Data Mining Techniques for Protein Sequence Analysis

Stephen Edward Hamby

Thesis submitted to the University of Nottingham

for the degree of Doctor of Philosophy

December 2009

Abstract

This thesis concerns two areas of bioinformatics related by their role in protein structure and function: protein structure prediction and post translational modification of proteins. The dihedral angles Ψ and Φ are predicted using support vector regression. For the prediction of Ψ dihedral angles the addition of structural information is examined and the normalisation of Ψ and Φ dihedral angles is examined. An application of the dihedral angles is investigated. The relationship between dihedral angles and three bond J couplings determined from NMR experiments is described by the Karplus equation. We investigate the determination of the correct solution of the Karplus equation using predicted Φ dihedral angles. Glycosylation is an important post translational modification of proteins involved in many different facets of biology. The work here investigates the prediction of N-linked and O-linked glycosylation sites using the random forest machine learning algorithm and pairwise patterns in the data. This methodology produces more accurate results when compared to state of the art prediction methods. The black box nature of random forest is addressed by using the trepan algorithm to generate a decision tree with comprehensible rules that represents the decision making process of random forest. The prediction of our program GPP does not distinguish between glycans at a given glycosylation site. We use farthest first clustering, with the idea of classifying each glycosylation site by the sugar linking the glycan to protein. This thesis demonstrates the prediction of protein backbone torsion angles and improves the current state of the art for the prediction of glycosylation sites. It also investigates potential applications and the interpretation of these methods.

Acknowledgements

I thank Jonathan Hirst for his excellent supervision, Ben Bulheller for designing the website that makes the glycosylation prediction software available for use, Clare Evans, Craig Bruce, and Petros Kountouris for useful discussions, the BBSRC for a studentship and the University of Nottingham for the use of high performance computing.

Contents

List of Figures

List of Tables

List of Common abbreviations

For abbreviations of amino acid names and sugars refer to appendices A and D respectively.

ASA	Accessible Surface Area
ATP	Adenosine Tri Phosphate
BLAST	Basic Local Alignment Search Tool
BLOSUM	BLOcks of Amino Acid SUbstitution Matrix
CART	Classification and Regression Trees
CASCOR	CAScade CORrelation
CASP	Critical Assessment of methods for protein Structure Prediction
CCI	Correctly Classified Instances
Da	Dalton
DESTRUCT	Dihedral-Enhanced STRUCTure prediction
DNA	Deoxyribose Nucleic Acid
DSSP	Define Secondary Structure of Proteins
ER	Endoplasmic Reticulum
GPI	Glycophosphatidylinositol
HMM	Hidden Markov Model
kDa	kilo Dalton
MAE	Mean Absolute Error
MCC	Matthews Correlation Coefficient
NMR	Nuclear Magnetic Resonance
PAM	Percent Accepted Mutation
PCC	Pearson Correlation Coefficient
PDB	Protein Data Bank

PSI-BLAST	Position Specific Iterated BLAST
PSSM	Position Specific Scoring Matrix
PTM	Post Translational Modification
RMSE	Root Mean Squared Error
RNA	Ribose Nucleic Acid
RSA	Relative Solvent Accessibility
SA	Solvent Accessibility
SNP	Single Nucleotide Polymorphism
SVM	Support Vector Machine
SVR	Support Vector Regression

Publications Arising From This Thesis

Hamby SE, Hirst JD, Prediction of glycosylation sites using random forests. *BMC Bioinformatics* 2008, **9**:500.

Chapter 1: Protein Bioinformatics

1.1 Introduction

Proteins are the work horses of biology. Both within the cell and without and across the whole spectrum of life there are proteins. They fulfil a structural role, such as in the case of collagen, which provides the framework of connective tissue, and they are the machines of biology, catalysing the chemical reactions of life throughout the biosphere. Protein biology has long been studied by scientists interested in a wide range of organisms. Proteins are important for the understanding of biology in healthy and disease states as well as providing drug targets against pathogenic organisms, and they are even potentially drugs themselves.

The function of a protein depends on its structure. Therefore, much effort has been devoted to the determination of protein structures. Experimentally, a wide range of techniques have been used to study protein structure and dynamics. X-ray crystallography and NMR have been used to determine protein structure, and NMR spectroscopy has also been used to study protein dynamics. These methods are expensive and time consuming. Some structures, particularly membrane proteins, are very difficult or impossible to characterise experimentally. A computational approach can, therefore, be advantageous. Bioinformatics methods aim to predict the structure of proteins using the amino acid sequence and properties of the amino acids that are readily available. Rather than predict the 3D structure of the entire protein, which is very difficult, due to the number of possible structures that a given sequence can adopt, the problem is often broken down into smaller tasks, such as the prediction of secondary structure or of dihedral angles. Much work has been done on the prediction of protein secondary structure.

Post translational modification (PTM) of proteins is heavily involved with the regulation of proteins and with structural and functional aspects of proteins. Glycosylation, which we focus on in the second part of this thesis, is involved in a wide number of biological processes. Therefore, it is important to be able to determine where a protein is glycosylated and precisely which carbohydrate is joined. Once again determination of this is expensive and time consuming. This has led to a computational approach being employed to determine the location of the glycosylation sites (and indeed other PTMs).

The bioinformatics problems dealt with in this thesis are sequence analysis problems. For this reason, we begin by introducing sequence analysis and reviewing the methods used to compare biological sequences, which are essential to the field. In the first part of this chapter, we review methods for predicting secondary structure, dihedral angles and surface accessibility of proteins. In chapter 3, we present our research on the prediction of dihedral angles of proteins and its potential applications.

The second part of this introduction gives some background on PTM of proteins. A major aspect of protein structure and function, PTMs are structural modifications to a protein where a small molecule is added to a specific amino acid. These modifications are important in many areas of biology, such as the regulation of proteins and DNA, and signalling between cells and molecules. We give an overview of the different types of PTM and introduce glycosylation, a PTM where carbohydrate is added. In chapter 4, we describe our work to predict glycosylation sites and use the model generated to find out information about what determines the location of a

glycosylation site.

1.2 Sequence Analysis

In biology, there are two main areas of sequence analysis: sequence comparison, i.e., multiple sequence alignment, and prediction using sequence analysis, although the first is often used as a starting point for the second. Multiple sequence alignment is the comparison of three or more sequences.[1] It is often used to find homologous sequences in a large database and to align known homologues. This process of locating and aligning homologous sequences is central to bioinformatics and is the most important area of sequence analysis. Initially, dynamic programming algorithms were prohibitively slow, with both stochastic and tree-based methods being attempted. The introduction of progressive alignment methods in the 1980s has been the foundation for modern sequence alignment methods, a selection of which are reviewed below. Progressive alignment allows a full alignment to be built up gradually, using a tree as a guide for the alignment. It forms the basis of some of the most popular sequence alignment programs, such as clustalW.[2] ClustalW creates a distance matrix using dynamic programming combined with a sequence weight matrix, such as PAM[3] or BLOSUM.[4] The neighbourhood joining method produces a guide tree for progressive alignment based on this matrix. The progressive alignment is carried out by conducting the pairwise alignment of sequences using the tree as a guide, thus aligning more and more sequences with each iteration, until the alignment is completed. Version 2.0 of this program[5] allows faster and more accurate alignments.

PAM[3] and BLOSUM are examples of mutation matrices; these are often used by

sequence alignment programs to improve the accuracy of alignments by including evolutionary information. The level of pairwise similarity between two given amino acids can be measured by the likelihood of an amino acid substitution occurring by chance versus being inherited. This can be quantified by the number of point mutations required to go from one amino acid to the other. This is known as the evolutionary distance between two amino acids. Dayhoff *et al*. used this principle to develop a series of mutation matrices[3]. These PAM matrices are derived from the assumption that evolution proceeds by way of single point mutations. Mutation matrices can be used to find the optimal sequence alignment, the one most likely to have occurred by evolution from a common ancestor rather than by chance.

The BLOSUM matrix is calculated in a similar way to the Dayhoff matrix. Henikoff and Henikoff use sequence blocks taken from regions highly conserved between sequence families[4]. The sum of pairwise sequences for these blocks is used to calculate an odds matrix in similar fashion to the Dayhoff matrix. Sequences are clustered based on percentage identity, in order to allow differing evolutionary distances to be included. This results in a series of matrices equivalent to the PAM matrices developed by Dayhoff. BLOSUM62 is the most commonly used, the 62 indicating it was compiled using clustering at 62% identity.

1.2.1 PSI-BLAST

One of the programs that has most revolutionised bioinformatics is PSI-BLAST (position specific iterated BLAST). [6] PSI-BLAST identifies homologous sequences from a database using BLOSUM62 matrices. It is also used to generate position specific scoring matrices (PSSMs). PSI-BLAST profiles are used in a number of

sequence alignment methods and in many other areas of sequence analysis as a way of representing the amino acid sequence. The generation of these profiles is discussed in chapter 4. PSI-BLAST is an enhancement of the BLAST algorithm,[7] used for searching protein and DNA databases for homologous sequences. BLAST uses well-defined sequence similarity measures in the form of PAM matrices to approximate the results that would be obtained using dynamic programming methods. PSI-BLAST improves over BLAST in both computation time and accuracy, by using sequence profiles to perform an iterative search of the database. PSI-BLAST allows for the generation of gapped alignments, reducing the number of potential alignments that need to be searched for the optimal alignment. PSI-BLAST also automatically generates a PSSM from the significant alignments found in a given iteration and uses this as input for the next iteration. Profile based searches are more sensitive to distant homologies than pairwise based alignments. A PSSM represents the similarity and evolutionary distance for each amino acid in a protein relative to all of the 20 standard amino acids (see chapter 3 for a more detailed description and an example). The PSSM profiles are often used as input to other methods, e.g., as the input to a prediction program, since they represent an amino acid sequence in a way, which includes evolutionary information. The level of conservation of a group of amino acids is important when relating sequence to function. In this work, we use PSSMs as an input for prediction of both secondary structure and real value dihedral angles (Chapter 3). We use PSI-BLAST with multiple iterations to generate these, because of the ease of obtaining PSSMs from PSI-BLAST and because of the track record of the program being used in a similar manner. Other methods described are described briefly below.

There have been many attempts to enhance the PSI-BLAST algorithm and many sequence alignment programs use PSI-BLAST profiles to enhance alignments. HHPred[8] combines PSI-BLAST with hidden markov models (HMMs). Rangwala and Karypis[9] use PSI-BLAST as the base for an incremental alignment method based on sequence windows. CTX-BLAST[10] incorporates a contextual alignment model into PSI-BLAST. Lee *et al.*[11] tackle the problem of an increased probability of the introduction of false positives with each subsequent iteration of PSI-BLAST by introducing a ranking of hits produced from the first and last iteration. Przybylski and Rost[12] boost the performance of PSI-BLAST using consensus sequences. We prefer the standard version of PSI-BLAST over these, as none of these readily outputs a PSSM without alteration, and not all are easily available.

1.3 Protein Structure

Sequence analysis has been used for the prediction of many different biological properties. It is common to use PSI-BLAST profiles to represent the protein sequence for prediction experiments. The goal is to determine some property of the protein from its amino acid sequence. Some of the most researched areas are structure prediction, both tertiary and secondary.

Anfinsen showed that all of the information about a protein can be determined from its primary structure.[13] The primary structure of a protein is the sequence of amino acids from N terminus to C terminus. The structures of the 20 amino acid types are given in Appendix A. There has been evidence for the contribution of environmental factors to protein folding and structure, and PTMs also play a role in determining the final structure of a protein. However, it is likely that a reasonable approximation of the 3D

structure of a protein in its native state can be determined from the primary structure with no additional information. This is, however, a calculation with too many permutations to be achieved *ab initio*. As a result of this, many less complex problems have been defined, to provide a bridge to predicting the complete structure. These include prediction of secondary structure, of dihedral angles, the prediction of surface accessibility of amino acid residues and prediction of residue contacts.

A stepping-stone to the 3D structure is the secondary structure. This consists of a series of structural motifs that occur often within proteins. These can be considered as building blocks, which form the bulk of the protein's structure. The secondary structure is most comprehensively described using the eight states assigned by the program DSSP[14], although this is often reduced to a three state description of secondary structure. DSSP assigns structure based mainly on the hydrogen bonding patterns of a protein. These are used to assign the states of α-helix and 3_{10}-helix, β-sheet, β-bridges, π-helices, turns and bends.

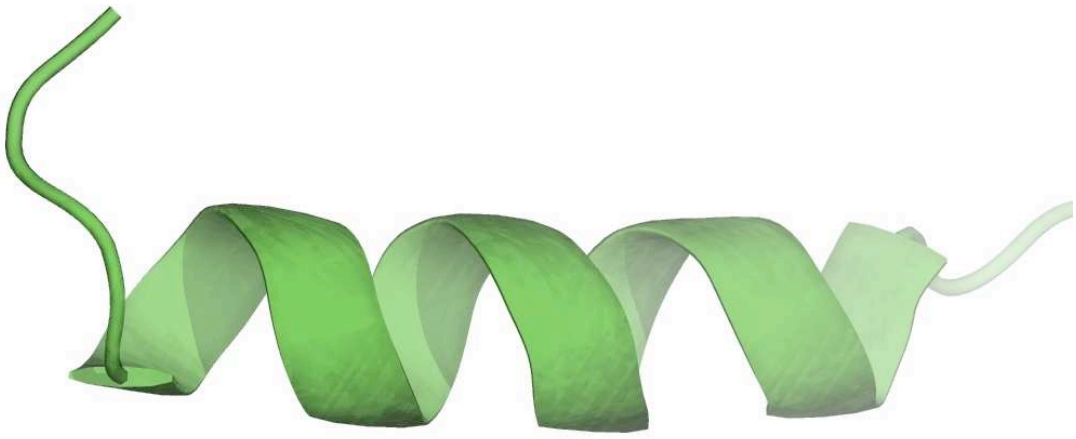


Figure 1.1. An example of an α helix from the UBA domain of the protein p62.

An α helix (figure 1.1) is a regular helical arrangement of the amino acids held in place by the formation of amide to carbonyl hydrogen bonds. There are 3.6 residues per turn, with a rise between residues of 1.5 Å. Such helical structures can range in length from four amino acids to 40 or more and are sometimes amphipathic in nature. The helical structure can be broken (or prevented) by the inclusion of Pro, Gly, Ser or Thr in the sequence. Residues that encourage helix formation are Ala, Glu, Leu and Met.

3_{10}-helices consist of three residues per turn and each hydrogen bond encloses a ring of ten atoms. Main chain hydrogen bonds are separated by three residues. Often occurring at the end of α-helices, 3_{10}-helices are less stable than α-helices, because the dipoles are less well aligned. This also means the packing is less energetically stable than that of an α-helix. In a π-helix, the hydrogen bonds are formed between residue *i* and residue *i*+5. This secondary structure element is a rare occurrence and long π-helices are not found.

β-sheet (figure 1.2) takes the form of a planar arrangement of the amino acids, where the amino acids line up in extended conformation in stretches of five to ten residues. Typically the surface of this is corrugated in shape. β-sheets can be either parallel or anti-parallel in form. In parallel β-sheets the direction of the amino acid chains within the sheet is the same throughout. These tend to have a longer distance in the sequence between the strands in the sheet, with long loop structures and random coil sections filling the gaps. Alternating direction of the amino acid backbone characterises anti-parallel β-sheets. This form of β-sheet is often linked by short loops such as β-turns. An isolated pair of parallel β-sheet type structures is known as a β-bridge.

The loops and turns in a protein structure can take several forms. Turns are usually three to ten amino acids in length, and have defined structure, whereas loops are generally disordered in structure (i.e., random coil) and can be any length. Loop regions, in particular, are often of functional importance and can change conformation in order to facilitate binding of molecules. β-turns may be as few as three residues long and consist of a single hydrogen bond between two residues, which bends the amino acid sequence into a hairpin shape. There are several variations on the basic β-turn and this type of structure may also be known as a bend or hairpin. Anything not conforming to the above structural motifs is classified as random coil. Such structures form a large proportion of protein structures.

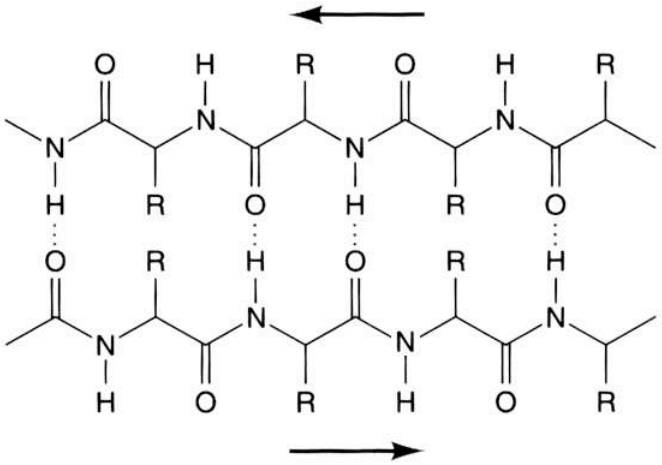


Figure 1.2. The hydrogen bonding pattern of an anti-parallel β-sheet. Dashed lines are hydrogen bonds and the arrows indicate the direction of the chain. (Image released under the creative commons licence.)

The supra-secondary structure of a protein is a further progression towards the 3D structure. Certain commonly identified structural motifs that are made up of secondary structure elements, such as those described above, can be the first to form when a

protein folds into its native structure. These include structures such as the β-barrel or the α-helical bundle, as well as structural motifs involving loops and both β-sheets or α-helices. Many such supra-secondary structure motifs have been described.

The tertiary structure of a protein is its complete 3D structure. This structure is determined by weak interactions between amino acid residues and side chains. Interactions such as steric hindrance, electrostatic, hydrophobicity related interactions, and van der Waals interactions all play a role.

1.3.1 Dihedral angles

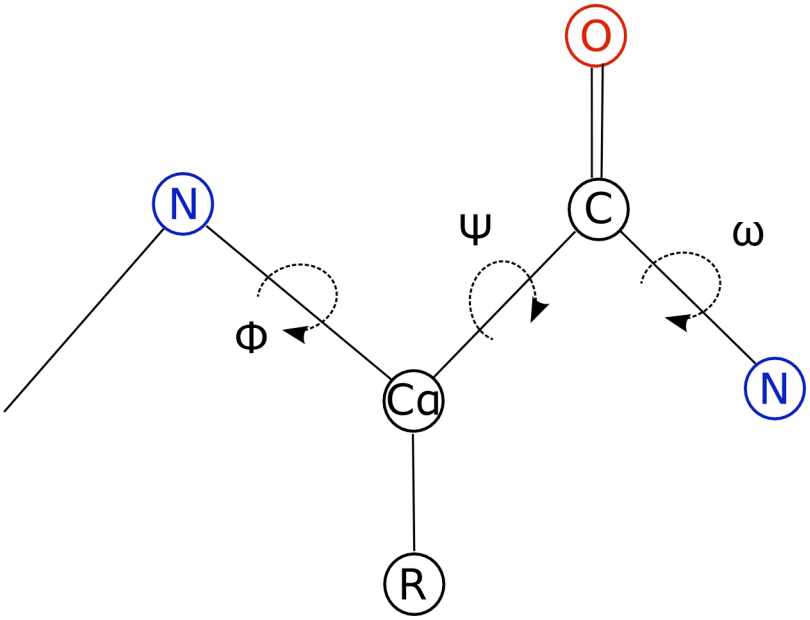


Figure 1.3. Dihedral torsion angles of the protein backbone. The location of the backbone dihedral torsion angles Φ, Ψ, and ω are shown by the dashed lines indicating rotation around a bond.

Dihedral torsion angles are described by rotation around the bonds along the protein backbone (figure 1.3). Of the three backbone dihedral angles, ω is generally planar, i.e., 0° in the *cis* conformation and 180° in the *trans* conformation. This is due to the

delocalisation of the carbonyl π electrons and the lone pair of the nitrogen atom. The other backbone dihedral angles are limited by steric hindrance. In practice, only 10% of the available conformations of Ψ and Φ angles are observed. Ramachandran[15] examined the available conformations in a selection of proteins, treating each atom as a hard sphere based on its van der Waals radius and disallowing steric clashes.

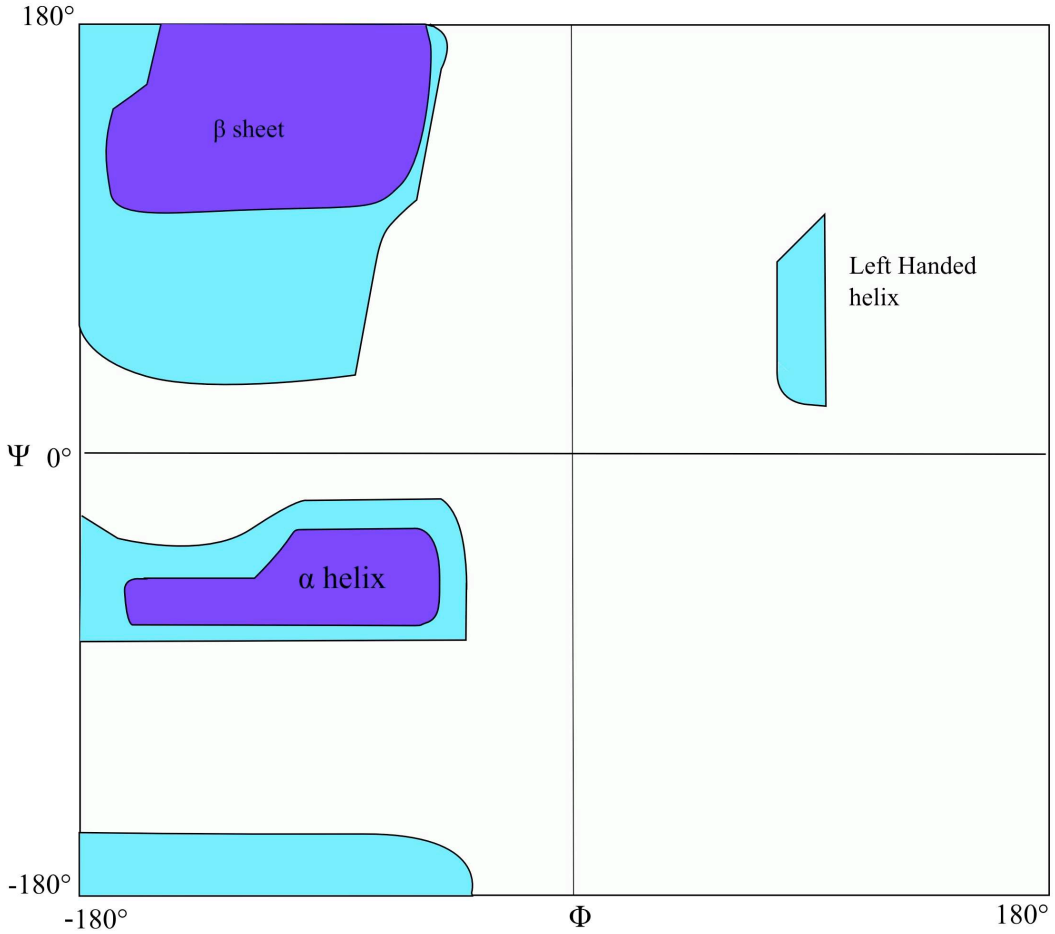


Figure 1.4. The Ramachandran plot. The regions are labelled with the type of secondary structure indicated by the angles found there. The light shaded areas encompass allowed angles including those of glycine. The dark shaded areas show angles not including glycine.

This resulted in the now classic Ramachandran plot (figure 1.4). Correlating secondary structure with the Ramachandran plot shows that regular structures have

similar values of Φ and Ψ for a given structure type. From the dihedral angles it is possible to get an approximation of the structure of a protein. Thus, it is useful to have a prediction of the dihedral angles for tasks such as structure prediction (both secondary and tertiary). There have been many attempts to predict the secondary structure of a protein and, more recently, several attempts have been made to predict Ψ and Φ dihedral angles.

