Protein Secondary Structure Prediction Using a Vector-Valued Classifier

Inaugural-Dissertation
to obtain the academic degree
Doctor rerum naturalium (Dr. rer. nat.)
submitted to the Department of Biology, Chemistry and Pharmacy
of Freie Universität Berlin

by

Matti Gerrit Korff

from Bad Oeynhausen, Germany

November 2015

Diese Arbeit wurde in dem Zeitraum von 1. Mai 2011 bis 3. November 2015 unter der Leitung von Prof. Dr. Ernst Walter Knapp am Institut für Chemie und Biochemie der Freien Universität Berlin im Fachbereich Biologie, Chemie und Pharmazie durchgeführt.

1. Gutachter: Prof. Dr. Ernst-Walter Knapp
2. Gutachter: Prof. Dr. Hermann-Georg Holzhütter

Disputation am 19.01.2016

Statutory Declaration

I hereby testify that this thesis is the result of my own work and research, except for any explicitly referenced material, whose source is listed in the bibliography. This work contains material that is the copyright property of others, which must not be reproduced without the permission of the copyright own.

Acknowledgements

Firstly, I like to thank Professor Ernst-Walter Knapp for the opportunity to work on such an interesting subject and for his supervision and valuable discussion. For the great atmosphere and help in dire times, I thank the entire research group.

For the invaluable and continuous support, I have to thank my family and my parents-in-law. For the proofreading of this work, I also like to thank my brother-in-law, Jim. Finally, the greatest thanks belongs to my wife Olga, because without her continuous backing this work would have not been possible.

Table of contents

Notation

Throughout this thesis a variety of mathematical symbols are used, which are listed here. Some of them are also introduced within the text.

$\mathbb{R}$	A set of real numbers
$\mathbb{Z}$	A set of integer numbers
$\in \mathbb{R}^n$	Real valued vector of dimension n
$\in \mathbb{R}^{n \times m}$	Real valued matrix of dimension $n \times m$
λ	A lower case letter is a scalar value
$\boldsymbol{v}$	A bold lower case letter is a vector
$\boldsymbol{M}$ or $\boldsymbol{m}$	A bold upper case or bold underlined lower case letter signifies a matrix
$< x >$	Pointy brackets mark the mean of a set of scalars or vectors
$\bar{x}$	The overline signifies the normalization of x to values between 0 and 1
$\boldsymbol{x}^T$ or $\boldsymbol{X}^T$	The superscript 'T' denotes the transpose of a vector or matrix
$\boldsymbol{1}_n$	A vector of ones with dimension n
$\boldsymbol{I}$	The unity matrix
N	Sample size
C	Number of classes

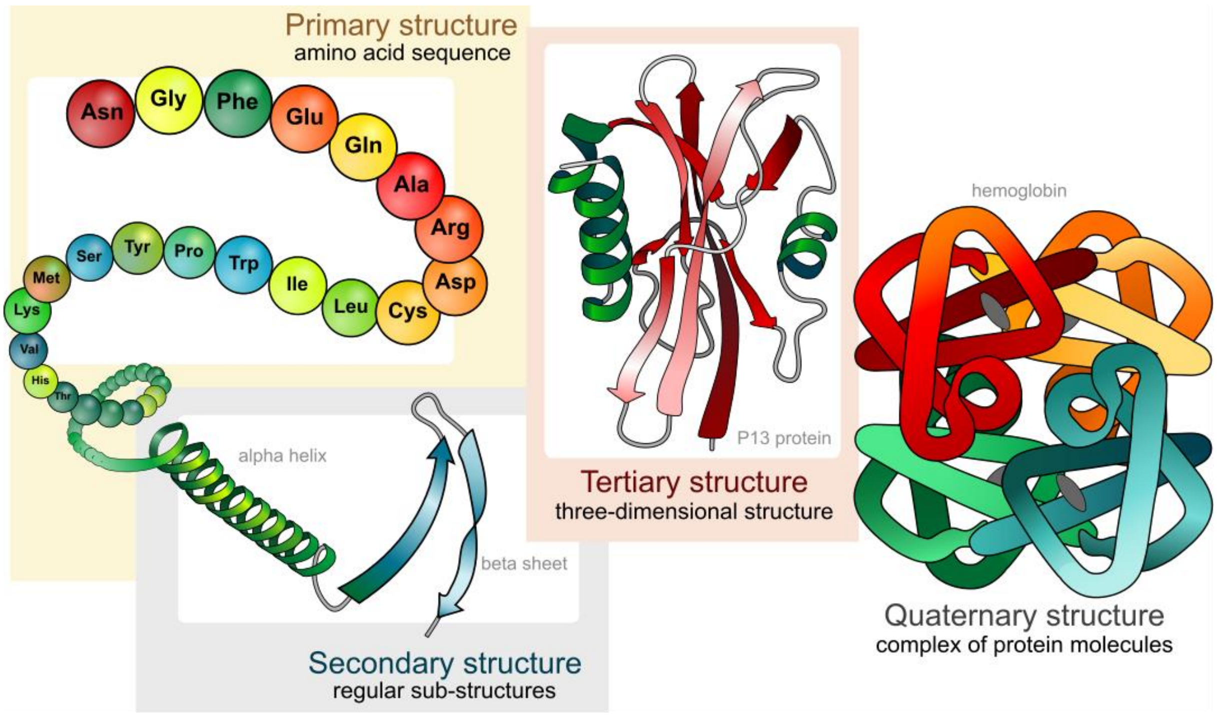

Figure 1-1: The four levels of protein structure: Primary, secondary, tertiary and quaternary. Original figure by Mariana Ruiz Villarreal (Villarreal, 2008).

1 Introduction

Proteins play an essential role in all living organisms. They participate in almost all cellular activities and perform a great variety of functions. These include immune responses, enzymatic reaction catalysis, cell signalling, cellular transport and cell reproduction, as well as structural functions. The critical role of proteins is reinforced by their abundance in eukaryotic cells, which consist of 70 % water and 15 % proteins (Nelson et al., 2008).

The structure of a protein defines its function. Therefore, knowledge of the protein structure is the fundamental basis for an in-depth understanding of its function. Proteins are linear chains of covalently bound amino acids, each of which has different chemical features. The length and composition of these chains is highly variable.

Four different levels of protein structure are defined (Lodish et al., 2000; Nelson et al., 2008). The first level, the primary structure, is the sequence of amino acids in the polypeptide chain. The length of a protein's polypeptide chain is highly variable, ranging from the 20-amino-acid long *cullin3* (de Paola et al., 2015) to the giant protein *titin*, responsible for the passive muscle elasticity (Labeit and Kolmerer, 1995) with a length of up to ~33000 residues. The second level is the secondary structure that describes the local order of a protein that is the local spatial

arrangement. The two most common classes of protein secondary structure are the α-helix and the β-strand. Neighbouring β-strands form β-sheets. Both α-helix and β-strand are stabilized through specific repeating hydrogen-bond patterns between different residues.
The three-dimensional folding of the protein chain is described by the third level, namely the tertiary structure. The protein tertiary structure is stabilized on one hand by covalent disulphide bonds between cysteine amino acids and on the other by a combination of different non-covalent interactions, which include hydrogen bonds, salt bridges and the hydrophobic effect of protein water interactions.
The fourth level, the quaternary structure, describes the arrangement of multiple protein chains and cofactors in space. Cofactors are non-peptidic chemical compounds that play a crucial role in protein function. Cofactors range from inorganic ions to complex organic or metallo-organic compounds. A prominent example is heme, a cofactor consisting of a porphyrin ring with a Fe^{2+} ion located in the centre. It is also a component of hemoglobin, the red pigment in blood. Haemoglobin is a protein complex of two α- and two β-chain haemoglobin proteins, each of which binds a heme cofactor.
The fold of a protein is defined as the spatial arrangement of the major secondary structure elements consisting of α-helices and β-strands (Lodish et al., 2000). Knowledge about the fold is key to understanding a protein's function. Furthermore, the fold between functionally similar proteins is more conserved than the amino acid sequence (Lodish et al., 2000). The fold is also used by the ***s**tructural **c**lassification **o**f **p**roteins* (SCOP) (Fox et al., 2014) database to assign a protein to a structural family.
The experimental determination of a protein's structure is usually done by either X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy. Of the two methods, X-ray crystallography is more widely employed. To determine a protein structure by X-ray crystallography, the protein sample has to be crystallized, which is an art in itself. NMR has the advantage that the protein can also be analysed in solution, a more natural state for most proteins. However, NMR is limited towards smaller sized proteins. Both methods require an immense investment of labour and equipment (Güntert, 2004).
These methods stand in stark contrast to new automated genome-sequencing methods, which have greatly increased the number of proteins with known sequences of amino acids. Meanwhile, the number of known protein structures increases at a much slower pace. The result is a wide gap between the number of known sequences and known structures of

proteins. The current release (April 2015) of the protein sequence database *UniProt* (Consortium, 2014; Gasteiger et al., 2003) contains 548,208 entries, whereas the **p**rotein **s**tructure **d**atabase (PDB) (Berman et al., 2003) contains only 108,263 entries. The discrepancy in size is even larger, if one considers the high degree of redundancy in the PDB (Berman et al., 2000).

The structural characterization of proteins is one of the major challenges of molecular biology. Experimental methods are constrained by technical and resource limitations. Therefore, accurate methods of predicting a protein's structure from its amino acid sequence are a modelling goal. An important step towards predicting the spatial atomic arrangement of a protein from its sequence is an accurate prediction of the protein's secondary structure. Knowledge of the secondary structure would significantly reduce the number of degrees of freedom in tertiary structure prediction (Lesk et al., 2001). A variety of methods have been developed to predict a protein's secondary structure from its sequence of amino acids, reaching accuracies of greater than 80 %.

In this work a revised version of the secondary structure prediction program *SPARROW (Rasinski, 2011) was developed, called *SPARROW$^+$. *SPARROW$^+$ uses a unique vector-valued scoring function to solve the multiclass problem of secondary structure prediction. The aim of the present work is to further improve the prediction quality and to introduce a new confidence measure to judge the reliability of prediction.

1.1 Protein structure

Amino acids are the basic building blocks of proteins and in this context are usually referred to as amino acid residues or just residues. All amino acids share the same basic constituents, a primary amine and a carboxyl group. The difference is in the variable residual group. Proline is the only exception, because of its cyclic structure, it has a secondary amine group instead of a primary amine. There are 23 proteinogenic amino acids with different chemical structures and features (Lodish et al., 2000; Nelson et al., 2008). Only 21 of these are encoded in the nuclear genes. Selenocysteine and pyrrolysine are generated post-translational. N-formylmethionine is commonly the initial amino acid in bacterial proteins, but is often removed post-translational. This leaves 20 genetically encoded amino acids also referred to as the standard amino acids that constitute natural proteins.

Amino acid	one letter code	Amino acid	one letter code
Alanine	A	Leucine	L
Arginine	R	Lysine	K
Asparagine	N	Methionine	M
Aspartate	D	Phenylalanine	F
Cysteine	C	Proline	P
Glutamine	Q	Serine	S
Glutamate	E	Threonine	T
Glycine	G	Tryptophan	W
Histidine	H	Tyrosine	Y
Isoleucine	I	Valine	V

Table 1-1: The 20 standard amino acids with the associated one letter code.

Amino acids are connected to each other through peptide bonds. The amine group of one amino acid covalently binds to the carboxyl group of another through a dehydration reaction. Hence, a linear chain of amino acids is assembled. The peptide bond has a double bond character, therefore rotation around this bond is not possible at room temperature. The only possible backbone rotations are with respect to the φ ψ dihedral angles (Lodish et al., 2000). The end of the protein chain with a free carboxyl group is referred to as the C-terminus, while the end with the primary amine is called the N-terminus. Residues are indexed from the C- to the N-terminus, since this is the same sequence in which they are synthesized in the ribosomes.

1.1.1 Amino acid propensities

On account of their diverse physicochemical behaviour most amino acids have a propensity for a particular secondary structure class. This propensity, although it is the basis of secondary structure prediction, is insufficient for an accurate classification (Chou and Fasman, 1974a). The two amino acids glycine and proline have a distinct influence on a protein's secondary structure. Both are predominantly found at the beginning or end of α-helix and β-strand, but rarely within them. Glycine's lack of a sidechain allows it a great conformational freedom. Therefore, it is often found in regions classified as *turn*. Proline on the other hand, has reduced conformational freedom because of its cyclic structure. Furthermore, due to of its secondary

Helix class	Hydrogen bond R_i-R_{i+n}	Translation per turn	Residues per turn
α-helix	n = 4	1.5 Å	3.6
3_{10}-helix	n = 3	2.0 Å	3.0
π-helix	n = 5	1.15 Å	4.1

Table 1-2: Properties of the three helix subclasses, α-, 3_{10}- and π-helix. Shown are the hydrogen bonding pattern, translation and residues per full turn.

amine functional group it cannot be a hydrogen-bond donor. Hence, proline is intolerant to both α-helix and β-strand secondary structures and is predominantly found in turn or coil regions (Chou and Fasman, 1978; Smith et al., 1980), where it often forms the transition between coil and other secondary structure classes (Chou and Fasman, 1974a; Li et al., 1996). The β-branched (isoleucine, threonine and valine) and aromatic (phenylalanine, tryptophan and tyrosine) amino acids are bulky and thus have a preference for β-strands for steric reasons. In a helical conformation, possible sidechain conformers could cause steric clashes with the protein backbone (Baldwin and Rose, 1999; Lezon et al., 2006; Srinivasan and Rose, 1999). In contrast, amino acids with unbranched sidechains (alanine, glutamate, leucine, lysine and methionine) have a propensity for helices.

1.1.2 Protein secondary structure

α-helix and β-strand are the most prominent but not the only secondary structure classes. Besides the α-helix, two additional classes of helices, 3_{10}- and π-helix, can be distinguished. There are also the turn, isolated β-bridge, bend and coil classes. These are the eight classes discriminated by the program DSSP (Kabsch and Sander, 1983), the common program for characterizing protein secondary structures. The properties of the different classes will be outlined in the following.

1.1.2.1 Helices

The α-helix is by far the most abundant of all helix classes. Characteristic of the helix class is a repeating hydrogen bonding pattern. Hydrogen bonds exist between the accepting carbonyl group of residue R_i and the donating amine group of residue R_{i+n}. The value of n depends on the helix class: 3 for 3_{10}-helix, 4 for α-helix and 5 for π-helix. To allow the repetition of such a hydrogen bonding pattern, the backbone must follow a tightly packed helical path, whereby the amino acid side chains point outwards. Characteristics of the helix classes are listed in Table 1-2**Fehler! Verweisquelle konnte nicht gefunden werden.**.

In contrast to the abundance of the α-helix, the π-helix class is the rarest and is usually not found alone. In most cases it appears together with an α-helix, forming a bulge in the helix structure. It is assumed that π-helices form through an insertion of an additional residue into α-helices (Cooley et al., 2010; Hollingsworth et al., 2009; Riek and Graham, 2011). The 3_{10}-helix can occur in isolation, but is often found at the beginning or end of an α-helix. In stand alone form, 3_{10}-helices are shorter than α-helices of average length.

1.1.2.2 β-sheet and β-strand

The β-sheet is the second type of regular secondary structure class. A β-sheet consists of laterally connected β-strands, which are distinguished by a specific hydrogen bonding pattern and ϕ, ψ dihedral angles (Astbury and Woods, 1934; Pauling and Corey, 1951). In difference to helices, the hydrogen bonds in β-strands are not between sequentially close residues. Depending on the directions of the interacting chains, a β-strand is called parallel or antiparallel.

1.1.2.3 Turns

The turn class describes directional changes in the protein backbone. The connection of two antiparallel β-strands by two or three residues is an example of a turn. To be classified as a turn, a backbone hydrogen bond between the carbonyl group of R_i and the amine group of R_{i+n} must exist. The definition of the turn class is therefore similar to that of the helix class, such that a helix could be considered as multiple conjoined turns. Based on the backbone dihedrals, turns can be separated into different types.

1.1.2.4 Isolated β-bridges

β-bridges can be described as a β-strand of one residue length. In a way similar to turns, multiple conjoined β-bridges form a β-strand. A β-bridge in isolation is assigned to the isolated β-bridge class.

1.1.2.5 Coil

The coil class cannot be considered as a true class, since it describes structureless protein regions. Hence, residue assigned as coil cannot be assigned to any other secondary structure class. Therefore, the coil class varies depending on the selection of secondary structure classes that are used for assigning protein secondary structures. If classes such as turn or bend are not considered separately, the corresponding residues usually belong to the coil class. Alternatively, the coil class can also be defined as ϕ, ψ dihedral angles different from those of strand or helix (Fitzkee et al., 2005).

1.1.3 Protein secondary structure assignment

A variety of different methods have been proposed to assign the secondary structure to a protein. These range from assignments based on hydrogen bonding patterns to geometrical considerations such as backbone dihedral angles, or combined methods. DSSP (Kabsch and Sander, 1983) and to a lesser extent STRIDE (Frishman and Argos, 1995) are the most used secondary structure assignment programs. Both employ the protein backbone hydrogen bonding patterns as the basis of their assignments. The difference is the method of estimating hydrogen bond energy. DSSP uses an electrostatic energy function, whereas STRIDE employs an empirical energy function. The positions of hydrogen atoms are usually not known from X-ray crystallography, because of the hydrogen atom's low electron density. Hence, both programs estimate the hydrogen-atom positions based on backbone atom geometry, although the method of estimation differs between the two programs. Depending on the repeating hydrogen bonding patterns found, residues are assigned as helical (α-, 3_{10}-or π-helix) or β-strand.

The secondary structures assigned by different methods can differ significantly, especially in the transition region from one secondary structure class to the other (Pylouster et al., 2010). Since a clear definition of secondary structure is still missing, it is difficult to compare the quality of different assignments (Tyagi et al., 2009).

1.2 Protein structure prediction

Knowing the structure of a protein is the basis to understand its molecular function. Many more protein sequences are known than structures, with the difference continuously increasing. Methods to predict a protein’s structure from its sequence are therefore in great demand.

The basis for protein structure prediction is the *sequence-protein-structure paradigm*, also referred to as the *lock-and-key hypothesis* (Anfinsen, 1973). According to which a protein achieves its biological function only by folding into a unique structure determined by its amino acid sequence. Although cases have been found where this hypothesis is invalid it mostly holds true (Dunker et al., 2008; Wright and Dyson, 1999).

Tertiary structure prediction methods are divided into four groups (Floudas et al., 2006). The first group consists of the first principle methods without database information (Osguthorpe, 2000). These are *ab initio* methods that aim at predicting a protein’s structure based on potential energy functions that describe the physics of a current conformational state and where only this potential function is used to predict the structure. They have the advantage that new structures can be found.

The next group encompasses the first principle methods using database information (Rohl et al., 2004; Srinivasan and Rose, 1995). These methods are different to the first group, because they build the starting point structure from structural fragments found in protein databases. Hence, short fragments of the protein sequence are compared to fragments of known protein structures (Floudas et al., 2006).

The third group of methods are the fold recognition and threading methods (Bowie et al., 1991; Jones et al., 1992). Here, the motivation is that structure is evolutionary more conserved than sequence. Thus, proteins with different sequences could have similar structures. Threading methods fit a protein sequence into a template structure.

The final group of methods are the comparative modelling and sequence alignment strategies (Martí-Renom et al., 2000; Sanchez and Šali, 1997). These methods align the whole protein sequence against proteins with known structure. The spatial model is derived from the structure of a homologous protein.

The first and second methods aim to predict the protein structure though simulation of the natural folding process. In contrast, the third and fourth methods predict a protein structure by comparing the sequence with available fold libraries and template structures. These methods also use protein secondary structure prediction to improve fitting and alignments (Ginalski et al., 2003; Kelley and Sternberg, 2009; Roy et al., 2010; Solis and Rackovsky, 2004).

1.2.1 Protein secondary structure prediction

In comparison to protein tertiary structure prediction, the problem of secondary structure prediction is simpler. In the former case, a procedure is needed that provides the spatial positions of all residues of a protein sequence. Thus, it needs to convert information relying on a one-dimensional quantity, i.e. the sequence, to a three-dimensional one. Whereas in the latter case, the one-dimensional sequence information needs to be converted only into another one-dimensional quantity.

The *Chou-Fasman method* was one of the first approaches to protein secondary structure prediction. (Chou and Fasman, 1974a; Chou and Fasman, 1974b; Chou and Fasman, 1978). The method is based on the analysis of relative amino acid frequencies in the different secondary structure classes. From these frequencies, a set of probability parameters are derived to describe the appearance of each amino acid in each secondary structure class. The method achieved an accuracy of 50-60 %.

The *GOR-method* (Garnier et al., 1978) uses Bayesian statistics to predict secondary structure based on a model of conditional probabilities. This method improved the accuracy to 65 %.

It has been proposed, that the theoretical limit of protein secondary structure prediction is at around 90 % (Dor and Zhou, 2007; Rost, 2001). The limiting factors are on one hand the variations in secondary structure assignments (Pylouster et al., 2010; Tyagi et al., 2009), and on the other hand, non-local interactions, between spatially close but sequentially distant residue pairs. Current state of the art methods can reach an accuracy above 80 %. The accuracy improvements can be attributed to the usage of sequence profiles, greater datasets and more sophisticated methods from machine learning, in particular artificial neural networks (Buchan et al., 2013; Drozdetskiy et al., 2015; Yaseen and Li, 2014). Among the most prominent methods are PSIPRED, C3-Scorpion and Jpred, which will be described in more detail in the following.

1.2.1.1 PSIPRED

Originally developed in 1999 (Jones, 1999) PSIPRED has been continuously updated to its most recent release of version 3.5. In its initial release, it was the first method to use sequence profiles generated by PSI-BLAST (Altschul et al., 1997) as input, which, at this time, made it superior to any other method. Over the years, PSIPRED still remains one of the programs with the highest prediction quality and is also used at an interim stage of 3D structure prediction methods.

For the prediction, PSIPRED uses two artificial neural networks. The first network generates a sequence-structure correlation, which the second network uses as an input to generate a structure-structure correlation. Both networks have a similar layout: Both are feedforward neural networks, with a single hidden layer and three output nodes. Yet, the absolute number of input and hidden nodes differs between the networks. The first network has 315 input nodes, which come from the size of sliding window (15) multiplied by the number of features (21). The sliding window gives the number of neighbouring residues, which are also taken into account for the prediction. The number of features is given by the number of PSI-BLAST profile values (20) and an additional feature for the cases is the sliding window overlaps with the N- or C-terminus of a sequence.

The input layer of the second network consists of 60 nodes, given again by multiplying the window size (15) with the number of features (4). Here, the input features are possible secondary structure classes predicted in the first network (3) and the terminal overlap feature. The hidden layers comprises of 75 nodes in the first and 60 nodes in the second network. Since version 3.3, PSIPRED uses three variants of the first network, each with different weight sets, and averages over the outputs.

1.2.1.2 C3-Scorpion

Released in 2014 (Yaseen and Li, 2014), the overall setup of C3-Scorpion has many similarities to PSIPRED. Like PSIPRED, C3-Scorpion employs a sequence of feedforward neural networks, but C3-Scorpion uses an additional third network compared to the two networks used by PSIPRED. The major difference to PSIPRED is the usage of so-called *context-based statistics* as additional input features apart from the PSI-BLAST profiles.

The *context-based statistics* are derived through the characterization of high-order inter-residue interactions between neighbouring residues to generate pseudo-potentials based on *Sippl's potentials of mean force* method (Sippl, 1990). These potentials are generated for singlets, doublets and triplets and are an estimate of residues adopting specific secondary structures depending on their neighbouring residues. According to the publication,

C3-Scorpion significantly surpasses the prediction quality of every other method, because of the context-based statistics.
As mentioned, three consecutive neural networks are used for prediction. All networks use a window size of 15 residues and an input feature to signify terminal overlap. The first network has 360 input nodes, given by the window size (15) and the input features (24). The features consist of the PSI-BLAST profile values (20), the terminal overlap (1) and the context-based scores (3). The second network uses only the output of the first as an input and has accordingly 60 input nodes. The second network has therefore a filtering function. Finally, the third network is akin to the first network. The difference is the usage of modified context-based scores, which are set to “absolute favourable” if the probability to adopt a certain secondary structure given by the second network is larger than 90 %. The publication does not mention the number of hidden layers and the respective number of hidden nodes in the networks.

1.2.1.3 Jpred

The original Jpred server for protein secondary structure prediction was released in 1998 (Cuff et al., 1998). This server used six different, at that time, state of the art prediction methods and presented the results of them all. In 2000 Jpred 1 (Cuff and Barton, 2000) was released, which replaced the six prediction methods with the Jnet prediction algorithm. The previously used methods had all been made obsolete with the release of PSIPRED (Jones, 1999). Since then, Jpred and Jnet have been regularly updated to their most recent release Jpred 4 in 2015 (Drozdetskiy et al., 2015). In the original Jpred 1, a combination of profiles generated with PSI-BLAST and amino acid frequency profiles were used as inputs. The frequency profiles were obtained through multiple sequence alignments. In the subsequent versions (Cole et al., 2008), the frequency profiles were replaced with *Hidden Markov Model* (HMM) profiles (Eddy, 1998). Jpred uses a sequence of two feedforward artificial neural networks for the prediction. Hence, the output of the first network is the input of the second. Like in PSIPRED and C3-Scorpion, the second network serves as a filter. Instead of combining different feature types into a single feature vector like in C3-Scorpion, the PSI-BLAST and HMM profiles are used separately. Each feature type is the input of an entirely different sequence of networks. The final prediction is the consensus of both sets of networks. The first network uses a window size of 17 residues, while the second network uses a larger window of 19 residues. In each network, the hidden layer comprises 100 nodes. The absolute number of input nodes is not explicitly mentioned.

1.3 Statistical classification

The aim of statistical classification is to identify to which class or sub population an object belongs to, in relation to a data set containing objects whose class assignment is known. Hence, each object possess certain properties or features which define its class affiliation.

Protein secondary structure prediction is a classification problem, where the residues in a protein sequence are classified into specific secondary structure classes. In the previous chapter 1.2.1, different programs were described for protein secondary structure prediction. Although the implementation differed considerably between the programs, at the basic level the setup was the same. All programs use the same approach for classification, namely **a**rtificial **n**eural **n**etworks (ANN). However, ANNs are only one possible approach of a wide range of classification methods. This chapter will give an overview about two classification approaches, linear regression and artificial neural networks.

Linear regression is a *linear classifier*, which classifies an object depending on the value of a linear combination of its features with associated weights. For a binary classification problem, the operation of a linear classifier would corresponds to splitting classes in the high-dimensional feature space with a hyperplane. Linear classifiers are broadly separated into *generative* and *discriminative* models. The former models the distribution of classes while the latter models the boundaries of them. *Linear regression* belongs to the *discriminative* models. Strictly speaking, the *perceptron*, a special type of an ANN is also a linear classifier, as is discussed in chapter 1.3.2.1.

Thera are multiple reasons, why specifically these two approaches are presented. In the previous chapter 1.2.1, concerning protein secondary structure prediction methods, it was shown that ANNs enjoy a large popularity and have been successfully applied to this classification problem. The derivation and properties of the vector-valued classifier are closely related to the linear regression, as outlined in chapter 2.4.3.

1.3.1 Linear regression

Regression estimates continuous response variables while classification methods use discrete response variables. Yet, it is possible to use a regression model for classification, by providing a indicator matrix that details to which class a training sample belongs. To optimize the objective function of a regression model, the *method of least-squares* (Gauss, 1809; Legendre,

1805) is often employed. Independent of the error distribution the least-squares method is best linear unbiased estimator. For linear regression models, it has the advantage, that a unique solution is obtained. Hence, iterative approaches, as outlined for the artificial neural network in chapter 1.3.2.3, are not needed.
Linear regression models are not limited to linear functions, only the parameter space has to be linear. This allows to utilize quadratic functions while still maintaining a unique solution of the objective function. Quadratic functions may allow to separate classes that are not linearly separable. Besides the linear regression models, also non linear regression models such as exponential and logistic regression models exist. In this chapter, only multivariate linear regression models are outlined. A simple linear regression model can be described by

$$\alpha + \beta_1 x_1 + \cdots + \beta_F x_F + \varepsilon = y, \tag{1-1}$$

where α and $\beta_1, \ldots, \beta_F$ are the unknown parameters. α is the intercept and α the regression coefficients. To describe the relation with the regressors x, it is assumed that deviations from the expected value can be attributed to a random error ε. Often the constant term α is included in the set of the coefficients β, then an artificial feature $x_0 = 1$ has to be introduced. For a set of N independent samples of $y_1, .., y_N$ with corresponding $x_{i1}, \ldots, x_{iF}$, the equations can be summarized by

$$\begin{pmatrix} 1 & x_{11} & \cdots & x_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{NF} \end{pmatrix} \begin{pmatrix} \alpha \\ \vdots \\ \beta_F \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \tag{1-2}$$

which corresponds to the matrix notation

$$\boldsymbol{X\beta} + \boldsymbol{\varepsilon} = \boldsymbol{y}, \tag{1-3}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times (F+1)}$ is the *design matrix*, $\boldsymbol{\beta} \in \mathbb{R}^{F+1}$ the parameter vector, $\boldsymbol{\varepsilon} \in \mathbb{R}^{N}$ the error term and $\boldsymbol{y} \in \mathbb{R}^{N}$.

1.3.1.1 Parameter estimation

The coefficients summarized in the vector $\boldsymbol{\beta}$ define the strength and direction of the respective features. For linear models, the standard approach is the *method of least squares*. According to the *Gauss-Markov theorem*, the ***b**est **l**inear **u**nbiased **e**stimator* (BLUE) for the coefficients of a linear regression model is the least squares estimator. In this case, best means giving the lowest variance of the estimate compared to other unbiased, linear estimators. The assumptions of the theorem are, that the errors are uncorrelated, have mean of zero and are homoscedastic with finite variance. However, it is not necessary for the errors to be normal distributed. *Biased* estimators exists, that can give a lower variance, like the *ridge regression* described in context of regularization in chapter 1.3.3.

The residual r_i of a sample i is defined as the difference between the observed y_i and estimated value $\hat{y}_i$. Hence it is given by

$$r_i = y_i - \hat{y}_i, = y_i - \boldsymbol{x}_i{}^T \boldsymbol{\beta}, \tag{1-4}$$

and measures the distance between a data point $(\boldsymbol{x}_i, y_i)$ and the hyperplane defined by the model $y = \boldsymbol{x}_i{}^T \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is the parameter vector. The residual assess the degree of fit between the actual data and the model. The *sum of squared residuals* or *error sum of squares* is a measure of the overall model fit

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{N} r_i{}^2 = \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i{}^T \boldsymbol{\beta})^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}). \tag{1-5}$$

$L(\boldsymbol{\beta})$ is termed the *objective function*, which is to be minimized. Expanding equation (1-5) yields

$$L(\boldsymbol{\beta}) = \boldsymbol{y}^T \boldsymbol{y} - 2\boldsymbol{\beta}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{\beta}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\beta}. \tag{1-6}$$

Taking the vector derivative of $L(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$, gives the *normal equations* of least-squares

$$\nabla \boldsymbol{\beta} \cdot L(\boldsymbol{\beta}) = 2\boldsymbol{X}^T \boldsymbol{y} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\beta} = 0, \tag{1-7}$$

respectively

$$\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T \boldsymbol{y}, \tag{1-8}$$

The expression $\boldsymbol{X}^T \boldsymbol{X}$ is referred to as the *Gramian matrix* of $\boldsymbol{X}$, which is proportional to the *covariance matrix*. The Gramian matrix is *positive semi definite* and symmetric. If the inverse of $\boldsymbol{X}^T \boldsymbol{X}$ exists, the solution of the linear equation systems is

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{1-9}$$

With the estimated values of $\boldsymbol{\beta}$, the predicted values $\widehat{\boldsymbol{y}}$ from the regression are given by

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{P}\boldsymbol{y}, \tag{1-10}$$

where $\boldsymbol{P} = (\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ is the *projection matrix* onto the space spanned by the columns of $\boldsymbol{X}$.

