8(π)-3-VS	*Recall*	0,8377 ± 0,0004	0,7437 ± 0,0008	0,8683 ± 0,0036	0,7726 ± 0,0083	0,8457 ± 0,0048
		Prediction	0,8284 ± 0,0040	0,7289 ± 0,0062	0,8608 ± 0,0063	0,7600 ± 0,0100	0,8365 ± 0,0069
	3-3-3-VS	*Recall*	0,8291 ± 0,0005	0,7298 ± 0,0008	0,8594 ± 0,0030	0,7557 ± 0,0065	0,8411 ± 0,0045
		Prediction	0,8228 ± 0,0040	0,7198 ± 0,0062	0,8541 ± 0,0074	0,7464 ± 0,0097	0,8354 ± 0,0086

Tabelle 9: *Verhalten der einzelnen Stufen der untersuchten Vorhersageschemata (abgekürzt VS) für die strenge Klasseneinteilung*

geschemas. Wie ein Vergleich der dritten Stufen der in Tabelle 8 dargestellten Vorhersageschemata zeigt, werden diese Unterschiede in der Vorhersage der drei Sekundärstrukturklassen durch das MLP weitestgehend beseitigt, sodass in dieser Stufe für alle drei Vorhersageschemata die einzelnen Klassen bezüglich der einzelnen q_σ im gleichen Verhältnis zueinander stehen: Die helikale Klasse wird am besten vorhergesagt, gefolgt von strukturlosen Regionen und schließlich den β-Strängen als der unsichersten Klasse. Dabei ist es interessant festzustellen, dass sowohl die helikale Klasse, wie auch die β-Strangklasse im Übergang von der zweiten zur dritten Stufe des 8-8-3- und des 8-8(π)-3-Vorhersageschemas Einbußen in q_H und q_E aufweisen, während q_C für die strukturlosen Klassen steigt. Das Ausmaß dieses Effekts ist beim 8-8(π)-3-Vorhersageschema geringfügig stärker ausgeprägt als beim 8-8-3-Vorhersageschema. Beim 3-3-3-Vorhersageschema hingegen findet eine Senkung von q_C bei gleichzeitiger Erhöhung von q_H und q_E statt, wodurch die Differenz zwischen q_H und q_C, gegenüber der zweiten Stufe vergrößert wird.

Ein ähnliches Verhalten weisen die drei Stufen der drei untersuchten Vorhersageschemata für die strenge Klasseneinteilung auf, was Tabelle 9 entnommen werden kann. Auch hier kann festgestellt werden, dass in den ersten beiden Stufen des 3-3-3-Vorhersageschemas die Vorhersage der einzelnen Klassen mit deren Häufigkeit in den Datensätzen korrelieren. Im Fall der strengen Klasseneinteilung bestehen die Daten zu 32,73% aus Helices, zu 21,06% aus β-Strängen und zu 46,21% aus strukturlosen Regionen, was sich gegenüber der lockeren Klasseneinteilung in stärker akzentuierten Unterschieden zwischen den einzelnen

q_σ der ersten und zweiten Stufe des 3-3-3-Vorhersageschemas äußert. In den ersten beiden Stufen des 8-8-3- und 8-8(π)-3-Vorhersageschemas hingegen führt die Verwendung der strengen Klasseneinteilung gegenüber der lockeren Klasseneinteilung zu einer Erhöhung aller q_σ. Die dritte Stufe gleicht auch hier die Verhältnisse in der Vorhersage der einzelnen Klassen bei allen drei Vorhersageschemata an. Beim 3-3-3-Vorhersageschema führt das zu einer Erhöhung von q_E und einer Senkung von q_H sowie q_C, während bei 8-8-3- und 8-8(π)-3-Vorhersageschema abermals q_H und q_E zu Gunsten von q_C gesenkt werden, was im Fall des 8-8(π)-3-Vorhersageschemas stärker ausgeprägt ist als beim 8-8-3-Vorhersageschema. Diese Anpassung der Werte der einzelnen q_σ führt wiederum für alle drei Vorhersageschemata zu einer Erhöhung von Q_3 und R_3 im Übergang zur dritten Stufe. Außer den beschriebenen Unregelmäßigkeit im Übergang zur dritten Stufe scheinen die Übergänge zu nächsthöheren Stufen hin die in Tabelle 9 dargestellten Qualitätsmaße, vor allem den Q_3-Index, zu verbessern.

Die bisherigen Ergebnisse scheinen darauf hinzudeuten, dass das 8-8(π)-3-Vorhersageschema für die Vorhersage beider Klasseneinteilungen die beste Wahl darstellt, da es für beide Klasseneinteilungen sowohl in *Prediction* als auch *Recall* die höchsten Werte von Q_3 und R_3 aufweist. Es stellt sich nun die Frage, ob dieses Vorhersageschema verbessert werden kann, beziehungsweise, ob ein neues, besseres Vorhersageschema gefunden werden kann, um das Dreiklassenproblem zu lösen. Dieser Frage soll im folgenden Abschnitt nachgegangen werden.

5.3.2.1. Einführung neuer Klasseneinteilungen für die zweite Stufe

Um zu untersuchen, inwiefern sich die Vorhersage des 8-8(π)-3-Vorhersageschemas verbessern lässt, sollte zunächst eine genauere Analyse der ersten Stufe dieses Vorhersageschemas durchgeführt werden. Ziel dieser Untersuchung war es zunächst, festzustellen, wie die Klassifikation in der ersten Stufe *SPARROWs im Detail vonstatten geht. Dazu wurden die Konfusionsmatrizen (siehe Abschnitt 4.3.4) für alle KV_α gebildet, jeweils für *Recall* und *Prediction*, um einen Überblick über die Verteilung der einzelnen Klassen durch die erste Stufe zu erhalten. Der Einfachheit halber wurden die Einträge dieser Konfusionsmatrizen in Prozentsätze umgerechnet, wobei die folgende Formel verwendet wurde:

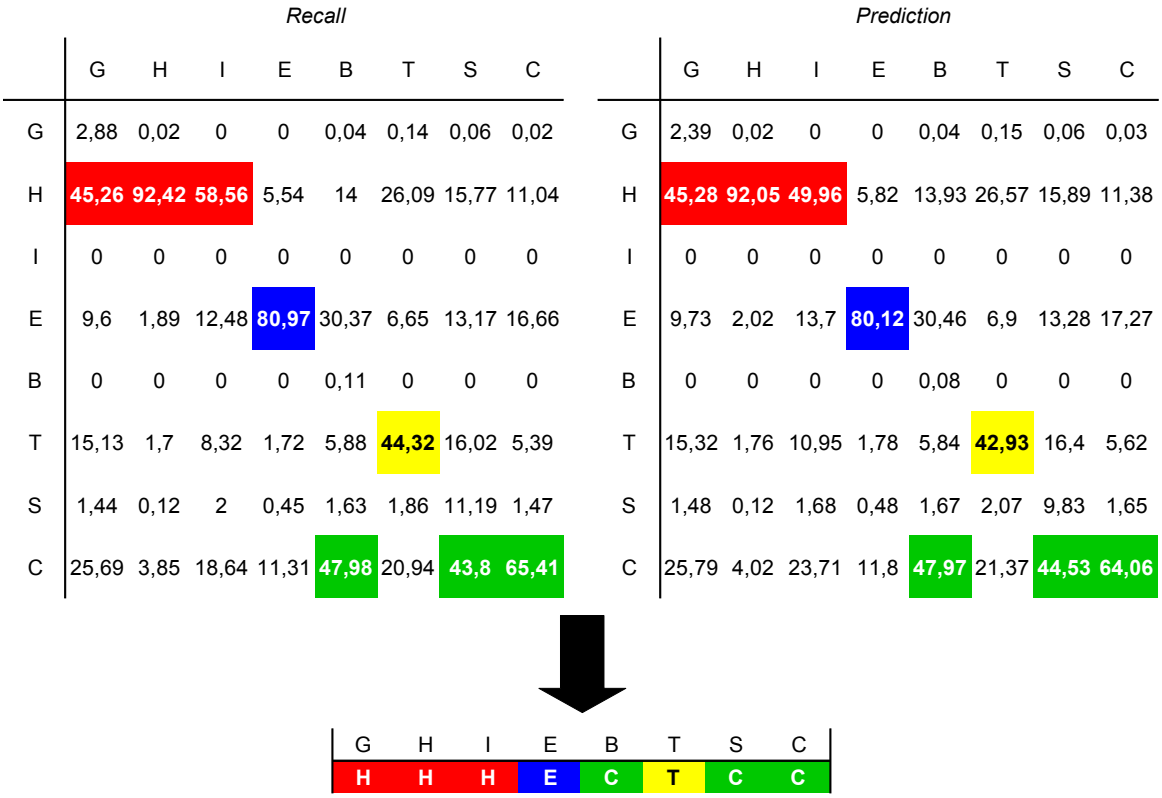


Abbildung 42: *Mittlere relative Konfusionsmatrizen für die erste Stufe des 8-8(π)-3-Vorhersageschemas*

$$\tilde{\omega}_{\sigma\rho} = 100 \cdot \frac{\omega_{\sigma\rho}}{\sum_{\sigma} \omega_{\sigma\rho}}$$

Auf diese Weise entspricht ein Eintrag $\tilde{\omega}_{\sigma\rho}$ der umgerechneten Konfusionsmatrix dem Prozentsatz an Residuen, die tatsächlich der Klasse ζ_ρ angehören, aber in der ersten Stufe fälschlich der Klasse ζ_σ zugewiesen wurden. Aus den so erhaltenen relativen Konfusionsmatrizen wurde jeweils eine mittlere relative Konfusionsmatrix aus den Daten aller D_α^{lern} und aller D_α^{test} erstellt. Hierbei flossen, wie auch schon bei den Mittelwerten in der Kreuzvalidierung, die Daten von KV_1 nicht in die Berechnung mit ein. Die sich aus diesen Berechnungen ergebenden mittleren relativen Konfusionsmatrizen sind in Abbildung 42 dargestellt. Im linken Teil der Abbildung ist die mittlere relative Konfusionsmatrix der D_α^{lern}, also des *Recalls*, dargestellt, während im rechten Teil die entsprechende Matrix der D_α^{test}, also der *Prediction,* abgebildet ist. Ein Vergleich der Zahlenwerte zeigt, dass beide Matrizen sehr ähnliche Einträge aufweisen, was bedeutet, dass der Klassifikator sowohl in *Re-*

call wie auch in der *Prediction* die Vertreter einer bestimmten Klasse in gleicher Weise auf die acht Klassen verteilt.

In Abbildung 42 ist in jeder Spalte beider Matrizen der jeweils höchste Wert farblich hervorgehoben, wobei für Maximalwerte, die sich in derselben Zeile befinden, die gleiche Farbe verwendet wurde. Dies hat folgenden Grund: Die Lage des Maximalwerts innerhalb einer Spalte zeigt, als was die meisten Vertreter der Klasse, die durch diese Spalte repräsentiert wird, klassifiziert werden. Fallen nun mehrere Maximalwerte in eine Zeile, so heißt das, dass durch den Klassifikator die betreffenden Klassen effektiv zusammenfallen. So geht aus Abbildung 42 hervor, dass α-Helices, 3_{10}-Helices und π-Helices vom Klassifikator hauptsächlich als α-Helices klassifiziert werden. Im Hinblick auf das Ziel, in der letzten Stufe von *SPARROW das Dreiklassenproblem zu lösen, können nun die drei helikalen Klassen für die zweite Stufe zu einer einzigen Klasse zusammengefasst werden. Da die zweite Stufe im 8-8(π)-3-Vorhersageschema prinzipiell bereits mit einer Zusammenführung dieser drei Klassen seitens der ersten Stufe arbeiten muss, ihr jedoch aufgezwungen wird, diese drei Klassen als separate Klassen zu behandeln, kann die Verwendung von acht Klassen in der zweiten Stufe zu einem suboptimalen Ergebnis führt. Aufgrund der in Abbildung 42 gezeigten Ergebnisse, kann deshalb eine Verwendung von weniger als acht Klassen in der zweiten Stufe möglicherweise ein besseres Ergebnis liefert.

Ein ähnlicher Effekt wie bei den helikalen Klassen zeigt sich im Fall von Biegungen, isolierten β-Brücken und strukturlosen Regionen, die vom Klassifikator alle der Klasse der strukturlosen Regionen zugeordnet werden. Wie im Fall der helikalen Klassen können auch diese drei Klassen zu einer gemeinsamen Klasse zusammengefasst werden, die in der zweiten Stufe die Klasse der strukturlosen Regionen bildet. Auf diese Weise ergeben sich für die zweite Stufe die helikale Klasse, bestehend aus α-Helices, 3_{10}-Helices und π-Helices, als erste Klasse und die Klasse der strukturlosen Regionen, bestehend aus Biegungen, isolierten β-Brücken und den ursprünglichen strukturlosen Regionen, als zweite Klasse. Daneben ergibt sich nach Abbildung 42 aus den β-Strängen und den Windungen jeweils eine eigenständige Klassen, sodass die zweite Stufe nach diesen Überlegungen vier Klassen besitzen sollte. Die entsprechende Klasseneinteilung ist in Abbildung 42 unten dargestellt.

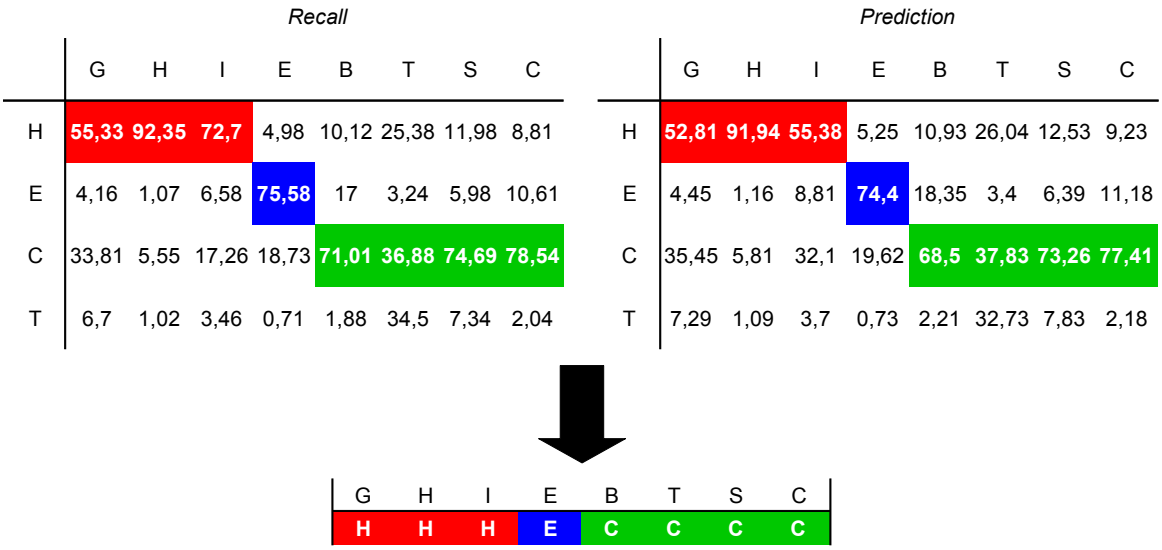


Abbildung 43: *Mittlere Verteilung der acht DSSP-Klassen auf die vier Klassen der zweiten Stufe*

Als nächstes sollte untersucht werden, wie sich die zweite Stufe nach einem Lernvorgang mit dieser neuen Klasseneinteilung verhält. Hierzu wurde eine mittlere Verteilung der acht ursprünglichen Klassen auf die vier Klassen, die diese Stufe unterscheidet, unter Verwendung von KV_2 bis KV_{10} berechnet. Das hierbei verwendete Verfahren entspricht demjenigen, das bereits für die Untersuchungen der ersten Stufe beschrieben wurde. Als Ergebnis dieser Berechnung ergeben sich in diesem Fall jedoch zwei 4×8 -Konfusionsmatrizen, deren Spalten den ursprünglichen acht Klassen entsprechen und deren Zeilen jeweils widerspiegeln, wie diese acht Klassen auf die vier Klassen in der zweiten Stufe verteilt werden. Diese Konfusionsmatrizen sind in Abbildung 43 dargestellt. Auch hier sind die Maximalwerte in jeder Spalte farbig hervorgehoben und die Maximalwerte innerhalb einer Zeile erhalten die gleiche Farbe. Dadurch fällt sofort auf, dass sich in der zweiten Stufe drei verschiedene Gruppierungen von Klassen ergeben, da die meisten Windungen in dieser Stufe nun der Klasse der strukturlosen Regionen zugewiesen werden, wodurch die Windung als eigenständige Klasse entfällt. Weiterhin zeigt sich, dass die drei helikalen Klassen, die in der ersten Stufe unterschieden werden, in der helikalen Klasse der zweiten Stufe noch stärker Verschmelzen. Dies gilt auch für die drei Klassen, die der strukturlose Klasse der zweiten Stufe zugewiesen sind. Schließlich zeigt ein Vergleich von Abbildung 42 und Abbildung 43, dass β-Stränge immer noch eine eigenständige Klasse bilden, jedoch auch, dass der Anteil der korrekten Vorhersagen dieser Klasse in Abbildung 43 geringer ist. Abbildung 43 legt nun nahe, dass in der dritten Stufe drei Klassen verwendet werden

sollten, die nach der Klasseneinteilung im unteren Teil dieser Abbildung gebildet werden. Die Klasseneinteilung weist jedoch das Problem auf, dass sie weder der strengen, noch der lockeren Klasseneinteilung entspricht, sondern eine neue Klasseneinteilung darstellt. Diese besitzt eine große Ähnlichkeit zur lockeren Klasseneinteilung und unterscheiden sich von dieser lediglich in der Zuordnung der isolierten β-Brücken. Diese gehören nach der lockeren Klasseneinteilung der Klasse der β-Stränge an, während sie im Fall der neuen Klasseneinteilung der Klasse der strukturlosen Regionen angehören. Aufgrund der großen Übereinstimmung mit der lockeren Klasseneinteilung soll diese neue Klasseneinteilung deshalb als *quasi-lockere Klasseneinteilung* bezeichnet werden. Die in Abbildung 42 eingeführte Klasseneinteinung soll wiederum, aufgrund ihrer Funktion als Übergang zwischen der vollständigen Klasseneinteilung und der quasi-lockeren Klasseneinteilung, als *intermediär*$_C$ bezeichnet werden. Dabei soll der Index *C* andeuten, dass die isolierten β-Brücken der Klasse der strukturlosen Regionen zugewiesen sind.

Es scheint, dass das in diesem Abschnitt präsentierte 8-4$_C$-3-Vorhersageschema, das in seiner zweiten Stufe die intermediär$_C$-Klasseneinteilung verwendet, sich vorzugsweise für die Vorhersage der quasi-lockeren Klasseneinteilung eignet. Da die zweite Stufe des Vorhersageschemas die acht ursprünglichen Klassen auf eine bestimmte Weise zusammenfasst, kann angenommen werden, dass eine dritte Stufe nicht viel an dieser Zusammenfassung wird ändern können. Insbesondere der Übergang von der intermediär$_C$-Klasseneinteilung zur strengen Klasseneinteilung erscheint aus diesem Grund nicht sinnvoll, weshalb von solchen Vorhersagen mittels des 8-4$_C$-3-Vorhersageschemas abgesehen wurde. Bei der lockeren Klasseneinteilung sieht dies jedoch etwas anders aus. Durch die Tatsache, dass die lockere und die quasi-lockere Klasseneinteilung sich nur rudimentär unterscheiden, und dass die isolierten β-Brücken eine recht kleine Klasse darstellen, wurde angenommen, dass mit dem 8-4$_C$-3-Vorhersageschema auch eine Vorhersage der lockeren Klasseneinteilung durchgeführt werden kann, wenn die isolierten β-Brücken für den Lernvorgang in der dritten Stufen aus der Klasse der strukturlosen Regionen entfernt und der Klasse der β-Stränge hinzugefügt werden.

Logischerweise ist davon auszugehen, dass ein solcher Ansatz suboptimal ist, jedoch durch die geringe Anzahl der isolierten β-Brücken Fehlklassifikationen der Vertreter dieser Klasse keinen großen Effekt auf die Gesamtvorhersage haben. Um jedoch auch ein sauberes Vorhersageschema zu präsentieren, in dem in der ersten Stufe acht Klassen und in der

Recall

	G	H	I	E	B	T	S	C
H	**55,96**	**92,5**	**73,37**	4,92	11,27	25,69	12,28	9,07
E	4,93	1,24	7	**77,54**	35,44	3,9	7,09	12,04
C	32,23	5,22	16,1	16,82	**50,99**	**35,45**	**73,07**	**76,76**
T	6,87	1,04	3,53	0,72	2,3	34,96	7,56	2,13

Prediction

	G	H	I	E	B	T	S	C
H	**53,34**	**92,06**	**57**	5,21	11,75	26,39	12,88	9,52
E	5,36	1,34	7,62	**76,37**	31,21	4,07	7,62	12,75
C	33,83	5,48	30,96	17,66	**54,58**	**36,4**	**71,47**	**75,44**
T	7,47	1,12	4,42	0,76	2,47	33,13	8,03	2,29

Abbildung 44: *Mittlere Verteilung der acht Klassen auf die vier Klassen in der zweiten Stufe unter Verwendung der intermediär$_E$-Klasseneinteilung*

zweiten vier Klassen verwendet werden, wurde zusätzlich die *intermediär*$_E$-Klasseneinteilung für die zweite Stufe aufgestellt. Diese entspricht der *intermediär*$_C$-Klasseneinteilung, jedoch mit dem Unterschied, dass die isolierten β-Brücken in diesem Fall als β-Stränge behandelt werden. Das daraus resultierende Vorhersageschema soll entsprechend als 8-4$_E$-3-Vorhersageschema bezeichnet werden. Mit diesem Vorhersageschema kann in der dritten Stufe nach der obigen Argumentation sowohl die lockere wie auch die quasi-lockere Klasseneinteilung in der dritten Stufe vorhergesagt werden, wobei in diesem Fall die Vorhersage der quasi-lockeren Klasseneinteilung als suboptimal angesehen werden kann. Eine Zusammenfassung der in diesem Abschnitt eingeführten Klasseneinteilungen ist in Tabelle 10 zusammen mit der lockeren Klasseneinteilung dargestellt.

	3_{10}-Helix	α-Helix	π-Helix	β-Strang	β-Brücke	Windung	Biegung	strukturlos
intermediär$_E$	H	H	H	E	E	T	C	C
locker	H	H	H	E	E	C	C	C
intermediär$_C$	H	H	H	E	C	T	C	C
quasi-locker	H	H	H	E	C	C	C	C

Tabelle 10: *Überblick über neue Klasseneinteilungen im Vergleich zur lockeren Klasseneinteilung*

Weiterhin zeigt Abbildung 44 die mittlere Verteilung der acht DSSP-Klassen auf die vier Klassen, welche die zweite Stufe des 8-4$_E$-3-Vorhersageschemas unterscheidet. Der Vergleich zwischen Abbildung 43 und Abbildung 44 zeigt, dass die Verwendung der intermediär$_E$-Klasseneinteilung gegenüber der intermediär$_C$-Klasseneinteilung keine größeren Veränderungen in der Verteilung der Klassen ergibt. Lediglich in der Verteilung der isolierten β-Brücken ergibt sich, wie zu erwarten war, eine Veränderung: Das 8-4$_E$-3-Vorhersageschema weist in der zweiten Stufe im *Recall* immer noch ca. 51% der β-Brücken, statt der 71% beim 8-4$_C$-3-Vorhersageschema, der Klasse der strukturlosen Regionen zu. Dafür

werden nun 35,44% als β-Stränge klassifiziert, gegenüber den 17% im 8-4_C-3-Vorhersageschema. Jedoch zeigt die Betrachtung der Maximalwerte in den einzelnen Spalten, dass auch in diesem Fall die Mehrheit der β-Brücken als strukturlos klassifiziert wird.

Es ist schwer vorherzusagen, welche Information das MLP jeweils aus den Daten der zweiten Stufen beider Vorhersageschemata extrahieren kann, da beide, was die Verteilung der acht ursprünglichen DSSP-Klassen angeht, nur marginale Unterschiede aufweisen. Die Ergebnisse, die im nächsten Abschnitt präsentiert werden, sollen nun zeigen, ob und wie sich diese Unterschiede auf die Vorhersagen der dritten Stufen beider Vorhersageschemata auswirken.

5.3.2.2. Zusammenfassung der Ergebnisse für das Dreiklassenproblem

Nachdem unterschiedliche Vorhersageschemata und Klasseneinteilungen für die Lösung des Dreiklassenproblems untersucht wurden, sollen die gewonnenen Ergebnisse in einem abschließenden Vergleich gegenübergestellt werden. Hierzu sollen für die lockere, die quasi-lockere und die strenge Klasseneinteilung diejenigen Vorhersageschemata betrachtet werden, die für die jeweilige Klasseneinteilung die beste Vorhersagequalität, also die höchsten Werte für Q_3 in der Kreuzvalidierung aufweisen. Dabei soll in diesem Abschnitt zudem ein Ausschnitt der Ergebnisse der Kreuzvalidierung für das 8-$4_C(\pi)$-3- und 8-$4_E(\pi)$-3-Vorhersageschema sowie die quasi-lockere Klasseneinteilung (siehe Abschnitt 5.3.2.1) beschrieben werden. Eine detaillierte Beschreibung und tabellarische Darstellung dieser Ergebnisse mit den neuen Vorhersageschemata und der neuen Klasseneinteilung ist in Anhang C präsentiert. Abbildung 45 zeigt für jede der drei Klasseneinteilungen die Werte der Q_3 -Indices der einzelnen Stufen der jeweils besten Vorhersageschemata in Form von Balkendiagrammen, jeweils für *Recall* und *Prediction*. Zusätzlich ist in jedem Diagramm der Q_3 -Index aller Stufen des 3-3-3-Vorhersageschemas, das der traditionellen Herangehensweise an das Problem der Sekundärstrukturvorhersage von Proteinen entspricht, dargestellt. Wie den einzelnen Diagrammen in Abbildung 45 jedoch entnommen werden kann, weist das 3-3-3-Vorhersageschema in der dritten Stufe, also in der endgültigen Vorhersage der Sekundärstruktur, den niedrigsten Wert für Q_3 in *Recall* und *Prediction* jeder Klasseneinteilung auf. Dies kann auch in den Tabellen 9 auf Seite 160, 16 auf Seite 237 und 17 auf Seite 239 festgestellt werden. Bei Betrachtung dieser Tabellen hinsichtlich der Standardabweichungen von Q_3 für *Recall* und *Prediction* zeigt sich zudem, dass der Unterschied in

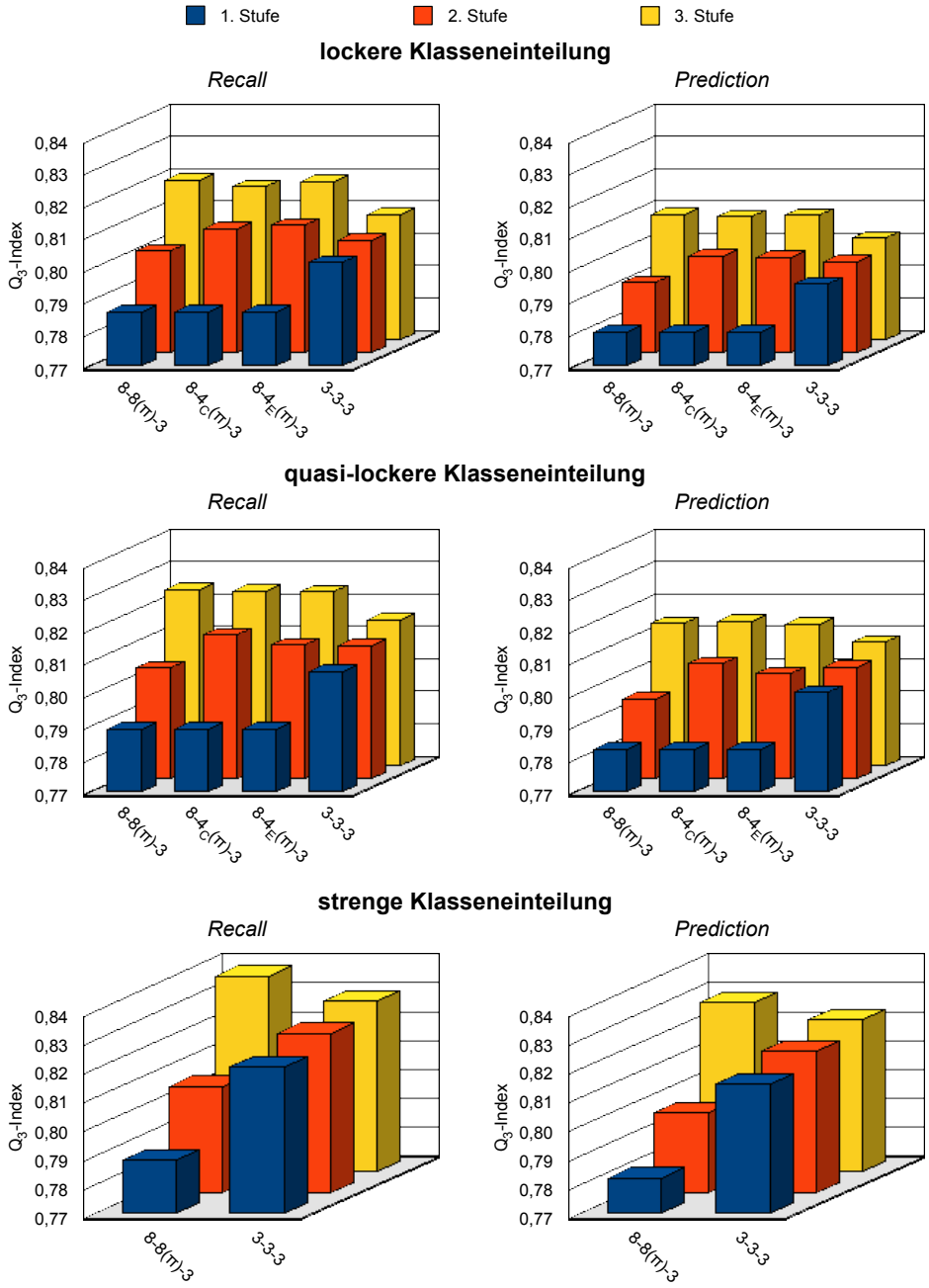


Abbildung 45: *Gegenüberstellung der besten Vorhersageschemata für die drei Klasseneinteilungen (locker, quasi-locker und streng) des Dreiklassenproblems*

Q_3 zwischen dem 3-3-3-Vorhersageschema und den übrigen Vorhersageschemata, die in Abbildung 45 aufgeführt sind, für jede Klasseneinteilung die jeweiligen Standardabweichungen übersteigt. Somit wird angenommen, dass das 3-3-3-Vorhersageschema den übrigen vorgestellten Vorhersageschemata, was den Wert von Q_3 angeht, unterlegen ist und die festgestellte Differenz sich nicht ausschließlich aus den aufgestellten Statistiken ergibt.

Nun stellt sich im Hinblick auf den Vergleich von *SPARROW mit anderen Methoden (siehe Abschnitt 5.4) die Frage, welches Vorhersageschema für *SPARROW verwendet werden sollte, wenn von einer der drei untersuchten Klasseneinteilungen ausgegangen wird. Üblicherweise würde sich für diese Entscheidung eine Betrachtung von Q_3 in der *Prediction* anbieten. Aus diesem Kriterium ergibt sich, dass das 8-8(π)-3-Vorhersageschema von den untersuchten Vorhersageschemata am besten dafür geeignet ist, die strenge Klasseneinteilung vorherzusagen. Bei den anderen beiden Klasseneinteilungen weisen die Ergebnisse keinen derart einfachen Trend auf. Wie aus Abbildung 45 hervorgeht und auch in den Tabellen 16 und 17 festgestellt werden kann, sind das 8-8(π)-3-, das 8-4_E(π)-3- und das 8-4_C(π)-3-Vorhersageschema äquivalent, zumindest was ihre Q_3-Indices in der *Prediction* angeht. In Tabelle 16 und 17 zeigen sich jedoch auch feine Unterschiede zwischen diesen drei Vorhersageschemata, wie beispielsweise die im Vorhersageverhalten der einzelnen Klassen in der lockeren Klasseneinteilung. Grob betrachtet sind sich die drei Vorhersageschemata also sehr ähnlich, besitzen jedoch im Detail einige Unterschiede, die sich in leicht abweichender Korrektheit der einzelnen Klassen äußert, was im Fall der lockeren Klasseneinteilung stärker ausgeprägt ist als für die quasi-lockere Klasseneinteilung. Damit besteht jedoch die auch Möglichkeit, dass die einzelnen Vorhersageschemata verschiedene Residuen auch unterschiedlich klassifizieren, wodurch sie trotz der hohen Ähnlichkeit, die sie im Mittel für große Datenmengen besitzen, bei der Vorhersage einzelner Proteine für einige Residuen teilweise sehr unterschiedliche Ergebnisse liefern würden. Die teilweise unterschiedlichen Werte der entsprechenden Standardabweichungen für diese Vorhersageschemata sprechen dafür. Aufgrund dieser möglichen Unterschiede würde es sich anbieten, für die Vorhersage der lockeren oder quasi-lockeren Klasseneinteilung die Vorhersagen des entsprechenden 8-8(π)-3-, 8-4_E(π)-3- und 8-4_C(π)-3-Vorhersageschemas in geeigneter Weise zu kombinieren, um eine weitere kleine Verbesserung der *Prediction* zu erreichen.

Eine weitere Beobachtung, die anhand von Abbildung 45 gemacht werden kann, ist der Trend im Wert von Q_3, der sich zwischen den einzelnen Klasseneinteilungen im *Recall* wie auch in der *Prediction* ergibt. So zeigt sich für das 3-3-3-, das 8-8(π)-3-, das 8-4_E(π)-3- und das 8-4_C(π)-3-Vorhersageschema, dass der Q_3-Index, der von jedem dieser Vorhersageschemata bei der lockeren Klasseneinteilung erreicht wird, unter dem Wert liegt, den das betreffende Vorhersageschema bei der quasi-lockeren Klasseneinteilung erreicht. Weiterhin weisen sowohl 3-3-3- als auch 8-8(π)-3-Vorhersageschema bei der strengen Klas-

seneinteilung höhere Werte von Q_3 auf, als dies bei der quasi-lockeren Klasseneinteilung der Fall ist. Die Unterschiede in den Q_3-Indices, die für ein bestimmtes Vorhersageschema zwischen den einzelnen Klasseneinteilungen in *Prediction* und *Recall* festgestellt werden können, übersteigen zudem die jeweilige Standardabweichung, wie Tabelle 9, 16 und 17 entnommen werden kann. Damit kann ein statistischer Effekt als Begründung für die festgestellten Unterschiede zwischen den Klasseneinteilungen ausgeschlossen.

