KAPITEL 3 Methoden

Im folgenden wird eine Methode entwickelt, die anhand von Aminosäuresequenzen mit bekannter Eigenschaft die für die Funktionsvorhersage relevanten physikochemischen Merkmale findet. Mit dieser Methode sollte es möglich sein, eine verläßliche Aussage über unbekannte Sequenzen bezüglich dieser Funktion zu machen. Zur Beantwortung dieser Fragestellung eignen sich Künstliche Neuronale Netze (KNN; [111, 82]). Diese Methode ist hier weiterentwickelt worden: Ein KNN mit adaptiver Kodierung (adaptive coding network, ACN) ist in der Lage, die wesentlichen Eigenschaften, die für alle Aminosäurepositionen in einem gegebenen Sequenzabschnitt gleich sind, zu bestimmen. Das heißt, es müssen nicht mehr alle physikochemischen Eigenschaften durchprobiert werden, um die beste Kombination zu finden, sondern nur wenige Trainingsläufe reichen aus, um das gegebene Problem elegant zu lösen. Damit bietet diese Methode die Möglichkeit mit einer sehr geringen Anzahl von zu bestimmenden Größen eine bessere Vorhersagegenauigkeit zu erzielen. Dieses ACN wird mit einfachen statistischen Methoden und mit herkömmlichen KNN verglichen.

Diese Methode kann anschließend auch für das rationale sequenzorientierte Protein-Design verwendet werden, z.B. mittels der Simulierten Molekularen Evolution (SME, [104]).

Die Analyse von bekannten Protein-Protein-Wechselwirkungen läßt Rückschlüsse auf das Design potentieller Inhibitor Peptide zu, die zur Entwicklung eines Modells genutzt wurden.

3.a Gütekriterien

Die Generalisierungsfähigkeit eines Klassifikators (eine Methode, die eine Klassifikation von Daten ermöglicht) kann mit Hilfe eines Trainings- und eines Testdatensatzes überprüft wer-

den. Die Vorhersage von Trainingssequenzen wird als Reklassifikation, die von Testsequenzen als Klassifikation bezeichnet. Mit P („positiv“) wird die Anzahl der positiven Sequenzen, die korrekt zugeordnet wurden, abgekürzt; N („negativ“) sind die negativen Sequenzen, die korrekt vorhergesagt werden, O („overprediction“) ist die Anzahl derjenigen Sequenzen, die als positiv vorhergesagt werden, jedoch negativ sind; U („underprediction“) sind die falsch negativ zugeordneten Sequenzen. Als Qualitätskriterium wurde der Korrelationskoeffizient (cc) nach Matthews verwendet:

Korrelationskoeffizient nach Matthews [77].

$$cc = \frac{P \cdot N - U \cdot O}{\sqrt{(P + O)(P + U)(N + U)(N + O)}} \qquad \text{(GLEICHUNG 1)}$$

Der cc hat einen Wertebereich von minus eins bis plus eins. Er gewichtet alle Vorhersagen gleich und berücksichtigt nicht die eventuellen Größenunterschiede der positiven und negativen Daten. In Abbildung 8A ist die cc-Funktion für den Fall der Signalsequenzen dargestellt. Hier sind Pos=76 positive und Neg=760 negative Sequenzen vorhanden. Die Overprediction (O) und Underprediction (U) ergeben sich nun durch:

$$\begin{aligned} O &= Neg - N \\ U &= Pos - P \end{aligned} \qquad \text{(GLEICHUNG 2)}$$

Dadurch ist cc als Funktion von P, N, O und U zu einer Funktion von P und N transformiert worden. Die Funktion ist im Bereich mittlerer Werte linear, nur an den Enden ergeben sich stärkere Abweichungen. Diese Abweichungen ergeben sich aus der ungleichen Anzahl von Positiven und Negativen Datenpunkten, wie ein Vergleich mit dem 1:1 Verhältnis zeigt (Abbildung 8B).

Im Falle der *cis*- bzw. *trans*-Prolyl-Gruppen sind Pos=324 bzw. Neg=5048 Daten vorhanden (Abbildung 9). Hier ist das Verhältnis von positiven zu negativen mit 1:15 noch etwas schlechter als bei den Schnittstellendaten (1:10). Der Verlauf der Korrelationsfunktion zeigt, daß in Bereichen von sehr wenigen oder sehr vielen N kleine Änderungen des Verhältnisses von P zu N relativ große Auswirkung auf die cc-Werte haben.

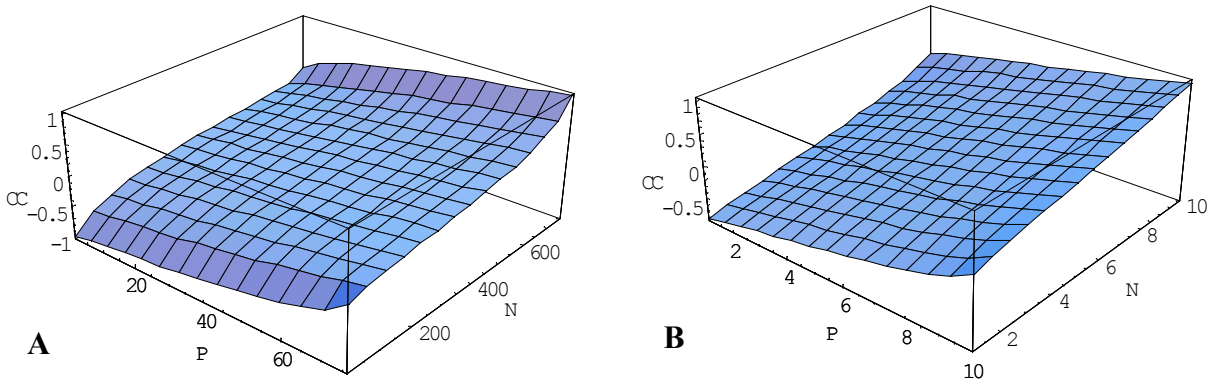

ABBILDUNG 8: cc-Werte als Funktion der Anzahl an positiven und negativen Beispielen. Die Overprediction und Underprediction ergeben sich aus der Anzahl aller positiven und negativen Daten: **A**: Pos=76; Neg=760; **B**: Pos=Neg=10;

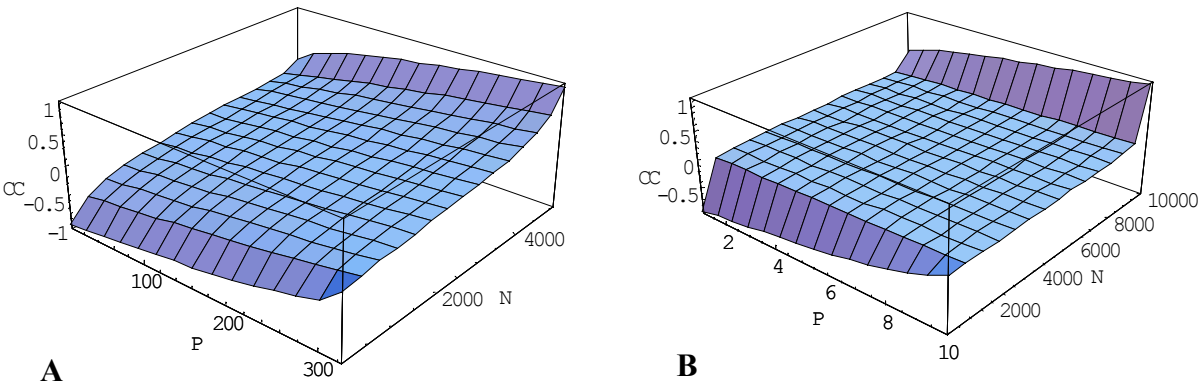

ABBILDUNG 9: cc-Werte als Funktion der Anzahl an positiven und negativen Beispielen. Die Overprediction und Underprediction ergeben sich aus der Anzahl aller positiven und negativen Daten: **A**: Pos=324; Neg=5048; **B**: Pos=10; Neg=10000.