1.3.1.2 Alternative representation

Alternatively, instead of merging the intercept and the regression coefficients into a single vector, both can be treated separately. The reasoning for the separation is that the intercept is independent of the features while the regression coefficients are not. Thus, no artificial feature has to be introduced. The separation yields for equation (1-2)

$$\alpha \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} + \begin{pmatrix} X_{11} & \cdots & X_{1F} \\ \vdots & \ddots & \vdots \\ X_{N1} & \cdots & X_{NF} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_F \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}. \tag{1-11}$$

In matrix notation, this corresponds to

$$\alpha\mathbf{1}_N + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \boldsymbol{y}, \tag{1-12}$$

where $\boldsymbol{1}_N \in \{1\}^N$ is a vector of dimension N that only consists of ones. The least squares of the matrix notation with a separate intercept parameter is defined by

$$L(\alpha, \boldsymbol{\beta}) = (\mathbf{1}_N\alpha + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})^2. \tag{1-13}$$

Deriving the equation with respect to the intercept parameter α and setting it equal to zero, yields

$$\frac{\partial}{\partial b} L(\alpha, \boldsymbol{\beta}) = 2 \cdot \mathbf{1}_N(\mathbf{1}_N b + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = 0. \tag{1-14}$$

With the relation $\boldsymbol{1}_N{}^T \cdot \boldsymbol{1}_N = N$ the equation can be simplified to

$$Nb + \mathbf{1}_N\boldsymbol{X}\boldsymbol{\beta} - \mathbf{1}_N\boldsymbol{y} = 0 \tag{1-15}$$

Thus the intercept parameter is defined by

$$b = \frac{\mathbf{1}_N{}^T \boldsymbol{y}}{N} - \frac{\mathbf{1}_N{}^T \boldsymbol{X}}{N}\boldsymbol{\beta}. \tag{1-16}$$

Introducing the definition

$$\langle y \rangle = \frac{\mathbf{1}_N{}^T \boldsymbol{y}}{N} = \frac{1}{N}\sum_i^N \boldsymbol{y}_i \tag{1-17}$$

and in analogy

$$\langle \boldsymbol{X} \rangle = \frac{\mathbf{1}_N{}^T \boldsymbol{X}}{N} = \frac{1}{N}\sum_i^N \boldsymbol{X}_i, \tag{1-18}$$

yields for equation (1-16)

$$b = \langle y \rangle - \langle \boldsymbol{X} \rangle \boldsymbol{\beta}. \tag{1-19}$$

The vector derivate $\nabla \boldsymbol{\beta}$ of equation (1-13) is given by

$$\nabla(\boldsymbol{\beta}) L\left(\underline{\boldsymbol{\beta}}, \boldsymbol{b}\right) = \boldsymbol{X}(\mathbf{1}_N b + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = 0. \tag{1-20}$$

Inserting equation (1-19) into equation (1-20) gives

$$\boldsymbol{X}^T \mathbf{1}_N \langle y \rangle - \boldsymbol{X}^T \mathbf{1}_N \langle \boldsymbol{X} \rangle \boldsymbol{\beta} + \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\beta} - \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{y} = 0. \tag{1-21}$$

Using the relation

$$N \langle \boldsymbol{X} \rangle^T \langle \boldsymbol{X} \rangle = \boldsymbol{X}^T \mathbf{1}_N \langle \boldsymbol{X} \rangle = N \frac{\boldsymbol{X}^T \mathbf{1}_N}{N} \langle \boldsymbol{X} \rangle, \tag{1-22}$$

and by introducing similar to equation (1-17) and (1-18) the mean for the matrix

$$\langle \boldsymbol{X}^T \boldsymbol{X} \rangle = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{1-23}$$

and the vector

$$\langle \boldsymbol{X}^T \boldsymbol{y} \rangle = \frac{\boldsymbol{X}^T \boldsymbol{y}}{N} \tag{1-24}$$

allows to rearrange equation (1-21), yielding

$$\langle \boldsymbol{X}\rangle^T\langle y\rangle - \langle \boldsymbol{X}\rangle^T\langle \boldsymbol{X}\rangle\boldsymbol{\beta} + \langle \boldsymbol{X}^T\boldsymbol{X}\rangle\boldsymbol{\beta} - \langle \boldsymbol{X}^T\boldsymbol{y}\rangle = 0. \tag{1-25}$$

Hence, $\boldsymbol{\beta}$ is defined by

$$\boldsymbol{\beta} = \frac{\langle \boldsymbol{X}^T\boldsymbol{y}\rangle - \langle \boldsymbol{X}\rangle^T\langle y\rangle}{\langle \boldsymbol{X}\rangle^T\langle \boldsymbol{X}\rangle - \langle \boldsymbol{X}^T\boldsymbol{X}\rangle}, \tag{1-26}$$

and the normal equations (1-8) become

$$(\langle \boldsymbol{X}\rangle^T\langle \boldsymbol{X}\rangle - \langle \boldsymbol{X}^T\boldsymbol{X}\rangle)\boldsymbol{\beta} = \langle \boldsymbol{X}^T\boldsymbol{y}\rangle - \langle \boldsymbol{X}\rangle^T\langle y\rangle, \tag{1-27}$$

1.3.1.3 Solving the system of linear equations

$\boldsymbol{\beta}$ can be computed from the normal equations, (1-8) and (1-27), by solving the system of linear equations. Since the *Gramian matrix* $\boldsymbol{X}^T\boldsymbol{X}$, is *positive semi-definite* and *well-conditioned* the *Cholesky decomposition* (Schwarz and Köckler, 2011) $\boldsymbol{R}^T\boldsymbol{R}$, where $\boldsymbol{R}$ is an upper triangular matrix, can be used.

$$\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y} \tag{1-28}$$

The solution is obtained in two steps. The first step is a forward substitution to solve for the intermediate variable $\boldsymbol{z}$

$$\boldsymbol{R}^T z = \boldsymbol{X}^T\boldsymbol{y}. \tag{1-29}$$

The second step is a backward substitution to solve for $\boldsymbol{\beta}$

$$\boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{z}. \tag{1-30}$$

The triangular nature of $\boldsymbol{R}$ facilitates the substitutions

1.3.1.4 Classification

Linear regression functions can be used for classification by using an indicator matrix, whereby each class is given a specific scalar index. For a binary classification the indices are usually minus and plus one. This allows to assign target data depending on the sign of the linear regression function. However, for more than two classes the classification becomes difficult. One approach is to separate the multiclass problem into multiple binary problems, that can

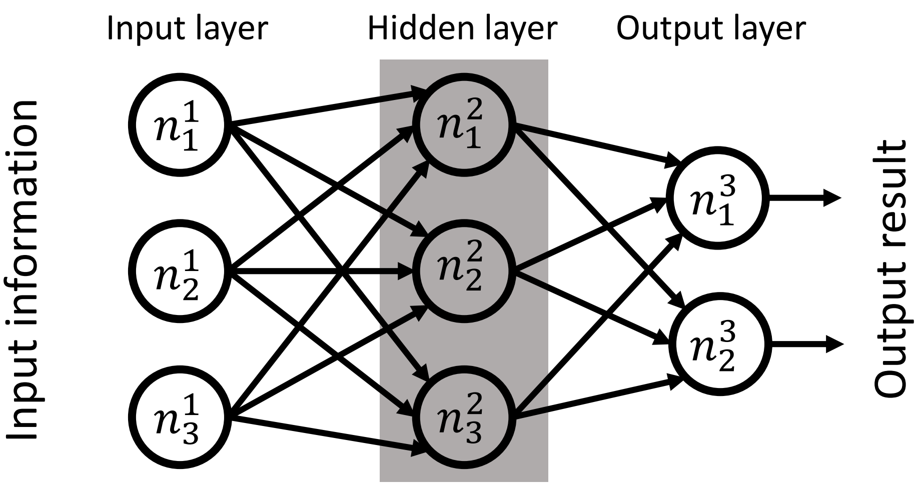

Figure 1-2: Artificial neural network with a single hidden layer. From three inputs, two outputs are generated. The annotation in each neuron shows the notation for numbering neurons and the associated parameters.

be solved independently with a specific linear regression function. Hence, either pairwise comparisons between two classes (one-vs.-one) are made or comparisons of each class against all others (one-vs.-all). A disadvantage of this approach is that class can be masked. Masking can occur if the decision boundaries of the binary classifications overlap. A decision boundary defines the value of the regression function at which target data is assigned to one or another class.

1.3.2 Artificial neural networks

Inspired by biological neural networks, like the brain, artificial neural networks (ANN) are a family of statistical learning methods that allow approximating unknown functions. These unknown functions are defined by a large number of inputs and the corresponding known results. An ANN is a system of interconnected nodes, also called neurons. The edges between the nodes have a numeric weight, which is adjusted through learning. ANNs have been used to solve a wide variety of problems, apart from protein secondary structure prediction, such as computer vision or speech recognition (Hinton et al., 2012; Krizhevsky et al., 2012).
The concept of ANNs stems from a computational model developed in 1943 (McCulloch and Pitts, 1943). Inspired by this work the first model of an ANN was created, called the *perceptron* (Rosenblatt, 1958; Rosenblatt, 1962). Although no clear definition of ANNs exist, a statistical model is called an ANN if it possess two characteristics. First, the model must contain sets of

adaptive weights, which are adjusted through learning and second the model must be able to approximate non-linear functions.

The nodes in an ANN receive inputs and generate outputs depending on these inputs. Each input has a specific weight, which defines its importance. The output of a node possess a non-vanishing value if the weighted sum of all inputs surpasses a threshold value. The output of a node is the input of the nodes in the next layer of neurons. Neurons can be separated into three types: The input, output and hidden neurons. Hidden neurons simply refer to neurons that are neither input nor output neurons. Input neurons, receive the input to the neural network. They can be understood as neurons, which only generate a specific output. Thus, output neurons serve as the result of the neural network, as such their outputs are not connected to other neurons.

The way neurons are connected, depends on the model of the ANN. In feedforward ANNs, the currently most common model, the networks can be separated into distinct layers. The output from the neurons in one layer is the input of the neurons in the following layer. As such, information is fed forwards. The neurons within a layer are not interconnected, thus have no loops. Of course, other models of ANNs exist, which contain feedback loops, these models are referred to as recurrent neural networks. Figure 1-2 shows an example of a simple feedforward network. The network consists of three layers, the input, hidden and output layer. Each layer contains the neurons of the respective type. A description of feedforward neural networks, that also concerns the mathematical description and the algorithm to adjust the weights, is given in the following chapters (Nielsen, 2015).

1.3.2.1 Perceptron

Perceptrons are the basic form of an artificial neural network (ANN) (Rosenblatt, 1958). They receive several binary inputs and produce a single binary output. Figure 1-3 shows a simple perceptron that receives the input values x whereby to each input belongs a weight w, the weights define the importance of the various input values. In order to activate a neuron the weighted sum of the input values has to surpass a threshold. The weighted sum can be expressed by $\sum_i w_i x_i$ or in vector form by the dot product $\boldsymbol{w} \cdot \boldsymbol{x}$. The bias b corresponds to the activations threshold and is defined by b = - *threshold*. Hence, the output y, also termed activation, can be described by a transfer function

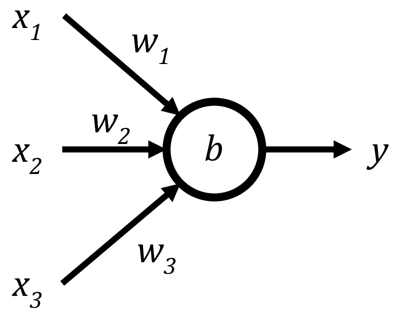

Figure 1-3: Layout of a single perceptron. This perceptron consists of input neurons x and a single output neuron (circle).To each input belongs a corresponding weight w . The output y generated by the output neuron depends on whether the weighted sum of the inputs minus the bias b , of the neuron, is greater or smaller than zero.

$$y = \begin{cases} 0 \text{ if } \boldsymbol{w} \cdot \boldsymbol{x} + \mathrm{b} \leq 0 \\ 1 \text{ if } \boldsymbol{w} \cdot \boldsymbol{x} + \mathrm{b} > 0 \end{cases}. \tag{1-31}$$

The limitation of single layer ANNs or perceptrons is their inability to separate non-linearly separable data. Hence, they are *linear classifiers*.

1.3.2.2 Sigmoid neuron

The disadvantage of perceptrons lies in their transfer function. Small changes in the weights of a perceptron can result in vastly different outputs. This is a problem for optimizing the weights in a learning procedure, where a small change of the weights should only lead to a small change in the overall result. Sigmoid neurons are similar to perceptrons, but the inputs are not either 0 or 1 but can be any real number in this range. Also the output is not 0 or 1 but given by the *sigmoid function*

$$\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + \mathrm{b}) = \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})}}. \tag{1-32}$$

Using a *step-function* instead of a *sigmoid function* would make the sigmoid neuron to a perceptron. The critical part of the sigmoid neuron is the smoothness of the σ function, which allows approximating changes in the output $\Delta\, a$, caused by changes in the weights w or bias b , through

$$\Delta y \approx \sum_i \frac{\delta y}{\delta w_i} \Delta w_i + \frac{\delta y}{\delta b} \Delta b \tag{1-33}$$

1.3.2.3 Gradient descent

Gradient descent is an optimization algorithm to find a local minimum of a function. Using an initial set of values for the variables of the function, gradient descent iteratively moves toward a set of variable values that minimize the function. The minimum is approached by iterative steps, which are proportional to the negative gradient at the current position.

To minimize a model, an objective function is used to ascertain the quality of the adjusted values. A simple yet effective objective function is the *quadratic objective function* or the *mean squared error*. For the network described in section 1.3.2.2 it would have the form

$$L(w,b) = \frac{1}{2N}\sum_{x} \|y(x) - y\|^2, \tag{1-34}$$

whereby, w denotes all weights, b is the bias, N the sample size, y the output and x the input. $y(x)$ denotes the desired output of the network using a specific input x. A more complex objective function is the *cross-entropy* function

$$L(w,b) = -\frac{1}{n}\sum_{x}\left[y \ln y + \left(1 - y(x)\right)\ln(1-y)\right], \tag{1-35}$$

which offers the advantage that it is not affect by learning slowdown like the *mean squared error* function (Nielsen, 2015). For a description of the gradient descent method, the simpler expression (1-34) is considered. Generalizing and thereby merging the variables w and b into a single vector v allows an easier description of the objective function. Thus, having a function with m variables $v_1, \ldots, v_m$ calculus gives the gradient difference through

$$\Delta \boldsymbol{L} \approx \frac{\delta \boldsymbol{L}}{\delta v_1}\Delta v_1 + \cdots + \frac{\delta \boldsymbol{L}}{\delta v_m}\Delta v_m \tag{1-36}$$

Then the change $\Delta \boldsymbol{L}$ in $\boldsymbol{L}$ produced by a small change $\Delta v \approx (\Delta v_1, \ldots, \Delta v_m)^T$ can be described by the gradient vector $\nabla \boldsymbol{L}$

$$\nabla \boldsymbol{L} = \left(\frac{\delta \boldsymbol{L}}{\delta v_1}, \ldots, \frac{\delta \boldsymbol{L}}{\delta v_m}\right)^T \tag{1-37}$$

Hence $\Delta \boldsymbol{L}$ can be written as

$$\Delta \boldsymbol{L} \approx \nabla \boldsymbol{L} \cdot \Delta \boldsymbol{v} \tag{1-38}$$

To achieve a negative $\Delta \boldsymbol{L}$

$$\Delta \boldsymbol{v} = -\eta \nabla \boldsymbol{L} \tag{1-39}$$

where η is a small, positive parameter called as the learning rate. The learning rate defines the speed of learning. So equation (1-38) becomes

$$\Delta L \approx -\eta \nabla L \cdot \nabla L = \eta \|\nabla L\|^2. \tag{1-40}$$

Because $\| \nabla L \|^2 \geq 0$, this guarantees that $\Delta L \leq 0$, thus L should only decrease and not increase. Thus, the gradient of v is

$$v \rightarrow v' = v - \eta \nabla L. \tag{1-41}$$

For a neural network equation (1-41) it can be written as

$$w_k \rightarrow w_k{}' = w_k - \eta \frac{\delta L}{\delta w_k} \tag{1-42}$$

$$b_l \rightarrow b_l{}' = b_l - \eta \frac{\delta L}{\delta b_l} \tag{1-43}$$

1.3.2.4 Backpropagation

While the previous chapter described how an artificial neural network (ANN) can adjust the weights and biases, it was not described, how the gradient is calculated. Backpropagation (Rumelhart et al., 1988) is an algorithm to compute the gradient for a variety of ANN types. The output of an ANN is the combination of the activations of all the neurons. The activation of the j^{th} neuron in the h^{th} layer is given by

$$y_j^h = \sigma\left(\sum_k w_{jk}^h y_k^{h-1} + b_j^h\right) \tag{1-44}$$

where y_k^{h-1} are the activations from the neurons in the previous $(h-1)^{th}$ layer. In vector form, this translates to

$$\boldsymbol{y}^h = \sigma(\boldsymbol{W}^h \boldsymbol{y}^{h-1} + \boldsymbol{b}^h) \tag{1-45}$$

To facilitate the computation of $\boldsymbol{y^h}$, an intermediate quantity, the *weighted input* $\boldsymbol{z^h}$, is introduced that is given by

$$\boldsymbol{z}^h = \boldsymbol{W}^h \boldsymbol{y}^{h-1} + \boldsymbol{b}^h. \tag{1-46}$$

Thus, equation (1-45) translates to $\boldsymbol{y}^h = \sigma(\boldsymbol{z}^h)$. The backpropagation algorithm requires two assumptions about the form of the objective function L in equation (1-34). Firstly, the objective function can be written as an average $L = \frac{1}{n}\sum_x L(x)$ over the values of the objective function $L(x)$ for individual training samples x. Secondly, it can be written as a function of the outputs of the network. The former is necessary to compute the partial derivatives $\partial L(x)/\partial w$ and $\partial L(x)/\partial b$ for a single training sample x. $\partial L/\partial w$ and $\partial L/\partial b$ are then recovered by averaging over all training samples.

By computing the partial derivatives $\delta L/w_{jk}^h$ and $\delta L/b_j^h$, backpropagation shows, how the weights and biases change the objective function. The computation of the partial derivatives requires another intermediate quantity δ_j^h, the error of neuron j in layer h, given by

$$\delta_j^h = \frac{\delta L}{\delta z_j^h}. \tag{1-47}$$

Backpropagation allows to calculate δ^h for every layer in the network and then to relate the error to the gradients of the weights and biases. It is based on four equations, which allow to calculate the error δ^h and the gradient of the objective function. The first equation concerns the error in the output layer H and is given by

$$\delta_j^H = \frac{\delta L}{\delta a_j^H}\sigma'(z_j^h), \tag{1-48}$$

which in matrix form translates to

$$\delta^H = \nabla_y L(x) \odot \sigma'(z^H). \tag{1-49}$$

The second equation relates to the error in the next layer δ^{h+1} and is defined by

$$\delta^h = ((\boldsymbol{w}^{h+1})^T \delta^{h+1}) \odot \sigma'(\boldsymbol{z}^h), \tag{1-50}$$

where superscript ‘T’ denotes the transpose of the matrix and $\odot$ denotes the *Hadamard* product. The *Hadamard* product is a special multiplication of two matrices with the same dimension, producing another matrix where each element ij is the product of elements ij of the original two matrices. Applying the transpose weight matrix gives a measure of the error

at the h^{th} layer. The *Hadamard* product then moves the error backwards through the activation function in layer h, giving the error in the weighted input of layer h.
The combination of equation (1-49) and (1-50) allows to calculate the error for any layer in the network. First (1-49) is used to obtain δ^{H} then (1-50) is iteratively applied to calculate the errors δ^{H-1}, δ^{H-2} …, δ^{h}. The third equation concerns the rate of change of the objective function related to any bias in the network. The error is equal to the change

$$\frac{\delta L}{\delta b_j^h} = \delta_j^h. \tag{1-51}$$

Finally, the fourth equation relates the rate of change of the objective function to any weights of the network

$$\frac{\delta L}{\delta w_{jk}^h} = y_k^{h-1}\delta_j^h. \tag{1-52}$$

Here k are the neurons in layer $h-1$ and j the neurons in layer h. This equation shows, that weight outputs from low-activation neurons ($y \approx 0$), learn slowly. The output layer neurons learn slowly in case of low ($y \approx 0$) or high activation ($y \approx 1$), they are then termed *saturated*. Any weights and biases to a saturated neuron learn slowly. In general, weights learn slowly if either the input is low-activation or the output neuron is saturated.
The backpropagation algorithm consists of five steps:

1. Set the activations y^1 for the input layer neurons.
2. Feedforward by computing $\boldsymbol{z}^h = \boldsymbol{w}^h\boldsymbol{y}^{h-1} + \boldsymbol{b}^h$ and $\boldsymbol{y}^h = \sigma(\boldsymbol{z}^h)$ for all $h = 2, 3, \dots, H$
3. Calculate the vector of the output error by $\boldsymbol{\delta}^H = \nabla_y L \odot \sigma'(\boldsymbol{z}^H)$.
4. Backpropagation of the error by computing $\boldsymbol{\delta}^h = ((\boldsymbol{w}^{h+1})^T\boldsymbol{\delta}^{h+1}) \odot \sigma'(\boldsymbol{z}^h)$ for all $h = H-1, H-2, \dots, 2$
5. Calculate the gradient of the objective function $\frac{\delta L}{\delta b_j^h} = \delta_j^h$ and $\frac{\delta L}{\delta w_{jk}^h} = y_k^{h-1}\delta_j^h$

The procedure of the steps 2-5 is repeated for different sample sets until the termination criteria is reached. This can be a maximum number of iterations or the change in the mean square error falls below a specific value.

1.3.2.5 Conclusion

Artificial neural networks are a powerful approach to multiclass classification problems. Their setup allows capturing high order correlations between inputs. Through the introduction of more hidden layers, this ability can be increased even further, also referred to as *deep learning*. Yet, the hidden layers make it difficult to interpret, what the network precisely learned, as such, it is somewhat a black box. Although more hidden layers increase the processing power and system flexibility, the system can become over specified and prone to overfitting. The number of parameters becomes too high.

1.3.3 Regularization

Regularization in statistics is a process of introducing additional information that allows to solve an ill-posed problem or to prevent overfitting of a model. Overfitting is the result if a complex model with a large number of parameters learns the noise of a dataset. The additional information is usually a form of penalty for complexity. Thus, regularization attempts to impose *Occam's razor* on a solution, thereby giving preference to simpler models. Regularization allows to tune the level of model complexity so the models are better at generalizing beyond the training data set. This leads to the two requirements for regularization.

First, two dataset are needed, a training dataset to adjust the model parameters and a validation dataset to assess the quality of the model. This can be realized through cross-validation. The second requirement is to use a regularization term with an associated tuning or regularization parameter λ . The regularization parameter is a hyperparameter that has to be adjusted empirically, which is often done by cross-validation.

1.3.3.1 Cross validation

In cross validation a dataset is partitioned into not overlapping subsets, the training and validation subsets. The parameters of a model are optimized with the training datasets, while the quality of the model is assed with the validation dataset. This procedure is repeated multiple times with different subsets. A models' quality can then be determined by the averaged errors for the validation datasets.

One approach to realize cross-validation, is the k-fold cross-validation. In this approach a dataset is partitioned into k subsets. One of these is used for validation, whereas the others are used for training. This is then repeated k-times for all subsets.

1.3.3.2 Regularization terms

For regularization, a norm $\lambda \,/\!/\boldsymbol{\beta}\,/\!/$ is added to a objective function $L\,(\boldsymbol{\beta})$, to minimize instead $L\,(\boldsymbol{\beta}) + \lambda \,/\!/\boldsymbol{\beta}\,/\!/$. Introducing a norm has the general effect that the values of small weights are emphasized. However, the way the weights shrink is different with each norm. Large weights are only allowed if they lead to considerable improvement of the objective function. It is a compromise of decreasing weights and minimizing the original objective function. To which of both aspects is given the preference, depends on the regularization parameter λ. A small λ minimizes the original objective function, while with a large λ the weights are made smaller.
If the weights are mostly small, changing a few inputs should not change the behaviour of the model much. Learning local noise is thereby suppressed. In contrast, a model with large weights may drastically change its behaviour if the input changes. Large weights allow the model to learn noise. Regularization constrains a model to lower complexity. Hence, these models are based on data patterns often occurring in the training data and are resistant to learn the noise of the training data.
There are two common regularization norms L_1 and L_2. The L_1 norm, also known as taxicab or Manhattan norm (Krause, 1973), is defined as the sum of absolute values of columns

$$\|\beta\|_1 = \sum_{i=1}^{n} |\beta_i|. \tag{1-53}$$

The name stems from the analogy to the distance a taxi has to drive in a rectangular street layout to get from the origin to its destination (Milner, 2007).
The L_2 norm is also referred to as the Euclidean length. As the name implies it gives the Euclidean distance in an n-dimensional Euclidean space

$$\|\beta\|_2 = \sqrt{\sum_{i=1}^{n} {\beta_i}^2} = \sqrt{\boldsymbol{\beta} \cdot \boldsymbol{\beta}}. \tag{1-54}$$

Although both norms penalize large weights, the way the weights shrink is different. The L_1 norm shrinks the weights with a constant value towards 0. In contrast, using the L_2 norm, the value by which the weights shrink is proportional to the weight itself. If a large weight has a major contribution to the original objective function, the L_1 norm shrinks the value less than the L_2 norm. For weights with small values, the opposite happens. Hence, the L_1 norm concentrate a model on a small number of import weights, while other weights have zero value. L_1 norms are therefore a method of *feature selection*. L_2 norms on the other hand lead to smaller weights in general. For L_1 norms it has to be considered that derivations of equation (1-53) with $\beta = 0$ are not defined. Therefore, iterative training procedures have to be used to determine the optimal parameters.

1.3.3.3 Tikhonov regularization

The Tikhonov regularization (Tikhonov and Arsenin, 1977) is most commonly used regularization method for ill-posed problems. Ill-posed refers to problems that do not have the properties of a well-posed problem such as that a solution exists, the solution is unique and the behaviour of the solution changes continuously with the initial conditions (Hadamard, 1902). Having system of linear equations, equation (1-3), describing an ill-posed problem

$$\boldsymbol{y} = \boldsymbol{X\beta}$$

the least squares approach, see equation (1-5)

$$\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2,$$

will lead to an over- or underdetermined system of equations. In particular using the inverse of $\boldsymbol{A}$ can amplify noise, rendering the influence of near singular values very large. Introducing a regularization term to equation (1-8) in form of a suitably chosen Tikhonov matrix $\boldsymbol{\Gamma}$ gives

$$\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \|\boldsymbol{\Gamma\beta}\|^2 = \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \lambda\|\boldsymbol{I\beta}\|^2, \tag{1-55}$$

where often a multiple of the identity matrix $\boldsymbol{I}$ is chosen for $\boldsymbol{\Gamma}$, whereby the regularization parameter λ is the multiplier. The identity matrix has a value of 1 in the diagonal and 0 elsewhere. The introduced matrix corresponds to an L_2 norm, that improves the conditioning of the problem allowing to determine a numerical solution

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{1-56}$$

2 Methods

The essential elements needed to construct a secondary structure prediction method are: Dataset, input data representation, classifier, scoring function and confidence measure. To train and validate a prediction method requires datasets. These contain protein sequences with known secondary structures. A requirement of the dataset is that it gives an exhaustive overview of the possible protein secondary structure space. Furthermore, the dataset needs to be non-redundant to prevent bias. The input data must also be appropriately represented. Not only the target residue, but also its neighbouring residues, are considered and converted into an amino-acid profile. Furthermore, in most cases, only three of the eight protein secondary structure classes are of interest. Therefore, a scheme is necessary to reduce the number of classes. At the core of the prediction method is the multi-class classifier, which assigns a secondary structure to a residue based on the input data. In the currently most popular protein secondary structure prediction methods, see chapter 1.2.1, artificial neural networks are used. In contrast, the method presented in this work, *SPARROW$^+$, uses a vector-valued classifier. To assess the reliability of the predictions made by the classifier, it is necessary to have a confidence measure. The specifications of each component are detailed in this chapter. The details of implementation are also outlined. Independent of the prediction method, quality measures are described to compare the performance of different methods.

2.1 Dataset

The dataset used for learning and cross validation is the ASTRAL40 (Berman et al., 2003; Brenner et al., 2000; Chandonia et al., 2002; Chandonia et al., 2004; Fox et al., 2014) dataset in version 2.03 of SCOP (**S**tructural **C**lassification **o**f **P**roteins) (Andreeva et al., 2014). The ASTRAL40 dataset is a compilation of all different structural folds found in the PDB (**P**rotein **D**ata **B**ank) (www.wwpdb.org) (Berman et al., 2003), whose sequence identity is lower than 40 %. SCOP primarily classifies proteins according to their α-helix and β-strand content. However, there are additional classes, which are independent of the secondary-structure content. Using the PDB directly would result in a skewed dataset, since some proteins are overrepresented in the PDB. For lysozyme and its mutants, for example, hundreds of structures are available, which have minimal structural differences.

The ASTRAL40 dataset was further processed. First, certain SCOP classes were removed from the dataset and special amino acids like selenocysteine and -methionine were exchanged for their standard amino acid counterpart. Special amino acids were removed, because they are ignored by BLAST and are treated as gaps. Hence, the profile generation could be improved if only standard amino acids are used.
The ASTRAL40 dataset was filtered to remove certain SCOP classes (Fox et al., 2014): *Membrane, cell surface proteins and peptides*, *coiled coil proteins*, *low-resolution proteins* and small proteins. The amino-acid composition at the surface of membrane proteins is very different from those of soluble proteins, which would compromise training of the classifier. Hence, it will be more useful to train a classifier for membrane and soluble proteins separately. *Coiled coil proteins* are rope like entangled protein chains. The interactions between the chains affect the structure of the single chains but cannot be considered in the structure prediction. For *low-resolution proteins* the secondary structure assignment is unreliable, whereas for small proteins with a chain of fewer than 50 amino acids it is not possible to generate meaningful sequence profiles.

2.2 Reduction schemes

The secondary structure assignment program DSSP (Kabsch and Sander, 1983) assigns residues into one of eight different secondary structure classes, which are described in detail in chapter 1.1.2. In practice, the secondary structure of a protein is rarely described with all eight classes but usually with only three classes. The three classes consist of helix, strand and coil class. The level of detail provided by the eight classes is in most cases not required or would not give further insights. To reduce the eight-class to a three-class description requires the merging of classes together into super-classes. A reduction scheme defines which classes are merged together.
Obviously, a variety of reduction schemes are possible, as illustrated in Table 2-1. This table lists reduction schemes tested during the development of *SPARROW$^+$ (*pure*, *mixed*, *all* and *mixed beta*) and those of competing secondary structure prediction programs (*psipred*, *scorpion* and *Jpred*). All reduction schemes agree to merge the coil, turn and bend classes into the coil super-class, since all of these three classes have no similarities with either the helix or strand super-classes. However, there are two major differences between the reduction

DSSP	reduction schemes						
classes	*pure*	*all*	*mixed*	*beta mix*	*psipred*	*scorpion*	*Jpred*
α-helix	helix	helix	helix	helix	helix	helix	helix
3_{10}-helix	coil	helix	coil	coil	helix	helix	coil
π-helix	coil	helix	helix	helix	coil	helix	coil
β-strand	strand	strand	strand	strand	strand	strand	strand
isolated β-bridge	coil	coil	coil	strand	strand	strand	strand
coil	coil	coil	coil	coil	coil	coil	coil
turn	coil	coil	coil	coil	coil	coil	coil
bend	coil	coil	coil	coil	coil	coil	coil

Table 2-1: Reduction schemes for the three super-classes (helix, strand and coil). The leftmost column lists the eight secondary structure classes as defined by DSSP and the top row lists the reduction schemes. The cells show, to which super-class a DSSP class belongs in a specific reduction scheme.

schemes. The first difference is, which helical classes make up the helix super-class. The second difference is in the composition of the strand super-class, whether the isolated β-bridge and β-strand class are merged or not.

The reasoning why certain classes should or should not be merged will be explained in the following, starting with the composition of the helix super-class. One extreme is, to merge all helix classes together, since they all have a very similar definition and structure. The opposite extreme is to put only the α-helix into the helix super-class, because the α-helix is by far the most frequent of all helix classes and all longer helices are exclusively α-helices. Between these extremes are the two cases, where the α-helix class is merged with either the 3_{10}- or the π-helix class. The difference between 3_{10}-helices and π-helices is, that the former can be found as an isolated helix, whereas the latter mostly occurs in the context of a α-helix. Interestingly, this difference can be used as an argument for both cases. In the first case, α- and 3_{10}-helices should be merged, because of their isolated existence. In the other case, α- and π-helices should be merged, because π-helices are only found in context of α-helices.

The other major difference is the question, whether to merge the strand class with the isolated β-bridge class. On the one hand, a β-strand is defined by DSSP as a sequence of β-bridges, therefore both classes should be merged. On the other hand, the DSSP definition for β-bridges is rather weak. Isolated β-bridges can be found at arbitrary positions in a protein

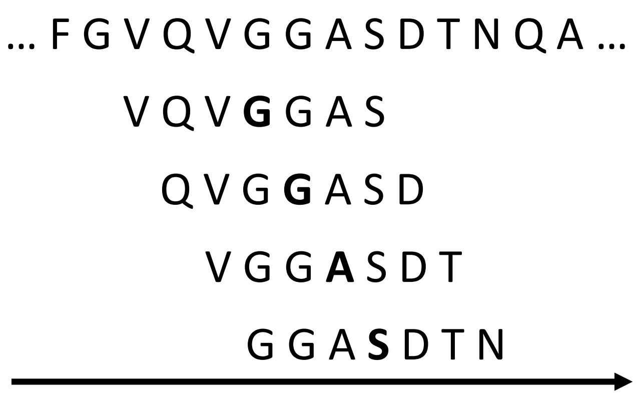

Figure 2-1: Sliding window for secondary structure prediction. To classify a residue of an amino acid sequence, all neighbouring residues covered by a sequence window are also included in the classification process. After classification of the highlighted residue (bold letter), the window moves one residue further.

with no structural similarities to a β-strand. Hence, isolated β-bridges should belong to the coil super-class.

2.3 Input data representation

Secondary structure classification of a specific residue is done by considering not only the residue of interest but also its sequentially neighbouring residues. Secondary structure classes result through the interactions of neighbouring residues. As such, a sequence window of a certain size is centred on the target residue to account for the influence of the neighbouring residues. The sequence window has to be limited and cannot encompass the entire protein, since this would result in a dramatic increase in the number of parameters necessary for the classifier. If the number of parameters is too large over-fitting may occur. Therefore, a sliding sequence window is commonly used in secondary structure prediction methods. The window is successively placed over every residue in a protein sequence meaning essentially the window slides over the sequence. The concept of the sliding window is illustrated in Figure 2-1. In general, symmetric windows are used. Thus, the number of neighbours is identical for both sides of the residue of interest. The window normally slides from the C- to the N-terminus of a protein sequence. For symmetrical windows, it would not make a difference if the window slid in the opposite direction. This would however be an issue if asymmetrical windows are used. Predictions with asymmetrical windows have shown no advantage over symmetrical ones (Bettella, 2009).

2.3.1 Representation of the sequence window

For classification, the input data has to be represented in an appropriate mathematical format. Different approaches are used to represent an amino-acid-sequence vector mathematically. One approach is to employ orthogonal binary vectors with 20 components per residue, thereby every residue position of the protein sequence is described by

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{20} \end{pmatrix}, a_t \in \{0,1\} \tag{2-1}$$

The components of this vector are zero except the one corresponding to the amino acid type at a specific position of the protein sequence, where the corresponding component adopts the value unity.

A sequence window covering $2n+1$ residues can be represented by a matrix $\boldsymbol{S} \in \{0,1\}^{(2n+1)\times 20}$, where n is the number of neighbouring residues in one direction from the residue of interest. For residue i the vector takes the form

$$\boldsymbol{S}_i = (\boldsymbol{a}_{i-n} \dots \boldsymbol{a}_i \dots \boldsymbol{a}_{i+n}). \tag{2-2}$$

Results of the earlier version of the original SPARROW showed that such a simple codification of the amino acid sequence is insufficient (Bettella, 2009). Instead, representing the input data for secondary structure prediction through *position specific sequence profiles* has proven to be a superior alternative (Rost and Sander, 1993a; Rost and Sander, 1993b). These profiles are used in variations by PSIPRED (Buchan et al., 2013), C3-Scorpion (Yaseen and Li, 2014) Jpred (Cuff and Barton, 2000) and *SPARROW$^+$. In all cases the profiles are generated through the program PSI-BLAST (**P**osition **S**pecific **I**terative-**BLAST**) (Altschul et al., 1997) that belongs to BLAST (**B**asic **L**ocal **A**lignment **S**earch **T**ool) methods (Altschul et al., 1990). BLAST encompasses a variety of methods to analyse protein- or DNA-sequences by comparing them with sequence databases. This allows for the identification of sequences that resemble the query above a certain significance threshold. To compare sequences, BLAST uses a series of local sequence alignments, meaning that only small parts of sequences are compared.

BLAST is a heuristic method, generating a lookup table for short subsequences, referred to as words in a query sequence. In the database search, BLAST seeks words that when aligned with the query and scored with a *substitution matrix* exceed a certain threshold. Words in the database that exceed the threshold are then extended in both directions to find the optimal local alignment.

Substitution matrices describe the probability that one amino acid in the sequence changes to a different amino acid. The probabilities for these transformations are expressed in *log-odds* scores. A score matrix S_c is defined as

$$S_{c,ij} = \log \frac{p_i \cdot M_{ij}}{p_i \cdot p_j} = \log \frac{M_{ij}}{p_j}, \tag{2-3}$$

where M_{ij} is the probability that amino acid i can be replaced by amino acid j. p_i and p_j are the frequencies at which the amino acids occur. The larger S_c, the more likely is the transformation. Popular substitution matrices are the PAM (**P**oint **A**ccepted **M**utation) (Eck and Dayhoff, 1966) and BLOSUM (**BLO**cks **Su**bstitution **M**atrix) (Henikoff and Henikoff, 1992) matrices. The PAM matrix is generated from a global sequence alignment, whereas the BLOSUM matrix is obtained through the alignment of sequence "blocks".