5.4. Anwendung

Nach den in Abschnitt 5.3 beschriebenen Tests mit den unterschiedlichen möglichen Vorhersageschemata zur Umsetzung der endgültigen Version von *SPARROW, stellt sich nun die Frage, wie sich das daraus hervorgehende Vorhersageprogramm im Vergleich mit anderen Vorhersageprogrammen behaupten kann. Für die hierzu notwendigen Vergleichstests wurden lediglich Programme in Betracht gezogen, von denen eine Version in Form eines herunterladbaren Programms zur Verfügung steht. Zudem wurden die Versionen der einzelnen Programme so gewählt, dass das Datum der Veröffentlichung bei jedem Programm vor dem Veröffentlichungsdatum der Version 1.73 des ASTRAL40-Datensatzes lag. Der Grund hierfür ist, dass als Testdatensatz, auf dem der Vergleich zwischen den Programmen, wie in Abschnitt 5.1.2 beschrieben, durchgeführt werden sollte, Proteine verwendet werden, die im Übergang von Version 1.71 auf 1.73 des ASTRAL40-Datensatzes neu hinzukamen. Um einen fairen Vergleich auf diesen Daten ziehen zu können ist es jedoch wichtig, dass die Testdaten nach Möglichkeit nicht bereits in den Lerndaten verwendet wurden. Es wurde angenommen, dass dies bei Programmen, die vor der Erscheinung von Version 1.73 des ASTRAL40-Datensatzes veröffentlicht wurden, auszuschließen sei. Nach den beschriebenen Auswahlkriterien wurden die folgenden Programme für den Vergleich in Betracht gezogen:

- PROF v1.0 [32]
- Prospect 2 [69]
- PSIPRED v2.5 [30]
- SSpro 4.01 [29]
- SPARROW [1]

Neben den aufgezählten Programmen kamen auf Grundlage der beschriebenen Auswahlkriterien prinzipiell noch GOR-IV [61], PHD [27] und PREDATOR [70] in Frage. Diese wurden jedoch wegen der Vergleichsweise geringen Vorhersagequalität (siehe [1]) nicht in den Vergleich einbezogen. Die Ergebnisse für die einzelnen Programme, die mit *SPARROW verglichen werden sollten, entsprechen in allen Fällen bis auf SPARROW den bereits in [1] verwendeten Ergebnissen. Im Fall von SPARROW wurde das MLP für einige weitergehende Tests (siehe Abschnitt 5.4.3), sowie aufgrund der Tatsache, dass das in [1] verwendete MLP wegen einer Aktualisierung SPARROWs nicht mehr zur Verfügung stand, neu trainiert. Durch die leichten Abweichung in den Parametern des MLP, die sich daraus ergeben, zeigen sich gegenüber [1] einige geringfügige Unterschiede in den in dieser Arbeit präsentierten Ergebnissen für SPARROW. Der Lernvorgang mit SPARROW wurde, wie es auch bei *SPARROW der Fall ist, mit Hilfe des ASTRAL40-Datensatzes in der Version 1.71 durchgeführt, wodurch SPARROW und *SPARROW die einzigen Programme im Vergleichstest sind, bei denen sichergestellt ist, dass die Lerndaten identisch und von den Testdaten disjunkt sind.

Der Vergleich zwischen *SPARROW und den übrigen fünf Programmen soll einerseits anhand des Q_3-Index auf dem Testdatensatz gezogen werden, wobei Residuen gemäß der lockeren Klasseneinteilung klassifiziert werden. Grund hierfür ist die Tatsache, dass diese Klasseneinteilung bereits im EVA-Projekt zum Vergleich von PROF, Prospect, PSIPRED und SSpro verwendet wurde, sodass angenommen wird, dass die betreffenden Programme diese, oder aber eine sehr ähnliche, Klasseneinteilung gelernt haben, weshalb ein Vergleich auf dieser Klasseneinteilung berechtigt erscheint. Andererseits soll untersucht werden, wie der mittlere Q_3-Index pro Protein bei den einzelnen Programmen aussieht. Dieser berechnet sich als Mittelwert der Q_3-Indices der einzelnen Proteine, die sich wiederum aus die Teilung der Anzahl der korrekt klassifizierten Residuen innerhalb eines Proteins durch dessen Länge ergeben. Durch den Vergleich des mittleren Q_3-Index pro Protein soll die mittlere Korrektheit einer einzelnen Vorhersage, wie sie von einem Benutzer des jeweiligen Programms durchgeführt werden würde, abgeschätzt werden. Der auf Residuen basierende Q_3-Index spiegelt diesen Sachverhalt nur bedingt wieder, da er auf dem gesamten Datensatz beruht und eher das generelle Verhalten der einzelnen Programme auf größeren Datenmengen ohne den Kontext einzelner Proteine abschätzt. Um auf Grundlage des mittleren

Q_3-Index pro Protein einen detaillierteren Vergleich zu ermöglichen und so eventuelle Unterschiede zwischen den einzelnen Programmen besser hervorzuheben, muss jedoch zusätzlich auch die Länge der Proteine innerhalb des Testdatensatzes untersucht werden. Die Ergebnisse des durchgeführten Vergleichs sollen in Abschnitt 5.4.2 präsentiert werden. Vorher muss in Abschnitt 5.4.1 jedoch kurz auf die endgültige Umsetzung von *SPARROW, wie sie für den Vergleich verwendet wurde, eingegangen werden.

5.4.1. Die endgültige Umsetzung von *SPARROW

Mit dem Q_3 der *Prediction* als Auswahlkriterium ergibt sich aus Abschnitt 5.3.2.2, dass es drei mögliche Vorhersageschemata gibt, die für die lockere Klasseneinteilung für eine endgültige Umsetzung von *SPARROW in Frage kämen: Das 8-8(π)-3-, das 8-4_E(π)-3- und das 8-4_C(π)-3-Vorhersageschema. Diese drei Vorhersageschemata sollen nach der Argumentation in Abschnitt 5.3.2.2 für die endgültige Umsetzung von *SPARROW auf geeignete Weise kombiniert werden. Hierfür existiert eine Methode, die als *Ensemble Averaging* bekannt ist und ein gängiges Verfahren zur Kombination von MLPs darstellt, denen derselbe Lerndatensatz zugrunde liegt, die sich aber trotzdem in einigen Details, wie beispielsweise der Handhabung der Daten, unterscheiden [45]. Dies ist hier der Fall, da jedes der drei Vorhersageschemata eine eigene Art besitzt, die einzelnen Klassen zu handhaben. Eine Kombination der drei Vorhersageschemata könnte Nutzen aus diesen Unterschieden ziehen, sodass die Vorhersagequalität insgesamt gegenüber den einzelnen Vorhersageschemata verbessert werden würde. Da jedoch die Unterschiede zwischen den Vorhersageschemata relativ gering sind, kann es sein, dass auch der Gewinn, der aus einer solchen Kombination gezogen wird, sich in Grenzen hält.

Der vorherigen Argumentation entsprechend wurde für die Umsetzung der endgültigen Fassung von *SPARROW eine Kombination von 8-8(π)-3-, 8-4_C(π)-3- und 8-4_E(π)-3-Vorhersageschema mittels *Ensemble Averaging* gewählt. Jedes der drei Vorhersageschemata erzeugt für jedes Residuum eines Proteins mit Hilfe der dritten Stufe drei Werte, mit deren Hilfe die Klassifikation dieses Residuums von dem jeweiligen Vorhersageschema vorgenommen wird. Diese sollen als Vektoren $\vec{y}_1, \vec{y}_2, \vec{y}_3 \in \mathbb{R}^3$ bezeichnet werden, wobei $\vec{y}_1$ die Ausgabe des 8-8(π)-3-Vorhersageschemas, $\vec{y}_2$ die Ausgabe des 8-4_C(π)-3-Vorhersageschemas und $\vec{y}_3$ die Ausgabe des 8-4_E(π)-3-Vorhersageschema repräsentiert. Für die Kombination mittels *Ensemble Averaging* ist es notwendig, die Ausgangsvektoren aller

		Q_3	R_3	q_H	q_E	q_C
8-8(π)-3-VS	*Recall*	0,8170	0,7170	0,8678	0,7769	0,7938
	Prediction	0,8137	0,7114	0,8726	0,7652	0,7854
8-4_C(π)-3-VS	*Recall*	0,8163	0,7152	0,8657	0,7596	0,8031
	Prediction	0,8138	0,7112	0,8711	0,7501	0,7952
8-4_E(π)-3-VS	*Recall*	0,8181	0,7176	0,8631	0,7472	0,8164
	Prediction	0,8148	0,7123	0,8686	0,7329	0,8094

Tabelle 11: *Verhalten der drei besten Vorhersageschemata auf den Daten des Vergleichstests*

drei Vorhersageschemata zu summieren. Die eigentliche Klassifikation wird somit anhand eines Vektors $\vec{y} = \vec{y}_1 + \vec{y}_2 + \vec{y}_3$ vorgenommen.

Für die Vergleichstests wurden zunächst für alle drei Vorhersageschemata Lernvorgänge mit der Version 1.71 des ASTRAL40-Datensatzes durchgeführt. Um zu Überprüfen, inwiefern die Verwendung von *Ensemble Averaging* zu einer Verbesserung des Vorhersageverhaltens beitragen könnte, wurden für alle drei Vorhersageschemata Q_3, R_3, q_H, q_E und q_C sowohl für den Lerndatensatz wie auch für den im Vergleichstest zu verwendenden Testdatensatz gemessen. Diese Ergebnisse sind in Tabelle 11 für alle drei Vorhersageschemata gezeigt, wobei *Recall* in diesem Fall die Vorhersage des Lerndatensatzes, also ASTRAL40 Version 1.71, und *Prediction* die Vorhersage auf dem Testdatensatz der Vergleichstests beschreibt. Aus diesen Ergebnissen geht hervor, dass das 8-4_E(π)-3-Vorhersageschema Lern- und Testdatensatz am besten vorhersagt, jedoch sind die Unterschiede zwischen den einzelnen Vorhersageschemata erwartungsgemäß recht klein. Es lassen sich aber auch Tendenzen in den Vorhersagen der einzelnen Klassen erkennen, von denen eine Kombination mittels *Ensemble Averaging* profitieren könnte. So scheint der 8-4_E(π)-3-Vorhersageschema in diesem Fall strukturlose Regionen besonders gut vorherzusagen, während das 8-8(π)-3-Vorhersageschema zur Klassifikation von β-Strängen geeignet zu sein scheint. Inwiefern diese Unterschiede jedoch in der Kombination der Vorhersageschemata zu einer Verbesserung der Vorhersagequalität beitragen, soll der nächste Abschnitt zeigen, in dem *SPARROW, das durch ebendiese Kombination umgesetzt wurde, mit anderen Programmen zur Sekundärstrukturvorhersage verglichen wird.

	$Q_3^{(Res)}$	$Q_3^{(Prot)}$
PROF	0,7455	0,7418 ± 0,1130
Prospect	0,7737	0,7756 ± 0,0858
PSIPRED	0,8194	0,8189 ± 0,0807
SPARROW	0,8102	0,8111 ± 0,0791
*SPARROW	0,8153	0,8174 ± 0,0787
SSpro	0,7896	0,7864 ± 0,0951

Tabelle 12: *Zusammenfassung der Ergebnisse des Vergleichstests in Form des Q_3-Index über alle Residuen des Testdatensatzes ($Q_3^{(Res)}$) sowie des mittleren Q_3-Index pro Protein ($Q_3^{(Prot)}$)*

5.4.2. Ergebnisse des Vergleichstests

In diesem Abschnitt sollen die Ergebnisse des Vergleichs zwischen *SPARROW und den ausgewählten Verfahren beschrieben werden. Da der Vergleich ausschließlich auf dem Testdatensatz durchgeführt wird, enthält dieser Abschnitt auch nur die Ergebnisse, die die einzelnen Verfahren auf diesem Datensatz erreichen konnten. Von Vergleichen auf der Version 1.71 des ASTRAL40-Datensatzes wird abgesehen, da der Test herausfinden soll, wie sich die einzelnen Programme bei einer möglichst echten Vorhersage, die den Normalfall ihres Betriebs darstellt, relativ zueinander verhalten.

Die Ergebnisse der Vorhersage auf dem Testdatensatz für alle Verfahren, bestehend aus den (auf Residuen basierten) Q_3 -Indices und den mittleren Q_3 -Indices pro Protein für jedes Verfahren sind in Tabelle 12 zusammengefasst. Dabei zeigt die linke Spalte die als $Q_3^{(Res)}$ bezeichneten Q_3 -Indices, die unter Berücksichtigung aller Residuen des Testdatensatzes berechnet wurden. Die rechte Spalte hingegen beinhaltet die mittleren Q_3 -Indices pro Protein, aufgeführt als $Q_3^{(Prot)}$, die sich als Mittelwert der Q_3 -Indices der einzelnen Proteine im Testdatensatz ergeben. Die entsprechende Spalte enthält neben diesem Mittelwert auch die dazugehörige Standardabweichung. Die Ergebnisse in Tabelle 12 zeigen, dass PSIPRED von den aufgeführten Verfahren das beste Verhalten auf dem Testdatensatz zeigt, gefolgt von *SPARROW und schließlich SPARROW. Weiterhin fällt auf, dass die übrigen Verfahren, also PROF, Prospect und SSpro, einen vergleichsweise großen Abstand zu den besten drei Verfahren aufweisen. Die Betrachtung des Verhaltens von *SPARROW zeigt zudem, dass die Kombination der drei in Tabelle 11 aufgeführten Vorhersageschemata tatsächlich eine, wenn auch sher geringe, Verbesserung bringt.

Als nächstes stellt sich die Frage, inwiefern vom Verhalten für Testdatensatz auf das allgemeine Verhalten des jeweiligen Verfahrens geschlossen werden kann. Um dies genauer zu

untersuchen ist es hilfreich zu überprüfen, ob die einzelnen Verfahren eine Korrelation des Q_3 -Index mit der Sequenzidentität gegenüber dem Lerndatensatz aufweisen. Es kann angenommen werden, dass die Existenz eines klaren positiven Trends des Q_3 -Index von niedrigen Sequenzidentitäten hin zu hohen prinzipiell für die Überanpassung des jeweiligen Programms spräche. Je stärker diese Überanpassung ausgeprägt ist, desto stärker sollte dieser Trend ausgeprägt sein. Aus diesem Grund soll das Verhalten von *SPARROW in Abhängigkeit von der Sequenzidentität untersucht und mit den übrigen Programmen in Beziehung gebracht werden. Dabei soll die von BLAST für jedes Protein des Testdatensatzes gelieferte maximale Sequenzidentität zu den Proteinen des Lerndatensatzes verwendet werden, die für den Testdatensatz alle Werte zwischen 16% und 40% annimmt. Entsprechend ihrer Sequenzidentität zum Lerndatensatz können die Proteine des Testdatensatzes gruppiert und mit Hilfe der Residuen dieser Teilmengen jeweils ein Q_3 -Index für jeden Wert der Sequenzidentität berechnet werden. Zudem kann auch ein Q_3 -Index für alle Residuen berechnet werden, die zu Proteinen gehören, deren Sequenzidentität unterhalb eines bestimmten Wertes liegt oder diesem gleicht. Zur Untersuchung der Abhängigkeit des Q_3 -Index von der Sequenzidentität sollen diese beiden Größen verwendet werden, die in Abbildung 46 auf der folgenden Seite dargestellt sind. Die oberen beiden Diagramme repräsentieren den Q_3 -Index für die einzelnen Werte der Sequenzidentität, wobei das linke Diagramm die Anzahl an Residuen für jeden Wert der Sequenzidentität zeigt und das rechte Diagramm die dazugehörigen Werte des Q_3 -Index darstellt. Aus den Graphen der einzelnen Programme geht kein eindeutiger Zusammenhang zwischen Q_3 -Index und Sequenzidentität hervor. Zwar scheint es, dass fast alle Programme für kleine Werte der Sequenzidentität vergleichsweise hohe Q_3 -Indices erreichen können, jedoch scheint sich dies als statistische Fluktuation zu ergeben, die aus der geringen Anzahl an Residuen resultiert, die zu Proteinen mit derart niedriger Sequenzidentität gehören. Was dennoch aus den Graphen hervorgeht ist, dass sich die einzelnen Programme für fast alle Werte der Sequenzidentität so verhalten, wie es auch in Tabelle 12 zu beobachten ist, was zeigt, dass die Qualität der Vorhersage im wesentlichen von der Sequenzidentität unabhängig ist. Zudem besitzen die Graphen der einzelnen Programme, mit Ausnahme von SSpro, qualitativ einen sehr ähnlichen Verlauf. Die gemachten Beobachtungen bestätigen sich auch bei Betrachtung der integrierten Größe in Abbildung 46 unten.

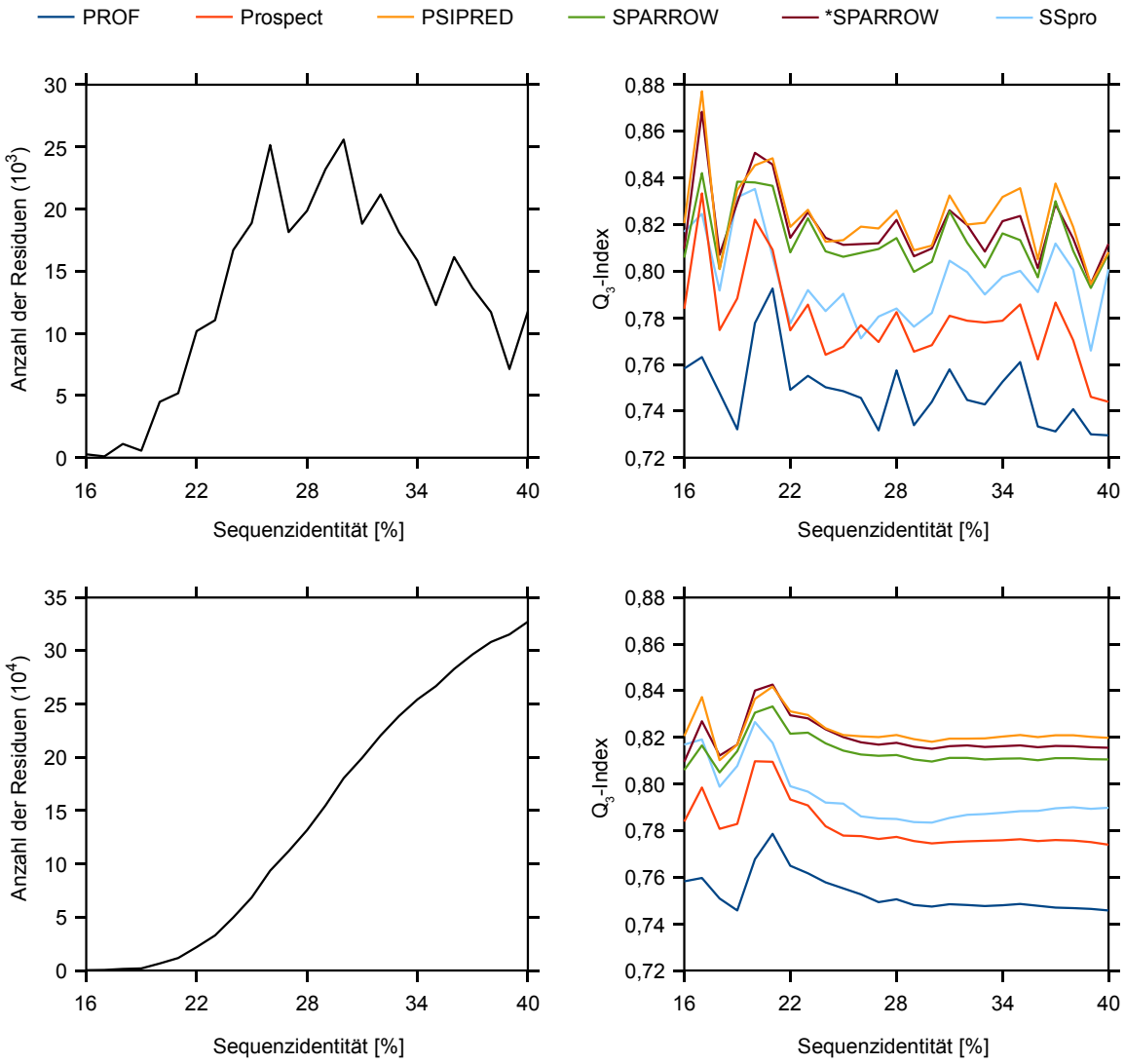


Abbildung 46: *Abhängigkeit des Q_3-Index von der Sequenzidentität gegenüber dem Lerndatensatz für die einzelnen Werte (oben) wie auch für Werte bis zu einem bestimmten Wert (unten)*

Neben der Sequenzidentität liefert BLAST ein weiteres, detaillierteres Maß zur Beschreibung der Ähnlichkeit zweier Sequenzen, nämlich den BLAST Score. Dieser Wert wird vom BLAST-Algorithmus jedem gefundenen Alignment zugeordnet, wobei der BLAST Score mit der Güte des Alignments zunimmt. Je höher der Wert des BLAST Scores ist, desto wahrscheinlicher sind beide Sequenzen homolog und desto unwahrscheinlicher ist es, dass ein Alignment mit einem höheren BLAST Score zufällig gefunden werden kann. Mit Hilfe dieses zweiten Ähnlichkeitsmaßes wurde versucht das Verhalten der sechs Programme etwas detaillierter zu untersuchen. Dies erwies sich jedoch schwieriger als anfangs erwartet, da die BLAST Scores sehr ungleichmäßig über den Testdatensatz verteilt sind.

Aufgrund der Tatsache, dass BLAST den BLAST Score bis auf eine Nachkommastelle genau ausgibt, ist dieser gewissermaßen diskret. Auf dem Testdatensatz ergeben sich 275 verschiedene Werte des BLAST Scores, die aus dem Intervall zwischen 21,9 und 270 stammen. Die Schwierigkeit liegt darin, dass ca. 75% aller Residuen des Testdatensatzes zu Proteinen mit einem BLAST Score unterhalb von 70 gehören. Durch diese Verteilung ist es schwierig, eine geeignete graphische Darstellung der Abhängigkeit des Q_3-Index vom BLAST Score zu finden. Für die verwendete graphische Darstellung wurden für die BLAST Scores 50 Intervalle zwischen 20 und 270 gebildet, wobei jedes Intervall eine Breite von 5 besaß. Die Proteine des Testdatensatzes wurden ihrem BLAST Score entsprechend gemäß dieser Intervalle gruppiert und für jedes Intervall wurde ein Q_3-Index unter Berücksichtigung aller Residuen der ihm zugeordneten Proteine berechnet. Die so entstandene graphische Darstellung ist in Abbildung 47 auf der folgenden Seite zu sehen, wobei die oberen Diagramme die Population (links) und den Q_3-Index (rechts) für die einzelnen Intervalle des BLAST Scores widerspiegeln, während die unteren Diagramme die Integrale der oberen Diagramme darstellen. Dabei ist links die Anzahl der Residuen aufgetragen, die Proteinen mit BLAST Scores unterhalb eines bestimmten Wertes angehören, und rechts die dazugehörigen Q_3-Indices. Die Betrachtung der oberen Graphen zeigt die Schwierigkeit dieser Darstellung: Die teilweise starke Überlagerung der einzelnen Graphen erschwert es insbesondere für die höheren Werte der BLAST Scores einen Vergleich zwischen den einzelnen Programmen zu ziehen. Es gibt zudem wenige Proteine mit hohen BLAST Scores, weshalb die Graphen für die höheren Werte des BLAST Scores, also Werte über 100, nicht verlässlich sind. Für BLAST Scores unterhalb von 100 zeigen die einzelnen Programme jedoch ein qualitativ ähnliches Verhalten in Abhängigkeit vom BLAST Score. Dabei lässt sich für BLAST Scores zwischen 20 und 50 ein positiver Trend des Q_3-Index erkennen, der bei allen Programmen mehr oder weniger gleich stark ausgeprägt ist. Eine Betrachtung der unteren Graphen zeigt, dass die Rangordnung der Güte der Vorhersageprogramme, die Anhand des Q_3-Index in Tabelle 12 aufgestellt wurde, von den Programmen bereits bei der ausschließlichen Berücksichtigung von Proteinen mit kleinen BLAST Scores angenommen wird und sich durch hinzunehmen von Proteinen mit großen BLAST Scores nicht mehr ändert. Dies könnte zwar der hohen Population der Proteinsequenzen mit geringem BLAST Scores zugeschrieben werden, jedoch stellt sich die Rangordnung bereits unter Berücksichtigung von circa einem Zehntel der Testdaten eindeutig erkennbar ein.

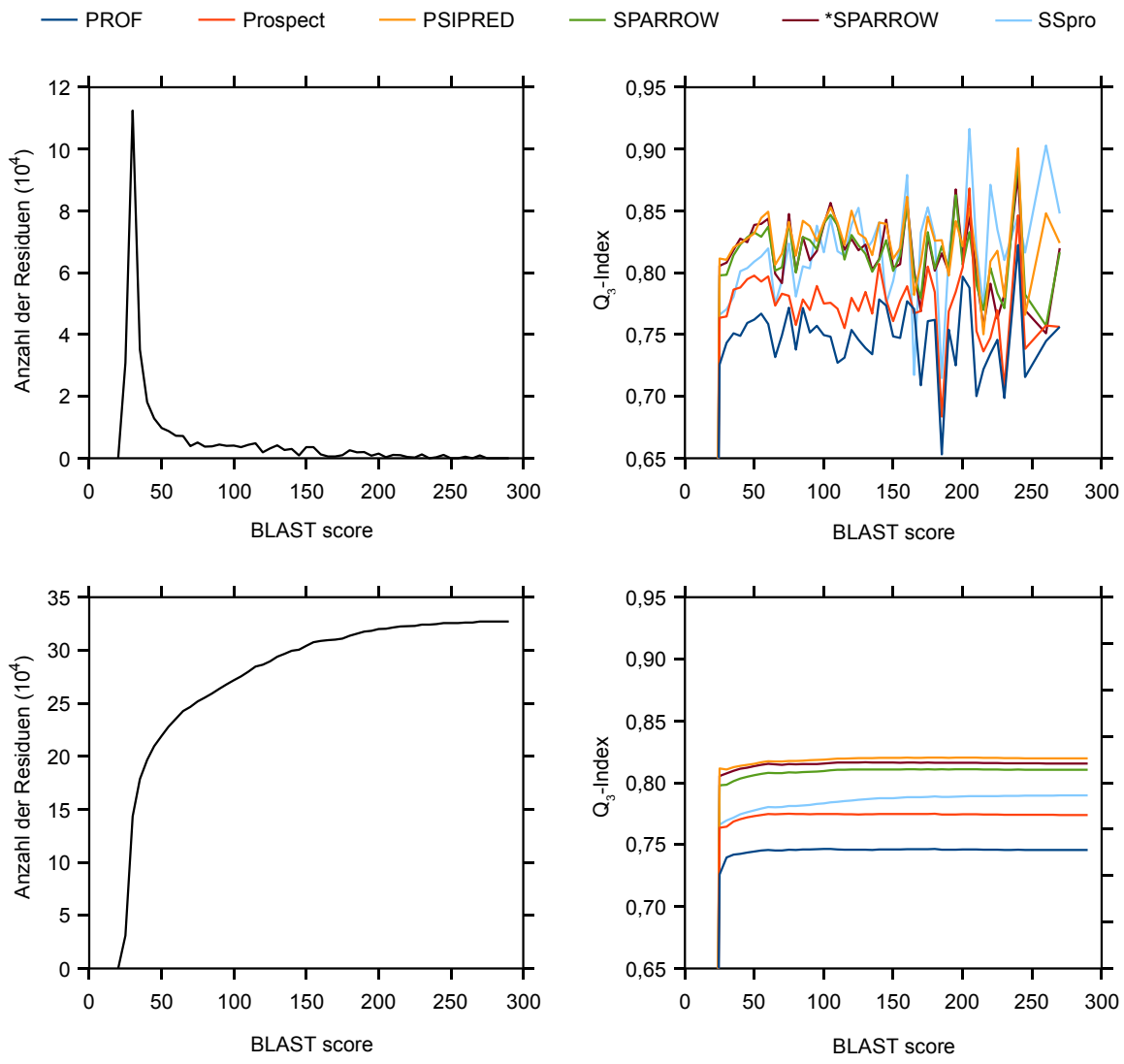


Abbildung 47: *Abhängigkeit des Q_3-Index vom BLAST Score für bestimmte Intervalle (oben) wie auch für Werte bis zu einem bestimmten Wert (unten)*

Als nächstes soll eine genauere Untersuchung des mittleren Q_3-Index pro Protein für *SPARROW wie auch für die anderen Programme vorgenommen werden. Zwar zeigt laut Tabelle 12 die Sortierung der untersuchten Programme nach $Q_3^{(Prot)}$ keinen Unterschied zur Sortierung bezüglich $Q_3^{(Res)}$, jedoch erlaubt $Q_3^{(Prot)}$ die Untersuchung, inwiefern es Unterschiede in der Vorhersagequalität der einzelnen Programme in Abhängigkeit von der Proteinlänge gibt. Für diese Untersuchung war es nötig, die Proteine des Testdatensatzes entsprechend ihrer Länge zu gruppieren, die von 21 als kleinstem Wert bis zu einem Maximum von 1146 Residuen variiert. Zur Unterteilung dieses Intervalls in Gruppen, wurden zunächst alle Proteine bis zu einer maximalen Länge von 40 Residuen zu einer Gruppe zu-

sammengefasst. Die verbliebenen Proteine wurden ihrer Länge entsprechend in Schritten von 20 Residuen zusammengefasst, sodass die zweite Gruppe Proteine mit einer Länge zwischen 40 und 60 Residuen enthält, die nächste zwischen 60 und 80 und so weiter. Dieses Schema wurde bis zu einer Länge von 480 Residuen fortgesetzt. Die verbliebenen Proteine, also alle mit einer Länge über 480 Residuen, bilden die letzte Gruppe. Auf diese Weise entstanden 24 Gruppen von Proteinen, für die jeweils ein separater mittlerer Q_3 -Index pro Protein für jedes Programm berechnet werden konnte, der graphisch aufgetragen werden sollte. Zusätzlich wurde auch ein mittlerer Q_3 -Index pro Protein für alle Proteine unterhalb einer bestimmten Länge unter Verwendung der obigen Gruppierung berechnet, sodass auch überprüft werden konnte, inwiefern ein bestimmtes Programm sich im Mittel besser zur Vorhersage von Proteinen bis zu einer bestimmten Maximallänge eignet. Beide Größen sind zusammen mit der Population der einzelnen Gruppierungen in Abbildung 48 auf der folgenden Seite dargestellt, wobei die oberen Diagramme für die Intervalle der Proteinlängen stehen und die unteren Diagramme das Verhalten der Programme für Proteine unterhalb einer bestimmten Länge zeigen. Aus den oberen Diagrammen geht hervor, dass *SPARROW bei Proteinen mit einer Länge zwischen 40 und 60, sowie bei Längen zwischen 60 und 80 von den untersuchten Programmen das beste Verhalten an den Tag legt, dabei jedoch relativ nahe bei PSIPRED und SPARROW liegt. Insbesondere bei der Gruppe 40-60 könnte dies sich jedoch aufgrund der relativ geringen Zahl an Proteinen innerhalb dieser Gruppe auch zufällig ergeben. Weiterhin zeigt SSpro für Proteine mit einer Länge von 460-480 das beste Verhalten, jedoch ist in diesem Fall die Anzahl der Proteine in dieser Gruppe derart gering, dass dieser Effekt eher zufällig ist. Ansonsten zeigen die sechs Verfahren qualitativ einen sehr ähnlichen Verlauf über die Proteinlängenitervalle. Eine Betrachtung der unteren Diagramme zeigt weiterhin, dass *SPARROW für Proteine bis zu einer Länge von 160 im Mittel das beste Verhalten zeigt. Insgesamt sind mehr als 50% der Proteine im Testdatensatz (genauer gesagt 1098 von 1919 Proteinen mit insgesamt 124300 Residuen) kürzer als 160 Residuen, weshalb diesem Ergebnis Aussagekraft beigemessen wird. Mit steigender Proteinlänge wird das Verhalten von PSIPRED jedoch besser als das von *SPARROW, was auch mit der Aussage der oberen Diagramme übereinstimmt und PSIPRED im Endeffekt den besten Wert des mittleren Q_3 -Index pro Protein in Tabelle 12 verleiht. Ein ähnlicher Effekt zeigt sich bei SSpro und Prospect: Während Prospect für kleine Proteine besser geeignet zu sein scheint, ist bei Einbeziehung des ge-

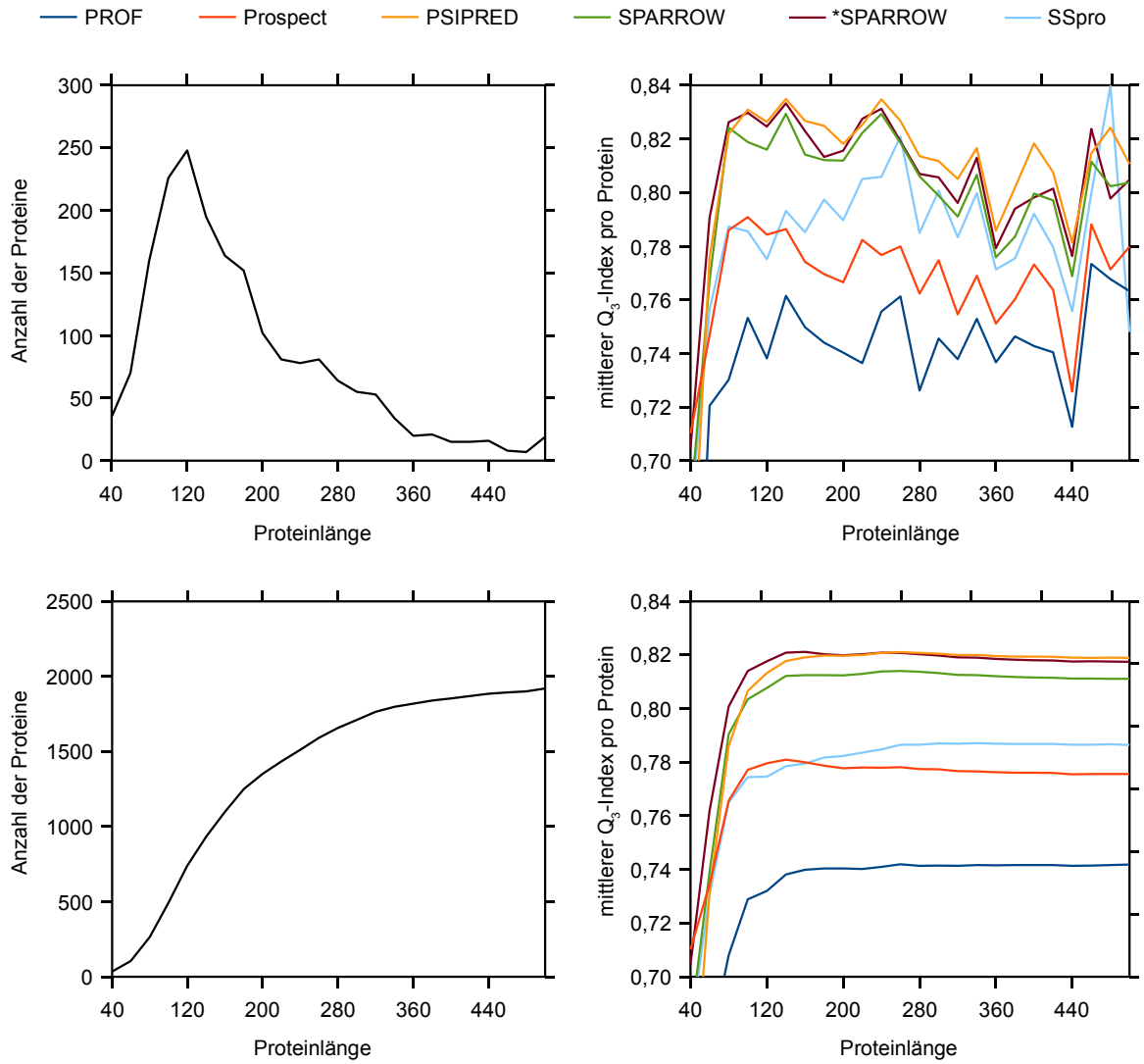


Abbildung 48: *Mittlerer Q_3-Index pro Protein in Abhängigkeit von der Proteinlänge für bestimmte Intervalle (oben) und bis zu einer Maximalgröße (unten)*

samten Testdatensatzes SSpro das bessere von beiden. Bis auf diese zwei Fälle ist die Rangodnung der Programme für Proteine bis zu einer bestimmten Größe jedoch invariant gegenüber Tabelle 12.

