Master Thesis

Structure prediction of proteins using nearest neighbor trees

Author(s):
Signorell, Gian Andrea

Publication Date:
2003

Permanent Link:
https://doi.org/10.3929/ethz-a-004616559

Rights / License:
In Copyright - Non-Commercial Use Permitted
Structure Prediction of Proteins Using Nearest Neighbor Trees

Author: Gian Andrea Signorell

Topic proposed by Prof. Gaston Gonnet

November 11, 2003
Abstract

In this thesis an implementation of a nearest neighbor tree for the prediction of secondary structure of proteins is described. The data structure of the nearest neighbors tree uses chemical and physical properties of the amino acids. This features are chosen out of a bigger set of features according to a best basis analysis.

A training set and a test set are generated from the CB513 set [6] and are independent from each other.

The different programs are implemented in the DARWIN-environment [1].

Different feature sets where used to make predictions and their prediction accuracies reached up to 57.3% in the $Q_3$-measure [28].
Contents

1 Introduction .......................................................... 3
  1.1 General Proceeding ............................................. 3
  1.2 Types of Secondary Structures ............................... 4
    1.2.1 Alpha Helices .......................................... 4
    1.2.2 Beta Sheets ............................................. 6
    1.2.3 Turns .................................................... 7

2 Generating a Training and a Test Data Set ....................... 9
  2.1 Choice of Structures for the Test and Training Set ......... 9
    2.1.1 Non-Redundancy ....................................... 9
    2.1.2 Representative Sets of Proteins ....................... 10
  2.2 The CB513-Set ............................................... 10
    2.2.1 Darwin and CB513 ..................................... 10

3 Selection of Variables Used in the Prediction ..................... 12
  3.1 Amino Acids and their Properties ........................... 12
  3.2 Best Basis .................................................. 13
    3.2.1 Approximation: Stepwise Regression .................. 13
    3.2.2 Other Heuristics ..................................... 14
  3.3 myBestBasis: A DARWIN Implementation ..................... 14
    3.3.1 The Features ......................................... 14
    3.3.2 The Calculation of the Best Basis .................... 15
  3.4 The Results ................................................ 16
    3.4.1 Parameters ............................................ 16
    3.4.2 The Best Basis ........................................ 17
    3.4.3 How to continue? ...................................... 18

4 Nearest Neighbors .................................................. 19
  4.1 NearestNeighbours: A DARWIN Implementation ............... 20
    4.1.1 myNN .................................................. 20
    4.1.2 init .................................................... 20
    4.1.3 myMap .................................................. 21
    4.1.4 myDist ................................................ 21
    4.1.5 norm ..................................................... 21
    4.1.6 myTreeBuild .......................................... 21
    4.1.7 NNSearch .............................................. 22
    4.1.8 Search_kNN ............................................ 22
    4.1.9 myPredict .............................................. 23
5 Tests of the Predictions 24
5.1 Accuracy Tests for the Prediction .......................... 24
  5.1.1 Q3 ........................................... 24
  5.1.2 SOV ........................................ 25
5.2 The Results .......................................... 26
  5.2.1 (3-0-3-0-5-5-3-3) ................................ 28
  5.2.2 (5-1-5-1-7-7-5-5) ................................ 30
  5.2.3 (1-1-7-1-9-9-5-9) ................................ 33
  5.2.4 (5-5-5-5-5-5-5-5) ................................ 35
  5.2.5 (0-0-5-0-7-7-0-7) ................................ 37
  5.2.6 (0-0-7-9-9-0-9) ................................ 39
  5.2.7 (0-9-0-9-0-9-0-9) ................................ 42

6 Conclusion 45
  6.1 Interpretation of the Results .............................. 45
  6.2 Other Methods of Structure Prediction .................. 46
    6.2.1 First Generation ................................ 46
    6.2.2 Second Generation ................................ 47
    6.2.3 Third Generation ................................. 48
  6.3 The End ............................................ 48

A Values of the Features 53
  A.1 Hydrophobicity ...................................... 53
  A.2 Molecular Weight ................................... 54
  A.3 Charge ............................................. 55
  A.4 Surface ............................................ 56
  A.5 Solubility .......................................... 57
  A.6 Chou-Fasman Parameter for Alpha-Helices ............. 58
  A.7 Chou-Fasman Parameter for Beta-Strands .............. 59
  A.8 Chou-Fasman Parameter for Turns ..................... 60
1 Introduction

The goal of this thesis is to predict structural elements of proteins using nearest neighbor trees. Structural elements would be helices, strands, parses or the lack of those elements. Features to use in the prediction may include hydrophobicity, size, charge and possibly other characteristics of amino acids. The Swiss-Prot database [3] will be chosen as the primary source of information.

The main problems expected to be solved in this project are the selection of relevant features for the prediction and the implementation of the nearest neighbor tree.

The implementation will use only information on the amino acid structure. This fact categorizes the algorithm as one of the second generation in structure prediction.

From a biochemical point of view, the information on the amino acid sequence is sufficient to predict the structure of a protein [8]. The chemical properties of the amino acid residues allow the protein to fold in a energy minimal conformation, which is unique. The interactions of amino acids, also far away from each other in the sequence, make this calculations very complicated. In the used models, only short range interactions are taken into account, which sets this generation of structure prediction a limit in the accuracy.

Therefore a third generation of structure prediction methods was developed and is currently very successful in predicting secondary structures. This generation of algorithms uses, additional to the sequence information, evolutionary information on protein families.

1.1 General Proceeding

The general proceeding to achieve the goal of secondary structure prediction can be divided in three main parts:

- Two data sets will be compiled: One that is used to fit the parameters and to build the nearest neighbors tree and the other to test the resulting prediction model.

- A best basis algorithm will be used to find the variables which are the most relevant ones in a least squares solution.
• A nearest neighbors tree will be built using a combination of the variables. The prediction of the secondary structure will be done using this tree.

At the end we have a set of known structures, which were not used to determine the model: the test set. This set is used to determine the quality of the predictions.

1.2 Types of Secondary Structures

There are three common secondary structures in proteins, namely alpha helices, beta sheets, and turns. Parts of structures which cannot be classified as one of the standard three classes are usually grouped into a category called "other" or "random coil". This last designation is unfortunate as no portion of protein three dimensional structure is truly random and it is usually not a coil. A number of "other" secondary structures types have been proposed, however they represent a small fraction of residues and may not be seen as a general structural principle of proteins. In this work I will often refer to such a part of the sequence as no structure or as coil which means this parts are not part of one of the three structures helix, sheet or turn.

As a source of more precise information on secondary structure types [20] may be useful.

1.2.1 Alpha Helices

The alpha helix is the most abundant helical conformation found in globular proteins accounting for 32-38% of all residues [20]. The average length of an alpha helix is 10 residues. Fig. 1 shows a picture of a alpha helix.

The abundance of this particular form of secondary structure stems from the following properties:

• The backbone conformation of the peptide in an alpha helix lies in the center of a minimum energy region.

• The dipoles of hydrogen bonding backbone atoms are in near perfect alignment.

• The radius of the helix allows for favorable van der Waals interactions across the helical axis.

• Side chains are well staggered minimizing steric interference.
Hydrogen bonds within an alpha-helix display repeating pattern in which the backbone C=O of residue i hydrogen bonds to the backbone HN of residue i+4.

Residues in positions (i, i+3) and (i, i+4) are positioned in such a way as to force interaction of their side chains. This can have a stabilizing effect if the residues are of opposite charge or are both hydrophobic. Interaction between aromatic rings (Phe) at position (i) and His at position (i+4) appears to have a stabilizing effect on the helical conformation.

There are also other types of helices such as the 3_10-helix or the π-helix, but they are rare. This two types of helices will be denoted as helix in the
secondary structure information of Swissprot and also in the used CB513-data set: Whenever I will use the term helix in this thesis there will be no differentiation between this three types of helices, but in the most cases one can assume an alpha-helix.