1.3.2 Secondary Structure Prediction

Elements of secondary structure were first proposed by Pauling[16]. Attempts to predict the location of the secondary structure elements have been ongoing ever since, using a variety of methodologies. These are too numerous for all of them to be described here. However, the most important are presented below with some historical perspective. We use secondary structure prediction in both the areas of research covered in this thesis. It is a major component of our work to predict dihedral angles as we hypothesise that using secondary structure prediction will improve the accuracy of dihedral angle prediction, and in chapter 4 we use secondary structure prediction to add extra information to the input for our glycosylation predictor.

Initial attempts at secondary structure prediction centred around amino acid propensity. GOR[17], now at version V[18], in its first version only used single residue statistics within a sliding sequence window. The statistics are generated for residues within the sequence window, with the objective of predicting the state of the residue at the centre of the window. By sliding the window along the length of the sequence, it is possible to obtain predictions for the entire protein. Subsequent versions of the program improved by adding pairwise statistics (version II[18]) and information

theoretic methods (version III and IV[18]). In version V, the information theory methods previously employed are combined with evolutionary information via PSSMs obtained from PSI-BLAST. The GOR algorithm uses a combination of information theory and Bayesian statistics to predict secondary structure. When combined with PSI-BLAST, this produces an accuracy of 73.5% Q3, although this has since been improved upon. Q3 is the three state prediction accuracy for protein secondary structure (equation 1.1)

$$Q3 = \frac{p(\alpha) + p(\beta) + p(coil)}{N} \qquad (1.1)$$

where p is the number of residues correctly predicted for a given secondary structure type and N is the total number of residues.

PHD[19] is a neural network based prediction method, consisting of a three level feed forward network. The first level of the network takes as input a multiple sequence alignment and assigns the central residue of a 20 amino acid sliding window to one of the standard three structural classes. The second level takes the output of the first as input and once again predicts the structure class. The input from the preceding network is expressed as a 17 amino acid window, with amino acids represented by three binary outputs describing the structure classification. A number of such networks were trained and the outputs were fed into the third level of the network. The third level represents a jury decision by arithmetic average that produces a final prediction for secondary structure class. This schema produces accuracy of 70% (Q3).

Another neural network based method, and one of the most popular secondary structure prediction programs available, which is still widely used, is PSIPRED.[20]

This method uses PSSMs generated by PSI-BLAST as input for a neural network. This was the first method to use such profiles directly rather than compiling a complete multiple sequence alignment. The output of the initial feed forward neural network is then filtered by a second network to give the secondary structure prediction. This gives an accuracy of around 76%. Here we use predictions from PSIPRED as input for glycosylation prediction in the hope of improving accuracy (see chapter 4).

HMMSTR[21] uses HMMs for secondary structure prediction as well as other protein properties. The model is based on sequence structure motifs and uses a voting scheme to determine the final structure. The HMMs used are different in that they are not the typical profile HMMs used in many other models. The model uses the correlations between sequence structure motifs to reduce the number of parameters in the model, and predicts secondary structure with an accuracy of 74.3% (Q3).

JNET[22] uses a neural networks arrangement which is similar to that used by PHD, using PSSMs produced by PSI-BLAST as input. JNET predicts secondary structure with an accuracy of 76.4% (Q3). A different approach is taken by Pollastri *et al.*[23] Their program SSPro uses a bidirectional recurrent neural network for secondary structure prediction with PSSMs. The performance of SSPro is 77%-80% (Q3), dependent on the testing set used. A brute force local clustering method is proposed by Jiang.[24] This tries to take into account long range interactions, which form a significant component of protein structure.[25]

DESTRUCT[26] is a neural network based method, which predicts both secondary

structure and dihedral angles. Initially, two neural networks are trained, one predicting secondary structure and one predicting the Ψ dihedral torsion angle. The output from each is fed into subsequent neural networks in an iterative manner; the predictions of Ψ are used to enhance predictions of secondary structure and visa versa. The resulting prediction of secondary structure is comparable in accuracy to PSIPRED. This prediction program will be discussed in some detail later, as it forms a basis for some of the work in this thesis.

Montgomerie *et al*. developed the program proteus.[27] This includes structural alignments as part of the prediction process, and achieves a Q3 score of 88% by combining the secondary structure predictors of PSIPRED, JNET and TRANSSEC[27] (TRANSSEC was developed by the authors) as a "Jury of experts". Birzele and Kramer[28] define a new representation of secondary structure based on frequently occurring patterns. The authors use this to perform a prediction with a Support Vector Machine (SVM) classifier, which is comparable to PSIPRED in accuracy. A more traditional approach to prediction using SVMs is taken by Karypis.[29] Using a novel kernel function cascaded SVMs are trained to predict three state secondary structure with of 79.3%. SVM classification is an opaque method. He *et al*.[30] use an SVM in combination with a decision tree to extract meaningful rules with regard to protein secondary structure. The SVMs average 77.6% accuracy for helix, 80.7% for sheet and 70% for coil. Zhong and co-authors[31] use k-means clustering to divide the training sets into representative clusters. This allows them to use their method of clustering SVMs to make predictions of secondary structure, which are in the region of 80%, although this varies across the clusters obtained. Won *et al*.[32] use genetic algorithms to evolve a HMM for secondary structure prediction. The model developed improves

over HMMSTR, but is still much less accurate than state of the art models for secondary structure prediction. Prof[33] is an ensemble method consisting of multiple neural networks combined using simple linear discrimination and a further neural network. The neural networks used are based on the methods GOR and PSIPRED. The combined result gives an accuracy of 77% (Q3). Yao *et al*.[34] predict secondary structure using a probabilistic model. The authors combine the dynamic Bayesian method with a neural network. This combination gives results that are comparable with state of the art methods.

1.3.3 Prediction of dihedral angles

The Ramachandran plot clearly shows the link between the dihedral angles and protein structure. Thus, it is useful to predict dihedral angles. The DESTRUCT method[26] uses dihedral angles to improve the accuracy of secondary structure and predictions of secondary structure to improve the accuracy of dihedral angle predictions. DESTRUCT was the first server to predict real value dihedral angles, achieving a Pearson correlation coefficient (PCC, see chapter 2 for definition) of 0.47. Previously, HMMSTR[21] predicted categories for dihedral angles, using HMMs in a manner similar to the method for predicting secondary structure described above. A more recent method for predicting dihedral angle regions is DHPred.[35] The authors define dihedral angle regions H, E and O (outlier) based on the Ramachandran plot. Residues are classified as belonging to a particular dihedral angle region using SVMs. The authors employ a two level approach to classification. First, sequences represented by PSSMs generated by PSI-BLAST are input into the first SVM classifier, which outputs predictions for each of the three dihedral angle regions considered. Secondly, these predictions are combined with the PSSM with a sequence

window size of seven and used as input for the second SVM classifier, which produces the final predictions for the state of each residue. This results in an accuracy of approximately 80%, comparable to the accuracy of PSIPRED and other secondary predictors, a legitimate comparison given that the dihedral regions correspond to the three state secondary structure of proteins. DESTRUCT was primarily motivated towards secondary structure prediction, with the dihedral prediction having the sole purpose of improving secondary structure accuracy. For this reason, only the Ψ dihedral angles are predicted by DESTRUCT. The other dihedral angles are less significant with regard to the definition of secondary structure, although Φ does play an important role when considering the tertiary structure.

Real Spine[36] improves substantially upon DESTRUCT. The first version gives a correlation coefficient of 0.62 between predicted and actual Ψ dihedral angles. The authors use twin neural networks. The inputs to both networks consist of PSSM profiles combined with predicted secondary structure information. The two networks produce a consensus prediction by averaging the output of the networks. Real Spine also predicts relative solvent accessibility (RSA) using the same methodology. The prediction for RSA has a correlation coefficient of 0.74. Real-Spine 2,[37] the second version, substantially improves over Real Spine, using a very simple alteration. Due to the properties of the sigmoidal function, the neural networks of Real Spine function poorly with respect to predictions in the region between -36° and +36°. The dihedral angle distribution is shifted by a normalisation step so that there are relatively few angles in this region. This small adjustment improves prediction accuracy to 0.75 (PCC). Unlike its predecessor, Real-Spine 2 does not predict RSA values. However, it is the first to produce real-value predictions for Φ as well as Ψ. The method employed

for predicting Φ is similar to that for Ψ. However, the normalisation used is different to allow for the differing distribution of Φ.

Whilst there are limits to predictive power, it is clear that there is much scope for improvement in the prediction of real value dihedral angles. Such improvement would enable the dihedral angles to be used to accurately predict the 3D structure of proteins, and potentially to aid in the assignment of structure from NMR spectra. In this work we hypothesise that we can improve on the above prediction methods by selecting a new machine learning method and use secondary structure prediction to enhance the accuracy of the prediction method. We selected Support vector regression (SVR) for this purpose. A detailed discussion of the reasoning behind this is presented in chapters two and three. Later, we also apply the normalisation methodology of Real Spine to SVR.

1.3.4 Hydrophobicity and Surface Accessibility

In our work to predict glycosylation sites, we use information about both hydrophobicity and surface accessibility, as these are both key properties. We include them, as PTMs can only take place on the outside of a protein and so finding those residues with high surface accessibility may improve accuracy. Here, we give an overview of these two properties and review methods for surface accessibility prediction.

1.3.4.1 Assignment of Hydrophobicity and Surface Accessibility

Hydrophobicity is a defining property of protein structure. A molecule is hydrophobic if it is repelled by water and hydrophilic if the opposite is true, hydrophobicity is the

degree by which molecules are repelled by water and is a sliding scale. Amino acids can be divided into two groups based on their hydrophobicity i.e. whether they are hydrophobic or polar. Hydrophobic amino acids are more likely to be found in the centre of a globular protein, or in the membrane bound sections of a membrane protein. Polar amino acids are more likely to be found near the surface. There are various hydrophobicity scales,[38] which rank the amino acids according to their degree of hydrophobicity. There are, however, instances when buried residues are polar or exposed residues are hydrophobic, usually due to structural considerations or because of the need for functionality.

Another approach is to consider the surface area that is exposed to solvent whilst in a given protein. This is known as the accessible surface area (ASA). The solvent accessibility of a protein is a related quantity, which can be determined by estimating the number of water molecules in contact with the amino acid's surface. This value is calculated from molecular coordinates by DSSP, and is known as relative solvent accessibility when expressed on a continuous scale normalised with respect to the maximum solvent accessibility of each residue.[39]

1.3.4.2 Prediction of Solvent Accessibility

RSA can be predicted as either a real value or projected onto a series of discrete states. Other methods also predict the ASA area directly. The neural network based predictor developed by Rost and Sander[39] achieves only modest accuracy for a ten state prediction of RSA. The ten states are selected to give a finer grained distinction of RSA levels near to the protein core. The method then uses an arrangement of neural networks to predict RSA. The arrangement of neural networks is similar to that used

in PHD, described previously.

The majority of methods predict solvent accessibility as a number of categories, although it is ideally preferable to obtain a real value, as RSA is a sliding scale. In our work we use real value predictions. However, we also give a brief overview of some categorical predictors for both historical context and completeness. Li and Pan[40] use multiple linear regression to predict two state solvent accessibility. Yuan *et al.*[41] predict two state solvent accessibility using SVM classifiers. Pollastri *et al*[42]. use bidirectional recurrent neural networks (RBNN) to predict both solvent accessibility and contact number. Multiple networks are trained and evaluated by cross validation. RVP-NET[43] uses neural networks to predict real values for solvent accessibility. This method gives a PCC of 0.45-0.46 depending on the test data used. Kim and Park[44] predict relative solvent accessibility as both a two and three state classification, employing various thresholds. SVM predictions are combined by the use of a directed acyclic graph based scheme. The resulting method, PsiSVM, produces an accuracy of 78% for two state prediction. Wang *et al*.[45] use multiple linear regression to predict real values for solvent accessibility. The sequences are represented by PSSMs and prediction is comparable to other methods.

Multiple linear regression is also employed by Qin *et al*.[46] for the prediction of both solvent accessibility and secondary structure. QBES[47] presents a substantially different approach to predicting solvent accessibility. The authors use quadratic programming as a means of minimising a simple energy function. Gianese and Pascarella[48] employ a consensus method comprising the predictors JPRED, AccPro and PP.[49] PP was produced by Gianese *et al.*[49] and uses profiles of conditional

probabilities to perform its task. The three predictors are combined using a state mapping approach to two state RSA prediction to produce the consensus. SVM Cabins[50] integrates the two approaches of classification and regression to improve the accuracy of solvent accessibility prediction.

SABLE[51] is a method based on neural networks for regression in order to predict real values for RSA. The method is trained on a large non-redundant dataset derived from Pfam[52]. The sequences are represented using PSSMs extracted from PSI-BLAST. The authors test both feed forward and Elman[53] networks for prediction. The final predictor achieves a correlation coefficient of 0.66. We chose to use SABLE in this work for several reasons. Firstly, it is both freely available and open source allowing ease of use and integration with our existing software, whilst being reasonably accurate in predicting real values for solvent accessibility. Although more accurate methods exist, they were not easily available for use.

1.4 Post Translational Modification

Proteins are synthesised in the body by way of transcription and translation, before folding into their final structure. All proteins start out as a DNA sequence. This is transcribed into messenger RNA. In the case of eukaryotic organisms, the sequence undergoes RNA splicing, which can alter the order of exons to produce novel products. Introns are removed during this process. The messenger RNA is transformed into protein sequence by the ribosome, transfer RNA brings the appropriate amino acids to the ribosome and adenosinetriphosphate (ATP) is consumed to bind together the amino acids using the messenger RNA code as a template. The protein folds into its final structure, either in the cytosol or is transferred to the endoplasmic reticulum

(ER). Many proteins require chaperones to fold into the final structure. These chaperone proteins are also involved in a quality control process to ensure correct folding.

After proteins are created in the body they can undergo a variety of modifications essential for the correct folding and functionality of the protein[54]. There are over a hundred types of PTM, but some of the most common are briefly discussed below. These modifications can be either transient or permanent and often confer function on the protein in question. They occur across the entire spectrum of life. These modifications can be classified by the molecule added during the modification. There are also PTMs involving protein cleavage by proteases. However, it is beyond the scope of this thesis to discuss these here. The most common types of PTM are phosphorylation, acylation, alkylation, glycosylation, and oxidation, although there are many others. In this work we focus on the prediction of glycosylation sites from sequence with some additional work aimed at understanding the models that generate these predictions. Our hypothesis is that even where no consensus sequence motif exists there will be certain amino acids that favour glycosylation. So for this reason we use information on the pairs of amino acids surrounding glycosylation sites with the hypothesis that this will lead to a higher accuracy of glycosylation prediction.

1.4.1 Post Translational Modification Overview

Here we give a brief overview of some common types of PTM before going on to give a more extensive overview of glycosylation. Phosphorylation[54,55] occurs upon the addition of a phosphate group to either Ser, Thr or Tyr. This particular modification is important for regulating cell processes and for signalling both within and between

cells. Some phosphorylation sites are transitive yin yang sites. In such cases the site can either be phosphorylated or glycosylated, depending on the physiological conditions and the environment of the protein. Such sites are often important as regulatory elements in the cell cycle and in signalling processes. As such, the functions and regulation of phosphorylation and cytoplasmic O-glycosylation are intricately linked. Acylation encompasses the addition of fatty acid chains of length C_2, C_{14}, C_{16}, and the 8kDa chain of ubiquitin. Acetylation[56] is the addition of multiple lysine residues to the histone terminus. In myristoylation[57] a mirosyl group is added via glycine to the protein and effects the movement of the protein towards membrane interfaces. In palmitoylation[58] the acyl group is transferred to the thiolate chain of cystine. This modification is also involved in the membrane anchoring of proteins. Ubiquitylation[59] involves the carboxyl terminus of the protein ubiquitin being added to lysine. This is either added as a single molecule (mono-ubiquitylation) or as the stepwise addition of multiple ubiquitin segments. Poly and mono-ubiquitylation both are involved with the direction of proteins to new locations within the cell. Alkylation involves the addition of alkyl substituents of varying size to several different amino acids. N-linked methylation[60] of Lys and Arg in histones is an important part of the transcriptional regulation cycle complementing acylation. Protein S-Prenylation[61]: C_{15} and C_{20} lipid groups can be added to protein, e.g. in the Ras family of proteins. Disulfide bridges[62] are formed by the oxidation of the thiolate side chain of cysteine. These modifications are important in linking protein chains and providing stability in protein structure.

1.4.2 Glycosylation

Glycosylation[63] involves the addition of highly complex carbohydrate chains to

protein at either Asn, Ser or Thr residues and occasionally at Cys. As prediction of glycosylation sites is a major subject of this thesis, we discuss in detail the types of glycosylation after giving an overview of carbohydrate chemistry, and the methods for prediction of glycosylation sites currently available.

1.4.3 Structure of Glycans

1.4.3.1 Carbohydrates

Carbohydrates are chains of monosaccharides that fulfil a wide variety of functions. In the context of PTM, oligosaccharides are added to protein under various conditions. Oligosaccharides are usually taken to be chains of between two and ten monosaccharides of varying composition. They are often branched and vary greatly in composition. Polysaccharides are large molecules that are polymers of repeating sugar motifs, either repeated mono- or disaccharides or a more complicated arrangement.

1.4.3.2 Monosaccharides

Monosaccharides are of the form $C_x(H_2O)_n$. They possess a carbonyl group, either an aldehyde or a ketone; *n* ranges from three to nine. All monosaccharides, except dihydroxy acetone, are chiral about at least one carbon atom. The carbon atoms are numbered as per standard organic chemistry rules and monosaccharides are almost always cyclical in form. There are many monosaccharides, which can be conscripted to make up the oligosaccharides encountered in glycosylation. Some of these sugars only occur in plants or in prokaryotes. The most common found in vertebrate glycosylation[66] are D-Glucose (Glc), N-Acetyl-D-Glucosamine (GlcNAc), D-Galactose (Gal), N-Acetyl-D-galactosamine (GalNAc), D-Mannose (Man), D-Xylose (Xyl), D-Glucoronic Acid (GluA), L-Fucose (Fuc) and N-Acetylneuraminic acid

(NeuAc, also known as sialic acid).

1.4.3.3 Glycosidic linkages

The monosaccharides are linked together by two possible types of glycosidic linkage α and β. These linkages can also occur between various different carbon atoms. The type of linkage is labelled corresponding to the numbers of the carbon atoms concerned. β1-4 and β1-6 linkages are particularly common. The variety of linkages between the monosaccharides, and the potential for branching of the oligosaccharide chains, account for the large number of potential oligosaccharide structures, of which nature only uses a small fraction. The glycosidic linkage is very flexible and allows for the formation of multiple conformations of glycan, whilst maintaining the rigidity of the constituent sugars, which tend to be relatively rigid.

1.4.3.4 Oligosaccharides

Oligosaccharides are the most common addition in glycosylation. They are polymers of varying composition of monosaccharides, ranging from two to 30 monomers. Oligosaccharides can be named with respect to the number of monosaccharides they comprise: disaccharide for two, trisaccharide for three, etc. These polymers have a reducing and non-reducing terminus, in much the same way that proteins have amino and carboxy termini. The reducing end has an available anomeric centre when in free form and is referred to in this way after the attachment to a hydroxy group, e.g., in glycosylation. It is possible for an oligosaccharide to have no reducing end, e.g., sucrose, which has its glycosidic linkage between the two anomeric centres and thus has no reducing end.

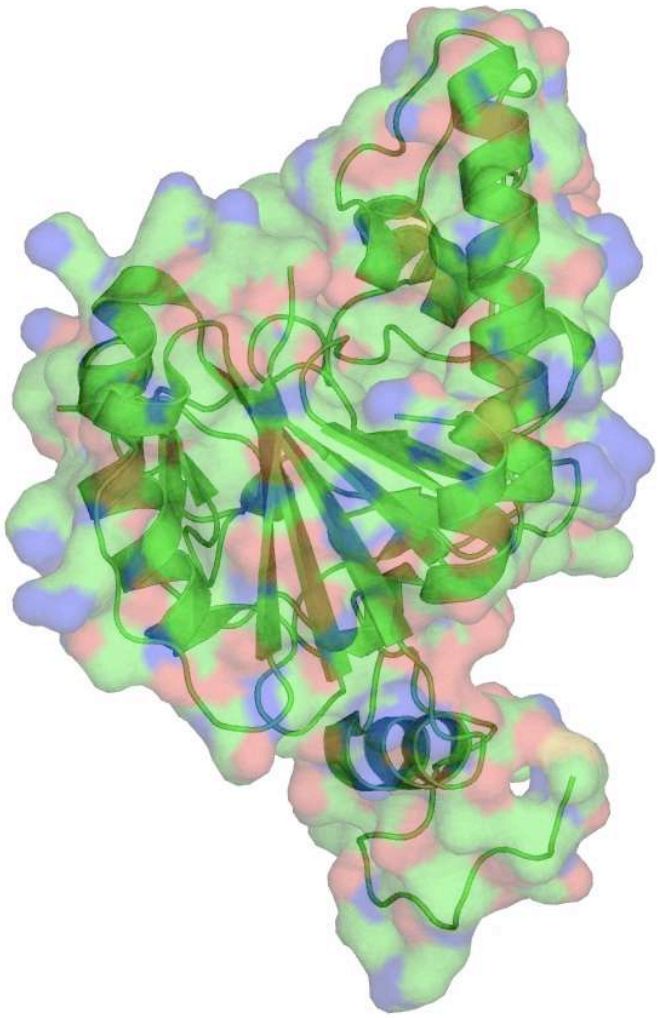


Figure 1.5. Glycosyltransferase A. Generated from the PDB structure using PyMol. The secondary structure is shown in ribbon form with the surface of the protein projected over it. Colours of the surface show the charge distribution of the amino acids on the surface. Green is neutral, red is hydrophobic and blue hydrophilic.

1.4.4 Glycosyltransferases

This large family of enzymes[64] is responsible for initiating glycosylation and for elongating glycan chains. Since the substrate specificity of such enzymes is essential to the prediction of glycosylation sites, we briefly review them here. The substrates of these enzymes are varied, but all have in common the transfer of glycans, either monosaccharide or oligosaccharide, to a new substrate. Most of these enzymes are

concerned with chain elongation and in this case the substrate is another glycan. However, receptor substrates can be lipids, peptides, small molecules or DNA. The donor substrates are also varied, for example, dolichol and other lipids. In general, the specificity of glycosyltransferases is such that one enzyme catalyses the formation of one glycosidic linkage e.g. glycosyl transferase A (figure 1.5). However, in several cases multiple enzymes catalyse formation of the same glycosidic linkage. Examples of this are the fucosyltransferases III-VIII, which catalyse the same alpha 1-3 linkage to attach fucose to N-acetyllactoseamine[65]. There are also rare cases where a single enzyme is capable of forming more than one type of glycosidic linkage, e.g., the case of fucosyltransferase III that can catalyse formation of alpha 1-3 linkages as well as alpha 1-4 linkages. There are also enzymes, which have more than one active site.

1.4.5 N-Linked Glycosylation

In this thesis, one of the major objectives is prediction of both types of glycosylation site: N-linked and O-linked. Here and in the following sections we introduce both in some detail. This provides some context to our work and highlights the significance of the modifications. N-linked glycosylation[62,55] is the addition of an oligosaccharide to Asn. This occurs at the consensus sequence Asn Xxx Ser/Thr,[55] where the Xxx is anything except Pro. This sequence is necessary, but not sufficient, for glycosylation. N-linked glycosylation takes place co-translationally in the lumen of the ER. Initially, the glycan is pre-assembled on a lipid dolichol molecule[55], which acts as a scaffold. This molecule is synthesised on the inner surface of the membrane of the ER, beginning with the transfer of GlcNAc-P to the lipid-like precursor Dolichol-P. 14 sugars are added to dolichol. Oligosaccharyltransferase attaches this glycan precursor molecule to Asn. The transfer to Asn takes place at the consensus sequence found in a

protein which is undergoing synthesis and transport through the membrane into the lumen of the ER. Oligosaccharyltransferase is a multi-subunit protein complex, which is embedded in the ER membrane. The glycan precursor is transferred to Asn as the protein emerges into the lumen of the ER[55].