Als abschließende Untersuchung wurde die Verteilung der Q_3-Indices pro Protein, also der Richtigkeit einzelner Vorhersagen für die sechs Programme, betrachtet. Hierzu wurde gezählt, wie oft das jeweilige Programm unter Betrachtung aller Proteine des Testdatensatzes einen proteinbasierten Q_3-Index aufweist, der aus einem von hundert nicht überlappenden Intervallen zwischen 0 und 1 (mit einer Breite von 0,01) stammt. Auf diese Weise konnte für jedes Programm ein Histogramm der Q_3-Indices pro Protein erzeugt werden,

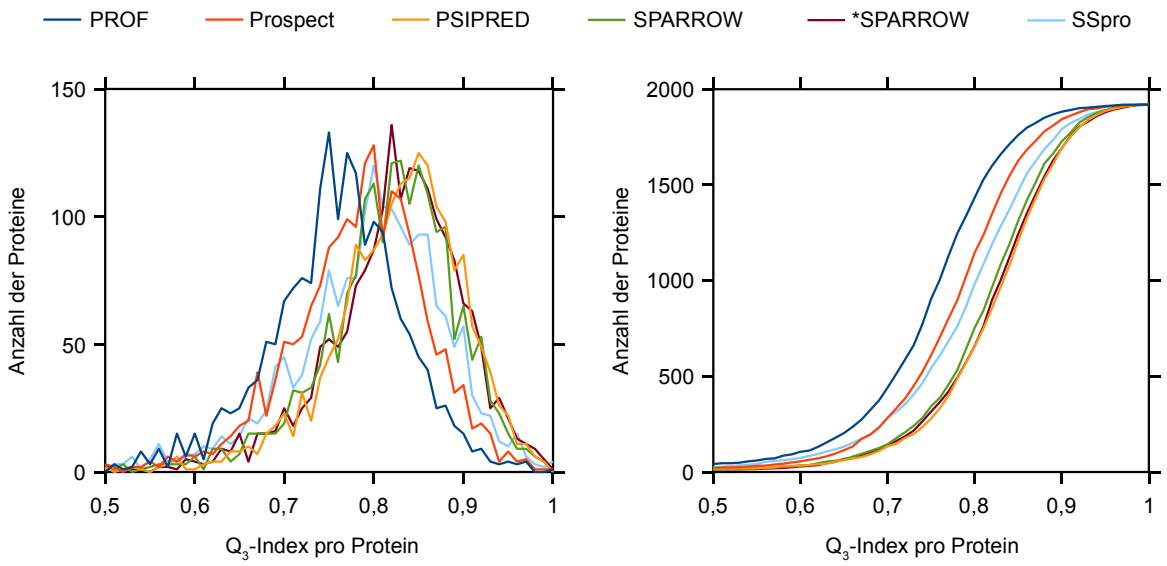


Abbildung 49: *Histogramm der Q_3-Indices pro Protein (links) und dessen Akkumulation (rechts)*

anhand dessen sich das Verhalten der einzelnen Programme abermals vergleichen lässt. Zusätzlich zu diesem Histogramm wurde auch dessen Akkumulation erzeugt, die zeigt, wie oft über den gesamten Datensatz das jeweilige Programm einen proteinbasierten Q_3-Index unterhalb eines bestimmten Wertes aufwies. Beide Größen wurden in Abhängigkeit vom Q_3-Index in Abbildung 49 graphisch dargestellt, wobei die Histogramme der einzelnen Programme sich im linken Diagramm befinden und die Akkumulationen im rechten Diagramm enthalten sind. Für die graphische Darstellung wurde der Bereich zwischen 0,5 und 1 gewählt, da für alle Programme Proteine mit Q_3-Indices unterhalb von 0,5 sehr selten sind, wie aus dem rechten Diagramm in Abbildung 49 hervorgeht. Auf den ersten Blick weisen die Histogramme im linken Diagramm in ihrem Verlauf eine Ähnlichkeit zur Dichtefunktion der Binomialverteilung auf. Es zeigt sich ein mehr oder weniger symmetrischer Verlauf der Graphen um den jeweiligen Mittelwert und auch die Standardabweichungen sind erkennbar. Dabei fällt insbesondere auf, dass die Verläufe der Graphen für SPARROW, *SPARROW und PSIPRED sich von den übrigen Graphen abheben. Diese drei Graphen liegen sehr nahe beieinander, während die Graphen der anderen Verfahren unterschiedlich stark zu den niedrigeren Werten des Q_3-Index hin verschoben sind. Diese Tendenz ist anhand der Graphen im rechten Diagramm, die der Verteilungsfunktion der Binomialverteilung sehr stark ähneln, besser erkennbar. Weiterhin zeigt das rechte Diagramm auch eine sehr starke Ähnlichkeit in den Verläufen der Kurven von PSIPRED und

*SPARROW. Dabei ist anzumerken, dass diesem Diagramm entsprechend ein Programm bei einzelnen Vorhersagen umso besser sein sollte, je später der entsprechende Graph anfängt zu steigen.

5.4.3. Kombinierte Verfahren

In diesem Abschnitt soll untersucht werden, inwiefern mittels einer Kombination von *SPARROW mit PSIPRED beziehungsweise mit SPARROW ein verbessertes Verfahren zur Vorhersage der Sekundärstruktur von Proteinen erzeugt werden kann. Zur Überprüfung, ob eine Kombination zweier Vorhersageprogramme gerechtfertigt ist, ist es zunächst von Vorteil zu untersuchen, inwiefern die Mengen der falsch vorhersagten Residuen dieser Programme übereinstimmen. Je stärker diese Mengen sich unterscheiden, umso größer sollte potentiell die aus einer Kombination resultierende Verbesserung sein. Eine Abschätzung auf dem Lerndatensatz zeigt, dass SPARROW und *SPARROW in ca. 80% ihrer falschen Vorhersagen übereinstimmen, während diese Übereinstimmung bei PSIPRED und *SPARROW bei ca. 76% liegt. Dies bedeutet, dass auf den übrigen 20% bei einer Kombination von SPARROW und *SPARROW, beziehungsweise 24% bei einer Kombination von PSIPRED und *SPARROW, potentiell mit einer Verbesserung der Vorhersage gerechnet werden kann.

Die Erfahrung zeigt, dass SPARROW bei der Verwendung gleicher Daten in der Regel mehr Überanpassung aufweist als *SPARROW. Aus diesem Grund wurde beschlossen, dass *SPARROW in einer Kombination mit SPARROW mittels *Ensemble Averaging* höher gewichtet werden sollte als SPARROW, um dieses erhöhte Maß an Überanpassung zu kompensieren. Daraus folgend wird der Ausgabevektor der Kombination von SPARROW und *SPARROW (kurz SPARROW + *SPARROW) nach der Formel $\vec{y}=\vec{y}_1+\vec{y}_2+\vec{y}_3+\vec{y}_S$ berechnet, wobei $\vec{y}_1$, $\vec{y}_2$ und $\vec{y}_3$ die Ausgabevektoren der in Tabelle 11 dargestellten Vorhersageschemata bilden, deren Kombination mittels *Ensemble Averaging* den Ausgabevektor *SPARROWs ergibt. $\vec{y}_S$ stellt die Ausgabe von SPARROW dar. Aufgrund der Tatsache, dass die einzelnen Komponenten von $\vec{y}_1$, $\vec{y}_2$, $\vec{y}_3$ und $\vec{y}_S$ demselben Wertebereich entstammen, besitzt *SPARROW einen dreimal so großen Einfluss auf SPARROW + *SPARROW wie SPARROW. Für die Kombination von PSIPRED und *SPARROW (kurz PSIPRED + *SPARROW) hingegen wurde angenommen, dass beide Programme denselben Einfluss auf die Kombination haben sollten. Deshalb wird der Ausgabevektor

	$Q_3^{(Res)}$	$Q_3^{(Prot)}$
PSIPRED	0,8194	0,8189 ± 0,0807
SPARROW	0,8102	0,8111 ± 0,0791
*SPARROW	0,8153	0,8174 ± 0,0787
PSIPRED + *SPARROW	0,8254	0,8256 ± 0,0779
SPARROW + *SPARROW	0,8180	0,8196 ± 0,0783

Tabelle 13: *Ergebnisse der Kombinationen von *SPARROW mit PSIPRED (PSIPRED + *SPARROW in der Tabelle) beziehungsweise SPARROW (SPARROW + *SPARROW in der Tabelle) in Form des Q_3-Index über alle Residuen des Testdatensatzes ($Q_3^{(Res)}$) und des mittleren Q_3-Index pro Protein ($Q_3^{(Prot)}$) im Vergleich zu den ursprünglichen Programmen*

dieser Kombination nach der Formel $\vec{y} = \frac{\vec{y}_1 + \vec{y}_2 + \vec{y}_3}{3} + \vec{y}_P$ berechnet. Dabei ist $\vec{y}_P$ der Ausgabevektor von PSIPRED, der derart angepasst wurde, dass er hinsichtlich Wertebereich und der Zuordnung seiner Komponenten zu den einzelnen Klassen mit den Vektoren $\vec{y}_1$, $\vec{y}_2$ und $\vec{y}_3$ kompatibel ist. Die Ergebnisse, die mit diesen beiden Kombinationen erreicht wurden, sind in Tabelle 13 dargestellt. Zusätzlich sind zum Vergleich der Kombinationen mit den ursprünglichen Programmen die Ergebnisse für PSIPRED, SPARROW und *SPARROW aus Tabelle 12 gezeigt.

Aus der Tabelle geht zunächst hervor, dass beide Kombinationen gegenüber den jeweiligen ursprünglichen Programmen eine Verbesserung in $Q_3^{(Res)}$ und $Q_3^{(Pred)}$ darstellen. Diese Verbesserung gegenüber den ursprünglichen Programmen ist bei PSIPRED + *SPARROW größer als bei SPARROW + *SPARROW. Weiterhin zeigt sich, dass in diesem Vergleich PSIPRED gegenüber SPARROW + *SPARROW einen höheren Wert von $Q_3^{(Res)}$ aufweist, SPARROW + *SPARROW jedoch den höheren Wert für $Q_3^{(Prot)}$ besitzt. Der Unterschied zwischen SPARROW + *SPARROW und PSIPRED ist jedoch für $Q_3^{(Res)}$ wie auch für $Q_3^{(Prot)}$ gering, sodass nicht sicher ist inwiefern dieser Unterschied noch signifikant ist. Die höchsten Werte für $Q_3^{(Res)}$ und $Q_3^{(Prot)}$ weist, wie aus Tabelle 13 hervorgeht, PSIPRED + *SPARROW auf, das für beide Werte einen großen Abstand zu den übrigen Verfahren aufweist.

Zur genaueren Untersuchung ist in Abbildung 50 auf der folgenden Seite der residuenbasierte Q_3-Index über die Sequenzidentität aufgetragen, wobei links der Q_3-Index für die einzelnen Werte der Sequenzidentität dargestellt ist und rechts das Integral dieser Größe über ein Intervall. Für beide Kombinationen kann festgestellt werden, dass die jeweilige Kurve in beiden Diagrammen stets über den Kurven der Programme liegt, die diese

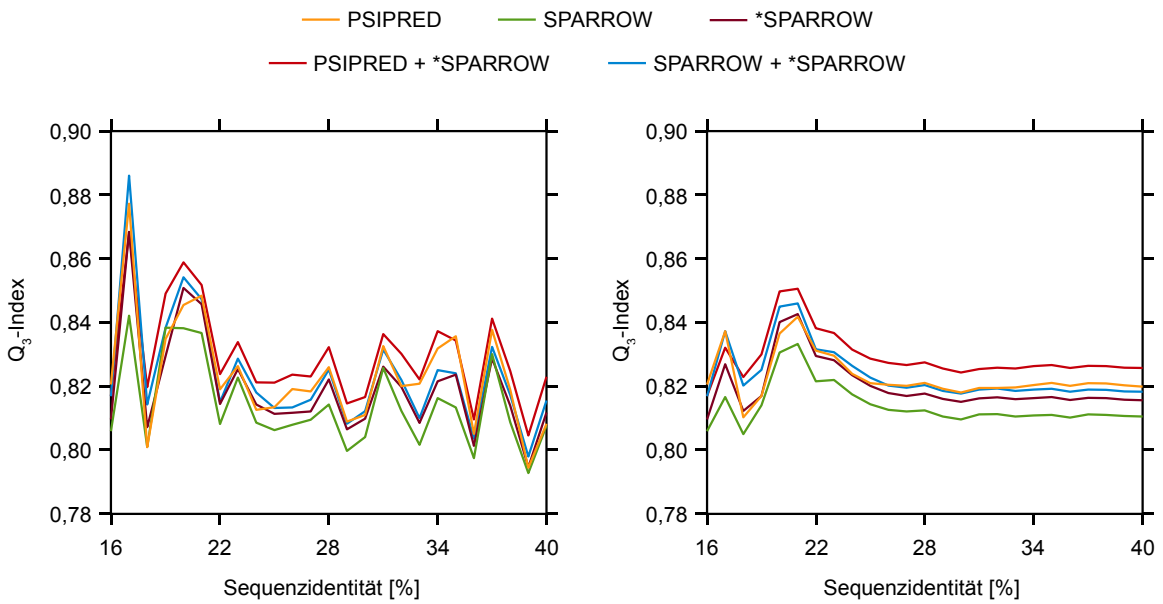


Abbildung 50: *Abhängigkeit des Q_3-Index von der Sequenzidentität gegenüber dem Lerndatensatz für bestimmte Werte (links) und bis zu einem bestimmten Maximalwert (rechts) für die Kombinationen von *SPARROW mit PSIPRED (PSIPRED + *SPARROW) beziehungsweise SPARROW (SPARROW + *SPARROW) im Vergleich zu den ursprünglichen Programmen*

Kombination bilden. Lediglich im linken Diagramm scheinen die Graphen von *SPARROW und SPARROW + *SPARROW in einigen Fällen zu überlappen, jedoch liegt diese Kombination niemals unter dem Q_3-Index für *SPARROW. Bei den Graphen von PSIPRED + *SPARROW und PSIPRED hingegen ist der Abstand stärker ausgeprägt sodass fast kein Überlappen dieser Graphen festgestellt werden kann. Lediglich im Fall von Proteinen mit 17% Sequenzidentität zum Lerndatensatz besitzt PSIPRED einen deutlich höheren Q_3-Index als PSIPRED + *SPARROW, jedoch enthält die entsprechende Gruppe vergleichsweise wenige Residuen und kann aus diesem Grund vernachlässigt werden. Weiterhin zeigt sich, dass SPARROW + *SPARROW in vielen Fällen ein besseres Verhalten als PSIPRED besitzt, so zum Beispiel bei Proteinen mit 23% und 24% Sequenzidentität zum Lerndatensatz. Bei diesen beiden Gruppen von Proteinen, bestehend aus 11055 und 16697 Residuen, ist im linken Diagramm ein deutlicher Unterschied gegenüber PSIPRED zu erkennen. Interessant ist dabei, dass bereits *SPARROW im Fall der zweiten Gruppe ein geringfügig besseres Verhalten zeigt, das durch SPARROW + *SPARROW jedoch merklich verbessert wird.

Um die Untersuchung des Verhaltens der Kombination in Abhängigkeit von der Ähnlichkeit der vorhergesagten Proteine zum Lerndatensatz zu vervollständigen, soll an dieser

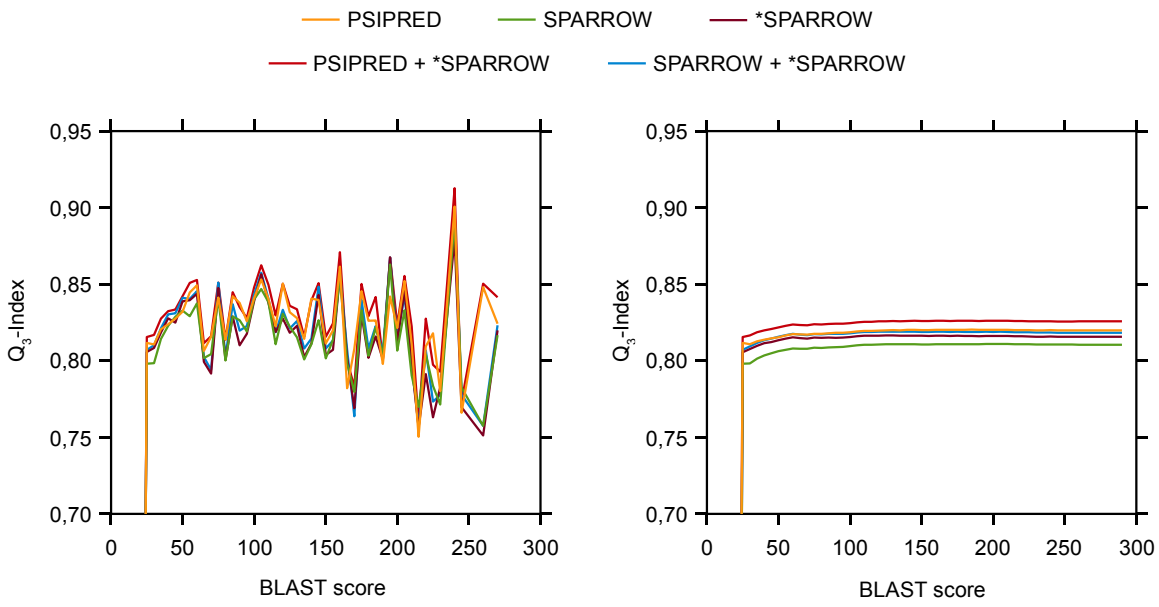


Abbildung 51: *Abhängigkeit des Q_3-Index vom BLAST Score auf bestimmten Intervallen (links) wie auch bis zu einem bestimmten Wert (rechts) für die Kombinationen von *SPARROW mit PSIPRED (PSIPRED + *SPARROW) beziehungsweise SPARROW (SPARROW + *SPARROW) im Vergleich zu den ursprünglichen Programmen*

Stelle auf Abbildung 51 verwiesen werden. Diese enthält, wie bereits Abbildung 47, den Q_3 -Index in Abhängigkeit vom BLAST Score, wobei links der Q_3 -Index für die einzelnen Werte und rechts das Integral über Intervalle des BLAST Scores gezeigt wird. Die starke Überlagerung der dargestellten Kurven erschwert ihre Analyse, jedoch kann festgestellt werden, dass auch hier die Graphen beider Kombinationen stets über denen der jeweiligen ursprünglichen Programme liegen. Das rechte Diagramm zeigt zudem, dass die Kurve für SPARROW + *SPARROW sehr nahe bei der Kurve von PSIPRED liegt und in einigen Fällen kaum von dieser unterschieden werden kann. Weiterhin zeigt sich, dass der Abstand zwischen dem Graphen für PSIPRED + *SPARROW und den Graphen der übrigen Vorhersageprogramme groß ist.

Eine weitere interessante Frage ist, inwiefern die Kombinationen bei der Vorhersage einzelner Proteine in Abhängigkeit von deren Länge von dem Verhalten der ursprünglichen Programme abweicht. Hierzu soll Abbildung 52 auf der folgenden Seite verwendet werden, in der das linke Diagramm den mittleren Q_3 -Index für Proteine mit Längen innerhalb bestimmter Längenintervalle zeigt und das rechte die entsprechende Größe bis zu einer bestimmten Länge visualisiert. Wie dem linken Diagramm entnommen werden kann, verbessert SPARROW + *SPARROW das Vorhersageverhalten gegenüber SPARROW und

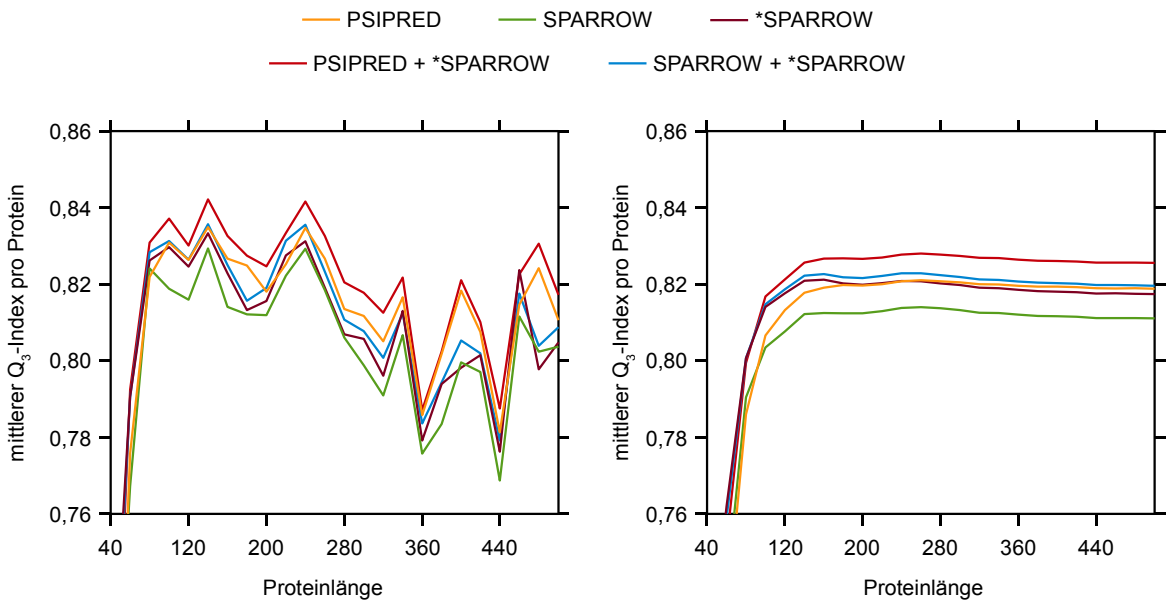


Abbildung 52: *Mittlerer Q_3-Index pro Protein in Abhängigkeit von der Proteinlänge auf bestimmten Intervallen (links) sowie bis zu einer Maximallänge (rechts) für die Kombinationen von *SPARROW mit PSIPRED (PSIPRED + *SPARROW) beziehungsweise SPARROW (SPARROW + *SPARROW) im Vergleich zu den ursprünglichen Programmen*

*SPARROW für fast alle Proteinlängen. Oberhalb einer Länge von 340 zeigen sich jedoch einige Fälle, in denen die Graphen von *SPARROW und SPARROW + *SPARROW zusammenfallen, sodass eine Verbesserung in diesen Fällen nicht ablesbar ist. Ähnlich verhält es sich mit PSIPRED + *SPARROW, wobei der entsprechende Graph jedoch stets über denen von PSIPRED und *SPARROW liegt. Die einzige Ausnahme stellen Proteine mit einer Länge zwischen 440 und 460 Residuen dar, bei denen die Graphen von PSIPRED + *SPARROW und *SPARROW zusammenfallen. Im Fall kurzer Proteine wird der Unterschied im Verhalten von *SPARROW und PSIPRED durch SPARROW + *SPARROW weiter akzentuiert. Dadurch weist SPARROW + *SPARROW in weit mehr Fällen, als es bei *SPARROW der Fall ist, einen höheren mittleren Q_3-Index pro Protein auf als PSIPRED. Dies führt wiederum dazu, dass SPARROW + *SPARROW, wie das rechte Diagramm zeigt, für Proteine bis zu einer bestimmten Maximallänge stets einen höheren mittleren Q_3-Index als PSIPRED aufweist. Der Unterschied zwischen beiden Verfahren ist, wie bereits im Fall von *SPARROW, besonders für kurze Proteine sehr stark ausgeprägt und nimmt mit steigender Maximallänge immer weiter ab. Die Stärke von SPARROW + *SPARROW liegt also hauptsächlich in der Vorhersage kurzer Proteine bis zu einer Länge von 240 Residuen, was ca. 79% des Testdatensatzes ausmacht. In diesem

Bereich zeigt auch das linke Diagramm für die meisten Proteinlängenintervalle ein besseres Verhalten von SPARROW + *SPARROW gegenüber PSIPRED. Durch den Vorsprung auf dieser Teilmenge der Testdaten, behält die Kombination ihren Vorsprung gegenüber PSIPRED im rechten Diagramm auch für längere Proteine bei, sodass sie im Endeffekt einen höheren mittleren Q_3 -Index pro Protein über den gesamten Datensatz besitzt. Weiterhin ist der große Unterschied zwischen dem Graphen für PSIPRED + *SPARROW und denen von PSIPRED und *SPARROW im rechten Diagramm auffällig. Dieser Graph liegt fast immer deutlich den übrigen dargestellten Graphen, wobei sehr kurze Proteine die Ausnahme bilden, da dort die Graphen für SPARROW + *SPARROW und *SPARROW oberhalb des Graphen für PSIPRED + *SPARROW liegen.

Bei Betrachtung der Abbildungen 50, 51 und 52 zeigt sich eine Gemeinsamkeit dieser drei Abbildungen: In allen drei Abbildungen ist der qualitative Verlauf des Graphen von SPARROW + *SPARROW jeweils in beiden Diagrammen augenscheinlich ähnlicher zu *SPARROW als zu SPARROW. Lediglich stückweise ist eine Ähnlichkeit zum Verlauf des jeweiligen Graphen von SPARROW festzustellen. Ähnlich verhält es sich mit den Graphen für PSIPRED + *SPARROW, deren Verlauf größtenteils dem Verlauf des jeweiligen Graphen für PSIPRED ähnelt, jedoch stückweise auch stärkere Ähnlichkeiten zu dem entsprechenden Graphen für *SPARROW aufweist.

Als abschließende Untersuchung soll die Verteilung der Q_3 -Indices pro Protein für beide Kombination mit SPARROW, *SPARROW wie PSIPRED verglichen werden. Diese ist in Abbildung 53 auf der folgenden Seite dargestellt, wobei das linke Diagramm die Population der einzelnen Werte des Q_3 -Index enthält, während das rechte Diagramm, analog zu Abbildung 49 auf Seite 181, die Integration des linken Diagramms enthält. Wie das linke Diagramm von Abbildung 53 zeigt, weisen beide Kombinationen eine Verteilung der Q_3 -Indices auf, die denen der jeweiligen ursprünglichen Programme gegenüber zu größeren Q_3 -Indices hin verschoben ist. Dies lassen bereits die entsprechenden Werte von $Q_3^{(\mathrm{Prot})}$, wie sie in Tabelle 13 dargestellt sind, vermuten. Dabei ist interessant, dass das Maximum des Graphen für SPARROW + *SPARROW stärker ausgeprägt zu sein scheint, als es bei allen anderen Graphen der Fall ist. Für den Graphen für PSIPRED + *SPARROW hingegen scheint das Maximum weniger stark ausgeprägt zu sein als bei *SPARROW, jedoch ist es dafür breiter als bei PSIPRED wie auch *SPARROW. Die Betrachtung des rechten Dia-

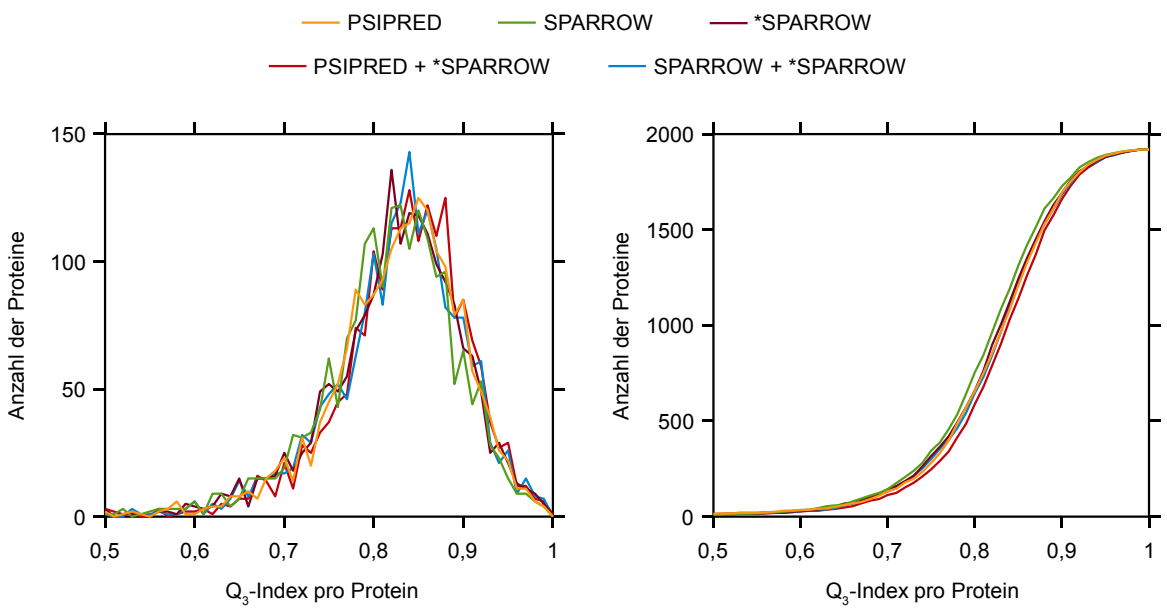


Abbildung 53: *Histogramm der Q_3-Indices pro Protein (links) und dessen Akkumulation (rechts) für die Kombinationen von *SPARROW mit PSIPRED (PSIPRED + *SPARROW) beziehungsweise SPARROW (SPARROW + *SPARROW) im Vergleich zu den ursprünglichen Programmen*

gramms zeigt weiterhin, dass in der Integration der Verteilung die Graphen von PSIPRED und SPARROW + *SPARROW kaum zu unterscheiden sind. Demgegenüber hebt sich der Graph von PSIPRED + *SPARROW deutlich von den übrigen dargestellten Graphen ab.

6. Diskussion

Die in Kapitel 5 dargestellten Ergebnisse zeigen, dass das neuartige Lernverfahren, das der vektorwertige Klassifikator darstellt, erfolgreich auf Mehrklassenprobleme angewandt werden kann. Bisher wurde jedoch eine weitergehende Interpretation und Diskussion dieser Ergebnisse ausgelassen. Es ist vor allem notwendig, anhand der Ergebnisse in Kapitel 5 die Funktionsweise und Generalisierungsfähigkeit von *SPARROW wie auch den Vergleich von *SPARROW mit anderen Vorhersageprogrammen zu diskutieren. Dies soll in diesem Kapitel geschehen, bevor im nächsten Kapitel auf mögliche weitere Entwicklungen und Erweiterungen der Methode eingegangen wird.