1.2.2 Beta Sheets

Besides alpha helices, beta sheets (Fig. 2) are another major structural element in globular proteins containing 20-28% of all residues [20]. The basic unit of a beta sheet is a beta strand (which can be thought of as a helix with two residues per turn). The beta strand is then like the alpha helix, a repeating secondary structure. However, since there are no intrasegment hydrogen bonds and van der Waals interactions between atoms of neighboring residues are not significant due to the extended nature of the chain, this extended conformation is only stable as part of a beta sheet where contributions from hydrogen bonds and van der Waals interactions between aligned strands exert a stabilizing influence.

Figure 2: Beta Sheet
Beta sheets are found in two forms designated as "Antiparallel" or "Parallel" based on the relative directions of two interacting beta strands. The average length of a beta sheet is about 6 residues and most beta sheets contain less than 6 strands. Side chains from adjacent residues of a strand in a beta sheet are found on opposite sides of the sheet and do not interact with one another. Therefore, like alpha-helices, beta-sheets have the potential for amphiphilicity with one face being polar and the other being apolar. However, unlike alpha-helices which are comprised of residues from a continuous polypeptide segment (i.e., hydrogen bonds between CO of residue i and NH of residue i+3), beta sheets are formed from strands that are very often from distant portions of the polypeptide sequence.

1.2.3 Turns

Turns are the third of the three "classical" secondary structures with approximately one-third of all residues in globular proteins are contained in turns that serve to reverse the direction of the polypeptide chain (Fig. 3).

![Figure 3: One possible Turn](image)

This is perhaps not so surprising as the diameter of the average globular protein domain is roughly 25 Å (an extended polypeptide conformation would require 7 residues to traverse the domain before having to change di-
reactions). Turns are located primarily on the protein surface and accordingly contain polar and charged residues.
2 Generating a Training and a Test Data Set

To make a prediction about the secondary structure of proteins it is essential to have data of secondary structures that are already known. With the help of this data we can make a model as a generalization of the relationships between features and structures and, as result of this generalization, make predictions of not yet known proteins.

Data is widely available, since some thousands of structures have been solved, i.e. the three dimensional structure of these is known. The Protein Data Bank (PDB) [2] contains this solved structures and their three dimensional information.

The performance of a prediction model has to be measured in the end by a test set, which is independent from the training set. Since new structures are solved continuously, the use of this structures would generate a good measure for the prediction. However this new structures are not guaranteed independent, since similar (homologous) sequences can be part of the training set.

2.1 Choice of Structures for the Test and Training Set

Most secondary structure prediction methods include a set of parameters that must be estimated. Values for the parameters are obtained by statistical analysis or by learning from a set of proteins for which the tertiary structure is known. This is the training set of proteins. Testing predictive accuracy on the training set leads to unrealistically high accuracies. An objective test of a secondary structure prediction method will predict the structures of a test set of proteins that are not in the training set and show no detectable sequence similarity with the training set. If the test is to be balanced, then both training and test sets should have a similar distribution of secondary structure classes and types [7].

2.1.1 Non-Redundancy

There is a big variety of secondary structure prediction methods. Most of these methods use information on already solved structures, so there are a lot of standard test sets available and they have a very good theoretical foundation.

The biggest effort in generating these data sets was put in the reassurance of non-rendundancy.

The simplest form of non redundancy is to ensure a certain upper limit for sequence identity. 25% is a common threshold for this method. Percent-
age identity can be a poor measure of sequence similarity, particularly for values below 30%. Percentage identity is dependent upon both the length of the alignment and the composition of the sequences. Thus, two sequences of similar unusual amino acid composition may give high values of percentage identity, even when unrelated.

There are more sophisticated methods to measure the similarity between two protein sequences. These use the scores of the alignment compared to randomized scores of alignments of the sequences [5].

2.1.2 Representative Sets of Proteins

The test set has to be representative for the data base (ideally for all existing proteins), i.e., all known sequence families should be included, and they should be included only once [9].

2.2 The CB513-Set

The generation of data sets can be quite complicated and is a main ingredient for a good quality of a prediction. There has been a lot of effort in the generation of good data sets and a lot of them are available for the use of secondary structure prediction.

James A. Cuff and Geoffrey J. Barton generated in [5] a set of 513 sequences, which are suitable for cross-validation of secondary structure prediction methods without artifacts due to internal homology.

This 513 non redundant sequences can be used to test new secondary structure prediction methods. 396 sequences are derived from the 3Dmd database of protein domains [13] plus 117 proteins from the Rost set of 126 non redundant proteins. All sequences in this set have been compared pairwise, and are non redundant to a 5 x standard deviation cut-off.

How this set was exactly generated can be read in [5]. The database can be downloaded at [6].

The set was generated for predictions and is therefore suitable for our prediction model. Another advantage is, that the results from our model can be compared to other results derived from this set.

2.2.1 Darwin and CB513

The implementation of the nearest neighbors tree will be done in DARWIN [1]. To generate this tree, we need the training and test data to be available in DARWIN. For this purpose I wrote a perl-script, which takes each file from the CB513 set and generates one Darwin loadable file (darwin file).
This file contains the sequence, links to the PDB entries and information on the secondary structure. The information on the secondary structure was taken from the DSSP-field of the CB513-data and then changed, the same way Swiss-Prot does it, into the Darwin format. This format differentiates between helix (h), sheet (s) and turn (t). Other forms are matched, according to [3] into one or none of this categories.
3 Selection of Variables Used in the Prediction

The number of sequences used in for the prediction is rather limited, since we want to have an independent set. Nevertheless, the number of variables we could use in the prediction is big: Every amino acid has properties such as acidity, charge, polarity or size. In addition we can use a high number of incident amino acids to make our prediction better. The combination of amino acids in pairs or more may also have an impact on the quality of the prediction.

All in all this gives us a huge choice of variables we could use to predict the secondary structure. Since the calculations would take too long, if we would use all possible variables, we have to decide, which variables we want to use for the prediction.

It seems quite obvious to use the variables that contribute most to a good solution and omit those, which have only a small influence on the final prediction.

3.1 Amino Acids and their Properties

Amino acids are the building blocks of proteins. The tertiary structure, and thus also the secondary structure of a protein depends only on the sequence of amino acids from which it is built [8]. This determination results from various properties of the amino acids.

Every amino acid has characteristic physical and chemical properties, which result in a very distinct behavior of the amino acid. This different behaviors arise from the side chains, each amino acid has. The side chains of amino acids are the parts where most of the chemical interactions with other parts of the protein take place.

Information on amino acids, their structure and their chemical and physical properties are widely available on the WWW. An introduction can be found at [14], information on physical and chemical properties and their values at [23], [21], [15], [24] or [18].

Physical properties can be mass, size or charge. Chemical properties can be polarity, hydrophobicity, pH or the existence of aromatic groups in a side chain. According to these properties the amino acids can be split up into different groups, which intersect each other. An overview of this groups and the amino acids contained can be seen in Fig. 4. (Fig. 3).
3.2 Best Basis

The "best basis" problem is to find $k$ variables which minimize the error in the predictions. The problem itself is really difficult, in fact it is NP-complete. The solution of this problem is practically impossible since it would need $O\left(\binom{n}{k} k^3\right)$ operations. We have to check all possibilities to choose $k$ out of all $n$ variables and then solve the corresponding least squares problem to get the error for this set of variables.

This approach results in a much too high number of calculations for our goal, so we will have to use approximate algorithms.

3.2.1 Approximation: Stepwise Regression

One method to find an approximation is stepwise regression. This greedy algorithm chooses in each step the optimal variable to come to the final set:

We start with an empty basis and include one independent variable at a time. We choose the variable which reduces the error by the most.