2.3.2 PSI-BLAST profiles

PSI-BLAST is an iterative method commonly used to identify distant relatives of a protein. In the first step, a list of similar protein sequences is generated through a BLAST search. From this list of sequences, a profile is generated that corresponds to a mean sequence. The obtained profile is then used for a second query, resulting in a larger list of protein sequences. This list is then used to generate another profile. The process can be repeated multiple times. Hence, PSI-BLAST provides an amino-acid profile for every residue of the protein sequence. It can be represented by a profile matrix with dimension $l \times 20$, where *l* is the length of the protein sequence and 20 corresponds to the number of standard amino acids.

Index	Residue	amino acid																			
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
10	L	1	-1	-1	-1	-2	3	-1	1	1	0	0	0	1	-1	-2	-1	-1	-2	0	0
11	Y	-4	-5	-6	-5	-1	-6	-6	-7	-4	2	-1	-5	-3	3	-4	-3	-4	7	7	1
12	Q	-3	-3	-3	-2	-6	6	0	-1	0	-3	-4	-2	-4	1	-2	-2	-3	1	5	-1
13	L	-4	-3	-4	-2	-2	-5	-5	0	-2	1	5	-5	-2	-2	-5	-2	0	0	5	-2
14	Q	-1	-1	0	-1	-5	5	0	3	-3	-4	-4	2	-2	-5	-4	0	1	-2	-3	-3
15	N	-1	-3	4	3	-5	-1	1	4	-1	-3	-3	-2	-4	-5	-2	0	-2	-5	-1	-4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 2-2: Excerpt of a PSI-BLAST profile. The two left most columns denote the consecutive residue index and the amino acid at this position of the protein sequence. The remaining 20 columns show the odds for every amino acid type to occur at this position. The top row lists the amino acid types to which the columns correspond.

The value of an element of the profile matrix is a score describing the odds that a certain amino acid type occurs at the specific sequence position. The scores s_i shown in Figure 2-2 are calculated independently for each column with

$$s_i = \frac{\ln\left(\frac{q_i}{p_i}\right)}{\lambda_u} \tag{2-4}$$

where q_i are the columns target frequencies for aligned pairs of amino acids. p_i are background frequencies and λ_u is the statistical parameter for ungapped local sequence alignments.

Positive scores stand for high probability of occurrence and negative numbers accordingly, for low probability. As depicted in Figure 2-2, the probabilities are position specific, an amino acid may have a high probability of occurrence in one position but not in the other. Because of the large number of sequence alignments, information about amino acid conservation can be obtained from the profiles. Thus, if an amino is preserved at a specific position throughout evolution the corresponding profile value is very high. The profiles also provide global information about the entire protein, since interacting amino acids are often exchanged in a process, called correlated mutations (Altschuh et al., 1987; Altschuh et al., 1988). Such correlated mutations indicate which amino acids, though sequentially distant, are structurally close. The residue profiles also provide information on the preference of a segment of the protein chain for certain types of amino acids, such as aromatic or polar amino acids.

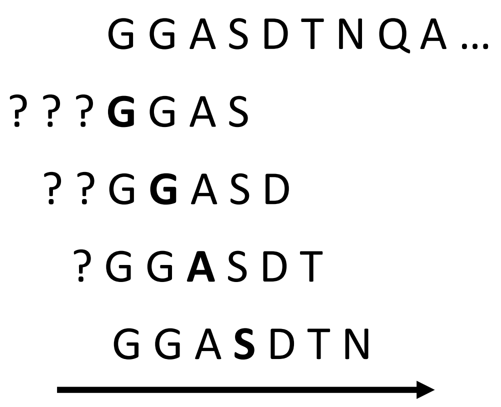

Figure 2-3: Incomplete window overlap at the terminal residues. A sliding window translates over an amino acid sequence, with the central residue highlighted in bold type. For terminal residues, the window does not completely overlap with the sequence. These non-overlapping window positions are shown with the character "?"

Therefore, the profiles contain a lot of information useful for secondary-structure prediction. Consequently, the vector-valued classifier of *SPARROW^{+} uses PSI-BLAST profiles as a representation of the input data. Thus, a sequence window is represented by a matrix $\boldsymbol{P} \in \mathbb{Z}^{20 \cdot (2n+1)}$ with 20 as the number of standard amino acids and *n* as the number of neighbouring residues in one direction of the target residue.

2.3.3 Terminal residues

The classifier uses a sliding window to make predictions for each residue. The sliding window allows for the consideration of not only the residue of interest but also its neighbouring residues. However, as soon as the window size is greater than one, a problem arises at the N- and C-terminal residues of a protein chain. At these residues, the window does not completely overlap with the protein sequence, illustrated in Figure 2-3. The problem is therefore, which values should be used for these undefined window positions.

In the classifier version presented here, the problem of incomplete overlap is solved by using default profile values of -1, the most abundant profile value. However, the larger the window size, the more frequently the window overlaps, increasing the influence of the default values on the prediction quality.

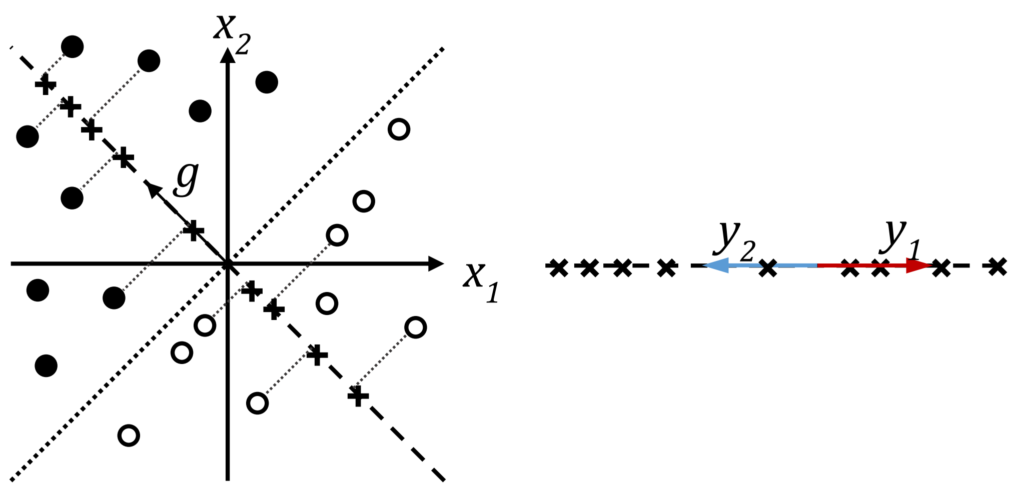

Figure 2-4: Fisher's linear discriminant. In a two-dimensional space, target data from two different classes is projected on a line, defined by the normal vector $\boldsymbol{g}$. Thus, two-dimensional data is projected into a one-dimensional space. In one-dimensional space the data can be classified by its relative orientation to the class vector y_1 and y_2.

2.4 Multi-class classifier

To solve the multi-class problem of secondary structure prediction, there are two approaches available. One is to separate the problem into different binary classification problems, which are then solved independently. The other is to solve the problem directly with a multi-class classifier, such as an artificial neural network (ANN) described in chapter 1.3.2. The first approach is already well developed with few opportunities for improvement, whereas besides ANNs, no multi-class classifier has gained widespread popularity. The vector-valued classifier presented here, is an updated version of the vector-valued classifier developed for the program *SPARROW (Rasinski, 2011).

2.4.1 Two-class classification

Normally, to solve a multi-class problem consisting of C classes, the comparison is done by C values that are connected via decision rules. Thus, the multi-class problem is split into C two-class problems. Two-class problems are a special case of multi-class problems, since only one value is sufficient for classification. It should therefore be possible to also solve a multi-class problem in a space of C-1 dimensions. *Fisher's linear discriminant* (Fisher, 1936) projects the multidimensional target data onto a line as illustrated in Figure 2-4. Hence, the classification is done in a one-dimensional space by the relative distance to the zero point. The position in the one-dimensional space can be described by a scalar s, which is obtained through the scalar product of

$$s = \boldsymbol{g} \cdot \boldsymbol{x}, \tag{2-5}$$

where $\boldsymbol{g}$ is the vector describing the direction of the line in a two-dimensional space and $\boldsymbol{x}$ the input data. Classification is then done by the sign of s. In a one-dimensional space, two unit vectors are defined, which are $\boldsymbol{y}_1 = 1$ and $\boldsymbol{y}_2 = -1$. Both vectors have the maximal distance possible to each other in one-dimensional space. Classifying by the sign of s corresponds to a classification depending upon which product is larger, $s \cdot y_1$ or $s \cdot y_2$.
Hence, a classification function $f(\boldsymbol{x})$ projects a point $\boldsymbol{x} \in \mathbb{R}^n$ from n-dimensional to one-dimensional space $s = f(\boldsymbol{x})$. Then, taking into account the products of s with the unit vectors, y_1 and y_2 the classification of the point $\boldsymbol{x}$ is defined by

$$\boldsymbol{x} \in \zeta_\alpha \Leftrightarrow \alpha = \max_j (s \cdot y_j), \tag{2-6}$$

where ζ_α denotes the class affiliation of $\boldsymbol{x}$. Since the unit vectors y_j define each class completely, they are subsequently referred to as *class vectors*.

2.4.2 Multi-class classification

A solution for C classes can be developed by generalizing the notation developed for the two-class classification. The data is represented by a set of vectors $x_i \in \mathbb{R}^n$ with $1 \leq i \leq N$, each of which are assigned to a class $Z = \{\zeta_j | 1 \leq j \leq C\}$. Analogous to section 2.4.1, this classification problem can be solved by a projection of $\boldsymbol{x}_i$ into a vector space of (C-1)-dimensions. To project the data, a vector-valued scoring function is used given by

$$\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^{C-1}. \tag{2-7}$$

The classification depends on which class vector projection has the greatest overlap. This can be described by the scalar product of the vector-valued scoring function with the class vectors. Hence, the classification rule is defined as

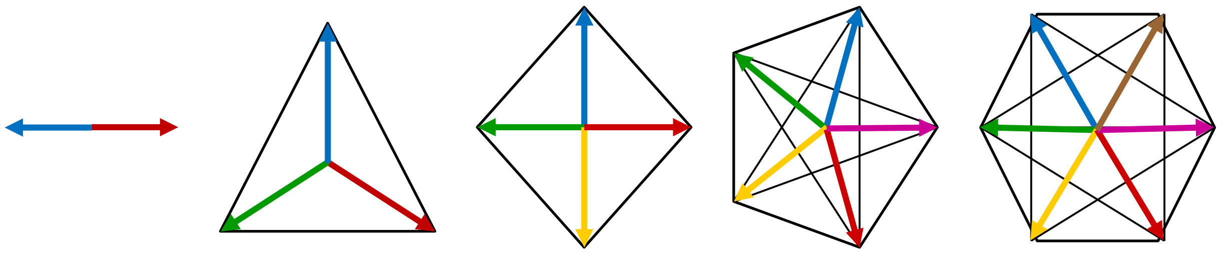

Figure 2-5: Two-dimensional projections of class vectors and the corresponding regular simplex for 2 to 6 classes. From left to right the simplexes are referred to as line segment, triangle, tetrahedron, pentachoron, hexatron.

$$\boldsymbol{x} \in \zeta_\alpha \Leftrightarrow \alpha = \max_j\left(\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y}_j\right). \tag{2-8}$$

To classify a feature vector, two things are necessary: The class vectors and the vector-valued function project the data into the (C-1)-dimensional space spanned by the class vectors.

2.4.2.1 Class vectors

Each class is defined by a single class vector. All class vectors $\boldsymbol{y}_k$ should have the following three properties: Firstly, each class must possess a class vector $\boldsymbol{y}_k \in \mathbb{R}^{C-1}$. Secondly, all class vectors are unit vectors, or of equal length for a simple comparison using a scalar product. Thirdly, class vectors should have the maximal distance to each other. To achieve the maximum distance between C class vectors in a (C-1)-dimensional space they must be the corner vectors of a regular simplex (Coxeter, 1973) or in other words they span a generalized tetrahedron. Figure 2-5 illustrates various regular simplexes.

2.4.2.2 Generalized tetrahedron

Starting from the standard Cartesian basis in an n-dimensional space $(n+1=C)$ $\{\boldsymbol{e}_j, j=1,2,\dots,n\}$ with $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \delta_{i,j}$. The centre of geometry of the generalized tetrahedron is in the origin of the coordinate system. Now the vectors $(\boldsymbol{y}_k, k=1,\dots,n+1)$ are constructed, which point from the origin of the coordinate system to the corners of the generalized tetrahedron. The vectors $\boldsymbol{y}_k$ are normalized to unity and obey the relation:

$$\boldsymbol{y}_i \cdot \boldsymbol{y}_j = -\frac{1}{n} \tag{2-9}$$

Starting by setting $y_1 = e_1$. Next,

$$\boldsymbol{y}_2 = -\frac{1}{n}\boldsymbol{y}_1 + \frac{\sqrt{n^2-1}}{n}\boldsymbol{e}_2 \tag{2-10}$$

is used. The third step yields

$$\boldsymbol{y}_3 = -\frac{1}{n-1}(\boldsymbol{y}_1 + \boldsymbol{y}_2) + a_3\boldsymbol{e}_3, \tag{2-11}$$

which is right, since $\boldsymbol{y}_3 \cdot \boldsymbol{y}_i = -\frac{1}{n-1}\left(1+\frac{1}{n}\right) = -\frac{1}{n}$ for $i = 1, 2$. Normalizing $\boldsymbol{y}_3$ yields:·

$$\boldsymbol{y}_3 = \frac{2}{(n-1)^2}\left(1-\frac{1}{n}\right) + {a_3}^2 = \frac{2}{n(n-1)} + {a_3}^2 = 1 \tag{2-12}$$

from which follows $a_3 = \sqrt{1-\frac{2}{n(n-1)}}$. Hence,

$$\boldsymbol{y}_3 = -\frac{1}{n-1}(\boldsymbol{y}_1 + \boldsymbol{y}_2) + \sqrt{1-\frac{2}{n(n-1)}}\,\boldsymbol{e}_3. \tag{2-13}$$

Thus in the last step it is

$$\boldsymbol{y}_{n+1} = -\sum_{j=1}^{n}\boldsymbol{y}_j \tag{2-14}$$

which is valid since $\boldsymbol{y}_{n+1} \cdot \boldsymbol{y}_i = -\left(1-\frac{n-1}{n}\right) = -\frac{1}{n}$ for $i = 1, 2, \dots, n$ and $\boldsymbol{y}_{n+1}$ is of course also normalized:

$$\boldsymbol{y}_{n+1} \cdot \boldsymbol{y}_{n+1} = n\boldsymbol{y}_1 \cdot \boldsymbol{y}_1 - n(n-1)\boldsymbol{y}_1 \cdot \boldsymbol{y}_2 = n - n(n-1)\frac{1}{n} = 1. \tag{2-15}$$

Now providing the general expression for $k+1,\ k \geq 2$:

$$\boldsymbol{y}_{k+1} = -\frac{1}{n-k+1}\sum_{j=1}^{k}\boldsymbol{y}_j + a_{k+1}\boldsymbol{e}_{k+1}, \tag{2-16}$$

which is valid, since $\boldsymbol{y}_{k+1} \cdot \boldsymbol{y}_i = -\frac{1}{n-k+1}\left(1+\frac{k-1}{n}\right) = -\frac{1}{n}$, for $i = 1,\ 2, \dots k$. Normalizing $\boldsymbol{y}_{k+1}$ yields

$$\boldsymbol{y}_{k+1} \cdot \boldsymbol{y}_{k+1} = \frac{1}{(n-k+1)^2}\left(k - \frac{k(k-1)}{n}\right) + {a_{k+1}}^2 = 1. \tag{2-17}$$

Hence ${a_{k+1}}^2 = 1 - \frac{k}{n(n-k+1)}$.

2.4.3 Vector-valued classification function

Chapter 2.4 gave the definition for a multi-class classification function $\boldsymbol{f(x)}$ by equation (2-7)

$$\boldsymbol{f}\colon \mathbb{R}^n \to \mathbb{R}^{C-1},$$

whereby $\mathbb{R}^n$ is the space of the input and $\mathbb{R}^{C-1}$ of the output data. Various ways are possible to define a vector valued scoring function. The simplest is to define $\boldsymbol{f(x)}$ as a vector of $C-1$ scalar scoring functions $f_k(\boldsymbol{x})$ with

$$f_k\colon \mathbb{R}^n \to \mathbb{R}. \tag{2-18}$$

These scoring functions are applied independently to the input data $\boldsymbol{x}$ and give the specific components of the projection of $\boldsymbol{x}$ into the (C-1)-dimensional space. Hence, $\boldsymbol{f(x)}$ is defined by

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_{C-1}(\boldsymbol{x}) \end{pmatrix}. \tag{2-19}$$

Accordingly, in the case of a $C=2$ the single scalar classifier $\boldsymbol{f(x)}=f_1(\boldsymbol{x})$ is obtained. The classifier is then a linear multivariate regression function described in chapter 1.3.1.

2.4.3.1 Definition of the scoring functions

The function type of every scoring function is arbitrary, as long as equation (2-18) is obeyed. Assuming only a nonlinear separability of the input data, a more complex projection is necessary than a function linear in parameter and feature space. A quadratic function is the simplest non-linear function and defined by:

$$f_k(\boldsymbol{x}) = f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) = \boldsymbol{x}^T \boldsymbol{B}_k \boldsymbol{x} + \boldsymbol{c}_k{}^T \boldsymbol{x} + a_k \tag{2-20}$$

where, the vector $\boldsymbol{\theta}_k$ contains all adjustable parameters $(\boldsymbol{B}_k, \boldsymbol{c}_k, a_k)$ of $f_k(\boldsymbol{x})$. The Matrix $\boldsymbol{B}_k \in \mathbb{R}^{n \times n}$ describes the influence the products $x_i x_j$ have on the value of kth-scoring function $f_k(\boldsymbol{x})$ of $\boldsymbol{f(x)}$. The vector $\boldsymbol{c}_k \in \mathbb{R}^n$, defines the direct influence x_i the input data vector $\boldsymbol{x}$ has on $f_k(\boldsymbol{x})$. Finally, the scalar $a_k \in \mathbb{R}$, the intercept, allows an of $\boldsymbol{x}$ independent adjustment of the value of $f_k(\boldsymbol{x})$. Therefore, equation (2-20) can be rewritten as

$$f_k(\boldsymbol{x}) = f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) = \sum_{i=1}^{n} \sum_{j=1}^{n} B_{kij} x_i x_j + \sum_{i=1}^{n} c_{ki} x_i + a_k. \tag{2-21}$$

In the case of protein secondary structure the quadratic function has an additional advantage. Through the multiplication of $x_i x_j$, different window positions are correlated with each other, which captures the interactions between neighbouring residues. In particular the helix secondary structure motifs, see chapter 1.1.2.1, are defined by interactions between sequentially close residues.

2.4.4 Optimization of the classifier

The parameters $\boldsymbol{\theta}_k$ of each scoring function $f_k(\boldsymbol{x}, \boldsymbol{\theta})$ can be combined to one global parameter matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{C-1})$, containing all adjustable parameters of the vector-valued scoring function $\boldsymbol{f}(\boldsymbol{x})$. Hence, equation (2-19) can be rewritten as

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\Theta}) = \begin{pmatrix} f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\ \vdots \\ f_{C-1}(\boldsymbol{x}, \boldsymbol{\theta}_{C-1}) \end{pmatrix}. \tag{2-22}$$

The parameters are determined such that the vector-valued scoring function $\boldsymbol{f}(\boldsymbol{x})$ projects an input vector $\boldsymbol{x} \in \mathbb{R}^n$ as close as possible to the corresponding class (j) vector $\mathbb{R}^{C-1}$

$$\boldsymbol{f}(\boldsymbol{x}_{j,n}, \boldsymbol{\Theta}) \approx \boldsymbol{y}_{j,n}, \tag{2-23}$$

According to the definition of the scoring function in equation (2-22), the parameters for each scoring function can be optimized separately. Thus

$$f_k(\boldsymbol{x}_{j,n}, \boldsymbol{\theta}_k) \approx \boldsymbol{y}_{k,j,n}, \tag{2-24}$$

Although equation (2-20) is a quadratic function in feature space, it is still a linear function in parameter space. As such, the best optimization approach for an unbiased estimator is the method of least squares. In chapter 1.3.1 the least squares method was used for a very similar problem, a linear regression function. Hence, equation (2-24) can be fulfilled by minimizing the objective function

$$L(\boldsymbol{\theta}_k) = \sum_{j=1}^{C} \sum_{n=1}^{N_j} \left(f_k(\boldsymbol{x}_{j,n}, \boldsymbol{\theta}_k) - \boldsymbol{y}_{k,j,n} \right)^2. \tag{2-25}$$

To determine the optimal parameters $\boldsymbol{\theta}_k$, the first derivative of the objective function with respect to the parameter vector must be set equal to zero.

2.4.4.1 Linearization of the scoring function

To simplify the calculation of the derivative, the quadratic scoring functions are linearized by combining the quadratic and linear terms into a single vector. The number of parameters n of a scoring function $f_k(\boldsymbol{x}, \boldsymbol{\theta}_k)$ is the sum of the parameters for the linear and quadratic features

$$n = n_l + n_q. \tag{2-26}$$

The number of the quadratic features is less than n_l^2, since the quadratic feature matrix defined by

$$\boldsymbol{X}_q = \boldsymbol{x} \cdot \boldsymbol{x}^T, \tag{2-27}$$

is symmetric. Hence, the number of quadratic features corresponds to the lower or upper triangular matrix of $\boldsymbol{X}_q$ and is given by

$$n_q = \frac{n_l \cdot (n_l + 1)}{2}. \tag{2-28}$$

For the total number of parameters n_{tot}, the intercept parameter also has to be taken into account, thus

$$n_{tot} = 1 + n_l + n_q. \tag{2-29}$$

Linearization of the triangular quadratic feature matrix $\boldsymbol{X}_q$ generates a feature vector, which is defined by

$$\boldsymbol{x}_q{}^T = \left(X_{q_{1,1}}, \dots, X_{q_{1,n_l}}, X_{q_{2,2}} \dots, X_{q_{n_l,n_l}}\right). \tag{2-30}$$

The quadratic and linear feature vectors are then concatenated into a single feature vector given by

$$\boldsymbol{x}_P{}^T = \left(\boldsymbol{x}^T \parallel \boldsymbol{x}_q{}^T\right), \tag{2-31}$$

that corresponds to

$$\boldsymbol{x}_P{}^T = \left(x_{l_1}, \dots, x_{l_{n_l}}, x_{q_1}, \dots, x_{q_{n_q}}\right). \tag{2-32}$$

In analogy, the same procedure holds true for the parameters of the scoring function. Since the feature matrix is symmetric, only the corresponding upper or lower triangular matrix has to be considered of the parameter matrix $\boldsymbol{B}$. The linear quadratic parameter vector $\boldsymbol{b}$ is generated by

$$\boldsymbol{b}^T = \left(\mathrm{B}_{1,1}, \dots, \mathrm{B}_{1,n_l}, \mathrm{B}_{2,2} \dots, \mathrm{B}_{n_l,n_l}\right). \tag{2-33}$$

Accordingly, the linear and quadratic parameters are all concatenated into a single parameter vector

$$\boldsymbol{\beta}^T = (\boldsymbol{c}^T \parallel \boldsymbol{b}^T), \tag{2-34}$$

which corresponds to

$$\boldsymbol{\beta}^T = \left(c_1, \dots, c_{n_l}, b_1, \dots, b_{n_q}\right). \tag{2-35}$$

Using the definitions of the weight and feature vectors, the scoring function can be rewritten as

$$f_k(\boldsymbol{x}, \boldsymbol{\theta}) = a_k + \boldsymbol{c}_k{}^T\boldsymbol{x} + \boldsymbol{x}^T\mathbf{B}_k\boldsymbol{x} = \boldsymbol{\beta}_k{}^T\boldsymbol{x}_P + \alpha_k. \tag{2-36}$$

Thus, the scoring function $f_k(\boldsymbol{x}, \boldsymbol{\theta})$ is a linear function in the parameter as well as the feature space from which the feature vectors $\boldsymbol{x}_P$ originate.

2.4.4.2 Derivation of the parameters of the scoring functions

A different parameter set is used for each scoring function. The parameter matrix $\underline{\boldsymbol{\beta}}$ comprises all parameter vectors, yielding

$$\underline{\boldsymbol{\beta}}^T = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \dots, \boldsymbol{\beta}_{Kc}). \tag{2-37}$$

Thus, the classification function is defined by

$$\boldsymbol{f}\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}, \boldsymbol{x}_{P_{j,n}}\right) = \underline{\boldsymbol{\beta}}\boldsymbol{x}_{P_{j,n}} + \boldsymbol{\alpha}, \tag{2-38}$$

where $\boldsymbol{\alpha}$ is the intercept parameter vector. The parameters are determined such that the scoring function is as close as possible to the corresponding class (j) vector

$$\boldsymbol{f}\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}, \boldsymbol{x}_{P\,j,n}\right) \approx \boldsymbol{y}_{j,n}. \tag{2-39}$$

In accordance with equation (2-25), equation (2-39) can be fulfilled by minimizing the loss function

$$L\left(\underline{\boldsymbol{\beta}}, \boldsymbol{a}\right) = \sum_{j=1}^{C} \sum_{n=1}^{N_j} \left(\boldsymbol{f}\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}, \boldsymbol{x}_{P\,j,n}\right) - \boldsymbol{y}_{j,n}\right)^2. \tag{2-40}$$

Hence, for scoring function *k* the parameters are $\boldsymbol{\beta}_k$ and α_k. The result of the scalar product in equation (2-38) is still a vector with K_c components as is f. It reads

$$\left(\underline{\boldsymbol{\beta}}, \boldsymbol{x}_{j,n}\right)^T = \left(\boldsymbol{\beta}_1 \boldsymbol{x}_{P\,j,n}, \boldsymbol{\beta}_2 \boldsymbol{x}_{P\,j,n}, \dots, \boldsymbol{\beta}_{Kc} \boldsymbol{x}_{P\,j,n}\right). \tag{2-41}$$

Since each scoring function is a linear function in parameter space, the best approach to derive the optimal weights is the method of least squares. The parameter values are determined by setting the gradient of the objective function L to zero. The gradient is given by the product of the vector derivative $\nabla_{\boldsymbol{\beta}_k} L$, defined by

$$\nabla_{\boldsymbol{\beta}_k} L = \left(\frac{\delta L}{\delta \beta_1}, \dots, \frac{\delta L}{\delta \beta_{n_{fk}}}\right)^T, \tag{2-42}$$

where subscript 'k' refers to the kth scoring functions of the vector-valued scoring function. The gradient of the objective function is given by

$$\frac{1}{2} \nabla_{\boldsymbol{\beta}_k} \cdot L\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}\right) =$$
$$\sum_{j=1}^{C} \sum_{n=1}^{N_j} \left[\left(f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right) - y_{k\,j,n}\right) \cdot \nabla(\boldsymbol{\beta}_k) \cdot f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right)\right] = 0, \tag{2-43}$$

Taking the derivative with respect to α_k yields

$$\frac{1}{2}\frac{\partial}{\partial \alpha_k} L\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}\right) =$$

$$\sum_{j=1}^{C}\sum_{n=1}^{N_j}\left[\left(f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right) - y_{k\,j,n}\right)\frac{\partial}{\partial b_k} f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right)\right] = 0. \tag{2-44}$$

Note that with equation (2-38)

$$\nabla_{\boldsymbol{\beta}_k} \cdot f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right) = \boldsymbol{x}_{P\,j,n}{}^{\boldsymbol{T}} \text{ and } \frac{\partial}{\partial b_k} f_k\left(\boldsymbol{\beta}_k, \alpha_k, \boldsymbol{x}_{P\,j,n}\right) = 1. \tag{2-45}$$

Hence, equation (2-43) can be written as

$$\sum_{j=1}^{C}\sum_{n=1}^{N_j}\left[\left(\boldsymbol{\beta}_k \boldsymbol{x}_{P\,j,n} + \alpha_k - y_{k\,j,n}\right) \cdot \boldsymbol{x}_{P\,j,n}\right] = 0 \tag{2-46}$$

and equation (2-44)

$$N \cdot \alpha_k + \sum_{j=1}^{C}\sum_{n=1}^{N_j}\left(\boldsymbol{\beta}_k \boldsymbol{x}_{P\,j,n} - y_{k\,j,n}\right) = 0. \tag{2-47}$$

Accordingly, equation (2-47) can be rewritten as

$$\alpha_k = \frac{1}{N}\sum_{j=1}^{C}\sum_{n=1}^{N_j}\left(y_{k\,j,n} - \boldsymbol{\beta}_k \boldsymbol{x}_{P\,j,n}\right). \tag{2-48}$$

Introducing the means

$$\langle y_k \rangle = \frac{1}{N}\sum_{j=1}^{C}\sum_{n=1}^{N_j} y_{k\,j,n} \tag{2-49}$$

and in analogy

$$\langle \boldsymbol{x}_P \rangle = \frac{1}{N}\sum_{j=1}^{C}\sum_{n=1}^{N_j} \boldsymbol{x}_{P\,j,n}, \tag{2-50}$$

yields for equation (2-48)

$$\alpha_k = \langle y_k \rangle - \boldsymbol{\beta}_k \langle \boldsymbol{x}_P \rangle. \tag{2-51}$$

These definitions allow to rearrange equation (2-46), yielding

$$\sum_{j=1}^{K_c}\sum_{n=1}^{N_j}\left(\boldsymbol{x}_{P_{j,n}}\boldsymbol{x}_{P_{j,n}}{}^T\boldsymbol{\beta}_k - \langle\boldsymbol{x}_P\rangle\boldsymbol{x}_{P_{j,n}}{}^T\boldsymbol{\beta}_k - y_{k_{j,n}}\boldsymbol{x}_{P_{j,n}} + \langle y_k\rangle\boldsymbol{x}_{P_{j,n}}\right) = 0, \tag{2-52}$$

which can be translated to

$$\sum_{j=1}^{K_c}\sum_{n=1}^{N_j}\left(\boldsymbol{x}_{P_{j,n}}\boldsymbol{x}_{P_{j,n}}{}^{\boldsymbol{T}}\boldsymbol{\beta}_k - \langle\boldsymbol{x}_P\rangle\cdot N\langle\boldsymbol{x}_P\rangle^{\boldsymbol{T}}\boldsymbol{\beta}_k - y_{k_{j,n}}\boldsymbol{x}_{P_{j,n}} + \langle y_k\rangle\cdot N\langle\boldsymbol{x}_P\rangle\right) = 0, \tag{2-53}$$

To further simplify the equation, the matrix

$$\langle\boldsymbol{x}_P\boldsymbol{x}_P{}^T\rangle = \frac{1}{N}\sum_{j=1}^{C}\sum_{n=1}^{N_j}\left(\boldsymbol{x}_{P_{j,n}}\cdot\boldsymbol{x}_{P_{j,n}}{}^T\right) \tag{2-54}$$

and the vector

$$\langle y_k\boldsymbol{x}_P\rangle = \frac{1}{N}\sum_{j=1}^{C}\sum_{n=1}^{N_j}\left(y_{k_{j,n}}\cdot\boldsymbol{x}_{P_{j,n}}\right) \tag{2-55}$$

are defined, which yields

$$(\langle\boldsymbol{x}_P\boldsymbol{x}_P{}^T\rangle - \langle\boldsymbol{x}_P\rangle\langle\boldsymbol{x}_P\rangle^T)\boldsymbol{\beta}_k = \langle y_k\boldsymbol{x}_P\rangle - \langle y_k\rangle\langle\boldsymbol{x}_P\rangle \tag{2-56}$$

or alternatively

$$\langle(\boldsymbol{x}_P - \langle\boldsymbol{x}_P\rangle)(\boldsymbol{x}_P - \langle\boldsymbol{x}_P\rangle)^T\rangle\boldsymbol{\beta}_k = \langle y_k(\boldsymbol{x}_P - \langle\boldsymbol{x}_P\rangle)\rangle. \tag{2-57}$$

Hence, equation (2-57) describes a system of linear equations

$$\boldsymbol{M}\cdot\boldsymbol{\beta}_k = \boldsymbol{v}_k, \tag{2-58}$$

where the covariance matrix $\boldsymbol{M}$ is defined by

$$\boldsymbol{M} = \langle\boldsymbol{x}_P\boldsymbol{x}_P{}^T\rangle - \langle\boldsymbol{x}_P\rangle\langle\boldsymbol{x}_P\rangle^T, \tag{2-59}$$

and the column vector $\boldsymbol{v}_k$ is given by

$$\boldsymbol{v}_k = \langle y_k(\boldsymbol{x}_P - \langle \boldsymbol{x}_P \rangle)\rangle. \tag{2-60}$$

By solving the system of linear equations, the optimal solution for $\boldsymbol{\beta}_k$ is determined. This allows in the second step to calculate $\boldsymbol{\alpha}_k$ through equation (2-51). The matrix $\boldsymbol{M}$ is independent of the scoring function. Hence, for solving any $\boldsymbol{\beta}_k$, the same $\boldsymbol{M}$ is used. In contrast, the vector $\boldsymbol{v}_k$ depends on the scoring function.
The parameters of all scoring functions $f_k(\boldsymbol{x}, \boldsymbol{\theta}_k)$ can be combined to one global parameter set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{(C-1)}\}$. Thus, the set contains all parameters of the classification-function $f(\boldsymbol{x})$ that need to be optimized.

The optimal parameters $\boldsymbol{\Theta}$, thus those with the smallest quadratic error, are obtained by solving an array of linear equations. All of which use the same coefficient matrix $\boldsymbol{M}$ but different right side vectors $\boldsymbol{v}_k$. Every linear equation system provides the optimal parameters $\boldsymbol{\theta}_k$ for the kth scoring function $f_k(\boldsymbol{x}, \boldsymbol{\theta}_k)$ of the classificatory function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\Theta})$. Therefore, the vector-valued classifier (VVC) is trained by directly calculating the parameters from the training dataset.

Each of the scoring functions of the VVC corresponds to linear regression function described in chapter 1.3.1. For the case of two classes, the VVC is identical to a classification based on a linear regression function. In this case, the class vectors and indicator matrix would be equal. However, for three or more classes classification by linear regression and the VVC is different. For linear regression, multiple independent binary classifiers are trained. Each of these is either trained to separate one class from the others (one-vs.-rest) or to separate between two specific classes (one-vs.-one). In contrast, the individual scoring functions of the VVC make no binary classifications, the classifications is achieved through the combination of all scoring functions. Because of its similarities to linear regression, the VVC also shares the same disadvantages like masking and the sensitivity to extreme outliers.