Within the lumen of the ER the oligosaccharide is trimmed of some of its constituent monosaccharides[66] (figure 1.6). Glucosidase I and II remove the first two Glc residues. Subsequently, a mannose residue is removed by alpha-mannosidase. This appears to be an important control step for the folding process of the protein with the assistance of chaperone proteins. The oligosaccharides assist in keeping the protein in solution during and after the folding process, thereby, indirectly assisting the function of the chaperone proteins. The chaperones are known to bind to specific points on the immature glycan, thus targeting incorrectly folded proteins for degradation. Once in the Golgi body, alpha mannosidase removes up to a further four mannose residues[66], leaving $Man_5GlcNAc_2$. This structure forms the basis for all other N-linked glycan chains. There are often some glycans that escape some of these precursor steps.

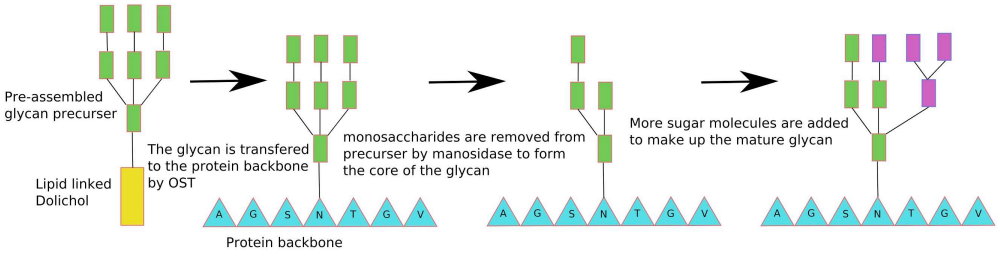


Figure 1.6. The synthesis and maturation of N-glycans. Green residues represent mannose, purple rectangles are other sugars of unspecified identity, amino acids are shown by blue triangles and the large yellow rectangle is a lipid dolichol molecule.

These are expressed as oligomannose, and cannot form complex or hybrid glycan types. The trimming of glycan precursors only occurs in multi-cellular organisms. In

yeast, for example, extra mannose residues are added to the glycan where in multi-cellular organisms they would be removed.

There are several types of glycan that can be constructed in the Golgi body. The Golgi body contains many specific glycosidases and glycosyltransferases capable of adding or removing different sugars to produce high mannose, hybrid or complex glycan types. All N-linked glycans have a trimannosyl core structure ($Man_3GlcNAc_2$). This is the base for many types of linear or branched oligosaccharide. High mannose type oligosaccharides contain between five and nine mannose residues attached to the GlcNAc residues within the trimannosyl core structure. The complex oligosaccharide type does not contain any mannose residues outside of the core structure. Characteristically complex glycans have a disaccharide GlcNAc(beta1-4)Gal attached to the trimannosyl core. This may be a repeating unit or the base for the build up of a complex structure with two, three or four branches. These structures are produced by the stepwise addition of monosaccharides by various glycosyltransferases. Hybrid oligosaccharides possess features of both complex and high mannose type oligosaccharides.

1.4.5.1 Function of N-glycans

As well as assisting in the protein folding process[67], N-linked glycans have a number of well documented functions[54,68]. They have other structural roles in maintaining the conformations of proteins in the appropriate state, as well as preventing non-specific interactions and assisting in the orientation of cell surface molecules. N-linked glycans are important as cell adhesion molecules and for protein signalling, e.g., blood group determinants are oligosaccharides, which can either be N-linked or O-linked glycans.

They also play a role in the serum clearance of proteins.

1.4.6 O-Linked Glycosylation

There are several types of O-linked glycosylation, characterised by the glycan binding to an oxygen atom with an alpha glycosidic linkage. We discuss the common types and their function concentrating on mucin type glycosylation, which is central to the section of this work on predicting glycosylation.

1.4.6.1 O-GalNAc or mucin type modification

Mucin type glycoproteins[63,64,69] are usually large molecules (typically greater than 200kDa), which are heavily glycosylated at clusters of Ser and Thr residues, to the extent that one in three amino acids may be glycosylated. The glycan chains that are added to these proteins are varied in composition and structure. These proteins exhibit regions of tandem repeats of variable length[70]. These contain numerous glycosylation sites and are usually replete with Pro residues, which encourage glycosylation[64]. Mucin glycoproteins are often secreted or embedded in the membrane. Membrane based glycoproteins mediate cellular adhesion and are involved in cellular signalling. Secreted mucins contribute to the mucosal defences of the body, and are one of the key ingredients in mucosal secretions, giving them their viscosity. O-linked glycans are synthesised in the Golgi body. The synthesis consists of the stepwise addition of monosaccharides to the oligosaccharide chain by glycosyltransferases. There is no trimming and reassembly of the glycan after synthesis unlike N-linked glycans. Whilst O-linked glycans can be long and complex structured oligosaccharides, they can also be short and relatively simple. Most commonly, the monosaccharides GalNAc, GlcNAc, Gal, Fuc, and sialic acid are found in O-linked glycans, although others have

been observed. In contrast to N-linked glycans, O-linked oligosaccharides have less branching in the structure, being based on a biantennary (two branched) core. O-glycans may be classified by the core structure, which falls into one of eight types (see figure 1.7). The synthesis of all mucin type glycans begins with the attachment of GalNAc to either Ser or Thr. This is catalysed by polypeptide-N-acetyl-galactosaminyltransferase. This produces the Tn epitope GalNAcα1-Ser/Thr. This can then be sialylated by α-2,6-sialyltransferase to give sial Tn. This disaccharide cannot be extended further. Alternatively the GalNAc can be extended to form one of the core glycan structures detailed above[63]. The core structures may be elongated by the further addition of monosaccharides, or the core can be substituted for by a terminal monosaccharide, e.g. Fuc or sialic acid.

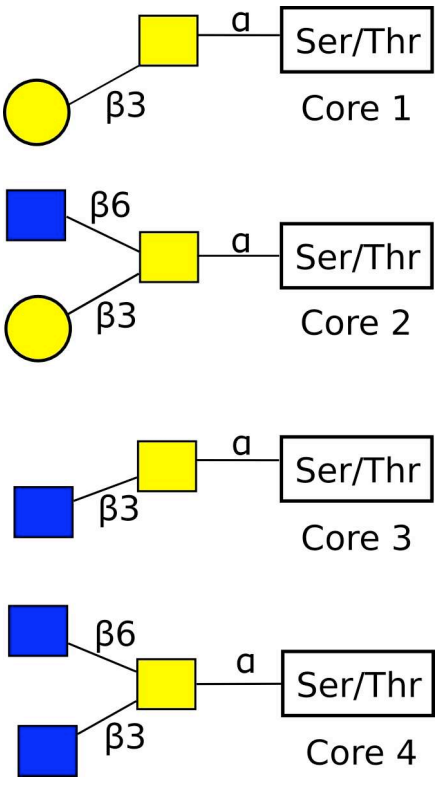


Figure 1.7. The four most common O-Glycan core structures. Symbols follow standard conventions as in reference 66. Here yellow circles are Galactose, yellow squares are GalNAc, and blue squares are GlcNAc.

The different core structures tend to be expressed in different structures in different concentrations. The core glycans can be further built upon to synthesize complex glycans of varying structure.

1.4.6.2 Functions of O-linked mucin glycans

Mucin type O-linked oligosaccharides have been implicated in a wide range of functions[71]. They take a role in the protection of the body against disease. Mucins are produced at biological membranes. The physical properties of these molecules enable them to protect the underlying epithelial cells from infection by bacteria and from extreme environments, e.g., the acidic conditions in the stomach. They provide lubrication, e.g., in the respiratory tract, and act as anti-adhesins, keeping lumen opposing surfaces from sticking together. Mucin glycans are ligands for selectins, mediating the leukocyte homing during the inflammatory response. Mucins can also act as an antifreeze.

Mucins play an important role in bacterial adhesion. Many pathogenic bacteria bind to O-linked oligosaccharides. Thus, this can hinder or occasionally enhance infection. Some species of gut bacteria use mucin proteins as a sole energy source. O-linked glycans also play an important role in sperm-egg recognition and binding. O-linked oligosaccharides are also carried by several types of haemopoietic and immune system cells. They prevent the agglutination of leukocytes both to themselves and to endothelial cells. There are dramatic changes to the glycosylation of T cells during maturation and activation, and oligosaccharides play a role in the interactions between T-cells and B-lymphocytes. Many less common O-glycan modifications also occur within the ER and Golgi. O-mannosylation is a common type of glycosylation in the

brain of mammals and important for binding laminin to the extra-cellular matrix. Alpha linked mannosylation is a common glycosylation of proteins. Initially mannose is added to Ser/Thr by a mannosyltransferase, which is unique to this pathway, as are the subsequent N-acetylglucosaminyltransferases.

1.4.7 Glycosylation of cytosolic and nuclear proteins

Cytoplasmic and nuclear proteins often undergo multiple additions of β-O-GlcNAc[72]. In addition to mucin glycosylation, we also included cytosolic and nuclear glycosylation sites in our prediction experiments. So we review their structure and function here, to show the benefit of their prediction. These molecules are added as lone monosaccharides, with no further elongation. This type of modification is present across a wide variety of species, including almost all eukaryotes and protozoa, as well as fungi, plants and animals. The modification also occurs on viral proteins. These proteins are often also phosphorylated, and this modification has factors in common with phosphorylation. The O-GlcNAc modifications often occur at sites similar to those modified by phosphokinases and O-GlcNAc modifications are reversible. The composition of occupied GlcNAc sites on a given set of proteins is dynamically changing in response to cell signalling and the various stages of the cell cycle. The interplay between phosphorylation and glycosylation is important in many of the regulatory processes in the cell.

The modification of proteins with O-GlcNAc occurs post translationally and is carried out by the highly conserved enzyme β-N-acetylglucosaminyltransferase, which is itself glycosylated and phosphorylated, probably to regulate its activity. This enzyme occurs in most species and has 85% homology across species. O-GlcNAc

glycosylation is necessary for survival, even at the level of a single cell. The possible functions of this modification are varied and not well characterised. O-GlcNAc modification is vital to the function of nucleoporin proteins that mediate the transport of macromolecules in and out of the nucleus via the nuclear transport complex. The O-GlcNAc modification is essential for the recognition of nuclear transport signals, and pores deficient in GlcNAc are structurally defective.

O-GlcNAc is also associated with chromatin. The labelling of regions of chromatin with GlcNAc plays a functional role in transcription, with a dramatic reduction in O-GlcNAc modifications occurring in regions undergoing active transcription. Glycosylation also has importance in regulating translation in association with phosphorylation. O-GlcNAc is also important for the modification of structural proteins in the cytoskeleton. Due to the significance of O-GlcNAc involvement in cell process regulation, malfunction of this modification is causative in a number of disease states. The disruption of glycosylation and phosphorylation may be relevant in malignancies. The dysfunction of O-GlcNAc is also implemented in many neurodegenerative diseases and type 2 diabetes. Nuclear and cytoplasmic proteins are also modified with complex glycans. Whilst further research and analysis is still required, the existence of these complex glycans is suggested in numerous studies.[64] Other types of glycosylation, such as glycophosphatydilinositol anchors and proteoglycans, are not predicted by our glycosylation program and, thus, are not covered here.

1.4.8 Prediction of Glycosylation

Experimental methods of determining glycosylation include mass spectrometry

analysis with lectins (glycan binding proteins), NMR analysis, and, more recently, methods involving 2D gel electrophoresis and other methods analogous to those used in proteomics. The mass study of the change in expression of glycans in a cell in health and disease is known as glycomics. These methods are expensive, difficult and time consuming. It is useful to use computational prediction of the location of glycosylation sites to reduce the experimental effort required to determine the glycosylation sites in a protein and also to understand the glycosylation process itself.

1.4.8.1 Previous methods

Several prediction methods have been published previously. We give a brief overview of these methods here. Prediction of glycosylation sites from sequence information has traditionally centred around neural network methods. NetOglyc[64] predicts mucin type O-linked glycosylation sites from sequence using a neural network trained on the OGLYCBASE[73] dataset. The authors test numerous representations of the amino acid sequence and find the best results are obtained with a PSSM representation. This is used to train a two layer feed forward neural network. This was superseded by Li *et al.*[74] who use SVMs to predict the glycosylation sites. The authors develop an independent dataset from uniprot[75], where positive sites are chosen based on uniprot annotations, and negative sites are chosen at random. Three SVM models are trained based on different combinations of sequence information and information concerning the properties of the amino acids. Li *et al*. produce a significant improvement over NetOglyc. Two more recent methods were published during the completion of this work. Carega *et al*.[76] have used an SVM ensemble based method to predict O- and N-linked glycosylation and C-mannosylation. Each SVM in the ensemble is a trained on a balanced subset of the training data, the predictions being obtained from the

combined outputs of the SVM classifiers. This method improves over previous methods. CKSAAP[77] uses *k* spaced amino acid pairs as input information to predict O-linked glycosylation sites. This is similar to our method, which is presented in chapter 4. Pairwise patterns were used to train an SVM model on a dataset taken from the Swiss prot database[78]. The combination of Ser and Thr sites in the training set produced slightly better predictions than when separate training sets were used for each type of prediction. This may be due to the increase in training set size, but could possibly be due to similarities in the features of Ser and Thr glycosylation sites.

1.4.8.2 Motivation and Objectives for New Predictive Methods

There is still opportunity for improvement in the prediction of glycosylation sites. There is a limit on the accuracy that is attainable, due to the possibility of both undiscovered sites and various errors, both in the prediction and experimental data, that cannot be eliminated. However, the theoretical threshold has not yet been reached. It is for this reason that we pursue an improved prediction method by using pairwise patterns with a novel machine learning method. Another aspect to this work is that the methods produced up until now have been black box methods. This means their decision processes are hidden from the user. We seek to remedy this and to provide some biological comprehension of the prediction model produced by random forest.

1.5 Thesis Overview

In chapter two, a discussion is presented of the machine learning algorithms and statistical methods used in chapters three and four. We give an overview of the methods we will be using, along with some background and the reasons for choosing

these methods. In chapter three, the relationship between dihedral angles and secondary structure of proteins is explored. After producing a secondary structure predictor with the idea of using this information to improve dihedral angle predictions, we concentrate on the prediction of Φ and Ψ dihedral angles using SVR, both with and without the prediction of secondary structure. Our hypothesis here is that SVR, a machine learning algorithm as yet untried for dihedral angle prediction, will offer improvement in accuracy. We also hypothesise that predicted secondary structure information will improve the predictions still further. Chapter 4 is concerned with techniques to improve the accuracy of glycosylation prediction, by using random forests combined with pairwise pattern information, and with the extraction of biologically meaningful rules from the random forest. We reason that it is likely that the amino acids have dominant influence over which residues are glycosylated even in the absence of a consensus sequence. Therefore, we generate pairwise patterns and use them to generate information about whether a given residue is likely to be glycosylated with the hypothesis that this will improve prediction accuracy when combined with a machine learning algorithm. For our machine learning algorithm we choose random forest, which has not been used before for predicting glycosylation sites, but has a good track record of sequence based prediction. The hypothesis is that this will improve prediction accuracy. Our conclusions are offered for consideration in chapter 5.

1.6 References

1. Wallace IM, Blackshields G, and Higgins DG. Multiple sequence alignments. *Curr. Opin. Str. Bio.* 2005 **15**:261-266.
2. Thompson JD, Higgins DG, and Gibson TJ. CLUSTAL improving the

sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 1994 **22**:4673-4680.

3. Dayhoff MO, Editor of Atlas of Protein sequence and Structure Vol. 5 suppl. 3 Nat. Biomed. Res. Found., Washington, DC.
4. Henikoff S and Henikoff JG. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.* USA 1992 **89**:10915-10919.
5. Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A Lopez R, Thompson JD, Gibson TJ, and Higgins DG. ClustalW and ClustalX version 2.0, *Bioinformatics* 2007 **23**:2947-2948.
6. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, and Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 1997 **25**:3389-3402.
7. Altschul SF, Gish W, Miller W, Myers EW, and Lipman DJ. Basic local alignment search tool, *J. Mol. Biol.* 1990 **215**:403-410.
8. Soding J, Biegert A, and Lupas AN. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Res.* 2005 **33**:W244-W248.
9. Rangwala, H and Karypis G. Incremental window-based protein sequence alignment algorithms. *Bioinformatics* 2006 **23**:e17-e23.
10. Gambin A, and Wojtalewicz P. CTX-BLAST: context sensitive version of protein BLAST *Bioinformatics* 2007 **23**:1686-1688.
11. Lee ML, Chan MK, and Bundschuh R. Simple is beautiful: a straightforward approach to improve the delineation of true and false positives in PSIBLAST

searches. *Bioinformatics* 2008 **24**:1339-1343.

12. Przbylski D, and Rost B. Powerful fusion: PSI-BLAST and consensus sequences. *Bioinformatics* 2008 **24**:1987-1993.
13. Anfinsen CB. Principles that govern the folding of folding chains. *Science* 1973 **181**: 223-230.
14. Kabsch W and Sander C. A dictionary of protein secondary structure. *Biopolymers*, 1983, **22**:2577-2637.
15. Ramachandran GN, Ramakrishnan C, and Sasisekharan V. Stereochemistry of polypeptide chain configurations. *J. Mol. Biol.* 1963 **7**:95-99.
16. Pauling L, and Corey RB. Configurations of polypeptide chains with favoured orientations around single bonds. *Proc. Nat. Acad. Sci.* 1951 **37**:729-740.
17. Garnier J, Osguthorpe DJ, and Robson B. Analysis and implications of simple methods for prediction of α-helical and β-structural regions in globular proteins. *J. Mol. Biol.* 1978 **120**:97-120.
18. Kloczkowski A, Ting KL, Jernigan RL, and Garnier J. Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence. *PROTEINS: Struct. Funct. Genet.* 2002 **49**:154-166.
19. Rost B and Sander C. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.* 1993 **232**:584-599.
20. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 1999 **292**:195-202.
21. Bystroff C, Thorsson V and Baker D. HMMSTR: a hidden markov model for local sequence-structure correlations in proteins *J. Mol. Biol.* 2000, **301**:173-190.

22. Cuff JA and Barton GJ. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *PROTEINS: Struct. Funct. Genet.* 2000 **40**:502-511.

23. Pollastri G, Pryzybylski D, Rost B and Baldi P. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *PROTEINS: Struct. Funct. Genet.* 2002, **47**:228-235.

24. Jaing F. Prediction of protein secondary structure with a reliability score estimated by local sequence clustering. *Prot. Eng.* 2003, **16**:651-657.

25. Hvidsten TR, Kryshtafovych A and Fidelis K. Local descriptors of protein structure: A systematic analysis of the sequence–structure relationship in proteins using short- and long-range interactions. *PROTEINS: Struct. Funct. Bioinf.* 2008, **75**:870-884.

26. Wood MJ, and Hirst JD. Protein secondary structure prediction with dihedral angles. *PROTEINS: Struct. Funct. Bioinf.* **59**:476-481.

27. Montgomerie S, Sundararaj S, Gallin WJ and Wishart DS. Improving the accuracy of protein secondary structure prediction using structural alignment. *BMC Bioinformatics* 2006, **7**:301.

28. Birzele F and Kramer S. A new representation for protein secondary structure prediction based on frequent patterns. *Bioinformatics* 2006 **22**:2628 – 2634.

29. Karypis G. YASSPP: Better kernels and coding schemes lead to improvements in protein secondary structure prediction. *PROTEINS: Struct. Funct. Bioinf.* 2006 **64**:575-586.

30. He J, Hu HJ, Harrison R, Tai PC and Pan Y. Rule generation for protein secondary structure prediction with support vector machines and decision tree. *IEEE Trans. NanoBio.* 2006 **5**:46-53.

31. Zhong W, He J, Harrison R, Tai PC, and Pan Y. Clustering support vector machines for protein local structure prediction. *Expert Systems with Applications* 2007 **32**:518-526.

32. Won KJ, Hamelryck T, Prugel-Bennett A, and Krogh A. An Evolutionary method for learning HMM structure: prediction of protein secondary structure. *BMC Bioinformatics* 2007 **8**:357.

33. Ouali M and King RD. Cascaded multiple classifiers for secondary structure prediction *Prot. Sci.* 2000 **9**:1162-1176.

34. Yao XQ, Zhu H, and She ZS. A dynamic Bayesian network approach to protein secondary structure prediction. *BMC Bioinformatics* 2008 **9**:49.

35. Zimmerman O and Hamsmann UHE. Support vector machines for prediction of dihedral angle regions. *Bioinformatics*, 2006 **22**:3009-3015.

36. Dor O, and Zhou Y. Real-SPINE: An integrated system of neural networks for real-value prediction of protein structural properties. *PROTEINS: Struct. Funct. Bioinf.* 2007 **68**:76-81.

37. Xue B, Dor O, Faraggi E, and Zhou Y. Real-value prediction of backbone torsion angles. *PROTEINS Struct. Funct. Bioinf.* 2008, **72**:427-433.

38. Kurgan LA, Stach W and Ruan J. Novel scales based on hydrophobicity indices for secondary protein structure. *J. Theor. Biol.* 2007 **248**:354-366.

39. Rost B and Sander C. Conservation and prediction of solvent accessibility in protein families. *PROTEINS: Struct. Funct. Genet.*1994 **20**:216-226.

40. Li X and Pan MX. New method for accurate prediction of solvent accessibility from protein sequence. *PROTEINS: Struct. Funct. Genet.* 2001, **42**:1-5.

41. Yuan Z, Burrage K, and Mattick JS. Prediction of protein solvent accessibility

using support vector machines. *PROTEINS: Struct. Funct. Genet.* 2002, **48**:566-570.

42. Pollastri G, Baldi P, Fariselli P and Casadio R. Prediction of coordination number and relative solvent accessibility in proteins. *PROTEINS: Struct. Funct. Genet.* 2002, **47**:142-153.
43. Ahmad S, Gromiha MM and Sarai A. Real value prediction of solvent accessibility from amino acid sequence. *PROTEINS: Struct. Funct. Genet.* 2003, **50**:629-635.
44. Kim H and Park H. Prediction of protein relative solvent accessibility with support vector machines and long-range interaction 3D local descriptor. *PROTEINS: Struct. Funct. Bioinf.* 2004, **54**:557-562.
45. Wang JY, Lee HM and Ahmad S. Prediction and evolutionary information analysis of protein solvent accessibility using multiple linear regression. *PROTEINS: Struct. Funct. Bioinf.* 2005, **61**:481-491.
46. Qin S, He Y and Pan XM. Accessibility with an improved multiple linear regression method. *PROTEINS: Struct. Funct. Bioinf.* 2005, **61**:473-480.
47. Xu Z, Zhang C, Liu S and Zhou Y. QBES: Predicting real values of solvent accessibility from sequences by efficient, constrained energy optimization. *PROTEINS: Struct. Funct. Bioinf.* 2006, **63**:961-966.
48. Gianese G and Pascarella S. A consensus procedure improving solvent accessibility prediction. *J. Comput. Chem.* 2006, **27**:621-626.
49. Gianese G, Bossa F and Pascarella S. Improvement in prediction of solvent accessibility by probability profiles. *Prot. Eng.* 2003 **16**:987-992.
50. Wang J-Y, Lee HM and Ahmad S. SVM-Cabins: prediction of solvent accessibility using accumulation cutoff set and support vector machine.

PROTEINS: Struct. Funct. Bioinf. 2007, **68**:82-91.

51. Adamczak R, Porollo A, and Meller J. Accurate prediction of solvent accessibility using neural networks-based regression. *PROTEINS Struct. Funct. Bioinf.* 2004, **56**:753-767.
52. Finn RD, Tate J, Mistry J, Coggill PC, Sammut JS, Hotz HR, Ceric G, Forslund K, Eddy SR, Sonnhammer EL and Bateman A. The Pfam protein families database. *Nucl. Acids Res.* 2008, **36**:D281-D288.
53. Elman JL. Finding structure in time, *Cognitive Science*, 1990, **14**:179-211.
54. Walsh CT, Garneu-Tsodikova S and Gatto Jr GJ. Protein posttranslational modifications: the chemistry of proteome diversifications. *Angew. Chem. Int. Ed.* 2005, **44**:7342-7372.
55. Blom N, Sicheritz-Ponten T, Gupta R, Gammeltoft S and Brunak S. Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence. *Proteomics* 2004, **4**:1633-1649.
56. Eberharter A and Becker PB. Histone acetylation: a switch between repressive and permissive chromatin. *EMBO reports*, 2002, **3**:224-229.
57. Boutin JA. Myristoylation. *Cell Signal* 1997, **1**:15-35.
58. Bijlmakers M-J and Marsh M. The on-off story of protein palmitoylation. *Trends Cell Biol.* 2003, **13**:32-41.
59. Haglund K and Dikic I. Ubiquitylation and cell signalling. *EMBO J.* 2005, **24**:3353-3359.
60. Bedford MT and Richard S. Argenine methylation: an emerging regulator of protein function. *Molecular Cell,* 2005, **18**:263-172.
61. Zhang FL, and Casey PJ. Protein prenylation: molecular mechanisms and functional consequences. *Annu. Rev. Biochem.* 1996, **65**:241-269.