Since the best basis problem does not allow a continuous approximation, this technique cannot guarantee global optimality.
3.2.2 Other Heuristics

There are other heuristics that find good approximations for the best basis problem. One group of them searches, starting from a random set, for better neighbor sets and finds in this way a local optimum. If this procedure is repeated several times, there is a good chance to find a appropriate variable-set which fits our needs. This approach is in general much better then stepwise regression.

Two approaches to find a local optimum are steepest descent and early abort:

**Steepest Descent** searches all neighbors. The next variable set will be the one that made the biggest step towards the optimum. A local optimum is found in relatively few steps, but each step is computational intense.

**Early Abort** searches the neighbors only until something better is found. This is repeated until no better set is found. The computation for each step is faster, but there are used more steps to find the local optimum.

Both methods are faster if you remember which computations you have already done. This computations will not be redone and thus it saves time.

3.3 myBestBasis: A DARWIN Implementation

To get the best basis in a least squares sense, I wrote a program in DARWIN [1].

3.3.1 The Features

In the first implementation, the features are read from a file *features* this file offers functions to return the values and also a description of the source of the values.

In the *myBestBasis()*-main program, the values will be normalized to $\mathcal{N}(0, 1)$ to give them all the same weight. There is also assigned a name for each feature. For unknown amino acids, which are X in DARWIN, the average value is taken. This normalization and the setting X to the average value is done in the function *norm()* also located in *BestBasis.drw*.

The following features are used:

- Hydrophobicity
• Molecular weight
• Isoelectric point (pH)
• Accessible surface area
• Solubility †
• Chou-Fasman parameter for alpha-helices
• Chou-Fasman parameter for beta-strands
• Chou-Fasman parameter for turns

†: Since three values of solubility are not numeric but 'very high', solubility was not used in first tests.

To use also the feature "Solubility", several values where tested for the use instead of 'very high'. The best results where finally achieved by setting this values to 0. Even if this values have not a strict physical meaning, the fact that the error was the smallest, justifies this choice for further calculations.

The values chosen for the features and their sources are described in Appendix A.

3.3.2 The Calculation of the Best Basis

After initialization of the database (the CB513-database described above) the features are read in and normalized.

To calculate the best basis a matrix A and a vector b are constructed.

A contains the numerical values of every feature selected for each base in the sliding window. A is calculated only once and is the same for the prediction of helices, sheets and turns. Its dimension is $n \times m$, where $n$ is the size of the sliding window times the number of features selected. $m$ is the number of amino acids used for the computation of the best basis, which is the total number of amino acids in the used sequences minus the border, so that the first and the last sliding window have the full number of bases.

The vector b is calculated for helix sheet and turn separately. It contains 0 or 1, dependent if the according structural element is present at the center base or not.

The actual calculation of the best basis is done using DARWIN-functions:
**WeightObservations**(A,b,w) prepares the matrices and vector in such a way, that they can be used for further calculations. w is a vector that contains weights, i.e. if there is more than one equal observation, you can weight it more.

Since there are much more 0’s than 1’s, the trivial solution to always predict "no secondary structure" would give us very good results. This is not the solution we want, we want a solution where we predict "structure" and "no structure" with about the same quality. To achieve this goal we give the relatively rare cases of "structure" more weight. We do this by setting the values of w higher, if there is a 1 in the vector b and lower if there is a 0. The weights for the 0s and 1s have the the same relations like the numbers of 0s and 1s in each of the three kinds of structures.

The results of **WeightObservations**(A,b,w) are AtA: matrix (numeric), btA: array (numeric) and btB: numeric. This results are then further used in the DARWIN-function **SvdBestBasis**().

**SvdBestBasis**(AtA,btA,btB,names,k, o, try) finds the best set of k variables to do a least squares fit. It is a heuristic, not an exact algorithm and its accuracy depends on how many tries are performed. One try is the start with a random set and the finding of the local minimum. If this step is performed more than one time, one can see, if this leads always to the same result and, if not, which of the found local minima has the lowest value for the quadratic norm.

The function returns a SvdResult data structure.

### 3.4 The Results

The output is directed to a file. This file contains information on the actual test and the results of **SvdBestBases**(). The information on the test is mainly the values of the parameters and the features used for the determination of the best bases.

#### 3.4.1 Parameters

To see if the sets have the tendency towards a certain best set, the number of sequences was raised in different steps to the total number of sequences available in the training set. It seems that once, enough sequences are used, the set does not change significantly. The total number of 269 sequences in the training set seems to be sufficiently large with approximately 40000 bases.
The solubility values of three amino acids (Cysteine, Lysine and Threonine) are indicated in the sources as 'high' or 'very high' and not as numerical values. This makes it difficult to use them in numerical computations. Proline has the highest numerical value of the amino acids that are indicated, so I tested it with the values of \(1 \times \text{proline}, 2 \times \text{proline}\) and \(10 \times \text{proline}\). The results, where \(1 \times \text{proline}\) was used, had the smallest quadratic norm of the residuals. Because of this result the values of these amino acids were set to the value of \(0.5 \times \text{proline}, 0.1 \times \text{proline}\) and finally to 0. In the case where 0 was used, the solubility had the biggest influence and the quadratic norm of the error was reduced by the most. In further calculations the values of the three 'very high'-soluble proteins were set to 0. This choice has not really a physical background, but since the results are the best in this way it is the accurate choice.

The SvdBestBasis()-parameter \(\text{try}\) has a default value of 15. This means the search for a minimum will be started 15 times with different start sets and then the best set is given as output. Several values of \(\text{try}\) where tested to see, if the resulting sets and the resulting quadratic norm of the error change. There where cases, where this results changed between 10 and 15 as values for \(\text{try}\). To improve the quality of the results, the value of \(\text{try}\) was set to 20 for further calculations.

Window size is another parameter that can be varied in the calculation of the best basis. The window size takes into account which amino acids are considered for the calculations. The bigger the window size is the bigger is the set of variables to choose from. Since we take only into account the short range interactions in the polypeptide chains and we ignore the long-range interactions, the window size is not taken too big. The quadratic norm decreases if a bigger window is taken, but the decrease is not very significant for window sizes greater than nine. For window sizes greater than nine, the quadratic norm decreases only for sheets. This is a result of the chemical nature of the \(\beta\)-strands, which are connected with other strands further away (in the amino acid sequence) to sheets. The decrease even for sheets becomes very small (approximately 3%) for a window size-change from 9 to 11. The window size which gives relevant contributions depends also from the chosen feature. In further calculations, i.e. the Nearest Neighbors, each feature will have its own window size.

### 3.4.2 The Best Basis

After all the tests, we have to decide, which variables should be kept for further calculations. The results don't show a clear set of features and their
window sizes, but show possibilities for sets which can be used in the Nearest Neighbor Tree.

**The prediction of helices** seems to depend on a high degree from the Chou-Fassman parameter for helices, the solubility and the charge. This three features decrease the quadratic norm the most and these features have also the highest solutions, which indicate strong structure formers or breakers.

**The prediction of sheets** depends also on the Chou-Fassman parameter for helices, further it depends also on the Chou-Fassman parameter for turns and on the solubility. These three features had the main impact on the quadratic norm. Furthermore the features weight and surface area had high values, which indicate a strong tendency to form or break a structure.

**The prediction of Turns** depends, from the quadratic norm decrease, mostly from solubility, hydrophobicity and the Chou-Fassman parameters for sheets. Like for the prediction of sheets, the values for the features weight and surface area are very high.

### 3.4.3 How to continue?

The next step in the secondary structure prediction will be the implementation of the nearest neighbor tree. The features which are the most significant in a least squares sense (found with best basis) should give the best results for the tree.

Since there is not a really clear result, which features should be used, more tests will be made and then, this results will be compared to the results of the best basis. The question will be if the nearest neighbor tree works best with the features that had the biggest norm decrease or with those that had the highest positive or negative values, indicating *structure formers* or *structure breakers*.