3.b Kodierung der Daten

Die Proteinsequenzdaten liegen als Ketten von Buchstaben vor. Um sie mit einer mathematischen Methode auszuwerten, müssen die einzelnen Buchstaben in Zahlen transformiert werden. Für diese Kodierung gibt es mehrere Möglichkeiten. Alle Kodierungen ordnen jeder der 20 Aminosäuren eine oder mehrere Zahlen zu.

Bei der verteilten Kodierung wird jeder Aminosäure ein Vektor von einer Eins und neunzehn Nullen zugeordnet. Diese Kodierung hat den Vorteil, daß kein *a priori* Wissen in die Kodierung übernommen wird, denn alle Aminosäuren werden hier unabhängig voneinander kodiert; alle Vektoren sind orthogonal zueinander. Jedoch müssen für jede Aminosäure 20-dimen-

sionale Vektoren eingeführt werden. (Eigentlich müssen die Vektoren nur 19 dimensional sein, damit alle Aminosäuren voneinander unabhängig sind, jedoch ist die Beschreibung mit 20 Dimensionen einfacher und übersichtlicher.) Um einen Sequenzausschnitt von 12 Aminosäuren zu beschreiben, wird ein 240-dimensionaler Vektor benötigt.

Bei der Verwendung von binären physikochemischen Eigenschaften für die Kodierung werden die Aminosäuren durch eine Eins oder eine Null pro Eigenschaft repräsentiert. So werden die Aminosäuren A, G, S durch eine Eins und alle anderen durch eine Null ersetzt, wenn die Eigenschaft „sehr klein“ zur Beschreibung verwendet wird (Gleichung 3). Bei der Verwendung von reellen physikochemischen Eigenschaften werden durch Experimente bestimmte reelle Größen für die Aminosäuren eingesetzt. Der Nachteil der physikochemischen Kodierungen liegt in den Annahmen, auf denen die Kodierungen basieren. So kann es sein, daß die gewählte Kodierung nicht die charakteristischen Eigenschaften des Problems beschreibt.

A ⇒ {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
C ⇒ {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
D ⇒ {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
E ⇒ {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
F ⇒ {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
G ⇒ {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
H ⇒ {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
I ⇒ {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
K ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
L ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
M⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}
N ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}
P ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}
Q ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0}
R ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0}
S ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}
T ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0}
V ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0}
W⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0}
V ⇒ {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}

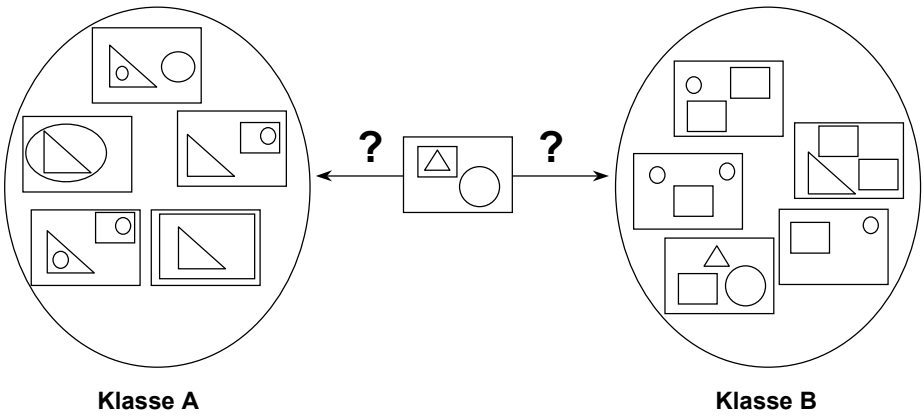

ABBILDUNG 11: Bongard-Problem: Gehört das in der Mitte abgebildete Symbol zur Klasse A oder zur Klasse B? Welches Merkmal ist entscheidend?

So wie bei einem Bongard Problem die Eigenschaften Kreis, Quadrat oder Dreieck nicht geeignet sind, eine Zuordnung des in der Mitte von Abbildung 11 abgebildeten Beispiels zu erreichen, können auch die gewählten Eigenschaften in biochemischen Fällen nicht geeignet sein, das gegebene Problem zu lösen. Zudem kann eine Eigenschaft alleine nicht ausreichen. Mehrere Eigenschaften können durch Zusammensetzen von Einzeleigenschaften entstehen. (Das Muster im gewählten Bongard Problem gehört zur Klasse A. Das zugrundeliegende Merkmal heißt: Zweifach umschlossene Fläche, oder ein Objekt in einem anderen Objekt.)

Soll ein Datensatz durch die Eigenschaften „sehr klein“ und „aliphatisch“ beschrieben werden, so werden Alanin, Glycin und Serin auf den Vektor

$$A, G, S \Rightarrow \hat{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{(GLEICHUNG 3)}$$

abgebildet; die Aminosäuren Leucin, Isoleucin und Valin werden durch

$$L, I, V \Rightarrow \hat{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \text{(GLEICHUNG 4)}$$

beschrieben, alle anderen durch

$$T, C, M, P, D, N, E, Q, K, R, H, F, Y, W \Rightarrow \hat{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \qquad \text{(GLEICHUNG 5)}$$

Die Sequenz AVG wird also auf

$$AVG \Rightarrow \vec{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \text{(GLEICHUNG 6)}$$

abgebildet.

In Abschnitt 3.g.iii wird im Zusammenhang mit Künstlichen Neuronalen Netzen beschrieben, wie eine geeignete Kodierung unter Beibehaltung eines möglichst kleinen Datenraums gefunden werden kann, ohne sich auf physikochemische Eigenschaften zu beschränken.

3.c Vergleich von Kodierungen

Der Raum, der von der verteilten Kodierung aufgespannt wird, ist der größte sinnvolle Raum, in dem die Sequenzen dargestellt werden können. In dieser Kodierung werden keine Korrelationen zwischen den Aminosäuren eingeführt. Alle anderen, kleineren Räume, bilden Unterräume des verteilt kodierten Raumes. Diese können nach wie vor die wesentlichen Zusammenhänge zwischen den Sequenzpositionen beinhalten. Es wird nun eine Möglichkeit vorgestellt, diese Unterräume miteinander zu vergleichen. Ein Unterraum wird durch ein Basissystem aufgespannt, wobei das Basissystem aus Basisvektoren besteht. Jeder Vektor hat die Dimension 20, wobei jede Komponente eine Aminosäure repräsentiert (in alphabetischer Reihenfolge des Einbuchstabenkodes). Ein Basissystem ist die Zusammensetzung von mehreren Kodierungsvektoren (Kodierungsmatrix). So besteht das Basissystem der verteilten Kodierung aus 20 Basisvektoren. Die Eigenschaft „sehr klein“ wird durch

$$\text{sehr klein} = \{1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0\} \qquad \text{(GLEICHUNG 7)}$$

beschrieben.

Es sollen nun verschiedene Basissysteme miteinander verglichen werden. Dabei ist zum einen interessant, inwieweit ein Basisvektor eines Basissystems in einem anderen Basissystem dargestellt wird und wie sich zwei Basissysteme gleicher Dimension zueinander verhalten. Es sollten beide Werte auf Eins normiert sein, um die Interpretation zu erleichtern.