62. Bardwell JCA. Building bridges: disulphide bond formation in the cell. *Molecular Microbiology* 1994, **14**:199-205.

63. Varki A, Cummings RD, Esko JD, Freeze HH, Stanley P, Bertozzi CR, Hart GW, and Etzler ME. (Editors) Essentials of glycobiology 2nd edition, Cold Spring Harbour Laboratory Press, Cold Spring Harbour, New York, 2009.

64. Julenius K, Mølgaard A, Gupta R and Brunak S. Prediction, conservation, analysis, and structural characterization of mammalian mucin-type O-glycosylation sites. *Glycobiology* 2005, **15:**153-164

65. Breton C, Oriol R, and Imberty A. Conserved structural features in eukaryotic and prokaryotic fucosyl transferases. *Glycobiology* 1998 **8**:87-94.

66. Betanbaugh MJ, Tomiya N,Narang S, Hsu JTA and Lee YC. Biosynthesis of Human type N-Glycans in heterogenous systems. *Curr. Opin. Struct. Biol* 2004 **14**:601-606.

67. Helenius A and Aebi M. Intracellular functions of N-linked glycans. *Science* 2001 **291**:2364-2369.

68. Lis H and Sharon N. Protein glycosylation: structural and functional aspects. *Eur. J. biochem.* 1993, **218**:1-27.

69. Hanisch F-G. O-Glycosylation of the mucin type. *Biol. Chem.* 2001, **382**:143-149.

70. Silverman HS, Parry S, Sutton-Smith M, Burdick MD, McDermott K, Reid CJ, Batra SK, Morris HR, Hollingsworth MA, Dell A, and Harris A. In vivo glycosylation of mucin tandem repeats. *Glycobiology* 2001, **11**:459-471.

71. Seregni E, Botti C, Massaron S, Lombardo C, Capobianco A, Bogni A, Bombardieri E. Structure function and gene expression of epithelial mucins. *Tumori*, 1997 **83**:625-632.

72. Snow DM, Hart GW, Nuclear and cytoplasmic glycosylation. *Int. Rev. Cytol.* 1998, **181**:43-74.

73. Gupta R, Birch H, Rapacki K, Brunak S and Hansen JE. O-GLYCBASE version 4.0: a revised database of O-glycosylated proteins. *Nucleic Acids Res.* 1999, **27**:370-372.

74. Li S, Liu B, Zeng R, Cai Y and Li Y. Predicting O-glycosylation sites in mammalian proteins by using SVMs. *Comput. Biol. Chem.* 2006, **30:**203-208.

75. Bairoch A, Apweiler R, Wu CH, Barker WC, Boekmann B, Ferro S, Gasteiger E, Huang H, Lopez R, Magrane M, Martin MJ, Natale DA, O'Donovan C, Redaschi N and Yeh L-SL. The universal protein resource (UniProt). *Nucleic Acids Res.* 2005, **33**:D154-D159.

76. Caragea C, Sinapov J, Silvescu A, Dobbs I and Honaver V. Glycosylation site prediction using ensembles of support vector machines classifiers. *BMC Bioinformatics* 2007, **8:**438.

77. Chen YZ, Tang YR, Sheng ZY, and Zhang Z. Prediction of mucin-type O-glycosylation sites in mammalian proteins using the composition of k-spaced amino acid pairs. *BMC Bioinformatics* 2008, **9:**101.

78. Boeckmann B, Bairoch A, Apweiler R, Blatter M-C, Estreicher A, Gasteiger E, Martin MJ, Michoud K, O'Donovan C, Phan I, Philbout S, and Schneider M. The Swiss prot protein knowledge base and its supplement TrEMBL in 2003. *Nucleic Acids Res.* 2003, **31**:365-370.

Chapter 2: Machine learning algorithms

2.1 Introduction

The choice of machine learning algorithms that could be used to solve the bioinformatics problems outlined in chapter 1 is vast. In this chapter, we give some background to the machine learning algorithms used in the thesis and outline the algorithms themselves. We begin by defining the basic problems of classification and regression. We move on to introduce decision trees, which form an integral part of two of the machine learning methods used in this thesis. After this we introduce SVR and finally the rule extraction method trepan. Random forest is an ensemble method, which uses a group of decision trees to perform classification. It is used extensively to predict glycosylation sites in chapter 4. It was selected as a method as yet untried for the prediction of glycosylation sites, which promised to be good for classification based on sequence data. Our approach of using pairwise patterns requires the forest to be able to handle mixed data, which it indeed does. For these reasons we hypothesise that the use of random forest with pairwise pattern information will give better accuracy over the state of the art. Another reason for its selection was the possibility of parallelisation of the method, perhaps with each tree on a different processor, although in practice an ensemble of forests was used with each forest trained on a different cluster node.

Kernel machine learning methods and, in particular, SVM algorithms for regression are used in chapter 3 of this thesis for prediction of protein backbone dihedral torsion angles from the amino acid sequence. Here we introduce both SVM for classification and regression. One leads naturally on to the other and indeed it is hard to understand one without the other. We chose to use real value, rather than categorical, prediction.

Categorical predictions are limited to a hard margin of classification between a relatively small number of divisions, which typically mirror secondary structure types. In contrast, dihedral angles are flexible throughout a wide range of values, and it is valuable to find as close to the real value as possible for use in 3D structure prediction. This also allows for regions of the protein, which do not have a well-defined structure, and may not be well predicted by a secondary structure based assignment of dihedral angle categories. Having selected regression, it became necessary to choose a method for performing that regression that was both untried for dihedral angle prediction, and had the potential to improve accuracy. Previous methods all use neural networks. SVR has been shown to have comparable or better accuracy to neural networks, and had not been tested for prediction of dihedral angles. Therefore, we hypothesised that SVR will improve the accuracy of prediction of dihedral angles over the state of the art.

Returning to chapter 4, the random forests method is a black box method and its decision process is not human interpretable. It is beneficial to view the decision making process of a machine learning algorithm for two reasons. Firstly, meaningful biological rules can be extracted from the decisions made by the algorithm. Such rules may be testable experimentally and may yield previously undiscovered biological principles. The second reason is that any mistakes which are reducing the accuracy of the machine learning algorithm may be highlighted, allowing for future improvement. For this purpose we selected trepan. Trepan was chosen as an algorithm which can easily be connected to the random forest in order to interpret its decision process. The trepan algorithm produces a decision tree based on the training data and the predictions of the random forest, thus giving a clear set of rules from which the

predictions can be interpreted.

2.2 Classification

There are many different types of machine learning algorithm used for data mining and bioinformatics, but all have the same basic premise of using a set of known examples to obtain information about new data or new information from existing data. The known examples are usually referred to as the training set. The model may be evaluated with a test set of unlabelled or unknown instances. Examples are often represented as a vector containing features that describe a given example or instance. The data can either be labelled or unlabelled.

Machine learning algorithms can be divided into supervised and unsupervised learning[1]. Supervised learning typically consists of relating a series of attributes of the data to a specific class or numerical value known as a label of that specific instance. This relationship is termed the model and such methods are often called prediction methods. Unsupervised learning, in contrast, refers to methods that group instances without any reference to a pre-specified label. This area covers methods such as clustering. Supervised learning methods include, amongst others, decision trees, neural networks, and SVR. Supervised learning methods can further be divided into classification and regression methods. Classification methods fit each instance to a series of discrete classes based on a model derived from known examples. Regression, in contrast, fits the data to a real value distribution.

Classification is an extension of concept learning[1], whereby a set of examples is used to learn the general definition for a concept. A concept is defined as a category or

description, e.g., of an object or a set of objects such as the concept of an animal. Within concept learning a possible goal could be to learn what constitutes an animal, by learning from features present in a selection of creatures. Concept learning is restricted to a function mapping the set of possible examples to the Boolean set {True, False}. Classification is not restricted to the Boolean set, and maps the set of possible examples for a given problem to a predefined set of class labels. Equations in this section and the sections up until section 2.4 are adapted from reference 2, unless otherwise stated.

For a training set S consisting of attributes $A=\{a_1, a_2, a_3, a_4....a_n\}$ and a target attribute Y from an unknown fixed distribution D over the labelled instance space, the goal is to induce a classifier, which has the minimum generalisation error. The generalisation error,[2] E, is the rate of misclassification over the distribution D defined for nominal attributes:

$$E(DT(S),D) = \sum_{\langle x,y \in U \rangle} D(x,y) \cdot L(y,DT(S)(x)) \tag{2.1}$$

where U is the labelled instance space, defined as the Cartesian product of all input attribute domains and the target attribute domain. $DT(S)$ is the decision tree for training set S. L is the zero one loss function[2]:

$$L(y,DT(S)(x)) = \begin{Bmatrix} 0 \, if \; y = DT(S)(x) \\ 1 \, if \; y \neq DT(S)(x) \end{Bmatrix} \tag{2.2}$$

The sum operator is replaced with integration for numeric attributes.

2.3 Decision trees

Decision trees[2] are one type of machine learning technique used most commonly for classification, although their use as a regression method is also possible. For the purposes of this work, we shall focus on decision trees used for classification, or classification trees. Classification trees are predictive models, which have been used in a variety of fields. The classification tree divides the data among a series of classes based on a number of decisions. Each decision is carried out based on one or more rules, such as whether a specific amino acid is present or not.

A decision tree is structured hierarchically (figure 2.1). A decision tree is made up of a number of nodes, paths and leaves.

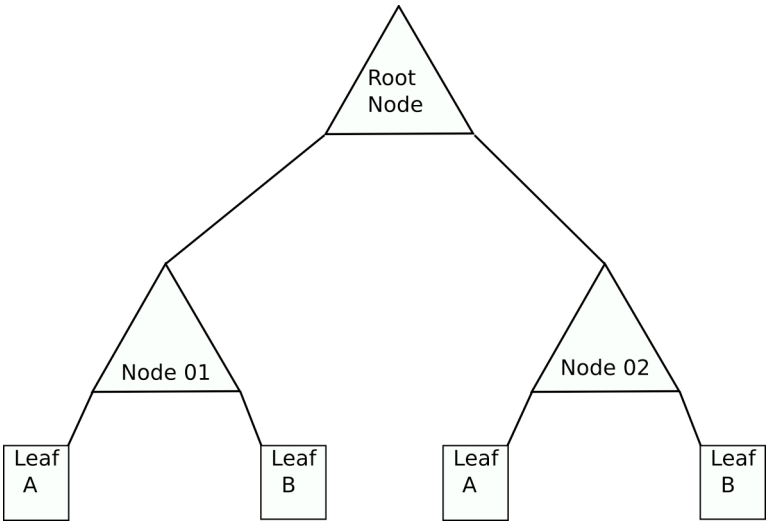


Figure 2.1. A general schematic of a simple decision tree. Here nodes are numbered and split the data according to some rule. A leaf node designates the class, either A or B. The root node is the starting point of the decision tree and performs the intitial split of the data. Paths between nodes are shown as unbroken lines.

Each of the nodes in the tree has a number of possible paths leaving it, which either join to other nodes further down the tree known as child nodes, or to a leaf node, which is a node assigning a specific class to an instance. The first node in the tree is known as the root node. Instances are classified as one of the designated classes by

sequentially following a path through the tree based on the attributes of the data in relation to the rules at each node, until a leaf node is reached, when the data is labelled as belonging to the corresponding class.

2.3.1 Decision tree induction

An inducer is an algorithm, which is used to create a decision tree from a set of examples known as training data. Each instance of the training data is represented as a vector of attributes labelled with a classification. The inducer takes the training data and forms a model that describes the relationship between the attributes and labels of the data. Construction of the decision tree begins at the root node. At each new node in the tree the data is divided based upon the features of the data. This process is continued iteratively until one of the stopping criteria is reached (see later). There are many ways to find the optimal split in the data at each node. These include both univariate and multivariate splitting criteria. Univariate splitting criteria are dependent on one attribute of the data. Thus, univariate decision tree inducers are concerned with identifying the best attribute with which to split the data. The approaches can be classified into criteria based on impurity, with or without normalisation, binary criteria, and those criteria based on information or distance.

2.3.2 Splitting Criteria

2.3.2.1 Impurity based criteria

One of the aspects of a decision tree, which is important for choosing a decision tree algorithm is the method used at each node to decide where to split the data in order to share it between the two child nodes. Here we give an overview of some of the basic types of splitting criteria along with some examples, which are relevant to the methods

used in this thesis. In the next section, we will introduce some common decision tree algorithms the principles of which are relevant to the trepan algorithm described later in this chapter and used in chapter 4 for rule extraction. After introducing some example decision trees, we see how they can be grouped into ensembles, before introducing random forest, one of the principal methods used in this work.

The splitting of the data at a given node in the tree can be chosen based on the purity of information obtained by the split. If the split is completely pure, then the measure is 1 and 0 if all the components are equally distributed (i.e. if the split is arbitrary). Information gain[3] uses entropy as a measure of the impurity in the data. Given a training set S with discrete attributes $a_{1\ldots i}$, a target attribute y with possible outcomes $c_{1\ldots i}$ and σ is a selection of attributes chosen from S, the information gain is calculated as:

$$InformationGain(a_i,S) = Entropy(y,S) - \sum_{v_{i,j}\in dom(a_i)} \frac{\left|\sigma_{a_i=v_{i,j}} S\right|}{|S|} \cdot Entropy\left(y,\sigma_{a_i=v_{i,j}} S\right) \quad (2.3)$$

where:

$$Entropy(y,S) = \sum_{c_j\in dom(y)} -\frac{\left|\sigma_{y=c_j} S\right|}{|S|} \cdot \log_2 \frac{\left|\sigma_{y=c_j} S\right|}{|S|} \quad (2.4)$$

The Gini index[2] takes as its impurity measure the divergence between the target attributes.

$$Gini(y,S) = 1 - \sum_{c_j\in dom(y)} \left(\frac{\left|\sigma_{y=c_j} S\right|}{|S|}\right)^2 \quad (2.5)$$

This leads to an evaluation criterion for the selection of the best attribute a_i on which to split the data, which is:

$$GiniGain(a_i,S) = Gini(y,S) - \sum_{v_{i,j} \in dom(a_i)} \frac{\left|\sigma_{a_i = v_{i,j}} S\right|}{|S|} \cdot Gini\left(y,\sigma_{a_i = v_{i,j}} S\right) \tag{2.6}$$

The Gini index is used in the CART method described below and is also related to the method employed by trepan. The nature of impurity based criteria means that they favour attribute domains with a large number of values. This bias can reduce the accuracy of a decision tree, as an attribute with many values will show the highest information gain, when in fact it may not give the greatest accuracy. Thus, it is useful to normalise the impurity measure. Normalising the information gain[4] as follows gives rise to the gain ratio:

$$GainRatio(a_i,S) = \frac{InformationGain(a_i,S)}{Entropy(a_i,S)} \tag{2.7}$$

The attribute with the best ratio gain is selected. This measure can unduly favour attributes with a very small denominator. So it has been suggested that a two stage approach is used where first the information gain is calculated for all attributes allowing consideration of only those attributes that have better than or equal performance to the mean information gain, and thus allowing selection of the best gain ratio. The gain ratio is used as part of the splitting algorithm of trepan described in section 2.6.2.3 below.

2.3.2.2 Binary Criteria

These criteria are designed for binary decision trees and are based on the division of the attribute into two sub domains. The twoing criterion[5] is an equivalent of the Gini index for binary decision trees. It can be employed when the attribute domains are relatively wide and the Gini index would be inappropriate. It is defined as:

$$twoing\left(a_i, dom_1(a_i), dom_2(a_i), S\right) =$$
$$0.25 \cdot \frac{\left|\sigma_{a_i \in dom_1(a_i)} S\right|}{|S|} \cdot \frac{\left|\sigma_{a_i \in dom_2(a_i)} S\right|}{|S|} \cdot \left(\sum_{c_i \in dom(y)} \left| \frac{\left|\sigma_{a_i \in dom_1(a_i) ANDy = c_j} S\right|}{\left|\sigma_{a_i \in dom_1(a_i)} S\right|} - \frac{\left|\sigma_{a_i \in dom_2(a_i) ANDy = c_j} S\right|}{\left|\sigma_{a_i \in dom_2(a_i)} S\right|} \right| \right)^2 \tag{2.8}$$

where dom_1 and dom_2 are sub-domains of the attribute a. The twoing criteria is used in the CART algorithm to form part of its splitting process.

Other binary criteria include the orthogonal[2] criterion and the Kolmogorov-Smirnov criterion[6]. Many splitting criteria have been considered in the literature and those described above represent a selection of the most commonly employed. Each criterion performs better in some circumstances than others and whilst it is possible to select the splitting criteria for a given problem, performance gain may be minimal.

2.3.3 Stopping criteria and pruning

A decision tree requires a stopping criterion to limit it to a set number of nodes, i.e., to define the end of the process of inducing the decision tree. Stopping criteria have a number of common forms: the number of nodes has reached a maximum, the depth of the tree has reached a preset limit or the best splitting criterion is below a given

threshold. Other possibilities relate to the minimum number of instances reaching each potential child node, or to whether all instances in the training data are able to be given a classification.

It is usual for some nodes to be pruned away from the tree after induction. Dependent on the stopping criterion, a tree may end up being large and over-fitted to the training data or conversely small and under-fitted. A pruning method can allow the decision tree to over-fit and then to remove sections of the tree, which are not improving the generalisation accuracy of the tree. Pruning is also carried out to reduce the complexity of a decision tree while maintaining its accuracy, or to produce a compact description of a concept or classification. The tree is often measured against the original tree, in order to assess the loss in accuracy incurred by pruning.

A simple pruning method is error pruning[3]. In a bottom to top traversal of the nodes in a tree, the algorithm checks to see if replacing a node with the most frequent classification will reduce the generalisation error of the tree. If the accuracy is not reduced, then the node is pruned. This process is repeated until no pruning of the tree can be carried out without reducing accuracy. A similar method, minimum error pruning[7] examines the l-probability error rate estimation before and after pruning and if pruning the node does not affect the error rate then it is accepted.

Many other methods have been applied to the pruning of decision trees, such as pessimistic pruning[4], error based pruning[4] and minimum description length pruning[8].

2.3.4 Some decision tree algorithms

Below we discuss some of the more popular decision tree algorithms that have been developed. This is by no means an exhaustive survey. Those examples given below are designed to give an overview of some of the major types of decision tree. One of the simplest decision tree induction algorithms[9], ID3 uses information gain as a splitting criterion. As a stopping criterion, the tree is considered complete when either each instance belongs to a single class or when the information gain from continuing is no longer greater than zero. No pruning of the tree is conducted, and no support is available for handling anything other than discrete attributes.

The C4.5 algorithm[4] is an improvement on ID3 by the same author. It is similar to ID3. However, the splitting criterion is the gain ratio and the stopping criterion is based on the number of instances. When the number of instances available for splitting falls below a certain threshold then no further splitting is carried out. C4.5 also incorporates error-based pruning after the growing of the decision tree, and supports numeric attributes.

Developed by Brieman *et al*., Classification and regression trees (CART)[5] constructs binary trees using twoing criteria to split at each node producing exactly two child nodes. The initial tree is pruned using cost complexity pruning. CART can calculate the cost of misclassification in the induction of the tree when appropriate information is provided. CART can also generate regression trees, whereby a real number is predicted at each leaf node rather than a class. In splitting for regression trees CART seeks to minimize the prediction squared error (least squared deviation). At each leaf node the predicted value is based on the weighted mean of each node. There are many

other algorithms for decision tree classification, and it is not appropriate to discuss them all here. However, more detail of algorithms such as CHAID[10] and QUEST[11] can be found in the references listed below, particularly Rokach and Maimon[2].

2.4 Ensembles of decision trees

2.4.1 Decision forests: general principles

Decision forests are ensembles of decision trees[2]. An ensemble method is one that combines a number of models of either the same or of different types. Each of these models is trained to solve the same problem. Ensemble methods improve accuracy, and through distributed computing can allow for larger datasets to be considered by spreading the ensemble over a number of computers. Whilst there are many methods for ensemble learning, typically many of them have the same building blocks. In most cases labelled training data is used for the training of the ensemble. Typically all the elements of the ensemble are trained on data taken from the same training set. There is an inducer, which is the algorithm that develops each classifier based on the training set. There is an ensemble generator, which is responsible for generating a varied set of classifiers. Finally, there is a combination protocol, by way of which the various classifiers in the ensemble are combined to produce predictions. There are certain properties of ensemble classifiers, which must be considered in order to achieve an effective ensemble. These include, whether the classifiers interact with each other, how the individual classifiers are combined, the method by which variety between the classifiers is generated, the size of the ensemble, which inducer is used and whether the same or varied inducers are used throughout the ensemble, and what proportion of overlap there is in the training data presented to the individual classifiers.

There should be some diversity among the classifiers. If the classifiers are identical, the result will be the same as for an individual classifier. Many methods exist for ensuring there is sufficient diversity. Here, we give a brief overview of the different methods and some specific examples. However, we begin with the closely related property of dependence between classifiers. A decision forest is said to be dependent if the trees within it interact with each other and independent if they do not. Dependent classifiers fall into two main types: model guided instance selection and incremental batch learning[12]. In model-guided instance selection, the training process is iterative, the training data in a given iteration being manipulated by the models generated in previous iterations. Typically, such models only learn from mis-classified instances, ignoring those instances that were correctly classified by previous iterations. Boosting, such as in Adaboost,[13] is an example of such a learning scheme. This method improves the output of a learning classifier system by repeatedly running it on various distributions of the training data and producing a composite classifier from the resulting models. Incremental batch learning simply uses the classifier generated in a previous iteration as prior knowledge for the next. The classifier generated at the final iteration is taken as the resulting trained model.

Independent methods are those in which the classifiers in the ensemble have no knowledge or interaction with each other. Each of the classifiers is typically trained on a different subset of the training data. These subsets may overlap or they may be disjoint. Independent methods have the advantage that the combination method is independent of the induction. So multiple types of classifier can easily be combined and independent decision forests can easily be run on parallel architectures. Examples of independent methods include bagging[14], wagging[15], and random forest[16].

2.4.2 Diversity

Diversity is essential to ensemble machine learning. The classifiers must be as diverse as possible, whilst remaining within the bounds of the problem being considered. The classifiers must also remain consistent with one another in order to produce meaningful results. The required diversity can be generated using a variety of methods. The training data can be manipulated, the feature space can be partitioned or each classifier can be targeted at a different subset of the problem. In addition, manipulation of the inducer itself and hybridization of the various types of inducer can be used to create diversity within an ensemble.

Manipulation of the inducer is probably the simplest way of generating diversity. The variability can often be generated by the manipulation of the parameters of the induction algorithm, for example, altering the threshold parameter in the C4.5 decision tree[4] or altering of the topology of neural networks. The starting point for training the inducer can also be altered, e.g., the initial weights of a neural network. The method used by an inducer to traverse the so called 'hypothesis space' can be varied, leading the different classifiers to develop varied hypotheses for a classification problem. This can be done by introducing random variance, or by a method such as collective performance based strategy,[17] whereby a cost penalty is introduced into the training algorithm, which encourages diversity.

The training data can be split into sub-sets, with each classifier being trained on an overlapping or disjoint sub-set. Resampling is used to generate overlapping subsets of the data. Some methods use the distribution of the training data. Others use a random distribution. Other methods, such as Adaboost[13], change the weights of the training

data, rather than sampling with replacement.