Best Basis, a least squares method, and Nearest Neighbors are two different methods to solve the problem. While Best Basis tries to minimize the error for a training set and then predicts the structures with the found parameters, Nearest Neighbors finds clusters with the training set and then classifies the new structures. It is therefore also possible, that there will be different results for the quality of features.
4 Nearest Neighbors

With the calculation of the best basis, we have now an idea, which variables are relevant for the calculation of the secondary structure. The prediction could be done by using the parameters found for this variables.

An other possibility to predict the secondary structure is the method of the nearest neighbors. In this method we store all data points of known helices, sheets, turns and structure free bases. For a new data point we search in this database the closest data points, called 'Nearest Neighbors', and we decide according to the values of these neighbors, which structural type occurs at this point.

The main task in this method is to build an appropriate data structure, which fits our needs to search a set of neighbors in a very big database.

There are several possible searching algorithms and data structures for nearest neighbors problems:

**Sequential Search** means to store all points in an array and then, to search, test every entry if it is a neighbor.

This technique is far to costly to apply it to a data set of the size we used in this project.

**Range Searching** means finding all records in a database that satisfy specified range restrictions on a specified set of attributes. In general, we assume that we have a set of records with certain attributes that take values from some ordered sets. It is an important problem in practical applications, but its solution is difficult.

**Quad Trees** in $k$ dimensions split the searching space in $2^k$ subspaces at each node. The point stored at each node splits the remaining data points according to all possible subspaces. A binary search tree is a special case for quad trees, where $k = 1$.

Quad Trees are not practical for the prediction problem since quad trees for $n$ dimensions have internal nodes with $2^n$ descendants. This is totally impractical for large $n$.

**k-d Trees** (*k*-dimensional trees) are a generalization of binary search trees: simply cycle through the dimensions while going down the tree, i.e.
for the root use the first dimension for the decision about subtrees, for the second level use the second dimension etc.

For very high dimensions as used in prediction, this algorithm has to be modified to be useful. If we would cycle over all dimensions we would normally exhaust the tree before we even have tested some of the dimensions.

The solution to this problem, that will be used for the Nearest Neighbor calculations is based upon using a linear combination of dimensions at each level (instead of using a single dimension at a time).

4.1 NearestNeighbors: A DARWIN Implementation

I implemented a Nearest Neighbor algorithm, which is based on k-d trees. The algorithm follows closely the proposed one mentioned in the Wissenschaftliches Rechnen - Script which can be found in [25]. Some changes to improve the efficiency where made, since we have to deal with a big number of calculations. The program is divided in to several procedures to enhance the overview and to allow recursive calls of the procedures. In the following subchapters each of this procedures will be described.

4.1.1 myNN

The procedure myNN coordinates the building of the tree. It has not much own functionality but calls the other procedures in order to build the desired tree. In myNN the window sizes of the different features can be changed or set to 0 if this feature is not considered. Further the dataSize can be changed. This allows to use only a part of the data set to build the tree, which was very useful to test the program, since it’s easier to get an overview over a smaller tree. Further the waiting time for the tree to be built is smaller.

4.1.2 init

This procedure initializes the program i.e. it sets all the paths needed for the further calculation, reads in the features and the training-database, makes a description of the features and calculates the frequencies of the different amino acids in the training set. This procedure is called only once in the procedure myNN. It has only to be called again, if more features are added to the features-file or if values in this file are changed.
4.1.3 myMap

The training set contains only information on the amino acids and the secondary structure.

To use this information the values of the chosen features have to be mapped to each entry. The procedure myMap therefore maps the numerical values to each amino acid and builds the matrix C.val in which each data point is represented as a vector of values.

In order to store the secondary structure information, a vector, v.val, is constructed. Each value represents the secondary structure at a given data point. This can either be h for helix, s for sheet or t for turn.

Additional the dimensions of C.val are calculated: m_dim and n_dim.

The names of the chosen features and their position in the sliding window are assigned to the variable ft_names.

4.1.4 myDist

This short procedure calculates the distance between to data points. The normalization of the points is done in the function norm.

4.1.5 norm

The different dimensions (features) represent different magnitudes and different scales. If we want to compare them or calculate distances, each dimension should have the same weight. Instead of weighting we can also normalize each dimension so that the individual variables appear to be distributed as $\mathcal{N}(0, 1)$.

The different amino acids in the training set do not appear in the same frequency, thus the frequency must been taken into account, when we want to norm the values of the features.

This normalization and weighting is done in the procedure norm. In the initialization norm is called for each feature that is read in.

4.1.6 myTreeBuild

To build the nearest neighbors tree, we use this procedure, which calls itself recursively. In each recursion the given set is split into two subsets. The hyperplane that separates the two sets should do this in the direction where the values have the largest variance.
The construction of the root node and the separation of the set into two subsets can be done in five steps. This is repeated in every step of the recursion until the stop criterion is reached.

- 1. Compute the covariance matrix of all points.
- 2. Compute the eigenvector (called \( \alpha \)) with the largest eigenvalue.
- 3. Compute the median of \( \alpha x \) and call it \( \alpha_0 \). (For all vectors \( x \) in \( C_{\text{val}} \)).
- 4. Split all points in two sets (\( \alpha x < \alpha_0 \) or \( \alpha x > \alpha_0 \)).
- 5. Build the two descendants recursively.

The recursion terminates when the given set is smaller than a certain threshold. All the data vectors in this set are then kept in a "bucket" which has to be searched with linear search when we search for neighbors. The threshold is chosen the same like \( n \cdot \text{dim} \). In this way we avoid that the \( \text{Cov} \)-matrices are singular. (A \( \text{Cov} \)-matrix of dimension \( n \times n \) with fewer than \( n \) data points will be singular.)

### 4.1.7 NNSearch

Given a point (by its \( n \)-dimensional coordinates), this function searches the tree and returns a set of neighbors which are not farther away than \( \epsilon \) from the searched point.

This is done by searching the tree and exclude the branches, where all leaves contain points, that are further away than \( \epsilon \). At the end the remaining buckets are searched by looking at every entry and by comparing the distance to the initial point with \( \epsilon \).

The procedure returns a set of indices, which indicate the entries in the matrix \( C_{\text{val}} \) and their distance to the given point.

NNSearch is called by the procedure \( \text{Search}_k\text{NN} \) to search the \( k \) nearest neighbors.

### 4.1.8 Search\(_k\)NN

To search the \( k \) nearest neighbors this procedure calls \( \text{NNSearch} \) with a growing \( \epsilon \).

From a start value \( \text{hhigh} \), the value of \( \epsilon \) is multiplied in each round by a certain factor until the found set of neighbors is bigger than \( k \). This set
is then sorted by distance to the initial point. The \( k \) closest points are returned.

The efficiency depends strongly on the chosen start value and on the factor it is multiplied with each round. Good values can be found if one looks at the distances a certain set has from a random point and how they are distributed.

4.1.9 \textbf{myPredict}

\textit{myPredict} is a small procedure which calls \textit{Search\_kNN} and looks up the secondary structures in \texttt{v\_val} at the resulting points. It returns the structure that appears in the majority of the neighbors of the searched point. If two structures (or \textit{no structure}) have the same probability, the structure of the prediction made before is returned. This is of course only done, if this is a structure with the biggest probability. In other cases \textit{no structure} is returned.

Additional to the prediction of the structure, the probability of this structure is also returned (i.e. the number of neighbors with this structure divided by the total number of neighbors).
5 Tests of the Predictions

To get the quality of the prediction, we need to test it with an independent set. This independent set, the test set, was generated already in the beginning, at the same time, the training set was generated. It contains about the same amount of structures like the training set and is independent from it.

The quality of the prediction has to be measured in a way, that it is comparable with other methods of structure prediction. The two measures I used are SOV and $Q_3$, described in the subsequent chapters.