Like an artificial neural network (ANN), see chapter 1.3.2, the VVC is a multiclass classifier. Using quadratic features is roughly similar to an ANN consisting of single hidden layer. The difference to an ANN is that an ANN takes into account correlations of an arbitrary order whereas the VVC is limited to quadratic correlations. Another difference is the training process of both methods. The parameters of the VVC are optimized by solving a linear equation system that has a unique solution. The ANN on the other hand, requires an iterative optimization procedure, like the back-propagation algorithm described in chapter 1.3.2.4.

2.4.4.3 Regularization of the objective function

To be able to solve the linear equation system defined by equation (2-57) requires the covariance matrix to be *well-conditioned*. Regularization terms are a way to obviate potential *ill-conditioning* of the matrix. Furthermore, regularization is a method to prevent overfitting of the model. Chapter 1.3.3 describes regularization in greater detail. For systems of linear equations the common approach is to introduce a suitable *Tikhonov matrix* $\boldsymbol{\Gamma}$ as an L$_2$ norm to the objective function. For $\boldsymbol{\Gamma}$, a multiple of the identity matrix $\lambda \boldsymbol{I}$ was chosen, yielding

$$L\left(\underline{\boldsymbol{\beta}}, \boldsymbol{a}\right) = \sum_{j=1}^{C} \sum_{n=1}^{N_j} \left(\boldsymbol{f}\left(\underline{\boldsymbol{\beta}}, \boldsymbol{\alpha}, \boldsymbol{x}_{P_{j,n}}\right) - \boldsymbol{y}_{j,n}\right)^2 + \lambda \boldsymbol{I}. \tag{2-61}$$

The elements of the identity matrix $\boldsymbol{I}$ are unity along the diagonal and zero elsewhere. According to equation (2-61), the linear equation system defined by equation (2-57), becomes

$$[\langle \boldsymbol{x}_P {\boldsymbol{x}_P}^T \rangle - \langle \boldsymbol{x}_P \rangle \langle \boldsymbol{x}_P \rangle^T + \lambda \boldsymbol{I}] \boldsymbol{\beta}_k = \langle y_k (\boldsymbol{x}_P - \langle \boldsymbol{x}_P \rangle) \rangle, \tag{2-62}$$

which corresponds to the matrix notation

$$(\boldsymbol{M} + \lambda \boldsymbol{I}) \boldsymbol{\beta}_k = \boldsymbol{v}_k. \tag{2-63}$$

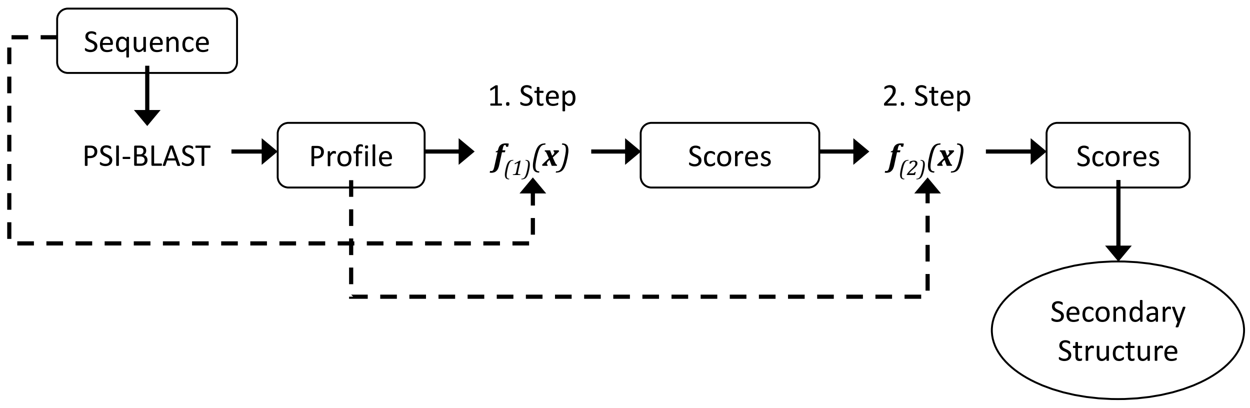

Figure 2-6: Flow diagram describing the vector-valued classifier implemented in *SPARROW^{+}.

2.5 Implementation of the classifier

The vector-valued scoring classifier outlined in chapter 2.4 is the basis of the protein secondary structure prediction program *SPARROW^{+}. *SPARROW^{+} is the successor to the previous version *SPARROW (Rasinski, 2011), which itself is a further development of the program SPARROW (Bettella et al., 2012). The name SPARROW is an abbreviation for ***S**econdary structure **P**redicting **ARR**ays of **O**ptimized **W**eights*. Figure 2-6 gives an overview of the implementation of the vector-valued classier.

Similar to other secondary structure prediction programs such as PSIPRED, Jpred and C3-Scoprion whose prediction procedure consists of multiple consecutive steps, *SPARROW^{+} utilizes two prediction steps. The first step generates a structure-to-sequence correlation, which is refined in the second step. Hence, the second step is a form of post processing by considering structure-to-structure correlations based on the predictions for several neighbouring residues. Assuming, that large segments with a uniform classification are correctly classified, eventual single outliers are considered potentially wrongly classified residues. Therefore, a residue within a large segment of residues being in a different secondary structure class may change its class. The second step is therefore a smoothing of the prediction from the first step.

Apart from the connection of input and output, both steps are entirely independent. Hence, different values for parameters such as number of classes or window size can be utilized. Concerning the number of classes, the only restriction is that the number of classes does not increase from the first to the second prediction step, since merging classes is easier than separating them (Rasinski, 2011).

The general setup of both prediction steps is similar; both utilize a vector-valued classifier and a symmetric sliding window. As described in section 2.3.1 the input feature vector considers the number of residues defined by the window size. Hence, for the prediction of a residue also its neighbouring residues are taken into account. Both prediction steps utilize PSI-BLAST profiles from the target sequence as input features. In the first step, the profiles are quadratic features, in the second step they are linear features. In both steps, different feature types are combined. In the first step quadratic profile and linear sequence features are combined while in the second step a combination of quadratic structure and linear profile features is used. To account for the additional linear features, an additional linear term is introduced into equation (2-20). Because of the linearization of the scoring functions only the size of total feature and parameter vectors increases. Accordingly, the equations to determine the parameters do not change.

2.5.1 Implementation of the first prediction step

The first prediction step of *SPARROW⁺ creates a sequence to structure correlation, which is realized by the vector-valued classifier $\boldsymbol{f}\colon \mathbb{R}^{20 \times r + 20 \times r} \rightarrow \mathbb{R}^{C-1}$. C is the number of classes and r is the size of the symmetric sliding window. Using a suitable learning set as defined in 2.1 the parameters are determined according to the procedure outlined in section 2.4.4. The number of adjustable parameters for every scoring function $f_k(\boldsymbol{x}, \boldsymbol{\theta}_k)$ in the first prediction step is equal to the number of the different features, as defined in equation (2-26). However, to account for the additional linear sequence features an additional linear term is added, yielding for the number of features and parameters

$$n_{fk(1)} = n_{\boldsymbol{P}} + n_{\boldsymbol{P}(q)} + n_{\boldsymbol{S}}. \tag{2-64}$$

As described in chapter 2.3, both, the linear profile and sequence features have the same dimension, and thus the same number of parameters

$$n_{\boldsymbol{P}} = n_{\boldsymbol{S}} = 20 \cdot r. \tag{2-65}$$

The number of non-redundant quadratic profile features is defined by equation (2-28), yielding

$$n_{\boldsymbol{P}(q)} = \frac{(20 \cdot r) \cdot (20 \cdot r + 1)}{2}. \tag{2-66}$$

Hence, the total number of features is given by

$$n_{fk(1)} = 20 \cdot r + \frac{(20 \cdot r) \cdot (20 \cdot r + 1)}{2} + 20 \cdot r.$$

Taking the parts of the PSI-BLAST profile defined by the sliding window, a profile matrix $\boldsymbol{P} \in \mathbb{R}^{20 \times r}$ is obtained. The sequence matrix $\boldsymbol{S} \in \mathbb{R}^{20 \times r}$ is obtained in the same way. From these matrices the linear feature vectors $\boldsymbol{x}_{\boldsymbol{P}(l)}$ and $\boldsymbol{x}_{\boldsymbol{S}(l)}$ are generated, with the linear notation of the matrices for the linear profile feature vector defined by

$$\boldsymbol{x}_{\boldsymbol{P}}{}^{T} = \left(P_{1,1}, \ldots, P_{1,20}, P_{2,1} \ldots, P_{r_1,20}\right) \tag{2-67}$$

and accordingly for the linear sequence feature vector

$$\boldsymbol{x}_{\boldsymbol{S}}{}^{T} = \left(S_{1,1}, \ldots, S_{1,20}, S_{2,1} \ldots, S_{r_1,20}\right). \tag{2-68}$$

From the linear feature vector, the quadratic feature vector can be determined, given by equations (2-27) and (2-30). The total feature vector is the concatenation of the different feature vectors

$$\boldsymbol{x}_{f(1)}{}^{T} = \left(\boldsymbol{x}_{\boldsymbol{P}}{}^{T} \parallel \boldsymbol{x}_{\boldsymbol{P}(q)}{}^{T} \parallel \boldsymbol{x}_{\boldsymbol{S}}{}^{T}\right), \tag{2-69}$$

which yields

$$\boldsymbol{x}_{f(1)}{}^{T} = \left(x_{\boldsymbol{P}_1}, \ldots, x_{\boldsymbol{P}_{n_P}}, x_{\boldsymbol{P}(q)_1}, \ldots, x_{\boldsymbol{P}(q)_{n_{P(q)}}}, x_{\boldsymbol{S}_1}, \ldots, x_{\boldsymbol{S}_{n_{S(l)}}}\right). \tag{2-70}$$

For a residue i a feature vector $\boldsymbol{x}_{f(1),i}$ is generated, from which an output vector $\boldsymbol{s}_{i(1)} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta})$ is calculated.

2.5.2 Implementation of the second prediction step

Based on the output of the first prediction step, the second step generates a structure to structure and structure to profile correlation. Analogous to the first prediction step, a vector-valued classifier is used for the classification. Consequently, the classifier makes a projection $\boldsymbol{f}: \mathbb{R}^{(C-1)\times r+20\times r} \rightarrow \mathbb{R}^{C-1}$. The only major difference to the first step is the combination of structure and profile features. The matrices $\underline{\boldsymbol{S}} \in \mathbb{R}^{(C-1)\times r}$ and $\boldsymbol{P} \in \mathbb{R}^{20\times r}$ describe the structure and the profile features of residues within the sliding window. In an analogous manner as in the first step, linear representations $\boldsymbol{x}_{\boldsymbol{S}}$ and $\boldsymbol{x}_{\boldsymbol{P}}$, of the structure and profile matrix are generated. The linear structure vector is given by the output of the first prediction step

$$\boldsymbol{x}_{\boldsymbol{s}}{}^{T} = \left(\underline{\boldsymbol{s}}_{1,1}, \dots, \underline{\boldsymbol{s}}_{1,K_{C(2)}}, \underline{\boldsymbol{s}}_{2,1} \dots, \underline{\boldsymbol{s}}_{r_2,K_{C(2)}}\right), \tag{2-71}$$

while the definition $\boldsymbol{x}_{P}$ is analogous to equation (2-67). The number of features of the second prediction step is the summation of the features.

$$n_{fk(2)} = n_{\boldsymbol{s}} + n_{\boldsymbol{s}(q)} + n_{\boldsymbol{P}}. \tag{2-72}$$

The number of linear profile features, given by equation (2-73), is identical to the first prediction step. The linear structure features are defined by

$$n_{\boldsymbol{s}} = (C-1)\cdot r. \tag{2-73}$$

The number of non-redundant quadratic structure features is defined by equation (2-28), yielding

$$n_{\boldsymbol{s}(q)} = \frac{\big((C-1)\cdot r\big)\cdot\big((C-1)\cdot r+1\big)}{2}. \tag{2-74}$$

The total number of features in the second prediction step is then

$$n_{f(2)} = (C-1)\cdot r + \frac{\big((C-1)\cdot r\big)\cdot\big((C-1)\cdot r+1\big)}{2} + 20\cdot r.$$

The total feature vector is the concatenation of the different feature vectors

$$\boldsymbol{x}_{f(2)}{}^{T} = \left(\boldsymbol{x}_{\boldsymbol{s}}{}^{T} \parallel \boldsymbol{x}_{\boldsymbol{s}(q)}{}^{T} \parallel \boldsymbol{x}_{\boldsymbol{P}}{}^{T}\right), \tag{2-75}$$

yielding

$$\boldsymbol{x}_{f(1)}{}^{T} = \left(x_{\boldsymbol{s}(l)_1}, \dots, x_{\boldsymbol{s}(l)_{n_{P(l)}}}, x_{\boldsymbol{s}(q)_1}, \dots, x_{\boldsymbol{s}(q)_{n_{P(q)}}}, x_{\boldsymbol{P}(l)_1}, \dots, x_{\boldsymbol{P}(l)_{n_{S(l)}}}\right). \tag{2-76}$$

2.6 Quality measures

To assess the prediction quality of *SPARROW$^+$ on an arbitrary test set, a quality measure has to be defined. These would also allow to compare *SPARROW$^+$ with other secondary structure prediction programs. To measure the prediction quality, the input vector as well as the classification need to be known. This together provides the information whether a residue was classified correctly or not. The dataset is expected to have the form $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid 1 \leq i \leq N\}$ in this case D is an arbitrary data set used for classification consisting of N residues. The prediction is done through the scalar product of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\Theta})$ with the individual class vectors. A prediction $\boldsymbol{x}_i \in D$ is correct if

$$\boldsymbol{x}_i \in \zeta_\sigma \wedge \sigma = \max_j\left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \boldsymbol{y}_j\right), 1 \leq \rho \leq N^C \tag{2-77}$$

2.6.1 Accuracy index Q_C

The Q_C-accuracy index is a simple measure of the prediction quality of a classifier on a dataset D. It puts the number of correct predictions into relation to the size of the dataset. With equation (2-77), the Q_C-accuracy is defined by

$$Q_K = \frac{\sum_{\sigma=1}^{K} \left|\left\{\boldsymbol{x}_i \in D \middle| \boldsymbol{x}_i \in \zeta_\sigma \wedge \sigma = \max_j\left(\boldsymbol{f}(\boldsymbol{x}_i) \cdot \boldsymbol{y}_j\right)\right\}\right|}{M} \tag{2-78}$$

In a similar way the Q_3-accuracy of an individual class ζ_σ in the dataset D can be calculated by

$$q_\sigma = \frac{\left|\left\{\boldsymbol{x}_i \middle| \boldsymbol{x}_i \in D \in \zeta_\sigma \wedge \sigma = \max_j\left(\boldsymbol{f}(\boldsymbol{x}_i) \cdot \boldsymbol{y}_j\right)\right\}\right|}{|\{\boldsymbol{x}_i | \boldsymbol{x}_i \in \zeta_\sigma\}|} \tag{2-79}$$

Using the Q_C-accuracy, the percentage of correctly predicted residues in the dataset can be calculated by multiplying with 100.

2.6.2 Generalized Matthews correlation coefficient

The *generalized Matthews correlation coefficient* (MCC) (Gorodkin, 2004) is a more sophisticated quality measure than the Q_C-accuracy. It is an extension of the MCC (Matthews, 1975) to multiple classes, using an extension of the *Pearson's correlation coefficient*. The generalized MCC is calculated through the values of a *confusion* matrix $\boldsymbol{\Omega}$. One element of this matrix is defined by:

$$\Omega_{\sigma\rho} = \left|\left\{\boldsymbol{x}_i \in D \middle| \boldsymbol{x}_i \in \zeta_\sigma \wedge \sigma = \max_j\left(\boldsymbol{f}(\boldsymbol{x}_i) \cdot \boldsymbol{y}_j\right)\right\}\right| \tag{2-80}$$

The elements $\Omega_{\sigma\rho}$ tell how much of the input data $\boldsymbol{x}$, belonging to class ζ_σ are also classified as ζ_σ. The diagonal matrix elements are the correctly classified input data. In case of a perfect prediction each diagonal element $\omega_{\sigma\rho}$ would be identical to the size of the class ζ_σ in D. Hence all other matrix elements were equal to zero. Based on the confusion matrix the MCC value can be calculated through

$$R_{N^c} = \frac{\sum_{\alpha=1}^{C}\sum_{\beta=1}^{C}\sum_{\gamma=1}^{C}\left(\Omega_{\alpha\alpha}\Omega_{\beta\gamma} - \Omega_{\alpha\beta}\Omega_{\gamma\alpha}\right)}{\sqrt{\sum_{\alpha=1}^{C}\left(\sum_{\beta=1}^{C}\Omega_{\alpha\beta}\sum_{\gamma=1}^{C}\sum_{\delta=1,\gamma\neq\alpha}^{C}\Omega_{\delta\gamma}\right)}\sqrt{\sum_{\alpha=1}^{C}\left(\sum_{\beta=1}^{C}\Omega_{\beta\alpha}\sum_{\gamma=1}^{C}\sum_{\delta=1,\gamma\neq\alpha}^{C}\Omega_{\gamma\delta}\right)}}. \tag{2-81}$$

The values for R_C are in a range of $-1 \leq R_C \leq 1$, whereby 1 corresponds to perfect classification. The value -1 is asymptotically reached in the extreme misclassification case of a confusion matrix W with all zeros but in two symmetric entries $\Omega_{\rho\sigma}$ $\Omega_{\sigma\rho}$ (Jurman et al., 2012).

The MCC of an individual class ζ_σ, with $\sigma = \alpha$, in the dataset D can be calculated by

$$r_\sigma = \frac{\sum_{\beta=1}^{N^c}\sum_{\gamma=1}^{N^c}\left(\Omega_{\alpha\alpha}\Omega_{\beta\gamma} - \Omega_{\alpha\beta}\Omega_{\gamma\alpha}\right)}{\sqrt{\sum_{\beta=1}^{N^c}\Omega_{\alpha\beta}\sum_{\gamma=1}^{N^c}\sum_{\delta=1,\gamma\neq\alpha}^{N^c}\Omega_{\delta\gamma}}\sqrt{\sum_{\beta=1}^{N^c}\Omega_{\beta\alpha}\sum_{\gamma=1}^{N^c}\sum_{\delta=1,\gamma\neq\alpha}^{N^c}\Omega_{\gamma\delta}}} \tag{2-82}$$

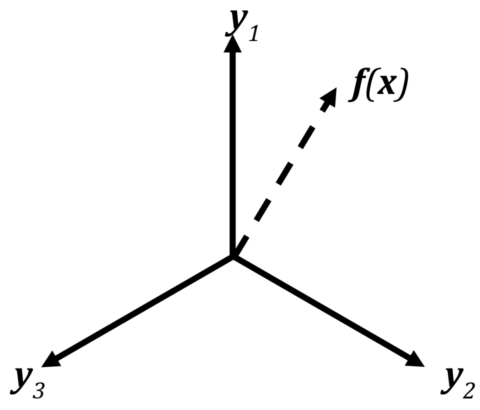

Figure 2-7: Depiction of the generalized tetrahedron for C=3.$\boldsymbol{y}_1$,$\boldsymbol{y}_2$ and $\boldsymbol{y}_3$ are the class vectors and f is the result of the vector-valued scoring function.

2.6.3 Confidence measure of the vector-valued scoring function $\boldsymbol{f}$

To assess the reliability of a prediction a measure of confidence is needed. This measure should correspond to probability of correctness. For multi-class prediction the probabilities for every class assignment are need. The confidence measure employed by *SPARROW$^+$ is first outlined for three classes and then generalized for n-classes.

2.6.3.1 Three classes.

The ideal class vectors for three classes are the vectors $\boldsymbol{y}_i$ in a two-dimensional space, with $i=1,2,3$, with $\boldsymbol{y}_i\boldsymbol{y}_i=1$ and $\boldsymbol{y}_i\boldsymbol{y}_j=1/2$ for $i\neq j$. Therefore, $\sum_{i=1,2,3}\boldsymbol{y}_i=\boldsymbol{0}$ and the sum of the projections of $\boldsymbol{f}$ with the $\boldsymbol{y}_i$ vanishes. If $\boldsymbol{f}=\boldsymbol{0}$ the probabilities for the three classes are all equal: $p_1=p_2=p_3=1/3$. If $\boldsymbol{f}=\boldsymbol{y}$, $p_i=1$ and $p_k=0$ $k\neq i$.

If $|\boldsymbol{f}|=1$ and $\boldsymbol{f}\boldsymbol{y}_3=-1$ then $\boldsymbol{f}\boldsymbol{y}_3=1/2=\boldsymbol{f}\boldsymbol{y}_2$ In this case we should have $p_3=0$ and $p_1=1/2=p_2$. The expression

$$p_i=\frac{1}{3}(1+\boldsymbol{y}_i\boldsymbol{f}) \tag{2-83}$$

fulfils for $|\boldsymbol{f}|=1$ and $\boldsymbol{f}\boldsymbol{y}_3=-1$ the conditions $p_1=1/2=p_2$ and $p_3=0$. For $\boldsymbol{f}=\boldsymbol{0}$ it also fulfils $p_1=p_2=p_3=1/3$. However, it is wrong for $\boldsymbol{f}=\boldsymbol{y}_1$, since $p_1=2/3$ and $p_2=p_3=1/6$. In particular the latter ($p_2=p_3=1/6$) should not happen. Alternatively, p_i can be defined by

$$p_i=\max\left\{0,\frac{1}{3}(1+2\boldsymbol{y}_i\boldsymbol{f})\right\}. \tag{2-84}$$

As long as all three terms $1+2\boldsymbol{y}_i\boldsymbol{f}$ for $i=1,2,3$ are positive, the sum

$$P=\sum_{i=1}^{3} p_i \tag{2-85}$$

is unity and the p_i are properly defined probabilities. If one p_i vanishes (i.e. $\boldsymbol{y}_i\boldsymbol{f}<-1/2$) $P>1$. To make in these cases sure that the probabilities are normalized to unity we redefine them as follows: 1

$$\hat{p}_i=\frac{p_i}{p}. \tag{2-86}$$

For $\boldsymbol{f}=\boldsymbol{y}_1$ we have $\boldsymbol{f}\boldsymbol{y}_2=-1/2=\boldsymbol{f}\boldsymbol{y}_3$ and therefore $p_2=p_3=0$. For $\boldsymbol{f}\boldsymbol{y}_2=1/2=\boldsymbol{f}\boldsymbol{y}_3$ we have $\boldsymbol{f}\boldsymbol{e}_3=-1$, $p_3=0$ and $p_1=2/3=p_2$. Hence, we have to use the normalized probabilities $\hat{p}_i$. The whole procedure is also valid if $|\boldsymbol{f}|>1$.

For n classes the ideal class vectors $\boldsymbol{y}_i$, obey $\boldsymbol{y}_i\boldsymbol{y}_i=1$ and $\boldsymbol{y}_i\boldsymbol{y}_j=-1/(n-1)$ for $i\neq j$. The expression analogue to equation (2-84) is

$$p_i=\max\left\{0,\frac{1}{n}(1+(n-1)\boldsymbol{y}_i\boldsymbol{f})\right\}. \tag{2-87}$$

2.6.3.2 Confidence of predicted class

By definition the predicted class, has the highest probability p_i of all classes. Hence, the value of p_{pred} is in the range $1/n<p_{pred}<=1$. Normalizing this range gives the confidence c.

$$c_{pred}=\left(p_{pred}-\frac{1}{n}\right)\cdot\frac{n}{(n-1)}. \tag{2-88}$$

3 Results

This chapter describes the results of protein secondary structure predictions made with the program *SPARROW$^+$. The results are separated into three parts that comprise setup, validation and benchmark. The setup part describes the optimization of the hyperparameters and the stepwise assembly of the vector-valued classifier used by *SPARROW$^+$. An initial one-step classifier using only a single type of features, namely sequence profiles, is successively enhanced yielding finally a two-step classifier utilizing multiple feature types. The hyperparameters of *SPARROW$^+$ are the class reduction scheme, size of the sliding window, type of BLAST database and the regularization parameter. These parameters are not optimized by the objective function, but can only be manually tuned by comparing the classifier performance on datasets not used for training. The approach here is an 8-fold cross-validation. After the determination of the hyperparameters, the basic setup of *SPARROW$^+$ is completed. The last part of the setup phase is a description of the stepwise assembly of the final layout of the vector-valued classifier. This gives insights into how each enhancement of the classifier affects the prediction quality. The enhancements consist of an additional set of linear sequence features, a second prediction step and finally an additional set of linear profile features in the second prediction step.

After assembling the classifier and determining its parameters, the validation part describes the prediction performance. This includes the prediction quality per amino acid and the confidence measure. The idea behind the confidence measure is to assess the ambiguity of a prediction. Correlating the confidence measure and the prediction quality can provide a quality estimate of a prediction.

Lastly, the benchmark part describes the performance of *SPARROW$^+$ in comparison to other popular protein secondary structure prediction programs. The prediction quality of all programs is evaluated for a number of datasets used for protein structure prediction.

The properties of the dataset on which *SPARROW$^+$ is trained and validated are critical. To assess the prediction quality of a secondary structure prediction method the Q_3-accuracy and generalized MCC value are used, see equation (2-78) and (2-81). For the class specific prediction quality, equations (2-79) and (2-82) are used for the Q_3-accuracy and generalized MCC value, respectively.

No. protein domains ASTRAL40	12,109	
No. domains filtered	11,147	
No. residues filtered	2,056,403	
Secondary structure classes	Absolute no.	Relative no. [%]
α-helix	687,211	33.4
3_{10}-helix	76,704	3.7
π-helix	11,135	0.5
β-strand	439,048	21.5
Coil	417,948	20.3
Turn	226,673	11.0
Bend	176,643	8.6
Isolated β-bridge	21,041	1.0

Table 3-1: Composition of the dataset. Listed is the total size of version 2.03 of the filtered ASTRAL40 (Berman et al., 2003) dataset and its absolute and relative secondary structure composition. The secondary structure classes were assigned with DSSP (Kabsch and Sander, 1983)

3.1 Dataset

For training and validation, version 2.03 of the ASTRAL40 dataset (Berman et al., 2003; Fox et al., 2014) is used. The dataset is processed as described in chapter 2.1. All coiled coil, membrane and cell surface proteins as well as all peptides and short proteins with a length less than 50 residues are removed. Furthermore, special amino acids are replaced by their standard amino acid analogues. The filtered dataset contains 11,147 protein domains with a total of 2,056,403 residues. With the program DSSP (Kabsch and Sander, 1983), every residue is assigned into one of eight secondary structure classes. The details and secondary structure composition of the dataset is listed in Table 3-1.

In the dataset, the most abundant classes are α-helix (33.4 %), β-strand (21.5 %) and coil (20.3 %). Apart from these three classes, only turn (11.0 %) and bend (8.6 %), whose definitions are closely related to that of coil, have a greater occurrence. The remaining three structure classes are 3_{10}-helix (3.7 %), isolated β-bridge (1.0 %) and π-helix (0.5 %). Notably, there is a major difference in the population of the helix classes. The α-helix population is one

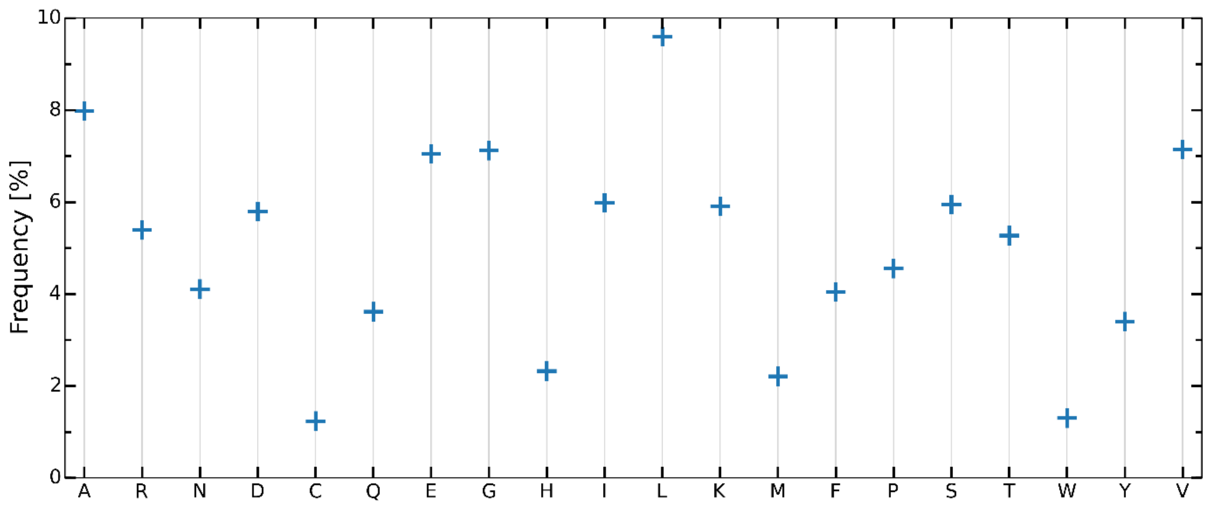

Figure 3-1: Amino acid composition of the processed ASTRAL40 (Berman et al., 2003; Brenner et al., 2000; Chandonia et al., 2002; Chandonia et al., 2004) dataset. The figure shows, the frequency with which an amino acid occurs in version 2.03 of the processed ASTRAL40 dataset described in Table 3-1 in chapter 3.1.

magnitude larger than the 3_{10}-helix population and approximately two magnitudes larger than the π-helix population.

The vast difference in the sizes of the class populations shows that the different classes are extremely unbalanced. Hence, by merging classes together a more balanced population distribution can be achieved.

The amino acid composition of the dataset, shown in Figure 3-1, does not deviate from the amino acid frequencies of the *UniProt* (Consortium, 2014) database from 15.08.2015. The rarest amino acids are cysteine (C) and tryptophan (W), closely followed by histidine (H) and methionine (M). The most abundant amino acids are alanine (A) and leucine (L).

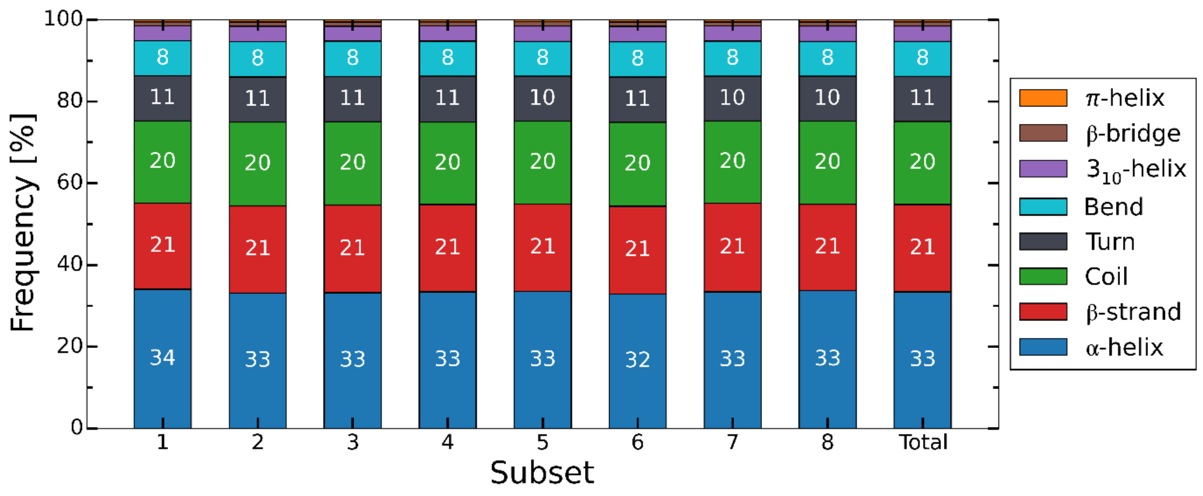

Figure 3-2: Secondary structure composition of the complete processed ASTRAL40 (Berman et al., 2003; Brenner et al., 2000; Chandonia et al., 2002; Chandonia et al., 2004) dataset and its eight subsets. The secondary structure was assigned with the program DSSP (Kabsch and Sander, 1983). Because of its very small size, the π-helix class is barely visible at the top of the bars.

3.2 Cross-validation

To obtain statistical measures of the variations in prediction quality, a cross-validation is performed. Therefore, the filtered ASTRAL40 dataset is split into eight separate subsets of about equal size. One subset is designated the prediction set while the remaining subsets are merged into the training set. This procedure is alternated for all subsets, leading to an 8-fold cross-validation.

For a successful cross-validation, the subsets need to have similar compositions of SCOP (Andreeva et al., 2014) classes and contain approximately the same number of residues. Hence, in a first step, the filtered ASTRAL40 dataset is ordered by protein size, and in a subsequent step, the proteins are successively distributed over the eight subsets. The described allocation procedure achieves a very similar secondary structure and SCOP class composition for the different subsets, as Figure 3-2 and Figure 3-3 respectively illustrate.

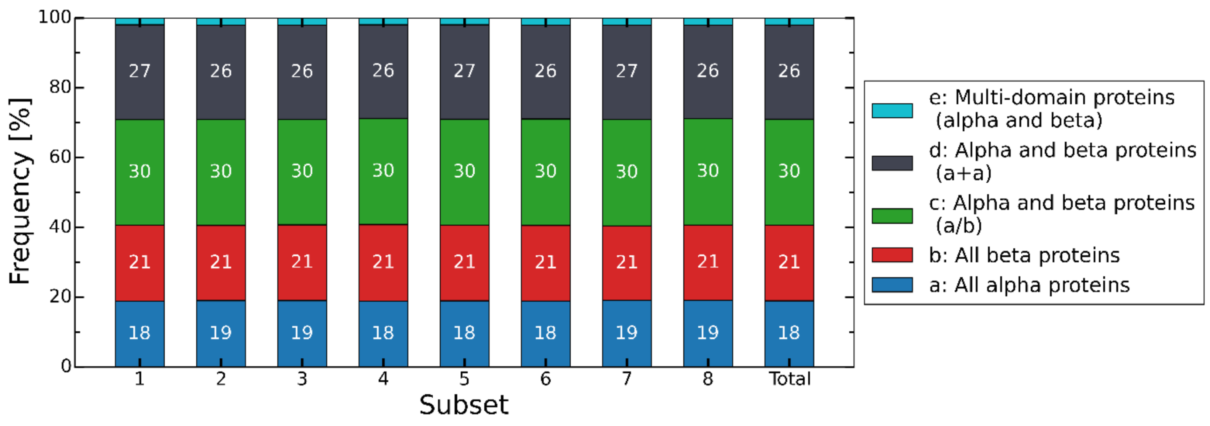

Figure 3-3: SCOP (Andreeva et al., 2014) class composition of the complete dataset and its eight subsets. After processing the ASTRAL40 (Berman et al., 2003; Brenner et al., 2000; Chandonia et al., 2002; Chandonia et al., 2004) dataset, only five SCOP classes remain. The classes comprise proteins consisting of α-helices (**a**), β-sheets (**b**), mainly parallel β-sheets (**c**), mainly anti-parallel β-sheets (**d**) and two or more domains from different classes (**e**).