Zunächst soll gezeigt werden, wie ein Basisvektor C in einem Basissystem $M = \{c_1, c_2, ..., c_N\}$ aus N Basisvektoren repräsentiert ist. Die Fragestellung anders formuliert lautet: Inwieweit stimmt die Richtung, die durch den Basisvektor gegeben ist, mit den Richtungen, die durch das Basissystem aufgespannt werden, überein. Da nicht die Größe der Korrelationen interessiert,

sondern nur die Qualität, also nicht die Länge der Vektoren sondern nur die Richtung, werden sowohl der Basisvektor als auch die Basisvektoren des Basissystems normiert:

$$\vec{e_C} = \frac{\vec{C}}{\sqrt{\vec{C} \cdot \vec{C}}}$$
$$\vec{E_M} = \left\{ \frac{\vec{c_1}}{\sqrt{\vec{c_1} \cdot \vec{c_1}}}, \frac{\vec{c_2}}{\sqrt{\vec{c_2} \cdot \vec{c_2}}}, \ldots, \frac{\vec{c_N}}{\sqrt{\vec{c_N} \cdot \vec{c_N}}} \right\} \qquad \text{(GLEICHUNG 8)}$$

mit $\vec{e_C}$ als Einheitsvektor und $\vec{E_M}$ als normiertem Basissystem. Anschließend werden die Basisvektoren des Basissystems orthogonalisiert nach Gram-Schmidt [54].

$$\vec{M_G} = \text{GramSchmidt}(\vec{E_M}) \qquad \text{(GLEICHUNG 9)}$$

Die resultierenden Vektoren sind jetzt nicht mehr normiert. Ihre Länge spiegelt die Korrelation zwischen ihnen wider. Das Euklidische Skalarprodukt zwischen den einzelnen Basisvektoren und dem Basissystem ergibt einen Vektor dessen Betrag, D, den gemeinsamen Anteil zwischen Basis und Basissystem quantifiziert.

$$D = \sqrt{(\vec{M_G} \cdot \vec{e_C}) \cdot (M_G \cdot \vec{e_C})} \qquad \text{(GLEICHUNG 10)}$$

Der Wert von D liegt zwischen Null und Eins.

Zwei Basissysteme können wie folgt miteinander verglichen werden: Das Skalarprodukt zwischen zwei n x m-Matrizen, A und B, ist über die Spur aus dem Produkt der beiden Matrizen definiert [54]:

$$\langle A, B \rangle = \text{Spur}(A \cdot B^T)$$
$$\text{Spur}(A) = \sum_i A_{i,i} \quad , \qquad \text{(GLEICHUNG 11)}$$

wobei T der Transpositionsoperator ist. Anschließend wird die Normierung durchgeführt, welche den Cosinus des Winkels zwischen den zwei Matrizen darstellt:

$$\cos(A, B) = \frac{\langle A, B \rangle}{\sqrt{\langle A, A \rangle \cdot \langle B, B \rangle}} \qquad \text{(GLEICHUNG 12)}$$

Dieser Wert liegt somit wieder zwischen Null und Eins. Der Cosinus ist jedoch nicht unabhängig von der Reihenfolge der einzelnen Basen. Jedoch ist die Reihenfolge für den Vergleich von zwei Kodierungen nicht relevant, denn eine Kodierung (hydrophob, polar) sollte gleich (polar, hydrophob) sein. Deshalb müssen alle möglichen Permutationen für eines der Basissysteme

durchprobiert werden. Der maximale Wert ergibt dann den gewünschten Wert für den Vergleich von zwei Basissystemen.

3.d Informationsanalyse

Bevor komplexe Analyseverfahren angewendet werden, sollte der Datensatz auf Besonderheiten überprüft werden. So kann mittels der Informationsanalyse [44, 112, 113] überprüft werden, ob die Sequenzpositionen Aminosäuren gleichverteilt enthalten, oder ob bestimmte Aminosäuren an einzelnen Positionen häufiger vorkommen.

Die Wahrscheinlichkeit, daß die Aminosäure Alanin an einer bestimmten Position, i, in der Sequenz vorkommt, $p_i(A)$, berechnet sich aus der Häufigkeit von Alanin an der Position i, $Anz_i(A)$, über die Anzahl der Beispielsequenzen, Anz(Sequenzen), die vorhanden sind:

$$p_i(A) = \frac{Anz_i(A)}{Anz(\text{Sequenzen})} \qquad \text{(GLEICHUNG 13)}$$

Dabei ist die Summe der Wahrscheinlichkeiten aller 20 Aminosäuren gleich Eins $\left(\sum_{X=A}^{Y} p_i(X) = 1\right)$. Der Grad der Unbestimmtheit oder auch die Entropie eines Zustandes (Position i und Aminosäure X) berechnet sich zu

$$S_i(X) = -p_i(X) \cdot \log(p_i(X)) . \qquad \text{(GLEICHUNG 14)}$$

Für den Fall, daß der Logarithmus zur Basis 2 berechnet wird, ist die Entropie an der gegebenen Sequenzposition gleich der Anzahl der binären Entscheidungen, die getroffen werden müssen, um den Zustand zu bestimmen. Diese Größe wird in bit gemessen. Für den Fall, daß an einer bestimmten Sequenzposition nur die Aminosäuren Alanin und Glycin auftreten, gibt es nur zwei mögliche Zustände. Die Wahrscheinlichkeiten p_i für dieses Szenario könnten $p_i(A) = 0,5$ und $p_i(G) = 0,5$ sowie $p_i(\text{Rest}) = 0,0$ sein. Damit ergibt sich für die Information $0.5 * -\log_2(0,5) + 0.5 * -\log_2(0,5) = 1$. Es reicht also eine Frage, um den Zustand zu bestimmen: Ist es Alanin? Wenn ja, dann ist es Alanin, wenn nein, dann ist es Glycin.

Die auf Eins normierte Information gibt ebenfalls ein Maß für die Bestimmtheit oder Unbestimmtheit eines Systems an und ist durch

$$\text{Information} = -\frac{1}{\log(20)} \sum_{X=A}^{Y} p_i(X) \cdot \log(p_i(X)) \qquad \text{(GLEICHUNG 15)}$$

gegeben.

Große Werte bedeuten eine große Unbestimmtheit, kleine Werte sagen aus, daß wenige Entscheidungen nötig sind, um den Zustand zu bestimmen. Bei einem kleinen Informationswert ist es eher möglich, Regeln für eine Klassifikation abzuleiten, als bei großen Werten.

3.e Lineare Trennverfahren

Lineare Trennverfahren gehören zu den einfachsten Methoden für die Klassifikation von Datenmengen. Hier werden die Projektion auf die Schwerpunktsgerade und der Bayes-Prediktor vorgestellt.

3.e.i Schwerpunktanalyse (Zentroid-Verfahren)

Unterscheiden sich zwei Datenmengen in einem gegebenen Raum, so können die Daten anhand der Projektion auf die Gerade, die die beiden Schwerpunkte (Zentroide) verbindet, klassifiziert werden.

Der Schwerpunkt einer Klasse mit M Daten, $\vec{x}$, der Dimension N ist durch

$$\vec{\bar{x}} = \begin{bmatrix} \bar{x}_1 \\ \dots \\ \bar{x}_N \end{bmatrix} \text{ mit } \bar{x}_1 = \frac{1}{M}\sum_{j=1}^{M} x_{1,j} \qquad \text{(GLEICHUNG 16)}$$

definiert. Der Richtungsvektor der Schwerpunktsgeraden ist

$$\vec{s} = \frac{\vec{\bar{y}} - \vec{\bar{x}}}{\sqrt{\sum_{j=1}^{M} (\bar{y}_j - \bar{x}_j)^2}}. \qquad \text{(GLEICHUNG 17)}$$

Damit ergibt sich für die Länge der Projektion eines Vektors $\vec{v}$ auf die Schwerpunktsgrade:

$$p(\vec{v}) = (\vec{v} - \vec{\bar{x}}) \cdot \vec{s}. \qquad \text{(GLEICHUNG 18)}$$

Liegen die Daten in zwei getrennten Wolken vor, oder ist eine der Mengen auf einen Punkt konzentriert, so kann ein Schwellenwert gefunden werden, der eine Klassifikation erlaubt. p ist somit der Abstand der einzelnen Punkte vom Schwerpunkt einer Datenmenge entlang der Verbindungsgeraden.