From the quality of prediction, we can now determine, which set of features fits best and how many neighbors have to be chosen for the best results in the prediction. The question, if the best prediction is achieved by using the variables resulting from the best basis algorithm, can be solved now.

5.1 Accuracy Tests for the Prediction

The results we obtain from our tests have to be compared to other prediction methods. To do this, we use two methods that are widely used. Even if they have different definitions and meanings, they are both widespread through the community and they are considered to be the measures of choice.

5.1.1 $Q_3$

$Q_{index}$: $(Q_{helix}, Q_{strand}, Q_{nostructure}, Q_3)$ gives percentage of residues predicted correctly as helix, strand, "no structure" or for all three conformational states. The definition of $Q_{index}$ is as follows.

For a single conformational state:

$$ Q_i = \frac{\text{number of residues correctly predicted in state } i}{\text{number of residues observed in state } i} \times 100 $$

where i is either helix, strand or nostruct.

For all three states:

$$ Q_3 = \frac{\text{number of residues correctly predicted}}{\text{number of all residues}} \times 100 $$
In first tests I also calculated $Q_{\text{turn}}$, but this state was predicted very poor. In later calculations and in the accuracy calculation with $SOV$, turns where regarded as nostructure and the state with the second high probability was taken.

5.1.2 SOV

The traditionally used $Q_3$ measure that gives an overall number of residues predicted correctly can be misleading.

It seems that measures concentrating on how well secondary structure elements are predicted instead of individual residues better reflect the nature of three-dimensional protein structure. As an effort to make evaluation of secondary structure prediction more structurally meaningful the segment overlap measure (SOV) was defined [28]. The definition is as follows:

Segment OVerlap quantity measure for a single conformational state:

$$SOV(i) = \frac{1}{N(i)} \sum_{N(i)} \frac{MINOV(S1; S2) + \delta(S1; S2)}{MAXOV(S1; S2)} \times LEN(S1)$$

$S1$ and $S2$ are the observed and predicted secondary structure segments (in state i, which can be either H, E or C);

$LEN(S1)$ is the number of residues in segment $S1$;

$MINOV(S1; S2)$ is the length of actual overlap of $S1$ and $S2$, i.e. the extent for which both segments have residues in state i, for example H;

$MAXOV(S1; S2)$ is the length of the total extent for which either of the segments $S1$ or $S2$ has a residue in state i;

$\delta(S1; S2)$ is the integer value defined as being equal to the MIN{[(MAXOV(S1; S2) - MINOV(S1; S2)); MINOV(S1; S2); INT(LEN(S1)/2); INT(LEN(S2)/2)]}

THE SUM is taken over $S$, all the pairs of segments $S1; S2$, where $S1$ and $S2$ have at least one residue in state i in common;
$N(i)$ is the number of residues in state $i$ defined as follows:

$$N(i) = \sum_{S(i)} LEN(S1) + \sum_{S'(i)} LEN(S1)$$

Two sums are taken over $S$ and $S'$.

$S(i)$ is the number of all the pairs of segments $S1;S2$, where $S1$ and $S2$ have at least one residue in state $i$ in common.

$S'(i)$ is the number of segments $S1$ that do not produce any segment pair.

**Segment Overlap quantity measure for all three states:**

$$SOV(i) = \frac{1}{N} \sum_i \sum_{S(i)} \frac{MINOV(S1;S2) + \delta(S1;S2)}{MAXOV(S1;S2)} \times LEN(S1)$$

where the normalization value $N$ is a sum of $N(i)$ over all three conformational states ($i = \text{HELIX, STRAND, COIL}$):

$$N = \sum_i N(i)$$

A **COIL** corresponds to a random structure and is equivalent to no struct in the predictions.

An online interface to test predictions is available under [27] and structures until the length of 2000 residues can be tested there. To test bigger amounts of structure at once, I took the C program available from the same place and changed the maximum length of structures allowed. This program implements the above described measures of accuracy.

### 5.2 The Results

The nearest neighbors tree can be built over different sets of features and the window size of each feature can be varied. Each of this feature/window sizes combination results in a different tree with a different prediction of the secondary structure.

I tested several trees to see how good they predict secondary structures in the test set. The feature/window size combinations used to build the
trees where chosen according to the results of the best bases calculation made earlier.

The number of neighbors used to make the prediction also affects the results. To see how strong the number of neighbors change the accuracy of the prediction, different numbers of neighbors where used.

The following sections show the accuracy of the different trees with the corresponding number of neighbors. Each section is dedicated to one tree i.e. to one feature/window size combination. The trees are identified by the size of the windows for all features. The features are ordered and so the indication of the window size is enough information to distinguish one tree from another.

The order of the features is the following:

- Hydrophobicity
- Molecular weight
- Charge : pH
- Accessible surface area
- Solubility g/100g
- Chou-Fasman parameter for helices
- Chou-Fasman parameter for sheets
- Chou-Fasman parameter for turns

The meaning of (3-0-3-0-5-5-3-3) is therefore: Window size 3 for hydrophobicity, window size 3 for charge, window size 5 for solubility, window size 5 for the Chou-Fasman parameter for helices, window size 3 for the Chou-Fasman parameter for sheets and window size 3 for the Chou-Fasman parameter for turns.

The number of neighbors to be chosen to get the best results is rather surprising. Values between 1 and 10 where expected, since it was mentioned so in the *Wissenschaftliches Rechnen-Skript* [25]. This was also the reason why I first tested values in this order of magnitude. The best results where achieved with the highest values, so I extended the tests to higher numbers of neighbors. The best results seem to appear in the magnitude of 50 to several hundred neighbors, depending on which set of feature/window size the nearest neighbor tree was built. This means that there is a maximum in the total accuracy score at this size of the neighbor set.
5.2.1 (3-0-3-0-5-5-3-3)

This feature set was the first one tested. Its feature/window size combination was derived from the results of the best basis tests. The window sizes are rather small, since we want to take into account only the short range interactions of the residues. Furthermore the total number of 22 variables used to build the tree, allowed a faster building of the tree and also a faster prediction of the approx. 40000 residues in the test set.

With this choice of features, the most important contributions to the norm decrease are included.

A summary of the prediction accuracy can bee seen in Fig. 5 and Fig. 6.

![Figure 5: Q3 Analysis of the (3-0-3-0-5-5-3-3)-test](image)

The overall prediction accuracy reached values up to 55.2% measured in Q3, respectively 40.8% measured in SOV. This values where reached when 100 - 250 neighbors where chosen.
Figure 6: SOV Analysis of the (3-0-3-0-5-5-3-3)-test

As you can see in Fig. 5 and Fig. 6 there is a huge difference in the prediction accuracy of helices and coils compared to the accuracy in the prediction of strands.

The prediction of strands is poor with its accuracies of under 33% in both measures ($Q_3$ and $SOV$).

Helices had much better prediction accuracies reaching almost 70% correctly predicted residues ($Q_3$) or 42% correctly predicted segments ($SOV$).

The prediction of coils reached similar values like the prediction of helices. With an amount of maximal 60.7% correctly predicted residues and 45.2% correctly predicted segments, the prediction accuracies for coils are even better than those of helices.
The $Q_3$ total-score reached its top, when 50 neighbors were used.

Each structure has a different behavior of the prediction accuracy (see Fig. 5):

The prediction of helices gets better, the more neighbors are used. In the tested range of neighbors, there could not be determined, where the maximum of prediction accuracy is, since, the overall score was not increasing anymore, when the number of neighbors was increased.

The prediction of sheets has its maximum at a number of neighbors between 20 and 50. The prediction accuracy at this values is about the same as the prediction accuracy, when only 1 neighbor is used. If higher values are taken, the (already poor) prediction accuracy gets smaller and smaller.