*6.1. *SPARROW als Lösung des Achtklassenproblems*

Die Ergebnisse für das Achtklassenproblem bilden ein instruktives Beispiel für die Funktionsweise der verwendeten Lernverfahren. Dies gilt sowohl für das in Abschnitt 4.2.5 beschriebenen Lernverfahren für den vektorwertigen Klassifikator, durch den die ersten beiden Stufen *SPARROWs realisiert sind, als auch für den Back-Propagation-Algorithmus (siehe Abschnitt 3.5.4.3) für das MLP in der dritten Stufe. Wie bereits in Abschnitt 5.3.1 beschrieben wurde, kann die Vorhersagequalität für die einzelnen Klassen (gemessen in Form der q_σ, siehe auch Abschnitt 4.3.4) in der ersten Stufe des 8-8-8- und 8-8(π)-8-Vorhersageschemas mit deren Anteil in den Kreuzvalidierungsdaten korreliert werden. Klassen mit vielen Vertretern (beispielsweise α-Helices, β-Stränge und strukturlose Regionen) werden somit besonders gut vorhergesagt, während die kleinsten Klassen (also β-Brücken und π-Helices) eine verschwindende Korrektheit aufweisen. Die Ursache dieser Korrelation zwischen q_σ und der Größe der entsprechenden Klasse ζ_σ ist eine Konsequenz des Lernverfahrens für den vektorwertigen Klassifikator, das den mittleren quadratischen Fehler als zu optimierendes Kriterium verwendet: Das Lernverfahren erzwingt eine möglichst hohe Zahl korrekter Klassifikationen über den gesamten Lerndatensatz, die sich in einem geringeren mittleren quadratischen Fehler äußert. Bei der Berechnung des mittleren quadratischen Fehlers als Mittelwert der Abweichung der einzelnen Lernbeispiele vom jeweiligen Sollwert, spielen Klassen mit wenigen Vertretern eine geringere Rolle, weshalb das Lernverfahren der Richtigkeit der Klassifikation solcher Klassen eine geringere Priorität beimisst. Eine solche Priorisierung muss aufgrund der Tatsache erfolgen, dass der vektorwer-

tige Klassifikator eine Lernmaschine endlicher Kapazität ist. Mit anderen Worten passt das Lernverfahren also die einzelnen q_σ dergestalt an, dass der jeweilige vektorwertige Klassifikator einen optimalen Wert von Q_8 auf dem Lerndatensatz erreicht. Somit ist Q_8 die Größe, die das Lernverfahren primär optimiert, während die einzelnen q_σ eine untergeordnete Rolle spielen. Prinzipiell gilt diese Aussage logischerweise gleichermaßen für die ersten beiden Stufen von *SPARROW, jedoch auch für das MLP in der dritten Stufe, da auch der Back-Propagation-Algorithmus den mittleren quadratischen Fehler über den Lerndatensatz minimiert.

Die sekundäre Rolle der q_σ im Lernvorgang wird in den höheren Stufen *SPARROWs, insbesondere in der dritten Stufe deutlich. So kann laut Abschnitt 5.3.1 gegenüber der ersten Stufe von *SPARROW eine Abnahme von q_C in der *Prediction* in der zweiten Stufe festgestellt werden, bei der eine statistische Fluktuation als Ursache ausgeschlossen werden kann. Aufgrund des höheren Wertes von Q_8 in der *Prediction* erscheint es jedoch, dass diese Abnahme von q_C für die Gesamtvorhersage günstig ist. Interessanterweise kann ein solcher Effekt im *Recall* nicht beobachtet werden, bei dem sowohl Q_8 als auch alle anderen Qualitätsmaße einen Zuwachs erfahren. Somit stellt sich die Frage, inwiefern eine Korrelation zwischen der Abnahme von q_C und der Zunahme von Q_8 in der *Prediction* besteht. Die Antwort auf diese Frage kann durch Betrachtung der Konfusionsmatrizen für die erste Stufe des 8-8-8- beziehungsweise 8-8(π)-8-Vorhersageschemas, wie sie in Abbildung 42 auf Seite 162 dargestellt sind, gefunden werden. Dort ist, wie bereits in Abschnitt 5.3.2.1 beschrieben wurde, eine starke Vermischung von α-Helices beziehungsweise strukturlosen Regionen mit anderen DSSP-Klassen zu beobachten. Diese Tatsache spricht dafür dass in der ersten Stufe eine übermäßige Vorhersage, oder auch *Übervorhersage*, dieser beiden Klassen stattfindet. Tatsächlich lässt sich anhand der ursprünglichen Konfusionsmatrizen berechnen, dass die erste Stufe ca. 22% mehr Helices und ca. 20% mehr strukturlose Regionen vorhersagt als insgesamt in den Daten vorhanden sind. Unter Einbeziehung der Korrektheit beider Klassen bedeutet dies, dass ca. 24% aller vorhergesagten α-Helices und ca. 45% aller vorhergesagten strukturlosen Regionen in Wirklichkeit zu einer anderen DSSP-Klasse gehören. Das sind im Fall der α-Helices ca. 8% und im Fall der strukturlosen Regionen ca. 9,6% des gesamten Datensatzes, die eine falsche Vorhersage durch die Übervorhersage dieser beiden Klassen erfahren. Gleichzeitig sind diese bei-

den Klassen diejenigen von den acht DSSP-Klassen, bei denen die Übervorhersage in der ersten Stufe am stärksten ausgeprägt ist. Bei der Betrachtung der zweiten Stufe (also des 8-8-Vorhersageschemas) zeigt sich, dass diese beiden Klassen im *Recall* unter den acht DSSP-Klassen die geringste Verbesserung von q_σ gegenüber der ersten Stufe aufweisen. Mittels der Konfusionsmatrizen der zweiten Stufe kann nun gezeigt werden, dass die Übervorhersage dieser beiden Klassen in der zweiten Stufe gegenüber der ersten reduziert ist. Dadurch weisen in der zweiten Stufe im Fall der α-Helix nur noch 7,2% aller Residuen des Datensatzes und im Fall der strukturlosen Regionen nur noch 9% aller Residuen des Datensathes eine Fehlklassifikation aufgrund der Übervorhersage beider Klassen auf. Die zweite Stufe ist somit tendenziell weniger dazu geneigt Residuen als α-Helices oder strukturlose Regionen zu klassifizieren, was der Vorhersage der übrigen Klassen und der Gesamtvorhersage zu Gute kommt und zudem die relativ geringe Zunahme von q_H und q_C erklärt. Diese Reduktion der Übervorhersage strukturloser Regionen und die kleineren Datensätze in der *Prediction* scheinen zudem zu der Abnahme des Wertes von q_C in der *Prediction* der zweiten Stufe führen. Wie die obige Argumentation jedoch zeigt, ist die Übervorhersage von α-Helices und strukturlosen Regionen in der zweiten Stufe immernoch recht groß. Es scheint jedoch, dass die dritte Stufe diese Übervorhersage weiter verringert, was an der Abnahme von q_H und q_C im Übergang zur dritten Stufe des 8-8-8- und 8-8(π)-8-Vorhersageschemas abgelesen werden kann. Weiterhin scheint die Verwendung von Wahrscheinlichkeitstermen (siehe Abschnitt 4.3.3.1) diesen Effekt geringfügig zu verstärken. Die zweite und dritte Stufe zeigen somit, dass auch eine Verschlechterung der Vorhersage großer Klassen von den einzelnen Lernverfahren zugunsten eines besseren Wertes von Q_8 erzwungen werden kann. Dies ist jedoch nur bei einem hierarchischen Aufbau, wie *SPARROW ihn besitzt, möglich, in dem eine Stufe die Trennung der Daten, wie sie von einer vorhergehenden Stufe stammt, beispielsweise durch den Abbau von Übervorhersage für die einzelnen Klassen verbessert. Dabei scheint die Verbesserung, die dieser hierarchische Aufbau im Fall des Achtklassenproblems mit sich bringt, mit jeder Stufe in der Hierarchie abzunehmen, wie die geringere Verbesserung der Vorhersage im Übergang von der zweiten zu dritten Stufe im Vergleich zum Übergang von der ersten zur zweiten zeigt. Dies könnte für eine Form von Sättigungseffekt sprechen.

Unter den gängigen Programmen zur Vorhersage der Sekundärstruktur von Proteinen ist dem Autor dieser Arbeit lediglich SSpro8 [29] bekannt, welches das Achtklassenproblem

direkt löst. Bei diesem Programm erweist es sich jedoch als schwierig einen direkten Vergleich zu ziehen. Das Programm ist als Webservice zugänglich, steht jedoch nicht in einer herunterladbaren Version zur Verfügung, weshalb das Arbeiten mit großen, aussagekräftigen Datensätzen sich mangels Automatisierungsmöglichkeiten sehr schwierig gestalten würde. In [29] sind Ergebnisse verfügbar, die das Verhalten der Software auf bestimmten Datensätzen zeigen. Ein direkter Vergleich mit diesen Ergebnissen birgt jedoch auch Schwierigkeiten, da aufgrund der sehr unterschiedlichen Größen der jeweiligen Lerndatensätze ein fairer Vergleich zwischen SSpro8 und *SPARROW unmöglich ist. Da *SPARROW sehr viel mehr Daten bekannt sind, als es damals bei SSpro8 der Fall war, würde *SPARROW auf den in [29] verwendeten Testdatensätzen besser abschneiden, zumal diese bei *SPARROW aller Wahrscheinlichkeit nach eine Teilmenge des Lerndatensatzes bilden. Für einen fairen Vergleich wäre es also notwendig, denselben Lerndatensatz zu verwenden, der auch in [29] verwendet wurde, jedoch ist dieser nicht zugänglich. Zudem müsste auch die Generierung der Profile für diesen Lerndatensatz auf die gleiche weise durchgeführt werden wie es in [29] der Fall war, was sich in Anbetracht der Tatsache, dass die hierzu verwendeten Datenbanken mittlerweile sehr viel mehr Daten enthalten, als schwierig erweist. Daher wurde in dieser Arbeit von einem direkten Vergleich beider Programme abgesehen. Jedoch können die Ergebnisse aus [29] als Orientierung verwendet werden, um das Verhalten beider Programme zumindest qualitativ zu vergleichen. Die Betrachtung dieser Ergebnisse zeigt, dass SSpro8 bei der Vorhersage der acht DSSP-Klassen qualitativ ein sehr ähnliches Verhalten aufweist wie die erste Stufe des 8-8-8- beziehungsweise 8-8(π)-8-Vorhersageschemas, jedoch hinsichtlich der q_σ aller Klassen (außer der π-Helices, für die SSpro8 und die erste Stufe *SPARROWs die gleiche verschwindende Vorhersagequalität aufweisen, und 3_{10}-Helices, für die SSpro8 ein besseres Verhalten zeigt), wie auch den Werten des Q_8 -Index hinter *SPARROW liegt. Dies könnte trotz der Unterschiede in den Daten und Profilen, die für die Tests beider Programme verwendet wurden, dafür sprechen, dass beide Programme bereits unter Verwendung der ersten Stufe von *SPARROW zumindest vergleichbar sind. Die Tatsache, dass das Verhalten der ersten Stufe von den anderen beiden Stufen *SPARROWs verbessert wird, bestärkt die Annahme, dass beide Programme zumindest vergleichbar sein sollten. Insgesamt zeigt die Untersuchung des Achtklassenproblems, dass *SPARROW prinzipiell in der Lage ist auch komplexere Klassifikationsprobleme zu lösen, was für die Mächtigkeit des Verfahrens spricht.

6.2. *SPARROW als Lösung des Dreiklassenproblems

Die Tatsache, dass in den ersten beiden Stufen des 3-3-3-Vorhersageschemas die einzelnen q_σ mit der Größe der jeweiligen Klassen korrelieren ergibt sich wie beim 8-8-8- beziehungsweise 8-8(π)-8-Vorhersageschema aus dem verwendeten Lernverfahren (siehe Abschnitt 6.1). Wie die Ergebnisse in Abschnitt 5.3.2 zeigen ist das 3-3-3-Vorhersageschema im Fall des Dreiklassenproblems das einzige, für das diese Regelmäßigkeit zutrifft. In den ersten beiden Stufen der übrigen Vorhersageschemata zur Lösung des Dreiklassenproblems, also des 8-8-3-, 8-4_E-3- und 8-4_C-3-Vorhersageschemas und deren Erweiterungen mittels der in Abschnitt 4.3.3.1 beschriebenen Wahrscheinlichkeitsterme, kann keine Übereinstimmung der Größenordnung der einzelnen q_σ in mit den Größen der einzelnen Klassen festgestellt werden. Diese Tatsache ergibt sich daraus, dass die ersten beiden Stufen dieser Vorhersageschemata im Lernvorgang vom Dreiklassenproblem abweichende Klassifikationsprobleme lösen, wie beispielsweise das Achtklassenproblem in ihren ersten Stufen oder Vierklassenprobleme in der zweiten Stufen des 8-4_E-3- beziehungsweise 8-4_C-3-Vorhersageschemas. Dadurch sind die Parameter dieser Stufen für eine andere Anzahl und Verteilung von Klassen optimiert, weshalb eine Verschiebung der für das Dreiklassenproblem berechneten q_σ gegenüber dem 3-3-3-Vorhersageschema festgestellt werden kann. Aus dem selben Grund, wie auch aufgrund der Tatsache dass dass Acht- beziehungsweise Vierklassenproblem schwieriger zu lösen ist als das Dreiklassenproblem, weisen die ersten beiden Stufen dieser Vorhersageschemata gegenüber denen des 3-3-3-Vorhersageschemas geringere Werte von Q_3 auf. Unter Verwendung einer einfachen Entscheidungsregel wie sie in Abschnitt 5.3.2 beschrieben wurde, die durch eine geeignetes Zusammenfassen einzelner Klassen in den ersten beiden Stufen dieser Vorhersageschemata eine Vorhersage von drei Klassen simuliert, erscheinen diese Vorhersageschemata in ihren ersten beiden Stufen gegenüber dem 3-3-3-Vorhersageschema als Lösung des Dreiklassenproblems als suboptimal. Bei Verwendung einer intelligenten Entscheidungsregel, als welche das MLP in der dritten Stufe *SPARROWs interpretiert werden kann, weisen diese vormals suboptimalen Vorhersageschemata in der dritten Stufe eine Vorhersagequalität auf, die die des 3-3-3-Vorhersageschemas übersteigt. Diese Tatsache verdeutlicht die Wichtigkeit der dritten Stufe, die es aufgrund ihrer Funktion als intelligente Entscheidungsregel ermöglicht in den ersten beiden Stufen Informationen von mehr als drei Klassen zur Lösung des Dreiklassenproblems zu verwenden. Die Verwendung dieser Informationen in den ersten beiden

Stufen von *SPARROW stellt, wie die Ergebnisse der dritten Stufen in Abschnitt 5.3.2 belegen, gegenüber dem direkten Weg mittels des 3-3-3-Vorhersageschemas einen Vorteil in der Vorhersage von drei Klassen und zudem eine der Stärken des Konzepts von *SPARROW dar.

Interessanterweise macht es dabei aus Sicht der dritten Stufe keinen Unterschied, ob in der zweiten Stufe vier oder acht Klassen verwendet werden, da 8-8-3-, 8-4_E-3- und 8-4_C-3-Vorhersageschema, beziehungsweise deren Erweiterungen in der *Prediction* hinsichtlich Q_3 äquivalent sind. Der Grund hierfür ist, dass die zweite Stufe des 8-8-3-Vorhersageschemas die acht Klassen, die sie unterscheidet, derart vermischt, dass das MLP in der dritten Stufe implizit mit vier Klassen arbeitet. Andererseits unterscheiden sich die zweiten Stufen von 8-4_E-3- und 8-4_C-3-Vorhersageschema lediglich in der Handhabung der isolierten β-Brücken, die wie Anhang C beschreibt eine sehr kleine Klasse bilden. Die Auswirkungen, die die unterschiedliche Handhabung dieser kleinen Klasse haben, scheinen in der zweiten Stufe entsprechend gering zu sein. Es kann lediglich ein höherer Wert von q_E für das 8-4_E-Vorhersageschema und ein höherer Wert für q_C für das 8-4_C-Vorhersageschema festgestellt werden, was zeigt, dass die Zuweisung der isolierten β-Brücken zu der jeweiligen Klasse eine geringe Verbesserung in der Vorhersage dieser Klasse bringt. Die Unterschiede zwischen beiden Vorhersageschemata sind jedoch nicht sehr groß. Das MLP scheint diese unterschiedlichen Informationen (und damit auch die q_σ) derart anzupassen, dass sowohl für die lockere als auch für das quasi-lockere Klasseneinteilung der bestmögliche Wert von Q_3 erreicht wird. Bei den geringen Unterschieden zwischen den zweiten Stufen beider Vorhersageschemata resultiert dies in der Äquivalenz ihrer dritten Stufen hinsichtlich Q_3 in der *Prediction*. Aus der Tatsache, dass 8-8-3-, 8-4_E-3- und 8-4_C-3-Vorhersageschema in der *Prediction* äquivalent sind, folgt, dass die Verbesserung, die diese drei Vorhersageschemata (sowie ihr Erweiterungen) gegenüber dem 3-3-3-Vorhersageschema bringen, nicht allein durch die erhöhte Parameterzahl beim 8-8-3-, 8-4_E-3- und 8-4_C-3-Vorhersageschema erreicht wird. Diese Behauptung kann damit begründet werden, dass das 8-8-3-Vorhersageschema gegenüber dem 8-4_E-3- und 8-4_C-3-Vorhersageschema in der zweiten und dritten Stufe doppelt so viele Parameter besitzt, jedoch in der *Prediction* zu diesen beiden Vorhersageschemata äquivalent ist. Wäre die Anzahl der Parameter allein ausschlaggebend, so müsste das 8-8-3-Vorhersageschema eine messbare Verbesserung gegenüber dem 8-4_E-3- und 8-4_C-3-Vorhersageschema darstellen. Dies wiederum spricht da-

für, dass das Konzept von *SPARROW mit seiner Möglichkeit, in den einzelnen Stufen unterschiedliche Klassifikationsprobleme mit unterschiedlicher Anzahl an Klassen zu lösen, eine Verbesserung gegenüber der direkten Handhabung der drei vorherzusagenden Klassen in jeder Stufe darstellt.

6.3. Generalisierungsfähigkeit und Überanpassung

Während der Validierungsphase (siehe Abschnitt 5.3) sollte unter anderem die Frage geklärt werden, ob und wie gut der vektorwertige Klassifikator generalisieren kann. Im Prinzip würde eine gute Generalisierungsfähigkeit in diesem Zusammenhang bedeuten, dass das System einerseits ein geringes Maß an Überanpassung besitzt, also der Abstand zwischen der Vorhersagequalität in Recall und Prediction im Mittel nicht allzu groß ist. Andererseits sollte das Verhalten des Systems bei unterschiedlichen Kombinationen von Lern- und Testdaten nicht zu sehr variieren. Das Problem dieser Definition von Überanpassung ist, dass kein objektives Maß für Überanpassung beziehungsweise Generalisierungsfähigkeit existiert. Die Formel zur Abschätzung der Überanpassung von MLPs (siehe Abschnitt 3.1.4) stellt lediglich eine Richtlinie dar und liefert kein objektives Maß für das Ausmaß an Überanpassung, dem ein MLP unterliegt. Zudem ist diese Regel nicht ohne weiteres auf beliebige Lernmaschinen übertragbar. Damit bleibt als einzige Möglichkeit der argumentative Weg unter Zuhilfenahme der durch die Kreuzvalidierung gewonnenen Ergebnisse, die nach der Argumentation in Abschnitt 3.1.4 zu diesem Zweck herangezogen werden können.

In der Kreuzvalidierung wurde das einfachste mögliche Mehrklassenproblem, nämlich das Dreiklassenproblem (Abschnitt 5.3.2), wie auch ein komplexeres Problem, das sich durch das Achtklassenproblem (Abschnitt 5.3.1) ergibt, untersucht. Um zunächst der Frage nach der Generalisierungsfähigkeit des vektorwertigen Klassifikators nachzugehen, sollen die ersten beiden Stufen der in Abschnitt 5.3 untersuchten Vorhersageschemata für *SPARROW betrachtet werden, da diese durch vektorwertige Klassifikatoren umgesetzt worden sind. Die Ergebnisse der Kreuzvalidierung auf dem Drei- und Achtklassenproblem (Tabellen 9 auf Seite 160, 16 auf Seite 237, 17 auf Seite 239 und 7 auf Seite 155) zeigen, dass die Standardabweichungen von Q_3 (bzw. Q_8) wie auch R_3 (bzw. R_8) im Vergleich zu dem jeweiligen Mittelwert sehr gering sind. Insbesondere im *Recall* sind die Standardabweichungen so gut wie vernachlässigbar, jedoch spielt für die Abschätzung der Generali-

sierungsfähigkeit die *Prediction* eine größere Rolle, da die Fähigkeit unbekannte Daten vorherzusagen die eigentliche Generalisierung darstellt. Hier ist die Standardabweichung zwar höher als im *Recall* (was sich aus der zehnmal kleineren Datenmenge in der *Prediction* ergibt), jedoch zeigt ein Vergleich mit dem entsprechenden Mittelwert, dass die Unterschiede in der Güte der Vorhersage zwischen den einzelnen Datensätzen sehr gering sind. Dies zeigt, dass der vektorwertige Klassifikator keine starke Bevorzugung einer bestimmten Kombination von Lern- und Testdaten aufweist. Vielmehr scheint das Lernverfahren für alle KV_α der Kreuzvalidierung ein sehr ähnliches Verhalten zu zeigen, was für die generelle Anwendbarkeit des Lernverfahrens spricht. Das hinzunehmen der dritten Stufe in die Betrachtung ändert nichts an der obigen Argumentation, da die dritte Stufe die Standardabweichungen der beiden Werte gegenüber der zweiten Stufe kaum verändert. Dies wiederum bedeutet, dass jedes der Vorhersageschemata, die zur Umsetzung von *SPARROW untersucht wurden, in diesem Sinne gut generalisiert.

Bei weiterer Betrachtung der genannten Tabellen fällt jedoch auch auf, dass die einzelnen Vorhersageschemata mit einem gewissen Maß an Überanpassung behaftet sind, das sich in einer mehr oder weniger ausgeprägten Differenz in Q_3 und R_3 (bzw. Q_8 und R_8 im Fall des Achtklassenproblems) zwischen *Recall* und *Prediction* äußert. Diese Differenz ist beim Achtklassenproblem stärker ausgeprägt als beim Dreiklassenproblem, was nicht zuletzt an der erhöhten Parameterzahl beim Acht- gegenüber dem Dreiklassenproblem liegt. Die Größenordnung dieser Differenz zwischen *Recall* und *Prediction* ist jedoch ähnlich zu dem in [29] für SSpro (beziehungsweise SSpro8) und in [1] für SPARROW angegebenen Unterschied zwischen *Recall* und *Prediction*. Deshalb wird angenommen, dass im Fall von *SPARROW der für die einzelnen Vorhersageschemata gemessene Unterschied zwischen *Recall* und *Prediction* eine vergleichbare Größenordnung besitzt, wie es bei anderen Programmen zur Sekundärstrukturvorhersage der Fall ist.

Somit sprechen die in der Kreuzvalidierung gemessenen Mittelwerte und Standardabweichungen von $Q_{N_3^{Klassen}}$ und $R_{N_3^{Klassen}}$ dafür, dass sowohl der vektorwertige Klassifikator als auch das damit umgesetzte *SPARROW eine mit ähnlichen Verfahren vergleichbare Generalisierungsfähigkeit besitzen. Durch die Größe der verwendeten Datensätze kann zudem ausgeschlossen werden, dass die Ergebnisse, auf denen die obige Argumentation aufbaut, das Ergebnis eines Zufalls sind. Dies wiederum bedeutet, dass der vektorwertige Klassifi-

kator prinzipiell auf beliebige Klassifikationsprobleme und Datensätze anwendbar sein sollte, solange die verwendeten Lerndatensätze gegenüber der Parameterzahl nicht zu klein sind und dadurch das Maß an Überanpassung verstärken. Wie bereits erwähnt, existiert kein objektives Maß für die Überanpassung des vektorwertigen Klassifikators, sodass es im Ermessen des Anwenders liegt, ein geeignetes Verhältnis von Lerndatensatz und Parameterzahl zu finden.

6.4. Anzahl der Klassen in der Sekundärstrukturvorhersage

In dieser Arbeit stand aus einer Reihe von Gründen das Dreiklassenproblem im Mittelpunkt. Einer davon ist, dass Vorhersageschemata, die das Dreiklassenproblem lösen, generell eine höhere Genauigkeit in ihren Vorhersagen besitzen als es bei Vorhersageschemata zur Lösung des Achtklassenproblems der Fall ist (siehe Abschnitte 5.3.1 und 5.3.2.2, beziehungsweise Anhang C). Prinzipiell wäre es von Vorteil, die acht DSSP-Klassen direkt zu lernen, da somit keinerlei Informationsverlust stattfindet. Die Verwendung von drei Klassen erfordert immer eine Interpretation der von DSSP gelieferten Klassen um zu bestimmen, welche Klassen zu einer der drei Klasse vereinigt werden sollen. Dies führt dazu, dass teils sehr unterschiedliche Reduktionen der acht DSSP-Klassen zu drei Klassen existieren. Allein schon in dieser Arbeit werden drei unterschiedliche Klasseneinteilungen zur Erzeugung der drei Klassen beschrieben, wobei die lockere Klasseneinteilung, wie bereits in Abschnitt 4.1.3.2 erwähnt, schon früher vom EVA-Projekt zum Vergleich verschiedener Methoden zur Sekundärstrukturvorhersage von Proteinen verwendet wurde und die strenge Klasseneinteilung in [1] bei der Entwicklung von SPARROW Anwendung fand. Die quasilockere Klasseneinteilung, die in dieser Arbeit entwickelt wurde, stellt eine dritte Möglichkeit dar die acht DSSP-Klassen auf drei Sekundärstrukturklassen zu reduzieren. In [30] wird wiederum eine Klasseneinteilung erwähnt, die für die Tests mit PSIPRED verwendet wurde und auf Rost und Sander [76] zurückgeht. Nach dieser Klasseneinteilung bilden 3_{10}-Helices und α-Helices die helikale Klasse, isolierte β-Brücken und β-Stränge die zusammengesetzte β-Strangklasse und die restlichen Klassen die Klasse der strukturlosen Regionen. Jede dieser Klasseneinteilungen entspringt einer logischen Interpretation der acht DSSP-Klassen und für jede können unterstützende wie auch widersprechende Argumente gefunden werden. Gegenwärtig liegt es aufgrund des Mangels einer einheitlichen Klasseneinteilung im Ermessen des einzelnen, welche Klasseneinteilung er verwendet. All diesen

Klasseneinteilungen ist jedoch gemein, dass sie eine Interpretation der acht DSSP-Klassen bezüglich Ähnlichkeit oder auch Wichtigkeit der einzelnen Klassen erfordern. Diese Interpretationen können sich jedoch, wie die obigen Beispiele zeigen, je nach Argumentation sehr stark unterscheiden. Die direkte Verwendung der acht Klassen würde das Problem der Interpretation beseitigen indem es zur Basis der genannten Klasseneinteilungen zurückkehrt: Den acht DSSP-Klassen.

Jedoch erscheint es beim gegenwärtigen Stand der Technik nicht sinnvoll, die acht Klassen direkt zur Vorhersage der Sekundärstruktur von Proteinen zu verwenden. Es scheint, dass einerseits die Kapazität heutiger Ansätze, wie beispielsweise *SPARROW und SSpro8, nicht ausreicht um das Problem vollständig abzubilden. Das Achtklassenproblem stellt ein sehr komplexes Problem dar, was zu einer geringeren Trennbarkeit der betreffenden Daten führt und unter Umständen ein höheres Maß an nichtlinearität erfordert, die bisherige Methoden nicht bieten können. Diese Komplexität äußert sich nicht nur in dem deutlich niedrigeren Wert von Q_8 beim Achtklassenproblem im Vergleich zum Wert von Q_3 im Fall des Dreiklassenproblems, sondern auch in der Vorhersagequalität der einzelnen Klassen bei beiden Klassifikationsproblemen. Diese ist im Fall des Achtklassenproblems generell niedriger als beim Dreiklassenproblem, wie die Ergebnisse in dieser Arbeit wie auch in [29] belegen. Für die folgende Argumentation sollen die Ergebnisse der vorliegenden Arbeit herangezogen werden: Die Klassen der α-Helices und β-Stränge im Fall der vollständigen Klasseneinteilung und die helikale und β-Strangklasse bei der strengen Klasseneinteilung sind prinzipiell gleich definiert (vergleiche Tabelle 2 auf Seite 111). Der Vergleich zwischen dem 8-8(π)-8-Vorhersageschema (siehe Tabelle 7 auf Seite 155) und dem 8-8(π)-3-Vorhersageschema (siehe Tabelle 9 auf Seite 160), zeigt zwar einen gewissen Unterschied in der Klassifikation dieser beiden Klassen, der zu Gunsten des 8-8(π)-8-Vorhersageschemas ausfällt. Jedoch ist die Verbesserung, die das 8-8(π)-8-Vorhersageschema bei der Klassifikation dieser Klassen birgt, gering im Vergleich zum allgemeinen Verlust an Genauigkeit (Verringerung von Q_8 gegenüber Q_3) und der geringeren Korrektheit in der Vorhersage der übrigen Klassen beim 8-8(π)-8-Vorhersageschema.

Weiterhin stellt beim Achtklassenproblem die teils sehr geringe Genauigkeit bei der Klassifikation bestimmter Klassen, wie beispielsweise π-Helices und β-Brücken einen Widerspruch zu der Idee dar, Informationsverlust zu vermeiden. So wird durch das behandeln von acht Klassen zwar mehr Information geboten, jedoch ist diese Information aufgrund

der teils sehr geringen Genauigkeit der kleineren Klassen unzuverlässig. Grund für diese Unzuverlässigkeit ist, wie bereits in Abschnitt 6.1 angemerkt, die geringe Zahl der Vertreter dieser Klassen, die dafür sorgt, dass sie im Lernverfahren gegenüber den großen Klassen nicht ins Gewicht fallen. Eine Möglichkeit dieses Problem zu umgehen wäre die Einführung einer Gewichtung oder Priorisierung, die den einzelnen Klassen im Lernverfahren ein größeres Gewicht beimisst und somit zu einer verbesserten Vorhersage der entsprechenden Klasse führen sollte. Erste Tests mit diesem Ansatz haben jedoch gezeigt, dass die Verwendung einer solchen Priorisierung zu Einbußen in der Gesamtvorhersage führt. Dies lässt sich so erklären, dass durch eine höhere Priorisierung einer kleinen Klasse, die anderen Klassen logischerweise im Lernverfahren an Gewicht verlieren und somit die Genauigkeit der Vorhersage dieser Klassen sinkt. Jedoch ist das Problem dabei, dass große Klassen einen größeren Einfluss auf den Wert von Q_8 haben als kleine. Die Verbesserung der Vorhersage einer kleinen Klasse bei gleichzeitiger Abnahme der Vorhersage einer großen Klasse führt somit logischerweise zu einem Verlust in Q_8. Aus Abschnitt 6.1 geht weiterhin hervor, dass die Lernverfahren für den vektorwertigen Klassifikator wie auch für das MLP die Parameter der einzelnen Stufen derart bestimmen, dass ein optimaler $Q_{N_3^{Klassen}}$ bei gegebener Verteilung der Klassen erreicht wird, wobei die Korrektheit der Klassifikation der einzelnen Klassen im Hintergrund steht. Auch dies würde dafür sprechen, dass jedwede Priorisierung der Klassen im Fall der untersuchten Mehrklassenprobleme zu niedrigeren Werten für Q_8 führt.

Inwiefern also Veränderungen der Methode zu einer Verbesserung der Vorhersage der acht DSSP-Klassen beitragen können ist fraglich. Die oben erwähnte Modifikation, wie auch eine weitere Erhöhung der Parameterzahl können da sehr wahrscheinlich keine Verbesserung bringen. Einzig ein Lernverfahren, das statt des Q_8-Index die einzelnen q_σ direkt optimiert, könnte unter Umständen dazu beitragen, dass der Informationsgehalt in der Klassifikation der einzelnen DSSP-Klassen erhöht wird. Aufgrund der obigen Argumentation erscheint es jedoch sinnvoller, bei der Sekundärstrukturvorhersage von Proteinen zunächst beim Dreiklassenproblem zu bleiben. Verfahren die das Achtklassenproblem lösen, sollten jedoch nicht prinzipiell als uninteressant angesehen werden. So könnte die Vorhersage des 8-8(π)-8-Vorhersageschemas beispielsweise als zusätzliche Information verwendet werden, um die Vorhersage von drei Klassen durch *SPARROW zu unterstützen oder

dem Benutzer zusätzliche Informationen über die vorhersagten Klassen liefern. Weiterhin stellen die Ergebnisse, die *SPARROW auf dem Achtklassenproblem erreicht, einen wichtigen Beleg für die allgemeine Vergleichbarkeit und Kapazität des vektorwertigen Klassifikators dar. So zeigt sich in Abschnitt 6.1, dass bereits die erste Stufe *SPARROWs, die mittels eines vektorwertigen Klassifikator umgesetzt wurde, mit SSpro8 vergleichbar oder sogar besser als dieses sein könnte. Zwar ist diese Aussage mangels eines direkten Vergleichs beider Programme unsicher, jedoch spricht sie andererseits auch dafür, dass vektorwertige Klassifikatoren und neuronale Netze sich gleichermaßen gut zur Lösung bestimmter Probleme eignen können. Dies scheint insbesondere für das Achtklassenproblem der Sekundärstrukturvorhersage zu gelten, das wie bereits beschrieben wurde ein sehr komplexes Problem ist.

Neben dem Achtklassenproblem bestünde durch die Architektur von *SPARROW auch die Möglichkeit ein Vierklassenproblem zu lösen, indem die als intermediär$_C$ beziehungsweise intermediär$_E$ bezeichnete Klasseneinteilung (siehe Abschnitt 5.3.2.1) für die Vorhersage in der dritten Stufe verwendet wird. Erste Tests mit dem 8-4$_C$- und dem 8-4$_E$-Vorhersageschema zeigen, dass beide Vorhersageschemata mit Q_4 -Indices bei ca. 0,7505 beziehungsweise 0,7459 zwischen den Vorhersageschemata für acht Klassen und denen für drei Klassen anzuordnen sind. Inwiefern die Vorhersage von vier Klassen in der dritten Stufe jedoch zu einer weiteren Verbesserung dieser Werte führt und die Bearbeitung des Vierklassenproblem gegenüber Acht- und Dreiklassenproblem Vorteile bringt, ist zur Zeit ungeklärt und muss durch weitere Experimente untersucht werden. Auch in diesem Fall wäre jedoch, wie auch im Fall des Achtklassenproblems, das Berücksichtigen der Vorhersage eines Klassifikators mit vier Klassen zur Verbesserung der Vorhersage von drei Klassen denkbar.