3.e.ii Bayes-Prediktor

Der Bayes-Prediktor nutzt vorhandene Informationen, um eine Klassifizierung durchzuführen. Angewendet auf Aminosäuresequenzen bedeutet das, daß die Häufigkeit von Aminosäuren (X) an den verschiedenen Positionen (i) der Sequenzen eines bekannten Datensatzes genutzt wird, um auf unbekannte Sequenzen zu schließen. Dabei werden für jede Klasse (α, β) nach Gleichung 13 die relativen Häufigkeiten berechnet. Die Logarithmen der Häufigkeiten berechnen sich nach

$$q^{\alpha}(i, X) = \log(p_i^{\alpha}(X)), \qquad \text{(GLEICHUNG 19)}$$

wobei i über die Positionen und X über die 20 Aminosäuren geht.

Für eine unbekannte Sequenz werden die entsprechenden $q^{\alpha}(i, X)$ für jede Klasse summiert. Die Sequenz wird der Klasse α zugeordnet, wenn die folgende Ungleichung erfüllt ist:

$$\sum_{i, j} (q^{\alpha}(i, X) - q^{\beta}(i, X)) > \log \frac{Anz(\beta)}{Anz(\alpha)}, \qquad \text{(GLEICHUNG 20)}$$

andernfalls wird sie der Klasse β zugeordnet.

3.f Mahalanobis-Distanzanalyse

Die Mahalanobis-Distanz [73] kann dazu benutzt werden, die Ähnlichkeit von Proteinen zu bestimmen [13]. Wie oben beschrieben, ergeben sich bei einer Kodierung mit C Kodierungsvektoren in einem Fenster mit N Aminosäureresten (N*C) Dimensionen. Liegen M Proteine vor, so gilt für die einzelnen Punkte:

$$\vec{x}_j = \begin{bmatrix} x_{j, 1} \\ x_{j, 2} \\ \cdots \\ x_{j, C \cdot N} \end{bmatrix}, \; j = (1, 2, \ldots, M) \qquad \text{(GLEICHUNG 21)}$$

wobei $x_{j, 1}, x_{j, 2}, \ldots x_{j, C \cdot N}$ die einzelnen Kodierungen (1 bis C) an den Positionen (1 bis N) repräsentieren.

Der Mittelwert wird nach

$$\vec{\bar{x}} = \frac{1}{M} \sum_{i = 1}^{M} \vec{x}_i \qquad \text{(GLEICHUNG 22)}$$

berechnet.

Die Konnektivität der einzelnen Kodierungsvektoren untereinander kann durch eine Korrelationsmatrix, S, berücksichtigt werden. Korrelationsterme werden in der Matrix

$$S = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,N} \\ s_{1,1} & s_{1,1} & \cdots & s_{1,N} \\ \cdots & \cdots & \cdots & \cdots \\ s_{1,1} & s_{1,1} & \cdots & s_{1,N} \end{bmatrix} \qquad \text{(GLEICHUNG 23)}$$

mit $s_{i,m} = \frac{1}{M}\sum_{j=1}^{M}(x_{j,i} - \overline{x_i})(x_{j,m} - \overline{x_m})$, $(i, m = 1, 2, \ldots, N)$

zusammengefaßt. Aus Gleichung 23 ist zu erkennen, daß die Diagonalelemente der Matrix S (m = i) die Summen der quadratischen Abstände zum Schwerpunkt $(\overline{X})$ für die einzelnen Eigenschaften und Positionen sind. Die Nicht-Diagonalelemente sind in der Regel von Null verschieden. Diese Terme beschreiben den Zusammenhang jeweils zweier Elemente.

Die Mahalanobis-Distanz, D, kann nun wie folgt definiert werden:

$$D^2(\hat{x}, \hat{\overline{x}}) = (\hat{x} - \hat{\overline{x}})^T S^{-1} (\hat{x} - \hat{\overline{x}}) \qquad \text{(GLEICHUNG 24)}$$

wobei S^{-1} die Inverse der Matrix S darstellt.

Eine unbekannte Sequenz kann mit Hilfe der Mahalanobis-Distanz einer von mehreren Sequenzklassen zugeordnet werden. Dazu wird zuerst die S-Matrix der gegebenen Sequenzklasse aufgestellt. Nun kann die Mahalanobis-Distanz der zu untersuchenden Sequenz von allen bekannten Klassen berechnet werden. Die gesuchte Gruppe ist dabei diejenige, zu der der kleinste Abstand besteht:

$$D(\hat{x}, {}^{q}\hat{\overline{x}}) = \mathrm{Min}\{D((\hat{x}, {}^{\text{Klasse 2}}\hat{\overline{x}}), D(\hat{x}, {}^{\text{Klasse 1}}\hat{\overline{x}}))\} \qquad \text{(GLEICHUNG 25)}$$

wobei q die Klasse mit dem kleinsten Abstand und $\hat{x}$ der zu untersuchende Vektor ist. Der Operator Min liefert das kleinste seiner Argumente. Gleichung 25 besitzt alle grundlegenden Eigenschaften, die ein Abstand haben sollte: Angenommen A, B und C sind drei Punkte im Raum, in dem die Mahalanobis-Distanz definiert ist, so folgt daraus: 1) $D(A, B) \geq 0$, wobei das Gleichheitszeichen nur für den Fall $A \equiv B$ gilt; 2) $D(A, B) \equiv D(B, A)$; und 3) $D(A, B) + D(B, C) \geq D(A, C)$ [13].

Weitere Beispiele für die Anwendung der Mahalanobis-Distanzanalyse enthalten die Arbeiten von Chou et al. [13].

3.g Künstliche Neuronale Netze

Die bisher beschriebenen Methoden verwenden statistische Maße wie Mittelwerte um Klassifikationsfunktionen abzuleiten. Eine flexible Möglichkeit, sowohl lineare als auch komplexere Zusammenhänge darzustellen, bieten Künstliche Neuronale Netze (KNN). Sie sind eine mathematische Vorschrift, um Daten in einem hochdimensionalen Raum in einen niedriger dimensionalen Raum abzubilden. Mit ihrer Hilfe ist es oft möglich, eine Sequenz-Aktivität-Relation (SAR) zu beschreiben. Dabei dienen Sequenzdaten als Eingabedaten und der dazugehörige Ausgabewert quantifiziert den Grad einer biologischen Funktionialität. Hier wird eine Klassifikationsfunktion so gewählt, daß eine bestmögliche Unterscheidung zwischen zwei Klassen getroffen werden kann. Zuerst wird anhand des einfachsten KNN, des Perzeptrons gezeigt, welche mathematischen Funktionen hinter dieser Methode stehen. Anschließend werden mehrlagige KNN und schließlich ein neues KNN mit adaptiver Kodierung vorgestellt. Nach der Beschreibung der Funktion der KNN werden Möglichkeiten vorgestellt, die freien Parameter zu bestimmen. Auch Selbstorganisierende Karten (SOM, oder auch Kohonenkarten) gehören zu den Künstlichen Neuronalen Netzen. Die Gewichte oder freien Parameter von KNN können einfach bildlich dargestellt werden.