3.2.1 Significant differences

In all cross validations conducted for the various setups of the vector-valued classifier, the same subset compositions initially generated are always used. This makes it possible to calculate the significance of the performance difference amongst the classifiers by means of the *t*-test with paired samples. The deviations in prediction performance are similar for all classifiers. Hence, the predictions for some subsets always performed better or worse. Therefore, the differences between some classifiers, although at times extremely small, are still highly significant. For results presented here, the *P* value of a two-tailed pairwise *t*-test is always below 0.0001.

3.3 Setup

The aim of the setup phase was twofold: Assessing the influence of the different hyperparameters and classifier enhancements on the prediction quality and the tuning of the hyperparameters. The hyperparameters of interest were the value of the regularization parameter λ, the size of the sliding window and the choice of the BLAST database and class reduction scheme. To tune the hyperparameters, only the first step of *SPARROW$^+$ was used. The only input were quadratic profile features and the window size was set to 11 residues in

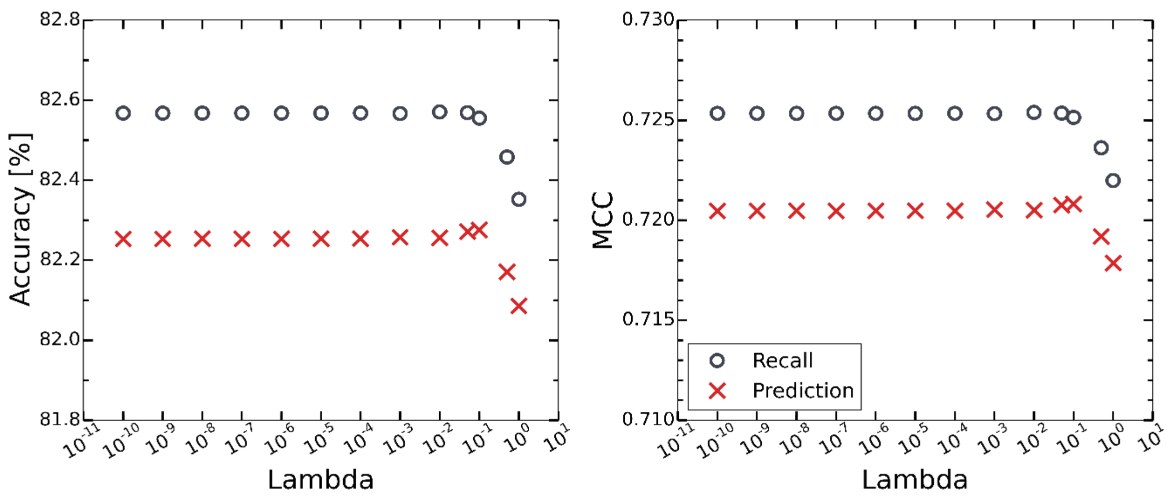

Figure 3-4: Influence of the regularization parameter λ on prediction and recall quality. For prediction and recall, the first step of *SPARROW$^+$ is used with a window size of 11 residues. The *UniRef90* BLAST database is used for profile generation and the *mixed* class reduction scheme is used for super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the generalized MCC value (equation(2-81)).

accordance to the approach used for *SPARROW (Rasinski, 2011) to determine hyperparameters. After the determination of the hyperparameters, the enhancements were successively introduced until the classifier reached its final setup. The effect of the enhancements were tested on a selection of different window sizes.

3.3.1 Regularization parameter

Training a classifier bears the risk of overfitting, which results in a classifier perfectly able to recall the training dataset but unable to generalize to perform accurate predictions of unknown data. This problem also applies to the vector-valued classifier. The risk of overfitting increases with the number of adjustable parameters but can be mitigated by training datasets with a large sample size. Another approach to prevent overfitting is to use regularization terms, as described in chapter 1.3.3.2, in the optimization term of the coefficients of the scoring function. For the determination of the coefficients of the vector-valued classifier, see equation (2-63), a regularization term in the form of an L_2 norm is used. The extent of regularization is adjusted by the regularization parameter λ. Increasing the value of λ diminishes the weights of insignificant features in the scoring function, allowing only important features to possess large weights. Besides preventing overfitting, regularization

also prevents the occurrence of near singularities when solving the system of linear equations system to determine the coefficients.
The recall and prediction quality for different λ values is depicted in Figure 3-4. In the case of overfitting, the difference between recall and prediction quality would decrease at larger λ values. However, the quality of recall and prediction was largely unaffected by the regularization parameter λ. The difference was constant; only at λ values of 0.01 was a slight decrease in the difference observable. At λ values greater than 1, both prediction and recall quality degrades equally. Thus, the choice of λ has no major effect on the classification quality. The difference of 0.4 % in the Q_3-accuracy (equation (2-78)) and 0.005 in the generalized MCC value (equation (2-81)) implies that a greater number of parameters are used than would be necessary. Yet, overfitting is prevented by a large sample size compared to the number of adjustable parameters. In the case of a vector-valued classifier using a window size of 11, there are 24,531 adjustable parameters. However, the sample size of about 1.8 10^6 is still nearly two orders of magnitude larger. For all following predictions, a λ value of 0.01 is chosen, with no distinction between linear and quadratic features. In the second prediction step, based primarily on quadratic secondary structure features, a λ value of 10^{-10} is used for all types of features. Such a small λ is taken, since the number of parameters is greatly diminished in the second step. Structure features have only two dimensions compared to the twenty dimensions of sequence or profile features.

3.3.2 BLAST databases

Profile features were obtained from the *position specific scoring matrices* (PSSM) generated for a given protein sequence by the program PSI-BLAST, as described in chapter 2.3.2. Through a series of local pairwise sequence alignments, PSI-BLAST calculates the odds that specific amino acid types occur at equivalent sequence positions in a series of related proteins. A positive profile value of an amino acid type at a specific sequence position indicates that this amino acid is likely to occur at that position, whereas negative values mean it is unlikely. Values around zero indicate that no preference in either direction exists. Hence, the profile values provide information about the conservation of an amino acid type. High positive profile values show a high degree of conservation whereas values close to zero mean an amino acid is not conserved. High negative profile values indicate a form of anti-conservation.

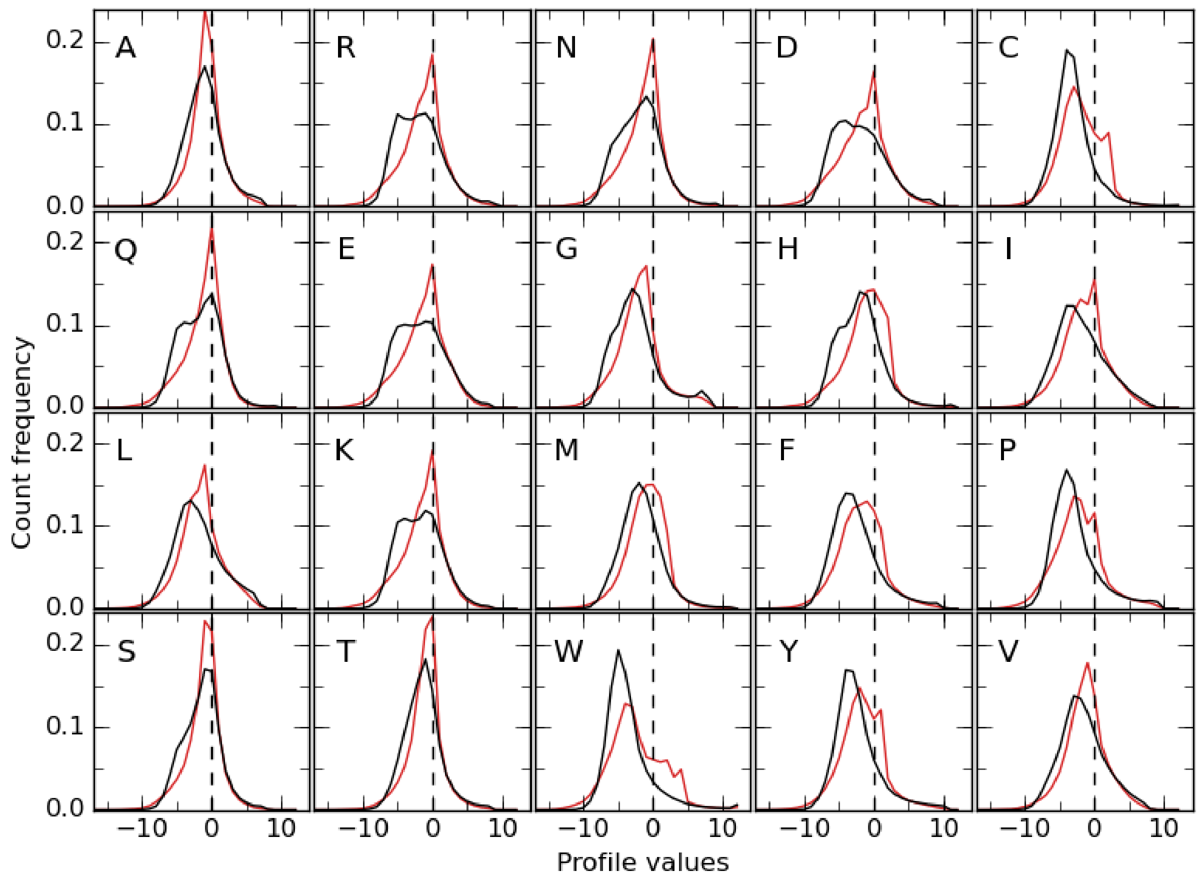

Figure 3-5: Normalized distribution of PSI-BLAST (Altschul et al., 1997) profile values per amino acid type generated from the filtered *UniRef90* (Suzek et al., 2007) (**black**) and the *non-redundant* (**red**) BLAST database. The particular amino acid is specified by one letter code and the dashed line marks a profile value of zero. Profile values can vary between -17 and 14.

To perform sequence alignments, PSI-BLAST needs to search a BLAST database. Various databases are available. Here, two were tested, the *non-redundant* (*nr*) (Altschul et al., 1997) (downloaded 19.3.2015) and the *UniRef90* (downloaded 11.3.2015) BLAST databases. The *non-redundant* database is as its name implies non-redundant, therefore no sequence occurs more than once. The *UniRef90* database clusters all protein sequences with at least 90 % sequence-identity together. From these clusters, one representative sequence is chosen. Before generating profiles with PSI-BLAST, the *UniRef90* database was filtered with the program *pfilt* (Jones and Swindells, 2002) to mask coiled-coil, transmembrane and unordered sequence regions. These masked regions are therefore ignored by PSI-BLAST.

The normalized profile value distribution per amino acid and BLAST database is depicted in Figure 3-5. The profile value distributions differ~~s~~ significantly between the *non-redundant* and the filtered *UniRef90* BLAST databases. For most amino acids, the profile values generated via

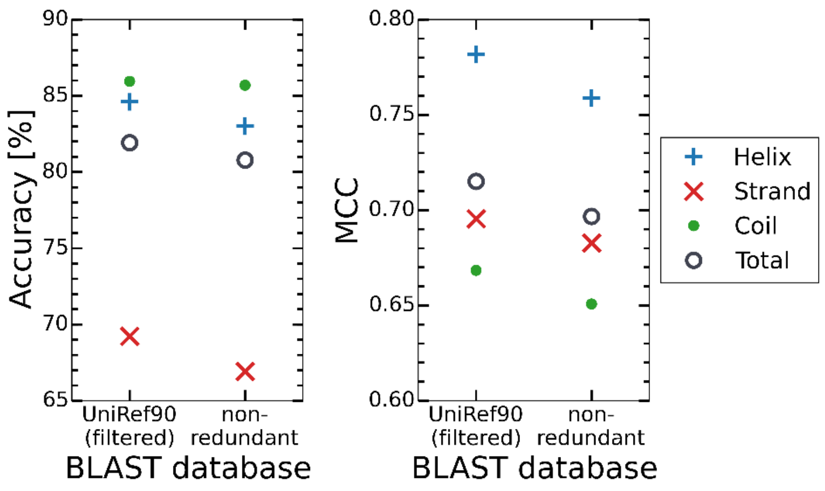

Figure 3-6: Prediction quality achieved with the filtered *UniRef90* (Suzek et al., 2007) and the *non-redundant* BLAST databases. For prediction, the first step of *SPARROW$^+$ is used with a window size of 11 residues. The *mixed* class reduction scheme is used for super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil separately.

the *non-redundant* database often have a distinct maximum at values of -1 or 0. In addition, the shape of the distributions is narrow and skewed negatively. However, most aromatic amino acids (phenylalanine (F), tyrosine (Y) and tryptophan (W)) as well as cysteine (C) and proline (P) have a much broader distribution with a second maximum.

Profile value distributions obtained from the *UniRef90* database are mostly broader and the maximum occurs at more negative profile values. This is especially pronounced for charged amino acids and their amine counterparts, i.e. asparagine (N) and glutamine (Q). One can discriminate those with a well-defined maximum (Asparagine (N) and histidine (H)) and those without (Arginine (R), aspartate (D), glutamine (Q), glutamate (E) and lysine (K)). The hydroxyl (Serine (S) and threonine (T)) and the aliphatic amino acids (Alanine (A), glycine (G), isoleucine (I), leucine (L) and valine (V)) show the fewest differences to the *nr* profile value distributions. The shape for methionine (M) is also very similar for both databases and is only shifted to more negative values for the *UniRef90* profiles.

All aromatic amino acids (Phenylalanine (F), tryptophan (W) and tyrosine (Y)) as well as cysteine (C) and histidine (H) show major losses in positive profile values and a distinct maximum at negative values. However, the opposite, a gain of positive values, albeit minimal, could only be observed for glycine (G).

The similar shapes of the distributions of positive profile values show that the extent of conserved amino acids is equal for both BLAST databases. The shift of profile values from around zero to more negative values suggests a more pronounced anti-conservation in *UniRef90* profiles. In contrast, profiles from the *nr* database lack the distinction between anti-conservation and not conserved. Taking the overall occurrence of the amino acids into account, see Figure 3-1, most of the rarest occurring amino acids (C, W, H, P, Y, F, M) have more positive values in the *nr* profiles. Since the *nr* database is non-redundant, it contains far more sequences with high similarity to each other, which could inflate the odds for rare amino acids to occur. The major differences in profile value distribution between the *nr* and filtered *UniRef90* BLAST databases manifests in significant prediction quality differences. A higher prediction quality is achieved using the filtered *UniRef90* BLAST database. The Q_3-accuracy (equation (2-78)) is 1 % and the generalized MCC value (equation (2-81)) 0.1 higher. Considering the individual super-classes, the usage of the filtered *UniRef90* database leads to an equally higher accuracy for the helix and strand super-classes but not the coil super-class. Yet, the gains in the MCC value are similar for all three super-classes.
Hence, although the number of true positive coil classifications did not change, the number of true positive helix and strand classifications increased by reducing the number of false positive coil classifications. Apparently, the more distinct difference between not conserved and anti-conservation of the filtered *UniRef90* profiles facilitated the separation of the helix and strand super-classes from the coil super-class.

3.3.3 Secondary structure class reduction schemes

In the most common secondary structure assignment program DSSP (Kabsch and Sander, 1983) a residue is assigned to one of eight different secondary structure classes. The eight classes comprise the three helix classes α-, 3_{10}-, π-helix and the classes β-strand, turn, bend, isolated β-bridge and random coil. Commonly, only three classes are used, namely helix, strand and coil. In chapter 2.2, class reduction schemes are introduced to reduce the eight secondary structure classes to three super-classes. To merge the different classes, a variety of class reduction schemes are possible. A selection of common class reduction schemes is shown Table 2-1 in chapter 2.2.
The choice of the class reduction scheme influences the prediction quality, as Figure 3-7 illustrates. The total prediction quality differs by about 2 % in Q_3-accuracy (equation (2-78))

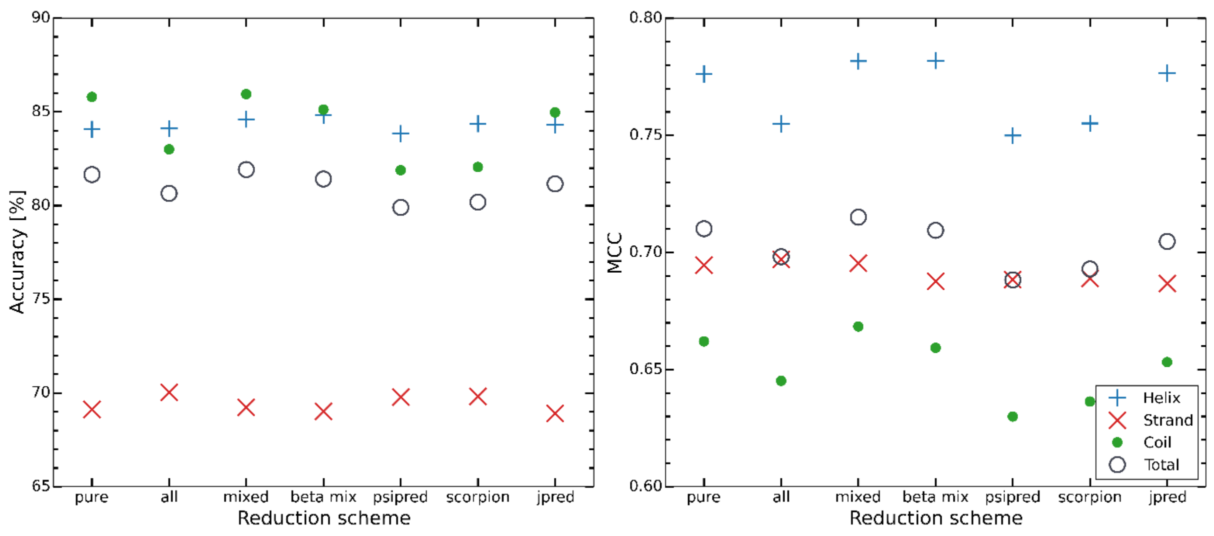

Figure 3-7: Prediction quality of different class reduction schemes. A description of the class reduction schemes is given in Table 2-1 in chapter 2.2. The class reduction schemes of the alternative methods PSIPRED (Jones, 1999), C3-Scorpion (Yaseen and Li, 2014) and Jpred (Cuff and Barton, 2000) are also provided. For prediction, the first-step of *SPARROW$^+$ with a window size of 11 residues is used and the *UniRef90* BLAST database is used for profile generation. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil.

between the different schemes. The *mixed* class reduction scheme yields the overall best results. Especially the coil super-class achieves the highest quality in this scheme. The *beta mix*, *psipred*, *scorpion* and *Jpred* class reduction schemes have in common, that both the β-strand and isolated β-bridge class belong to the strand super-class. Although not obvious by the Q_3-accuracy, the MCC value (equation (2-81)) shows that this results in a lower prediction quality for the strand super-class.

In protein crystallography, the 3_{10}-helix is often associated with the helix super-class. The question of which class to merge the 3_{10}-helix with is not a trivial one. For whichever super-class is chosen, 3_{10}-helices should predominantly be classified as being part of the chosen super-class, whether it is helix or coil. Merging the 3_{10}-helix class with the coil super-class leads to fewer misclassifications than merging it with the helix super-class, as shown in Figure 3-8. If the 3_{10}-helix is part of the coil super-class, the classifications are up to 80 % correct. In contrast, if it is part of the helix super-class, only about 40 % are correctly classified. Thus, the 3_{10}-helix seems to have more in common with the coil than with the helix super-class. The higher prediction quality of the *Jpred* class reduction scheme compared to the *psipred* scheme shows, that α- and 3_{10}-helix should not be in the same super-class. In contrast, having the π- and α-helix in the same super-class improves prediction quality, as

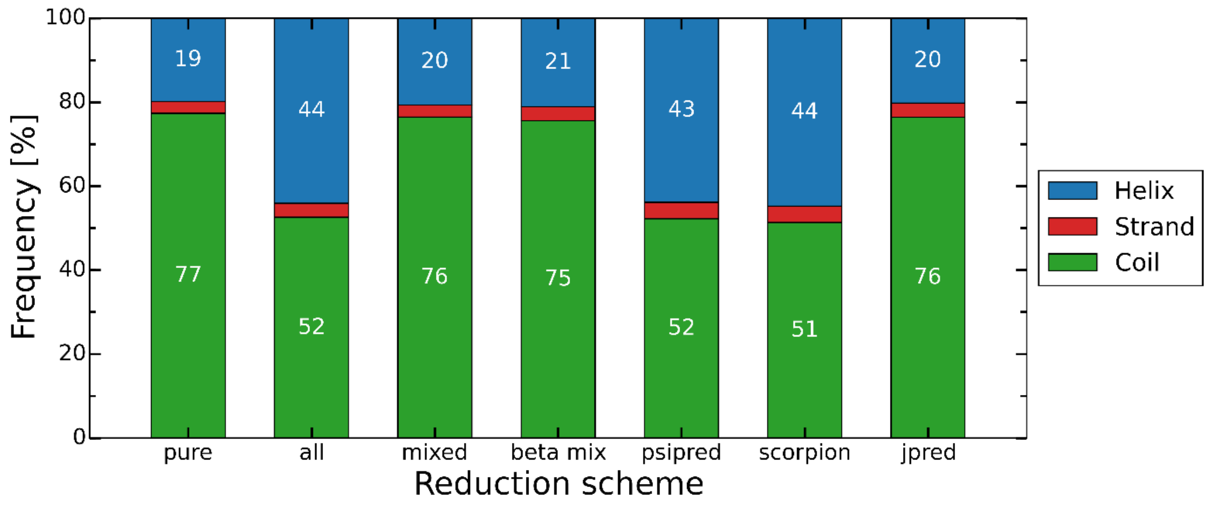

Figure 3-8: Distribution of 3_{10}-helix class classifications into the three super-classes depending on different class reduction schemes. A description of the class reduction schemes is given in Table 2-1 in chapter 2.2.

evident by the prediction quality differences between the *pure* and *mixed* to the *all*, *psipred* and *scorpion* class reduction schemes. Interestingly, the strand super-class always has the lowest Q_3-accuracy but a higher MCC value than the coil super-class, which achieves high Q_3-accuracy. Hence, the coil super-class prediction has a high ratio of true positives but also of false positives.

In summary, the following assumptions are supported: Firstly, there are major differences between α- and 3_{10}-helix, and secondly, despite their related names, isolated β-bridges and β-strands have little in common. The *mixed* class reduction scheme outperforms every other class reduction scheme in terms of prediction quality. Hence, this class reduction scheme is used for all further predictions described in the following chapters.

3.3.4 Window size

The window size is a critical classifier parameter, which has a direct impact on the prediction quality. The window size defines how many neighbouring residues are considered in the classification. In previous works (Bettella, 2009), it could be shown, that the prediction quality and window size are correlated non-linearly. While overall, a larger window improves prediction quality, the rate of improvement decreases with successively increasing window size. A larger window size might improve prediction quality by capturing more residue interactions within the window. However, with increasingly larger window sizes, the

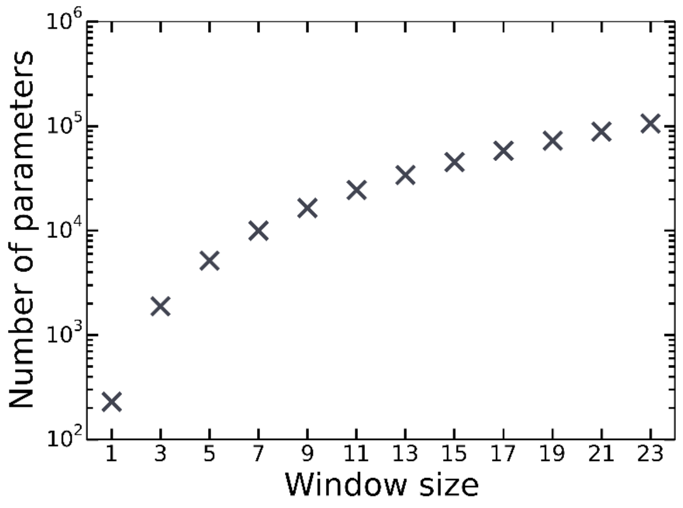

Figure 3-9: Logarithm of the number of parameters (equal to number of features) as a function of window size for the first step of *SPARROW$^+$.

computational demands grow geometrically, illustrated in Figure 3-9. Hence, achieving high prediction quality with small window sizes is of major interest.

Figure 3-10 illustrates, that the window size has indeed a major influence on the prediction quality. As expected, a larger window size results in a higher prediction quality. The plots suggest a logarithmic dependency between the prediction quality and window size. Interestingly, even a window size of 1 is sufficient to achieve a Q_3-accuracy (equation (2-78)) of nearly 60 % and a generalized MCC value (equation (2-81)) of 0.35. Hence, the central

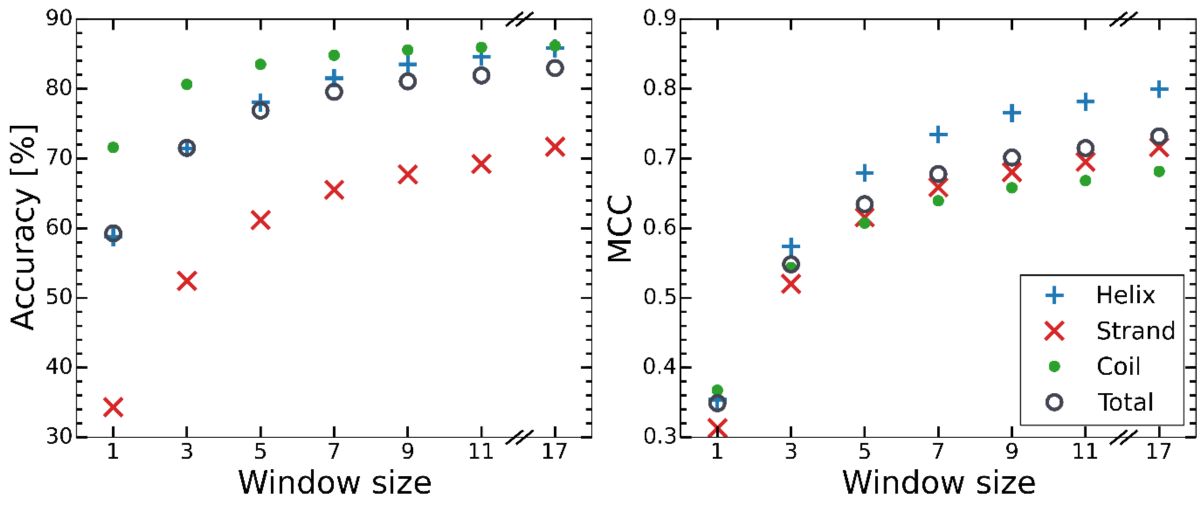

Figure 3-10: Prediction quality using linear and quadratic PSI-BLAST profile features as a function of window size. For prediction the first-step of *SPARROW$^+$ utilizing the *UniRef90* BLAST database for profile generation and the *mixed* class reduction scheme is used for the super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil.

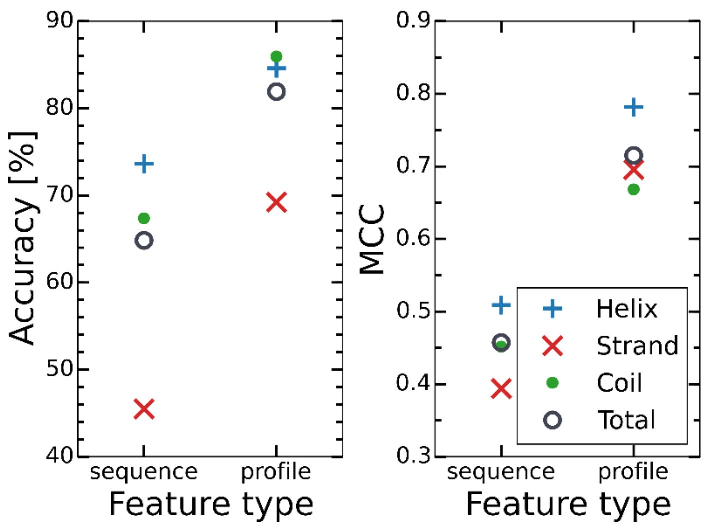

Figure 3-11: Prediction quality of the first step of *SPARROW$^+$ using quadratic sequence or profile features. For the predictions, a window size of 11 residues is used and the *mixed* class reduction scheme for the super-class definition. The profiles are generated with the *UniRef90* BLAST database. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil separately.

residue of the sliding window contains most of the information to which super-class a residue might belong to.

Since helical structures form through local interactions, small window sizes are already enough for an adequate prediction quality of the helix super-class. Strand structures, however, form through space global intra-protein interactions, which explains the greater window size necessary to accurately predict the strand super-class. The coil super-class is a very diverse class, since it encompasses a large number of the eight different DSSP classes. Some of the coil subclasses like turn or 3_{10}-helix are defined by local interactions. Hence, smaller windows are enough for an adequate prediction. Other subclasses like bend and random coil are essentially defined as structureless. Thus, they would potentially need larger window sizes for a proper description.

3.3.5 Classifier enhancement

Besides optimizing parameters like the BLAST database or the class reduction scheme, the classifier itself can also be enhanced. In the previous sections, the classifier consisted of a single prediction step using linear and quadratic profile features. To improve the prediction quality, enhancements are introduced: A second prediction step is added, which uses the secondary structure scores from the first step as input features. Also, additional linear input features in both prediction steps are introduced. In the first step, linear sequence features are added, followed by linear profile features in the second step. As previously mentioned, the window size is a critical classifier parameter concerning the demand on CPU time and memory. Accordingly, reducing the prediction quality differences between window sizes would be highly beneficial. The classifier enhancements are therefore tested on a wide range of window sizes.

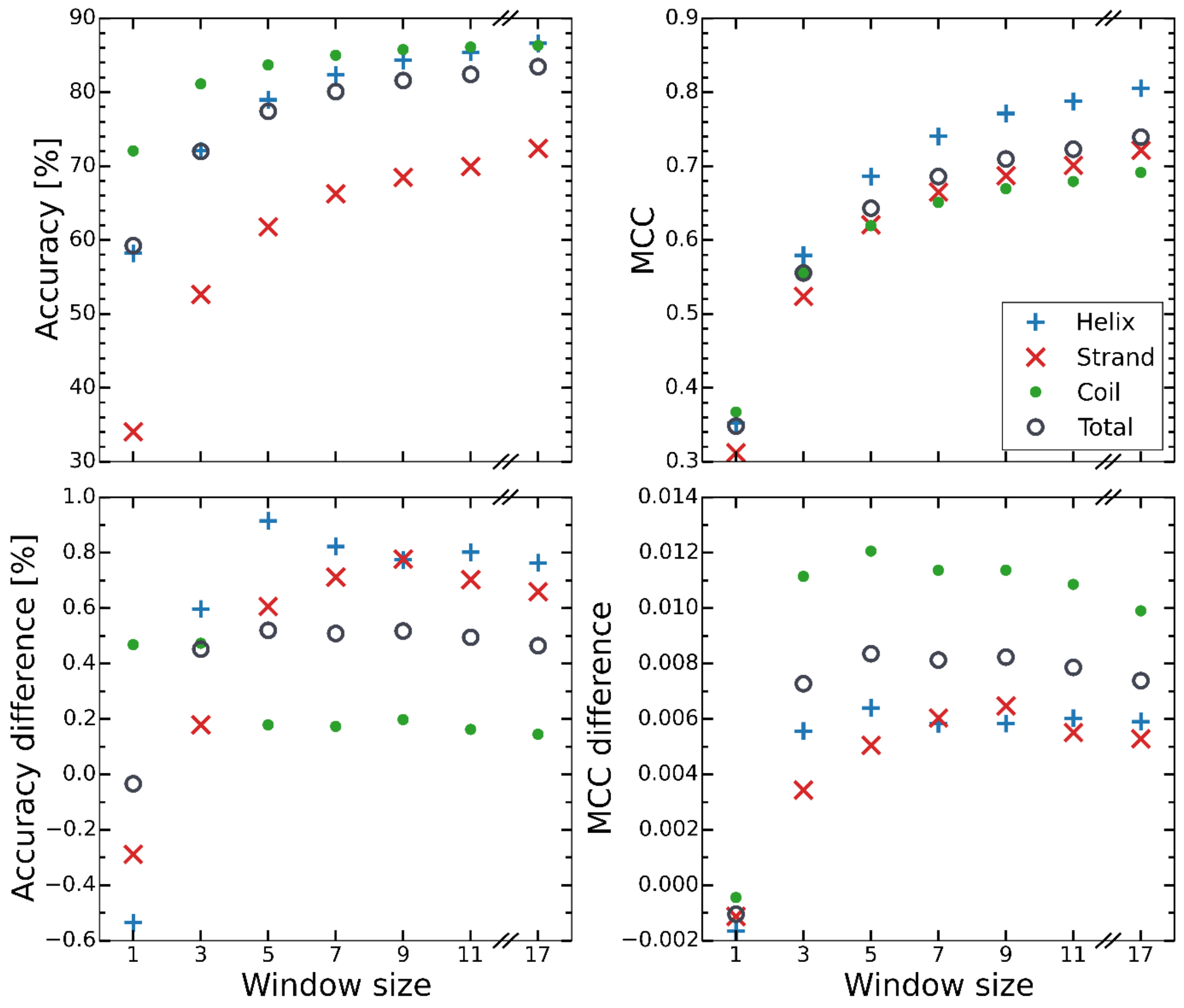

Figure 3-12: Prediction quality of the first step of *SPARROW$^+$ using additional linear sequence features as input (**top**) and the prediction quality difference to the corresponding classifier without these additional features (**bottom**). The classifier utilizes the *UniRef90* BLAST database for profile generation and the *mixed* class reduction scheme for the super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil separately.

3.3.5.1 Additional sequence feature

As described in chapter 2.3.2, profile features are generated by pairwise sequence alignments between sequentially related proteins. The information contained in each profile entry incorporates global information about the entire protein sequence not just a specific position. Accordingly, the application of profile features dramatically increases protein secondary structure prediction quality compared to using only sequence features.

Figure 3-11 shows the difference between predictions with sequence and profile features. All super-classes gain prediction quality, with the highest gains made for the strand super-class. The total difference is about 15 % for Q_3-accuracy (equation (2-78)) and 0.35 for the generalized MCC value (equation (2-81)).

Introducing sequence features therefore seems to be a paradox. However, the advantage sequence features might have over profile features is their specificity. To be meaningful, sequence profiles must be generated from related proteins. If the proteins in the sequence alignment are too different, the obtained scores can be shifted towards profile values close to zero. The situation is similar to the difference between the filtered *UniRef90* and the *non-redundant* BLAST databases described in section 3.3.2. In this situation, sequence information is advantageous, because it is specific for the respective protein.

As shown in Figure 3-12, the additional linear sequence features lead to an overall improvement in prediction quality, except for a window composed of a single residue. For all window sizes greater than one, the Q_3-accuracy gains are in the range of 0.5 %. The super-classes benefit differently from the sequence feature. Q_3-accuracy gains are similar for both helix and strand. Interestingly, the gains for strand are highest at a window size of 9 residues. At larger or smaller window sizes, the gains are lower. In contrast, for coil, the gains are very similar for all window sizes.