6.5. Die quasi-lockere Klasseneinteilung

In Abschnitt 6.4 wurde bereits angemerkt, dass jeder Klasseneinteilung eine Interpretation der acht DSSP-Klassen zugrunde liegt, die durch eine entsprechende Argumentation über die Ähnlichkeit und Bedeutung der einzelnen Sekundärstrukturtypen gestützt wird. Dabei wird die Ähnlichkeit der Sekundärstrukturtypen zueinander zumeist auf Grundlage strukturbiologischer Überlegungen bestimmt. Die in dieser Arbeit bisher verwendete Argumentation zur Rechtfertigung des quasi-lockeren Klassifikationsschemas ist jedoch rein infor-

mationstechnischer Natur und begründet die Einführung der Klasseneinteilung ausschließlich anhand von Ähnlichkeiten in den Eingangsdaten. Tatsächlich muss aus der Sicht von *SPARROW ein gewisses Maß an Ähnlichkeit zwischen den Eingangsvektoren, die zu Vertretern bestimmter Klassen gehören, existieren, die das starke Überlappen einiger Klassen in der Klassifikation erklären würde (siehe Abbildung 42 auf Seite 162 und Abbildung 43 auf Seite 164). Zudem könnte auch angeführt werden, dass der Q_3 -Index, der bei der quasi-lockeren Klasseneinteilung gegenüber der lockeren Klasseinteilung erhöht ist, dafür sprechen könnte, dass die quasi-lockere Klasseneinteilung eher den strukturbiologischen Gegebenheiten entspricht. Dabei ist anzumerken, dass dieser Effekt nicht nur im Fall von *SPARROW sondern auch im Fall von PSIPRED beobachtet werden kann. So kann durch die Definition der Richtigkeit der Klassifikation gemäß der quasi-lockeren Klasseneinteilung der Q_3 von PSIPRED auf dem Testdatensatz der Anwendungstests von 0,8194 bei der lockeren Klasseneinteilung auf 0,8235 erhöht werden. Diese Begründung allein reicht jedoch nicht aus, um auch die strukturbiologische Relevanz der quasi-lockeren Klasseneinteilung zu klären, weshalb dieser Sachverhalt einiger weiterer Argumentation bedarf.

Wie in Abschnitt 5.3.2.1 beschrieben, unterscheiden sich die lockere und die quasi-lockere Klasseneinteilung lediglich in der Zuweisung der isolierten β-Brücken. Dieser Unterschied soll anhand einiger struktureller Überlegungen erörtert werden. Die helikale Klasse, die in beiden Klassifikationsschemata identisch definiert ist, beinhaltet alle helikalen Sekundärstrukturtypen. Diese besitzen, was Wasserstoffbrücken und Torsionswinkel angeht, Gemeinsamkeiten, die sich in der Periodizität beider Eigenschaften entlang der Sequenz äußern (siehe Abschnitte 2.3.4.1 bis 2.3.4.3). Aufgrund dieser strukturellen Ähnlichkeiten liegt es nahe, die drei helikalen Klassen zu einer Klasse zusammenzufassen, wie das in beiden Klasseneinteilungen der Fall ist. Aus der Sicht der lockeren Klasseneinteilung kann nun auch argumentiert werden, dass β-Stränge und isolierte β-Brücken eine Gemeinsamkeit in den sie ausmachenden Wasserstoffbrücken besitzen, die sequenziell weit voneinander entfernte Residuen eines Proteins verbinden. Wie in Abschnitt 2.4.1.2 erörtert, können diese Wasserstoffbrücken durchaus eine wichtige Rolle für die Faltung und die Stabilität eines Proteins besitzen, jedoch können auch andere Faktoren zur Stabilität von Proteinen beitragen (siehe Abschnitt 2.4.1.1 und 2.4.1.3). Zwar ist es aufgrund der potentiellen Rolle von Wasserstoffbrücken verständlich, dass ein Bestreben besteht β-Stränge und β-Brücken aufgrund dieser Eigenschaft zu einer Klasse zu vereinen, jedoch ist es auch ausschließlich

diese Eigenschaft die sie teilen. Was die Torsionswinkel angeht, so existiert im Fall der isolierten β-Brücken im Gegensatz zu β-Strängen diesbezüglich keinerlei Konvention (siehe Abschnitt 2.3.4.5). Somit stellen die β-Stränge unter dem Aspekt der Torsionswinkel einen sehr regelmäßigen Sekundärstrukturtyp dar, während isolierte β-Brücken diesbezüglich hochgradig irregulär sind. Zudem bestehen isolierte β-Brücken in der Regel aus einzelnen, wie der Name bereits aussagt, isolierten Residuen, während β-Stränge dadurch charakterisiert sind, dass sie einen Abschnitt von mehreren Residuen umfassen. Durch diese Unregelmäßigkeiten, die den isolierten β-Brücken zu eigen sind, ähneln diese strukturlosen Regionen eines Proteins. Aus diesem Grund erscheint es logisch, die isolierten β-Brücken auch der Klasse der strukturlosen Regionen zuzuordnen, was in der quasi-lockeren Klasseneinteilung realisiert wird.

Die Zuweisung der übrigen DSSP-Klassen zu den strukturlosen Regionen wird von beiden Klasseneinteilungen (quasi-locker und locker) einheitlich gemacht. Dies muss bei den eigentlichen strukturlosen Regionen aus der DSSP-Klassifikation wie auch den Biegungen nicht weiter diskutiert werden, da beide Sekundärstrukturtypen keinerlei Restriktionen bezüglich Wasserstoffbrücken und Torsionswinkeln besitzen und somit vom strukturellen Aufbau prinzipiell ähnlich sind. Es bleibt die Frage zu klären, ob die Klassifikation von Windungen als strukturlose Regionen sinnvoll ist. Windungen besitzen durch gewisse Bedingungen an Wasserstoffbrückenmuster und Torsionswinkel (siehe Abschnitt 2.3.4.6) eine Ähnlichkeit zu Helices oder β-Strängen, jedoch mangelt es ihnen an der Periodizität, die diesen Sekundärstrukturtypen eigen ist. So können Residuen entlang einer Windung verschiedenen Windungstypen zugeordnet sein, wodurch die sequentiellen Abstände zwischen den Bindungspartnern der einzelnen Wasserstoffbrücken wie auch die Torsionswinkel variieren können. Durch diesen Mangel an Regelmäßigkeit erscheint es eher sinnvoll, die Windung der Klasse der strukturlosen Regionen zuzuordnen. Erwähnenswert ist dabei, dass Windungen einen sehr einzigartigen Sekundärstrukturtyp darstellen, der die Regelmäßigkeit von Wasserstoffbrücken und Torsionswinkeln, wie sie bei Helices und β-Strängen auftreten, mit dem Mangel an Periodizität dieser beiden Eigenschaften, wie er im Fall der strukturlosen Klassen vorkommt, vereint. Dies mag einer der Gründe sein, wieso sich die Klasse der Windungen durch das Zusammenfassen der DSSP-Klassen nach Abschnitt 5.3.2.1 in Abbildung 42 auf Seite 162 als eigenständige Klasse in der intermediär$_C$- beziehungsweise intermediär$_E$-Klasseneinteilung herauskristallisiert.

Alles in allem zeigt die obige strukturelle Betrachtung, dass die quasi-lockere Klasseneinteilung auch vom strukturbiologischen Standpunkt aus sinnvoll erscheint. Zudem gibt die obige Argumentation auch einen Grund dafür, warum eine Ähnlichkeit in den Eingangsvektoren der Vertreter bestimmter Klassen existieren muss: Alle bisher entwickelten Verfahren zur Sekundärstrukturvorhersage von Proteinen basieren, wie auch bei *SPARROW, auf der Verwendung von Sequenzfenstern zur Vorhersage der Sekundärstruktur eines Residuums (siehe Abschnitt 4.1.3). Diese Sequenzfenster stellen lediglich Ausschnitte der Sequenz dar, die zwar durch die Einführung von Sequenzprofilen ein gewisses Maß an Delokalisierung enthalten, jedoch was die verarbeiteten Informationen angeht immer noch sehr lokalisiert sind. Auf Grundlage dieser Informationen müssten somit aus Sicht einer Methode zur Vorhersage der Sekundärstruktur von Proteinen diejenigen Sekundärstrukturtypen ein großes Maß an Ähnlichkeit besitzen, die Aufgrund lokaler Muster in den Wasserstoffbrücken und Torsionswinkeln gewisse Übereinstimmungen zeigen. Daraus würde sich nach der obigen Argumentation die Zusammenfassung der acht DSSP-Klassen nach der quasi-lockeren Klasseneinteilung ergeben.

Die Lokalität der verarbeiteten Information und die sich daraus ergebende Ähnlichkeit der Sekundärstrukturtypen und Eingangsvektoren scheint somit für die Verbesserung des Q_3-Index im Fall der quasi-lockeren Klasseneinteilung gegenüber der lockeren verantwortlich zu sein. Wie dieser Abschnitt zudem verdeutlicht, besitzt die quasi-lockere Klasseneinteilung jedoch auch eine strukturbiologische Bedeutung, die mit lokalen Ähnlichkeiten in den einzelnen Sekundärstrukturtypen einhergeht. Aus beiden Gründen erscheint es sinnvoll, die quasi-lockere Klasseneinteilung der lockeren vorzuziehen. Inwiefern die strenge Klasseneinteilung aufgrund des generell höheren Q_3-Index und der in Abschnitt 4.1.3.2 gegebenen Begründung allein der quasi-lockeren Klasseneinteilung vorzuziehen ist, ist fraglich. Zwar kann argumentiert werden, dass lediglich α-Helices und β-Stränge wirklich regelmäßige und stabile Sekundärstrukturtypen darstellen, womit die anderen sechs Sekundärstrukturtypen per definitionem strukturlos wären, jedoch beinhaltet diese Argumentation keinerlei Information über die strukturellen Gemeinsamkeiten der einzelnen Sekundärstrukturtypen. Aus informationstechnischer Sicht kann sich der höhere Q_3-Index bei der strengen Klasseneinteilung daraus ergeben, dass zwei der drei Klassen dieser Klasseneinteilung von DSSP-Klassen (α-Helices und β-Stränge) gebildet werden, die von der Vereinigung der restlichen sechs DSSP-Klassen als dritter Klasse getrennt werden. Diese Aufgabe sollte ge-

nerell einfacher sein als die Trennung mehrerer gemischter Klassen voneinander, da die Ähnlichkeiten in Struktur und Eingangsvektoren innerhalb ein und derselben DSSP-Klasse, insbesondere bei großen und regelmäßigen Klassen wie α-Helices und β-Strängen, groß sein sollte. Diese These scheinen die hohen Werte von q_H und q_E beim Achtklassenproblem zu untermauern, da sie dafür sprechen, dass die entsprechenden Klassen von *SPARROW sehr gut erkannt werden können. Diese Erkennbarkeit beider Klassen sollte es wiederum einfach machen, sie von den restlichen DSSP-Klassen zu separieren.

Prinzipiell wäre es von Vorteil, wenn zunächst eine einheitliche Klasseneinteilung gewählt werden könnte, um die Vergleichbarkeit der einzelnen Methoden zur Sekundärstrukturvorhersage von Proteinen zu verbessern. Wie dieser Abschnitt zeigt, besitzt jede Klasseneinteilung je nach Argumentation Vor- und Nachteile. Mit welchen Argumenten eine einheitliche Klasseneinteilung gewählt werden sollte ist eine Frage, die gegenwärtig schwer zu klären ist. Grundsätzlich sollte bei so einer Argumentation den strukturbiologischen Argumenten vor den informationstechnischen Argumenten Vorzug gegeben werden, denn schließlich sind Proteine chemische Verbindungen und nicht bloße künstliche Datenobjekte der digitalen Welt. Jedoch sollten die informationstechnischen Argumente dabei nicht außer Acht gelassen werden, da sie eine neue Sicht auf Proteine bieten, die eine Erweiterung der bestehenden Sicht darstellen könnte.

6.6. *SPARROW im Vergleich mit anderen Vorhersagemethoden

Laut des in Abschnitt 5.4.2 dargestellten Vergleichstests ist *SPARROW unter den getesteten Programmen an zweiter Stelle, was den Q_3 -Index auf dem gesamten Datensatz (sowohl residuen- wie auch proteinbasiert) angeht. Die Ergebnisse zeigen, dass *SPARROW in der Korrektheit der gemachten Vorhersagen seinem Vorgänger SPARROW überlegen ist und somit eine erfolgreiche Verbesserung des Konzepts von SPARROW darstellt. Zudem zeigt sich, dass *SPARROW, was den mittleren Q_3 -Index pro Protein angeht, sehr nahe bei PSIPRED angesiedelt ist (vergleiche Tabelle 12 auf Seite 174) und für die Vorhersage kurzer Proteine sogar besser geeignet scheint (siehe Abschnitt 5.4.2). Nun stellt sich einerseits die Frage nach der Aussagekraft dieser Ergebnisse und andererseits danach, in welcher Hinsicht *SPARROW gegenüber SPARROW eine konzeptionelle Verbesserung darstellt.

Prinzipiell spricht die Größe des Testdatensatzes dafür, dass die gezeigten Ergebnisse und

die daraus gezogenen Schlussfolgerungen signifikant sind. Einerseits ist die Menge an Proteinen und Residuen im Testdatensatz groß genug um statistische Effekte als Ursachen für die Unterschiede der einzelnen Programme auf diesem Datensatz auszuschließen. Zusätzlich sind diese Unterschiede groß genug, um der Einordnung der einzelnen Methoden entsprechend ihrem residuenbasierten Q_3-Index Glauben zu schenken. Die Ergebnisse der Kreuzvalidierung in der Validierungsphase von *SPARROW (Anhang C) zeigen, dass die Standardabweichung des Q_3-Index der einzelnen Vorhersageschemata zur Umsetzung *SPARROWs für die lockere Klasseneinteilung in der *Prediction* im Mittel bei ungefähr 0,0032 liegt (Tabelle 16, Seite 237). Diese Standardabweichungen wurden auf Datensätzen erreicht, die sowohl von der Zahl der Proteine als auch der Zahl der Residuen ungefähr ein Drittel des Testdatensatzes in den Vergleichstests ausmachen. Dies wiederum würde bedeuten, dass für eine Reihe hinsichtlich des Mittelwerts von Q_3 äquivalenter Ansätze die Streuung des Q_3-Index auf dem Testdatensatz grundsätzlich unterhalb von 0,0032 liegen sollte. Die Unterschiede im Q_3-Index der sechs in Abschnitt 5.4.2 verglichenen Programme liegen jedoch bei minimal 0,0041 (Tabelle 12, Seite 174). Damit sind die Unterschiede zwischen den Q_3-Indices der einzelnen Vorhersageprogramme auf dem Testdatensatz größer als die geschätzte Standardabweichung, die äquivalente Methoden auf diesem Datensatz haben sollten. Weiterhin zeigen die Graphen in Abschnitt 5.4.2, insbesondere in den Diagrammen der Abbildungen 46 bis 49 die das Integral der jeweils dargestellten Größe zeigen, dass die Rangordnung, die sich aus den Q_3-Indices der einzelnen Vorhersageprogramme ergibt, bereits für Teilmengen des Testdatensatzes eingenommen wird. Dies spricht für die Gültigkeit der gefundenen Rangordnung. Als einziger Kritikpunkt erweist sich lediglich das Problem, dass die einzelnen Vorhersageprogramme sehr wahrscheinlich auf unterschiedlichen Lerndatensätzen trainiert wurden. Bei SPARROW und *SPARROW ist der Lerndatensatz bekannt und in beiden Fällen identisch, so dass gewährleistet ist, dass ein durch den Q_3-Index auf dem Testdatensatz gezogener Vergleich in diesem Fall fair ist. Jedoch ist bei den anderen Programmen im Vergleich unbekannt, welche Lerndaten verwendet wurden, sodass nicht ausgeschlossen werden kann, dass die Lerndaten für ein bestimmtes Vorhersageprogramm entweder veraltet waren oder aber eine Teilmenge des Testdatensatzes enthielten. Der erste Fall würde bedeuten, dass das betreffende Programm einen entschiedenen Nachteil auf dem Testdatensatz haben könnte. Der zweite würde einen Vorteil für das betreffende Vorhersageprogramm bedeuten, da Daten,

die bereits im Lerndatensatz enthalten sind, in der Regel mit einer höheren Genauigkeit vorhergesagt werden können. Selbst mit der Annahme, dass die einzelnen Programme möglichst viele Proteine, deren Strukturen zur Zeit des Lernvorgangs bekannt waren, gelernt haben, können diese beiden Fälle nicht ausgeschlossen werden. Jedoch spricht die Tatsache, dass alle sechs Vorhersageprogramme ein sehr ähnliches Verhalten in Abhängigkeit von der Sequenzidentität zum Lerndatensatz (siehe Abbildung 46 auf Seite 176) und in Abhängigkeit vom BLAST Score (siehe Abbildung 47 auf Seite 178, für kleine BLAST Scores) aufweisen, dafür, dass die Auswirkungen dieser Effekte klein sind. Zwar kann im Fall des BLAST Scores ein kleiner positiver Trend des Q_3 für kleine BLAST Scores beobachtet werden, der dafür sprechen könnte, dass die einzelnen Programme Proteine mit höherer Homologie zum Lerndatensatz besser erkennen. Jedoch ist dieser Trend einerseits bei allen sechs Programmen gleichermaßen vorhanden, andererseits aufgrund der äußerst ungleichmäßigen Verteilung der BLAST Scores kein verlässliches Vergleichsmaß, sodass der positive Trend bei kleinen BLAST Scores der obigen Annahme nicht widerspricht.

Eine weitere Frage nach der Vergleichbarkeit der einzelnen Vorhersageprogramme ergibt sich aus deren Architekturen. Die meisten Programme im Vergleichstest besitzen zwei Stufen, während SPARROW und *SPARROW jeweils drei Stufen besitzen. Dies wirft die Frage auf, inwiefern SPARROW und *SPARROW durch deren Architektur den anderen Programmen gegenüber im Vorteil sind. Zur Klärung dieses Punktes soll Abschnitt 4.2.4 herangezogen werden, laut dem jede Komponentenfunktion des vektorwertigen Klassifikators in Form einer linearen Funktion $f_k(\vec{x},\vec{\theta}_k)=\vec{W}_k^T\vec{X}+b_k$ dargestellt werden kann. Diese lineare Darstellung der Komponentenfunktion entspricht der Gleichung eines linearen Neurons, also eines Perceptrons mit einer linearen Aktivierungsfunktion (siehe Abschnitt 3.5.4.1). Damit kann der vektorwertige Klassifikator als eine Schicht linearer Neuronen aufgefasst werden, was wiederum bedeutet, dass die erste und zweite Stufe von *SPARROW jeweils einer Schicht linearer Neuronen entspricht. Diese beiden Stufen zusammengenommen entsprächen somit einem MLP mit einer versteckten Schicht, die durch die ersten Stufe *SPARROWs realisiert wäre, und einer Ausgabeschicht, die sich aus der zweiten Stufe *SPARROWs ergäbe. Zur Umsetzung eines solchen MLPs müsste die versteckte Schicht ebensoviele Replikationen des vektorwertigen Klassifikators in der ersten Stufe *SPARROWs enthalten, wie sich Residuen im Fenster befinden, dass von der zweiten Stufe *SPARROWs bearbeitet wird. Dieser Aufbau besitzt eine gewisse Ähnlichkeit zu

den in Abschnitt 3.5.4.4 beschriebenen TDNNs. In diesem Fall würde jede dieser Kopien für ein Residuum im Fenster der zweiten Stufe verantwortlich sein, sodass die Ausgabeschicht mit den korrekten Daten versorgt werden würde. Durch den Versatz der einzelnen Kopien zu einander würde das resultierende MLP effektiv ein größeres Sequenzfenster besitzen als der vektorwertige Klassifikator der ersten Stufe. Dieses Fenster würde aus der Vereinigung der sich überlappenden Sequenzfenster der einzelnen Kopien ergeben. Somit zeigt sich, dass die ersten beiden Stufen *SPARROWs zu einem MLP, oder auch TDNN, mit einer versteckten Schicht äquivalent sind, wodurch *SPARROW selbst von seiner Architektur her zu zwei hintereinander geschalteten MLPs äquivalent ist. Die gleiche Argumentation kann auch bei SPARROW verwendet werden, sodass beide Programme trotz ihrer augenscheinlich höheren Anzahl an Stufen zu Programmen mit zwei Stufen auf Grundlage neuronaler Netze architektonisch äquivalent sind.

Der Vergleich zwischen SPARROW und *SPARROW zeigt, dass *SPARROW eine Verbesserung gegenüber seinem Vorläufer darstellt. Es stellt sich nun die Frage, woher diese Verbesserung stammen mag. Eine Möglichkeit besteht darin, dass die vektorwertigen Klassifikatoren, die die ersten beiden Stufen von *SPARROW bilden, robust gegenüber Konfliktfällen und Fällen von Ungewissheit sind. Für den vektorwertigen Klassifikator lässt sich mit der Betrachtung aus Abschnitt 4.3.3.1 (insbesondere Abbildung 39 auf Seite 140) zeigen, dass Konfliktfälle im Fall des Dreiklassenproblems nur dann auftreten, wenn zwei der Skalarprodukte eines Projektionsvektors mit den Klassenvektoren identische Werte annehmen. Dies ergibt sich bei einer Projektion auf die Winkelhalbierende zwischen zwei Klassenvektoren, welche im Fall des Dreiklassenproblems einer Gerade im zweidimensionalen Raum entspricht und damit einen eindimensionalen Raum darstellt. Diese Tatsache ergibt sich aus der Anordnung der Klassenvektoren, aus der resultiert, dass das Skalarprodukt des Projektionsvektors mit einem Klassenvektor nicht von den Skalarprodukten mit den übrigen Klassenvektoren unabhängig ist. Dadurch sind beim vektorwertigen Klassifikator Fälle von Ungewissheit, bei denen alle drei Skalarprodukte gleich sind, nur dann möglich, wenn eine Projektion in den Koordinatenursprung stattfindet. Dieser definiert einen dimensionslosen Raum. In SPARROWs Fall sind Konfliktfälle in einer der ersten beiden Stufen dadurch charakterisiert, dass zwei der drei Klassifikatoren zweier Klassen, die diese Stufen ausmachen, identische Werte liefern. Da die Werte dieser drei Klassifikatoren bei SPARROW unabhängig voneinander sind und als Koordinaten in einem dreidi-

mensionalen Raum aufgefasst werden können, bilden die Konfliktfälle eine Ebene im dreidimensionalen Raum. Mögliche Konfliktfälle sind somit in einem zweidimensionalen Raum enthalten. Auf ähnliche Weise kann gezeigt werden, dass Fälle von Ungewissheit, bei denen alle drei Klassifikatoren den gleichen Wert liefern, auf einer Geraden im dreidimensionalen Raum liegen und somit einen eindimensionalen Raum bilden. Somit besitzen beide Teilräume bei SPARROW mehr Dimensionen als die betreffenden Räume beim vektorwertigen Klassifikator, weshalb Konflikte wie Ungewissheit bei SPARROW mit größerer Wahrscheinlichkeit auftreten können als bei *SPARROW. Dies ist eine der möglichen Erklärungen für das bessere Verhalten *SPARROWs. Hinzu kommt die Tatsache, dass im vektorwertigen Klassifikator alle Klassen in jeder Komponentenfunktion einheitlich behandelt werden, wodurch jede dieser Komponentenfunktionen das zu lösende Mehrklassenproblem bearbeitet, während die Klassifikatoren bei SPARROW lediglich Zweiklassenprobleme lösen, sodass die einzelnen Klassen von unterschiedlichen Klassifikatoren unterschiedlich behandelt werden. Dadurch kann der vektorwertige Klassifikator, als Lernmaschine endlicher Kapazität, innerhalb seiner Möglichkeiten die bestmögliche Lösung für das zu lösende Dreiklassenproblem der Sekundärstrukturvorhersage finden, während SPARROW durch die Lösung der einfacheren Zweiklassenprobleme die Kapazität seiner Parameter nicht optimal einsetzt. Bei SPARROW lösen die einzelnen Klassifikatoren die Zweiklassenprobleme zwar sehr gut, ihnen fehlt jedoch der Zusammenhang aller Klassen, wie ihn der vektorwertige Klassifikator oder die MLPs besitzen, um auf dieser Grundlage das Mehrklassenproblem in optimaler weise zu lösen.

Weiterhin scheint ein Vorteil von *SPARROW darin zu liegen, dass die acht DSSP-Klassen erst in der letzten Stufe zu den drei vorherzusagenden Klassen reduziert werden. Wie bereits in Abschnitt 6.2 diskutiert, ist eine solche Vorgehensweise bei der Lösung des Dreiklassenproblems von Vorteil und kann somit durchaus einen großen Einfluss auf die Verbesserung *SPARROWs gegenüber SPARROW haben. Ein weiterer Vorteil liegt darin, dass *SPARROW aus einer Kombination dreier Vorhersageschemata besteht, die in Tabelle 11 auf Seite 173 aufgeführt sind. Eine solche Kombination ist bei SPARROW, das Aufgrund seiner Struktur auf ein Vorhersageschema beschränkt ist, nicht möglich. Nun könnte angebracht werden, dass gerade diese Kombination mehrerer Vorhersageschemata die Verbesserung *SPARROWs gegenüber SPARROW erklären könnte, da Kombinationen mehrerer Methoden grundsätzlich eine Verbesserung gegenüber den ursprünglichen

Methoden darstellen. Jedoch weisen bereits die einzelnen Vorhersageschemata eine Verbesserung gegenüber SPARROW auf, die zwar geringer ausfällt als bei der Kombination, aber dennoch groß genug ist, um den Unterschied beider Methoden zu zeigen.

Schließlich sollte erwähnt werden, dass *SPARROW ein geringfügig größeres Sequenzfenster als SPARROW besitzt, was auch zum besseren Verhalten *SPARROWs gegenüber SPARROW beitragen könnte. Inwiefern dies jedoch bei dem Unterschied zwischen SPARROW und *SPARROW ausschlaggebend ist, ist eine Frage die weiterer Klärung bedarf aber hier nicht erörtert wird. Alle genannten Argumente sprechen jedoch dafür, dass vektorwertige Klassifikatoren und deren Anwendung in *SPARROW einen konzeptionellen Vorteil gegenüber SPARROW besitzen.

6.7. *SPARROWs Rolle in kombinierten Verfahren

Wie Abschnitt 5.4.3 zeigt, können durch die Kombination von *SPARROW mit einem anderen Vorhersageprogramm verbesserte Verfahren zur Vorhersage der Sekundärstruktur von Proteinen erzeugt werden. Die beiden untersuchten Kombinationen SPARROW + *SPARROW sowie PSIPRED + *SPARROW stellen jeweils eine Verbesserung gegenüber den einzelnen Programmen dar, die zudem nach der in Abschnitt 6.6 verwendeten Argumentation mittels der Standardabweichung des Q_3 in der *Prediction* der Kreuzvalidierung als signifikant angesehen werden kann. Mit der gleichen Argumentation kann man vermuten, dass die Unterschiede zwischen PSIPRED und SPARROW + *SPARROW, die bezüglich $Q_3^{(\mathrm{Res})}$ und $Q_3^{(\mathrm{Prot})}$ bestehen, nicht signifikant sind. Für einige der dargestellten Diagramme, insbesondere in den Abbildungen 50, 51 und 53, fallen die Graphen von SPARROW + *SPARROW und PSIPRED an vielen stellen zusammen. Im Fall von Abbildung 52 liegt der Graph von SPARROW + *SPARROW für alle Proteinlängen, jedoch insbesondere für Proteine bis zu einer Länge von 240, oberhalb des Graphen von PSIPRED. Dies spricht dafür, dass SPARROW + *SPARROW bei kurzen Proteinen im Mittel ein besseres Verhalten besitzt als PSIPRED. Über den gesamten Datensatz gemittelt ist der Unterschied zwischen SPARROW + *SPARROW und PSIPRED jedoch, wie bereits erwähnt, zu klein um signifikant zu sein. Insgesamt sprechend die gewonnenen Ergebnisse dafür, dass diese beiden Verfahren äquivalent sind. Die Ähnlichkeit der Graphen *SPARROW und SPARROW + *SPARROW zeigt zudem, dass *SPARROW in dieser Kombination mehr Einfluss besitzt, was einerseits auf die höhere Gewichtung im

Ensemble Averaging zurückgeht andererseits auch die bessere Vorhersagequalität von *SPARROW gegenüber SPARROW zur Ursache hat. Im Fall der zweiten Kombination hingegen, also PSIPRED + *SPARROW, besitzt PSIPRED trotz der gleichen Gewichtung beider Vorhersageprogramme mehr Einfluss als *SPARROW, was sich auch hier in der hohen qualitativen Ähnlichkeit im Verlauf der in Abschnitt 5.4.3 dargestellten Graphen von PSIPRED + *SPARROW und PSIPRED äußert. Der Grund hierfür ist das etwas bessere Vorhersageverhalten von PSIPRED im Vergleich zu *SPARROW. Weiterhin kann für diese Kombination anhand aller Graphen in Abschnitt 5.4.3 wie auch anhand von Tabelle 13 auf Seite 183 festgestellt werden, dass diese Kombination den übrigen Verfahren im Vergleichstest deutlich überlegen ist. Dabei erscheint es bei Betrachtung der Graphen des mittleren Q_3-Index pro Protein in Abhängigkeit von der Proteinlänge, dass PSIPRED + *SPARROW davon profitiert, dass *SPARROW für kurze Proteine ein besseres Verhalten als PSIPRED zeigt. Weiterhin ist auffällig, dass PSIPRED + *SPARROW gegenüber PSIPRED eine größere Verbesserung darstellt als es bei SPARROW + *SPARROW gegenüber *SPARROW der Fall ist. Dies scheint sich aus der kleineren Schnittmenge falscher Vorhersagen bei PSIPRED und *SPARROW gegenüber der entsprechenden Schnittmenge bei SPARROW und *SPARROW zu ergeben. Dies bestätigt die in Abschnitt 5.4.3 aufgestellte Annahme, dass die potentielle Verbesserung eines kombinierten Verfahrens umgekehrt proportional zur Größe der Schnittmenge der falschen Vorhersagen der ursprünglichen Programme ist. Des weiteren kann auch der konzeptionelle Unterschied beider Programme für diese Verbesserung verantwortlich sein, der zwischen PSIPRED und *SPARROW größer ist, als zwischen SPARROW und *SPARROW und für die Wichtigkeit *SPARROWs für das gute Vorhersageverhalten sprechen würde.

Zusammenfassend geht aus dem Verhalten beider Kombinationen gegenüber den ursprünglichen Programmen hervor, dass *SPARROW aufgrund seiner Einzigartigkeit, die sich aus der Verwendung des neuartigen Konzepts des vektorwertigen Klassifikators ergibt, gut für die Umsetzung kombinierter Verfahren geeignet ist. Solche Verfahren können wiederum, wie Abschnitt 5.4.3 im Fall von PSIPRED + *SPARROW zeigt, zu einer signifikanten Verbesserung der Qualität der vorhergesagten Sekundärstruktur führen, was nicht zuletzt auch an dem vergleichsweise guten Vorhersageverhalten von *SPARROW liegt. Diese Möglichkeit zur Kombination bildet damit eine weitere Stärke von *SPARROW.

7. Ausblick

Wie aus dem vorherige Kapitel hervorgeht, stellen der vektorwertige Klassifikator und *SPARROW ein vielversprechendes Konzept dar, um Mehrklassenprobleme im Allgemeinen und das Problem der Sekundärstrukturvorhersage im Speziellen zu lösen. Wie jedes neuartige Verfahren kann *SPARROW bei weitem nicht als ausgereift bezeichnet werden, jedoch bietet es andererseits, wie jedes neuartige Verfahren, auch viele Verbesserungsmöglichkeiten. Dieses Kapitel stellt eine Sammlung von Ideen zur Verbesserung oder Erweiterung des Konzepts von *SPARROW wie auch zu weiteren Entwicklungen auf dem Gebiet der Sekundärstrukturvorhersage von Proteinen dar. Dabei beschreibt Abschnitt 7.1 eine direkte Modifikationen von *SPARROW, während Abschnitt 7.2 die Erweiterung der Vorhersagen von *SPARROW um eine Konfidenz in ihre Korrektheit behandelt. In Abschnitt 7.3 wiederum soll eine weitere Anwendungsmöglichkeit des Prinzips des vektorwertigen Klassifikators diskutiert werden. Schließlich beinhaltet Abschnitt 7.4 einige generelle Ideen zu weiteren Entwicklungen auf dem Gebiet der Sekundärstrukturvorhersage von Proteinen.

7.1. Verwendung von Gradientenverfahren

Das Lernverfahren für den vektorwertigen Klassifikator optimiert dessen Parameter, indem die exakte Lösung einer Anzahl linearer Gleichungssysteme bestimmt wird (siehe Abschnitt 4.2.5). Dies stellt einerseits einen sehr großen Vorteil des Verfahrens dar, da durch die exakte Lösung das Optimum der Fehlerfunktion möglichst genau erreicht wird, sodass bei einer genügend hohen Menge an Lerndaten der optimale vektorwertige Klassifikator für das jeweilige Klassifikationsproblem erzeugt werden sollte. Jedoch ist in der Regel nicht in allen Fällen eine genügend hohe Datenmenge vorhanden, um bei gegebener Parameterzahl Überanpassung vollkommen zu vermeiden. Dadurch kann das Lernverfahren einen vektorwertigen Klassifikator liefern, der optimal zur Vorhersage des Lerndatensatzes geeignet ist, jedoch nicht notwendigerweise auch für unbekannte Daten optimal ist. Zudem hat das Verfahren auch den großen Nachteil, dass die Koeffizientenmatrizen der linearen Gleichungssysteme sehr viel Speicherplatz benötigen, der sowohl bei der physischen Speicherung der Matrizen als auch bei der Lösung eines linearen Gleichungssystems vorhanden sein muss.