3.g.i Perzeptron

Das Perzeptron (erstmals vorgestellt von Rosenblatt [97]) ist das einfachste KNN (Abbildung 12). Es besteht aus dem Eingabevektor ($\vec{k} = (k_1, \ldots, k_m)^T$) und einem Ausgabevektor ($\vec{A} = (A_1, \ldots, A_q)^T$). Da es sich bei dem Ausgabevektor in dieser Arbeit immer um ein Skalar handelt ($q = 1$), gilt $\vec{A} = A$. Die Eingabewerte werden dann im Perzeptron mit einem Gewichtsvektor ($\vec{w}$) multipliziert. In der Ausgabeschicht werden diese Ergebnisse addiert.

$$\begin{aligned} A &= \vec{k}\vec{w} \\ &= k_1 w_1 + k_2 w_2 + \ldots + k_m w_m \\ &= \sum_{i=1}^{m} k_i w_i \end{aligned} \qquad \text{(GLEICHUNG 26)}$$

Diese Gerade verläuft durch den Ursprung. Um diese Einschränkung zu umgehen, wird ein weiterer Parameter, der Schwellenwert (θ), eingeführt, womit sich die allgemeine Geradengleichung ergibt:

$$A = \vec{k}\vec{w} - \theta \qquad \text{(GLEICHUNG 27)}$$

Die Ausgabeeinheit wird in der Regel noch durch eine Transferfunktion verrechnet, die die Ausgabewerte auf einen bestimmten Wertebereich abbildet. Durch das Einführen einer nicht-linearen Transferfunktion, werden in das KNN Nicht-Linearitäten eingeführt, die das Approximieren beliebiger Funktionen ermöglichen. In dieser Arbeit wird die hyperbolische Tangensfunktion ($Tanh(x)$) verwendet, die einen Wertebereich von {-1,1} besitzt. Damit ergibt sich für das einfachste Neuronale Netz

$$A = Tanh(\vec{k}\vec{w} - \theta) \qquad \text{(GLEICHUNG 28)}$$

Wird nun dem Eingabeneuron ein Eingabevektor $\vec{k}$ übergeben, ergibt sich ein Wert zwischen minus Eins und Eins. Dieser Wert muß anschließend bezüglich der biologischen Fragestellung bewertet werden.

Für die Bewertung unterschiedlicher Netze wird der quadratische Fehler als Summe der Differenzen des Ausgabewertes und des erwarteten Sollwerts über alle Eingabevektoren berechnet. Das Netz mit dem geringsten quadratischen Fehler ist von allen betrachteten Netzen am besten zur Beschreibung des Problems geeignet.

Für die Zuordnung zu einer bestimmten Klasse wird die Sprungfunktion, Θ, eingeführt, die Eins ergibt, wenn der Wert größer oder gleich Null ist und andernfalls Null:

$$\Theta(x) = \begin{cases} 0 & \text{für } x \geq 0 \\ 1 & \text{für } x < 0 \end{cases} \qquad \text{(GLEICHUNG 29)}$$

Null steht somit für eine Klasse und Eins für eine andere Klasse.

Die Gewichte $\vec{w}$ und der Wert θ sind freie Parameter und müssen so gewählt werden, daß möglichst viele Eingabewerte $\vec{k}$ den richtigen Gruppen zugeordnet werden. In einem Lernverfahren werden die Gewichtsparameter einem Datensatz angepaßt, bei dem die Zuordnung schon bekannt ist (überwachtes Lernen). Auf verschiedene Lernverfahren wird in Abschnitt 3.g.iv näher eingegangen.

Anhand eines kurzen Rechenbeispiels soll die Funktionsweise eines Perzeptrons erklärt werden (vergleiche hierzu Abbildung 12 und 13). Gegeben seien die drei zweidimensionalen Vektoren $\vec{x}_1 = \{0{,}75;1{,}5\}$, $\vec{y}_2 = \{0{,}0;\ 1{,}0\}$ und $\vec{x}_3 = \{1{,}5;0{,}5\}$, wobei $\vec{x}_1$ und $\vec{x}_3$ zu einer Gruppe gehören sollen und $\vec{y}_2$ zu einer anderen. Die Vektoren $\vec{x}_1$, $\vec{y}_2$ und $\vec{x}_3$ werden dem Eingabevektor des Perzeptrons $\vec{k}$ übergeben. Diese Werte werden mit dem Gewichtsvektor $\vec{w}$ multipli-

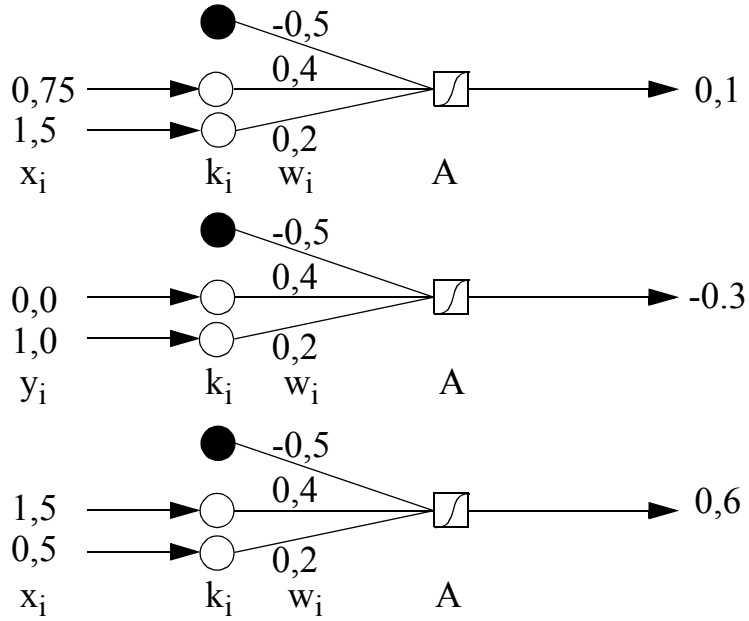

ABBILDUNG 12: Perzeptron mit drei Eingabevektoren $\vec{k}$, dem Gewichtsvektor w={0,4; 0,2; $\theta = -0,5$ } der Ausgabeschicht A mit Tanh als Transferfunktion und dem Ergebnis. Weiße Kreise bilden die Eingabeneuronen, schwarze den Wert -1. Die Verrechnungseinheit mit der Transferfunktion wird durch Quadrate symbolisiert.

ziert und der Ausgabeeinheit (A) übergeben. Hier wird die Summe über alle Produkte gebildet und die Transferfunktion angewendet.

Der Lernalgorithmus (Abschnitt 3.g.iv) sucht nun eine Gerade, welche die Punktgruppen möglichst korrekt aufteilt. Dabei werden die Variablen w_1, w_2 und θ der Geradengleichung

$$K_1 w_1 + K_2 w_2 - \theta = 0 \qquad \text{(GLEICHUNG 30)}$$

so optimiert, daß eine Separation erfolgen kann. Es gelten weiterhin folgende Bedingungen:

$$\begin{array}{l} \text{Gruppe 1: } A \geq \theta \\ \text{Gruppe 2: } A > \theta \end{array} \quad \text{mit } \vartheta = 0 \qquad \text{(GLEICHUNG 31)}$$

Nach einer zufälligen Initialisierung der Gewichte wurde die Evolutionsstrategie [92] verwendet, um die Gewichte zu optimieren.

Ein mögliches Ergebnis ist in Abbildung 13 für einen Schwellenwert von $\theta = -0{,}5$ dargestellt. Die daraus folgende Separationsgerade ist ebenfalls dargestellt.