Coils are predicted with the highest accuracy if the set of neighbors has the size 5. When the number of neighbors is changed to higher values, the accuracy gets worse. The values do not decrease as much as in the case of sheets, but there are still significant differences if we compare the value of 5 neighbors (60.7%) and the one of 250 neighbors (56.8%).

The $SOV$-score (see Fig. 6) had its best values at a very high number of neighbors (100-250).

The behavior of the accuracy for each kind of structure is quite similar to the behavior, when we measured with $Q_3$:

The accuracy for helices grows, as more neighbors are used.

The accuracy for strands reaches its highest values when about 50 neighbors are used.

The prediction of coils reached its best values at a higher number of neighbors as in the $Q_3$ measurements; at values between 50 and 150 neighbors.

5.2.2 (5-1-5-1-7-7-5-5)

This was the second set of features, I used to build a tree and to predict the secondary structures. Similar like (3-0-3-0-5-5-3-3)-set, this one takes into account the biggest contributions of the norm decrease found in with the best bases algorithm. The window sizes where increased to include more variables.

As expected, the results were better than those before. The figures Fig. 7 and Fig. 8 show an overview over several tests made with different sizes of the neighbor sets. The cost of this augmentation of the prediction

30
accuracy was the longer computational time used to build the tree and to calculate the prediction.

The number of variables used for this tree is 36.

Figure 7: $Q_3$ Analysis of the (5-1-5-1-7-7-5-5)-test

The highest prediction accuracies reach with this feature/window size combination 57.2% in the $Q_3$-measure and 48.5% in the $SOV$-measure.

The highest values for the overall prediction accuracy are reached, when 50 - 100 neighbors are used in the $Q_3$-measure and when 500 neighbors are used in the $SOV$-measure.

Like in the previous tree, the behavior of the prediction accuracy depends on the structure we predict.

The prediction of coils gets better, the more neighbors are used. There could not be determined a maximum for the prediction accuracy in the range of neighbors that where tested (up to 1000).
Figure 8: \textit{SOV} Analysis of the (5-1-5-1-7-7-5-5)-test

The $Q_3$-measures reached values up to 80.6\% when 1000 neighbors were used. The \textit{SOV}-measure reached values up to 60.5\% with this number of neighbors.

The prediction of sheets reached its maximum accuracy (36.1\%) with 20 neighbors in $Q_3$ and in \textit{SOV} (35.7\%) also with 20 neighbors.

When the set of neighbors gets bigger, the prediction accuracy gets smaller and smaller and has under 30\% accuracy for sets with 500 or more neighbors.

Coils are predicted with the highest accuracy if 5 neighbors are chosen for the $Q_3$-measure (61.1\%) and if 150 - 250 neighbors are used in the \textit{SOV}-measure (49\%).

The overall behavior of this tree is very similar like the one tested before.
5.2.3 (1-1-7-1-9-9-9-5-9)

The prediction of sheets was very poor with the previous trees. To get a better result for sheets, the window sizes of the features which gave a higher norm decrease in the error for sheets, was increased.

The most relevant change to the previous set was the change of the window size of the Chou-Fasman parameter for turns to a value of 9. On the other hand, the window size for hydrophobicity was changed from 5 to 1. This was done, because hydrophobicity is not so relevant in the best basis analysis of sheets. Furthermore this avoided that the tree grows to big, which affects the computational time.

The total number of variables in this tree is 42.

A summary of the resulting accuracies from this tree can be seen in Fig. 9 and Fig. 10.

![Graph showing analysis of (1-1-7-1-9-9-5-9) test](image)

**Figure 9: Q₃ Analysis of the (1-1-7-1-9-9-5-9)-test**
Figure 10: SOV Analysis of the (1-1-7-1-9-9-5-9)-test

The prediction with this tree was not as good as in the previous trees. Even if the number of variables used to build this tree was higher than the number of variables used to build the previous trees, the results, specially for the sheets where not better.

The overall accuracy for this tree reached 56.2% in $Q_3$-measure when 50 neighbors where used. The highest score (47.5%) in SOV-measure was reached when 250 neighbors where used.

The prediction accuracy for helices gets better, the more neighbors we use. In the $Q_3$-measure, we reach values up to 83.5% when 750 neighbors are used. This is the highest number of neighbors used for this tree, since the total score, which is more relevant does not improve further.

In the SOV-measure we get an accuracy of 62.2% when 750 neighbors are used.
The prediction accuracy for strands did not improve with this feature/window size combination. The maximal value in the $Q_3$-measure of 34.7% was reached, when 20 neighbors where used to predict the structure.

In $SOV$ the maximum was also reached, when 20 neighbors where used. The maximum value with this measure is 33.3%.

For the prediction of coils, the accuracies are in similar ranges like in the previous sets. In the $Q_3$-measure, the maximum values is reached when 5 neighbors are used (57.9%).

In the $SOV$-measure the maximum value of 45.2% is reached, when 150 neighbors are used to predict the structure.

### 5.2.4 (5-5-5-5-5-5-5)

The (5-5-5-5-5-5-5)-set was chosen to get a good reference. All the windows have the same size i.e. all features have the same importance in this set.

The number of variables in this set is 40. This leads to a high computation time, comparable to the computation time in the (1-1-7-1-9-9-5-9)-set.

A summary of the resulting accuracies from this tree can be seen in Fig. 11 and Fig. 12.

The overall prediction accuracy for this feature/window size combination reached higher values than the previous trees. The values are still in about the same range.

Measured with $Q_3$, the maximal accuracy of 57.3% was reached, when 50 neighbors where used for the calculation of the secondary structure.

In the $SOV$-measure the value of 48.6% was reached with 500 neighbors. Since no tests where made in the range between 500 and 2000 neighbors there may be higher values in this range.

Even with a high number of neighbors tested, the accuracy for the prediction of helices still increases. It reached values up to 84.9% in $Q_3$ and 64% in $SOV$.

In the prediction of sheets, the best results in the $Q_3$-measure are reached when 50 neighbors are used. The prediction accuracy is 36.4% correctly predicted sheet-residues.

Measured with the $SOV$-score the highest values of 36.5% correctly predicted segments is also reached, when 50 neighbors are used.
Figure 11: $Q_3$ Analysis of the (5-5-5-5-5-5-5-5)-test

The prediction of coils reached its highest accuracy in the $Q_3$-measure, when 5 neighbors are used. It reaches a score of 61.7%.

The highest $SOV$-score is reached at a higher number of neighbors: 48% prediction accuracy is reached with 500 neighbors. Like in the overall prediction accuracy, higher values can be expected in the range between 500 and 2000 neighbors, where no tests were made.

Even if this feature set has no strong connection to the best basis analysis done earlier, the results are in a good range compared to the other feature sets. Even though, most of the largest contributions to the norm decrease are included in this feature/window size set.

There were no tests in the range of 500 and 2000 neighbors since the computation with this amount of neighbors uses a lot of time. Further, compared to the other trees, the high values were reached at higher
numbers of neighbors than expected.

5.2.5 (0-0-5-0-7-7-0-7)

This set is with its 26 variable rather small. Like in the (1-1-7-1-9-9-5-9)-set, I tried to get a better prediction for the sheets. Additionally this set has left out 4 of the features totally, so the prediction of the secondary structure elements is based only on 4 different properties of the amino acids.

An overview of the prediction accuracies made with this feature/window size combination can be seen in Fig. 13 and Fig. 14.

The total prediction accuracies for this tree where, taken into account the small number of variables, satisfying. Measured with the $Q_2$-score it reached 55.7% prediction accuracy when 100 neighbors where used.
In the $SOV$-measure, the accuracy reached a maximum of 46.8% if 200 neighbors were used.

The prediction of helices gets better, the more neighbors are used. For this feature/window size set a maximal number of 250 neighbors was tested. The maximal values reached are therefore smaller as in the other tests.

Measured with the $Q_3$-score the prediction accuracy for helices reached 76%.