The difference in Q_3-accuracy and MCC value improvement for the coil super-class can again be attributed to a loss in false positive coil predictions. Improving the rate of true positive predictions of the other two classes also improved the coil class.

3.3.5.2 Second prediction step

The output of a secondary structure prediction can be used as the input for a consecutive prediction step. This allows correlations between predicted secondary structures to be taken into account. Correlating predictions from different positions in the sliding window can smoothen the prediction. If the central position in the sliding window belongs to a different super-class than the majority of the surrounding positions, it can be reclassified to fit the surroundings. Considering the predictions of neighbouring residues can also be interpreted as an indirect increase of the window size. The second prediction step is a form of post processing. Figure 3-13 shows that a subsequent prediction step has indeed a major positive effect on the prediction quality. However, the extent depends very much on the window size. The smaller the window, the larger the yield from the second prediction step, whereby the relation is non-linear. The exception to this relation is the window size of one, where similarly to adding linear sequence features, no gains are achieved at all. The overall result of the second prediction step is a reduction of the prediction quality difference between window sizes.

Considering the individual super-classes, the effects are quite different. Regarding the results of the Q_3-accuracy (equation (2-78)), at a window size of 3 the strand super-class gains more than 5 % Q_3-accuracy, while a window size of 17 leads to a slight loss in Q_3-accuracy. For the helix super-class, the effect is similar but not as pronounced with only a 3 % increase at a window size of three. Yet, for the coil super-class, the gains are low at a smaller window size, only about 2.5 % at a size of 3, yet are still positive with 1 % at a window size of 17. At window sizes of 7 and 9, the Q_3-accuracy improvements are nearly identical for all super-classes.

The improvement of prediction for the MCC value (equation (2-81)) is slightly different. Except for a window size of 1, the MCC value improves for all super-classes. The overall trend is similar to the trend of the Q_3-accuracy. At smaller window sizes the gains are higher than at larger window sizes. In contrast to the Q_3-accuracy, the relative MCC value difference between super-classes is similar at all window sizes. The super-class that improves the most is the helix super-class, followed by the strand and then the coil super-class.

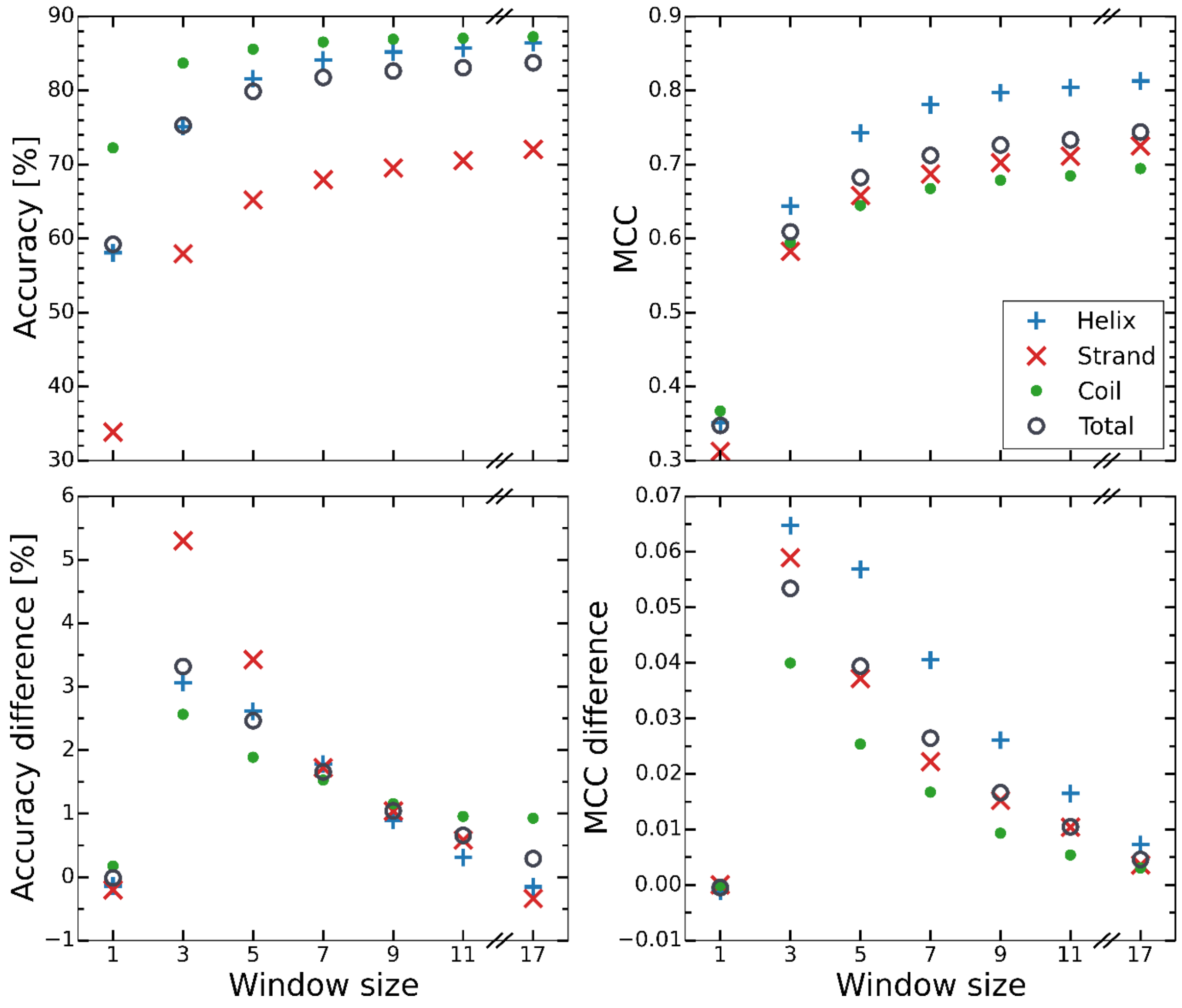

Figure 3-13: Prediction quality of the second step of *SPARROW⁺ (**top**) and the prediction quality difference to the first step (**bottom**). The second prediction step utilizes solely the output of the first step as input. For prediction, the classifier utilizes the *UniRef90* BLAST database for profile generation and the *mixed* class reduction scheme for the super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil.

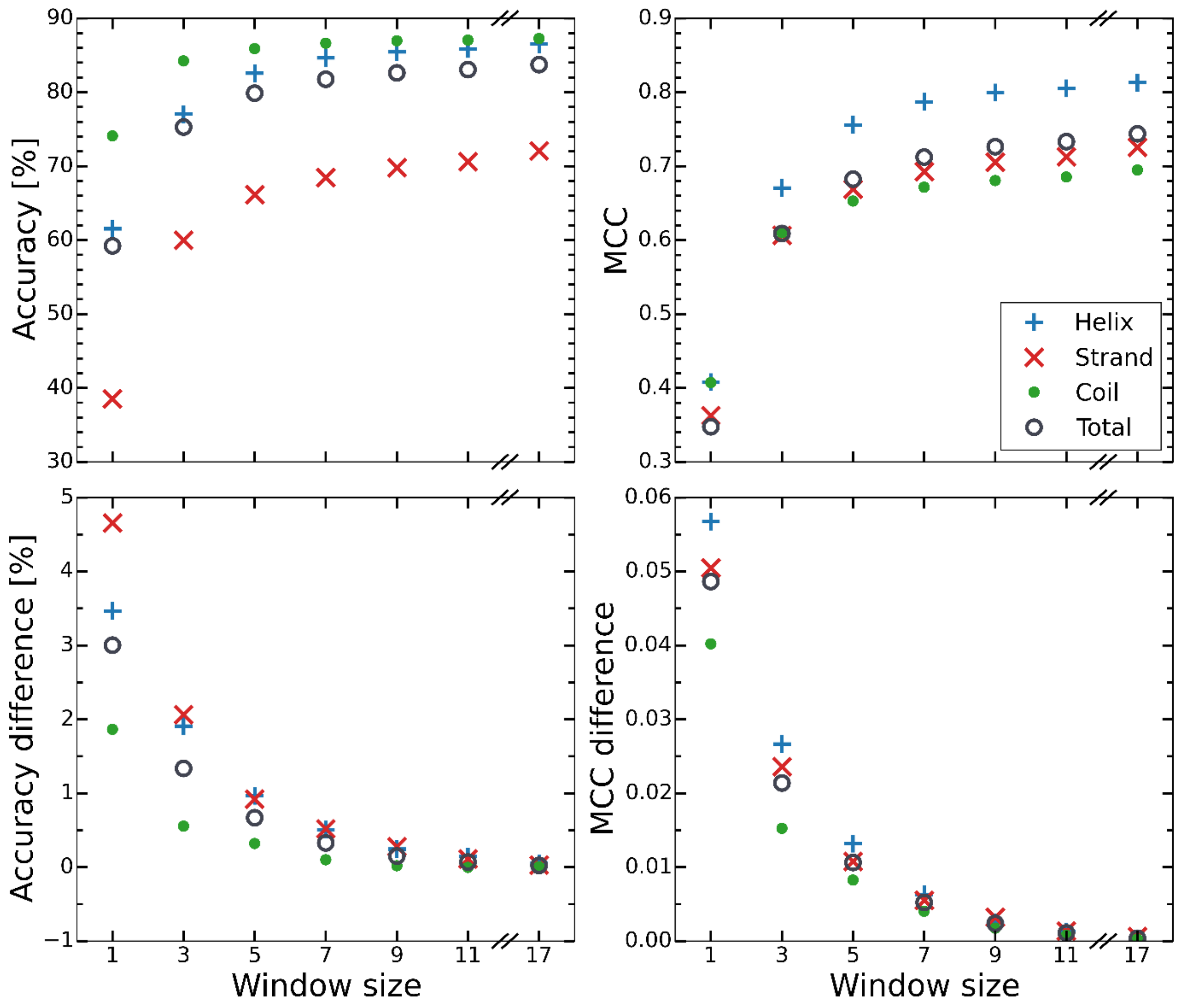

Figure 3-14: Prediction quality of the second step of *SPARROW$^+$ using additional linear PSI-BLAST profile features as input (**top**) and the prediction quality difference to the corresponding classifier without these additional features (**bottom**). For prediction, the classifier utilizes the *UniRef90* BLAST database for profile generation and the *mixed* class reduction scheme for the super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil separately.

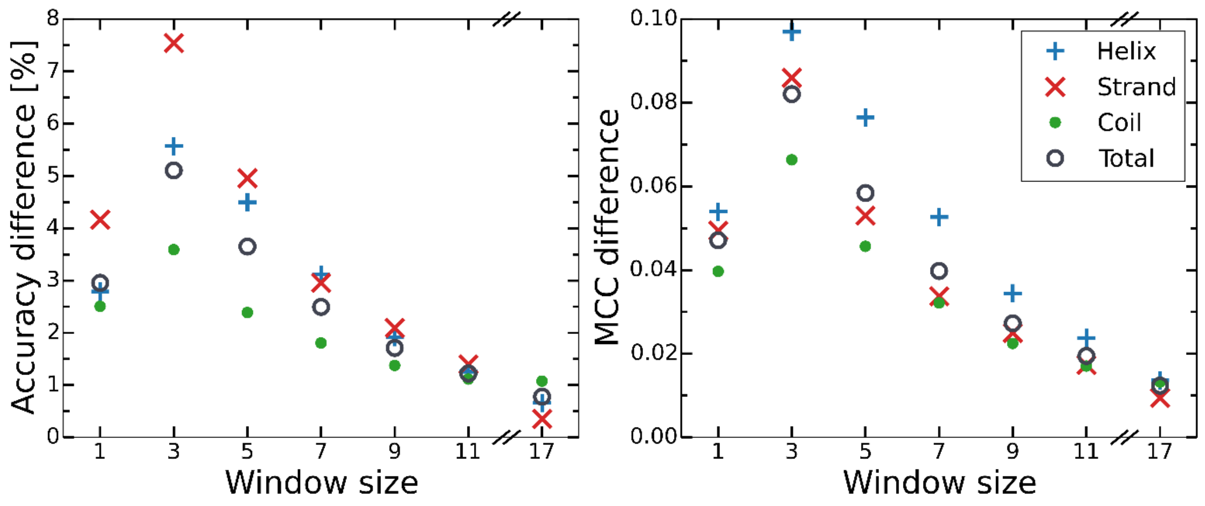

Figure 3-15: Prediction quality difference of the second step of *SPARROW$^+$ with additional sequence and profile features and first step using only PSI-BLAST profile features. For prediction, the classifier utilizes the *UniRef90* BLAST database for profile generation and the *mixed* class reduction scheme for the super-class definition. Shown is the Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) for the total data as well as for the three super-classes helix, strand and coil separately.

3.3.5.3 Additional profile feature

As in the first prediction step, additional linear features can also be added in the second prediction step. Adding sequence features in the first step enables correlations between profile and sequence features to be considered. In the second step, linear profile features are added, yielding structure to profile feature correlations. The addition of linear profile features in the second step has broadly a similar effect as adding linear sequences features in the first step. As illustrated in Figure 3-14, the prediction quality improves, but the extent depends on the window size. The larger the window size, the smaller the improvement. More specifically, at a window size of 17 adding linear profile features no longer improves prediction quality. However, there is one major distinction between the first and second prediction step. The addition of linear profile features in the second step also improves the prediction quality for window sizes of one.

3.3.5.4 Overall impact of prediction enhancements

Figure 3-15 shows the prediction quality difference between the second prediction step of *SPARROW$^+$ with additional input features and the first prediction step without those. A general trend that all enhancements have in common is that the gains in prediction quality decrease with larger window sizes. At a window size of 3 the Q_3-accuracy (equation (2-78))

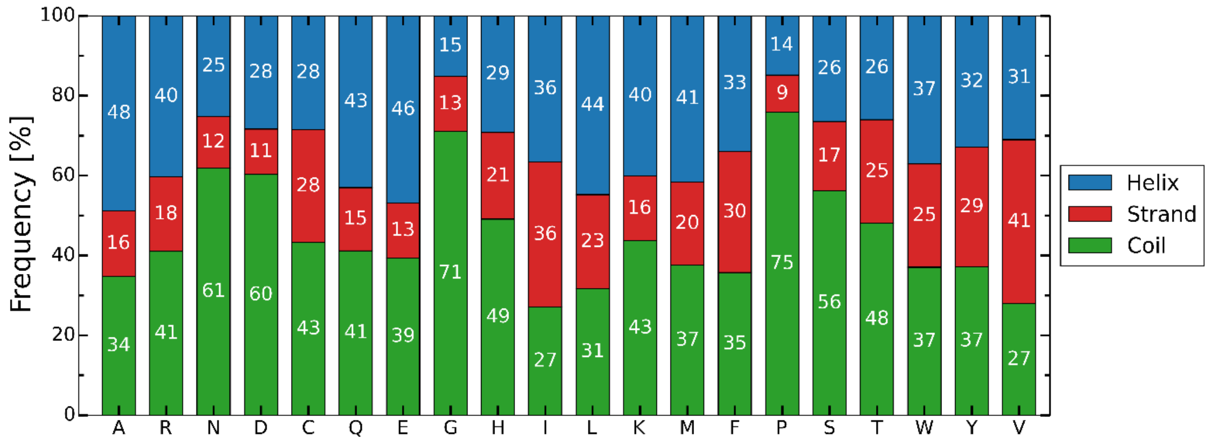

Figure 3-16: Super-class frequencies per amino acid type in the processed Astral40 (Berman et al., 2003; Brenner et al., 2000; Chandonia et al., 2002; Chandonia et al., 2004) dataset using the *mixed* class reduction scheme.

improves by about 5 %, while at a window size of 17 the improvements are less than 1 %. For the MCC value (equation (2-81)) the trend is similar. Hence, enhancing the classifier has less impact on the prediction quality for larger window sizes.

The trend by which the prediction quality improves is similar for all super-classes. Considering the MCC value, the helix super-class always improves most from classifier enhancements, followed in the majority of cases by the strand super-class. At a window size of 11, the improvements for coil and strand are identical and at a window size of 17, coil even improves more. Apart from windows of one or three residues, the MCC value difference is larger between the helix and strand super-classes than between the strand and coil super-classes. However, when considering the Q_3-accuracy, the strand super-class improves the most, followed by the helix and lastly the coil super-classes. Similar to the MCC value, the trend of Q_3-accuracy improvement changes at a window size of 11 residues. The improvement is nearly identical for all super-classes at this window size. At larger window sizes of up to 17 residues, the trends observed for smaller windows are reversed.

3.4 Validation

After assembling the final setup of the vector-valued classifier, as described in the previous part, the performance of *SPARROW$^+$ can now be analysed. Two aspects are of interest, the prediction quality for individual amino acids and the reliability of the confidence measure defined in chapter 2.6.3.2. Amino acids have distinctly different properties, which result in

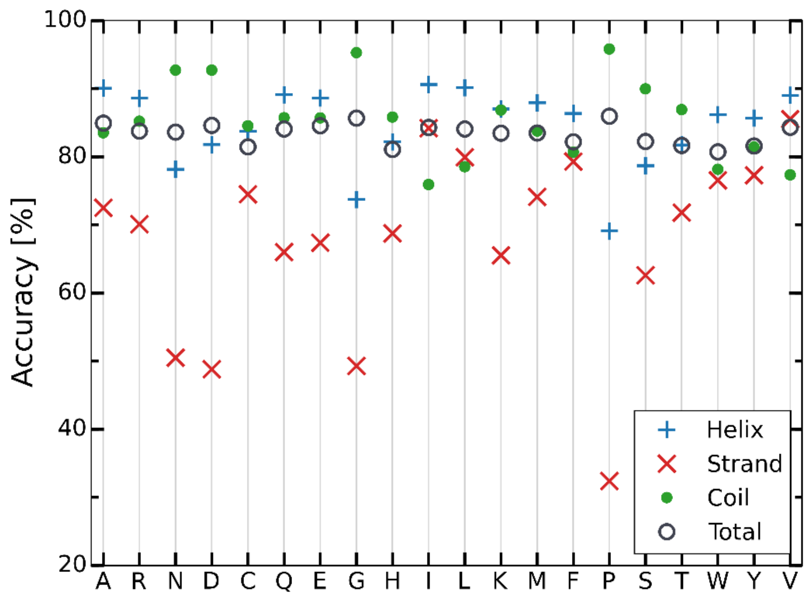

Figure 3-17: Prediction quality for each of the 20 standard amino acids with *SPARROW⁺. For the predictions, a window size of 17 residues and the *mixed* class reduction scheme were used. Shown is the Q_3-accuracy (equation (2-78)) for the total dataset and for the three super-classes.

prediction quality variations between the amino acids. The confidence measure is an internal reliability measure. Correlating it with the Q_3-accuracy makes it also an external measure.

3.4.1 Amino acid preferences

Amino acids have different properties and biological functions defined by their chemical structure, described in chapter 1.1.1. These differences, as discussed in chapter 3.3.2, induce the different shapes of the BLAST profile value distributions. In addition, the occurrence of amino acids is not uniform for all protein secondary structure classes. Thus, in certain classes, some amino acids are predominantly found. In fact, most amino acids have a preference to occur in a specific super-class, as Figure 3-16 illustrates. Asparagine (N), aspartate (D), glycine (G) and proline (P) frequently occur in coil, while isoleucine (I) and valine (V) prefer strand. The helix super-class is preferred by alanine (A), glutamic acid (E) and leucine (L). These results agree with literature findings described in section 1.1.1. Amino acids with a large side chain, like isoleucine (I), phenylalanine (F), tryptophan (W), and tyrosine (Y) prefer the strand class for steric reasons. Similarly, amino acids with a small side chain, like alanine (A), glutamine (Q) and glutamate (E) prefer the helix class. Glycine (G) and proline (P) are often found at the boundaries of different secondary structures. G because of its conformational freedom and P for its lack of it.

The variations in the profile value distributions and the frequency per super-class also influence the prediction quality, as shown in Figure 3-17. The total prediction Q_3-accuracy (equation (2-78)) varies between the amino acid types. Histidine (H) and cysteine (C) have the lowest and alanine (A) and glycine (G) the highest Q_3-accuracy. The amino acid specific differences in Q_3-accuracies are even more pronounced for the individual super-classes. The Q_3-accuracy in strand is low for proline (P), whereas it is high for phenylalanine (F).

It is generally difficult to predict strand accurately, which is in agreement with the findings shown before. Strand is the smallest super-class and the prediction quality is particularly bad for amino acids rarely found in strand. This is further compounded if the amino acid occurs infrequently also in the other super-classes. Proline is an example for this case, as it is found rarely, particularly in strand. Although, not as rare as proline: N, D and G are also rarely found in strand, thus displaying a low predication quality for strand.

3.4.2 Confidence measurements

In chapter 2.6.3 a confidence measure was defined to estimate the reliability of the classifications made by *SPARROW$^+$. The confidence is based on a probability measure to approximate the probability that a residue belongs to a certain super-class. By definition, the probability is always highest for the predicted super-class. In the case of three super-classes, the minimal probability for the predicted class must be greater than 1/3. The confidence is the normalization of the interval 1/3-1 to 0-1. By this definition, the confidence can be considered as an internal measure of the prediction reliability. As such, it should correlate with actual prediction quality measures, Q_3-accuracy (equation (2-78)) and MCC value (equation (2-81)). Plotting the prediction quality as a function of the confidence, depicted in Figure 3-18, shows that such a correlation exists. The prediction quality is proportional to the confidence level. The standard deviation of the Q_3-accuracy and generalized MCC value is highest at a low confidence level, becoming successively smaller with increasing confidence. The highest standard deviation is 5 % for the Q_3-accuracy and 0.05 for the generalized MCC value.

Apart from the correlation between prediction quality and confidence, the distribution of the probability measure is of interest. In particular the differences between correct (true positive and true negative) and incorrect (false positive and false negative) classifications. As Figure 3-19 shows, the distributions for correct and incorrect classifications are distinctly different. The probability distributions for correct classifications have a low spread and the

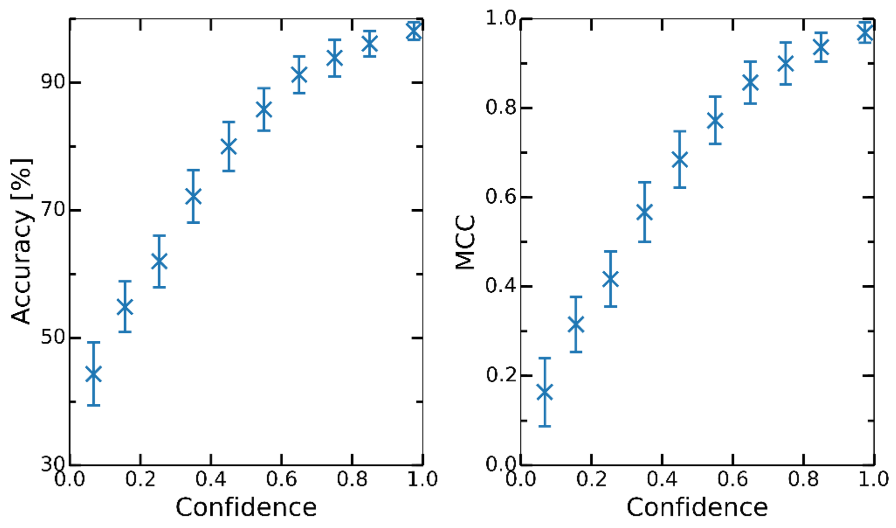

Figure 3-18: Prediction quality as a function of confidence (equation (2-88)). The Q_3-accuracy (equation (2-78)) and generalized MCC value (equation (2-81)) are calculated from 100 randomly chosen samples for each confidence interval. The mean and standard deviation are obtained by repeating the calculations 50 times. For the predictions, the final two-step classifier of *SPARROW$^+$ with additional sequence and profile features is utilized. The *mixed* class reduction scheme for the super-class definition and a sliding window of 17 residues is used.

maxima are at the extreme values of 0 and 1. Interestingly, for true positive classifications, the probability distribution is nearly uniform in a probability range from 0.55 to 0.95. In contrast, the distributions for the incorrect classifications have a larger spread and the maxima are at mid-range probability values, of 0.35 and 0.5. The probability distribution is similar for all super-classes, both for correct and incorrect classifications. Only the strand super-class has a slightly higher frequency at mid-range probability levels, for both true and false positive classifications.

The wide spread and the position of the maxima of distributions for the incorrect classifications suggests, that the projections of a score vector on the different class vectors of each super-class all have a similar length. This can only occur, if the scores obtained from the prediction are all close to zero.

Overall, the probability measure developed for the vector-valued classifier is a good measure to assess prediction reliability. The low population of incorrect classifications at high probability levels further reinforces the reliability of the vector-valued classifier and the confidence measure. Incorrect and correct classifications have major distribution differences,

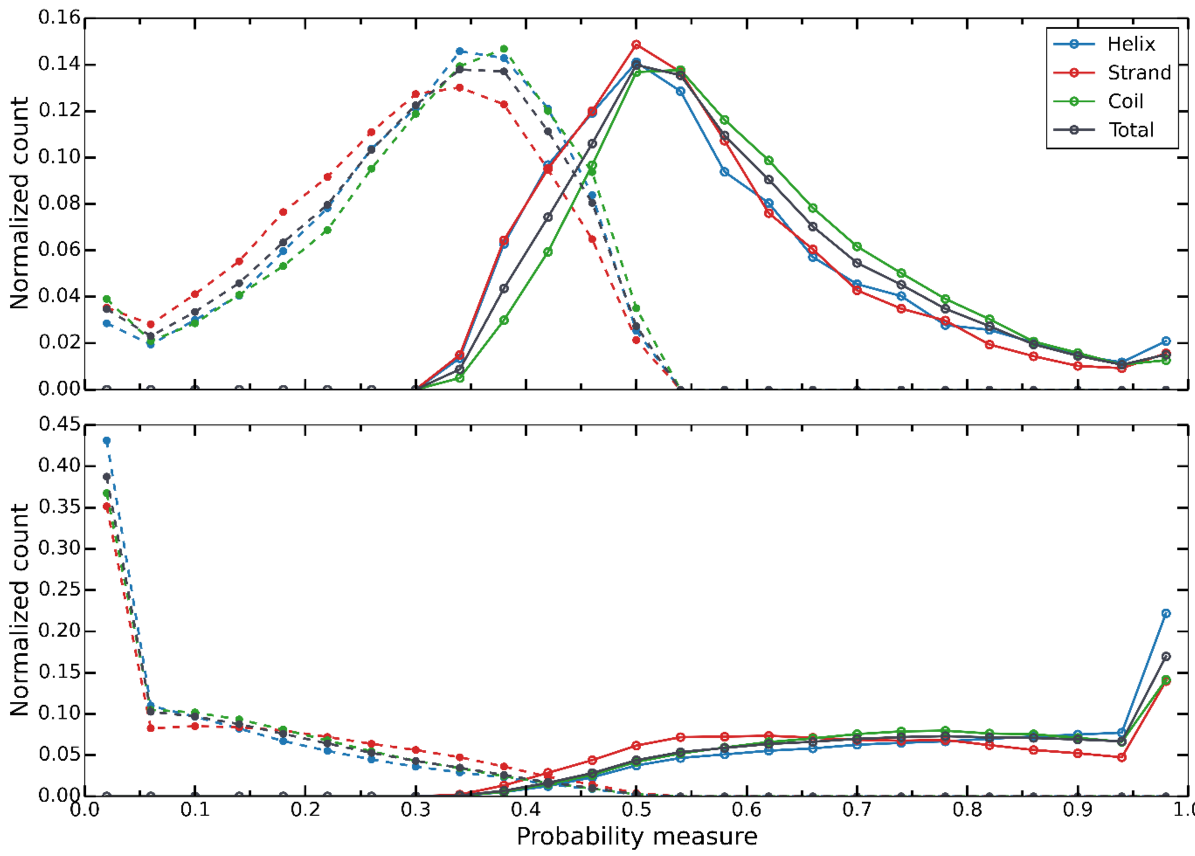

Figure 3-19: Histograms of the probability measure (equation (2-84)) of positive (solid line) and negative (dashed line) classifications. Shown are the distributions for false positive and negative classifications (**top**) and for true positive and negative classifications (**bottom**). The total and super-class specific distributions are displayed. The histogram consists of 25 bins covering a probability measure interval from 0 to 1. The bin counts were normalized to 1 for the positive and negative plots separately. For the predictions, the final two-step classifier of *SPARROW$^+$ with additional sequence and profile features is utilized. The *mixed* class reduction scheme for the super-class definition and a sliding window of 17 residues is used.

45 % of the correct classifications have a confidence above 0.8, while only 9 % of the incorrect classifications are above this confidence level.

3.4.2.1 Correlation between confidence and Q_3-accuracy

As mentioned in the previous section, correlating the confidence with the prediction quality measures would allow the validity of the prediction to be estimated even for an individual residue. Hence, using a polynomial function of the fourth order, the Q_3-accuracy was fitted to the confidence. For the fitting, the 8th subset of the filtered ASTRAL40 dataset was used. The Q_3-accuracy cannot be calculated from a single prediction, but only from a set of predictions. Therefore, the confidence range from 0 to 1 was separated into smaller confidence intervals.

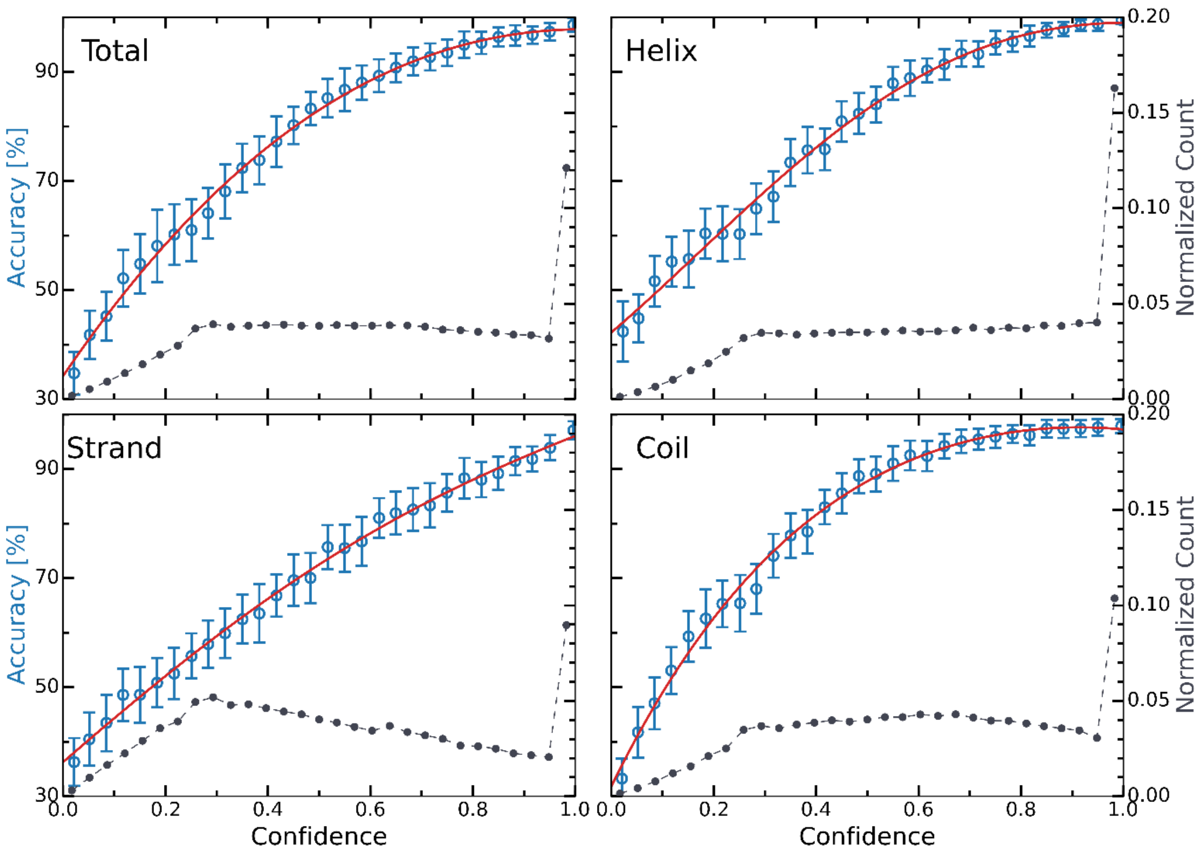

Figure 3-20: Correlation between confidence (equation (2-88)) and the Q_3-accuracy (equation (2-78)) for the total data and the individual super-classes. Each plot shows a confidence histogram (**black**), the sampled Q3-accuracy (**blue**) and the fitted fourth order polynomial (**red**). The confidence histograms consist of 30 bins and are normalized to 1. The Q_3-accuracy is calculated from 100 randomly chosen samples within a confidence interval. The mean value and standard deviation are obtained by repeating the Q_3-accuracy calculations 50 times. Shown is Q_3-accuracy of the second step of *SPARROW$^+$ using linear sequence features as additional inputs in the first step and linear profile features in the second. The *mixed* class reduction scheme for the super-class definition and a sliding window of 17 residues is used.

For each interval, predictions were performed for a set of 100 randomly chosen residues and the corresponding confidence values were monitored. The quality measures were calculated from these random samples. This procedure was repeated 50 times for each confidence interval to obtain the mean value and standard deviation. In accordance with Figure 3-18, Figure 3-20 shows that the Q_3-accuracy strongly correlates with the confidence measure. However, the specific shape of the plot differs per super-class. The plots for the helix and coil super-classes have a steep slope, similar to a logarithmic function. In contrast, the plot for the strand super-class has a shallow slope and suggests a linear correlation between confidence and Q_3-accuracy (equation (2-78)). At a confidence of 1.0, an Q_3-accuracy of 95 % is achieved for the helix and strand super-classes, whereas only 93 % is reached for the coil. The confidence distribution also varies depending on the super-class. In agreement with the previous findings, the Q_3-accuracy variation is in all cases larger at low confidence levels. Interestingly, for the helix and coil super-classes, the Q_3-accuracy plot has a kink at a confidence level around 0.25. Furthermore, for all super-classes, the distributions are similar to the true positive distribution shown in Figure 3-19. Hence, they all share a maximum at a confidence of 1.0. However, the maximum is more pronounced for the helix super-class than for the strand super-class. For both, the helix and coil super-classes the distribution is uniform from a confidence of 0.3 onwards, whereas for the strand super-class it declines beyond this value. Accordingly, the distribution of the strand super-class has similarities to the distribution of the false positive, which also displays a decrease at a higher confidence.

Figure 3-20 shows that the Q_3-accuracy can be fitted to the confidence. Q_3-accuracy and confidence are, apart from small variations, in good agreement. At a confidence above 0.7, classifications as helix or coil are to 90 % correct. The differences in the Q_3-accuracy plots between super-classes, suggest using super-class specific fitting functions instead of a general one.