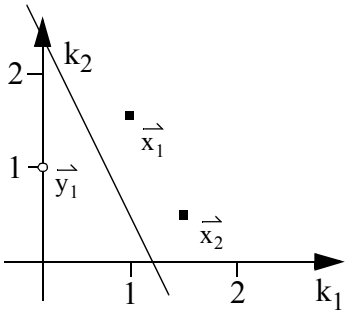

ABBILDUNG 13: Darstellung der Vektorpunkte und der Separationsgeraden im Raum von k_1 und k_2.

Mit Gleichung 30 und Gleichung 31 ergibt sich für das Beispiel

$$\begin{aligned} 0{,}4k_1 + 0{,}2k_2 &= 0{,}5 \\ \Leftrightarrow 2{,}5 - 2k_1 &= k_2 \end{aligned} \qquad \text{(GLEICHUNG 32)}$$

mit $\vartheta = -0{,}5$, $w_1 = 0{,}4$ und $w_2 = 0{,}2$ als Parameter der Separationsgerade.

3.g.ii Mehrlagige Neuronale Netze

Die allgemeine Formel für ein einlagiges Neuronales Netz mit einem Ausgabewert ist:

$$A = f(\vec{k}\vec{w} - \theta) \text{ mit } f(x) = \mathrm{Tanh}(x). \qquad \text{(GLEICHUNG 33)}$$

Durch Einführen von weiteren Schichten (Hiddenschichten) werden mehrlagige Netze erzeugt. Mit einem zweilagigen KNN können konkave Funktionen approximiert werden, wie aus Gleichung 34 ersichtlich ist:

$$A = f(\vec{w}' \cdot f_p(\vec{k}W - \vec{\theta}) - \theta') \qquad \text{(GLEICHUNG 34)}$$

mit $f_p(\vec{x}) = f(x_1, \ldots, x_p)$ und p der Anzhal der Hiddenneuronen (siehe Abbildung 14). W ist eine (m, p)-Gewichtsmatrix (m = Dimension des Eingabevektors). Hier treten die Gewichte als quadratischer Term auf.

Mit dreilagigen Netzen können beliebige Funktionen approximiert werden [33]. Die Komponenten der einzelnen Gewichtsvektoren können sowohl positive als auch negative Werte annehmen.

$$A = f(\vec{w}'' \cdot f_q(f_p(\vec{k}W - \vec{\theta}) + W' \cdot -\vec{\theta}') - \theta'') \qquad \text{(GLEICHUNG 35)}$$

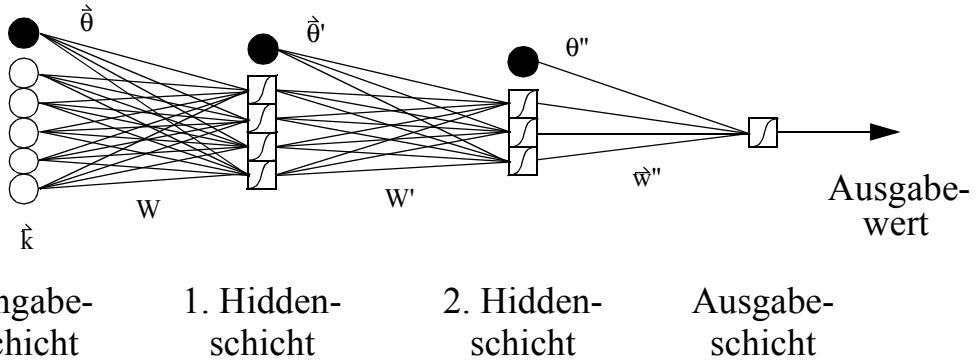

ABBILDUNG 14: Schematische Darstellung eines dreilagigen Netzes. Weiße Kreise bilden die Eingabeneuronen, schwarze den Wert -1. Die Quadrate bilden die Verrechnungseinheiten, die die Summen der Produkte aus dem Gewicht und dem entsprechenden Neuron der Schicht davor darstellen und übergibt diese der Transferfunktion (z.B. Tanh). Die Gewichte werden durch Verbindungslinien dargestellt.

Ein großer Vorteil der KNN ist die einfache Darstellungsmöglichkeit. Abbildung 14 stellt ein dreilagiges Netz dar. Sollen in komplizierteren KNN Verbindungen zwischen bestimmten Neuronen nicht verwendet werden, kann dies durch Weglassen der entsprechenden Linien dargestellt werden.

3.g.iii Neuronale Netze mit adaptiver Kodierung

Die Kodierung der Daten ist entscheidend für eine zuverlässige Merkmalsextraktion. Für Aminosäuren gibt es eine Reihe von Beschreibungsformen, z.B. die physikochemische Eigenschaften. Für die Auswahl einer geeigneten Kodierung kann der Anwender die Kodierung vorgeben oder das Netz wählt maschinell aus einem vorgegebenen Satz unter selbstoptimierenden Bedingungen die geeignete aus. Hier wird eine noch weitergehendere Auswahl einer geeigneten Kodierung vorgestellt: die selbstadaptive Kodierung. Die Bestimmung der geeigneten Kodierung wird in den Lernprozeß des KNN mit einbezogen.