An important method of generating variety is to create new training examples based on the distribution of the training data. These examples are combined with the training data to form a new training set. The DECORATE algorithm[18] creates these examples to give maximum variance from the training data. The training is iterative, with the first iteration on the training data and subsequent iterations with the addition of artificial examples.

Variance across the ensemble can be introduced via the partitioning of the data into disjoint partitions. This is often done randomly, and overcomes the bottleneck created by the size of the data. Each classifier is trained on a disjoint sub-set, but the whole ensemble processes the total amount. Also it is possible to use clustering techniques, e.g., SVM cabins[19] partitions the data for training multiple SVMs in order to predict protein solvent accessibility. Both these approaches offer an improvement in accuracy and a way to overcome performance bottlenecks.

Rather than diversify the data or change the way it is represented, search space partitioning introduces variation by directing the classifiers in the ensemble to explore different areas of the search space. Each of these models is constructed independently, and then aggregated. The subspaces of the feature space can overlap or be disjoint and how much, if any, overlap between subspaces to allow is an important consideration. The divide and conquer approach divides the subspace into sub-sets. The instance space can be divided using either clustering techniques, such as k means clustering, to divide the space into mutually exclusive subsets, or by a hybrid classifier, such as a

naïve Bayes tree[20]. The feature sub-set selection approach manipulates the input attribute set. Each of the classifiers is given a different sub-set of the features, and thus receives a different projection of the training set. The features can be divided up by a random selection or by using reducts[2]. A reduct is the smallest set of features that can be chosen, whilst retaining the same predictive power as the whole feature set. This has the limitation of preventing the ensemble size from being larger than the feature set. A collective feature based strategy[21] is also possible, whereby after the initial random feature selection the sub-sets are refined using an iterative method, such as genetic algorithms or a hill climbing approach.

Diversity can also be generated by using several different types of classifier to form the ensemble. This approach also covers combining several classifiers with mathematical or analytical methods. The different classifiers may identify different aspects of the training data, and, therefore, this will go some way to overcoming the natural bias of each individual classifier. For example, Zhou and Jiang combine the C4.5 decision tree with neural networks[22]. They first train a neural network. This ensemble enhances the training set by adjusting the class labels and adding new examples. The new training set is used to generate a C4.5 tree. This is analogous to the trepan[23] method, described elsewhere in this thesis, in that it provides increased comprehensibility of the results.

2.4.3 The combiner

Once the individual classifiers have been generated, they must be combined in order to give the final output. Methods for this can be divided into two major categories, weighting methods, whereby each classifier is assigned a weight proportional to its

strength of classification, and meta learning approaches where a further machine learning process is used to select the best output. A majority voting approach is possibly the simplest method[2]. Each classifier is given a vote to determine class. The predicted class of a given instance is the one that is given the largest number of votes. A development of this is performance weighting,[24] where each classifier is weighted by its performance on a validation set. Other methods employ as weights the distribution of probability across the classifiers and the Bayesian posterior probability of the classifier with respect to the training set[2].

Meta combination methods use a meta learning method to select the best classification based on the results of the base classifiers. Stacking[25] combines multiple classifiers by learning from a training set, in which each instance is described by the target attribute and the predicted classifications produced by each of the base classifiers. This method can be improved by the inclusion of output probabilities for each prediction of the base classifiers. Arbiter trees[26] are a decision tree approach designed to combine classifiers. Each arbiter tree is developed to decide between two or more classifiers. For large numbers of classifiers, multiple arbiter trees may be constructed and used in a hierarchical fashion. A similar method, combiner trees[27], uses combiners instead of arbiters at each node in the tree to select the appropriate classification for a given instance. Grading[28] is a meta-learning method for deciding upon the correctness of the classifications produced by a base classifier. From each classifier, a training set is assembled of its predicted classifications attaching a new binary class, which indicates correct or incorrect with respect to that instance. Voting is carried out between the classifications of the base classifiers that are determined to be correct by the meta learner.

2.4.4 Random forest

The random forest is the principal algorithm used in this thesis for prediction of glycosylation sites. Random forest was chosen for this work as a machine learning algorithm which was untried with regard to the prediction of post translational modification, but had been used with some success in other areas of biology. It is able to take both categorical and numerical data as input, which is ideal for analysing data about pairwise patterns and sequence information. It is also fast relative compared to methods such as SVM. We chose to use random forest rather than a single decision tree, which has many of the same properties, because an ensemble method has greater accuracy than a single decision tree.

A random forest[16] is an ensemble classifier, h, composed of a number of tree-structured classifiers

$$\{h(x,Q_k),\ k=1,2....N\} \qquad (2.9)$$

where $\{Q_k\}$ are independently generated random vectors of attributes drawn from the same distribution (i.e. the distribution of the training data) and N is the total number of vectors. Each tree casts a single vote to determine class. Generally to generate a tree, a random vector Q_k is chosen, which is independent of previously chosen vectors $Q_{1...k-1}$. This vector should have the same distribution as its predecessors. The chosen vector is combined with the training data S to generate a classifier, resulting in a classifier $h(x, Q_k)$, where x is an input vector. For an ensemble, many such trees are generated, which vote to determine class.

A useful property of random forests is that increasing the number of trees does not induce over-fitting, due to the strong law of large numbers, whilst increasing the number of trees does improve accuracy. The strong law of large numbers deals with the stability of the mean of a random variable, and states that the mean will almost certainly converge as the number of samples tends to infinity, for samples chosen at random from a given distribution. The trees in a random forest are created using bagging in tandem with the selection of random features. This improves accuracy, whilst providing an internal error estimate, which can be calculated 'out of bag'. The out of bag error is the generalised estimate for a classifier trained upon a training set S from which are bootstrapped S_k training sets. Then for each instance I in the training data combine the votes from the classifiers trained on the sets S_k which do not include the instance I. This is known as the out of bag classifier, and the out of bag error is the error rate of this classifier on the training set. It is also possible to use random forests for regression. Each tree is generated in a similar fashion as to classification, but with a numerical value for the target attribute. Regression is then performed at the leaf nodes to determine a numerical result.

2.5 Kernel based machine learning

For our work on prediction of dihedral angles we wished to predict real value angles and thus required a method to perform regression (for a discussion of the rationale for choosing real value dihedral angles see chapter 1 and 3). We wanted a method that had not been tried before but was proven to be at least comparable to the state of the art methods used in previous work. SVR is comparable in accuracy to neural networks, which were used in previous work and has not been used for prediction of dihedral angles. Our hypothesis is that the use of SVR will achieve greater accuracy of

prediction of dihedral angles.

Here we introduce kernel machine learning, which is the basis for SVR, and then go on to introduce SVMs for classification and regression. We cover classification even though it is not explicitly used in this thesis as it gives an easy route to understanding the use of kernel machine learning for SVR. Originally developed by Vapnik and co-workers,[29] SVMs for both classification and regression fall into a category of machine learning algorithm known as kernel based machine learning[30]. Kernel methods are based around the idea of altering the representation of the data to fit to a particular type of method for classification. This is done by using a function, known as the kernel function, to map the input data into a higher dimensional space (figure 2.2). This feature space is such that a simple classification algorithm can be applied to classify the data, or regression can be performed to fit a function to the data. This has the additional advantage that the properties of a kernel function allow for the feature mapping to be computed using the inner product (also known as scalar or dot product) of the vectors in the input data. So machine learning can be performed without computing the feature mapping, thus dramatically reducing computational time for large datasets.

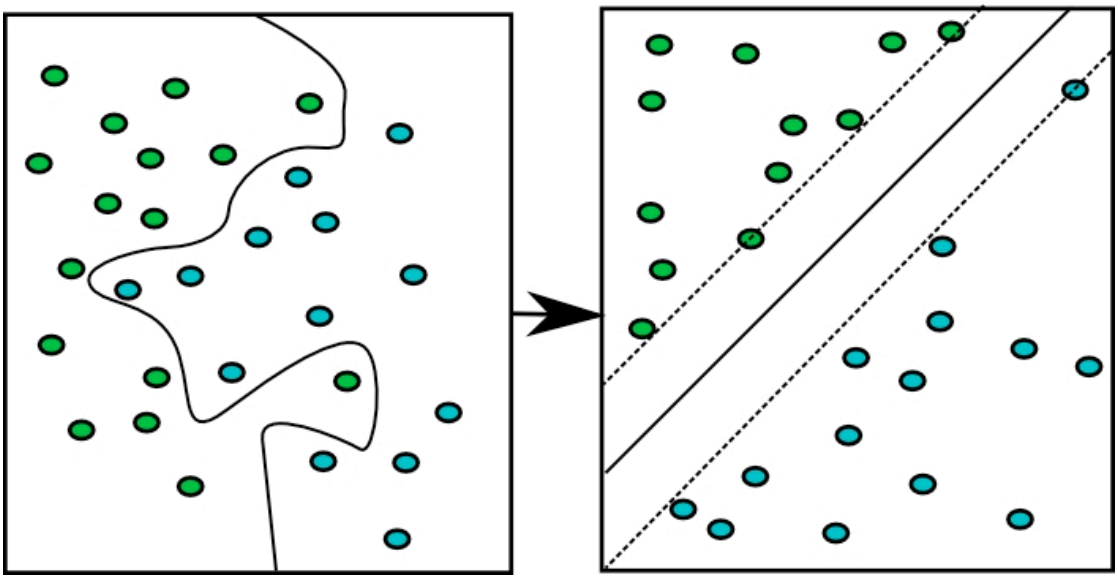


Figure 2.2 An illustration of kernel machine learning where a maximal margin hyperplane can be fitted to the data on the left after it has been raised into a higher dimensional space by a kernel function from its representation on the right.

This leads to a modular approach, where the input data is modified with the kernel function, the inner products are calculated and the machine learning algorithm is run on the data. The choice of kernel function is important and we will discuss the various types of kernel function tested in this work. Equations in this section of the work are adapted from[30] unless otherwise stated. A kernel function computes the inner product of the mapped image of two data points, x and z, in the embedding ω:

$$k(x,z) = \langle \omega(x), \omega(z) \rangle \tag{2.10}$$

for all $x, z \in X$ where X is the instance space.

2.5.1 Kernel types

Various types of kernel can be constructed which map the data into varying feature spaces. We will address the kernels that are used later in this thesis. We begin with the

linear kernel, then move onto the polynomial and Gaussian kernels, both of which have been used in numerous bioinformatics methods. The polynomial and Gaussian kernel functions were adapted from Vapnik.[29] The linear kernel is the simplest available:

$$k(x,z) = zx \tag{2.11}$$

The polynomial kernel can be constructed in various degrees as best suits the data, although the higher the degree the slower the computation time. For a vector space X of dimension n, the polynomial kernel is:

$$k_d(x,z) = \left(\langle x,z\rangle + R\right)^d \tag{2.12}$$

where R and d are parameters, with d being referred to as the degree of the kernel.

The Gaussian kernel is

$$k(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) \tag{2.13}$$

where $\sigma > 0$. σ is known as the kernel width, and is a user-defined parameter, which must be optimised for a particular dataset, along with the SVR parameters C and ε (see later in this thesis for more details on optimising for SVR). It is beyond the scope of this work to give a detailed mathematical characterisation of kernel machine learning. Here we give a summary of the SVMs for classification and regression. Full details and background are given in Shawe-Taylor and Cristianini[30].

2.5.2 Hard margin SVM

For the classification of data that has been raised into a higher dimensional feature space by a kernel function, the simplest method is to divide the data with a hyperplane in the higher dimensional feature space. In SVM classification, the hyperplane is a maximal margin hyperplane. This is defined by the data instances in each class either side of the hyperplane. The hyperplane is chosen to give the maximum margin between it and these data, which are known as support vectors. This gives a hard margin SVM classifier. However, outliers in the data may not be accurately classified by a hard margin cut off, which leads to the idea of a soft margin classifier, which takes into account possible outliers and improves classification accuracy.

2.5.3 Soft margin classifier

A soft margin classifier can be achieved by the introduction of slack variables, which allow the margin constraints to be violated. The rationale for this is that the hard margin SVM is very sensitive to noise in the data, which may make it impracticable for real world data sets. The introduction of slack creates a trade off between the slack variables and the margin of the hyperplane. This trade off is regulated by the regularisation parameter C. This parameter is optimised for each dataset on which the SVM is trained.

2.5.4 SVR

Using SVM for regression is possible using an ε-insensitive loss function, which ignores all errors below a threshold ε. The resulting band around the output function is referred to as a tube. Otherwise the procedure is similar to that of classification. A function is fitted to the data using regression, with a margin defined by the support

vectors, i.e., the error cut off ε. The threshold for the error cut off is defined by the user and is optimised along with the other parameters already described. The SVR algorithm is given a soft margin, in much the same way as for classification, by using slack variables with the trade off between the slack variables and the margin handled by the parameter C, as before. More detail on the properties of SVM for classification and regression along with pseudocode for the above algorithms is given in Shawe-Taylor and Cristianini[30].

2.6 Critical Assesment

In reviewing the various methods considered above, we aimed to select those most appropriate to use in our work. We considered the accuracy of the methods, but also factors such as availability and whether the method had been used before. It would have been less productive to re-implement methods from scratch. Where multiple methods are available with similar accuracy, the most tried and tested one was chosen. When it came to selecting a method for classification, which had not been used for glycosylation site prediction, there were many possible choices. We chose an ensemble method because of the increased accuracy over a single decision tree, but also because of the prospect of running in parallel. Random forest has been tested in a variety of machine learning problems in various fields (see chapter 4 and above for references). The choice of splitting criteria, methods for generating diversity, used in random forest are appropriate for our work. The fact that little work had been done on interpreting the model produced by random forest was also a factor in deciding on the choice. When it comes to the choice of SVR algorithm, it was important to use the soft margin version, since it gives a certain tolerance of outliers that should improve accuracy in the long run. Below we detail the methods chosen to assess the accuracy

of the work in this thesis. These were chosen to give a balanced assessment of the merits of the various methods, but were also to some extent based on the methods used to assess previous work, thus allowing a valid comparison. We then assess multiple possibilities for the method that could be used to interpret the random forest. The grounds for the final choice are given below.

2.6.1 Assessing accuracy

Throughout this work, common accuracy measures are used to assess the performance of the machine learning methods. In all the methods presented below, true positives (T_P) are those instances correctly identified as positive, true negatives (T_N) are correctly predicted negative instances, false negatives (F_N) are positive examples which are incorrectly predicted as negative and false positives (F_P) are negative examples incorrectly predicted as positive. Correctly classified instances (CCI) is a measure applied only to classification problems. This measure is the number of instances in the test set which have been assigned to the correct class as a fraction of the total, N:

$$CCI = \frac{T_p + T_n}{N} \tag{2.14}$$

This presents certain problems, as there is no indication of whether those instances are positive or negative examples and therefore no indication as to whether the prediction is a balanced one. Sensitivity (S_n), expressed here as a percentage, assesses the effectiveness at classifying positive examples:

$$S_n = \frac{T_p}{\left(T_p + F_n\right)} \times 100 \tag{2.15}$$

Specificity (S_p), also expressed as a percentage, assesses the accuracy of the classification of negative examples:

$$S_p = \frac{T_n}{\left(F_p + T_n\right)} \times 100 \tag{2.16}$$

The Matthews correlation coefficient (MCC)[31] is a measure of accuracy designed to take into account the ability of a classifier to classify correctly both positive and negative instances. It produces a value between -1 and 1, with 1 being a perfect prediction and -1 a completely incorrect prediction. 0 represents a random prediction.

$$MCC = \frac{(T_p T_n) - (F_p F_n)}{\sqrt{(T_n + F_n)(T_n + F_p)(T_p + F_n)(T_p + F_p)}} \tag{2.17}$$

The Pearson correlation coefficient is used to assess the accuracy of numerical prediction, by relating the mean and standard deviation of the predicted and observed values in the dataset:

$$r = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\left(n-1\right)\sigma_x \sigma_y} \tag{2.18}$$

where x is the observed data, y is the predicted data for n test instances $i_1, i_2, \ldots i_n$, $\bar{x}$ and $\bar{y}$ are the means of the observed and predicted data respectively and σ_x and σ_y are

the standard deviations of the observed and predicted data.

Cross-validation is a technique used for calculating the accuracy of a prediction when the amount of data available prohibits the formation of an independent test set. The data are divided up into a number of sections (typically 10). The data instances which form these sections are chosen by random sampling without replacement and the number of instances in each section is equal. One of the sections is set aside to act as the test set, and the remainder are used for training. This is repeated *N* times, where *N* is the number of sections the data were split into, with a different section of the data being used as the test set in turn. This is known as *N* fold cross validation.

Leave-one-out cross-validation is a variation on the above procedure. Instead of dividing the data into sections, the training procedure is repeated *n* times, where *n* is the number of instances in the training data. For each training cycle a different instance is left out of the training data, and is used as a test instance. This is repeated until every instance has been used as a test example. The accuracy is calculated over all of the test predictions.

2.6.2 Model Interpretability

Many machine learning algorithms, such as neural networks, SVMs and random forest, are inherently opaque. Characterised as black box methods, the decision making processes of these algorithms are not interpretable. In the case of neural networks, the weights of the network are known, but it is a non-trivial task to calculate the way the weights interact to make a prediction for a given instance. A similar state is true of SVMs. The kernel matrix cannot easily be mapped to the original data, in

such a way as to give a decision function in normal space, which again cannot be interpreted in the context of the problem. Random forest, which we use in chapter 4, is considered a black box method. The decision trees that make up the trained random forest cannot be read in the same way as, for example, a tree from an algorithm such as J48, which produces clear interpretable rules. In this work we seek to develop a method of interpreting the model produced by random forest. Many methods have been tried in order to extract meaningful information in the form of rules and interpretable decision processes from neural networks and SVM models. However, little work has been done on extracting rules from random forest. Many of the algorithms that are available for rule extraction from e.g. neural networks have been used for more than one machine learning method. Our intention is to take one such method and adapt it to extract comprehensible rules from random forest, giving us a detailed picture of the decisions made by our random forest model for glycosylation, and providing some biological insight. Since we intend to select an existing algorithm, we review examples of the major types of rule extraction method for neural networks and SVMs. A more comprehensive survey is available in references[32,33.] Not as much work has been done on the interpretability of random forest. Part of the novel work in this thesis focuses on this.

2.6.2.1 Neural networks

Of the methods below, several have also been used for interpreting SVM models as well as neural networks. Guo and Selman[34] use inductive logic programming to generate an ordered set of Horn clauses (see reference[34] for details) from an opaque machine learning model. Focusing on, neural networks, SVM, and random forest. They compare the resulting Horn clauses with a J48 decision tree, comparing each

Horn clause generated with the rule at a corresponding node in the decision tree. A sample set is created either from the training and testing data or from artificially created examples. Inductive logic programming is used in an iterative process to learn Horn clauses from this data and the responses of the machine learning algorithm being considered. This method gives a generalised framework for rule extraction and is one of the few methods to have been tested on random forest. Unfortunately, the code for this method is not freely available. So we were unable to use this method.

Another approach is to derive a decision tree, thus presenting comprehensible rules which mirror the original model. Assche and Blockeel[35] create a method that learns a single decision tree from an ensemble of neural networks. The algorithm developed is designed to avoid the use of artificially generated data as is commonly used in rule extraction methods. The decision tree is grown based on estimated probabilities, which are derived from the distribution of the prediction classes, rather than that of the training data. The stopping criterion for the tree is a non-equivalence preserving stopping criterion, whereby a node becomes a leaf node if all instances reaching it are placed in the same class. The tree is pruned after construction. This method is potentially adaptable to other machine learning algorithms. However, once again the code is not readily obtainable. A similar approach is used in CRED[36], which extracts meaningful rules from neural networks in the form of a decision tree. This method is, as many rule extraction algorithms are, bound to a specific type of learning classifier, in this case neural networks.

A different approach is to use rules to represent the network structure. The COMBO[37] algorithm generates confirming rules to explain when a neurone is switched on and

disconfirming rules to explain when a neurone is switched off. The rules are generated based on combinations of the weights of the network. The neural network considered is a feed forward network trained using the back propagation algorithm. A combination tree is generated from the network weights after they have been sorted. This tree is pruned to reduce the search space. Combinations of weights are tested against the network. Those that prove successful are used to generate rules. These rules are ordered to be representative of the network structure. This method is only implemented for a specific type of neural network. Whilst it could be adapted to other neural network architectures, adaptation to other machine learning algorithms is impossible.

A similar approach is taken by Setiono[38]. Rules are extracted from a pruned neural network using the activation weights of the hidden units of the network. The activation weights are clustered in order to discretise them. Rules are extracted from these clusters. This assumes the number of clusters is small. If there are a large number of activation thresholds then the unit may be split to form a sub network. The algorithm is applied to this sub-network. The rules generated are merged in order to find rules that directly relate the inputs and outputs of the network.

More recently, Setiono *et al*.[39] use a method based on recursion and decision trees to extract rules from back-propagation neural networks. After the network has been pruned to remove redundant nodes and connections, the network is trained on a mixed data set of discrete and continuous attributes. The classification rules are generated based upon the set of trained examples that are correctly classified by the network. Whilst discrete attributes remain, classification rules are generated, dividing the

feature space into subspaces based on the discrete attributes. This is done recursively. When only continuous attributes remain in the dataset, a hyperplane is generated using a machine learning algorithm. The authors use a neural network to divide the continuous attributes. The result is a set of classification rules. While this method is potentially adaptable and indeed produces rules in an easily comprehensible format, it is a complicated algorithm and the software is not readily available. It was judged that the effort to be gained from implementing the algorithm is not worth the perceived benefit, since other algorithms that are easier to adapt to random forest are available.

2.6.2.2 SVM

Rule extraction techniques have also been used to extract comprehensible rules from other black box methods, such as SVM classifiers. Many neural network methods are adaptable to SVMs, e.g., trepan, which uses an oracle, can easily be used for classifiers other than neural networks, since the oracle can be any binary classifier. Many other rule extraction methods such as Re-RX[39] described above, have also been applied to rule extraction from SVMs. Martens *et al*.[32] examine the applicability of some of these algorithms to the SVM rule extraction problem.

Relatively few methods have been developed specifically for SVM rule extraction. One example is SVM+prototypes[40]. This uses an iterative process, taking information from a trained SVM. The feature space is divided by a combination of the support vectors and prototype vectors obtained from clustering the data. Beginning with one prototype, each prototype is used to generate an ellipsoid. This undergoes a partitioning test to see if it is useful to divide the feature space further. If this test is negative, a rule is created based on the equation of the ellipsoid. Otherwise, further

partitioning into regions is carried out. The iterative process is completed when no further partitioning is required, or a threshold for the maximum number of regions is reached.

Chaves *et al*.[41] generate fuzzy rules to comprehend the models generated by SVM. The rules are generated from the support vectors, which are projected into a co-ordinate space. The input data used to generate the rules is used to create fuzzy sets. These fuzzy sets are used to create a rule for each support vector. The rule is of an "If … Then" format. The rules are chosen based on the fuzzy set with the highest membership for the support vector. In the case of similar degrees of membership to more than one fuzzy set, the rule chosen is the one with the best fuzzy accuracy and coverage.

The problem of dealing with higher dimensional data in the context of extracting comprehensible rules is particularly relevant to SVMs, given that they often raise the data into a high dimensional feature space. However, no rule extraction algorithms have been created which solve this problem. One possible approach is to map the data into a lower dimensional feature space, using an algorithm such as a self organising map. This approach can also be used to relate the model directly back to the training data or to a low dimensional map of the feature space. A further problem is the fusion of the domain specific knowledge of experts with rules generated from the rule extraction algorithm. The rules produced by the algorithm must be relatable to the domain knowledge. It may be that rules are not generated for things that an expert in the field would deem to be obvious. Interpretability by experts, is therefore, crucial and domain knowledge obtained from experts is still preferable in many cases[32].