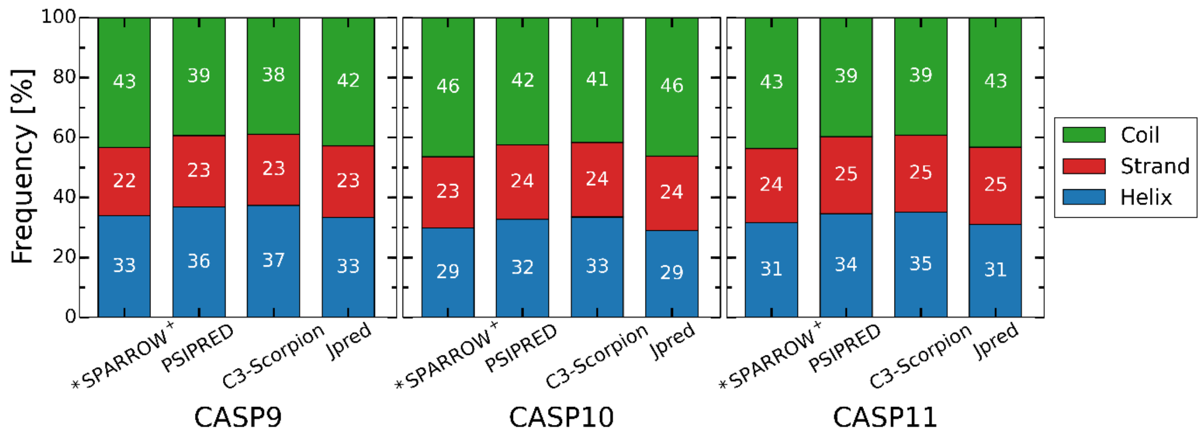

Figure 3-21: Super-class composition of the CASP9, 10 and 11 datasets with different class reduction schemes. Protein secondary structures in the datasets were assigned by DSSP (Kabsch and Sander, 1983) into one of eight classes. Subsequently, the eight classes were reduced to three super-classes helix, strand and coil using the class reduction schemes *mixed*, *psipred*, *scorpion* and *Jpred*. For a detailed description of the class reduction schemes, see Table 2-1 in chapter 2.2.

3.5 Benchmark

To assess the prediction quality of *SPARROW$^+$, it is benchmarked against other popular secondary structure prediction methods: C3-Scorpion (Yaseen and Li, 2014), Jpred (Drozdetskiy et al., 2015) and PSIPRED (Buchan et al., 2013; Jones, 1999) as well as against its predecessor *SPARROW (Rasinski, 2011). Chapter 1.2.1 gives an overview about all methods apart from *SPARROW. For the benchmark, the newest downloadable versions of C3-Scorpion, PSIPRED and *SPARROW were utilized (April 2015). Since no download version is available for Jpred, the web interface was used. For each downloaded program, the necessary BLAST database was generated according to the specifications provided in the documentation or corresponding publication. Hence, for C3-Scorpion the *non-redundant* (downloaded 19.3.2015) and for PSIPRED and *SPARROW the *UniRef90* (downloaded 11.3.2015) database was used for PISIBLAST profile generation. Both BLAST databases were filtered of transmembrane and unordered proteins with the program *pfilt* (Jones and Swindells, 2002). In their respective training processes, all three programs used different class reduction schemes to reduce the eight DSSP (Kabsch and Sander, 1983) classes to the three super-classes helix, strand and coil. The different class reduction schemes are described in Table 2-1 in chapter 2.2.

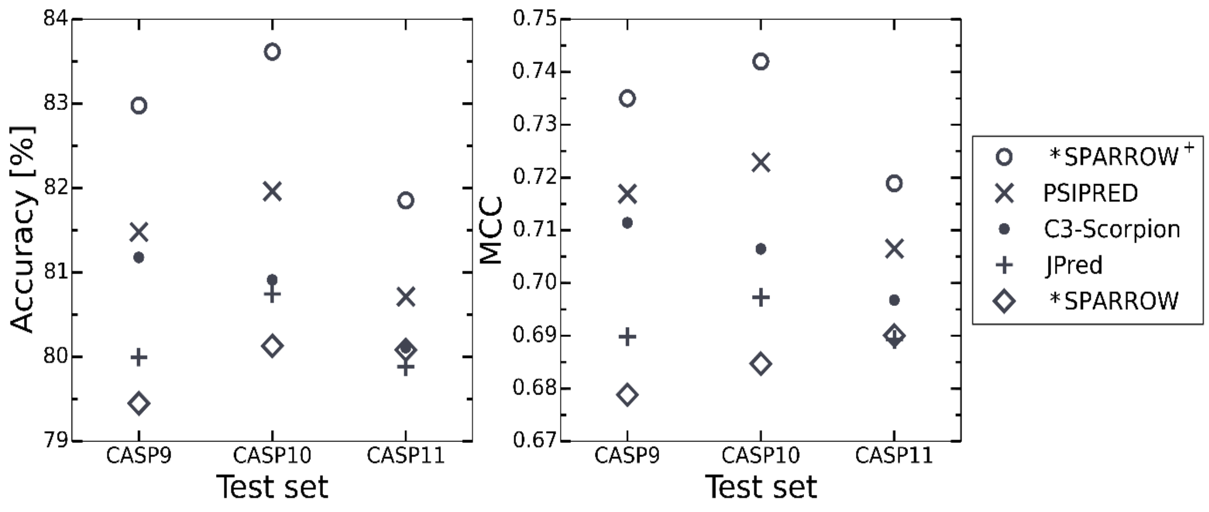

Figure 3-22: Prediction quality of different protein secondary structure prediction programs for the CASP9, 10 and 11 datasets. Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) are used as quality measures.

To calculate the prediction quality for each program, the predictions were compared with reference data sets of secondary structures. The secondary structures were determined by DSSP, whereby the eight DSSP classes were reduced to the three super-classes, according to the class reduction scheme defined for each program. For the benchmark, *SPARROW$^+$ uses a two-step prediction with additional sequence features in step one and additional profile features in step two. The sliding window size is set to 17 residues and the *mixed* class reduction scheme is used. For profile generation the *UniRef90* database was used. The benchmark consists of four datasets, which are three CASP datasets and an ASTRAL40 subset.

3.5.1 CASP datasets

CASP is the *Critical Assessment of protein Structure Prediction*, a regular contest to assess the prediction quality of protein structure prediction programs, in particular 3D structure prediction. The CASP contest allows reliable measures for a numerical assessment of predicted protein structure models to be established. The structures used in the CASP datasets were previously not available to the public, and were released only after the corresponding contest has ended. Therefore, the datasets can be used for fully blind testing (Moult et al., 2014). For this benchmark the CASP datasets from the years 2010 (CASP9), 2012 (CASP10) and 2014(CASP11) were used (Kryshtafovych et al., 2011; Kryshtafovych et al., 2014). The ASTRAL40 dataset (version 2.03) used to train *SPARROW$^+$ is older than the CASP11 dataset.

Hence, there is no overlap of structures between both sets. The number of residues is similar for all three CASP datasets. CASP9 contains 23,654, CASP10 22,627 and CASP11 19,200 residues.

All CASP datasets have a very similar super-class composition as illustrated in Figure 3-21. The coil super-class is the largest class making up 40 %. Second is the helix super-class with more than 30 % and finally, the strand super-class makes up around 20 %. For the *psipred* and *scorpion* class reduction schemes the helix super-class is around 3 % larger, because the 3_{10}-helix class belongs to the helix super-class, whereas in the *mixed* and *Jpred* schemes this class belongs to the coil super-class. The strand super-class is about 1 % larger for the *psipred*, *scorpion* and *Jpred* class reduction schemes, which accounts for the isolated β-bridge class included in the strand super-class. Accordingly, the coil super-class is largest for the *mixed* class reduction scheme, with a difference of 1 to 3 % to the other schemes. The composition of the secondary structure classes of the considered CASP datasets agrees with the ASTRAL40 dataset, see chapter 3.2.

The trend in prediction quality is similar for all CASP datasets as Figure 3-22 illustrates. *SPARROW$^+$ outperforms every other method in all datasets. The difference to the second best method in all datasets, PSIPRED, is 1 to 1.5 % Q_3-accuracy (equation (2-78)) and 0.01 to 0.02 in the value of the generalized MCC (equation (2-81)). The overall trend is also similar for all datasets, the quality differences are always in the same relative range. C3-Scorpion achieves a much higher prediction quality compared to Jpred in the CASP9 dataset, but not in the CASP10 or 11 datasets. The prediction quality of C3-Scorpion declines over the datasets from more than 81 % Q_3-accuracy to just above 80 %. With the exception ofC3-Scorpion, all programs perform best in the CASP10 dataset. In contrast to C3-Scorpion, the prediction quality of Jpred is quite similar in the CASP9 and 10 datasets. Yet, the quality is always lower compared to C3-Scorpion. The CASP11 dataset is apparently the most difficult dataset, since apart from *SPARROW, all other methods have the worst performance for this dataset. This suggests, that the proteins contained in this dataset have a low sequence similarity to proteins with known structures, making the generation of meaningful profiles and hence predictions difficult. *SPARROW has the worst prediction quality of all methods, the only exception being in the CASP11 dataset where the quality is slightly higher than that of Jpred.

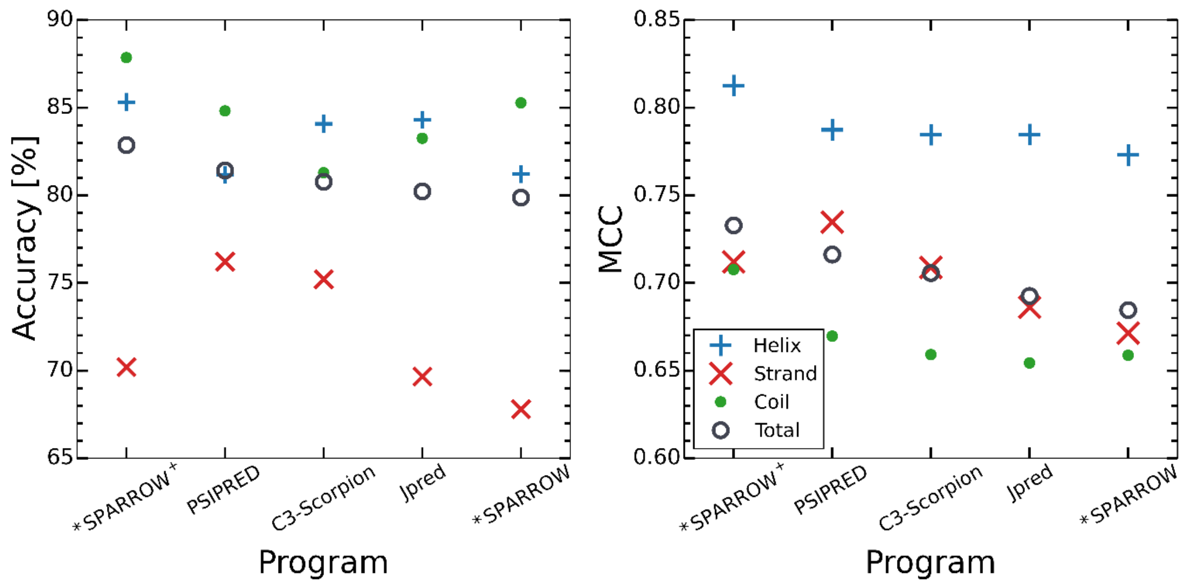

Figure 3-23: Prediction quality per super-class of different protein secondary structure prediction programs for a combined dataset consisting of CASP9, 10 and 11. Q_3-accuracy (equation (2-78)) and the MCC value (equation (2-81)) are used as quality measures.

To analyse the prediction quality per super-class, all CASP datasets were combined into one dataset, see Figure 3-23. Apart from the C3-Scorpion, the relative prediction quality between the programs is similar for all CASP datasets, only the absolute values differ. The combination increases the sample size making statistics more viable. As expected, the total prediction quality corresponds to the mean of the three CASP datasets as Figure 3-22 illustrates. *SPARROW$^+$ has the highest Q_3-accuracy with nearly 83 %, followed by PSIPRED with 81.5 %. C3-Scorpion reaches a Q_3-accuracy of 81 % and Jpred slightly more than 80 %. The Q_3-accuracy of *SPARROW is just below 80 %. For the MCC value the order is identical, with differences between the programs being around 0.01. *SPARROW$^+$ achieves an MCC value of more than 0.73, Jpred of 0.69 and *SPARROW of only 0.68.

While the total prediction quality varies only slightly between the programs, the prediction quality for the individual super-classes is in parts dramatically different. Yet, all programs show also a few common trends. For all considered prediction tools, the coil super-class has the lowest MCC value while still achieving a high Q_3-accuracy. For *SPARROW$^+$ and PSIPRED the Q_3-accuracy is actually the highest of all super-classes. In contrast, the helix super-class achieved both a high Q_3-accuracy and MCC value. The strand super-class has the lowest Q_3-accuracy but still a moderate MCC value. Apart from these similarities between the four programs, there are also major differences.

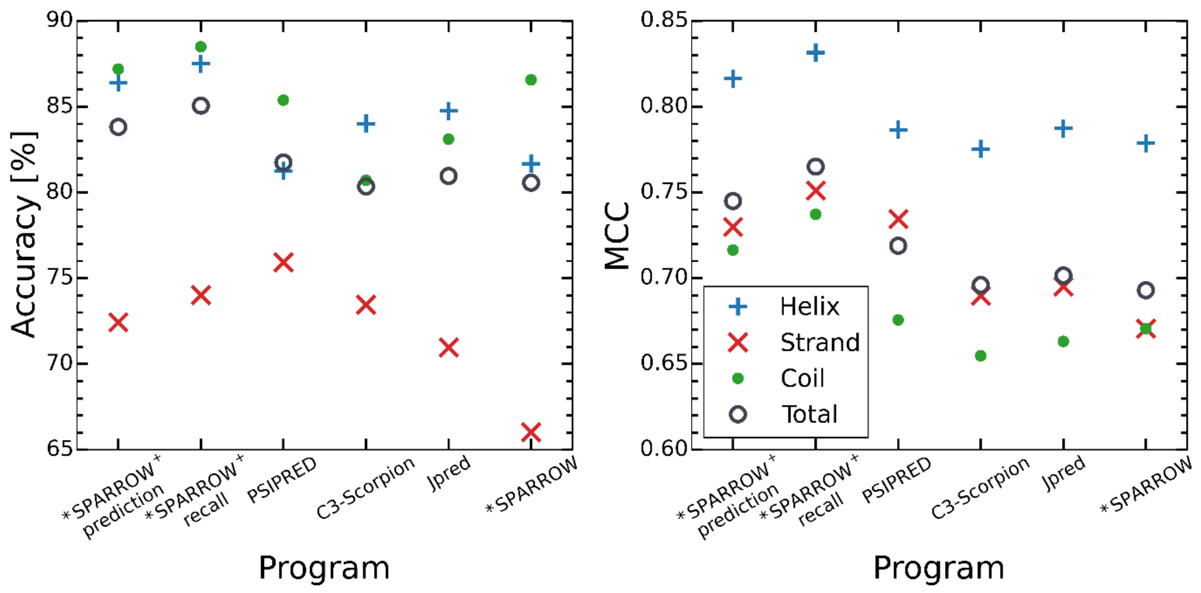

Figure 3-24: Prediction quality of different protein secondary structure prediction programs for an Astral40 subset. For *SPARROW+ the subset is either included in the training process (recall) or not (prediction). Q-accuracy (equation (2-78)) and the MCC value (equation (2-81)) are used as quality measures.

*SPARROW+ achieves the highest prediction quality of all programs for both the helix and coil super-classes. The high quality of the helix class shows that it is well defined with minimal overlap. However, the coil super-class has a great discrepancy between the Q_3-accuracy and MCC value, which suggests that the class overlaps with another, most likely the strand super-class. While the MCC values for the coil and strand super-classes are similar in *SPARROW, the Q_3-accuracy differs considerably. For the coil super-class, a Q_3-accuracy of 88 % is achieved, whereas for the strand super-class only 70 % are reached. Hence in *SPARROW+, the coil and strand super-classes overlap and the coil is overestimated at the cost of the strand super-class. PSIPRED reaches the highest prediction quality for the strand super-class and the lowest Q_3-accuracy for the helix super-class compared to the other programs. However, the MCC value for the helix super-class in PSIPRED is comparable to C3-Scorpion and Jpred. Having on one hand, the discrepancy of the helix Q_3-accuracy and MCC value, and on the other, the high Q_3-accuracy yet low MCC value for the coil class, suggests an overestimation of the coil class at the expense of the helix class for PSIPRED. The performance of C3-Scorpion is similar to that of PSIPRED. The major difference is that the helix and coil super-class accuracies are reversed. Yet, this reversal is not reflected in the MCC value. C3-Scorpion has the worst Q_3-accuracy for the coil class. Jpred achieves a high prediction quality for the helix and a high Q_3-accuracy for the coil super-class. However, Jpred also has the worst prediction quality for the strand super-class. This implies a major overlap between strand and coil, because of an overestimation of the coil class.

The MCC value for the helix and coil super-classes is very similar between PSIPRED, C3-Scorpion and Jpred. The major difference is the MCC value. Interestingly, the class reduction schemes of each program, as described in Table 2-1, are identical concerning the strand super-class. All programs include β-strand and the isolated β-bridge class into the strand super-class. The difference in MCC value could be caused by different weighting of the classes during the training process.

3.5.2 ASTRAL40 2.03 subset

The combined CASP dataset only contains 65,481 residues and may be a selection of more unusual proteins. Therefore, the methods PSIPRED, C3-Scorpion and Jpred are also tested on a subset of the ASTRAL40 dataset used in the cross validation of *SPARROW$^+$. The 8th subset, which contains 256,267 residues, is arbitrarily chosen. It can be assumed, that other methods have at least to some degree, used parts of the ASTRAL40 dataset version 2.03 in their learning process. For example, Jpred is trained on the ASTRAL40 dataset version 2.04, which has overlaps with the 2.03 version. Hence, for these methods, a benchmark based on this subset is not only prediction but also recall. Also, *SPARROW was trained on the ASTRAL40 dataset version 1.71. The difference in quality for recall (trained data) and true prediction is demonstrated showing recall and prediction data for *SPARROW$^+$.

Comparing Figure 3-24 and Figure 3-23 shows that the overall performance of the different programs is very similar for the ASTRAL40 subset and the combined CASP dataset. *SPARROW$^+$ has the highest prediction quality, followed by PSIPRED. Interestingly, Jpred achieves a higher prediction quality in this benchmark than C3-Scorpion. The prediction quality for the former is actually worse than in the combined CASP benchmark. In contrast, all other programs show an improvement of prediction quality. That Jpred was trained on the ASTRAL40 dataset version 2.04, can explain the advantage of Jpred over C3-Scorpion. The Astal40 subset is more recall than prediction for Jpred. The prediction quality variation between the ASTRAL40 subset and the combined CASP dataset of PSIPRED, Jpred and *SPARROW is in the range of the difference between the *SPARROW$^+$ recall and prediction data.

4 Discussion

Knowledge about the structure of a protein is critical for understanding its function. New sequencing methods have widened the gap between the number of known sequences and known structures. Computational methods for protein structure prediction are therefore in high demand. The secondary structure of a protein describes its local order and local spatial arrangement. The prediction of protein secondary structure can be an intermediate step to predict the complete spatial geometry of a protein.

Protein secondary structure prediction is a multi-class classification problem. Although eight secondary structure classes are defined by the widely popular program DSSP (Kabsch and Sander, 1983), usually only three secondary structure classes are discriminated. These are α-helix, β-strand and coil. Chapter 1.1.2 gives an overview over the secondary structure classes and chapter 1.1.3 about the assignment.

A variety of methods have been developed for protein secondary structure prediction, some of the currently most popular ones are described in chapter 1.2.1. All these methods have in common that they utilize an artificial neural network (ANN) for prediction. In chapter 1.3.2 the properties und modes of operation of ANNs are explained. In this thesis an alternative method for secondary structure prediction was developed, the program *SPARROW$^+$, which is a general overhaul of the program *SPARROW (Rasinski, 2011). Like its predecessor, *SPARROW$^+$ uses a vector-valued classifier (VVC) for prediction. The VVC consists of two consecutive prediction steps, where the input of the second step is the output of the first. In each step a set of linear regression functions is used whose number (C-1) is one less than the number of secondary structure classes C. Each class is defined by a class vector in a (C-1)-dimensional space. The class vectors point to the corners of a regular *simplex* in the (C-1)-dimensional space. Chapter 2.4.2.1 describes the class vectors, chapter 2.4.3 a general VVC and chapter 2.5 the concrete implementation of the VVC into *SPARROW$^+$.

In the results section it was analyzed, how the choice of the learning conditions, also called hyperparameters, affect prediction quality and how each enhancement of a single-step classifier improves prediction quality. The most important hyperparameters are given by the type of the BLAST (Altschul et al., 1990) database and the class reduction scheme. The former is used to generate profile features with the program PSI-BLAST (Altschul et al., 1997) and the latter determines, which of the eight secondary structure classes defined by DSSP (Kabsch and

Sander, 1983) are merged together to obtain only three super-classes. The results for different hyperparameters are described in chapter 3.3.2 and chapter 3.3.3 respectively. The results of the stepwise enhancement of single-step classifier using only profile features to a two-step classifier utilizing combinations of different feature types in each step is depicted in chapter 3.3.5. Compared with other state-of-art secondary structure prediction methods, *SPARROW$^+$ has a superior prediction quality in test sets. Described in chapter 3.5, *SPARROW$^+$ surpasses the so far best available method PSIPRED (Jones, 1999) generally by about 1.5 % in the Q_3-accuracy (equation (2-78)) and 0.02 in the MCC (equation (2-81)).
Finally, to assess the prediction quality of the VVC a confidence measure was developed. Correlating the confidence with the prediction quality shows, that both are proportional to each other. Hence, the confidence allows to estimate the prediction quality, as shown in chapter 3.4.2. In the following, some of the previously mentioned aspects of *SPARROW$^+$ are discussed in more detail.

4.1 Super-class distinction

To assess the prediction quality, two quality measures are used, the Q_3-accuracy (equation (2-78)) and the MCC (equation (2-81)), which both are defined in chapter 2.6. To evaluate the prediction quality of a super-class, both measures have to be taken into account. For more than two classes, the class specific Q_3-accuracy only considers the true positive classifications, thus the false positives are not included. Accordingly, a high Q_3-accuracy does not necessarily imply a good classification, because a class could be overestimated at the expense of the other classes. However, the Q_3-accuracy is still a measure easy to grasp. In contrast, the MCC considers both true and false positive classifications, which makes it a more robust measure but also more difficult to interpret. In combination, the Q_3-accuracy and MCC allow to analyze the prediction quality of a super-class.
*SPARROW$^+$ differentiates three secondary structure super-classes, namely helix, strand and coil. The Q_3-accuracy and MCC are quite different for all classes. Considering the individual super-classes, the coil super-class shows conspicuous results. In all figures, a high Q_3-accuracy is achieved for the coil super-class, which can reach about 90 %. Yet, at the same time the MCC is much worse, being on par with the strand super-class that achieves a much lower Q_3-accuracy.

Hence, the coil super-class is overestimated at the expense of the strand super-class. This can be explained by the different weighting of the super-classes, which naturally given by the sizes of the classes in the training sets. The weight of a super-class indirectly determines the number of parameters used for the prediction of this class. The greater the weight, the larger the number of parameters. As such, the weight is a measure of the relevance or importance of the different super-classes. In the current version of *SPARROW$^+$, the weight of super-class is determined by its size in the training set. Hence, the larger coil super-class has a greater weight than the smaller strand super-class. Therefore, *SPARROW$^+$ assigns more parameters to the coil, than to the strand super-class.

The discrepancy between Q_3-accuracy and MCC for the coil super-class and the generally low prediction quality for the strand super-class imply that the number of parameters for the strand class is insufficient. This problem may be further enhanced, since the strand super-class seems to be the most difficult class for prediction, primarily because β-strands are often stabilized by other β-strands forming H-bonds between backbone units that are spatial close but sequentially distant. In addition, the difference between recall and predication shown in chapter 3.3.1 for the adjustment of the regularization parameter can also be attributed to the large number of parameters used for the coil super-class.

The helix super-class is unaffected by an overlap with the coil super-class. There are two reasons for this: Firstly, helices are defined by short range interactions, which can be captured by a sliding window. Secondly, the helix super-class is larger than the strand super-class, making up about one third of the three super-classes, resulting in correspondingly larger weight. Hence, through the localized nature and large weight the number of parameters is sufficient to predict the helix super-class with high quality.

4.2 Hyperparameters

A variety of hyperparameters need to be adjusted to optimize the performance of *SPARROW$^+$. Among these, the choice of the BLAST database and the class reduction scheme proved to be the most crucial. The BLAST database is necessary for PSI-BLAST to generate the sequence profiles, the most important feature type and the class reduction scheme defines which DSSP classes are merged together into the three super-classes.

4.2.1 Sequence profiles

Two different databases were used for sequence profile generation, the *non-redundant* and the *UniRef90* BLAST database filtered with the program *pfilt* (Jones and Swindells, 2002). The former contains all non-redundant protein sequences and the latter only sequences with a maximum sequence identity of 90 % and no transmembrane, low complexity, coiled-coil or compositional biased sequence regions. From these databases very different sequence profiles are generated, which affect the prediction quality, as shown in section 3.3.2. Profiles generated from the *UniRef90* BLAST database improved the Q_3-accuracy by 1 % and the MCC by 0.02 as compared to the usage of profiles from the non-redundant BLAST database.

The removal of sequentially highly similar proteins as well as transmembrane or unordered proteins from the BLAST database improved the prediction quality. Hence, higher quality profiles could be generated from the filtered *UniRef90* BLAST database. Filtering the database and reducing the number close homologs helps to circumvent problems that can occur with iterative search methods such as PSI-BLAST (Cuff et al., 1998). Although these methods make it possible to find distant sequence homologies. However, these methods have major problems with multidomain proteins and proteins with regions of compositional bias. The former can cause that weak but relevant homologies are obscured because of common conserved protein domains. The latter are sequences containing low-complexity regions, such as coiled coils and transmembrane regions. These can cause an explosion of the search rather than a convergence because of the lack of any strong sequence signals. In contrast, premature convergence can occur if the profile generation is too strict only including very similar sequences. Hence, only sequences with the same domain organization are included but not homologues with different domain combinations (George and Heringa, 2002).

A general problem with iterative searches is "matrix migration" or "profile wander", which occurs when non-related homologs enter the search resulting in the loss of truly homologous sequences found in earlier iterations. This problem is particularly dangerous, if the non-related homolog belongs to a large family. A problem unique to PSI-BLAST is a loss of information because for profile generation only the highest scoring regions in a search are used, ignoring less conserved regions (George and Heringa, 2002).

Another critical source of errors for iterative methods, such as PSI-BLAST, is *homologous over-extension* (HOE), accounting for the largest fraction of PSI-BLAST errors. HOE occurs, if due to inaccurate domain alignments, parts of non-homologous domains are included in the profile that is used for the next search iteration. Thus, the signal from the original homology is suppressed. High alignment scores, which involve HOE may nevertheless correspond to true homologies. Hence, HOE can occur regardless of the chosen alignment threshold. Since HOEs are no statistical errors, they cannot be fixed by better statistics (Gonzalez and Pearson, 2010). HOE also explain, why profiles generated from manually curated alignments, such as *Pfam* (Finn et al., 2014), perform better than those generated from iterative searches, like PSI-BLAST, against the *non-redundant* BLAST database (Stojmirović et al., 2008). Sequences in the *non-redundant* database contain partial domains, which are primarily responsible for HOE, leading to a reduction in profile quality.

4.2.2 Class reduction schemes

The most popular program for protein secondary structure assignment, DSSP, discriminates eight different secondary structure classes. The different class are described in chapter 1.1.2 and how they are assigned by DSSP is explained in chapter 1.1.3. However, for most applications, only three different secondary structure classes are of interest: Helix, strand and coil. Since these classes are the combination of different DSSP classes, they can be referred to as super-classes.

As shown in chapter 2.2, there is no clear definition as to how the eight DSSP classes should be combined to the super-classes. The choice of the class reduction scheme has a major effect on the prediction quality, as the results in chapter 3.3.3 illustrate. The *mixed* class reduction scheme used in *SPARROW$^+$ achieved the overall best prediction quality. Assigning the 3_{10}-helix and isolated β-bridge to the coil instead of helix and strand super-classes, proved to be advantageous. The disadvantage is a further increase of the already large coil super-class, giving it even more weight.

4.2.3 Window size and classifier enhancements

The choice of the BLAST database and the class reduction scheme are parameters to improve the quality of the input data, which indirectly improves the overall performance of the vector-valued classifier (VVC). Parameters to directly influence the prediction quality of the VVC is the size of the sliding window or classifier enhancements. The former defines the

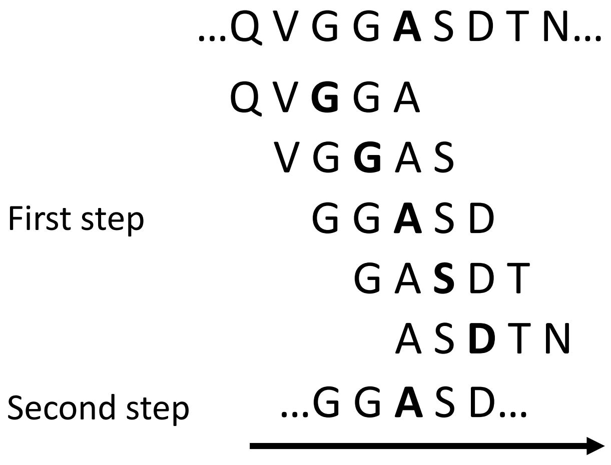

Figure 4-1: Implicit increase of the size of the sliding sequence window through second prediction step. For the prediction, a two-step classifier with a window size of 5 considers the same residues as a single-step classifier with a window size of 9. For the prediction of the window of the second-step of the two-step classifier the predictions for the windows of the first step are a perquisite. The residue to be classified is highlighted in bold type.

number of neighboring residues considered in a prediction and the latter comprises additional linear features or a second prediction step.

The prediction quality increases logarithmically with the window size, as Figure 3-10 shows. All enhancements improve the prediction quality, whereby the extent depends on the specific enhancement and window size. In general, all enhancements provide the greatest improvement at a smaller window size. At the largest considered window size of 17, improvements are minimal. Hence, enhancing the classifier allows to achieve with a smaller window size a similar prediction quality as with larger window sizes.

The number of adjustable classifier parameters is primarily determined by the window size. As Figure 3-9 depicts, the number of parameters increases geometrically with the window size. Accordingly, classifiers utilizing large windows have dramatically more parameters, making them computationally and resource demanding. In more practical terms, the triangular covariance matrix needed for training the VVC, see chapter 2.4.4.2, has a size of 13 gigabyte (GB) for a window size of 17 and of only 2.2 GB for a window size of 11 residue positions.

Apart from smoothing a secondary structure profile, predicted in the first step classifier, the second prediction step can also be interpreted as an implicit increase of the window size, as illustrated in Figure 4-1. For example, a two-step classifier using a window size of five residues, considers the same number of residues for the prediction as a single-step classifier with a

window size of 9. This implicit increase of the window size through the second prediction step can also explain, why the second step yields little improvement even for much larger windows. As seen in Figure 3-14 the improvement of prediction performance with an increased window size becomes minimal. However, using a larger window in the first prediction step, provides additional correlations, which cannot be obtained by the second prediction step.
Both prediction steps are independent, their only connection is through the structure features. These are the output of the first and the input of the second step. Accordingly, different parameter values, such as for instance for the window size can be used in both steps. A greater window size in the second step would not be as computationally expensive as in the first step, since *SPARROW$^+$ utilizes quadratic structure and linear profile features in the second step. The structure features have much fewer dimensions, only 3, compared to the 20 dimensions of the profile features. Increasing the window size in the second step could further increase the quality improvements made by the second step. In particular VVCs using a small window in the first step should benefit from this.

4.3 *SPARROW$^+$ compared to other methods

In chapter 3.5 the performance of *SPARROW$^+$ was compared with other popular secondary structure prediction methods, namely PSIPRED (Jones, 1999), C3-Scorpion (Yaseen and Li, 2014) and Jpred (Cuff and Barton, 2000). Chapter 1.2.1 gives an overview of these methods. In all test sets *SPARROW$^+$ achieved superior results than any of the other methods. *SPARROW$^+$ has many differences to the compared methods, which are also quite different between themselves. Foremost, *SPARROW$^+$ utilizes a vector-valued classifier (VVC), instead an artificial neural network (ANN) for prediction. Despite the different prediction method all methods have some parameters in common. All methods use a class reduction scheme to reduce the eight DSSP classes to three super-classes, a BLAST database to generate sequence profiles with PSI-BLAST and a sliding sequence window. However, the value of these parameters differs widely. The different class reduction schemes are listed in Table 2-1, showing that the major difference between all methods is the definition of the helix super-class. In contrast to the three compared methods, the *mixed* class reduction scheme used by *SPARROW$^+$ is the only one to assign the isolated β-bridge to the strand instead of the coil-class. The results in chapter 3.3.3 show the significant influence that the class reduction scheme has on the prediction quality. Concerning the BLAST database, only *SPARROW$^+$ and PSIPRED use the same setup in form of a filtered *UniRef90* database. Jpred also uses the *UniRef90* database, but unfiltered. C3-Scoprion on the other utilizes the *non-redundant* database but filtered. The size of the sliding window is 17 for *SPARROW$^+$ and Jpred and 15 for the other two methods. However, Jpred increases the window size to 19 in its second prediction step.

All these differences make it difficult to pinpoint the particular reasons for the variations in prediction quality and what precisely makes *SPARROW$^+$ superior to the compared methods. Of all compared methods, PSIPRED is the closest competitor to *SPARROW$^+$, in all test sets it achieved the second best prediction quality. Notably, the Q_3-accuracy for the strand super-class is always higher than the result of *SPARROW$^+$ highlighting the weakness of *SPARROW$^+$, discussed in section 4.1. Compared to C3-Scorpion and Jpred, *SPARROW$^+$ has more differences to them, than to PSIPRED. The similarities range from the choice of the BLAST database to the primary input features.

All parameters of *SPARROW$^+$ were optimized to achieve the highest prediction quality. As such, it is likely that the prediction quality difference is the sum of all optimizations. Hence, for secondary structure prediction the vector-valued classifier is comparable if not superior to artificial neural networks.

4.4 *SPARROW$^+$ compared to its predecessor

The predecessor to *SPARROW$^+$, is *SPARROW (Rasinski, 2011). Both programs are closely related, foremost by using a vector-valued classifier for secondary structure prediction. However, there is also a range of differences, beginning with the layout of the prediction method. As described in chapter 2.5, *SPARROW$^+$ utilizes two-prediction steps, whereby the input of each step is combination of different features types. Profile and sequence feature types in the first and structure and profile features types in the second step. While the vector-valued classifier of *SPARROW also consist of two-steps, there is also an additional third step in form of an ANN. Another difference is that *SPARROW only uses a single feature type in each step. Profiles in the first and structure feature types in the subsequent steps.