Eine Möglichkeit, sowohl das Problem der Überanpassung als auch des hohen Speicherbedarfs zu umgehen stellen iterative Lernverfahren dar, wie sie zur Optimierung nichtlinearer Probleme verwendet werden. Ein Beispiel eines solchen Verfahrens, dass für die Anwendung auf MLPs spezialisiert ist, stellt der Back-Propagation-Algorithmus (siehe Abschnitt 3.5.4.3) dar. Es existiert eine Reihe weiterer Verfahren wie zum Beispiel *Resilient Propagation* (kurz *Rprop*, [93]), das *Verfahren der konjugierten Gradienten* (engl. *conjugate gradient*, [94]) oder der einfache *Gradientenabstieg* (engl. *gradient descent*, [94]), die verwendet werden könnten, um eine Lösung der linearen Gleichungssysteme zu finden. Diese Verfahren haben einerseits den Vorteil, dass sie einen deutlich geringeren Speicherbedarf haben, als der aktuell verwendete Lernalgorithmus für den vektorwertigen Klassifikator: Während die notwendigen Koeffizientenmatrizen einige Gigabyte an Speicherplatz belegen, benötigt ein auf Gradienten basiertes Verfahren hauptsächlich so viel Speicher, wie der Lerndatensatz einnimmt, was in der Regel maximal einige hundert Megabyte erfordert. Zudem können gradientenbasierte Verfahren mittels des sogenannten *Early Stopping* [45] dazu gebracht werden, mit Hilfe eines geeigneten Kriteriums das Lernen abzubrechen bevor sich Überanpassung bemerkbar macht. Andererseits ist es auch möglich, dass das Optimieren der Fehlerfunktion mittels eines derart modifizierten Gradientenverfahrens weniger Zeit benötigt, als die Berechnung der Koeffizientenmatrix aus dem Lerndatensatz und die anschließende Lösung der linearen Gleichungssysteme.

Es könnte somit sinnvoll sein zu untersuchen, ob sich die Verwendung eines Gradientenverfahrens zur Optimierung des vektorwertigen Klassifikators positiv auf das Verhalten *SPARROWs auswirkt. Dabei könnte ein Vergleich zwischen der exakten Lösung, wie sie das aktuelle Lernverfahren berechnet, und der durch das Gradientenverfahren erzeugten Lösung gezogen werden, wobei neben den Unterschieden im Verhalten auch die Zeit berücksichtigt werden sollte, die die einzelnen Verfahren zur Berechnung der Parameter benötigen. Zudem kann verglichen werden, wie gut ein gradientenbasiertes Lernverfahren mit *Early Stopping* die Überanpassung von *SPARROW im Vergleich zur Regularisierung der exakten Lösung der linearen Gleichungssysteme beseitigt. Hierbei wäre eine Kreuzvalidierung zur Durchführung der Vergleiche ratsam. Schließlich würde die Verwendung eines auf Gradienten basierenden Verfahrens Komponentenfunktionen mit höherer Nichtlinearität erlauben, was unter Umständen die Vorhersagequalität positiv beeinflussen könnte.

7.2. Konfidenzfunktion für die Vorhersagen von *SPARROW

In der Anwendung ist für den Benutzer, neben der eigentlichen Vorhersage der Sekundärstruktur auch die Information von Interesse, wie wahrscheinlich es ist, dass diese Vorhersage korrekt ist. Üblicherweise wird dabei jedem Residuum ein Konfidenzmaß zugeordnet, dass aussagt, inwiefern der Klassifikation des betreffenden Residuums Vertrauen zu schenken ist. Standardmäßig wird dabei, ausgehend von einer Klassifikation gemäß der Entscheidungsregel der maximalen Konfidenz (siehe Abschnitt 3.4.2), die Differenz der beiden größten Komponenten des jeweiligen Ausgangsvektors zur Berechnung der Konfidenz in die Richtigkeit der Klassifikation eines Residuums verwendet. Die diesem Kriterium zugrunde liegende Idee ist, dass die Klassifikation umso unwahrscheinlicher korrekt ist, je näher die zweitgrößte Komponente des Ausgabevektors an dem Wert der größten Komponente liegt, die für die Vorhersage benutzt wird. Dieses einfache Verfahren wird gleichermaßen von SPARROW [1] wie auch von PSIPRED [30] verwendet und liefert dem Nutzer so einen Anhaltspunkt, wie sicher die Vorhersage der einzelnen Residuen ist.

In Zusammenhang mit der Entwicklung von *SPARROW wurde der Versuch unternommen, eine verbesserte Konfidenzfunktion mit Hilfe eines geeigneten Lernverfahrens zu konstruieren. Hierzu wurde *SPARROW selbst als Zufallsprozess angesehen, der in Form der Ausgangsvektoren eine Zufallsvariable $\vec{Y}$ mit einer unbekannten aber festen Verteilung $p(\vec{Y}=\vec{y})$ liefert. Die anhand dieser Zufallsvariable vorgenommene Klassifikation kann wiederum entweder richtig oder falsch sein, was durch eine unbekannte Wahrscheinlichkeitsverteilung $p(c=1|\vec{Y}=\vec{y})$ beschrieben wird. Hierbei soll c eine binäre Variable sein, die angibt ob die auf Grundlage von $\vec{y}$ gemachte Vorhersage korrekt ist $(c=1)$ oder nicht $(c=0)$. Auf diese Weise kann die gesuchte Konfidenzfunktion $f(\vec{y})$ als ein Klassifikator zweier Klassen aufgefasst werden, der die Ausgabevektoren $\vec{y}$ einer der zwei Klassen „Vorhersage korrekt“ und „Vorhersage falsch“ zuzuordnen versucht. Dabei sollte die Funktion auf das Intervall zwischen 0 und 1 beschränkt sein, sodass die von ihr gelieferten Werte als Wahrscheinlichkeit der korrekten Vorhersage aufgefasst werden können.

Mit diesem Hintergrund wurden zwei Konzepte für die Konfidenzfunktion erprobt. Das erste Konzept bestand in der Verwendung eines einfachen linearen Klassifikators $f(\vec{y})=\vec{w}\cdot\vec{y}+b$ wie er durch Fishers lineare Diskriminanzfunktion realisiert wird ([41], siehe Abschnitt 3.5.2). Die Parameter $\vec{w}$ und b wurden optimiert, indem der mittlere

quadratische Fehler auf einem Lerndatensatz minimiert wurde. Bei dieser Optimierung wurden die Sollwerte der Konfidenzfunktion derart gewählt, dass korrekt klassifizierte Residuen einen Wert möglichst nahe bei 1 haben sollten, während falsche Vorhersagen nahe bei 0 liegen sollten.

Als zweite Möglichkeit wurde die als *logistische Regression* (auch *Logit-Modell*, [95]) bezeichnete Methode verwendet, um eine geeignete Konfidenzfunktion zu konstruieren. Dabei handelt es sich um eine in den Wirtschaftswissenschaften verwendete Methode zur Berechnung der Wahrscheinlichkeit binärer Ereignisse anhand gegebener Daten, für die Daniel McFadden und James Heckman im Jahre 2000 den Nobelpreis für Wirtschaftswissenschaften erhielten. Das verwendete Modell für die Konfidenzfunktion besteht aus einer logistischen Funktion der Form:

$$f(\vec{y}) = \frac{1}{1+e^{-z}} \text{ , mit } z = \vec{w} \cdot \vec{y} + b$$

Diese ähnelt in groben Zügen der in Abschnitt 4.3.3.1 vorgestellten Wahrscheinlichkeitsfunktion zur Verbesserung der Vorhersagequalität der dritten Stufe *SPARROWs. Bei der logistischen Regression wird die obige logistische Funktion mit Hilfe der Maximum-Likelihood-Methode (siehe Abschnitt 3.2.2) optimiert, indem auf dem Lerndatensatz D^{lern} mit $D^{lern} = \{(\vec{y}_i, c_i) | 1 \le i \le N\}$ die Fehlerfunktion

$$E(\vec{w}, b) = \sum_{i=1}^{N} c_i \cdot \log\left(f(\vec{y})\right) \cdot (1 - c_i) \cdot \log\left(1 - f(\vec{y})\right)$$

maximiert wird. Diese Fehlerfunktion ergibt sich aus der Maximum-Likelihood-Methode unter der Annahme, dass

$$p(c | \vec{y}, \vec{w}, b) = \prod_{i=1}^{N} p(c_i | \vec{y}_i) = \prod_{i=1}^{N} \left(p(c=1 | \vec{y}_i)\right)^{c_i} \cdot \left(1 - p(c=1 | \vec{y}_i)\right)^{1-c_i}$$

ist (siehe Abschnitt 3.2.2). Mit dieser Vorgehensweise ergibt sich eine Funktion, die garantiert auf das Intervall 0 bis 1 begrenzt ist (was für die lineare Funktion nicht ohne weiteres zutrifft) und deren Wert als Wahrscheinlichkeit der korrekten Klassifikation eines bestimmten Residuums mittels des Ausgangsvektors $\vec{y}$ interpretiert werden kann. Prinzipiell ist es zudem auch im Fall der logistischen Regression möglich, den mittleren quadratischen Fehler über D^{lern} zu minimieren.

Aus den bisher durchgeführten Experimenten mit der Konfidenzfunktion, sowohl in Form der linearen Funktion als auch beider möglicher Umsetzungen der logistischen Regression, konnten bisher keine schlüssigen Ergebnisse erzielt werden, sodass nicht bestimmt werden konnte welche Methode am besten zur Umsetzung dieser Funktion geeignet ist. Jedoch kann auf Grundlage der Experimente die Aussage getroffen werden, dass es bei allen Umsetzungen der Konfidenzfunktion wichtig ist, dass die Komponenten innerhalb des Ausgangsvektors von *SPARROW ihrer Größe nach sortiert werden, bevor sie in die Konfidenzfunktion eingesetzt werden. Durch eine solche Sortierung ihrer Komponenten werden die Ausgangsvektoren implizit in Beziehung mit der Entscheidungsregel gesetzt, wodurch das Zweiklassenproblem der Korrektheit einer Vorhersage verlässlicher gelöst werden kann. Durch dieses Vorgehen wird auch vollständig vom ursprünglichen Dreiklassenproblem, in dem jeder der Komponenten eine bestimmte Bedeutung zugeordnet wird, abstrahiert. Zur Überprüfung, inwiefern eine der in diesem Abschnitt beschriebenen, komplexeren Konfidenzfunktionen der einfachen Differenz der beiden größten Komponenten des Ausgabevektors von *SPARROW vorzuziehen ist, sind jedoch weitere Untersuchungen notwendig.

7.3. MLPs als vektorwertige Klassifikatoren

Üblicherweise besitzen MLPs, wie Abschnitt 3.5.4.2 beschreibt, genausoviele Neuronen in ihrer Ausgabeschicht, wie Klassen im zu lösenden Klassifikationsproblem vorhanden sind. Nach der Argumentation in Abschnitt 4.2.1, die zur Einführung des vektorwertigen Klassifikators verwendet wurde, sollte eine um ein Neuron kleinere Ausgabeschicht bereits ausreichen um das jeweilige Klassifikationsproblem zu lösen. Prinzipiell spräche also nichts dagegen, das Konzept des vektorwertigen Klassifikators auf MLPs zu übertragen, indem die Architektur und das Lernverfahren der MLPs angepasst wird. Der Back-Propagation-Algorithmus müsste dabei lediglich insofern modifiziert werden, als dass statt der bisher verwendeten Sollwerte (siehe Abschnitt 3.5.4.3) die Klassenvektoren des vektorwertigen Klassifikators verwendet werden müssten (siehe Abschnitt 4.2.2). Weitere Modifikationen sind nicht notwendig, da der Back-Propagation-Algorithmus, wie auch das Lernverfahren für den vektorwertigen Klassifikator den mittleren quadratischen Fehler minimieren, sodass keine Anpassung des Back-Propagation-Algorithmus an eine neue Fehlerfunktion notwendig wäre. Die notwendige Anpassung der Architektur des MLPs beschränkt sich auf

die bereits erwähnte Verringerung der Anzahl an Neuronen in der Ausgabeschicht wie auch eine Änderung der Entscheidungsregel von der Regel der maximalen Konfidenz hin zu der in Abschnitt 4.2.1 beschriebenen Entscheidungsregel für den vektorwertigen Klassifikator.

Ein auf diese einfache Weise modifiziertes MLP hätte gegenüber der in Abschnitt 4.2.3 vorgestellten Klassifikatorfunktion den Vorteil eines höheren Grads an Nichtlinearität, der unter Annahme nichtlinear trennbarer Daten zu einem besseren Vorhersageverhalten führen könnte. Auf diese Weise könnte also ein vektorwertiger Klassifikator erzeugt werden, der mit einer geringeren Anzahl an Parametern auskommt und somit weniger schnell Gefahr läuft, unter Überanpassung zu leiden. Es sollte untersucht werden, wie sich ein derartiger, durch ein MLP realisierter vektorwertiger Klassifikator im Vergleich zu dem in dieser Arbeit vorgestellten vektorwertigen Klassifikator einerseits und einem normalen MLP andererseits verhält.

Zudem wäre es interessant zu untersuchen, inwiefern ein vektorwertiger Klassifikator auf Grundlage eines MLPs die ersten beiden Stufen *SPARROWs ersetzen könnte. In Abschnitt 6.6 wurde diskutiert, dass die ersten beiden Stufen *SPARROWs zu einem MLP äquivalent sind, das aus linearen Neuronen besteht, ein größeres Sequenzfenster als die entsprechende Klassifikatorfunktion in der ersten Stufe besitzt und in der versteckten Schicht Kopien dieser Klassifikatorfunktion enthält. Durch die Verwendung nichtlinearer Neuronen könnte sich die Notwendigkeit der beiden letzten Anpassungen des MLPs erübrigen, sodass ein einfacher, durch ein MLP realisierter vektorwertiger Klassifikator ausreichen könnte, um die beiden ersten Stufen *SPARROWs zu ersetzen. Dadurch ginge jedoch einer der Vorteile *SPARROWs teilweise verloren, der in der Möglichkeit besteht, in den einzelnen Stufen unterschiedliche Klassifikationsprobleme zu lösen. Zwar wäre es immernoch möglich, in jeder der beiden Stufen (vektorwertiger Klassifikator auf Grundlage eines MLP sowie die ursprüngliche dritte Stufe als zweite Stufe) unterschiedliche Probleme zu lösen, jedoch geht durch den Verlust des dreistufigen Aufbaus in dieser Hinsicht ein Freiheitsgrad verloren. In welchem Verhältnis ein allein mit Hilfe von MLPs umgesetztes *SPARROW zur ursprünglichen Realisierung steht ist gegenwärtig ungeklärt und bedarf einiger weiterer Untersuchungen.

7.4. Alternative Repräsentation der Daten

Theoretisch sollte es, unter der Annahme der Korrektheit der zugrunde liegenden Daten, ausreichender Größe des Lerndatensatzes wie auch eines idealen Klassifikators, möglich sein, die Sekundärstruktur von Proteinen vollkommen korrekt vorherzusagen. Moderne Verfahren erreichen eine Genauigkeit von ca. 82% (oder einen Q_3 -Index von ca. 0,82) und sind somit zwar nahe am Optimum, jedoch scheint es immer schwieriger zu werden, weitere Verbesserungen durch die reine Modifikation der Vorhersageprogramme zu erzielen. Die letzten großen Verbesserungen in der Vorhersage der Sekundärstruktur innerhalb der letzten zwanzig Jahre wurden durch die Einführung von PSI-BLAST-Sequenzprofilen zur Darstellung der Eingangsdaten durch David Jones [30] erreicht. Zwar hat sich seitdem die Genauigkeit von Sekundärstrukturvorhersagen verbessert, jedoch kann man dies in erster Linie der zunehmenden Zahl bekannter Proteinstrukturen wie auch den immer größer werdenden Sequenzdatenbanken, mit deren Hilfe genauere Profile erzeugt werden können, zuschreiben [96]. Nun stellt sich die Frage, ob die Genauigkeit der Sekudärstrukturvorhersage mit genügend großen Struktur- und Sequenzdatenbanken auf 100% gebracht werden kann? Laut [96] und [97] lautet die Antwort auf diese Frage nein. Dort wird eine theoretische Grenze für die Genauigkeit der Sekundärstrukturvorhersage von Proteinen angegeben, die bei 88% liegt. Die Begründung dieser Grenze basiert einerseits darauf, dass für viele der Sekundärstrukturtypen, wie zum Beispiel isolierten β-Brücken oder β-Strängen, Wechselwirkungen zwischen sequentiell weit voneinander entfernten Residuen essentiell sind, die durch heutige Verfahren, die alle relativ kleine Ausschnitte der Sequenz zur Vorhersage verwenden, nicht erfasst werden können (siehe auch Abschnitt 6.5). Dies führt dazu, dass ein wichtiger Faktor in der Bildung dieser Sekundärstrukturtypen bei der Vorhersage ausgelassen wird, weshalb die Genauigkeit in der Vorhersage dieser Sekundärstrukturtypen nach oben hin beschränkt ist.

Weiterhin wird in [96] angemerkt, dass DSSP selbst, das in der Sekundärstrukturvorhersage in den meisten Fällen als Referenz verwendet wird, das Problem besitzt, dass es Wasserstoffbrücken, auf deren Grundlage die Zuweisung der Residuen zu den einzelnen Sekundärstrukturtypen erfolgt, über den Schwellenwert seiner Energiefunktion (siehe auch Abschnitt 2.3.3.3) zuweist. Durch diese harte Klassifikationsgrenze, die zudem vom Benutzer verändert werden kann, können Abweichungen zwischen der Klassifikation, wie sie DSSP vorgibt, und der zugrunde liegenden Kristallstruktur auftreten. Dies mag einer der

Gründe sein, weshalb DSSP beispielsweise dazu zu neigen scheint, π-Helices weniger oft vorherzusagen als diese tatsächlich vorkommen [98]. So wird vermutet, dass π-Helices stabiler sind als bisher angenommen und dadurch zehnmal häufiger vorkommen, als sie von DSSP und ähnlichen Programmen gefunden werden [98]. Sollte diese Annahme tatsächlich zutreffen, würde das bedeuten, dass die gegenwärtig verwendeten Strukturdaten in der Verteilung und Zuweisung der einzelnen Sekundärstrukturtypen mit einem gewissen Fehler behaftet sind.

Weiterhin wird in [99] angemerkt, dass die diskrete Zuweisung von Sekundärstrukturtypen, wie sie gegenwärtig üblich ist, implizit von rigiden Proteinstrukturen ausgeht und thermische Fluktuationen in der Struktur vollkommen außer acht lässt. Diese Fluktuationen führen zu einer gewissen Flexibilität der Proteinstruktur, wie auch der Verteilung von Sekundärstrukturen entlang der Sequenz, die für das Funktionieren von Proteinen unabdingbar sein kann [99]. Das Fehlen besagter Flexibilität in der Repräsentation von Proteinstrukturen kann laut [99] ein Grund dafür sein, dass heutige Programme zur Vorhersage der Sekundärstruktur erhebliche Probleme mit Strukturen haben, die mittels NMR-Spektroskopie (siehe Abschnitt 2.3.3.2) ermittelt wurden.

Eine weitere Problematik, auf die der Autor der vorliegenden Arbeit hinweisen möchte, liegt darin, dass die Randbedingungen, bei denen eine Sekundärstruktur bestimmt wurde, gegenwärtig bei der Sekundärstrukturvorhersage vollkommen außer acht gelassen werden. Wie in Abschnitt 2.4.1 bereits erläutert, existiert eine Reihe von Faktoren, die die Struktur von Proteinen wie auch deren Stabilität beeinflussen. Veränderung von Druck, Temperatur und pH-Wert können einen großen Einfluss auf diese Faktoren haben, wodurch die Struktur eines Proteins drastische Veränderungen erfahren kann (siehe Abschnitt 2.4.2). Dies bedeutet, dass die Struktur eines Proteins zwar in der Primärstruktur kodiert ist, jedoch auch eine Funktion der Umgebung ist, in der sich das Protein befindet. Die Kristallographie bestimmt üblicherweise Proteinstrukturen unter spezifischen Bedingungen wie sie für die Ausbildung eines Kristalls günstig sind, was nicht notwendig mit den physiologischen Bedingungen in einer lebenden Zelle übereinstimmt. Da die Lebensbedingungen der Organismen auf der Erde, wie auch die Bedingungen in unterschiedlichen Organen eines einzelnen Organismus, teilweise stark variieren können, können die Randbedingungen in denen sich nativen Proteinstrukturen bilden, unterschiedlich sein. Aus diesem Grund kann Beispielsweise das menschliche Hämoglobin schwerlich unter gleichen Bedingungen untersucht

werden wie Proteine, die in heißen Quellen lebenden Bakterien entstammen, da ersteres bei den Temperaturen, die in heißen Quellen vorherrschen denaturieren würde und letzteres bei menschlicher Körpertemperatur unter Umständen von der nativen Struktur abweichen könnte. Zwar kann argumentiert werden, dass die Anforderungen, die die Lebensbedingungen eines Organismus an seine Proteine stellen, implizit auch in der Sequenz enthalten sein sollten, beispielsweise im Vorhandensein von Cysteinen zur Ausbildung von Disulfidbrücken (siehe Abschnitt 2.4.1.4). Jedoch kann damit nicht ausgeschlossen werden, dass beispielsweise ein Protein, das bei Zimmertemperatur stabil ist, und eines, das bei hohen Temperaturen stabil ist, in einem großen, zusammenhängenden Teil ihrer Sequenzen übereinstimmen, jedoch auf diesem im jeweils nativen Zustand sehr unterschiedliche Sekundärstrukturen aufweisen. Aus der Sicht eines auf Sequenzfenstern basierenden Vorhersageprogramms würde für ein solches Sequenzfenster mit gleicher Sequenz aber unterschiedlicher Sekundärstruktur einen Widerspruch darstellen, was dazu führen könnte, dass das betreffende Sequenzstück in beiden Fällen falsch vorhergesagt wird. Inwiefern dieses Problem von PSI-BLAST-Profilen umgangen wird, die positionsspezifisch sind und dadurch für beide Sequenzstücke je nach Position recht unterschiedlich sein können, ist gegenwärtig unklar und sollte entsprechend untersucht werden. Jedoch existieren auch Proteine, die in unterschiedlichen Teilen eines Organismus leicht abweichende Strukturen aufweisen. So besitzt das Hämoglobin beispielsweise in der Blutbahn und in der Lunge leicht abweichende Tertiärstrukturen, was durch den Unterschiedlichen pH-Wert in diesen beiden Teilen des Körpers bedingt ist und für die Funktion des Hämoglobins im Körper eine wichtige Rolle spielt. Trotz der identischen Sequenz und der identischen Darstellung als Profil, könnten in diesem Fall also auch zwei geringfügig abweichende Sekundärstrukturen existieren, die eine gewisse Unschärfe in die Korrelation von Sequenz und Struktur bringen. Wenn also die Struktur eines Proteins eine Funktion von Sequenz und Umgebungsbedingungen ist, so ist es ungünstig, den Einfluss, den die Umgebung auf die Bildung von Sekundärstrukturen besitzt, zu vernachlässigen. Ähnliches gilt für das Vorhandensein von Kofaktoren, wie zum Beispiel Metallatomen wie dem Eisen im Hämoglobin, die Auswirkungen auf die Struktur eines Proteins haben können, jedoch gegenwärtig vollkommen vernachlässigt werden.

Die obigen Argumente würden somit dafür sprechen, dass die Repräsentation der Daten, also die Repräsentation eines Proteins anhand der Sequenz (oder des Sequenzprofils) al-

lein, wie auch der Ausgangsdaten in Form der recht strikten Klassifikation, wie sie durch DSSP oder STRIDE vorgegeben wird, die Mechanismen zur Ausbildung von Sekundärstrukturen nicht vollkommen erfassen kann. Dies kann wiederum dazu führen, dass die Korrelation von Eingangs- und Ausgangsdaten mit einem Rauschen behaftet ist, das durch Methoden zur Sekundärstrukturvorhersage, wie sie heute existieren, nicht beseitigt werden kann und dadurch zu einer endlichen Genauigkeit der Vorhersage führt, wie sie in [96] beschrieben wird. Was die Repräsentation der Ausgangsdaten angeht, wurden in den letzten Jahren Anstrengungen unternommen eine neue, verbesserte Repräsentation der Sekundärstruktur zu finden. So wurde daran gearbeitet eine neue Methode zur Erkennung von π-Helices zu finden, die laut [98] gegenüber DSSP und ähnlichen Programmen eine Verbesserung darstellt. Des weiteren wurden Versuche mit Strukturprofilen zur kontinuierlichen Repräsentation der Ausgangsdaten durchgeführt ([22], [99], [100]). Diese enthalten für jedes Residuum acht Einträge, von denen jeder einer der acht DSSP-Klassen entspricht und einen kontinuierlichen Wert enthält, der die Wahrscheinlichkeit für die Zugehörigkeit des jeweiligen Residuums zu der entsprechenden Klasse beschreibt. Die bisher verwendete Repräsentation der Ausgangsdaten in Form einer diskreten Klassifikation, wie sie von DSSP und ähnlichen Programmen erzeugt wird, sollte durch diese Strukturprofile ersetzt werden, um beispielsweise eine detailliertere Darstellung der Sekundärstruktur zu erreichen und ihre bereits beschriebenen thermischen Fluktuationen zu modellieren. Dabei wird die Berechnung dieser Strukturprofile unterschiedlich gehandhabt. So werden in [22] die Strukturen der Proteine, die bei der Erzeugung von Sequenzprofilen in das multiple Sequenzalignment eingehen (siehe Abschnitt 4.1.3.1), verwendet um ein Strukturprofil in Form einer mittleren Sekundärstruktur zu bilden. Dieses Strukturprofil wird in [22] statt der üblicherweise verwendeten diskreten Klassifikation mit dem Sequenzprofil korreliert. Bei DSSPcont ([99], [100]) hingegen, ergeben sich die Einträge des Strukturprofils aus der Mittelung von zehn verschiedenen Ausgaben von DSSP, die für unterschiedliche Schwellenwerte der Energiefunktion zur Erkennung von Wasserstoffbrücken erzeugt wurden. Die Ergebnisse mit diesen beiden Ansätzen Fallen unterschiedlich aus: Im ersten Fall zeigt die Verwendung von Strukturprofilen für die Ausgangsdaten eines Vorhersageprogramms eine Verschlechterung der Vorhersagequalität, im Fall von DSSPcont scheinen die Ergebnisse für eine Verbesserung der Korrelation von Sequenz und Struktur, insbesondere bei Strukturen auf Grundlage von NMR-Spektroskopie, zu sprechen, was auf die korrekte Abbildung

thermischer Fluktuationen zurückgeführt wird. Die unterschiedlichen Ergebnisse zeigen, dass auf diesem Gebiet weitere Untersuchungen notwendig sind bevor an eine Ersetzung der diskreten Darstellung der Sekundärstruktur, wie sie in Form der DSSP-Klassen vorliegt, durch eine kontinuierliche gedacht werden kann. Jedoch zeigen sie auch, dass diese Herangehensweise ein gewisses Potential besitzt. Verfahren, die die Umgebungsbedingungen, in denen ein Protein sich befindet, oder die enthaltenen Kofaktoren einbeziehen, sind dem Verfasser der vorliegenden Arbeit nicht bekannt. Jedoch könnten die entsprechenden Informationen von den einzelnen Vorhersageprogrammen neben dem Sequenzprofil als zusätzliche Eingangsdaten verwendet werden. Inwiefern dies zu einer Verbesserung der Vorhersagen führt, bedarf jedoch weiterer Untersuchung.

Wie die Argumentation in diesem Abschnitt zeigt, ist es unwahrscheinlich, dass durch die gegenwärtige Repräsentation und Handhabung der Daten eine exakte Vorhersage von Proteinsekundärstrukturen erreicht werden kann. Vielmehr erscheint es wahrscheinlicher, dass eine obere Schranke für die Genauigkeit der Vorhersagen existiert, wie sie in [96] beschrieben ist. Sofern dies der Fall ist, ist dafür nicht etwa die endliche Kapazität der verwendeten Lernmaschinen verantwortlich, sondern vielmehr die verwendete Datenrepräsentation und -handhabung. Dies wiederum bedeutet, dass es unabdingbar ist neue Datenrepräsentationen zu untersuchen, um die Qualität von Sekundärstrukturvorhersagen zu verbessern. Als Beispiel dafür, wie stark sich die Verbesserung der Datendarstellung auf die Vorhersagequalität auswirkt, sei abermals die Einführung von Sequenzprofilen erwähnt, durch die ein großer Sprung in der Vorhersagequalität erreicht wurde. Diese sollten jedoch nicht als das Ende der Entwicklung geeigneter Datenrepräsentationen angesehen werden, sondern als einer der notwendigen Schritte zu einer korrekten Vorhersage der Sekundärstruktur von Proteinen. Veränderungen an den Vorhersagemechanismen selbst, wie auch die Vergrößerung von Sequenz- und Strukturdatenbanken sollten zwar wie bereits erwähnt zu Verbesserungen der Vorhersagequalität führen, jedoch ist eine vollständige Korrektheit der Vorhersagen nicht möglich, solange die verwendeten Daten und deren Korrelation inkonsistent sind (wie in [99] angemerkt wird, machen sich bei dem gegenwärtig erreichten Niveau an Vorhersagegenauigkeit, Inkonsistenzen in der Zuweisung der Sekundärstruktur durch DSSP und ähnliche Programme immer stärker bemerkbar). Vielmehr kann davon ausgegangen werden, dass bei Beibehaltung der gegenwärtigen Datendarstellung die erzielten Verbesserungen der Vorhersagequalität mit der Zeit immer kleiner ausfallen wer-

den. Dies ergibt sich ausgehend von dem in [96] beschriebenen Grenzwert der Vorhersagequalität und einem angenommenen asymptotischen Verhalten nahe dieses Grenzwertes. Aus diesem Grund wird der Übergang von dem Klassifikationsproblem, wie es von *SPARROW gegenwärtig gelöst wird, zu einem Regressionsproblem zur Vorhersage von Strukturprofilen in Betracht gezogen. Weiterhin wird über eine Anpassung der Eingangsdaten nachgedacht, um die Umgebung, in der sich ein Protein befindet, mit in die Vorhersage einfließen zu lassen. Diese Überlegungen sind gegenwärtig jedoch rein theoretischer Natur und bedürfen noch einiger Ausarbeitung, bevor sie in die Praxis umgesetzt werden können.

8. Zusammenfassung

Die Sekundärstrukturvorhersage von Proteinen als erster Schritt zur Aufklärung ihrer 3D-Strukturen und Funktionen ist eine wichtige Fragestellung der modernen Molekularbiologie. In dieser Arbeit wird mit dem Programm *SPARROW eine neue Methode zur Vorhersage der Sekundärstruktur von Proteinen vorgestellt, die auf einem neuartigen Lernverfahren basiert. Dieses als vektorwertiger Klassifikator bezeichnete Verfahren verwendet eine Klassifikatorfunktion um hochdimensionale Eingangsvektoren in einen niederdimensionalen Ausgangsraum zu projizieren um mit Hilfe der Orientierung dieser Ausgangsvektoren die Sekundärstrukturen zu klassifizieren. Um eine möglichst hohe Genauigkeit dieser Klassifikationen zu gewährleisten wird die Klassifikatorfunktion mittels empirischer Risikominimierung, einem Verfahren der statistischen Lerntheorie, optimiert.

Das *SPARROW zugrunde liegende Verfahren besteht aus drei Stufen, von denen die ersten beiden vektorwertige Klassifikatoren verwenden, während die dritte aus einem künstlichen neuronalen Netz besteht. Die erste Stufe korreliert die Sekundärstruktur von Residuen, die durch Sequenzausschnitte in Form von PSI-BLAST-Profilen ([78], [79]) repräsentiert werden, mit der Primärstruktur. Dabei wird die von DSSP [13] vorhergesagte Sekundärstruktur als Referenz verwendet. Die Vorhersage der ersten Stufe wird von der zweiten Stufe weiterverarbeitet, die damit eine Struktur-Struktur-Korrelation darstellt. Die dritte Stufe stellt eine weitere Struktur-Struktur-Korrelation dar, die aufgrund der nichtlinearen Natur des verwendeten neuronalen Netzes Korrelationen höherer Ordnung in die Vorhersage einbezieht. Durch diesen hierarchischen Aufbau verbessert jede Stufe effektiv die Vorhersage der vorherigen Stufe. Zudem macht es diese Architektur von *SPARROW möglich, über eine Änderung der Anzahl der berücksichtigten Klassen, in den einzelnen Stufen Einfluss auf den Informationsfluss durch das System zu nehmen.

Tests auf dem ASTRAL40-Datensatz ([90], [91], [92]) zeigen, dass *SPARROW bei der Vorhersage dreier Sekundärstrukturklassen (Helices, β-Stränge und strukturlose Regionen) eine Genauigkeit von 81,53% erreicht und somit mit existierenden Vorhersageprogrammen vergleichbar ist. Zudem zeigt dieser Wert, dass *SPARROW eine Verbesserung von ungefähr 0,5% gegenüber seinem Vorgänger SPARROW [1] erreicht. Aus den durchgeführten Untersuchungen geht weiterhin hervor, dass *SPARROW das beste Programm zur Vorhersage kurzer Proteine ist. In Kombination mit anderen Vorhersageprogrammen erweist sich

*SPARROW als wichtiger Beitrag zur Aufstellung kombinierter Vorhersageverfahren, die die Korrektheit der vorhergesagten Sekundärstruktur drastisch verbessern können. So kann in der Kombination mit PSIPRED [30] bereits mittels einer einfachen Kombinationsmethode eine Genauigkeit von 82,54% erreicht werden. Ein weiterer Vorteil von *SPARROW äußert sich in der Möglichkeit der Vorhersage der acht von DSSP vorhergesagten Sekundärstrukturklassen, was potentiell mit einer Genauigkeit von 68,28% möglich ist.

8.1. Summary

The prediction of the secondary structure of proteins as a first step towards elucidating their 3D structure and function is an important issue of modern molecular biology. In this thesis a new method for the prediction of protein secondary structure is presented in form of the program *SPARROW, which is based on a newly developed learning machine. This learning machine, denoted as *vector valued classifier*, uses a vector valued function to project high dimensional input data into a lower dimensional output space and uses the orientation of these projections within the output space to perform the classification of the input data into secondary structure classes. To maximize the accuracy of classifications the vector valued classifier is optimized by means of empirical risk minimization, a technique of statistical learning theory.

The prediction scheme underlying *SPARROW consists of three stages, the first two of which are realized by vector valued classifiers while the third consists of an artificial neural network. The first stage correlates the secondary structure of residues, represented by sequence windows encoded in form of PSI-BLAST profiles ([78], [79]), with the primary structure (sequence). The reference secondary structure is provided by DSSP [13]. The prediction of the first stage is further processed by the second stage, which thus performs a structure-structure correlation. The final stage, being another structure-structure correlation, makes use of nonlinear nature of the neural network to allow for higher order correlations in the prediction. For this hierarchic composition each stage effectively improves the prediction of the previous stages. Furthermore the architecture of *SPARROW allows to influence the flow of information through the system by changing the number of classes that each stage distinguishes.