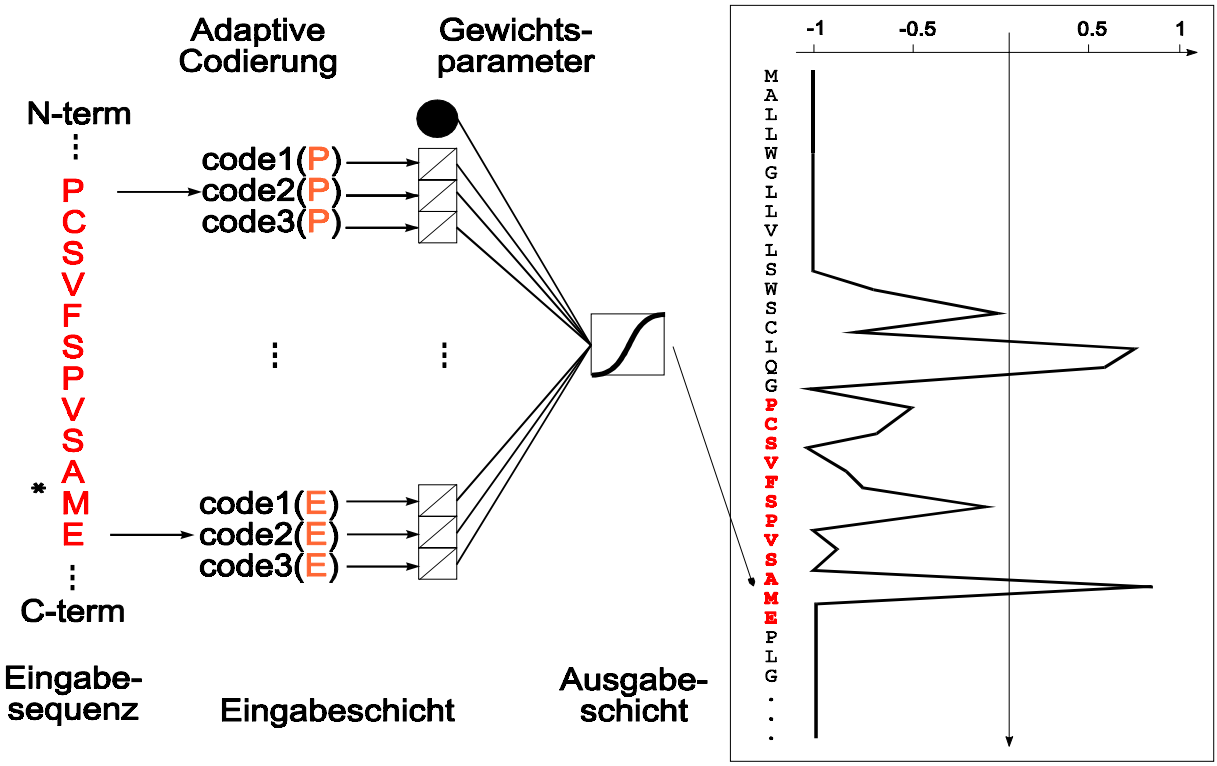

ABBILDUNG 15: Neuronales Netz mit adaptiver Kodierung (ACN) angewendet auf Sequenzdaten. Eine Sequenz wird mittels einer Kodierungstabelle (code1(X), code2(X), code3(X), wobei X eine der 20 Aminosäuren darstellt) in reelle Zahlen umgewandelt. Die Werte dieser Kodierungstabelle werden während des Lernprozesses optimiert. Die Verrechnungseinheiten (Quadrate) dieser Eingabeschicht besitzen keine Transferfunktion (f(x)=x). Entsprechend eines herkömmlichen KNN werden die Werte in der nächsten Schicht (Ausgabeschicht) mittels der Transferfunktion (f(x)=Tanh(x)) verrechnet. Die Ausgabe des ACN für die Sequenz von α-2-AntiPlasmin Precursor (A2AP-HUMAN) ist gezeigt. Es können auch mehrere Schnittstellen in der Sequenz vorkommen. Die diskreten Punkte sind mit Linien verbunden, um eine Interpretation zu vereinfachen.

Gleichung 36 beschreibt die Erweiterung eines einlagigen Netzes, mit N_c verschiedenen Kodierungen in einem Sequenzfenster der Länge N_s. Die adaptiven Eigenschaften können natürlich auch auf komplexere Neuronale Netze angewendet werden.

$$A = \mathrm{Tanh}\left(\sum_{n=1}^{N_c} \sum_{i=1}^{N_s} \mathrm{Cod}_n(X_i) \cdot w_{i,n} - \theta \right) \qquad \text{(GLEICHUNG 36)}$$

Ein Vorteil dieser Netzarchitektur in einem Perzeptron bietet die einfache Analysemöglichkeit. Die neuen Kodierungen können mit bekannten verglichen werden und die Architektur des Perzeptron erlaubt eine einfache Analyse der Gewichte. Dennoch ist dieses Verfahren wesentlich flexibler als das einfache Perzeptron.

3.g.iv Lernstrategien

Im folgenden werden die beiden verwendeten Algorithmen zur Bestimmung der optimalen Gewichte der Neuronalen Netze beschrieben. Die Optimierung der freien Parameter (Gewichte) wird auch Lernen genannt.

Der Backpropagation Lernalgorithmus führt den quadratischen Fehler (Abschnitt 3.g.i) eines Neuronalen Netzes auf seine Gewichte zurück. Im folgenden wird die Anwendung des Algorithmus auf die Neuronalen Netze mit adaptiver Kodierung (Gleichung 36) vorgestellt. Dabei wird der Schwellenwert θ in den Gewichtsvektor mit einbezogen:

$$\begin{aligned} \vec{w} &= \{w_1, w_2, \ldots, w_m, \theta\} \\ &= \{w_1, w_2, \ldots, w_m, w_{m+1}\} \end{aligned} \qquad \text{(GLEICHUNG 37)}$$

Die Kostenfunktion ($E(w)$), in unserem Fall der durchschnittliche quadratische Fehler über alle N_m Eingabemuster $\vec{x}^{\mu}$, ergibt sich als Funktion aller Gewichte:

$$\begin{aligned} E(w) &= \frac{1}{2N_m}\sum_{\mu}^{N_m}(A^{\mu}-\zeta^{\mu})^2 \\ &= \frac{1}{2N_m}\sum_{\mu}^{N_m}(\mathrm{Tanh}(h^{\mu})-\zeta^{\mu})^2 \end{aligned} \qquad \text{(GLEICHUNG 38)}$$

mit $h^{\mu} = \sum_{n=1}^{N_c}\sum_{i=1}^{N_s} \mathrm{Cod}_n(X_i)\cdot w_{i,n}$.

A ergibt sich aus Gleichung 36 und ζ ist der Sollausgabewert, für die jeweilige Beispielsequenz μ.

Die Änderung der Gewichte ergibt sich über ein Gradientenabstiegsverfahren:

$$\begin{aligned} \Delta w_{i,n} &= -\eta\frac{\partial E}{\partial w_{i,n}} \\ &= \eta\frac{1}{N_m}\sum_{\mu}^{N_m}(A^{\mu}-\zeta^{\mu})\mathrm{Tanh}'(h^{\mu})(\mathrm{Cod}_n(X_i)) \end{aligned} \qquad \text{(GLEICHUNG 39)}$$

wobei η die Lernschrittweite angibt, die im Laufe des Training langsam gegen Null konvergiert.

Somit berechnen sich die neuen Gewichte zu:

$$w_{i,n}^{neu} = w_{i,n}^{alt} - \Delta w_{i,n} \qquad \text{(GLEICHUNG 40)}$$

Die Kodierungen sind nicht von i abhängig und es gibt auch keinen Parameter entsprechend dem θ. Dadurch ergibt sich analog zu Gleichung 39:

$$\begin{aligned}\Delta Cod_n(X_i) &= -\eta\frac{\partial E}{\partial Cod_n(X_i)} \\ &= \eta\frac{1}{N_m \cdot N_s}\sum_{\mu}^{N_m}(\zeta^{\mu} - A^{\mu})Tanh'(h^{\mu})w_{i,n}\end{aligned} \qquad \text{(GLEICHUNG 41)}$$

und für die neue Kodierung entsprechend:

$$Cod_n(X_i)^{neu} = Cod_n(X_i)^{alt} - \Delta Cod_n(X_i) \qquad \text{(GLEICHUNG 42)}$$

Der zweite Algorithmus hat den Vorteil, daß das Konzept der Evolutionsstrategie direkt auf jede Netzarchitektur übertragen werden kann. Die Evolutionsstrategie [92, 109] ist ein Optimierungsverfahren, das von der Natur abgeschaut wurde. In der Natur wird aus einer unvorstellbar großen Anzahl von Möglichkeiten eine begrenzte Anzahl von funktionellen Proteinen gefunden. So gibt es für ein relativ kleines Protein mit einer Länge von 100 Aminosäuren 20^{100} verschiedene Realisierungen. Die Auswahl der funktionalen Proteine geschieht hier durch Mutation und Selektion. Bei der Bestimmung der optimalen Gewichte eines Neuronalen Netzes muß ebenfalls aus einer sehr großen Anzahl von Möglichkeiten eine geeignete Kombination gefunden werden. Ausgehend von zufällig gewählten Parametern (Gewichts- und Kodierungsvektoren) werden diese Werte variiert, um λ Nachkommen zu erzeugen. Jedes Individuum besitzt einen eigenen Satz von Parametern, sowie einen zusätzlichen Parameter (σ), der die Lernschrittweite oder die Variation der Nachkommen bestimmt. Für jeden Satz (λ Stück) an Parametern werden die Resultate des ACN für die Trainingsdaten bestimmt. Über den quadratischen Fehler läßt sich der beste Satz an Parametern bestimmen. In der vorliegenden Arbeit wurde immer das beste Netz für die Generierung der Nachkommen verwendet, ohne dieses in die nächste Generation mit aufzunehmen. Diese Vorgehensweise wird als (1, λ)-Strategie („1 Komma lambda Stategie“) bezeichnet. Bei einer (n+λ)-Strategie („1 Plus lambda Stategie“) werden die n besten Parametersätze zur Generation der Nachkommen verwendet und in die nächste Generation mitgenommen. Gleichung 43 zeigt die Berechnungsvorschrift für ein neues Gewicht w_{neu}; dabei sind u und u' Zufallszahlen zwischen Null und Eins.

$$w_{neu} = w_{alt} + \eta\sqrt{-2\log(u)}\sin(2\pi u') \qquad \text{(GLEICHUNG 43)}$$

η, die Lernrate, wird ebenfalls im Laufe der Zeit verändert. Hierbei erhält jeweils 1/3 der Nachkommen den Wert $1{,}3\eta$, 1/3 behält die Lernrate und 1/3 erhält den Wert $0{,}7\eta$. So ist

gewährleistet, daß das System zu einer Lernrate von Null konvergiert, da sich kleinere Lernraten für das Erlernen von detaillierten Gewichten besser eignen.