2.6.2.3 Random forest

A random forest model is not easily interpretable. Whilst it is composed of decision trees, which consist of rules that are in theory interpretable, first of all the rules are hidden within the bagging of the random forest, and secondly there are a number of trees which are taken from a random sampling of the data, and thus there will necessarily be redundancy within the trees, and possibly even disparity or conflict between different rules. So it is no trivial matter to decipher the decision process of a random forest. In chapter 4 we approach the interpretability of random forest by using trepan[23] to induce a clear readable decision tree, giving an interpretable model corresponding to the decision making process of the random forest. Trepan was chosen over the other methods discussed here, because it is easily adaptable to other machine learning algorithms and is freely available, whilst maintaining a high fidelity to the original model. Other methods were either not available, not adaptable to random forest or did not have the same simplicity of trepan's *M* of *N* rules (see below). Originally designed for extracting information from neural networks by Shavelik *et al.*[23], the trepan algorithm has been implemented in matlab[42] in a fashion which allows it to be applied to a variety of machine learning methods. The trepan algorithm induces a decision tree by submitting the training data to the original algorithm itself along with some examples generated by trepan based on the attribute distribution of the training data. The original model is referred to as the oracle. Its responses are used to induce a decision tree, which has a high fidelity with respect to the original model. Fidelity is the degree to which the predictions of the decision tree resemble those of the oracle. Trepan grows decision trees in a best first manner. It expands the node of the tree that has the greatest potential to increase the fidelity of the tree. Trepan chooses the best split at a given node using the gain ratio (section

2.3.2.1) combined with a hill climbing approach to select the best *M* of *N* rule. The *M* of *N* rules trepan uses are simple "if then" statements, whereby if *M* out of *N* conditions are met then the rule returns true, otherwise the rule returns false. The stopping criteria for trepan are two fold. The expansion of a node is halted if all examples reaching that node fall into a single class. The expansion of the tree is stopped either when all nodes reach this state or when a specified maximum number of nodes is reached. Trepan also deals with one particular problem of decision trees. Often very few instances of the training data can reach a given node in the tree, making the split or classification at a given node somewhat arbitrary. In order to overcome this shortcoming, trepan generates its own instances following the pattern of attributes in the training data.

This chapter has explained the rationale behind our choices of machine learning algorithm for use in this work. We presented the background theory behind each algorithm and examples, whilst providing a rationale for our final choices. The next chapter is the first major section of the thesis on predicting real value dihedral angles and protein secondary structure using the SVR algorithms outline here. Then we progress to our work on prediction of glycosylation sites in chapter 4 using random forest and interpreting the model using trepan.

2.7 References

1. Witten IH, and Frank E. Data mining: practical machine learning tools and techniques, 2nd edition. Morgan Kaufmann, San Francisco, 2005.
2. Rokach L, and Maimon O. Data Mining With Decision Trees Theory and Applications. Series in machine perception and artificial intelligence Vol. 69,

World Scientific Publishing Co, Pte. Ltd., Singapore, 2008.

3. Quinlan JR. Simplifying decision trees, *Int. J. Man-Mach. Stud.* 1987, **27**:221-234.

4. Quinlan JR. C4.5 Programs for Machine Learning, Morgan Kaufmann, Los Altos, 1993.

5. Breiman L, Friedman J, Olshen R and Stone C. Classification and regression trees. Wadsworth International Group, 1984.

6. Mballo C, and Diday E. The criterion of Kolmogorov-Smirnov for binary decision tree: Application to interval valued variables. *Intelligent Data Analysis* 2006, **10**:325-341.

7. Niblett T, and Bratko I. Proc. Expert systems 86 Cambridge, Cambridge University Press, 1986.

8. Quinlan JR, and Rivest RL. Inferring decision trees using the minimum description length principle. *Inform. Comput*. 1989, **80**:227-248.

9. Quinlan JR. Induction of decision trees, *Mach. Learn*. 1986, **1**:81-106.

10. Kass GV. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 1980, **29**:119-127.

11. Loh W-Y, and Shih Y-S. Split selection methods for classification trees. *Stat. Sinica* 1997, **7**:815-840.

12. Provost F, and Kolluri V. A survey of methods for scaling up inductive algorithms. *Data Min. Knowl. Disc*. 1999, **2**:131-169.

13. Freund Y, and Schapire RE. Proc. Machine Learning thirteenth international conference, 1996, pp325-332.

14. Breiman L. Bagging predictors. *Mach. Learn*. 1996, **24**:123-140.

15. Bauer E, and Kohavi R. An empirical comparison of voting classification algorithms: bagging boosting and variants. *Mach. Learn*. 1999, **35**:1-38.

16. Breiman L. Random Forests. *Mach. Learn*. 2001, **45**:5-32.

17. Rosen BE, Ensemble learning using decorrelated neural networks. *Connect Sci*. 1996, **8**:373-384.

18. Melville P, and Mooney RJ. Constructing diverse classifier ensembles using artificial training examples. *Proc. IJCAI* 2003, 505-512.

19. Wang J-Y, Lee HM and Ahmad S. SVM-Cabins: prediction of solvent accessibility using accumulation cutoff set and support vector machine. *PROTEINS: Struct. Funct. and Bioinf.* 2007, **68**:82-91.

20. Kohavi R. Scaling up the accuracy of naïve Bayes classifiers: a decision tree hybrid. Proc. Second international conference on knowledge discovery and datamining, 1996, 114-119.

21. Cunningham P, and Carney J. Diversity versus quality in classification ensembles based on feature selection. Machine learning: ECML 2000 Springer Berlin/Heidelberg, 2000, pp109-116.

22. Zhou Z, and Jiang Y. NeC4.5: Neural Ensemble based C4.5. *IEEE Trans. Knowl. Data En*. 2004, **16**:770-773.

23. Craven MW, and Shavlik JW. Extracting tree-structured representations of trained networks. In Advances in Neural Information Processing Systems, volume 8. MIT Press, Cambridge, MA, 1996.

24. Opitz D, and Shavlik JW. Generating accurate and diverse members of a

neural network ensemble. In Touretsky DS, Mozer MC, Hasselmo ME, eds. Advances in Neural Information Processing Systems, volume 8 531-541 MIT press, Cambridge, MA, 1996.

25. Wolpert DH. Stacked generalization, *Neural Networks* 1992 **5**:241-259.

26. Chan PK, and Stolfo SJ. Toward parallel and distributed learning by meta-learning. In AAAI workshop in knowledge discovery in databases, 1993 pp227-240.

27. Chan PK, and Stolfo SJ. On the accuracy of meta-learning for scalable data mining. *J. Intell. Inf. Syst.* 1997 **8**:5-28.

28. Seewald AK, and Furnkranz J. An evaluation of grading classifiers, in Advances in Intelligent Design, Springer Berlin / Heidelberg, 2001.

29. Cortes C, Vapnik V. Support-vector networks. *Mach. Learn.* 1995, **20**:273-297.

30. Shawe-Taylor J, and Cristianini N. Kernel methods for pattern analysis. Cambridge University Press, 2004.

31. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochem. Biophys. Acta* 1975, **405:**442-451.

32. Martens D, Huysmans J, Setiono R, Vanthienem J, and Baesens B. Rule extraction from support vector machines: An overview of issues and applications in credit scoring. *Studies in Computational Intelligence* 2008, **80**:33-63.

33. Jacobsson H. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 2005, **17**:1223-1263.

34. Guo Y. and Selman B. ExOpaque: A framework to explain opaque machine learning models using inductive logic programming. In 19th international conference on tools with artificial intelligence, 2007, pp. 226-229.

35. Assche AV, and Blockeel H. Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In Proc. of the 17th international conference on inductive logic programming, 2007.

36. Sato M, and Tsukimoto H. Rule extraction from neural networks via decision tree induction. *Neural Networks* 2001, **3**:1870-1875.

37. Krishnan R, Sivakumar G. and Bhattacharya P, A search technique for rule extraction from trained neural networks. *Pattern Recogn. Lett.* 1999, **20**:273-280.

38. Setiono R. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation* 1997, **9**:205-225

39. Setiono R, Baesens B and Mues C. Recursive neural network rule extraction for data with mixed attributes, *IEEE Trans. Neur. Networ.*, 2008, **19**:299-307.

40. Nunez H, Angulo C, and Catala A. Rule extraction from support vector machines. Proc. European symposium on artificial neural networks, D-side Publications, Bruges, 2002, pp107-112.

41. Chaves A da CF, Vellasco MMBR, and Tanscheit R. Fuzzy rule extraction from support vector machines. In Fifth international conference on Hybrid intelligent systems, 2005.

42. Browne A, Hudson BD, Whitley D, Ford MG, and Picton P. Biological data mining with neural networks: implementation and application of a flexible decision tree extraction algorithm to genomic problem domains.

Neurocomputing 2004, **57**:275-293.

Chapter 3: Dihedral angle prediction

3.1 Introduction

Protein secondary structure prediction[1] is an important problem in bioinformatics. Determination of the structure of proteins is a difficult and sometimes impossible task to achieve experimentally. Much work has been carried out towards the computational prediction of the tertiary structure of a protein, but it is impossible by brute force alone. The number of possible conformations an amino acid sequence can adopt is huge. Randomly trying conformations for a given protein, it would take up to 10^{32} years to find the correct one.[2] This is due to the vast conformational space available to proteins and is known as the Levinthal paradox. Currently, accurate 3D structure prediction is beyond the state of the art. As a stepping stone towards solving this problem, it is possible to predict the intermediate level of secondary structure and other properties. Secondary structure is linked to the Ψ backbone dihedral angles, as shown by Ramachandran[3]. In fact, the dihedral angles can be said to define, at least partially, the structure[4]. Based on this relationship, the program Destruct was previously developed[5], with the intention of using an iterative process to allow the prediction of dihedral angles to enhance the prediction of secondary structure and conversely to allow the prediction of secondary structure to enhance the accuracy of dihedral angle prediction.

Φ dihedral angle restraints are often used in the determination of 3D structures by experimental methods, such as NMR. These restraints can also play an important role in molecular dynamics simulations, carried out to analyse the structure and dynamics of a protein. Thus, it is useful to predict the Φ backbone angles where the 3D-structure is unknown. Such a prediction may allow the investigation of possible structures for

the protein.

Here, we aim to improve the prediction accuracy of both Φ and Ψ dihedral angles, both for use in Destruct or a similar method and for use in the prediction of 3D protein structures. Here we predict Φ and Ψ as independent quantities due to practical concerns. Support vector regression (SVR) is only capable of predicting one angle at a time, and even if that were not the case, predicting one value may be easier than predicting two interdependent values when neither of them are known. Several previous predictions have been made of both real value and categorical dihedral angle predictions. The best of these, Real Spine 2, uses twin neural networks, combined with a normalisation of the dihedral angles[6]. This normalisation circumvents some of the deficiencies of the sigmoidal function used in the neural network and significantly improves accuracy over the previous work by the same authors[7]. See chapter 1 for a review of dihedral prediction algorithms. We aim to improve upon this previous work by using SVR[8] to predict the dihedral angles. This machine learning algorithm is as yet untried for prediction of real value dihedral angles. All previous methodologies use neural networks, and SVR is known to be successful at solving a range of machine learning problems[8].

This leads us to hypothesize that the use of SVR[8] combined with accurate secondary structure predictions obtained using cascade correlation networks (CASCOR)[9] will lead to an improvement in accuracy of prediction of both Ψ and Φ dihedral angles. We also implement a similar normalisation scheme (see 3.2.8) to that used by Real Spine 2, in order to test whether this scheme will give rise to a similar improvement with a different machine learning algorithm. The effect of parameter optimisation of

the SVR algorithm is investigated, along with the choice of SVR kernel. We will begin the methods section of this chapter with a description of the data used for training and testing, followed by detailing the procedures for both secondary structure prediction and our experiments with the prediction of dihedral angles. The second half of the chapter will be concerned with the results of these experiments and the conclusions that can be drawn from them.

3.2 Methods

Our experiments began with a revival of the cascade correlation network[9] for the purposes of protein structure prediction. From the very beginning of the project, it was intended that CASCOR (an implementation of a cascade correlation neural network in C) play a role in the experiments. This is building on work previously conducted in the research group[5]. Therefore, our initial experiments approximately reproduce their work. We followed this with experiments for the prediction of dihedral angles. We begin by describing the datasets and the way we represent them for our experiments, as well as giving the rationale behind these choices.

3.2.1 Datasets

The CB513 dataset was compiled by Cuff and Barton[10] as a non-redundant set for protein structure prediction. Here, we use it as our main training set. Despite being a relatively old dataset, it was chosen for this work due to its tried and tested nature. The fact that it has been used by multiple algorithms allows for direct comparison when comparing our results with other methods particularly Destruct, and it is large enough to allow the possibility of accurate prediction without being so large as to hinder computational effectiveness. It consists of 513 proteins, which are non-redundant to

25% identity. We use all sequences of this dataset in our experiments. The CASP4 and CASP5 datasets[11,12] were originally produced for the critical assessment of techniques for protein structure prediction (CASP). They consist of proteins for which the structures had been recently determined and not yet published. This was intended to give a blind test of protein structure prediction methods in a variety of categories. We use all sequences in each of these datasets for which structures can be found in the PDB. CASP4 contains 34 protein sequences and CASP5 contains 61 proteins.

3.2.2 Data pre-processing and representation

Sequence data for all experiments was initially obtained in FASTA[13] format. Each sequence was first converted into a position specific scoring matrix (PSSM) using PSI-BLAST[14] against the nr (non-redundant) database[15]. PSSMs offer a representation of the data, which takes into account evolutionary information. The PSSM for a given protein of length L is an L x 20 matrix. Each of the 20 elements representing each amino acid is a log likelihood, accounting for evolutionary mutation, between that amino acid and each of the other amino acid types. In PSI-BLAST this fulfils the role of the substitution matrix, giving a more sensitive measure of the probability of an amino acid being at a given position. Within PSI-BLAST such matrices are used to perform multiple alignments in a very similar way to a normal sequence alignment. Gap scores, however, are taken from the initial BLAST search. See reference[12] for more details. Each residue is, thus, represented by a 20 x 1 vector taken from the PSSM for that protein.

For these experiments a sliding window approach was used. Each residue in turn is the

centre of a window consisting of the residue under examination and a number of residues either side of it. This window is moved along the amino acid sequence giving a separate sequence window for each residue. After generating the PSSM for each sequence, the data was converted into a sliding window format. We chose a window length of 15, the target amino acid and seven residues to either side. We decided upon this length of amino acid sequence based on previous work[5] and on the computational resources available. This means that each amino acid was represented by a sequence of 15 vectors of length 20 before the addition of labels to the training data.

To obtain the known dihedral angles for the training data, the PDB file for each protein was downloaded from the PDB[15] and the dihedral angles were assigned using DSSP[16]. DSSP was also used to assign secondary structure to the same proteins where required (see later in this chapter). The input values and labels were scaled to between 0.05 and 0.95 for both CASCOR[9] and for SVR. Since there is no fixed minimum and maximum for the PSSM, this was carried out based on the minimum and maximum values of the PSSM throughout our training and testing dataset, making the assumption that values outside this range are exceedingly rare. The scaling is shown in equation 3.1, which appears later. This scaling range was chosen due to restraints on input values for CASCOR. Due to the implementation of CASCOR, values outside this range will cause the software to function improperly. The change in network weights depends on the value of the inputs, e.g. if the input is zero the weights will never change. The scaling was carried out identically both for dihedral prediction and for the secondary structure prediction with CASCOR in order to be consistent.

$$y = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \right) \times 0.9 + 0.05 \qquad (3.1)$$

where y is the scaled value, x is the original value, x_{min} is the minimum value of x in the training data, and x_{max} is the maximum value of x in the training data.

We also experimented with the inclusion of additional amino acid properties in the information supplied to the SVR algorithm. These are included for the central three residues in the sequence window. These are likely to be the most influential residues in determining the dihedral angle. We limit the parameters to the central residues in order to avoid having an exorbitantly long input vector, which would greatly slow training of the SVR algorithm. These parameters are represented numerically and are scaled to the same range described above. The Graph shape index, hydrophobicity, volume, polarizability, iso-electric point, helix probability, and sheet probability were all taken from Meiler *et al.*[17] The authors used neural networks to reduce the dimensionality of the parameters. Here we use the initial values with no reduction in dimensionality, since SVR should be successful at predicting from such high dimensional data.

3.2.3 Secondary structure prediction with CASCOR

Our initial experiments attempt to reproduce the work of Wood and Hirst[5] using cascade correlation neural networks. Cascade correlation neural networks are a specific type of neural network that are designed to minimise training time, improving over back propagation networks[9]. The network is trained in stages, using the quick prop algorithm[18] as its learning algorithm. Starting with an initial network consisting of the input nodes and output nodes as specified by the user, all of which are interconnected, the network adds hidden units one by one, until either an error threshold is achieved or the maximum number of hidden units is reached. Each hidden

unit is added from a pool of candidates (figure 3.1), which are trained based upon connections with all nodes that are already part of the network using the examples from the training data. Training is targeted at maximising the correlation between the value of the candidate and the error of the output of the existing network. This is expressed as the sum over all nodes of the correlation between the value of the candidate unit and the residual output error of a given unit:

$$S = \sum_{o} \left| \sum_{p} \left(V_p - \overline{V} \right) \left(E_{p,o} - \overline{E}_o \right) \right| \tag{3.2}$$

where E_o is the residual output error at unit o, p is the training pattern, V is the value of the candidate unit and $\overline{V}$ and $\overline{E}_o$ are the mean values of V and E_o respectively. S is maximised using a gradient ascent method and the quick prop algorithm. Once the value of S has converged, the unit with the best correlation is added to the network and the process for choosing the next candidate unit begins. This training cycle continues until either the maximum number of nodes has been added to the network, or the error threshold required has been passed. Here we use CASCOR for secondary structure prediction reproducing the methodology of Wood and Hirst[19]. The network is trained on the CB513 dataset and tested using the CASP5 dataset, both represented and scaled as described above. This gives an input vector for training consisting of 300 (20×15) PSSM inputs and 3 example outputs describing the structure type for training. Each of these are either 0.5 (present) or -0.5 (not present) and are representative of helix, sheet or coil. An example input vector is shown in figure 3.2 below. The network produces three outputs, which are between 0.5 and -0.5, representing the three states of secondary structure helix, sheet, and coil. These outputs are interpreted such that 0.5 is positive and -0.5 is negative. The predicted secondary structure type is then the one that is closest to 0.5.

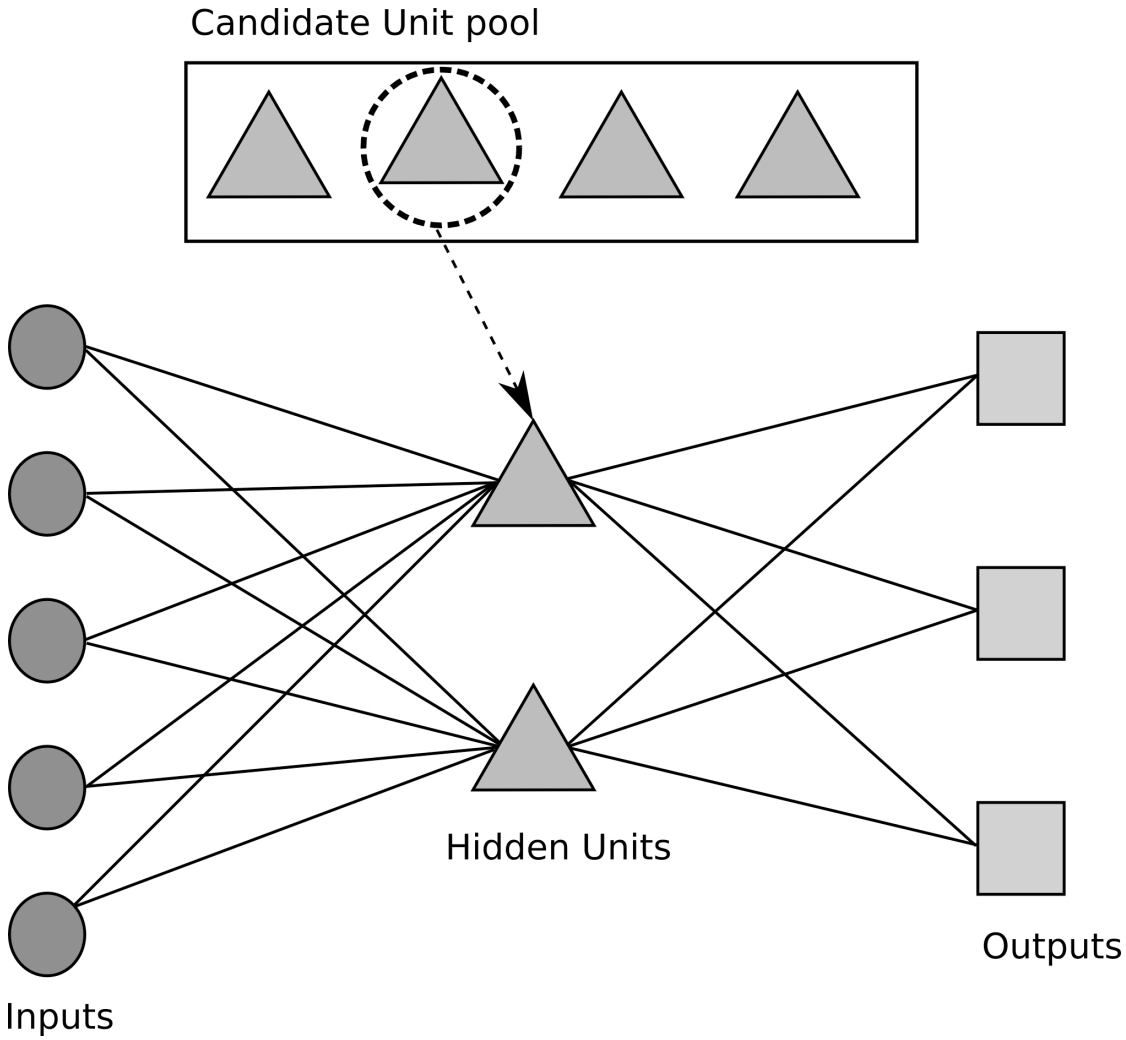


Figure 3.1 Schematic of the cascade correlation network: Inputs are represented by circles and outputs by squares. The hidden units are chosen from a pool of candidates as described in the text. These are integrated into the network as shown here.

0.391, 0.329, 0.329, 0.298, 0.329, 0.36, 0.174, 0.36, 0.391, 0.236, 0.298, 0.174, 0.143, 0.329, 0.174, 0.143, 0.205, 0.360, 0.329, 0.143, 0.205, 0.174, 0.329, 0.329, 0.360, 0.329, 0.174, 0.391, 0.329, 0.205, 0.267, 0.174, 0.236, 0.329, 0.236, 0.143, 0.267, 0.298, 0.329, 0.143, 0.174, 0.174, 0.329, 0.391, 0.205, 0.174, 0.360, 0.236, 0.205, 0.174, 0.267, 0.298, 0.329, 0.298, 0.360, 0.205, 0.174, 0.236, 0.391, 0.174, 0.205, 0.329, 0.174, 0.174, 0.174, 0.143, 0.205, 0.205, 0.205, 0.143, 0.205, 0.329, 0.329, 0.143, 0.329, 0.546, 0.143, 0.174, 0.205, 0.236, 0.329, 0.236, 0.391, 0.298, 0.298, 0.298, 0.205, 0.329, 0.329, 0.236, 0.205, 0.174, 0.267, 0.360, 0.205, 0.174, 0.205, 0.329, 0.298, 0.174, 0.174, 0.205, 0.329, 0.391, 0.298, 0.267, 0.205, 0.391, 0.360, 0.205, 0.298, 0.205, 0.205, 0.360, 0.205, 0.174, 0.236, 0.298, 0.298, 0.174, 0.205, 0.236, 0.329, 0.236, 0.174, 0.174, 0.391, 0.205, 0.205, 0.205, 0.174, 0.298, 0.422, 0.267, 0.329, 0.267, 0.205, 0.236, 0.267, 0.205, 0.205, 0.298, 0.298, 0.391, 0.298, 0.267, 0.174, 0.298, 0.298, 0.205, 0.298, 0.236, 0.205, 0.453, 0.298, 0.205, 0.205, 0.298, 0.236, 0.174, 0.205, 0.236, 0.391, 0.267, 0.298, 0.298, 0.236, 0.329, 0.360, 0.267, 0.267, 0.174, 0.205, 0.360, 0.205, 0.205, 0.298, 0.329, 0.298, 0.174, 0.236, 0.205, 0.267, 0.329, 0.329, 0.298, 0.143, 0.391, 0.422, 0.205, 0.267, 0.174, 0.205, 0.391, 0.205, 0.205, 0.205, 0.267, 0.236, 0.174, 0.205, 0.174, 0.236, 0.298, 0.422, 0.298, 0.143, 0.298, 0.267, 0.422, 0.267, 0.143, 0.143, 0.391, 0.174, 0.205, 0.267, 0.267, 0.236, 0.143, 0.174, 0.143 => +, -, -;

Figure 3.2. An example input vector for CASCOR, the => symbol differentiates the training data from its label, binary attributes are represented as + for 0.5 and – for -0.5.