*SPARROW can work with different class reduction schemes, such as the *loose* (The helix super-class consists of the 3_{10}-, α-, and π-helix classes) and *pure* (Only the α-helix class is part of the helix super-class) schemes, albeit the best results were achieved with the *pure* scheme. At the time of the development of *SPARROW, the *pure* scheme led to a similar class composition than the *mixed* scheme used in *SPARROW$^+$. The schemes are described in chapter 2.2. The only difference is, whether to include the π-helix in the helix (*mixed*) or coil (*pure*) super-class. The reason for the similar composition is due to a bug in the secondary structure assignment method DSSP that was corrected in 2011. Before that, nearly all π-helices were assigned as α-helices (Cooley et al., 2010; Zacharias and Knapp, 2014). Hence, in the original ASTRAL40 dataset used for *SPARROW, only 0.2 % π-helices are found, whereas in the ASTRAL40 dataset used for *SPARROW$^+$ the frequency is 0.5 %. Another difference between *SPARROW$^+$ and its predecessor is the type BLAST database used for learning, since *SPARROW generates its sequence profiles using an unfiltered *non-redundant* BLAST database. The results in chapter 3.5 show that *SPARROW$^+$ is vastly superior to *SPARROW in all test sets. The difference is more than 3 % Q_3-accuracy and about 0.05 for the MCC. Actually, *SPARROW shows the worst performance of all methods. This somewhat contradicts to the results documented for *SPARROW. In cross validation *SPARROW achieved a Q_3-accuracy of 82.8 % and an MCC of 0.73. On a special test set the Q_3-accuracy is 81.5% and the MCC 0.71. The set consisted of protein domains from the ASTRAL40 dataset version 1.73 that have at most of sequence identity to protein domains from the 1.71 version of the dataset.

Why the prediction quality listed in here differs widely from the documented results is unclear. One factor could be the BLAST database, since the BLAST database used originally for the training *SPARROW could not be reconstructed, because BLAST databases are continuously updated and enlarged. As documented in chapter 3.3.2, different BLAST databases can lead to different profiles and affect prediction quality. However, explaining the large difference solely with differences in the BLAST database seems unreasonable. An additional explanation could also be that for an unknown reason *SPARROW has become defunct. But, even the documented prediction quality of *SPARROW, which is higher than the actual one, is inferior to *SPARROW$^+$. One approach to comprehend the quality difference, is to compare the prediction quality of *SPARROW after the first prediction step with those of *SPARROW$^+$ utilizing only a single-step classifier using only profile features. In this approach, the setup of the vector-valued classifiers is identical only some parameters differ. *SPARROW$^+$ reaches in the described setup a Q_3-accuracy of 83.0 % and an MCC of 0.73. In contrast, the Q_3-accuracy and MCC for the first step of *SPARROW is 81.5 % and 0.70 respectively. Hence, already in the first prediction step both programs show major quality differences. The parameters that differ between them are the type of BLAST database, the training dataset and the reduction scheme. As previously mentioned the differences in the reduction scheme should play a negligible role. Concerning the training set, *SPARROW used the old version 1.71 of the ASTRAL40 dataset without filtering. The training data for *SPARROW$^+$ is from the newer version 2.03 of the ASTRAL40 dataset and was filtered, as described in chapter 2.1. The results suggest that an unfiltered training data set contains too much noise that interferes with the training. The last parameter is the BLAST database whose significant influence on the prediction quality is discussed in chapters 4.2.1 and in chapter 4.3 in context of other prediction methods. It is unclear which BLAST database *SPARROW used, but it is safe to assume that it was not filtered. Filtering the training data and the BLAST database has thus resulted in an enormous increase in prediction quality.

While the additional steps of *SPARROW yield a quality gain of about 1.3 % Q_3-accuracy, the second step and feature combination yield only very small improvements. It is a similar tendency as observed for different window sizes. Enhancements of the classifier were much more beneficial for smaller than larger windows. In other words, the higher the original quality, the lower are the gains through further enhancements. Furthermore, a Q_3-accuracy beyond 82 % is by itself very difficult to achieve. Taking into account the little gains made by the enhancements used in *SPARROW$^+$ it seems unlikely that including a third step, like in *SPARROW, would led to a significant further improvement if any.

5 Conclusions and Outlook

The protein secondary structure prediction program *SPARROW$^+$ presented here, achieves a prediction quality that is superior to the other state-of-the-art method, such as PSIPRED (Buchan et al., 2013), C3-Scorpion (Yaseen and Li, 2014) and Jpred (Cuff et al., 1998). For secondary structure prediction, *SPARROW$^+$ utilizes an optimized version of the vector-valued classifier developed for its predecessor *SPARROW (Rasinski, 2011). The vector-valued classifier consists of two consecutive prediction steps, whereby each step consists of multiple linear regression functions using a set of class vector as an indicator matrix. The output of the first step is part of the input of the second step. The classes are defined by class vectors that represent the corners of a regular *simplex*, whereby the number of corners is equal to the number of classes. *SPARROW$^+$ enhanced the prediction performance of the vector-valued classifier compared to the original *SPARROW by using a combination of features in each of the two prediction steps and by optimizing the hyperparameters, such as the choice of the BLAST database used for generation of the sequence profile features.

The enhancements to the vector-valued classifier improved prediction quality albeit the extent is highly dependent on the size of the sliding window. The more neighboring residues are taken into account with a larger window, the smaller, yet still significant, is the effect.

Compared to other secondary structure prediction methods, *SPARROW$^+$ achieves a higher overall prediction quality in all test sets. However, the prediction quality for the three super-classes helix, strand and coil is slightly unbalanced. While higher prediction quality is attained for the helix and coil super-classes compared to any other method, the results for the strand super-class are slightly worse. *SPARROW$^+$ overestimates the coil at expense of the strand super-class.

The confidence measure developed to assess the reliability of predictions of the vector-valued classifier correlates with the prediction quality. Furthermore, the distribution of correct and incorrect classifications with respect to the confidence is significantly different. The higher the confidence, the greater the number of correct classifications. The confidence allows to estimate the prediction quality, whereby estimates should be made for each super-class separately.

As discussed, hyper parameters such as the BLAST database and class reduction scheme are not unique to *SPARROW$^+$ but are used by every secondary structure prediction method.

Accordingly, other methods could also be improved be utilizing the parameter values developed for *SPARROW$^+$.

Although *SPARROW$^+$ is currently the secondary structure prediction method that achieves the highest prediction quality, there are of course opportunities for further improvement. Some aspect are outlined in the following sections. It is also planned to make *SPARROW$^+$ available as a web service and a downloadable executable.

5.1 Weight optimization

In the current version of *SPARROW$^+$, the relevance or weight of a super-class is determined by its size. Hence, the larger coil super-class has a greater weight than the smaller strand super-class. The size of the weights define the effective numbers of parameters used for the prediction of the super-classes. The greater the weight, the higher the effective number of parameters. Therefore, *SPARROW$^+$ assigns more parameters to the coil than to the strand super-class. Accordingly, the prediction quality for the strand super-class is the worst of all super-classes, as discussed in chapter 4.1.

In order to improve the prediction quality for the strand super-class, the super-class weights must be treated as another hyperparameter and manually adjusted. However, to increase the weight of the strand super-class the weights of the other super-classes must be reduced. The best approach would be to primarily reduce the weight for the coil super-class. The coil super-class consists of secondary structure classes, which are not well defined, rendering an explicit recognition difficult, while an implicit recognition that identifies the other two classes may be easier to accomplish.

5.2 Improvement of the sequence profiles

Sequence profiles are the primary input for *SPARROW$^+$. The profiles are generated with PSI-BLAST (Altschul et al., 1997) from a BLAST (Altschul et al., 1990) database, as described in chapter 2.3.2. The results in chapter 3.3.2 show that the choice of the BLAST database directly influences the prediction quality, whereby the so far best results were achieved with the *UniRef90* (Suzek et al., 2007) BLAST database. Thus, improving sequence-profile quality would allow to further improve prediction quality. As discussed previously in chapter 4.2.1 PSI-BLAST can exit an alignment prematurely, if too many sequence with high sequence identity are found. This could still be the case for the *UniRef90* database. An approach to circumvent this problem would be to utilize the *UniRef50* database.

5.3 Predictions of more than three classes

A total of eight different secondary structure classes are defined by the program DSSP. However, apart from special applications, only three classes are of interest. These are the α-helix, β-strand and random coil classes. The remaining classes are merged with these three. Accordingly, most secondary structure prediction programs, including *SPARROW$^+$, make predictions for these three classes. However, *SPARROW (Rasinski, 2011), the predecessor of *SPARROW$^+$, was successfully used to predict all eight DSSP classes. *SPARROW and *SPARROW$^+$, both, use a vector-valued classifier for secondary structure prediction, accordingly *SPARROW$^+$ could easily be enhanced for predictions of more than three classes. As described in chapter 2.4.4.2 the vector-valued classifier is trained by solving systems of linear equations. The number of linear equation systems corresponds to the number of classes. The difference between the systems is the vector on the right-hand side and the class vectors. Hence, to predict more than three classes, the only requirements would be to generate new class vectors and right-hand side vectors. Regardless of the number of classes, the same covariance matrix is used. Hence, the computationally most expensive aspect of the vector-valued classifier would stay the same.

Apart from the three classes commonly used and the eight classes defined by DSSP, class reduction schemes to four or six classes are another reasonable approach. Four classes would allow to separate the 3_{10}-helix class from the coil super-class into a separate class. The 3_{10}-helix class is a well-defined class that also occurs in sufficient frequencies to train the classifier. The turn and bend classes might be more frequent, but do not have such a rigorous definition as the 3_{10}-helix class, see chapter 1.1.2. Using six instead of the total of eight classes can be useful namely if the bend class (involving bending in the protein backbone) and the isolated β-bridge class (poorly defined) of DSSP are merged with the coil class (Zacharias and Knapp, 2014).

5.4 Secondary structure profiles as input data

The transition from one protein structure class to another is rather continuous than discrete. Residues in the transition regime between two secondary structure classes possess properties of both classes. In particular, helix structures, described in chapter 1.1.2.1, can show such mixed classes. The π-helix for example is nearly exclusively found within α-helices. Furthermore, at the end of a helix the terminal residues are often partly helix and partly turn or coil.

The problem is that in most cases only an absolute secondary structure assignment is performed. Hence, although multiple secondary structure classes would apply, only one is selected and the other classes are disregarded. The rules for class selection are defined by the different protein secondary structure assignment methods. Accordingly, the majority of the differences between assignment methods are due to these problems as described in chapter 1.1.3.

The reduction to a single assignment has two major disadvantages, particularly for secondary structure prediction. The first problem is that one secondary structure class is given preference over another. Thus, an arbitrary hierarchy of secondary structure classes is established. The second problem is that information is lost especially about the transition zones between secondary structures.

Incorporating the information about all secondary structure classes applicable to a specific residue could improve prediction quality. Particularly predictions for more than three classes should benefit from this approach. Hence, the vector-valued classifier should be trained based on secondary structure profiles. As it is, this already applies for the second prediction step that uses the output from the first prediction step as input.

Secondary structure profiles could be obtained from intermediate steps of the secondary structure assignment methods DSSP (Kabsch and Sander, 1983) and PSSC (Zacharias and Knapp, 2014). In this intermediate step, these methods provide the information about all the secondary structure classes found for a specific residue.

5.5 Terminal residue feature type

At terminal residues an incomplete overlap of the sliding window and the protein sequence occurs, as described in chapter 2.3.3. In the current version, *SPARROW$^+$ uses default values for those positions. In case of sequence and profile features -1 and for structure features 0. Another approach to circumvent the problem of incomplete overlap is to introduce a new terminal feature type in addition to the default values. The terminal feature type would signal, if the position in the sliding window is still on the sequence or already beyond the terminal residue. Such a feature type would also allow to differentiate between N- and C-terminal incomplete overlap. As described in chapter 1.2.1, such a feature type was successfully implemented in other programs (Cuff and Barton, 2000; Jones, 1999; Yaseen and Li, 2014).

6 Summary

Knowing a proteins structure is an essential prerequisite for understanding its function. The rate of protein sequencing greatly exceeds the rate by which protein structures can be experimentally solved. Methods to predict the protein structure based on the sequence are therefore in great demand. The prediction of the protein secondary structure is the first step in predicting the spatial structure. In addition, the knowledge of the secondary structure allows to characterize a protein into a structure category.

In this work a new protein secondary structure prediction method, *SPARROW$^+$, is presented. *SPARROW$^+$ is a further development of its predecessor *SPARROW (Rasinski, 2011). Like its predecessor, a vector-valued classifier is used for prediction. The vector-valued classifier allows to project high dimensional input data into a low dimensional classification space. Through the relative orientation of the vector-valued classifier to class vectors, input data are classified.

*SPARROW$^+$ consists of two consecutive prediction steps. Based on a target sequence a PSI-BLAST (Altschul et al., 1997) sequence-profile is generated, which together with the protein sequence is the input of the first prediction step. The results of secondary structure prediction from the first step and the PSI-BLAST profile are the combined input for the second prediction step.

*SPARROW$^+$ achieves a Q_3-accuracy of 84 % for the ASTRAL40 (Fox et al., 2014) dataset and an at least 1 % higher Q_3-accuracy than the respective second best method for the CASP9, 10 and 11 datasets. Hence, *SPARROW$^+$ is currently superior to all presently available methods, like PSIPRED (Buchan et al., 2013).

In its current form, *SPARROW$^+$ overestimates the coil class, the largest secondary structure class, at the expense of the strand class, which is the smallest class. This results in a high Q_3-accuracy for coil and low one for strand. However, the *Matthews Correlation Coefficient* (MCC) (Gorodkin, 2004) is similar for both classes.

During the Development of *SPARROW$^+$ central parameters with a major influence on the prediction quality could be identified. To these parameters belong the choice of the BLAST (Altschul et al., 1990) database for the generation of the sequence profiles with PSI-BLAST and the class reduction scheme to reduce the eight DSSP (Kabsch and Sander, 1983) classes to three. Using the *UniRef90* (Suzek et al., 2014) BLAST database containing only homologs with

90 % sequence identity and filtering it with *pfilt* (Jones and Swindells, 2002) to remove transmembrane and unordered proteins, improved prediction quality. *SPARROW$^+$ uses a class reduction scheme that accounts for the peculiarities of DSSP and new insights concerning the π-helix secondary structure.

During the implementation of the enhancements of *SPARROW$^+$ it became obvious that the size of the sequence window is of critical importance for the prediction quality. The gains in prediction quality through enhancements of the vector-valued classifier, such as a second prediction step or the combination different types of input data, depend on the window size. The smaller the considered sequence window, the greater the corresponding gains in prediction quality. Therefore, the enhancements reduce the prediction quality gained with an increase of the window sizes.

Specifically for the vector-valued classifier of *SPARROW$^+$ a multiclass confidence measure was developed. The confidence can be correlated to prediction quality measures allowing to predict them. From a confidence of 0.8 a Q_3-accuracy of 90 % can be expected. Furthermore, the vector-valued classifiers show different confidence distributions for true and false-positive classifications.

Zusammenfassung

Kenntnisse über die Struktur eines Proteins sind von größter Bedeutung um dessen Funktion zu verstehen. Die Geschwindigkeit mit der Proteinsequenzen bestimmt werden überschreitet bei weitem die Rate mit der Proteinstrukturen experimentell gelöst werden. Deshalb sind Methoden, um die Proteinstruktur an Hand seiner Sequenz vorherzusagen, sehr gefragt. Die Vorhersage der Sekundärstruktur von Proteinen ist der erste Schritt, um dessen dreidimensionale räumliche Struktur vorherzusagen. Weiterhin erlaubt die Kenntnis über die Sekundärstruktur eines Proteins dessen Zuordnung in eine Faltungsklasse.

In dieser Arbeit wird ein neues Programm, *SPARROW$^+$, zur Vorhersage der Sekundärstruktur von Proteinen vorgestellt. *SPARROW$^+$ ist die Weiterentwicklung seines Vorgänger *SPARROW (Rasinski, 2011). Wie sein Vorgänger wird für die Vorhersage ein vektorwertiger Klassifikator verwendet. Dieser Klassifikator erlaubt hoch dimensionale Eingangsdaten in einen niedrig dimensionalen Raum zu projizieren. Die Klassifizierung der Eingangsdaten erfolgt durch die relative Orientierung des vektorwertigen Klassifikators zu Klassenvektoren.

*SPARROW$^+$ besteht in zwei aufeinander folgenden Vorhersageschritten. Aus der Eingangssequenz wird ein PSI-BLAST (Altschul et al., 1997) Profil generiert, welches zusammen mit der Sequenz die Eingabe für die erste Stufe ist. Die Vorhersage der ersten Stufe und das PSI-BLAST Profil sind die kombinierten Eingangsdaten für die zweite Stufe.

*SPARROW$^+$ erreicht eine Q_3-Genauigkeit von 84 % auf dem ASTRAL40 (Fox et al., 2014) Datensatz und erzielt auf den CASP9, 10 und 11 Datensätzen eine 1 % höhere Q_3-Genauigkeit als die jeweilige zweitbeste Methode. *SPARROW$^+$ ist hiermit allen anderen aktuellen Methoden wie z.B. PSIPRED (Buchan et al., 2013) überlegen.

In seiner derzeitigen Form überschätzt *SPARROW$^+$ den Anteil der Coil-Strukturen, die größte Sekundärstrukturklasse, auf Kosten von Strand, der kleinsten Klasse. Dies führt zu einer hohen Q_3-Genauigkeit für Coil und einer niedrigen für Strand, wobei der *Matthews Correlation Coefficient* (MCC) (Gorodkin, 2004) für beide Klassen ähnlich ist.

Bei der Entwicklung von *SPARROW$^+$ konnten zentrale Parameter ermittelt werden, welche einen großen Einfluss auf die Vorhersagequalität haben. Zu diesen Parametern gehören die Wahl der BLAST (Altschul et al., 1990) Datenbank für die Generierung der Sequenzprofile mit PSI-BLAST und das Reduktionsschema um die acht DSSP (Kabsch and Sander, 1983) Klassen auf drei zu reduzieren. Bei der BLAST Datenbank zeigte sich das eine Reduzierung der

Homologen von 100 auf 90 % durch Verwendung der *UniRef90* (Suzek et al., 2014) Datenbank, sowie das Entfernen von Transmembran und ungeordneten Proteinen mittels *pfilt* (Jones and Swindells, 2002), die Vorhersagequalität erhöht. *SPARROW$^+$ verwendet ein Reduktionsschema, welches die Eigenheiten von DSSP sowie Erkenntnisse bezüglich der π-Helix berücksichtigt.

Bei der Implementierung von Erweiterungen für *SPARROW$^+$ zeigte sich, dass für die Vorhersagequalität die Größe des Sequenzfensters von entscheidender Bedeutung ist. Der Gewinn an Vorhersagequalität durch Erweiterungen des vektorwertigen Klassifikators durch eine zweite Stufe oder die Kombination von verschiedenen Typen von Eingangsdaten ist abhängig von der Fenstergröße. Je kleiner das Fenster desto größer ist der Gewinn an Genauigkeit der Vorhersage. Allerdings reduzieren diese Erweiterungen die Verbesserungen der Vorhersagequalität mit zunehmender Fenstergröße.

Speziell für den vektorwertigen Klassifikator von *SPARROW$^+$ wurde ein Multi-Klassen Konfidenzmaß entwickelt. Die Konfidenz lässt sich mit Vorhersagequalitätsmaßen korrelieren und ermöglicht so eine Vorhersage von selbigen. Ab einer Konfidenz von 0.8 ist eine Q_3-Genauigkeit von 90 % zu erwarten. Weiterhin zeigt sich, dass der vektorwertige Klassifikator unterschiedliche Konfidenzverteilungen aufweist für richtig und falsch-positive Klassifikationen.

7 References

1. Altschuh, D., A. Lesk, A. Bloomer, and A. Klug. 1987. Correlation of co-ordinated amino acid substitutions with function in viruses related to tobacco mosaic virus. *Journal of molecular biology*. 193:693-707.
2. Altschuh, D., T. Vernet, P. Berti, D. Moras, and K. Nagai. 1988. Coordinated amino acid changes in homologous protein families. *Protein engineering*. 2:193-199.
3. Altschul, S.F., W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. 1990. Basic local alignment search tool. *Journal of molecular biology*. 215:403-410.
4. Altschul, S.F., T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*. 25:3389-3402.
5. Andreeva, A., D. Howorth, C. Chothia, E. Kulesha, and A.G. Murzin. 2014. SCOP2 prototype: a new approach to protein structure mining. *Nucleic acids research*. 42:D310-D314.
6. Anfinsen, C.B. 1973. Principles that govern the folding of protein chains. *Science*. 181:223-230.
7. Astbury, W., and H.t. Woods. 1934. X-ray studies of the structure of hair, wool, and related fibres. II. The molecular structure and elastic properties of hair keratin. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*:333-394.
8. Baldwin, R.L., and G.D. Rose. 1999. Is protein folding hierarchic? I. Local structure and peptide folding. *Trends in biochemical sciences*. 24:26-33.
9. Berman, H., K. Henrick, and H. Nakamura. 2003. Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*. 10:980-980.
10. Berman, H.M., J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. 2000. The protein data bank. *Nucleic acids research*. 28:235-242.
11. Bettella, F. 2009. Protein secondary structure prediction using optimized scoring functions. Freie Universität Berlin, Germany.
12. Bettella, F., D. Rasinski, and E.W. Knapp. 2012. Protein secondary structure prediction with SPARROW. *Journal of chemical information and modeling*. 52:545-556.
13. Bowie, J.U., R. Luthy, and D. Eisenberg. 1991. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*. 253:164-170.
14. Brenner, S.E., P. Koehl, and M. Levitt. 2000. The ASTRAL compendium for protein structure and sequence analysis. *Nucleic acids research*. 28:254-256.
15. Buchan, D.W., F. Minneci, T.C. Nugent, K. Bryson, and D.T. Jones. 2013. Scalable web services for the PSIPRED Protein Analysis Workbench. *Nucleic acids research*. 41:W349-W357.
16. Chandonia, J.-M., N.S. Walker, L.L. Conte, P. Koehl, M. Levitt, and S.E. Brenner. 2002. ASTRAL compendium enhancements. *Nucleic Acids Research*. 30:260-263.
17. Chandonia, J.M., G. Hon, N.S. Walker, L.L. Conte, P. Koehl, M. Levitt, and S.E. Brenner. 2004. The ASTRAL compendium in 2004. *Nucleic acids research*. 32:D189-D192.
18. Chou, P.Y., and G.D. Fasman. 1974a. Conformational parameters for amino acids in helical, β-sheet, and random coil regions calculated from proteins. *Biochemistry*. 13:211-222.
19. Chou, P.Y., and G.D. Fasman. 1974b. Prediction of protein conformation. *Biochemistry*. 13:222-245.

20. Chou, P.Y., and G.D. Fasman. 1978. Empirical predictions of protein conformation. *Annual review of biochemistry*. 47:251-276.
21. Cole, C., J.D. Barber, and G.J. Barton. 2008. The Jpred 3 secondary structure prediction server. *Nucleic acids research*. 36:W197-W201.
22. Consortium, U. 2014. UniProt: a hub for protein information. *Nucleic Acids Research*:gku989.
23. Cooley, R.B., D.J. Arp, and P.A. Karplus. 2010. Evolutionary origin of a secondary structure: π-helices as cryptic but widespread insertional variations of α-helices that enhance protein functionality. *Journal of molecular biology*. 404:232-246.
24. Coxeter, H.S.M. 1973. Regular polytopes. Courier Corporation.
25. Cuff, J.A., and G.J. Barton. 2000. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*. 40:502-511.
26. Cuff, J.A., M.E. Clamp, A.S. Siddiqui, M. Finlay, and G.J. Barton. 1998. JPred: a consensus secondary structure prediction server. *Bioinformatics*. 14:892-893.
27. de Paola, I., L. Pirone, M. Palmieri, N. Balasco, L. Esposito, L. Russo, D. Mazzà, L. Di Marcotullio, S. Di Gaetano, and G. Malgieri. 2015. Cullin3-BTB Interface: A Novel Target for Stapled Peptides.
28. Dor, O., and Y. Zhou. 2007. Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins: Structure, Function, and Bioinformatics*. 66:838-845.
29. Drozdetskiy, A., C. Cole, J. Procter, and G.J. Barton. 2015. JPred4: a protein secondary structure prediction server. *Nucleic Acids Res*. 43:W389-394.
30. Dunker, A.K., I. Silman, V.N. Uversky, and J.L. Sussman. 2008. Function and structure of inherently disordered proteins. *Current opinion in structural biology*. 18:756-764.
31. Eck, R.V., and M.O. Dayhoff. 1966. {Atlas of Protein Sequence and Structure}.
32. Eddy, S.R. 1998. Profile hidden Markov models. *Bioinformatics*. 14:755-763.
33. Finn, R.D., A. Bateman, J. Clements, P. Coggill, R.Y. Eberhardt, S.R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E.L.L. Sonnhammer, J. Tate, and M. Punta. 2014. Pfam: the protein families database. *Nucleic Acids Research*. 42:D222-D230.
34. Fisher, R.A. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*. 7:179-188.
35. Fitzkee, N.C., P.J. Fleming, and G.D. Rose. 2005. The Protein Coil Library: a structural database of nonhelix, nonstrand fragments derived from the PDB. *Proteins: Structure, Function, and Bioinformatics*. 58:852-854.
36. Floudas, C., H. Fung, S. McAllister, M. Mönnigmann, and R. Rajgaria. 2006. Advances in protein structure prediction and de novo protein design: A review. *Chemical Engineering Science*. 61:966-988.
37. Fox, N.K., S.E. Brenner, and J.-M. Chandonia. 2014. SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic acids research*. 42:D304-D309.
38. Frishman, D., and P. Argos. 1995. Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Bioinformatics*. 23:566-579.
39. Garnier, J., D. Osguthorpe, and B. Robson. 1978. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of molecular biology*. 120:97-120.

40. Gasteiger, E., A. Gattiker, C. Hoogland, I. Ivanyi, R.D. Appel, and A. Bairoch. 2003. ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic acids research*. 31:3784-3788.
41. Gauss, C.F. 1809. Theoria motus corporum coelestium in sectionibus conicis solem ambientium auctore Carolo Friderico Gauss. sumtibus Frid. Perthes et IH Besser.
42. George, R.A., and J. Heringa. 2002. Protein domain identification and improved sequence similarity searching using PSI-BLAST. *Proteins: Structure, Function, and Bioinformatics*. 48:672-681.
43. Ginalski, K., J. Pas, L.S. Wyrwicz, M. Von Grotthuss, J.M. Bujnicki, and L. Rychlewski. 2003. ORFeus: detection of distant homology using sequence profiles and predicted secondary structure. *Nucleic acids research*. 31:3804-3807.
44. Gonzalez, M.W., and W.R. Pearson. 2010. Homologous over-extension: a challenge for iterative similarity searches. *Nucleic acids research*. 38:2177-2189.
45. Gorodkin, J. 2004. Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*. 28:367-374.
46. Güntert, P. 2004. Automated NMR structure calculation with CYANA. *In* Protein NMR Techniques. Springer. 353-378.
47. Hadamard, J. 1902. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*. 13:28.
48. Henikoff, S., and J.G. Henikoff. 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*. 89:10915-10919.
49. Hinton, G., L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T.N. Sainath. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*. 29:82-97.
50. Hollingsworth, S.A., D.S. Berkholz, and P.A. Karplus. 2009. On the occurrence of linear groups in proteins. *Protein Science*. 18:1321-1325.
51. Jones, D.T. 1999. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*. 292:195-202.
52. Jones, D.T., and M.B. Swindells. 2002. Getting the most from PSI–BLAST. *Trends in biochemical sciences*. 27:161-164.
53. Jones, D.T., W. Taylort, and J.M. Thornton. 1992. A new approach to protein fold recognition.
54. Jurman, G., S. Riccadonna, and C. Furlanello. 2012. A comparison of MCC and CEN error measures in multi-class prediction. *PloS one*. 7:e41882.
55. Kabsch, W., and C. Sander. 1983. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*. 22:2577-2637.
56. Kelley, L.A., and M.J. Sternberg. 2009. Protein structure prediction on the Web: a case study using the Phyre server. *Nature protocols*. 4:363-371.
57. Krause, E.F. 1973. TAXICAB GEOMETRY. *The Mathematics Teacher*. 66:695-706.
58. Krizhevsky, A., I. Sutskever, and G.E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. *In* Advances in neural information processing systems. 1097-1105.
59. Kryshtafovych, A., K. Fidelis, and J. Moult. 2011. CASP9 results compared to those of previous CASP experiments. *Proteins: Structure, Function, and Bioinformatics*. 79:196-207.

60. Kryshtafovych, A., K. Fidelis, and J. Moult. 2014. CASP10 results compared to those of previous CASP experiments. *Proteins: Structure, Function, and Bioinformatics*. 82:164-174.
61. Labeit, S., and B. Kolmerer. 1995. Titins: giant proteins in charge of muscle ultrastructure and elasticity. *Science*. 270:293-296.
62. Legendre, A.M. 1805. Nouvelles méthodes pour la détermination des orbites des comètes. F. Didot.
63. Lesk, A.M., L. Lo Conte, and T.J. Hubbard. 2001. Assessment of novel fold targets in CASP4: Predictions of three-dimensional structures, secondary structures, and interresidue contacts. *Proteins: Structure, Function, and Bioinformatics*. 45:98-118.
64. Lezon, T.R., J.R. Banavar, A.M. Lesk, and A. Maritan. 2006. What determines the spectrum of protein native state structures? *Proteins: Structure, Function, and Bioinformatics*. 63:273-277.
65. Li, S.-C., N.K. Goto, K.A. Williams, and C.M. Deber. 1996. Alpha-helical, but not beta-sheet, propensity of proline is determined by peptide environment. *Proceedings of the National Academy of Sciences*. 93:6676-6681.
66. Lodish, H.F., A. Berk, S.L. Zipursky, P. Matsudaira, D. Baltimore, and J. Darnell. 2000. Molecular cell biology. WH Freeman New York.
67. Martí-Renom, M.A., A.C. Stuart, A. Fiser, R. Sánchez, F. Melo, and A. Šali. 2000. Comparative protein structure modeling of genes and genomes. *Annual review of biophysics and biomolecular structure*. 29:291-325.
68. Matthews, B.W. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*. 405:442-451.
69. McCulloch, W.S., and W. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 5:115-133.
70. Milner, W. 2007. In Manhattan π Is 4: Taxicab Geometry. *Mathematics in School*. 36:33-34.
71. Moult, J., K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano. 2014. Critical assessment of methods of protein structure prediction (CASP)—round x. *Proteins: Structure, Function, and Bioinformatics*. 82:1-6.
72. Nelson, D.L., A.L. Lehninger, and M.M. Cox. 2008. Lehninger principles of biochemistry. Macmillan.
73. Nielsen, M.A. 2015. Neural Networks and Deep Learning. Determination Press.
74. Osguthorpe, D.J. 2000. Ab initio protein folding. *Current Opinion in Structural Biology*. 10:146-152.
75. Pauling, L., and R.B. Corey. 1951. The pleated sheet, a new layer configuration of polypeptide chains. *Proc Natl Acad Sci USA*. 37:251-256.
76. Pylouster, J., A. Bornot, C. Etchebest, and A.G. de Brevern. 2010. Influence of assignment on the prediction of transmembrane helices in protein structures. *Amino acids*. 39:1241-1254.
77. Rasinski, D. 2011. Lösen des Mehrklassenproblems der Sekundärstrukturvorhersage von Proteinen. Freie Universität Berlin, Germany.
78. Riek, R.P., and R.M. Graham. 2011. The elusive π-helix. *Journal of structural biology*. 173:153-160.
79. Rohl, C.A., C.E. Strauss, K.M. Misura, and D. Baker. 2004. Protein structure prediction using Rosetta. *Methods in enzymology*. 383:66-93.

80. Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*. 65:386.

81. Rosenblatt, F. 1962. Principles of neurodynamics: perceptrons and the theory of brain mechanisms. Spartan Books.

82. Rost, B. 2001. Review: protein secondary structure prediction continues to rise. *Journal of structural biology*. 134:204-218.

83. Rost, B., and C. Sander. 1993a. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proceedings of the National Academy of Sciences*. 90:7558-7562.

84. Rost, B., and C. Sander. 1993b. Prediction of protein secondary structure at better than 70% accuracy. *Journal of molecular biology*. 232:584-599.

85. Roy, A., A. Kucukural, and Y. Zhang. 2010. I-TASSER: a unified platform for automated protein structure and function prediction. *Nature protocols*. 5:725-738.

86. Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*. 5:3.

87. Sanchez, R., and A. Šali. 1997. Advances in comparative protein-structure modelling. *Current opinion in structural biology*. 7:206-214.

88. Schwarz, H.R., and N. Köckler. 2011. Numerische Mathematik. Wiesbaden : Vieweg + Teubner, Wiesbaden. 591 S. ; graph. Darst. pp.

89. Sippl, M.J. 1990. Calculation of conformational ensembles from potentials of mena force: an approach to the knowledge-based prediction of local structures in globular proteins. *Journal of molecular biology*. 213:859-883.

90. Smith, J.A., L.G. Pease, and K.D. Kopple. 1980. Reverse Turns in Peptides and Protein. *Critical Reviews in Biochemistry and Molecular Biology*. 8:315-399.

91. Solis, A.D., and S. Rackovsky. 2004. On the use of secondary structure in protein structure prediction: a bioinformatic analysis. *Polymer*. 45:525-546.

92. Srinivasan, R., and G.D. Rose. 1995. LINUS: a hierarchic procedure to predict the fold of a protein. *Proteins: Structure, Function, and Bioinformatics*. 22:81-99.

93. Srinivasan, R., and G.D. Rose. 1999. A physical basis for protein secondary structure. *Proceedings of the National Academy of Sciences*. 96:14258-14263.

94. Stojmirović, A., E.M. Gertz, S.F. Altschul, and Y.-K. Yu. 2008. The effectiveness of position-and composition-specific gap costs for protein similarity searches. *Bioinformatics*. 24:i15-i23.

95. Suzek, B.E., H. Huang, P. McGarvey, R. Mazumder, and C.H. Wu. 2007. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*. 23:1282-1288.

96. Suzek, B.E., Y. Wang, H. Huang, P.B. McGarvey, C.H. Wu, and U. Consortium. 2014. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*:btu739.

97. Tikhonov, A.N., and V.I.A.k. Arsenin. 1977. Solutions of ill-posed problems. Vh Winston.

98. Tyagi, M., A. Bornot, B. Offmann, and A.G. de Brevern. 2009. Analysis of loop boundaries using different local structure assignment methods. *Protein Science*. 18:1869-1881.

99. Villarreal, M.R. 2008. Main protein structures levels. p.s.l. en.svg, editor, Wikipedia, the free encyclopedia.

100. Wright, P.E., and H.J. Dyson. 1999. Intrinsically unstructured proteins: re-assessing the protein structure-function paradigm. *Journal of molecular biology*. 293:321-331.

101. Yaseen, A., and Y. Li. 2014. Context-based features enhance protein secondary structure prediction accuracy. *Journal of chemical information and modeling*. 54:992-1002.

102. Zacharias, J., and E.-W. Knapp. 2014. Protein secondary structure classification revisited: Processing DSSP information with PSSC. *Journal of chemical information and modeling*. 54:2166-2179.