Measured with the $SOV$-score the prediction of helix-segments reached a maximal accuracy of 56.9%.

Sheets where predicted poor again.

The maximal percentage of correctly predicted residues, measured with $Q_3$, was reached when 1 neighbor was used. The value there is 34.1%.

The best prediction of segments, measured with the $SOV$-score was reached with 20 neighbors (32.8%).

Figure 13: $Q_3$ Analysis of the (0-0-5-0-7-7-0-7)-test
Figure 14: SOV Analysis of the (0-0-5-0-7-7-0-7)-test

The best score in $Q_3$ for the prediction of coils is reached, when 10 neighbors are used: 56.1% of the residues are correctly predicted.

The prediction accuracy for segments (measured with the SOV-score) grows, the more neighbors are included. The maximal value of 46.9% was therefore reached when 250 neighbors were used.

5.2.6 (0-0-7-0-9-9-0-9)

The (0-0-7-0-9-9-0-9)-set chooses the same features like the set before. The window sizes of all chosen features was raised by two to get better results. Since the (0-0-5-0-7-7-0-7)-set had a very small set of variables and the results where in a good range, this set was chosen.

The number of variables is with 34 in a acceptable range and its computational time is therefore in similar bounds as the computational time for earlier trees.

39
The results for different sizes of neighbor sets can be seen in the figures Fig. 15 and Fig. 16.

Figure 15: $Q_3$ Analysis of the (0-0-7-0-9-9-0-9)-test

This set had bigger window sizes than the previous one but the results of the prediction accuracy measurements are not better.

The expectation that a set that contains 8 variables more than another leads to better results, was not fulfilled.

The maximal prediction accuracy for residues (measured in $Q_3$) was reached when 50 neighbors were taken into account for the prediction. It reached a value of 55.3% correctly predicted residues.

The $SOV$-measure for segments had it’s maximum when 200 to 250 neighbors were used. The prediction accuracy there was 47.8%.

Since the results for this tree did not show very good results, the size of the neighbor sets tested was limited to a size of 250.
The prediction of helices showed again the usual increase, as more neighbors are taken into account for the prediction.

At a neighbor set size of 250 (the maximal size tested) a $Q_3$-score of 78.5% was observed.

The prediction of segments, measured with the $SOV$-score, reached an accuracy of 61.0% at the neighbor set size of 250.

The prediction accuracies for strands had their maximums in both measures when 20 neighbors were used.

The prediction accuracy measured in $Q_3$ was 33.2% at this neighbor set size.

The $SOV$-score had a value of 33.3%.
For the prediction of coils, the maximal accuracies were reached at different sizes of neighbor sets.

The maximal accuracy for the $Q_3$-score of 57.6% was reached with 5 neighbors.

The maximal accuracy in the SOV-measure (45.8%) was reached with 200 neighbors.

5.2.7 (0-9-0-9-0-9-0-9)
The previous sets were chosen according to the norm decrease. This set is chosen according to the values, actually derived from the best basis analysis: Only features with high values were chosen. High positive values indicate structure formers, high negative values indicate the tendency for the structure to break.

The biggest difference to earlier sets is the big window size for the feature molecular weight and the 0-window size for the features solubility and charge.

The set has a size of 36 variables and is therefore reasonable in time and good comparable to the results of other sets.

A graphical overview of the prediction accuracies can be seen in the figures Fig. 17 and Fig. 18.

The values of the different accuracy measurements for this feature/window size combination show no significant difference to the other trees, where the features and their window sizes where chosen according to the biggest decrease of the norm in the best basis analysis.

One of the most important reasons for this is, that this tree had also a big part of the variables in common with the other trees since the highest values were often achieved with the variables who had also an important norm decrease.

The maximal prediction accuracy in the $Q_3$-measure was reached with 100 neighbors. The value of 56.2% has an average quality compared with other trees.

Measured in the SOV-score, the maximum accuracy was reached when 250 neighbors were used to dedicate the secondary structure. The value for this neighbor set size was 47.6%.
Figure 17: $Q_3$ Analysis of the (0-9-0-9-0-9-0-9-0)-test

For this tree more neighbor sets were tested. The maximal number of neighbors used was 2000. Also with this high number of neighbors, the quality in the prediction of helices still increases.

An accuracy of 83.2% correctly predicted residues (measured with in $Q_3$) was observed for the biggest neighbor set.

The maximal accuracy in SOV was also observed at the biggest neighbor set. It reached a value of 63.2% there.

Again the values for the prediction of strands are low and decrease as more neighbors are used for the prediction.

The $Q_3$-score is best (30.7%) when 1 neighbor is used. It has another high value (27.9%) at a neighbor set of the size 10.

The highest SOV-score is reached with 10 neighbors. It has a value of 28.7% there.
Figure 18: $SOV$ Analysis of the (0-9-0-9-0-9-0-9)-test

The results in the prediction of coils show again a similar behavior of accuracy like the other sets.

The maximal accuracy in the $Q_3$-measure was 59.2% with the usage of 100 neighbors.

The best $SOV$-score of 49.8% was reached at a size of the neighbor set of 500.
6 Conclusion

6.1 Interpretation of the Results

The results never reached prediction accuracies over 60% in the $Q_3$-measure or 50% in the $SOV$-measure (see Fig. 5 to Fig. 18).

Even though, the results show that the method of predicting secondary structure with help of nearest neighbor trees based on features of amino acids works. A random choice of structure at each residue would lead to a prediction accuracy of only 33.3%, since there are 3 possible states of structure to choose from (alpha-helix, beta-sheet or coil). The results of the $SOV$-analysis would be even worse, taken into account that structure elements cover a certain number of residues each.

Both approaches, best basis and nearest neighbors make a prediction about a structure for one residue at the time. This seems to be suboptimal because it is not very likely, that a sheet or helix structure is present at one residue, not present at the next and present again in the third. In this case it would be more likely to have this structure over all three residues or not present at all.

There was no big difference of prediction quality in the sets that differed in the choice of variables. The more variables were used, the better the results were in most cases.

The choice of features which had high values in the best basis analysis had no significant impact on the prediction accuracy. A reason for this is the big correlation of the variables with high solutions and the variables which decreased the error norm the most. This correlation was already observed in best basis.

All resulting trees showed a common behavior of the helix and sheet structures. Whereas the prediction accuracy of helices gets better the more neighbors are used, the prediction accuracy of sheets decreases in the same scale after have reached an optimum.

This behavior may indicate, that the prediction of helices and sheets are somehow connected. Specially if a high number of neighbors is used and the distance to the farthest neighbors increases, this effect seems to become more and more relevant.

In a further implementation this may be taken into account by incorporating the distance to the searched point into the final prediction.
In general, the prediction of sheets is worse than the prediction of helices. This effect is also observed in other prediction methods [11] and can be explained by the fact that sheet formation is determined by more non-local contacts than helix formation.

This method uses only information on the sequence itself. This restricts somewhat the power of the prediction. If structural information on the protein itself, information on its family [16] or other evolutionary information [17] are integrated into the prediction method, the prediction accuracies can be expected to be much higher.

Another way to improve the prediction accuracy would be to perform a multiple sequence alignment and then calculate the prediction with all those sequences [4].

Methods like this and also other methods to predict secondary structure are discussed in the following section and the results of my implementation are compared with them.

Taken into account the described restrictions and the still possible refinements of this method the results are very satisfying. They show the limits of this method but also that a nearest neighbor tree is a appropriate method to predict secondary structure.

6.2 Other Methods of Structure Prediction

To see the results of this implementation of the nearest neighbor tree in a bigger context, this section compares the results with results from other methods used to predict secondary structure of proteins.

A good paper on the accuracy of the different secondary structure prediction methods can be found in [11]. An overview of the different methods is given in [29] or in [19].

Another source to compare predictions is [10], there one can also compete with other projects in the prediction of structures not yet solved.