Output for each input will be displayed as the target amino acid in single letter code

and the predicted structure type as one of H, E, or C (helix, sheet or coil respectively). Structure predictions from CASCOR are then used as an aid to dihedral angle predictions (see below).

3.2.4 Dihedral prediction with SVR

Previous work in predicting Ψ dihedral angles has concentrated on the use of neural networks of varying types[5,6,7]. Here, we attack the problem using SVR. SVR was chosen as a regression method that has yet to be investigated for the prediction of real value dihedral angles, but has shown promising results in other areas. Support vector machines for classification and regression have been applied to a wide range of problems,[20] including biological ones[21]. The theory behind this methodology is explained in chapter 2. However, we give a brief overview here. The basic idea of SVR is to transform the data into a higher dimensional space in such a way that linear regression can be used to fit a function to the data. The data are transformed using a kernel function, which maps the data into a higher dimensional feature space. This takes the form of the kernel matrix. One useful property of kernel-defined feature spaces is that often the regression (or indeed classification) can be carried out using only the inner products (scalar or dot product) between two vectors of the training data without calculating the full mapping to the kernel defined feature space.

Given a training set $X = \{x_{1,\ldots\ldots}, x_l\}$, with labels y_i $\{i = 1,...,l\}$, the kernel function $k\langle x,y\rangle$ maps the training points x to a feature space F:

$$k\langle x,y\rangle = \big(f(x),f(y)\big) \tag{3.3}$$

where $f(x)$ and $f(y)$ are members of the feature space F. SVM classification involves the division of the data by fitting a maximal margin hyperplane to the data within the kernel feature space. If such a hyperplane exists, it can be found by convex optimisation of a quadratic function. However, this only gives a hard margin of classification, leading to misclassification of outliers. The introduction of slack variables allows for a soft margin classification. The margin of the hyperplane is defined by the support vectors obtained from the training data. For regression there is no hyperplane. However, ε-sensitive regression allows the fitting of a function to the training data. Errors below the threshold ε are ignored to give a tube around the function fitted to the data, analogous to the margin of the hyperplane in classification. As with classification, slack variables are used to give a soft margin to the regression function. The support vectors are those that define the margin of the regression function.

3.2.5 Kernel functions

The choice of kernel function is important. Since the philosophy of kernel-based machine learning is to manipulate the data into a feature space that allows the application of standard machine learning algorithms, the selection of an incorrect kernel would mean the data would not be optimally transformed. The kernels considered during this work are those in the PyML machine learning package. This package is written in C and python and utilises the libSVM library[22] for support vector machine algorithms. Details of the specific kernel functions are given below. The more general properties of kernel functions are discussed in chapter 2.

Since the requirement was for a rapid test of kernel functions, we choose the kernel

function using the CASP4 dataset as a representative dataset, which is considerably smaller than the training set, whilst consisting of sequences with a low homology to the training data. This allowed us to quickly eliminate unsuitable kernel functions, with the intention of a more thorough examination if needed. However, the results were clear enough that this was not required. We test each of the kernel functions given below. For the polynomial kernel, we test various degrees (2, 3, 4, 5).

Linear $$K(x,x_i) = x_i^T x \tag{3.4}$$

Polynomial kernel of degree *d* $$K(x,x_i) = \left(1 + x_i^T \frac{x}{c}\right)^d \tag{3.5}$$

Gaussian (also known as RBF) $$K(x,x_i) = \exp\left\{-\left\|x - x_i\right\|_2^2 / \sigma^2\right\} \tag{3.6}$$

where *d*, *c*, and σ are constants.

Input for these experiments was represented in a way similar to that for secondary structure prediction described above. The first part of the input vector was the PSSM representation of the 15 amino acid sequence window. This was followed for training by the dihedral angle scaled to between 0.05 and 0.95. This scaling was chosen to be within both the range required by the CASCOR software (to allow later use with secondary structure prediction) and with the SVR algorithm which requires a range of values between 0 and 1. This input vector was then converted into the sparse data format as required by the SVR implementation (see below). This has the label first followed by each of the data points labelled with its position an example of this input data is given in figure 3.3 below. The output is a real number between 0.05 and 0.95, which can be converted into a dihedral angle by reversing the scaling process.

We evaluated these experiments by ten-fold cross-validation. From these experiments

the Gaussian kernel was obviously the best choice for our experiments. We therefore use the Gaussian kernel in all of the experiments described in the remainder of this chapter.

0.6935 0:0.205 1:0.205 2:0.174 3:0.143 4:0.174 5:0.236 6:0.174 7:0.143 8:0.174 9:0.267 10:0.329 11:0.205 12:0.608 13:0.236 14:0.143 15:0.174 /...../ 289:0.143 290:0.143 291:0.391 292:0.174 293:0.205 294:0.267 295:0.267 296:0.236 297:0.143 298:0.174 299:0.143

Figure 3.3 An example input vector for dihedral angle prediction. Here the label is given first, followed by the training data as *position*:*value*, where *position* refers to the values position in the input vector. We have abbreviated this input vector for clarity. All missing data values have the same format as those shown.

3.2.6 Optimisation

After selecting the kernel, there are three user-defined parameters which require optimisation: C, ε and γ. ε is an error cut-off, that is, we will accept an error as long as it is not larger than ε. This allows the margin of the regression function to be a soft margin fit to the data between the two classes, taking into account variability and outliers within the data. C is the regularisation parameter for the SVR, which influences the weighting of the empirical loss function and error cutoff. γ is the kernel width of the Gaussian kernel. The parameters for the SVR algorithm and kernel can cover a wide range of values and the chosen values can greatly affect the output. In order to obtain a ball-park estimate for the parameters and hence narrow the search space, we used the work of Chersky and Ma[23], who present a mathematical method for finding optimal values for these parameters. The methods in this reference are only accurate on synthetic data and subsequent optimisation is still required when dealing with a real world problem, such as the one in this work.

To calculate an estimate of C, we use the following expression, based on the idea that C should be chosen according to the range of values involved.

$$C = \max\left\{\left|\bar{y} - 3\sigma_y\right|, \left|\bar{y} + 3\sigma_y\right|\right\} \tag{3.7}$$

where $\bar{y}$ and σ are the mean and standard deviation of y the label of the training data. We do not estimate the value of the parameter ε. Estimating ε relies on knowing the level of noise present in the data. For real world data this is generally not known. Chersky and Ma take the approach of estimating it using k nearest neighbours regression. In our case, since the usual range of values is small (0-0.2), we merely proceed with the optimisation for this parameter. For an estimate of the kernel parameter γ, Cherkassy and Ma propose an expression relating the parameter to the dimension d of the input to the regression problem.

$$\gamma = \frac{1}{2\left(\sqrt[d]{m}\right)^2} \tag{3.8}$$

where m is a constant. Using these equations, it is possible to estimate the values for C and γ. This gives a starting point for the grid-based optimisation described below.

We use a grid-based optimisation method for these experiments. Whilst other methods such as gradient descent[24] may have more accuracy, the major advantage to a grid based optimisation was the ease of parallelisation, allowing for a relatively fast optimisation procedure. There is also less risk of a local minimum giving a false result, since the grid is sampled over the complete search space. Optimisation is carried out using a randomly chosen subset of the CB513 dataset with an inner and outer cross-validation approach to prevent bias. Input and output is of an identical

format to the kernel choice experiments described above. A ten-fold cross-validation is carried out for each set of parameters on each fold of data in the dataset. The optimisation is carried out on a 10×10×10 grid of points in parameter space, meaning we have 10 points for each parameter spread over the range of values under consideration. We did not consider more values, as each combination of values requires a separate training and cross-validation operation for each fold of the outer cross-validation, giving 10000 jobs for a 10×10×10 optimisation using ten-fold cross-validation for both inner and outer parts. Therefore, optimising over a grid of higher resolution is impractical. For C, we consider values between one and 1000, with steps of 100, giving plenty of leeway around the result of Chersky and Ma. For ε we consider the full range of values: 0-0.2 with steps of 0.02 and for the kernel parameter γ the range is 0-2 with a step size of 0.2. Two is taken as the upper cutoff, as this is the point at which the kernel begins to behave as a polynomial kernel.

3.2.7 Training and evaluation for dihedral angle prediction

The SVR algorithm was trained on the CB513 dataset represented using PSSM profiles. The input format was identical to that described above for the kernel choice experiments. The prediction method was evaluated by ten-fold cross-validation and also on the CASP4 test set. In order to test whether the optimisation was making an improvement to prediction accuracy, an initial prediction was carried out using the un-optimised SVM. This first set of predictions was carried out using the protein sequence as PSSM alone to represent the data with no additional structural information. The data were presented the sliding window approach described in section 3.2.2. The input vector is in the sparse format as previously described. The second round of predictions was carried out in the same way, using parameter values

obtained through optimisation. Given the level of improvement obtained with the optimised parameter values, no further optimisation was carried out.

Secondary structure was predicted from sequence via CASCOR, an implementation of a cascade correlation neural network described above. The data were supplied to CASCOR in the form of PSSMs, using the same window encoding scheme as for dihedral angle prediction. The output is obtained in binary form, with three output nodes indicating a value for each type of structure (as described above). The final prediction is chosen based on the largest real value of the output. The largest value of the three nodes is chosen as the predicted type of secondary structure, equivalent to each being represented as a 0.5 or -0.5, where 0.5 is the structure predicted for each state and the other two nodes output -0.5.

0.6935 0:0.205 1:0.205 2:0.174 3:0.143 4:0.174 5:0.236 6:0.174 7:0.143 8:0.174 9:0.267 10:0.329 11:0.205 12:0.608 13:0.236 14:0.143 15:0.174 /...../ 289:0.143 290:0.143 291:0.391 292:0.174 293:0.205 294:0.267 295:0.267 296:0.2362 297:0.143103448276 298:0.174 299:0.143 300:1

Figure 3.4. Example input for prediction of dihedral angles with the addition of secondary structure. We have abbreviated the input vector for clarity. The missing values have identical format to those shown. The first part of the input vector is as shown in 3.3. The last attribute represents the predicted structural information as either a 1, 2 or 3 for helix, sheet or coil.

Structure predictions are converted to a numerical representation for the SVR algorithm. A 1, 2 or 3 is used to represent helix, sheet or coil, respectively. This information about the target residue is presented in conjunction with the PSSM profile for the surrounding sequence window. This gives an input vector in sparse format with the label followed by 300 PSSM values and one value representing the structure type at the target residue. An example input vector is shown in figure 3.4 above. The SVR algorithm is trained and evaluated in the same way as described above using the parameters obtained by optimisation previously.

3.2.8 Normalisation

Real Spine 2 uses a normalising procedure, employed to alter the distribution of dihedral angles to better suit a neural network using the sigmoidal activation function. Here we investigated the possibilities for improvement using the same normalising procedure with SVR. Our hypothesis was that this transformed distribution of the dihedral angles will be easier for the SVR algorithm to 'understand' and therefore it will achieve greater accuracy. For Ψ angles, the normalisation is carried out by adding 100° to the angles between -100° and 180° and 460° to the angles between -100 and -180. For Φ angles, angles below 10° we add 350° and angles above 10° we add -10° (equation 3.9). We scale all angles to between 0.05 and 0.95. We compare the results with and without normalisation below.

$$\begin{aligned}
&\Psi \geq -100^\circ \rightarrow \Psi_n = \Psi + 100^\circ \\
&\Psi \leq -100^\circ \rightarrow \Psi_n = \Psi + 460^\circ \\
&\Phi \geq 10^\circ \rightarrow \Phi_n = \Phi + -10^\circ \\
&\Phi \leq 10^\circ \rightarrow \Phi_n = \Phi + 350^\circ
\end{aligned} \qquad (3.9)$$

Equation 3.9 gives normalisation for different values of Ψ and Φ, where Ψ is the original value of Ψ, Ψ_n is the normalised value of Ψ, Φ is the original value of Φ, and Φ_n is the normalised value of Φ.

3.3 Results and Discussion

3.3.1 Initial predictions

The best results were obtained with the Gaussian kernel (Table 3.1). The linear kernel and polynomial kernels of degrees 2 and 3 produced no correlation between predicted and expected values. This is most likely because the feature space into which the data are transformed by the kernels is inappropriate for these experiments.

Table 3.1 Performance of SVR kernel functions evaluated by ten-fold cross-validation. All kernels were used on default settings.

Kernel	**Pearson Correlation Coefficient *r***
Linear	0.055
Polynomial degree 2	0.064
Polynomial degree 3	0.059
Polynomial degree 4	0.55
Polynomial degree 5	0.31
Gaussian	0.62

The polynomial kernels of degree 4 and 5 were much better suited for dihedral angle prediction, although the Gaussian kernel was superior to both and requires less run time. Polynomial kernels require increased run time as the degree of the polynomial increases. All subsequent experiments were carried out with the Gaussian kernel. It is probable that the success of this kernel is a reflection of the underlying distribution of the data.

In order to monitor the effectiveness of optimisation of the support vector machine parameters, an initial prediction for Ψ angles was carried out prior to optimisation of the SVR algorithm. We wished to gain some perspective on the improvement offered by optimisation of the SVR parameters, versus the time taken for the optimisation. The results of the initial Ψ angle prediction (Table 3.2) show an improvement over Destruct, which achieved a PCC (Pearson correlation coefficient) of 0.47, but are still lagging behind Real Spine (PCC = 0.62) and Real Spine 2 (PCC = 0.74). The

parameters chosen as a result of the optimisation procedure are C=1 ε=0.02 and γ= 0.201. This is based on a grid-based search on partial data with an inner and outer cross-validation. The improvement given by optimisation in this case is minimal, with an increase in Pearson correlation coefficient of just +0.02. It is probably coincidental that the default values for the SVR parameters were so close to those optimal for dihedral prediction.

Table 3.2. Results of SVR prediction of Ψ dihedral angles and comparison to previous work.

	SVR	Optimised SVR	Optimised SVR with Structure	Optimised SVR with Normalisation and Structure	Destruct[a]	Real Spine	Real Spine 2.0
Ten–fold cross-validation	0.55	0.57	0.58	0.64	0.47	0.62	0.74
CASP 4		0.56	0.57	0.63			

a. Whilst Destruct tests secondary structure prediction on both the CASP4 and CASP5 datasets, the corresponding accuracies for dihedral angle prediction are not reported.

The version of CASCOR trained for this work was slightly less accurate than that published in Wood and Hirst[19]. This is probably attributable to the lack of a post-processing step and to small differences in experimental setup, such as the initial weights of the neural net or the settings for programs such as PSI-BLAST. The accuracy of structural predictions is given as overall percentage accuracy (Q3) and as the percentage of correct helix, sheet and coil predictions. We obtain an overall accuracy of 67.0% Q3, with accuracy for helices of 78.3%, for sheet as 45.5% and 66.6% for coil. Most of the deficiency in prediction of sheet is from incorrect predictions of the ends of a section of sheet or from short sections of only a few

residues. There is an over-prediction of helix, with some β sheet residues identified as helix. With the addition of predicted secondary structure produced by CASCOR, the accuracy of the prediction improves to a Pearson correlation coefficient of $r = 0.58$, a 0.01 increase over the optimised SVR without predicted secondary structure.

3.3.2 Effect of normalisation

The normalisation procedure improved prediction accuracy to $r = 0.64$, as evaluated by ten-fold cross-validation, although this was still not as accurate as Real Spine 2, nor did it represent as much of an improvement as was achieved by Real Spine 2 over Real Spine. The reason for the limited improvement is probably related to the differences between the two machine learning algorithms. Real Spine 2 uses neural networks with a sigmoidal activation function, and the normalisation step was introduced in order to overcome a specific deficiency of the sigmoidal function in relation to Ψ angle prediction. The neural network performed very badly on angles between -36° and 36°, although all angles in Real Spine 1.0 were scaled to between 0 and 1. The authors used the normalisation to shift the distribution of the dihedral angles away from the problematic area, which was not approximated well by the sigmoidal function. Whilst the SVR technique is very different to the neural network, there is at the same time no real surprise that there is some improvement resulting from an adjustment to the distribution of dihedral angles in this fashion. The normalisation spreads out the angles and also removes any wrapping of regions of accepted angles over the 0-360° boundary.

3.3.3 Effect of addition of amino acid properties

As a further attempt to improve the prediction, we added some parameters associated

with the amino acids in the sequence to the input vector. We added these parameters for the central three residues of the sequence window. An example input vector is shown in figure 3.5 below.

0.773 0:0.205 1:0.205 2:0.174 3:0.143 4:0.174 5:0.236 6:0.174 7:0.143 8:0.174 9:0.267 10:0.329 11:0.205 12:0.608 13:0.236 14:0.143 15:0.174 /...../ 289:0.143 290:0.143 291:0.391 292:0.174 293:0.205 294:0.267 295:0.267 296:0.236 297:0.143 298:0.174 299:0.143 300:2.59 301:0.19 302:4.00 303:1.70 304:6.04 305:0.39 306:0.31 307:2.94 308:0.29 309:5.89 310:1.79 311:5.67 312:0.30 313:0.38 314:1.28 315:0.05 316:1.00 317:0.31 318:6.11 319:0.42

Figure 3.5. An example of the input including amino acid parameters. These parameters are included in the order described in the text between positions 300 and 319 in the input vector. This covers the central three residues of the sliding window. We have abbreviated the input vector shown for clarity. The missing values are of identical format to those shown.

We use the residue's graph shape index, hydrophobicity, volume, polarizability, iso-electric point, helix probability and sheet probability[17]. These are the same parameters as those used by Real Spine. A similar step was taken by Real Spine. However, in contrast to Real Spine, we saw a decrease in the accuracy of prediction upon inclusion of these parameters, giving a Pearson correlation coefficient of $r = 0.44$. The lower accuracy is likely due to noise being introduced into the data by the increase in the number of parameters. It is also possible that the SVM can no longer find a kernel feature space that fits the data as well and so cannot fit a function to the data with the same degree of accuracy. Real Spine used a second neural network trained on these parameters alone to represent the amino acid sequence, and combined this with a network predicting from PSSM. It is possible that a similar approach here would improve the results.

3.3.4 Predicting Φ Angles

Prediction of Φ angles was carried out by a similar method to that of Ψ angles. Once

again the normalisation procedure was employed. No further optimisation of the SVR parameters was carried out. The normalisation of Φ angles is anticipated to have less effect, since there is no particular region apparent which is particularly hard to predict. However, prediction of Φ is expected to be a harder problem overall, as there is less of a range of values which Φ can take.

Evaluated by ten-fold cross-validation, the Φ dihedral angles were predicted with a Pearson correlation coefficient of $r = 0.50$. This is not as good as Real Spine 2, which reached an accuracy of 0.70. It is possible that some small improvement could be gained by optimising the parameters of the SVR. However, this is unlikely to improve enough to surpass Real Spine. It is possible that the kernel function is not as well suited to prediction of Φ as it is to Ψ. It may also be possible to improve the prediction by performing a different normalisation of the data, which better facilitates the raising of the data into a higher dimensional space by the kernel function. The distribution of Φ angles does not have the same problems as that of Ψ angles, namely, that the area of allowed values for Φ does not cross over the central axis or wrap around from -180° to +180°. However, the angles are distributed over a tighter range making this a harder prediction problem for the SVR algorithm.

In general, the improvement given by normalising the distribution of dihedral angles is smaller than that gained by Real Spine 2 over Real Spine. This is most likely because of the differences between the machine learning methods. There is also to be taken into account the different scaling of the dihedral angles, which would reduce any effect presented similar to that observed in Real Spine.

3.3.5 Application of predicted dihedral angles to assigning NMR spectra

A potential application of predicted dihedral angles is in the assignment of NMR spectra. J coupling describes the coupling of two nuclear spins due to the bonding electrons between them. J coupling data obtained experimentally can be used to determine the dihedral angles along the protein backbone. These dihedral angles can be calculated using the Karplus equation[25]:

$$J_3 = A\cos^2(\Phi - 60) - B\cos(\Phi - 60) + C \qquad (3.10)$$

where A, B and C are empirically determined constants, Φ is the backbone angle and J_3 is the 3 bond J coupling determined by experiment. However, the Karplus equation has two possible solutions, and deciding on the correct dihedral angle is not trivial. Currently, dihedral angles are assigned by comparison of chemical shift data to known examples from a relatively small set of proteins (186 in the current version) using the Talos software[26]. This software is only able to assign 72% of dihedral angles and 1.8% of those are incorrect. This is done in preference to using the Karplus equation, as there is currently no easy way to determine which of the two possible solutions of the Karplus equation represents the correct dihedral angle.

3.3.6 Predicting the correct solution of the Karplus equation

The basic goal was to predict which of the solutions of the Karplus equation is the correct dihedral angle for a given amino acid in a given protein. Theoretically, each predicted dihedral angle will be closer to one of the possible solutions. This means that if the predicted dihedral angle is reasonably accurate then the solution closer to

the predicted angle will be correct. However, this approach assumes that the J coupling data from which the prospective dihedral angle is calculated, the constants in the Karplus equation and the predicted dihedral angle are of a reasonable accuracy.

Both Φ and Ψ angles can be calculated using the Karplus equation. However, there is a lack of data available for Ψ dihedral angles and they are less well used by NMR spectroscopists. The J couplings for HN-HA, which can be used to calculate Φ, are readily available for a number of proteins and are the most readily obtainable by experiment. For this reason we chose to use HN-HA couplings to calculate Φ dihedral angles. The choice of constants for the Karplus equation was made from the most recent experimental determination of the three available[27,28,29]. Therefore, A = 7.90, B = -1.05, C = 0.65. We took HN-HA J-coupling data from the BioMagRes databank[30]. These data were originally compiled from a variety of sources and were obtained experimentally with varying error rates and procedures. The data are representative of that typically obtained during NMR spectroscopy work.

We removed sequences with high sequence identity to other sequences within the data set to leave a dataset of 66 proteins, all with sequence identity of 35% or less with the CB513 dataset used to train the SVR model for Φ prediction. The proteins also had low sequence identity among themselves. We took the protein sequences for which we had J-coupling data, and calculated the two possible solutions to the Karplus equation. We obtained Φ dihedral angle predictions for these sequences using the method outlined above. We selected the solution of the Karplus equation that is closest to the predicted angle as the assigned angle for that particular amino acid. If no J coupling data is available for a given residue we supply the user with the predicted dihedral

angle (clearly denoted as such) in place of an assigned value, since this may still be useful to an experimentalist.

To assess the accuracy of the assignment, we compare the dihedral angles we obtain from J coupling data to those given by DSSP for the same proteins. We calculate the Pearson correlation coefficient in the same way as described above. This gives a Pearson correlation coefficient of r = -0.08 with a root mean squared error (RMSE) of 106.9° and a mean absolute error (MAE) of 96.2°. This includes those residues with predicted dihedral angles assigned to them. Excluding these gives a Pearson correlation coefficient of r = 0.04 an RMSE of 97.3° and MAE of 92.4°. These results show little success in assigning dihedral angles.

There are two problems with the concept outlined above, as shown by the results. The first and most pronounced problem is the experimental accuracy of the J coupling data. Whilst a reasonable accuracy of dihedral angle prediction is required, this need only be accurate enough to distinguish between the two values under scrutiny. However, this assumes an accurate set of experimental data. Error bars on the experimental data for HN-HA coupling in BioMagRes are between 1.0 and 4.0 radians per second. This combined with a MAE of 56.3° on the dihedral predictions means, in some cases, that the error is larger than the difference between the two solutions to the Karplus equation. Thus, it is impossible to assign Φ correctly more often than a random choice between the two solutions. This is borne out by the Pearson correlation coefficient.