Dieses Verfahren wird solange wiederholt bis eine bestimmte Güte erreicht ist oder eine bestimmte Anzahl von Zyklen durchlaufen wurde.

3.g.v Identifikation von Schnittstellen in Sequenzen unter Verwendung von Künstlichen Neuronalen Netzen

Ein trainiertes KNN sollte in der Lage sein, die richtige Schnittstelle zu erkennen. Es ist jedoch nicht ausgeschlossen, daß auch weitere Sequenzabschnitte als potentielle Schnittstellen erkannt werden (Abbildung 15). Wie kann die richtige von den falschen Sequenzen unterschieden werden? Da das Signalpeptid am N-Terminus des Präproteins lokalisiert ist, sind nur die jeweils ersten 60 Aminosäurepositionen berücksichtigt worden. Die längste Vorläufersequenz im verwendeten Datensatz ist 48 AS lang. Beim Training der Netze wurden als negative Datenpunkte die ersten 10 Aminosäurefenster des maturen Proteins verwendet. Deshalb sollte, zumindest 10 Positionen in Richtung C-Terminus von einer potentiellen Schnittstelle, keine weitere Schnittstelle vorkommen.

Da davon ausgegangen wird, daß die betrachteten Proteine eine Schnittstelle besitzen, kann somit der Algorithmus zur Selektion der korrekten Schnittstelle wie folgt beschrieben werden:

i) Bestimme alle potentiellen Schnittstellen der ersten 60 Aminosäuren (z.B. durch Anwenden des KNN).

ii) Beginne am N-Terminus und suche die nächste potentielle Schnittstelle, solange es innerhalb der nächsten 10 Positionen eine weitere potentielle Schnittstelle gibt.

iii) Markiere die letzte so ermittelte potentielle Schnittstelle als korrekt und beende den Algorithmus.

3.g.vi Selbstorganisierende Karten (Kohonenkarten, SOK)

Kohonenkarten [56, 106] sind mathematische Funktionen, die im Gegensatz zu den KNN, durch einen nicht-überwachten Lernprozeß trainiert werden. Sie bilden einen hochdimensionalen Eingabevektor auf einen niedrigdimensionalen Raum (1 oder 2 dimensional) ab. Wenn Beziehungen zwischen einzelnen Eingabevektoren in niedriger Ordnung vorliegen, werden diese Werte auf einer elastischen Oberfläche benachbart angeordnet. Diese Oberfläche wird durch Referenzpunkte aufgespannt, die durch die Eingabedaten definiert werden. Eine Abbil-

dung aus einem hochdimensionalen Raum auf einen niedrig dimensionalen Raum ist somit definiert. Solch eine Abbildung kann dazu benutzt werden, geordnete Beziehungen in einer Eingabemenge zu visualisieren. Wie andere nicht-überwachte Klassifikationsmethoden ist auch die Kohonenkarte in der Lage, Cluster in der Menge der Eingabevektoren zu lokalisieren und einen unbekannten Datenvektor mit einem dieser Cluster zu identifizieren.

Hier wurde das Programmpaket SOM_PAK Version 3.1 von Teuvon Kohonen verwendet (ftp://cochlea.hut.fi/pub/som_pak/ [56- 60]).

3.h Peptid Docking

Für das *de novo* Design von Inhibitorpeptiden gegen das Schnupfenvirus HRV 14 wurden zwei verschiedene Verfahren angewendet.

Zum einen wurde die von Schneider und Wrede entwickelte Methode der Simulierten Molekularen Evolution (SME) angewendet [104, 109, 110]. Die SME verwendet ebenso wie die Evolutionsstrategie einen evolutionären Algorithmus zur Verbesserung von Peptiden. Im Gegensatz zum vorher beschriebenen Evolutionsalgorithmus werden bei der SME nicht die Gewichte optimiert, sondern die Eingabesequenzen eines bereits trainierten KNN. Damit stellt also das KNN die Fitnessfunktion dar.

Im Laufe des Trainings wird ebenfalls auf einen möglichst kleinen Ausgabefehler hin optimiert. Die zu optimierenden Parameter sind im Falle der SME keine Zahlen, sondern Aminosäuren. Um große und kleine Mutationsschritte zu erlauben, muß ein Abstand zwischen zwei Sequenzen definiert werden. Anderenfalls ist es nicht möglich eine systematische Suche durchzuführen. Es muß möglich sein, sowohl eng verwandte als auch weit entfernte Nachkommen zu erzeugen. Deshalb werden spezielle Ähnlichkeitstabellen verwendet, die das Erzeugen von Mutanten mit ähnlichen Eigenschaften ermöglichen. Hier wurde eine Ähnlichkeitstabelle basierend auf physikochemischen Eigenschaften verwendet. Die Lernrate wird bei jeder Generation wie in 3.g.iv beschrieben angepaßt. Es werden so viele Generationen berechnet bis die Lernrate so klein ist, daß keine Mutationen mehr auftreten.

Des weiteren wurde eine Methode entwickelt, um Peptide mit neuartigen Sequenzen in eine vorgegebene drei-dimensionale Aminosäureumgebung einzupassen, so daß eine starke Bindung zu erwarten ist. Die Bindungsstärken der so erstellten Peptide wurden in der Gruppe von Prof. Dr. Zeichhardt (Institut für Infektionsmedizin, Freie Universität Berlin) experimentell bestimmt.

Diese Methode ist ein wissensbasierter Ansatz, der die relativen Häufigkeiten von Kontakten in bekannten Enzym-Inhibitor Komplexen benutzt, um potentielle Dockingpartner vorzuschlagen. Für jede Aminosäure eines mit einem Rezeptor bindenden Peptides werden die Aminosäuren des Bindungspartners in einem vorgegebenen Radius gezählt. Die so bestimmten relativen Häufigkeiten werden dazu benutzt, um in einer vorgegebenen drei-dimensionalen Umgebung Aminosäuren als Bindungspartner vorzuschlagen. Dabei wurde die Canyonregion des Rhinovirus als Beispielumgebung verwendet. Mit Hilfe des BIOSYM Programms INSIGHT II (MSI) werden an einer beliebigen, aussichtsreichen Position die Aminosäuren an der Canyonoberfläche gezählt. Aufgrund dieser Daten ergibt sich aus der Tabelle der relativen Häufigkeiten eine Reihenfolge von möglichen Aminosäuren, die an die betreffende Stelle binden können. Es wird versucht, die Aminosäuren in der Nähe der Oberfläche zu plazieren, so daß keine Überschneidungen mit Canyon-Aminosäuren auftreten und der Kontakt zwischen Aminosäure und Oberfläche jedoch möglichst groß ist. Die nächste Aminosäure wird unter den gleichen Vorraussetzungen an die erste gebunden. Auf diese Weise können längerkettige Peptide entwickelt werden.