The prediction methods can be divided into three generations, each generation distinguishable from the other by the amount of information taken into account to build the model.

6.2.1 First Generation

In the predicting methods of this generation, the information is coming from a single residue of a single sequence.
The Chou and Fasman prediction method (CF) dating from 1974 is certainly the best known member of this generation. This method which bases only on statistical information was used to calculate secondary structure by hand. A short description of this method can be found in [26]. A wide range of prediction accuracies (58% - 86%) have been reported. In general, prediction accuracy is better for a protein with a single type of secondary structure (all helix or all strand) than for a mixed type protein. Helices were found to be better predicted than strands and turns.

Another method based on single residue statistics is the GOR1 method. A description can also be found in [26]. The Garnier, Osguthorpe and Robson empirical statistical method of secondary structure prediction is based on the probability of an amino acid of any type being associated with a neighbor of any type at position \( j \), where \( j \) varies from +8 to -8 along the sequence. In [29] the accuracy is claimed to be 57% (\( Q_3 \)).

Methods using explicit rules also belong to this generation but where not particularly successful.

6.2.2 Second Generation

New in this generation was that local interactions are taken into account. The principal improvement of the 2nd generation profited from the growth of experimental information about protein structure. This data enabled to use the information contained in consecutive segments of residues. Typically 11-21 adjacent residues are taken from a protein and statistics are compiled to evaluate how likely the residue central in that segment is to be in a particular secondary structure state [11].

The range of methods in this generation includes statistical methods, nearest neighbor algorithms or neural-networks based prediction.

As an example of this generation the method of Yi and Lander used a combination of nearest neighbors and neuronal-networks to predict the secondary structure. The accuracy of this method is described in [29] to be 68%. A description of an improved version of this method can be found at [30].

My implementation of the nearest neighbor tree can be seen as a member of the second generation. It takes into account only information on the
sequence. The window sizes I used are smaller than the ones described above, which has impact on the prediction accuracy.

6.2.3 Third Generation

To get better results in the prediction, information coming from homologous sequences is incorporated. The availability of good MSA (multiple sequence alignment) algorithms give this methods a relevant advantage. This methods are currently state of the art. They often combine different approaches like neuronal networks [22], nearest neighbors, pattern recognition [12] and statistical information and are very complex in their implementation.

A good example from this family is PHD, a method of Rost and Sander. This method combines evolutionary information and neural networks to predict protein secondary structure [29]. This method states an accuracy of over 70% (Q3). Later versions of this methods reach prediction accuracies of 74%.

Other methods of this newest generation also reach values over 70%. [29] lists some of them: Jnet from Cuff and Barton and PSIPRED from Jones (75.7% accuracy) are besides the PHD method the best known methods nowadays.

6.3 The End

The aim of this project was to implement a combination of a best basis algorithm and a nearest neighbors tree to predict the secondary structure from the amino acid sequence of a protein.

Prior of any calculation I had to create a database, which had good properties for the training and testing of the prediction method. By finding and integrating the CB513-data set, good independent sets could be used for this purpose.

For the implementation of the best basis algorithm, DARWIN provided already a lot of functions usable for the calculation. For this program it was therefore important to understand the provided functions and to put them together in a sensible way.

The implementation of the nearest neighbor tree followed closely the description in [25] and was refined to reduce the computational time in the search function.
At the end, predictions with the test set were made. The predictions and the correct structure were compared and their accuracies were calculated with a tool provided by [27].

The results show a good capability to predict secondary structures. The prediction method used belongs to the second generation of structure prediction methods and has therefore not the high prediction accuracies of the modern methods which take into account much more information on the protein itself.

The programs themselves could be refined in various ways to improve the results of the prediction. Various improvements are mentioned earlier, in the comments to the single parts of the project. They could also be used as building blocks of a third generation prediction method if they are integrated properly.

Personally I got a good overview of the wide field of secondary structure prediction. The work showed also the limitations that have to be accepted if the time resources prevent a deeper preoccupation with possible improvements. The extent of a diploma thesis implies the concentration of the main goals.

I am satisfied with the results and also with the skills acquired for future projects.
References


52
A Values of the Features

A.1 Hydrophobicity

The values for hydrophobicity are taken from [23].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>-0.300</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>-1.100</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>-0.200</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>-1.400</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>6.300</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>-0.200</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>0.000</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>1.200</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>-1.300</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>4.300</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>6.600</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>-3.600</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>2.500</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>7.500</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>2.200</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>-0.600</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>-2.200</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>7.900</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>7.100</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>5.900</td>
</tr>
</tbody>
</table>
A.2 Molecular Weight

The weights of the amino acids are provided by DARWIN [1]. They are as follows:

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>89.079</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>174.188</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>132.104</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>133.089</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>121.144</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>146.131</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>147.116</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>75.052</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>155.142</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>131.160</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>131.160</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>146.174</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>149.198</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>165.177</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>115.117</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>105.078</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>119.105</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>204.213</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>181.170</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>117.113</td>
</tr>
</tbody>
</table>
A.3 Charge

pH: Isoelectric Point (charge). The values are taken from [21].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>6.00</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>11.15</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>5.41</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>2.77</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>5.02</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>5.65</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>3.22</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>5.97</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>7.47</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>5.94</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>5.98</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>9.59</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>5.74</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>5.48</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>6.30</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>5.68</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>5.64</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>5.89</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>5.66</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>5.96</td>
</tr>
</tbody>
</table>
A.4 Surface

Accessible surface area for the residue X in the tripeptide G-X-G. The source of this numbers is [15].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>115</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>225</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>160</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>150</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>135</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>180</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>190</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>75</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>195</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>175</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>170</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>200</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>185</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>210</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>145</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>115</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>140</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>255</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>230</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>155</td>
</tr>
</tbody>
</table>
A.5 Solubility

Solubility in g/100g. The values are derived from [24].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>16.65</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>15.00</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>3.53</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>0.778</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>very high</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>2.5</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>0.864</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>24.99</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>4.19</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>4.117</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>2.426</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>very high</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>3.381</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>2.965</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>162.3</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>5.023</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>very high</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>1.136</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>0.0453</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>8.85</td>
</tr>
</tbody>
</table>
A.6 Chou-Fasman Parameter for Alpha-Helices

The values for this feature and a general description can be found at [15].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>142</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>98</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>67</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>70</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>111</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>151</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>57</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>100</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>108</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>121</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>114</td>
</tr>
<tr>
<td>M</td>
<td>Methionine</td>
<td>145</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>113</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>57</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>77</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>83</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>108</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>69</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>106</td>
</tr>
</tbody>
</table>
A.7 Chou-Fasman Parameter for Beta-Strands

The values for this feature and a general description can be found at [15].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>83</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>93</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>89</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>54</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>119</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>110</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>37</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>75</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>87</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>160</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>130</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>74</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>105</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>138</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>55</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>75</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>119</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>137</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>147</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>170</td>
</tr>
</tbody>
</table>
A.8 Chou-Fasman Parameter for Turns

The values for this feature and a general description can be found at [15].

<table>
<thead>
<tr>
<th>AA</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
<td>66</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
<td>95</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
<td>156</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic acid</td>
<td>146</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
<td>119</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
<td>98</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic acid</td>
<td>74</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
<td>156</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
<td>95</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
<td>47</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
<td>59</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
<td>101</td>
</tr>
<tr>
<td>M</td>
<td>Methionone</td>
<td>60</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
<td>60</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
<td>152</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
<td>143</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
<td>96</td>
</tr>
<tr>
<td>W</td>
<td>Tryptophan</td>
<td>96</td>
</tr>
<tr>
<td>Y</td>
<td>Tyrosine</td>
<td>114</td>
</tr>
<tr>
<td>V</td>
<td>Valine</td>
<td>50</td>
</tr>
</tbody>
</table>