In benchmark tests on predicting three secondary structure classes (helices, β-strands and random coils) performed on the ASTRAL40 dataset ([90], [91], [92]) *SPARROW achie-

ved an accuracy of 81,53% and is thus comparable to established programs for secondary structure prediction. In addition *SPARROW improves the accuracy achieved by its predecessor SPARROW [1] by 0,5%. Another result of the aforementioned investigations is that *SPARROW outperforms other programs in the prediction of short proteins. Considering the creation of combined prediction methods *SPARROW proofs to be an important contribution. For example in combination with PSIPRED [30] using a simple combination method an accuracy of 82,54% can be measured. Another advantage of the concept of *SPARROW is the possibility to predict the eight secondary structure classes provided by DSSP, which yields a potential accuracy of 68,28%.

Anhänge

A. Implementierung

An dieser Stelle soll kurz auf die Implementierung des vektorwertigen Klassifikators und des dazugehörigen Lernverfahrens eingegangen werden. Der vektorwertige Klassifikator, samt Lernverfahren und Vorhersage, wurden in C++ implementiert und nutzen die Vorteile objektorientierter Programmierung, vor allem die Vererbung. Das Lernverfahren selbst besteht aus zwei Modulen: Eines zur Berechnung der Statistiken des Lerndatensatzes, die zur Aufstellung der linearen Gleichungssysteme notwendig sind, und eines zur Lösung dieser linearen Gleichungssysteme. In Abschnitt 4.2.5 wurde der formale Rahmen für beide Module, wie auch den vektorwertigen Klassifikator selbst, definiert. Für eine effiziente Implementierung von Klassifikator und Lernverfahren müssen jedoch einige Anpassungen an diesem formalen Rahmen vorgenommen werden. Zunächst soll auf die Anpassungen des vektorwertigen Klassifikators eingegangen werden, bevor auf die Implementierung der beiden Module, aus denen das Lernverfahren besteht, eingegangen wird.

A.1. Implementierung des vektorwertigen Klassifikators

Aufgrund der Tatsache, dass das Produkt zweier reeller Zahlen kommutativ ist, muss die Matrix $\underline{\underline{A}}_k$ für eine Komponentenfunktion $f_k(\vec{x}, \vec{\theta}_k)$ eines beliebigen vektorwertigen Klassifikators mit $\vec{x} \in \mathbb{R}^n$ symmetrisch sein (siehe Abschnitt 4.3.1). Demnach kann die Komponentenfunktion

$$f_k(\vec{x}, \vec{\theta}_k) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{kij} x_i x_j + \sum_{i=1}^{n} w_{ki} x_i + b_k$$

wegen $a_{kij} = a_{kji}$ auch als

$$f_k(\vec{x}, \vec{\theta}_k) = \sum_{i=1}^{n} a_{kii} x_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} 2 \cdot a_{kij} x_i x_j + \sum_{i=1}^{n} w_{ki} x_i + b_k$$

dargestellt werden..

Durch die Symmetrie von $\underline{\underline{A}}_k$ müssen die Parameter, die nicht zur Diagonalen der Matrix gehören, lediglich einmal gespeichert werden. Zusammen mit den Diagonalelementen von $\underline{\underline{A}}_k$ sind somit statt der n^2 ursprünglich angenommenen Elemente lediglich $\frac{n \cdot (n+1)}{2}$ Ele-

mente notwendig, um die Matrix $\underline{\underline{A}}_k$ vollständig zu speichern. Demnach ergibt sich die Gesamtzahl P_u der unabhängigen Parameter der Komponentenfunktion $f_k(\vec{x}, \vec{\theta}_k)$ als:

$$P_u = \frac{n \cdot (n+1)}{2} + n + 1$$

Diese fällt gegenüber der formalen Parameterzahl $P = n^2 + n + 1$ geringer aus. Im konkreten Fall der ersten Stufe von *SPARROW ergibt sich daraus mit $n = 20 \cdot r_1$ (Verwendung von Sequenzprofilen, Abschnitt 4.1.3.1) die bereits in Abschnitt 4.3.1 gezeigte Formel:

$$P_u^{(f_k)} = \frac{20 \cdot r_1 \cdot (20 \cdot r_1 + 1)}{2} + 20 \cdot r_1 + 1$$

Bei dem verwendeten Sequenzfenster von $r_1 = 17$ (Abschnitt 5.2.1) besitzt eine Komponentenfunktion $f_k(\vec{x}, \vec{\theta}_k)$ der ersten Stufe damit $P_u^{(f_k)} = 58311$ unabhängige Parameter, statt der $P^{(f_k)} = 115941$ eingangs angenommenen. Diese Reduktion der formalen Parameterzahl $P^{(f_k)}$ auf die Zahl der unabhängigen Parameter $P_u^{(f_k)}$ bringt für die erste Stufe mehrere Vorteile mit sich:

- **Effizientere Speicherung der Parameter**: Für die einzelnen Komponentenfunktionen ist nach der Reduktion der Speicherplatz, der zur Speicherung der Parameter benötigt wird, von 905,78 kB auf 455,55 kB gesunken. Dabei wird davon ausgegangen, dass die Parameter in Form von Fließkommazahlen mit doppelter Genauigkeit (*double-precision 64-bit IEEE 754 floating point*) in Binärform gespeichert werden.

- **Effizientere Klassifikation:** Die Anzahl der Multiplikationen, die für die Auswertung einer einzelnen Komponentenfunktion für einen Eingangsvektor notwendig sind, reduziert sich von $P^{(f_k)} - 1$ auf $P_u^{(f_k)} - 1$, was zu einer schnelleren Auswertung der Klassifikatorfunktion, insbesondere für größere $N_1^{Klassen}$, führt.

- **Effizienterer Lernvorgang:** Die Gleichungssysteme zur Berechnung der Parameter der Komponentenfunktionen werden kleiner, wodurch deren Lösung, und damit auch der Lernvorgang, zeiteffizienter wird. Zusätzlich werden die zur Aufstellung der Gleichungssysteme benötigten Kovarianzmatrizen kleiner, was deren Berechnung und Speicherung effizienter macht.

A.2. Implementierung des Lernverfahrens

Durch die Vereinfachung aus Abschnitt A.1 kann für einen allgemeinen vektorwertigen Klassifikator mit Eingangsdaten $\vec{x} \in \mathbb{R}^n$ die Anzahl der unabhängigen Gewichte $\tilde{P}_u$ (in Anlehnung an $\tilde{P}$, Abschnitt 4.2.4) einer Komponentenfunktion $f_k(\vec{x}, \vec{\theta}_k)$ als $\tilde{P}_u = P_u - 1$ definiert werden. Mit $\tilde{P}_u$ kann nun entsprechend ein $\vec{W}_k^{(u)} \in \mathbb{R}^{\tilde{P}_u}$ und ein $\vec{X}^{(u)} \in \mathbb{R}^{\tilde{P}_u}$ definiert werden, sodass $f_k(\vec{x}, \vec{\theta}_k) = (\vec{W}_k^{(u)})^T \vec{X}^{(u)} + b_k$. $\vec{W}_k^{(u)}$ enthält in diesem Fall alle unabhängigen Gewichte, bestehend aus den unabhängigen Elementen von $\underline{\underline{A}}_k$ gefolgt von den Elementen von $\vec{w}_k$. Die einzelnen a_{kij} sind hierbei Zeilenweise aus $\underline{\underline{A}}_k$ ausgelesen und so in $\vec{W}_k^{(u)}$ eingetragen, dass die ersten n Elemente die Elemente a_{k11} bis a_{k1n} der ersten Zeile von $\underline{\underline{A}}_k$ sind, die folgenden $n-1$ Elemente a_{k22} bis a_{k2n} der zweiten Zeile entstammen, und so weiter. Dabei ist allen a_{kij} mit $i \neq j$ der Faktor 2 vorangestellt, da diese gegenüber den Diagonalelementen a_{kii} doppelt vorkommen. Der Aufbau des Gewichtsvektors $\vec{W}_k^{(u)} \in \mathbb{R}^{\tilde{P}_u}$ wäre somit:

$$\vec{W}_k^{(u)} = \left(a_{k11}, 2 \cdot a_{k12}, \ldots, 2 \cdot a_{k1n}, a_{k22}, 2 \cdot a_{k23}, \ldots, a_{knn}, w_{k1}, \ldots, w_{kn}\right)^T$$

Der Merkmalsvektor $\vec{X}^{(u)} \in \mathbb{R}^{\tilde{P}_u}$ enthält die zu dem jeweiligen Gewicht gehörigen Werte:

$$\vec{X}^{(u)} = \left(x_1 x_1, x_1 x_2, \ldots x_1 x_n, x_2 x_2, x_2 x_3, \ldots, x_n x_n, x_1, \ldots, x_n\right)^T$$

Mit $\vec{W}_k^{(u)}$ und $\vec{X}^{(u)}$ kann die Rechnung aus Abschnitt 4.2.4 auf analoge Weise durchgeführt werden. Die Koeffizientenmatrix $\underline{\underline{M}}^{(u)} \in \mathbb{R}^{\tilde{P}_u \times \tilde{P}_u}$ der linearen Gleichungssysteme besteht nach dieser Rechnung aus den folgenden Zeilenvektoren

$$\vec{m}_p^{(u)} = \frac{1}{N} \sum_{i=1}^{N} \vec{X}_i^{(u)} X_{ip}^{(u)} - \left(\frac{1}{N} \sum_{i=1}^{N} X_{ip}^{(u)} \right) \cdot \left(\frac{1}{N} \sum_{j=1}^{N} \vec{X}_j^{(u)} \right)$$

Für ein einzelnes Element der Koeffizientenmatrix ergibt sich daraus:

$$m_{pq}^{(u)} = \frac{1}{N} \sum_{i=1}^{N} X_{iq}^{(u)} X_{ip}^{(u)} - \left(\frac{1}{N} \sum_{i=1}^{N} X_{ip}^{(u)} \right) \cdot \left(\frac{1}{N} \sum_{j=1}^{N} X_{jq}^{(u)} \right)$$

An der Gleichung fällt auf, dass $m_{pq}^{(u)} = m_{qp}^{(u)}$ sein muss. Die Koeffizientenmatrix ist also symmetrisch (was im Übrigen bereits für die ursprüngliche Matrix $\underline{\underline{M}}$ aus Abschnitt 4.2.4 gilt), sodass auch in diesem Fall eine Reduktion der zu speichernden Datenmenge möglich

		Parameter pro Komponentenfunktion		Koeffizientenmatrix	
1. Stufe	n	Anzahl	Größe [kB]	Einträge	Größe [GB]
formal	340	115941	905,78	13442083600	100,15
reduziert	340	58311	455,55	1700057205	12,67
2. Stufe	n	Anzahl	Größe [kB]	Einträge	Größe [GB]
formal	119	14281	111,56	203918400	1,52
reduziert	119	7260	56,71	26350170	0,2

Tabelle 14: *Speicherbedarf für die Parameter der Komponentenfunktionen der vektorwertigen Klassifikatoren der ersten beiden Stufen von *SPARROW und der zur ihrer Berechnung notwendigen Koeffizientenmatrizen bei Verwendung der formalen Definition beider Größen und nach der Reduktion*

ist indem Elemente, die nicht auf der Diagonalen liegen, jeweils einmalig gespeichert werden. Somit müssen zur Speicherung der Koeffizientenmatrix lediglich

$$\frac{\widetilde{P}_u \cdot (\widetilde{P}_u + 1)}{2}$$

Elemente berücksichtigt werden, statt der ursprünglichen $\widetilde{P}^2$, die sich formal aus der Rechnung in Abschnitt 4.2.4 ergeben.

Die Konsequenz aus dieser Betrachtung kann abermals Anhand der ersten Stufe von *SPARROW veranschaulicht werden: Für $r_1 = 17$ besitzt die ursprüngliche $\widetilde{P} \times \widetilde{P}$-Koeffizientenmatrix formal $115940^2 = 13442083600$ Elemente zu deren Speicherung $13442083600 \cdot 8\,\mathrm{B} = 100{,}15\,\mathrm{GB}$ notwendig wären. Hierbei wird abermals von einer Speicherung als Fließkommazahlen doppelter Genauigkeit ausgegangen. Die berechnete Größe entspricht auch der Größe des Hauptspeichers, der benötigt wird um die Matrix bei der Lösung der linearen Gelichungssysteme als ganzes aufzunehmen. Für die $\widetilde{P}_u \times \widetilde{P}_u$-Koeffizientenmatrix $\underline{\underline{M}}^{(u)}$, die sich aus der Reduktion der Anzahl der Gewichte der einzelnen Komponentenfunktionen ergibt, wären zunächst $58310^2 \cdot 8\,\mathrm{B} = 3400056100 \cdot 8\,\mathrm{B} = 25{,}33\,\mathrm{GB}$ zur Speicherung nötigt, die sich jedoch durch die Symmetrie der Koeffizientenmatrix auf $1700057205 \cdot 8\,\mathrm{B} = 12{,}67\,\mathrm{GB}$ reduzieren lassen. Wie das Beispiel zeigt, wird dadurch der Ressourcenbedarf bei Aufstellung und Speicherung der Koeffizientenmatrix, wie auch bei der Lösung der Gleichungssysteme, immens gesenkt.

Logischerweise kann die vorgestellte Reduktion gleichermaßen in der ersten wie auch in der zweiten Stufe von *SPARROW verwendet werden. Im Fall der zweiten Stufe ist der Speicherbedarf aufgrund der wesentlich kleineren Eingangsvektoren weniger kritisch. Tabelle 14 zeigt einen Vergleich zwischen dem Speicherplatz, der sich für die Speicherung

der Parameter des vektorwertigen Klassifikators, beziehungsweise einer Koeffizientenmatrix, ergibt, wenn von der formalen Definition beziehungsweise der reduzierten Form ausgegangen wird. Dargestellt ist der benötigte Speicherbedarf für die ersten beiden Stufen von *SPARROW, wie sie für die endgültige Version von *SPARROW (Abschnitt 5.4.1) verwendet werden, sowie die Größe n der jeweiligen Eingangsvektoren. Die Größe der Eingangsvektoren der zweiten Stufe ergibt sich daraus, dass die erste Stufe acht Klassen unterscheidet und für die zweite Stufe $r_2 = 17$ gilt (Abschnitt 5.2.1).

Mit Hilfe der Anpassungen aus diesem Abschnitt kann nun auf die Implementierung des Statistik- und des Lösungsmoduls eingegangen werden, die zusammen das Lernverfahren für den vektorwertigen Klassifikator bilden. Ihre Funktionsweise sollen die nächsten beiden Abschnitte beschreiben, wobei vom Problem der Sekundärstrukturvorhersage von Proteinen als zu lösendem Klassifikationsproblem ausgegangen wird.

A.2.1. Das Statistikmodul

Die Koeffizientenmatrix $\underline{\underline{M}}^{(u)} = \left\langle \vec{X}^{(u)} \left(\vec{X}^{(u)} \right)^T \right\rangle - \left\langle \vec{X}^{(u)} \right\rangle \left\langle \left(\vec{X}^{(u)} \right)^T \right\rangle$ besteht aus der Differenz zweier Größen, die unabhängig voneinander bestimmt werden können. Diese Tatsache macht sich das Statistikmodul zunutze, welches die Daten für die Koeffizientenmatrix erzeugt: Das Programme berechnet die beiden Größen $\underline{\underline{Z}} = N \cdot \left\langle \vec{X}^{(u)} \left(\vec{X}^{(u)} \right)^T \right\rangle$ und $\vec{z} = N \cdot \left\langle \vec{X}^{(u)} \right\rangle$ separat und speichert diese auch in getrennten Dateien. Die erste Größe ist eine Matrix, die für jede der betrachteten Klassen gleich ist und vom Programm elementweise berechnet wird, und zwar aus:

$$Z_{pq} = \sum_{i=1}^{N} X_{iq}^{(u)} X_{ip}^{(u)} \quad \text{mit} \quad q \geq p$$

Dabei handelt es sich um das q-te Element der p-ten Zeile der *Statistikmatrix* $\underline{\underline{Z}} \in \mathbb{R}^{\tilde{P}_u \times \tilde{P}_u}$. Diese elementweise Berechnung hat gegenüber der zeiteffizienteren Berechnung, bei der auf der gesamten Matrix operiert wird, einen großen Vorteil: Sie benötigt weitaus weniger Hauptspeicher. Das Programm ist so ausgelegt, dass es berechnete Matrixelemente in einem Puffer ablegt. Sind genug Elemente berechnet worden um den Puffer vollständig zu füllen, so wird der Inhalt des Puffers der Datei angehängt, die die Statistikmatrix enthalten soll, und der Puffer anschließend geleert. Der daraus resultierende gerin-

gere Speicherbedarf des Statistikmoduls erlaubt es, die Berechnung von $\underline{\underline{Z}}$ verstärkt zu parallelisieren, indem der Lerndatensatz D^{lern} in mehrere disjunkte Teilmengen D_l^{lern} unterteilt wird. Auf diese Weise kann eine Reihe partieller Statistikmatrizen $\underline{\underline{Z}}_l$ berechnet werden (eine für jedes D_l^{lern}), deren Summe die Statistikmatrix $\underline{\underline{Z}}$ des gesamten Lerndatensatzes D_l^{lern} ergibt. Diese parallelisierte Berechnung der einzelnen $\underline{\underline{Z}}_l$ kann mit dem Statistikmodul auf Computern mit mehreren Prozessoren durchgeführt werden, die denselben Hauptspeicher teilen, ohne dass es einem der parallelen Prozesse an Speicher mangelt. Die Möglichkeit der Parallelisierung führt wiederum zu einem zeitlichen Gewinn bei der Berechnung von $\underline{\underline{Z}}$, da die partiellen Statistikmatrizen $\underline{\underline{Z}}_l$ aufgrund der kleineren Datensätze, die bearbeitet werden müssen, schneller berechnet werden können als $\underline{\underline{Z}}$. Dieser Effekt verstärkt sich mit steigender Anzahl an D_l^{lern}.

Der *Statistikvektor* $\vec{z}$ ergibt sich aus der Summe aller Eingangsvektoren des Lerndatensatzes, wobei hier im Gegensatz zur Matrix $\underline{\underline{Z}}$ die Berechnung klassenweise für jede der acht von DSSP unterschiedenen Sekundärstrukturklassen erfolgt. Auf diese Weise werden zunächst acht verschiedene Statistikvektoren $\vec{z}_\sigma$ berechnet. Diese ergeben sich aus:

$$\vec{z}_\sigma = \sum_{i=1}^{N_\sigma} \vec{X}_{\sigma i}^{(u)}$$

Hierbei ist N_σ die Anzahl der Vertreter $\vec{X}_{\sigma i}^{(u)}$ der DSSP-Klasse ζ_σ innerhalb des Lerndatensatzes. $\vec{z}$ ergibt sich aus der Summe aller $\vec{z}_\sigma$. Die partiellen Statistikvektoren werden in unabhängigen Dateien abgespeichert und beanspruchen in der Regel nicht viel Speicherplatz: Die Statistikvektoren benötigen so viel Speicherplatz wie die Parameter einer Komponentenfunktion, was laut Tabelle 14 in der Regel einigen hundert Kilobytes entspricht. Die separate Speicherung der $\vec{z}_\sigma$ hat den Vorteil, dass diese auch für die Berechnung der rechten Seiten der linearen Gleichungssysteme verwendet werden können und eine Flexibilität in der Wahl des Klasseneinteilung zulassen. Dies wird in Abschnitt A.2.2 angesprochen. Mit der Aufstellung der $\vec{z}_\sigma$ und von $\underline{\underline{Z}}$ sind die Berechnungen des Statistikmoduls abgeschlossen. Alle weiteren Schritte, wie das Aufstellen der einzelnen linearen Gleichungssysteme und deren Lösung, übernimmt das Lösungsmodul.

A.2.2. Das Lösungsmodul

Basierend auf der in Abschnitt 4.2.4.1 eingeführten regularisierten Fehlerfunktion baut das Lösungsmodul zunächst die Koeffizientenmatrix aus den vom Statistikmodul produzierten Daten auf. Mit der Definition von $m_{pq}^{(u)}$ als

$$m_{pq}^{(u)} = \frac{1}{N}\sum_{i=1}^{N} X_{iq}^{(u)} X_{ip}^{(u)} - \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}^{(u)}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N} X_{jq}^{(u)}\right)$$

und den Definitionen von $\underline{\underline{Z}}_{pq}$ und $\vec{z}$ aus Abschnitt A.2.1 ergibt sich somit:

$$m_{pq}^{(u)} = \begin{cases} \frac{1}{N} Z_{pq} - \frac{z_p z_q}{N^2}, & q \geq p \\ \frac{1}{N} Z_{pq} - \frac{z_p z_q}{N^2} + \lambda_p, & q = p \end{cases}$$

Dabei entsprechen z_p und z_q der p-ten beziehungsweise q-ten Komponente von $\vec{z}$. Die einzelnen Elemente $m_{pq}^{(u)}$ werden direkt beim Einlesen von $\underline{\underline{Z}}_{pq}$ berechnet, sodass sich die reduzierte Koeffizientenmatrix nur einmal im Hauptspeicher befindet.

Die Berechnung eines Elements der rechten Seite $\vec{v}^{(u)}$ eines linearen Gleichungssystems ergibt sich aus:

$$v_{kp}^{(u)} = \frac{1}{N}\sum_{i=1}^{N} y_{ik} X_{ip}^{(u)} - \left(\frac{1}{N}\sum_{i=1}^{N} X_{ip}^{(u)}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N} y_{jk}\right)$$

Je nach Klasseneinteilung lässt sich der Lerndatensatz in $N^{Klassen}$ Klassen ζ_ρ unterteilen, bestehend aus den Eingangsvektoren $\vec{X}_{\rho i}^{(u)}$ und den dazugehörigen Ausgangsvektoren $\vec{y}_{\rho i}$. Jede dieser Klassen entspricht einer der DSSP-Klassen ζ_σ (im Fall des vollständigen Klasseneinteilung) oder einer Vereinigung mehrerer DSSP-Klassen (im Fall der lockeren, quasi-lockeren oder strengen Klasseneinteilung). Im ersten Fall kann $\vec{z}_\rho = \vec{z}_\sigma$ gesetzt werden, im zweiten Fall ist $\vec{z}_\rho$ die Summe der betreffenden $\vec{z}_\sigma$. Mit N_ρ als Anzahl der Elemente der Klasse ζ_ρ im Lerndatensatz kann die obige Summe in Teilsummen über die einzelnen Klassen zerlegt werden:

$$v_{kp}^{(u)} = \frac{1}{N}\sum_{\rho=1}^{N^{Klassen}} \left(\sum_{i=1}^{N_\rho} y_{\rho ik} X'_{\rho ip} - \left(\sum_{i=1}^{N_\rho} X_{\rho ip}^{(u)}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N_\rho} y_{\rho jk}\right)\right)$$

Bei dieser Form der Summierung entspricht $y_{\rho ik} = g_{\rho k}$, da in den Summen innerhalb der

äußeren Klammer alle $y_{\rho ik}$ mit $i=1,\ldots,N_\rho$ für ein bestimmtes ρ gleich sind und den k-ten Komponenten des jeweiligen Klassenvektors entsprechen. Somit ergibt sich für $v_{kp}^{(u)}$:

$$v_{kp}^{(u)}=\frac{1}{N}\sum_{\rho=1}^{N^{Klassen}}\left(g_{\rho k}\sum_{i=1}^{N_\rho}X_{\rho ip}^{(u)}-\left(\sum_{i=1}^{N_\rho}X_{\rho ip}^{(u)}\right)\cdot\left(\frac{1}{N}\sum_{j=1}^{N_\rho}g_{\rho k}\right)\right)$$

$$\Leftrightarrow v_{kp}^{(u)}=\frac{1}{N}\sum_{\rho=1}^{N^{Klassen}}\left(g_{\rho k}\sum_{i=1}^{N_\rho}X_{\rho ip}^{(u)}-\left(\sum_{i=1}^{N_\rho}X_{\rho ip}^{(u)}\right)\cdot\left(\frac{N_\rho}{N}g_{\rho k}\right)\right)$$

An dieser Stelle kann die Definition von $\vec{z}_\rho$, beziehungsweise einer Komponente dieses Vektors eingesetzt werden:

$$\Rightarrow v_{kp}^{(u)}=\frac{1}{N}\sum_{\rho=1}^{N^{Klassen}}\left(g_{\rho k}z_{\rho p}-z_{\rho p}\cdot\frac{N_\rho}{N}g_{\rho k}\right)$$

$$\Leftrightarrow v_{kp}^{(u)}=\frac{1}{N}\sum_{\rho=1}^{N^{Klassen}}\left(\left(1-\frac{N_\rho}{N}\right)g_{\rho k}z_{\rho p}\right)$$

Nach der Aufstellung der linearen Gleichungssysteme können die Parameter des vektorwertigen Klassifikators als deren Lösungen berechnet werden. Zu diesem Zweck wird die Cholesky-Zerlegung verwendet, die eine effiziente Methode zur Lösung linearer Gleichungssysteme mit symmetrischen Koeffizientenmatrizen darstellt [101]. Mittels dieser Zerlegung wird jedes der $N^{Klassen}-1$ Gleichungssysteme gelöst, wobei die Regularisierung hier einen weiteren Vorteil offenbart: Der Lösungsalgorithmus nimmt Divisionen durch die Diagonalelemente der Koeffizientenmatrix vor. Aufgrund der endlichen Genauigkeit von Computern kann es passieren, dass diese Werte zu nahe bei Null sind und so Singularitäten, also Divisionen durch Null, entstehen. Durch Addition einer kleinen positiven Zahl zu den Diagonalelementen können diese vermieden werden, ohne dass zu große Abweichungen in den Lösungen der linearen Gleichungssysteme entstehen. Diese Lösungen bilden die Gewichte $\vec{W}_k^{(u)}$ der Komponentenfunktionen $f_k(\vec{x},\vec{\theta}_k)$. Als letzten Schritt berechnet das Lösungsmodul b_k aus der Gleichung:

$$\Leftrightarrow b_k=\frac{1}{N}\sum_{i=1}^{N}\left(y_{ik}-\left(\vec{W}_k^{(u)}\right)^T\vec{X}_i^{(u)}\right)$$

Somit sind die Parameter $\hat{\vec{\theta}}$ der Klassifikatorfunktion, die den quadratischen Fehler über dem Lerndatensatz minimieren, vollständig bestimmt und der vektorwertige Klassifikator

$N^{Klassen}$	Klassenvektoren
2	g_1 = 1 g_2 = -1
3	$\vec{g}_1^T$ = (1 0) $\vec{g}_2^T$ = (-0,5 0,866025403784439) $\vec{g}_3^T$ = (-0,5 -0,866025403784439)
4	$\vec{g}_1^T$ = (1 0 0) $\vec{g}_2^T$ = (-0,333 0,943 0) $\vec{g}_3^T$ = (-0,333 -0,471 0,816) $\vec{g}_4^T$ = (-0,333 -0,471 -0,816)
8	$\vec{g}_1^T$ = (1 0 0 0 0 0 0) $\vec{g}_2^T$ = (-0,143 0,99 0 0 0 0 0) $\vec{g}_3^T$ = (-0,143 -0,165 0,976 0 0 0 0) $\vec{g}_4^T$ = (-0,143 -0,165 -0,195 0,956 0 0 0) $\vec{g}_5^T$ = (-0,143 -0,165 -0,195 -0,239 0,926 0 0) $\vec{g}_6^T$ = (-0,143 -0,165 -0,195 -0,239 -0,309 0,873 0) $\vec{g}_7^T$ = (-0,143 -0,165 -0,195 -0,239 -0,309 -0,436 0,756) $\vec{g}_8^T$ = (-0,143 -0,165 -0,195 -0,239 -0,309 -0,436 -0,756)

Tabelle 15: *Beispiele von Klassenvektoren, wie sie zur Lösung der Klassifikationsprobleme in den einzelnen Stufen der für *SPARROW verwendeten Vorhersageschemata verwendet werden, sowie für das Zweiklassenproblem*

zur Klassifikation bekannter wie unbekannter Daten bereit.

B. Beispiele für Klassenvektoren

An dieser Stelle sollen einige Beispiele von Klassenvektoren gezeigt werden, wie sie der in Abschnitt 4.2.2.1 beschriebene Algorithmus berechnet. Diese sind für alle Klassifikationsprobleme, die in den einzelnen Stufen der untersuchten Vorhersageschemata gelöst werden (also das Drei-, das Vier- und das Achtklassenproblem, Abschnitt 5.3), sowie das Zweiklassenproblem in Tabelle 15 zusammengefasst. Der Tabelle kann die genaue Funktionsweise des Algorithmus entnommen werden: Ausgehend von einem Vektor $\vec{g}_1$, der mit dem ersten Basisvektor des jeweiligen kartesischen Koordinatensystems gleichgesetzt wird, berechnet der Algorithmus die weiteren Klassenvektoren nacheinander, wobei bei jedem Klassenvektor gegenüber dem vorher berechneten eine Komponente weniger den Wert Null aufweist. Eine Ausnahme bildet der letzte Klassenvektor $\vec{g}_{N^{Klassen}}$ der sich, wie in Abschnitt 4.2.2.1 beschrieben, als negative Summe der übrigen Klassenvektoren ergibt. Weiterhin zeigt sich, dass alle Klassenvektoren in Tabelle 15 die in Abschnitt 4.2.2.1 geforderten Bedingungen an Betrag und Skalarprodukte erfüllen.

C. Endgültige Ergebnisse für die lockere und die quasi-lockere Klasseneinteilung

Nach den Betrachtungen in Abschnitt 5.3.2.1 war es notwendig, das Verhalten des 8-4_E-3- und des 8-4_C-3-Vorhersageschemas, beziehungsweise deren Erweiterungen in Form des 8-$4_E(\pi)$-3- und des 8-$4_C(\pi)$-3-Vorhersageschemas, auf der lockeren Klasseneinteilung zu untersuchen. Dabei sollte ein Vergleich zum Verhalten des 8-8(π)-3-, des 8-8-3- und des 3-3-3-Vorhersageschemas gezogen werden. Zu diesem Zweck wurde jeweils eine Kreuzvalidierung für das 8-4_E-3-, 8-4_C-3-, 8-$4_E(\pi)$-3- und 8-$4_C(\pi)$-3-Vorhersageschema unter Verwendung der lockeren Klasseneinteilung durchgeführt und deren Ergebnisse untereinander sowie mit denen des 8-8(π)-3-, 8-8-3- und 3-3-3-Vorhersageschemas verglichen. Zudem sollte auch die Frage geklärt werden, wie sich alle Vorhersageschemata zur Lösung des Dreiklassenproblems (8-8-3-, 8-4_E-3-, 8-4_C-3-, 8-8(π)-3-, 8-$4_E(\pi)$-3-, 8-$4_C(\pi)$-3- und 3-3-3-Vorhersageschema) bei der Vorhersage der quasi-lockeren Klasseneinteilung verhalten. Zur Beantwortung dieser Frage wurde mit allen sieben Vorhersageschemata jeweils eine Kreuzvalidierung mit der quasi-lockeren Klasseneinteilung durchgeführt und die so erhaltenen Ergebnisse wurden untereinander verglichen. Ein Ausschnitt der Ergebnisse der beiden obigen Untersuchungen wurde bereits in Abschnitt 5.3.2.2 beschrieben. In diesem Anhang sollen diese Ergebnisse in ihrer Gesamtheit im Detail besprochen werden.

Die Ergebnisse der Kreuzvalidierung auf der lockeren Klasseneinteilung sind in Form der Mittelwerte und Standardabweichungen der Qualitätsmaße Q_3, R_3, q_H, q_E und q_C in Tabelle 16 auf der folgenden Seite dargestellt. Diese Tabelle stellt eine Erweiterung von Tabelle 8 auf Seite 158 um die Ergebnisse des 8-4_C-3-, 8-4_E-3-, 8-$4_C(\pi)$-3- und 8-$4_E(\pi)$-3-Vorhersageschemas dar und enthält somit die Ergebnisse für jede der drei Stufen aller Vorhersageschemata, die zur Vorhersage der Sekundärstruktur nach der lockeren Klasseneinteilung verwendet wurden. Die Ergebnisse der erste Stufe sind in Tabelle 16 der Vollständigkeit halber aufgeführt, sollen jedoch aufgrund der Tatsache, dass sie gegenüber Tabelle 8 keine Veränderung erfahren haben (bis auf das 3-3-3-Vorhersageschema benutzen alle anderen Vorhersageschemata das 8-Vorhersageschema als erste Stufe), nicht weiter besprochen werden. In der zweiten Stufe kann festgestellt werden, dass das 8-4_E- und das 8-4_C-Vorhersageschema in *Recall* und *Prediction* hinsichtlich Q_3 und R_3 sehr ähnliche Werte liefert. Im *Recall* besitzt das 8-4_E-Vorhersageschema einen höheren Wert von Q_3 und R_3 als das 8-4_C-Vorhersageschema. In der *Prediction* besitzt das 8-4_C-Vorhersage-

	locker		Q_3	R_3	q_H	q_E	q_C
1.Stufe	8-VS	*Recall*	0,7866 ± 0,0005	0,6748 ± 0,0007	0,8792 ± 0,0005	0,7851 ± 0,0008	0,7058 ± 0,0010
		Prediction	0,7803 ± 0,0033	0,6653 ± 0,0052	0,8755 ± 0,0042	0,7769 ± 0,0054	0,6984 ± 0,0056
	3-VS	*Recall*	0,8018 ± 0,0005	0,6920 ± 0,0007	0,8306 ± 0,0006	0,6960 ± 0,0010	0,8331 ± 0,0007
		Prediction	0,7953 ± 0,0037	0,6817 ± 0,0058	0,8255 ± 0,0059	0,6868 ± 0,0064	0,8267 ± 0,0046
2. Stufe	8-8-VS	*Recall*	0,8015 ± 0,0004	0,6967 ± 0,0007	0,8884 ± 0,0004	0,7955 ± 0,0009	0,7281 ± 0,0008
		Prediction	0,7918 ± 0,0033	0,6819 ± 0,0051	0,8822 ± 0,0043	0,7834 ± 0,0047	0,7166 ± 0,0050
	8-4_C-VS	*Recall*	0,8084 ± 0,0004	0,7030 ± 0,0005	0,8858 ± 0,0005	0,7273 ± 0,0009	0,7837 ± 0,0010
		Prediction	0,7999 ± 0,0028	0,6897 ± 0,0044	0,8795 ± 0,0043	0,7165 ± 0,0047	0,7743 ± 0,0050
	8-4_E-VS	*Recall*	0,8093 ± 0,0004	0,7055 ± 0,0007	0,8878 ± 0,0005	0,7549 ± 0,0008	0,7694 ± 0,0008
		Prediction	0,7995 ± 0,0035	0,6902 ± 0,0055	0,8812 ± 0,0047	0,7415 ± 0,0054	0,7586 ± 0,0059
	3-3-VS	*Recall*	0,8049 ± 0,0005	0,6968 ± 0,0007	0,8340 ± 0,0006	0,6875 ± 0,0007	0,8420 ± 0,0004
		Prediction	0,7982 ± 0,0038	0,6863 ± 0,0059	0,8289 ± 0,0059	0,6780 ± 0,0064	0,8354 ± 0,0049
3. Stufe	8-8-3-VS	*Recall*	0,8177 ± 0,0006	0,7175 ± 0,0009	0,8513 ± 0,0033	0,7655 ± 0,0100	0,8160 ± 0,0060
		Prediction	0,8071 ± 0,0036	0,7011 ± 0,0056	0,8426 ± 0,0060	0,7519 ± 0,0118	0,8053 ± 0,0082
	8-8(π)-3-VS	*Recall*	0,8194 ± 0,0004	0,7203 ± 0,0007	0,8559 ± 0,0045	0,7697 ± 0,0074	0,8139 ± 0,0068
		Prediction	0,8087 ± 0,0034	0,7036 ± 0,0054	0,8470 ± 0,0067	0,7562 ± 0,0066	0,8031 ± 0,0087
	8-4_C-3-VS	*Recall*	0,8160 ± 0,0005	0,7146 ± 0,0007	0,8506 ± 0,0022	0,7525 ± 0,0062	0,8194 ± 0,0043
		Prediction	0,8070 ± 0,0028	0,7007 ± 0,0043	0,8429 ± 0,0056	0,7422 ± 0,0057	0,8101 ± 0,0062
	8-4_C(π)-3-VS	*Recall*	0,8175 ± 0,0003	0,7170 ± 0,0005	0,8546 ± 0,0036	0,7578 ± 0,0058	0,8167 ± 0,0050
		Prediction	0,8082 ± 0,0024	0,7028 ± 0,0041	0,8469 ± 0,0047	0,7473 ± 0,0036	0,8072 ± 0,0049
	8-4_E-3-VS	*Recall*	0,8173 ± 0,0005	0,7169 ± 0,0007	0,8520 ± 0,0038	0,7627 ± 0,0058	0,8160 ± 0,0061
		Prediction	0,8071 ± 0,0034	0,7010 ± 0,0055	0,8438 ± 0,0078	0,7495 ± 0,0076	0,8056 ± 0,0069
	8-4_E(π)-3-VS	*Recall*	0,8190 ± 0,0003	0,7195 ± 0,0005	0,8547 ± 0,0033	0,7665 ± 0,0080	0,8156 ± 0,0066
		Prediction	0,8086 ± 0,0037	0,7033 ± 0,0057	0,8464 ± 0,0062	0,7528 ± 0,0066	0,8051 ± 0,0082
	3-3-3-VS	*Recall*	0,8087 ± 0,0006	0,7034 ± 0,0010	0,8414 ± 0,0046	0,7475 ± 0,0073	0,8128 ± 0,0051
		Prediction	0,8016 ± 0,0036	0,6923 ± 0,0057	0,8359 ± 0,0073	0,7382 ± 0,0077	0,8053 ± 0,0077

Tabelle 16: *Verhalten der einzelnen Stufen der sieben Vorhersageschemata für die lockere Klasseneinteilung*

schema einen höheren Q_3 -Index als das 8-4_E-Vorhersageschema, jedoch einen niedrigeren Wert für R_3 . Bei Betrachtung der Standardabweichungen von Q_3 und R_3 im Verhältnis zu deren Unterschieden zwischen dem 8-4_C- und 8-4_E-Vorhersageschema spricht jedoch vieles dafür, dass das 8-4_C- und 8-4_E-Vorhersageschema in der *Prediction* hinsichtlich Q_3 und R_3 äquivalent sind. Im Vergleich mit den übrigen Vorhersageschemata zeigt sich, dass das 8-4_C- und das 8-4_E-Vorhersageschema für Q_3 und R_3 in *Recall* und *Prediction* höhere Werte aufweisen als die übrigen beiden Vorhersageschemata in der zweiten Stufe. Dabei ist der Abstand zum 8-8-Vorhersageschema stärker ausgeprägt als der zum 3-3-Vorhersageschema und liegt beim 3-3-Vorhersageschema zudem unterhalb der jeweiligen Standardabweichung. Die Verhältnisse, die zwischen 8-4_C- und 8-4_E-Vorhersageschema in der zweiten Stufe bestehen, bleiben in der dritten Stufe für das 8-4_C-3- und das 8-4_E-3-Vorhersageschema erhalten: Während das 8-4_E-3-Vorhersageschema im *Recall* höhere Werte

von Q_3 und R_3 aufweist, sind die entsprechenden Werte in der *Prediction* fast identisch. Zudem fällt auf, dass das 8-8-3-Vorhersageschema in der *Prediction* etwa die gleichen Werte für Q_3 und R_3 aufweist wie das 8-4_C-3- und das 8-4_E-3-Vorhersageschema, während es im *Recall* dem 8-4_C-3-Vorhersageschemata geringfügig überlegen ist. Eine ähnliche Aussage lässt sich auch für den Vergleich von 8-8(π)-3-, 8-4_C(π)-3- und 8-4_E(π)-3-Vorhersageschema hinsichtlich Q_3 und R_3 machen.

Bei der Vorhersage der einzelnen Klassen zeigt sich in der zweiten Stufe, dass q_H für *Recall* und *Prediction* für 8-4_E- und 8-4_C-Vorhersageschema einen ähnlichen Wert aufweist, was aufgrund der gleichen Zusammensetzung der helikalen Klasse bei der intermediär$_E$- und intermediär$_C$-Klasseneinteilung nicht verwunderlich ist. Dieser Wert ist desweiteren auch mit dem des 8-8-Vorhersageschemas vergleichbar. Für q_E ist der Wert beim 8-4_E-Vorhersageschema höher als beim 8-4_C-Vorhersageschema, wobei der Unterschied zwischen beiden Vorhersageschemata in *Recall* und *Prediction* die Standardabweichung von q_E übersteigt. Den höchsten Wert für q_E in der zweiten Stufe weist das 8-8-Vorhersageschema auf, das zudem einen großen Abstand zum 8-4_E-Vorhersageschema besitzt. Im Fall von q_C zeigt sich, dass das 8-4_C-Vorhersageschema für die Vorhersage strukturloser Regionen besser geeignet ist als das 8-4_E-Vorhersageschema, wobei der Unterschied zwischen beiden Vorhersageschemata auch hier die entsprechenden Standardabweichungen übersteigt. Jedoch ist das 8-4_C-Vorhersageschema dem 3-3-Vorhersageschema in der Vorhersage strukturloser Regionen unterlegen. Auch in der dritten Stufe werden Helices vom 8-4_E-3- und 8-4_C-3-Vorhersageschema gleichermaßen gut klassifiziert, wobei der Wert von q_H in beiden Fällen gegenüber der zweiten Stufe geringer ausfällt. q_E und q_C erfahren hingegen für diese beiden Vorhersageschemata gegenüber deren zweiten Stufen eine Verbesserung, die im Fall von q_C höher ausfällt. Ein Vergleich beider Werte in der dritten Stufe zeigt jedoch auch, dass die Differenz zwischen dem 8-4_C-3- und dem 8-4_E-3-Vorhersageschema bei q_E und q_C in der *Prediction* unterhalb des jeweiligen Wertes der Standardabweichung liegt. Aus diesem Grund ist die Signifikanz dieser Differenzen, insbesondere in der *Prediction*, fragwürdig. Ähnliches gilt auch für den Vergleich zwischen 8-8-3-, 8-4_C-3- und 8-4_E-3-Vorhersageschema hinsichtlich der Werte von q_H, q_E und q_C. Bei Verwendung der in Abschnitt 4.3.3.1 beschriebenen Wahrscheinlichkeitsterme zeigt sich, dass 8-8(π)-3-, 8-4_C(π)-3- und 8-4_E(π)-3-Vorhersageschema gleichermaßen gut für die Vor-

	quasi-locker		Q_3	R_3	q_H	q_E	q_C
1. Stufe	8-VS	*Recall*	0,7893 ± 0,0005	0,6792 ± 0,0007	0,8792 ± 0,0005	0,8097 ± 0,0007	0,7020 ± 0,0010
		Prediction	0,7830 ± 0,0034	0,6697 ± 0,0055	0,8755 ± 0,0042	0,8012 ± 0,0062	0,6948 ± 0,0057
	3-VS	*Recall*	0,8070 ± 0,0005	0,6978 ± 0,0008	0,8275 ± 0,0006	0,6994 ± 0,0010	0,8429 ± 0,0007
		Prediction	0,8005 ± 0,0039	0,6874 ± 0,0061	0,8223 ± 0,0058	0,6901 ± 0,0075	0,8365 ± 0,0050
2. Stufe	8-8-VS	*Recall*	0,8041 ± 0,0004	0,7008 ± 0,0007	0,8884 ± 0,0004	0,8171 ± 0,0009	0,7254 ± 0,0008
		Prediction	0,7946 ± 0,0034	0,6863 ± 0,0054	0,8822 ± 0,0043	0,8058 ± 0,0053	0,7140 ± 0,0052
	8-4_C-VS	*Recall*	0,8144 ± 0,0004	0,7112 ± 0,0006	0,8858 ± 0,0005	0,7558 ± 0,0007	0,7823 ± 0,0010
		Prediction	0,8055 ± 0,0030	0,6973 ± 0,0047	0,8795 ± 0,0043	0,7440 ± 0,0054	0,7726 ± 0,0051
	8-4_E-VS	*Recall*	0,8113 ± 0,0004	0,7077 ± 0,0007	0,8878 ± 0,0005	0,7754 ± 0,0008	0,7634 ± 0,0008
		Prediction	0,8023 ± 0,0036	0,6938 ± 0,0057	0,8812 ± 0,0047	0,7637 ± 0,0060	0,7538 ± 0,0059
	3-3-VS	*Recall*	0,8109 ± 0,0005	0,7039 ± 0,0007	0,8343 ± 0,0006	0,7076 ± 0,0009	0,8421 ± 0,0006
		Prediction	0,8044 ± 0,0039	0,6936 ± 0,0061	0,8291 ± 0,0058	0,6983 ± 0,0071	0,8359 ± 0,0051
3. Stufe	8-8-3-VS	*Recall*	0,8229 ± 0,0006	0,7238 ± 0,0008	0,8481 ± 0,0033	0,7717 ± 0,0079	0,8268 ± 0,0068
		Prediction	0,8128 ± 0,0036	0,7079 ± 0,0057	0,8395 ± 0,0078	0,7590 ± 0,0114	0,8166 ± 0,0076
	8-8(π)-3-VS	*Recall*	0,8244 ± 0,0004	0,7261 ± 0,0006	0,8556 ± 0,0037	0,7754 ± 0,0047	0,8219 ± 0,0041
		Prediction	0,8140 ± 0,0036	0,7098 ± 0,0056	0,8467 ± 0,0058	0,7629 ± 0,0073	0,8112 ± 0,0078
	8-4_C-3-VS	*Recall*	0,8221 ± 0,0005	0,7224 ± 0,0006	0,8485 ± 0,0022	0,7684 ± 0,0060	0,8260 ± 0,0040
		Prediction	0,8129 ± 0,0029	0,7080 ± 0,0046	0,8408 ± 0,0059	0,7569 ± 0,0096	0,8167 ± 0,0071
	8-4_C(π)-3-VS	*Recall*	0,8239 ± 0,0003	0,7252 ± 0,0005	0,8513 ± 0,0038	0,7737 ± 0,0046	0,8252 ± 0,0039
		Prediction	0,8146 ± 0,0029	0,7106 ± 0,0045	0,8435 ± 0,0063	0,7618 ± 0,0042	0,8159 ± 0,0074
	8-4_E-3-VS	*Recall*	0,8216 ± 0,0005	0,7217 ± 0,0007	0,8487 ± 0,0033	0,7724 ± 0,0089	0,8227 ± 0,0075
		Prediction	0,8121 ± 0,0036	0,7069 ± 0,0057	0,8405 ± 0,0075	0,7608 ± 0,0120	0,8131 ± 0,0082
	8-4_E(π)-3-VS	*Recall*	0,8236 ± 0,0005	0,7248 ± 0,0006	0,8534 ± 0,0030	0,7740 ± 0,0060	0,8225 ± 0,0044
		Prediction	0,8138 ± 0,0038	0,7095 ± 0,0060	0,8450 ± 0,0058	0,7619 ± 0,0099	0,8127 ± 0,0076
	3-3-3-VS	*Recall*	0,8151 ± 0,0006	0,7113 ± 0,0009	0,8421 ± 0,0079	0,7579 ± 0,0070	0,8188 ± 0,0035
		Prediction	0,8081 ± 0,0036	0,7004 ± 0,0057	0,8384 ± 0,0065	0,7485 ± 0,0085	0,8116 ± 0,0059

Tabelle 17: *Verhalten der einzelnen Stufen aller Vorhersageschemata für die quasi-lockere Klasseneinteilung*

hersage aller drei Klassen geeignet sind, da die bereits im Vergleich zwischen 8-8-3-, 8-4_C-3- und 8-4_E-3-Vorhersageschema festgestellten geringen Unterschiede in der *Prediction* in den meisten Fällen verringert werden. Lediglich im Fall von q_E kann in der *Prediction* festgestellt werden, dass der Unterschied zwischen 8-8(π)-3- und 8-4_C(π)-3-Vorhersageschema die Standardabweichung von q_E übersteigt.

Das Verhalten aller sieben Vorhersageschemata mit allen drei Stufen ist für die quasi-lockere Klasseneinteilung in Tabelle 17 auf der folgenden Seite zusammengefasst. Aus den dargestellten Ergebnissen geht hervor, dass das relative Verhalten der einzelnen Vorhersageschemata untereinander in allen drei Stufen den Verhältnissen bei der lockeren Klasseneinteilung ähnelt. Jedoch existieren gegenüber der lockeren Klasseneinteilung auch einige Unterschiede im Verhalten der einzelnen Vorhersageschemata. So weist in der zweiten Stufe im Fall der quasi-lockeren Klasseneinteilung das 8-4_C-Vorhersageschema hinsicht-

lich Q_3 und R_3 im *Recall* höhere Werte auf als das 8-4_E-Vorhersageschema, wodurch das Verhältnis dieser beiden Vorhersageschemata gegenüber der lockeren Klasseneinteilung umgekehrt ist. Jedoch ist im Fall der *Prediction* der Unterschied, der zwischen beiden Vorhersageschemata in Q_3 und R_3 entsteht, wie bei der lockeren Klasseneinteilung etwa in der Größenordnung der entsprechenden Standardabweichungen und soll aus diesem Grund vernachlässigt werden. In der dritten Stufe weisen das 8-8-3-, das 8-4_E-3 und das 8-4_C-3-Vorhersageschema auffallend ähnliche Werte von Q_3 und R_3 für *Recall* und *Prediction* auf und die Unterschiede zwischen den einzelnen Vorhersageschemata sind weniger ausgeprägt, als dies im Fall der lockeren Klasseneinteilung beobachtet werden konnte. Die Verwendung von Wahrscheinlichkeitstermen ändert an diesem Umstand nichts, sodass auch das 8-8(π)-3-, das 8-4_E(π)-3 und das 8-4_C(π)-3-Vorhersageschema sich im Fall der quasi-lockeren Klasseneinteilung in den Werten von Q_3 und R_3 in *Recall* und *Prediction* gleichen. Auch für die Vorhersage der drei Klassen zeigen die einzelnen Vorhersageschemata im Vergleich untereinander ein sehr ähnliches Verhalten, wie bei der lockeren Klasseneinteilung. Es ist lediglich festzustellen, dass sich q_E für das 8-4_E-3-Vorhersageschema im Übergang von der zweiten zur dritten Stufe nicht wesentlich ändert. Weiterhin zeigt eine Betrachtung der dritten Stufe, dass die bereits geringen Unterschiede, die zwischen 8-8-3-, 8-4_C-3- und 8-4_E-3-Vorhersageschema wie auch zwischen 8-8(π)-3-, 8-4_C(π)-3- und 8-4_E(π)-3-Vorhersageschema, in der Vorhersage der einzelnen Klassen für die lockere Klasseneinteilung erkennbar waren, im Fall der quasi-lockeren Klasseneinteilung noch geringer ausfallen.

Danksagung

Mein Dank gilt all den Menschen, ohne die das Entstehen dieser Arbeit nicht möglich gewesen wäre. Dies wäre zunächst Prof. Dr. Ernst-Walter Knapp, der es mir ermöglichte, diese Arbeit in seiner Arbeitsgruppe durchzuführen und sie durch seine Ratschläge und zahlreiche Diskussionen mitgestaltete. Weiterhin danke ich allen Mitgliedern der AG Knapp, den gegenwärtigen wie auch den ehemaligen, die die angenehme und kreative Arbeitsatmosphäre schufen, die für eine solche Arbeit unabdingbar ist. Insbesondere sei dabei Dr. Francesco Bettella erwähnt, der mir als guter Freund und Kollege zur Seite stand und mir mit seiner Erfahrung, bei der Lösung vieler Probleme weiterhalf, mit denen ich während Entwicklung von *SPARROW konfrontiert wurde. Weiterhin danke ich meinen Freunden, die in dieser Zeit viel Geduld mit mir bewiesen und trotzdem nie den Glauben an mich verloren. Die mich immer wieder daran erinnerten, dass das Leben nicht nur aus neuronalen Netzen, Proteinen und Algorithmen besteht. Und die mir immer wieder neuen Mut zusprachen, wenn dieser mich verließ, und mir immer mit Rat und Tat beistanden.

Ganz besonders danke ich meiner Familie, die während all der Zeit, die diese Arbeit beanspruchte, immer für mich da war. Wie ein Anker auf hoher See gaben sie mir Halt und ermöglichten es mir, in meiner langen Reise innezuhalten und meine Kräfte neu zu sammeln. Mit ihren Ratschlägen und ihrer Weisheit wiesen sie mir einem Kompass gleich meinen weiteren Kurs und füllten meine Segel mit ihrem Glauben an mich, sodass ich diesen Kurs immer weiter verfolgen konnte. Das gleiche gilt für meine geliebte Selene. Wie ihre Namenspatronin, die Mondgöttin der Antike, die einem verirrten Wanderer selbst in dunkelster Nacht mit ihrem Licht den Weg erhellte, ließ sie mich meinen Weg klar vor mir erkennen. Mit einem liebevollen Lächeln auf den Lippen, das den Zweifel aus meinem Herzen fegte, und mit Augen, die in die tiefsten Winkel meiner Seele blickten und mir diese wie Spiegel zeigten, nahm sie mich bei der Hand, begleitete mich auf diesem Weg und gab mir die Kraft ihn bis zum Ende zu gehen. Für dieses Lächeln, diese Augen, diese Hand und so viele andere Dinge danke ich ihr von ganzem Herzen. Mit ihrer Liebe, ihrer Wärme und ihrem Verständnis waren diese vier besonderen Menschen, meine Familie und meine geliebte Selene, mir die treuesten und besten Begleiter, die ich mir für diese Reise hätte wünschen können.

Abbildungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

1: Bettella F., Protein Secondary Structure Prediction Unsing Optimized Scoring Functions: A Comparative Statistical Method, Freie Universität Berlin, 2009

2: Voet D., Voet J. G., Biochemistry, Wiley, 1995

3: Sørensen S. P. L., Enzymstudien II. Mitteilung. Über die Messung und die Bedeutung der Wasserstoffionenkoncentration bei enzymatischen Prozessen, Biochemische Zeitschrift 21: 131-304, 1909

4: Kauzman W., Some factors in the interpretation of protein denaturation, Adv. in Protein Chem. 14: 1-63, 1959

5: Sanger F., Sequences, sequences, and sequences, Annu. Rev. Biochem. 57: 1-28, 1988

6: Niall H. D., Automated Edman degradation: the protein sequenator, Meth. Enzymol. 27: 942-1010, 1973

7: Pauling L., Itano H. A., Singer S. J., Wells I. C., Sickle Cell Anemia, a Molecular Disease, Science 110: 543-548, 1949

8: Ramachandran G. N., Ramakrishnan C., Sasisekharan V., Stereochemistry of polypeptide chain configurations, J. Mol. Biol. 7: 95-99, 1963

9: Berg J. M., Tymoczko J. L., Stryer L., Biochemistry, W. H. Freeman and Company, 2002

10: Overhauser A., Polarization of Nuclei in Metals, Phys. Rev. 91: 476, 1953

11: Overhauser A., Polarization of Nuclei in Metals, Phys. Rev. 93: 411-415, 1953

12: Anet F. A. L., Bourn A. J. R., Nuclear magnetic resonance spectral assignments from nuclear Overhauser effects, J. Am. Chem. Soc. 87 (22): 5250-5251, 1965

13: Kabsch W., Sander C., Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features, Biopolymers 22 (12): 2577 – 2637, 1983

14: Frishman D., Argos P., Knowledge-based protein secondary structure assignment, Proteins 23 (4): 566-579, 1995

15: Creighton T. E., Proteins: Structures and Molecular Properties, W. H. Freeman and Company, 1992

16: Rose G. D., Seltzer J. P., A new algorithm for finding the peptide chain turns in a globular protein, J. Mol. Biol. 113: 153-164, 1977

17: Lee B., Richards F. M., The interpretation of protein structures: estimation of static accessibility, J. Mol. Biol. 55: 379-400, 1971

18: Shrake A., Rupley J. A., Environment and exposure to solvent of protein atoms. Lysozyme and insulin., J. Mol. Biol. 79: 351-364, 1973

19: Anfinsen C. B., Principles that Govern the Folding of Protein Chains, Science 181: 223-230, 1973

20: Govindarajan S., Recabarren R., Goldstein R. A., Estimating the total number of protein folds, Proteins 35: 408-414, 1999

21: Mirny L. A., Abkevich V. I., Shakhnovich E. I., How evolution makes proteins fold quickly, Proc. Natl. Acad. Sci. 95 (9): 4976-4981, 1998

22: Baldi P. et al., Exploiting the past and the future in protein secondary structure prediction, Bioinformatics 15 (11): 937-946, 1999

23: Chou P. Y., Fasman G. D., Prediction of protein conformation, Biochemistry 13 (2): 222-245, 1974

24: Chou P. Y., Fasman G. D., Empirical prediction of protein conformation, Annu. Rev. Biochem. 47: 251-276, 1978

25: Chou P. Y., Fasman G. D., Prediction of the secondary structure of proteins from their amino acid sequence, Adv. Enzymol. Relat. Areas Mol. Biol. 47: 45-148, 1978

26: Garnier J., Osguthorpe D. J., Robson B., Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins, J. Mol. Biol. 120 (1): 97-120, 1978

27: Rost B., PHD: predicting one-dimensional protein structure by profile based neural networks, Meth. Enzymol. 266: 525-539, 1996

28: Pollastri G., McLysaght A., Porter: a new, accurate server for protein secondary structure prediction, Bioinformatics 21 (8): 1719-1720, 2005

29: Pollastri G., Przybylski D., Rost B., Baldi P., Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles, Proteins 47 (2): 228-235, 2002

30: Jones D. T., Protein secondary structure prediction based on position specific scoring matrices, J. Mol. Biol. 292 (2): 195-202, 1999

31: Abramczak R., Porollo A., Meller J., Combining Prediction of Secondary Structure and Solvent Accessibility in Proteins, Proteins 59 (3): 467-475, 2005

32: Ouali M., King R. D., Cascaded multiple classifiers for secondary structure prediction, Protein Science 9: 1162-1176, 2000

33: Lorenz E. N., Deterministic Nonperiodic Flow, Journal of the Atmospheric Sciences 20 (2): 130-141, 1963

34: Lorenz E. N., The predictability of hydrodynamic flow, Trans. N. Y. Acad. Sci. 25 (4): 409-432, 1963

35: Hilborn R. C., Sea gulls, butterflies, and grasshoppers: A brief history of the butterfly effect in nonlinear dynamics, American Journal of Physics 72: 425-427, 2004

36: Birkhoff G. D., Dynamical Systems, Amer. Math. Soc. Colloq. Publ., 1983

37: Franklin W. S., Review of P. Duhem, Traité Élémentaire de Méchanique fondée sur la Thermodynamique Vol. 1 & 2, 1897, Phys. Rev. 6: 170-175, 1898

38: Bayes T., An Essay towards solving a Problem in the Doctrine of Chances, Philosophical Transactions of the Royal Society of London 53: 370-418, 1763

39: Zhang H., The Optimality of Naive Bayes, 2004

40: Berger J. O., Statistical Decision Theory and Bayesian Analysis, Springer-Verlag, 1985

41: Fisher R. A., The Use of Multiple Measurements in Taxonomic Problems, Annals of Eugenics 7: 179-188, 1936

42: Vapnik V. N., The Nature of Statistical Learning Theory, Springer-Verlag, 1995

43: Azoff E. M., Neural Network Time Series: Forecasting of Financial Markets, Wiley, 1999

44: Shannon C. E., A Mathematical Theory of Communication, The Bell System Technical Journal 27, July, October: 379–423, 623–656, 1948

45: Haykin S., Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999

46: Bellman R. E., Adaptive Control Processes, Princeton University Press, 1990

47: Tax D. M. J., Duin R. P. W., Using two-class classifiers for multiclass classification, Proceedings. 16th International Conference on Pattern Recognition Vol. 2: 124-127, 2002

48: Graham P., A Plan for Spam, 2002, http://www.paulgraham.com/spam.html

49: Graham P., Better Bayesian Filtering, 2003, http://www.paulgraham.com/better.html

50: Rish I., An empirical study of the naive Bayes classifier, 2001

51: Hanson R., Stutz J., Cheeseman P., Bayesian Classification Theory, NASA Teechnical Report FIA-90-12-7-01, 1991

52: Kondak K., Hommel G., Online generation of stable gait for biped robots with feedback loop algorithm, 2004

53: Rosenblatt F., The perceptron : a probabilistic model for information storage and organization in the brain., Psychological Reviews 65 (6): 386-408, 1958

54: Gerstner W., Kreitner A. K., Markram H., Herz A. V. M., Neural codes: Firing rates and beyond, Proc. Natl. Acad. Sci. 94 (24): 12740-12741, 1997

55: Lang K., Hinton G., The development of the time-delayed neural network architecture for speech recognition, Technical Report CMU-CS-88-152, Carnegy-Mellon University, Pittsburgh, PA, 1988

56: Kohonen T., Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1): 59-69, 1982

57: Hopfield J. J., Neural networks and physical systems with emergent collective computational properties, Proc. Natl. Acad. Sci. 79 (8): 2554-2558, 1982

58: Hopfield J. J., Neurons with graded response have collective computational properties like those of two-sate neurons, Proc. Natl. Acad. Sci. 81: 3088-3092, 1984

59: Wills T. J. , Lever C., Cacucci F., Burgess N., O'Keefe J., Attractor dynamics in the hippocampal representation of the local environment, Science 308: 873-876, 2005

60: Vapnik V. N., Principles of risk minimization for learning theory, Advances in Neural Information Processing Systems 4: 831-838, 1992

61: Garnier J., Gibrat J. F., Robson B., GOR method for predicting protein secondary structure from amino acid sequence, Meth. Enzymol. 266: 540-553, 1996

62: Kloczkowski A., Ting K. L., Jernigan R. L., Garnier J., Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence, Proteins 49 (2): 154-166, 2002

63: Sen T. Z., Cheng H. T., Kloczkowski A., Jernigan R. L., A consensus data mining secondary structure prediction by combining GOR V and fragment database mining, Protein Sci. 15: 2499–2506, 2006

64: Cuff J. A., Barton G. J., Evaluation and improvement of multiple sequence methods for protein secondary structure prediction, Proteins 34 (4): 508-519, 1999

65: Meiler J., Mueller M., Ziedler A., Schmaeschke F., Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks, J. Mol. Model. 7 (9): 360-369, 2001

66: Guo J., Chen H., Sun Z., Lin Y., A novel method for protein secondary structure prediction using dual-layer SVM and profiles, Proteins 54 (4): 738-743, 2004

67: Aydin Z., Altunbasak Y., Borodovsky M., Protein secondary structure prediction for a single sequence using hidden semi-Markov models, BMC Bioinformatics 7 (1): 178, 2006

68: Chu W., Ghahramani Z., Podtelezhnikov A., Wild D. L., Bayesian segmantal models with multiple sequence alignment profiles for protein secondary structure and contact map prediction, IEEE/ACM Trans. Comput. Biol. Bioinform. 3 (2): 98-113, 2006

69: Xu Y., Xu D., Protein threading using PROSPECT: design and evaluation, Proteins 40 (3): 343-354, 2000

70: Frishman D., Argos P., Seventy-five percent accuracy in protein secondary structure prediction, Proteins 27 (3): 329-335, 1997

71: Lin K., Simossis V. A., Taylor W. R., Heringa J., A simple and fast secondary structure prediction method using hidden neural networks, Bioinformatics 21 (2): 152-159, 2005

72: Karplus K., Barret C., Hughey R., Hidden Markov Models for detecting remote protein homologies, Bioinformatics 14 (10): 846-856, 1998

73: Bondugula R., Xu D., MUPRED: a tool for bridging the gap between template based methods and sequence profile based methods for protein secondary structure prediction, Proteins 66 (3): 664-670, 2007

74: Keller J., Gray M., Givens J., A fuzzy k-nearest neighbour algorithm, IEEE Trans. Syst. Man and Cybernet. 15: 580-585, 1985

75: Berman H. M. et al., The Protein Data Bank, Nucl. Acids Res. 28 (1): 235–242, 2000

76: Rost B., Sander C., Prediction of protein secondary structure at better than 70% accuracy, J. Mol. Biol. 232 (2): 584-599, 1993

77: Rost B., Sander C., Improved prediction of protein secondary structure by use ofsequence profiles and neural networks, Proc. Natl. Acad. Sci. 90: 7558-7562, 1993

78: Altschul S. F. et al., Basic local alignment search tool, J. Mol. Biol. 215 (3): 403-410, 1990

79: Altschul S. F. et al., Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucl. Acids Res. 25 (17): 3389-3402, 1997

80: Eyrich V. A., Martí-Renom M. A., Przybylski D., Madhusudhan M. S., Fiser A., Pazos F., Valencia A., Sali A., Rost B., EVA: continuous automatic evaluation of protein structure prediction servers, Bioinformatics 17 (12): 1242–1243, 2001

81: Rost B., Eyrich V. A., EVA: large-scale analysis of secondary structure prediction, Proteins 5: 192–199, 2001

82: Koh I. Y., Eyrich V. A., Martí-Renom M. A., Przybylski D., Madhusudhan M. S., Eswar N., Graña O., Pazos F., Valencia A., Sali A., Rost B., EVA: evaluation of protein structure prediction servers, Nucl. Acids Res. 31 (13): 3311–3315, 2003

83: Coxeter H. S. M., Regular Polytopes, Dover Publications, 1973

84: Parks H. R., Wills D. C., An Elementary Calculation of the Dihedral Angle of the Regular n-Simplex, The American Mathematical Monthly 109 (8): 756-758, 2002

85: Gorodkin J., Comparing two K-category assignments by a K-category correlation coefficient, Comp. Biol. Chem. 28: 367–374, 2004

86: Murzin A. G., Brenner S. E., Hubbard T., Chothia C., SCOP: a structural classification of proteins database for the investigation of sequences and structures, J. Mol. Biol. 247: 536-540, 1995

87: Lo Conte L., Brenner S. E., Hubbard T. J. P., Chothia C., Murzin A., SCOP database in 2002: refinements accommodate structural genomics, Nucl. Acids Res. 30 (1): 264-267, 2002

88: Andreeva A., Howorth D., Brenner S. E., Hubbard T. J. P., Chothia C., Murzin A. G., SCOP database in 2004: refinements integrate structure and sequence family data, Nucl. Acids Res. 32: D226-D229, 2004

89: Andreeva A., Howorth D., Chandonia J.-M., Brenner S. E., Hubbard T. J. P., Chothia C., Murzin A. G., Data growth and its impact on the SCOP database: new developments, Nucl. Acids Res. 2008 36: D419-D425, 2007

90: Brenner S. E., Koehl P., Levitt M., The ASTRAL compendium for sequence and structure analysis, Nucl. Acids Res. 28 (1): 254-256, 2000

91: Chandonia J. M., Walker N. S., Lo Conte L., Koehl P., Levitt M., Brenner S. E., ASTRAL compendium enhancements, Nucl. Acids Res. 30 (1): 260-263, 2002

92: Chandonia J. M., Hon G., Walker N. S., Lo Conte L., Koehl P., Levitt M., Brenner S. E., The ASTRAL compendium in 2004, Nucl. Acids Res. 32: D189-D192, 2004

93: Riedmiller H., Braun H., Rprop - A Fast Adaptive Learning Algorithm, Proc. of ISCIS VII, 1992

94: Press W. H., Teukolsky S. A., Vettering W. T., Flannery B. P., Numerical Recipes in C++, Cambridge University Press, 2002

95: Hosmer D., Lemeshow S., Applied logistic regression, Wiley New York, 2000

96: Rost B., Rising accuracy of protein secondary structure prediction, Protein structure determination, analysis, and modeling for drug discovery : 207-249, 2003

97: Rost B., Sander C., Schneider R., Redefining the Goals of Protein Secondary Structure Prediction, J. Mol. Biol. 235: 13-26, 1994

98: Fodje M. N., Al-Karadaghi S., Occurrence, conformational features and amino acid propensities for the pi-helix, Protein Engineering 15 (5): 353-358, 2002

99: Andersen C. A., Palmer A. G., Brunak S., Rost B., Continuum Secondary Structure Captures Protein Flexibility, Structure 10: 175-184, 2002

100: Carter P., Andersen C. A., Rost B., DSSPcont: continuous secondary structure assignments for proteins, Nucl. Acids Res. 31 (13): 3293-3295, 2003

101: Press W. H., Flannery B. P., Teukolsky S. A., Vetterling W. T., Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, 1986