3.4 Conclusions

Here, we started this work with the hypothesis that SVR would improve the accuracy of dihedral angle predictions. Whilst our method improves over Destruct, our method is not as accurate as the state of the art prediction methods, such as Real Spine 2.0. Dihedral angles can accurately be predicted by SVR, though not as accurately as with neural networks. The normalising procedure used with a sigmoidal function is not as effective. It may be possible to improve upon the predictions using a modified normalisation of the data or by a differing scaling of the PSSM. This may better transform the data for prediction by SVR. There is also scope for developing a specialist kernel to deal with PSSM. Of course, the use of more accurate secondary structure predictions should also improve dihedral angle accuracy.

Another possibility for dihedral angle prediction is to divide the area of allowable dihedral backbone angles into bins and then to classify each residue based on the bin that they fall into. For secondary structure prediction this may be a better approach, since only areas of the dihedral angle space are required to define the secondary structure type. However for other applications, e.g. as restraints for molecular dynamics accurate dihedral angles are necessary and achievable as shown by Real Spine XI[31] a method published after the completion of this work, which improves over Real Spine 2.0, using conditional random fields. It is claimed the angles produced by Real Spine XI are accurate enough to be used to assign local protein structure. We must conclude that whilst SVR is an improvement over cascade correlation networks, it does not advance the accuracy of dihedral angle predictions over the approach used by state of the art methods such as Real Spine.

The method we introduce for NMR assignment could possibly be made into a viable alternative for assigning dihedral angles or providing dihedral angles for molecular dynamics simulations based on experimental data. However, it depends on both more accurate J coupling data for the protein as well as accurate dihedral predictions, perhaps with error bars of less than the error obtained for the Karplus equation.

3.5 References

1. Rost B. Review: Protein Secondary Structure Prediction Continues to Rise. *J. Struct. Biol.* 2001, **134**:204-218.
2. Levinthal C. Are there pathways for protein folding? *Journal de Chimie Physique et de Phyisico-Chimie Biologique*, 1968, **65**:44-45.
3. Ramachandran GN, Ramakrishnan C, and Sasisekharan V. Stereochemistry of polypeptide chain configurations. *J. Mol. Bio.* 1963 **7**:95-9.
4. Lovell SC, Davis IW, Arendall WB, deBakker PIW, Word JM, Prisant MG, Richardson JS, Richardson DC. Structure validation by Cα geometry: φ ψ and Cβ deviation. *PROTEINS: Struct. Fuct. Genet.* 2003 **50**:437-450.
5. Wood MJ, and Hirst JD. Protein secondary structure prediction with dihedral angles. *PROTEINS: Struct. Funct. Bioinf.* **59**:476-481.
6. Xue B, Dor O, Faraggi E, and Zhou Y. Real-value prediction of backbone torsion angles. *PROTEINS Struct. Funct. Bioinf.* 2008, **72**:427-433.
7. Dor O, and Zhou Y. Real-SPINE: An integrated system of neural networks for real-value prediction of protein structural properties. *PROTEINS: Struct. Funct. Bioinf.* 2007 **68**:76-81.
8. Shawe-Taylor J, and Cristianini N. Kernel methods for pattern analysis. Cambridge University Press 2004.

9. Fahlman SE, and Leibriere C. The cascade-correlation learning architecture. Carnegie Mellon University 2004 CMU-CS-90-100.
10. Cuff JA, and Barton GJ. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *PROTEINS: Struct. Funct. Genet.* 1999 **34**:508-519.
11. Moult J, Fidelis K, Zemla A, and Hubbard T. Critical assessment of methods of protein structure prediction (CASP) round IV. *PROTEINS: Struct. Funct. Genet.* 2001 *suppl.* **5**:2-7.
12. Moult J, Fidelis K, Zemla A, and Hubbard T. Critical assessment of methods of protein structure prediction (CASP)-round V. *PROTEINS Struct. Funct. Genet.* 2003 **53**:334-339.
13. Pearson WR, and Lipman DJ. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA.* 1988, **85**:2444-2448.
14. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, and Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*. 1997 **25**:3389-3402.
15. Sayers EW, Barrett T, Benson A, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Feolo M, Geer LY, Helmberg W, Kapustin Y, Landsman D, Lipman DJ, Madden TL, Maglott DR, Miller V, Mizrachi I, Ostell J, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Shumway M, Sirotkin K, Souvorov A, Starchenko G, Tatsusova TA, Wagner L, Yaschenko E, and Ye J. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.* 2009, *Database Issue* **37**:D5-D15.

16. Kabsch W, and Sander C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 1983 **22**:2577-2637.
17. Meiler J, Muller M, Zeidler A, and Schmaschke F. Generation and evaluation of dimension reduced amino acid parameter representation by artificial neural networks. *J Mol. Model.* 2001 **7**:360-369.
18. Fahlman SE. Faster-learning variations on back propagation: an empirical study. Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann.
19. Wood MJ, Hirst JD. Predicting protein secondary structure by cascade correlation neural networks. *Bioinformatics* 2004 **20**:419-420.
20. Moguerza JM, and Muñoz A. Support vector machines with applications. *Statist. Sci.* 2006 **21**:322-326.
21. He J, Hu H-J, Harrison R, Tai PC, and Pan Y. Rule generation for protein secondary structure prediction with support vector machines and decision tree. *IEEE Trans. Nanobio.* 2006 **5**:46-53.
22. Chang CC, and Lin CJ. LIBSVM: a library for support vector machines. [http://www.csie.ntu.edu.tw/~cjlin/libsvm] 2003, Tech. rep., Department of Computer Science, National Taiwan University.
23. Chersky V. and Ma Y. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks* 2004 **7**:113-126.
24. Chappelle O, Vapnik V, Bousquet O, and Mukherjee S. Choosing multiple parameters for support vector machines. *Mach. Learn*. 2002 **46** 131-159.
25. Karplus M. Contact electron spin coupling of nuclear magnetic moments. *J. Chem. Phys.* 1959 **30**:11-15.

26. Cornilescu G, Delagio F, and Bax A. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *J. Biomol. NMR* 1999, **13**:289-302.

27. Wang AC, and Bax A. Determination of the backbone dihedral angles ϕ in human ubiquitin from reparametrized empirical Karplus equations *J. Am. Chem. Soc.* 1996, **118**:2483-2494.

28. Hu J-S, and Bax A. Determination of ϕ and χ_1 angles in proteins from ^{13}C-^{13}C three-bond J couplings measured by three-dimensional heteronuclear NMR. How planar is the peptide bond? *J Am. Chem. Soc.* 1997, **119**:6360-6368.

29. Schmidt JM, Blumel M, Lohr F, and Ruterjans H. Self-consistent ^{3}J coupling analysis for the joint calibration of Karplus coefficients and evaluation of torsion angles. *J. Biomol. NMR*, 1999 **14**:1-12.

30. Ulrich EL, Akutsu H, Doreleijerd JF, Harano Y, Loannidis YE, Lin J, Livny M, Mading S, Maziuk D, Miller Z, Nakatani E, Schulte CF, Tolmie DE, Wenger RK, Yao H, and Markley JL. BioMagResBank *Nucleic Acids Res*. 2007, **36**:D402-D408.

31. Faraggi E, Yang Y, Zhang S, and Zhou Y. Predicting consensus local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. *Structure* 2009 **17**:1515-1527.

Chapter 4: Prediction of Glycosylation Sites using Random Forests

4.1 Background

The second part of this thesis is concerned with the prediction of protein glycosylation sites. We gave a detailed overview of PTM and glycosylation in chapter 1. Here, we give a brief overview of glycosylation of proteins and previous prediction methods. Most proteins do not perform their function without undergoing some form of PTM[1]. PTMs occur after the mRNA has been translated into peptide sequence and the polypeptide has begun to fold[2,3,4]. The importance of PTMs in protein function makes their characterisation of particular interest[2,3,4]. Accurate prediction, using computational methods, of sites in a protein sequence where a PTM occurs would facilitate protein annotation and would contribute to efforts in functional genomics.

Glycosylation[2,3,4], a common PTM, plays a role in protein folding, transport and half-life, as well as being involved in cell-cell interactions and antigenicity. Glycosylation is an enzymatic process, with the exception of glycation, and involves the addition of sugars to the protein to build up glycan chains. There are four types of glycosylation: N-linked, O-linked, C-mannosylation and GPI (glycophosphatidyl-inositol) anchor attachment. C-mannosylation involves the addition of α-mannopyranosyl to the indole of tryptophan. GPI anchors concern membrane anchoring of a protein by the addition of GPI near the C-terminus. N-linked and O-linked glycosylation are the most common types and this study focuses on these modifications.

N-linked glycosylation consists of the addition of a pre-assembled glycan chain to

Asn. This occurs co-translationally and influences protein folding. After its addition, the glycan chain undergoes a maturation process, which can produce a glycan of the high mannose, hybrid or complex types. The sequence motif Asn-Xxx-Ser/Thr[5], or in some rare cases Asn-Xxx-Cys, where Xxx is any amino acid except Pro, is required for N-glycosylation, although not sufficient on its own. O-linked glycosylation consists of the stepwise build-up of various sugars on Ser or Thr residues. O-glycosylation has no known consensus sequence[5]. However, Pro is often present around O-glycosylation sites[6] and O-glycosylation occurs more often in the β-strands of proteins[5].

Several glycosylation predictors have been produced[7,8,9,10]. Whilst these are not directly comparable, due to development on different datasets, the best predictors are NetOglyc 3.1, which is reported to predict correctly 76% of glycosylated residues[11] and 93% of non-glycosylated residues, and Oglyc[10] with a reported accuracy of 85% correctly classified instances. NetOglyc uses both sequence and predicted structural information (predictions of secondary structure and accessible surface area) to train a back propagation neural network. Oglyc uses SVMs trained on a combination of physical properties of amino acids and a binary representation of the sequence. In this chapter, we attempt to improve the prediction of glycosylation sites, using a new machine-learning algorithm well suited to prediction from protein sequence data.

Here we aim to develop a new method for predicting glycosylation sites of both O- and N-linked types. Our hypothesis is that, even when there is no consensus sequence for glycosylation, motifs in the sequence still play a role in determining whether a site is glycosylated. For this reason, we combine pairwise patterns and machine learning,

with the hypothesis that this will produce a more accurate prediction. We select random forest[12] as the machine learning algorithm. This has not previously been used to predict glycosylation sites. However, a decision tree based method is appropriate for sequence (categorical) data and random forest has the added benefit of not over-fitting[12].

The random forest algorithm[12] is based on decision trees. A decision tree consists of paths and nodes, with each node using a rule to decide between two or more paths. A rule is typically of the form 'If A then do B', where A is a condition relating to the descriptors of the input data and B is a step on the path through the trees. The last rule gives the classification of the input data example. Several decision trees are developed using a random selection of inputs and random feature selection at each node to grow the trees. The trees then vote on the class for a given input. There is no previous research into predicting glycosylation using random forests, although the algorithm has been widely used, including for prediction of protein-protein interactions[13,14], for analysis of microarray data[15] and identification[16] and prediction[17] of the function of SNPs (single nucleotide polymorphisms). The algorithm has been used for prediction of protein structure from NMR data[18] and amino acid sequence[19]. The random forest algorithm has several features[15], which make it suitable for applications such as the prediction of glycosylation sites. It can be used on a mixture of discrete and continuous descriptors, to classify binary or multi-class data sets and can cope with datasets where there are more variables than observations. The algorithm does not over-fit and continues to be successful, even when there is a large amount of noise in the data.

However, the models generated by random forest can be challenging to interpret. Therefore, we have employed trepan[20], an algorithm originally designed to allow the comprehension of neural networks. It has been adapted for use with other machine learning algorithms[21]. Trepan uses the machine learning algorithm as an "Oracle". By querying the Oracle with the training data and its own generated examples, trepan induces a decision tree using *m* of *n* rules (see section 4.2.4), thus giving a comprehensible picture of an otherwise opaque machine learning algorithm.

In this chapter, using the database of glycosylation sites OGLYCBASE[22] version 6.00, we analyse the amino acid frequencies around glycosylation sites. Using the O-unique dataset [http://www.cbs.dtu.dk/OGLYCBASE/cbsoglycbase.html] we apply the random forest algorithm implemented in weka[23], combined with information about pairwise patterns, to predict the location of glycosylation sites in a given protein. Pairwise pattern information has previously been used for protein sequence analysis: for example, to predict whether a coiled coil region adopts a leucine zipper structure[24] and to assist in the prediction of protein secondary structure from amino acid sequence[25]. We also experiment with the addition of predicted secondary structure, predicted surface accessibility, and hydrophobicity of the amino acids in an effort to increase the prediction accuracy. Our prediction program is known as GPP (glycosylation prediction program) and is available on-line at: http://comp.chem.nottingham.ac.uk/glyco/. We would like to interpret the models for the random forest algorithm, and thus gain some biological insight into glycosylation. Whilst random forest produces individual rules that are human readable, in the case of GPP for each of the three types of glycosylation there are ten models of ten trees each. There are redundancies and potentially even conflicts between the different models.

We aggregate these models into a single decision tree using the trepan algorithm[20], providing clear rules for each glycosylation type.

4.2 Methods

4.2.1 The dataset

The data for frequency analysis is taken from OGLYCBASE 6.00[22], which is available online from http://www.cbs.dtu.dk/databases/OGLYCBASE/. The OGLYCBASE database contains both experimentally verified and putative instances of N-, O-, and C-linked glycosylation sites. It comprises 242 protein sequences and 2413 verified glycosylation sites. The C-mannosylation data were not considered in our investigations, because there are too few experimentally verified sites in the dataset. Although several enzymes catalyse the attachment of a glycan to Ser and Thr, we have considered all cases in our dataset, with the expectation that the sequence patterns surrounding the glycosylated residue may nevertheless be similar, or at the very least that the machine learning algorithms may be able to detect and learn different sets of patterns within the dataset. For training and evaluation of GPP by ten-fold cross-validation, we use the O-unique dataset. This is a subset of OGLYCBASE and was used for the training of NetOGlyc. It contains only mammalian proteins and is non-redundant. Our predictions were based on only those glycosylation sites that have been experimentally verified. Unverified sites can sometimes be unreliable and false results may confound the predictions. The information retained from the database consisted of the sequence, database reference and the location in the sequence of the modified residues that have been experimentally verified. Both datasets were then split into three, according to whether the modified residues are Ser,

Thr or Asn. Within the O-unique dataset, the Ser dataset contains 1219 instances (395 positive and 824 negative), the Thr dataset contains 1068 instances (370 positive and 698 negatives) and the Asn dataset contains 589 instances (200 positive and 389 negatives). After removing duplicate sequence windows from the OGLYCBASE datasets, the Ser dataset contains 7285 instances (349 positive 6936 negative), the Thr dataset contains 6389 instances (695 positive and 5694 negative) and the Asn dataset contains 3508 instances (261 positives and 3247 negatives). Each instance was considered as the potentially modified residue and seven residues on either side, to give a 15 amino acid sequence window. This choice of window size was based on previous work[10], providing reasonable computational tractability in determining pairwise patterns in the data, and still maintaining sufficient information to predict glycosylation site location. In this work, we use the single letter code to represent the amino acids in a categorical fashion. The weight of each instance derived from the patterns was represented by a numerical attribute. The random forest algorithm can develop trees using a mixture of discrete and continuous data. So no additional processing of the data was necessary before presenting the data to weka to train the random forest algorithm. The prediction program outputs true or false for each glycosylation site. This is then aligned with the sequence to show the status of every potential glycosylation site in context. An example of the input and output is shown in figure 4.1.

4.2.2 Frequency Analysis

As a preliminary test of the validity of using pairwise patterns to predict glycosylation sites, we analysed the frequency of the amino acids surrounding glycosylation sites. We aim to detect any significant increases or decreases in the various amino acids in

the sequence immediately surrounding the glycosylation site, as they may indicate a given amino acid improves or inhibits the chances of glycosylation.

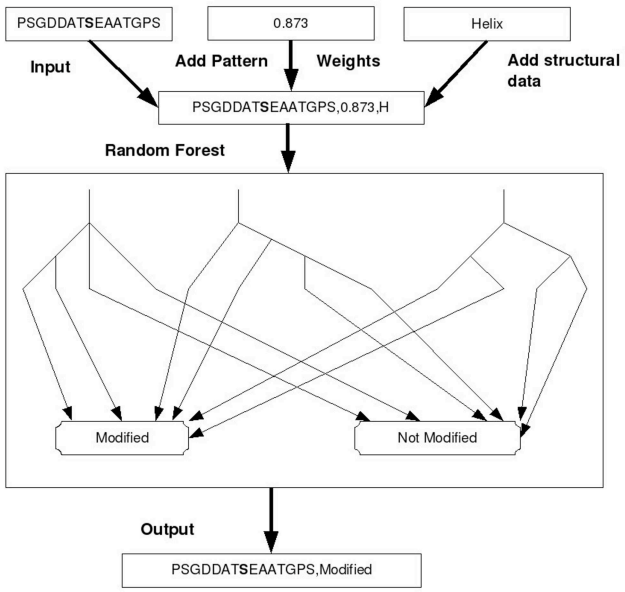


Figure 4.1. The flow of data through the prediction program. The top part of the figure demonstrates the input for random forest, in this case with the inclusion of predicted structural information. The output is shown below a depiction of a small random forest.

After removing all duplicate sequence windows of size 15 from OGLYCBASE, we determined the frequency of each type of amino acid at each position in the window. This was carried out for both modified and unmodified sites for the Ser, Thr, and Asn datasets and on all of these combined. The frequencies of the modified sites were considered to be significant if the difference between the expected frequency and the actual frequency was greater than 3σ, where σ is the standard deviation. The expected frequency of the residue *i* at position *j* was calculated as:

$$E_{ij} = \frac{F_{ij} N_m}{N_u} \tag{4.1}$$

where N_m is the number of sequence windows centred on modified residues, N_u is the

number of windows centred on unmodified residues and F_{ij} is the frequency of occurrence of residues i at position j in the unmodified windows. The standard deviation was estimated assuming a binomial distribution. We focus on frequent patterns in modified sequences, as there is no obvious reason to anticipate that strong negative sequence motifs have evolved to evade recognition by enzymes catalysing glycosylation.

The frequency of each possible unique pairwise arrangement of amino acids in the window was calculated. Patterns below a given frequency threshold were excluded from the final pattern set. To optimise the threshold for pattern exclusion a single data set was prepared for each residue type consisting of all positives and an equal number of negatives; the threshold was increased incrementally and each resulting pattern set was used for prediction. The thresholds that produced the best accuracy were used in the final prediction program. These were 22 for Asn, 31 for Ser and 15 for Thr.

Each pattern is given a weighting, to provide a measure of the probability that a sequence containing that pattern is a member of the modified class. For a pattern *x,* the pattern weight W_x is calculated as F_m/F_n, where F_m is the frequency of modified sequence windows in which pattern x occurs and F_n is the frequency of unmodified windows in which this pattern occurs. Each sequence window is compared against all of the significant patterns for that type of glycosylation site. Based on the patterns found, the sequence is given a pattern weight $W^{seq.}$:

$$W^{seq} = \sum_{x=1}^{k} \frac{W_x}{k} \qquad (4.2)$$

where W_x is the weight of pattern x, and k is the number of patterns found in the sequence. The weight and the sequence window are presented in the form of a string of letters (the single letter code for amino acid representation) comprising the sequence window and a numerical value (the weight), making use of the capability of weka[23] to handle a mixture of continuous and categorical data.

Predicted secondary structure information was combined with the pairwise pattern information described above. The program PsiPred[27] was used to predict the secondary structure of the residue at the centre of each sequence window and this was then placed after the window sequence and the corresponding weight from pattern analysis. PsiPred was selected on the basis of its tried-and-tested nature and its accuracy. The surface accessibility was predicted using the SABLE program[28]. The choice of SABLE was motivated by the method's competitive accuracy as well as its free availability and the availability of the source code. The surface accessibility is predicted as a number between 0 and 100, with 0 representing fully buried and 100 fully exposed. The data obtained from SABLE were added to the central residue of the corresponding instances in the training data. The hydrophobicity value of each central residue was added to the corresponding instance in the training data. These hydrophobicity values were taken from the literature[29]. The data flow through the prediction program is shown in figure 4.1.

4.2.3 Balancing the dataset

The dataset used for training and cross validation has a common problem encountered in machine learning applications[30]. There is a large imbalance between the two classes in the dataset. This leads to a prediction in which the majority class is over-predicted

and the minority class is predicted inaccurately. We needed to find a way of balancing the dataset in order to produce an accurate prediction program. Several approaches to dealing with this problem have been tried, both by artificially balancing the dataset and by way of algorithmic methods. Here we give an overview of several previous methods outlining their suitability and deficiencies, before going on to describe our own algorithm, which we developed to overcome the deficiencies of the algorithms described here. Random under sampling[31] seeks to balance the dataset by removing randomly selected excess examples from the majority class until it is the same size as the minority class. The disadvantage of this is that for each example removed from the dataset some information is lost. This method may not be suitable if the minority class is too small. Too few examples of the majority class will remain and the accuracy of the prediction will be drastically reduced. It is thought that the inaccuracy of the minority class is not solely down to the relative number of examples between the minority and majority classes but also the amount of information available for the minority class, i.e. the number of instances, and the level of noise in the data[30]. Random over sampling[32] adopts the opposite approach to random under sampling. Duplicate data is randomly sampled from the minority class until it reaches the same size as the majority class. This does improve accuracy of the minority class. However, there is a tendency to induce overfitting to the data by duplicating examples, the dataset may grow to a size that is computationally difficult to deal with.

In this work, based on theoretical considerations, we rejected both of the above methods as being unsuitable. Random under sampling drastically reduces the size of the dataset and we lose much of the available information, decreasing the overall accuracy, whereas random over sampling requires vast duplication of the data. Since

we seek to avoid duplicate data by using the O-unique data set, introducing a large number of duplicates of the positive examples is counter productive. There is also an issue with the size of the oversampled dataset and the available computational power.

Alternative methods have been proposed. Tomek links have been used for under sampling data[33]. A given pair of values (E_i, E_j) from different classes, separated by distance $d(E_i, E_j)$ is a tomek link if there is no example E_t such that $d(E_i,E_t) < d(E_i, E_j)$ or $d(E_j, E_t) < d(E_i, E_j)$. One or both of the examples forming a tomek link can be considered noise and thus this can be used to remove examples from the data to perform under sampling. However, this method does not deal with the problem of loss of information. We also reject the method of generating synthetic minority class examples used by SMOTE[32] for random over sampling, due to the potential unreliability of using artificially generated examples.

Several methods for balancing datasets combine multiple techniques using multiple classifiers. A selection of classifiers using a mixture of over and under sampling methods was tried by Estabrooks and Japkowizc[34], taking into account that it is not clear which sampling method is best. This method gives good results, particularly for prediction of the positive examples. However, our data is not amenable to oversampling so this method is not suitable here. Chan and Stolfo[35] use preliminary experiments to find a good class distribution and then create multiple datasets typically with all minority class instances and a selection from the majority class selected based on this distribution. A similar method has also been tried using an ensemble of SVMs[36]. However, these methods assume knowledge of a good class distribution and although this can be estimated, this both adds to the run time and is

not certain to identify such a distribution correctly.

Instead of the above methods, we propose a balancing method that loses the minimum of information, whilst avoiding duplication of data or generation of artificial examples. The method has similarities to both those described by Chan and Stolfo[35] and Yan *et al.*[36] However, we do not attempt to guide the distribution of the data in each of the data sets we create, since this requires good knowledge of the class distribution, which we do not have. We make use of ensemble machine learning methods to train multiple random forests. In order to retain the maximum information from the dataset, each of the datasets we create contains all of the positive examples, and a randomly chosen selection of negative examples equal in number to the positive examples. These examples are chosen without replacement so each of the datasets contains a different selection of negative data. The number of datasets used can then be chosen to correspond to the size of the starting dataset. Here we use ten datasets for each type of glycosylation site. The random forests trained on these datasets vote to determine class. Therefore, the predicting power of the negative examples is preserved, whilst we gain the advantages of an equal dataset and avoid the disadvantages of the under and over sampling methods.

4.2.4 Training the prediction program

Training of GPP has two main components. Firstly, a set of patterns is generated from the training data for each of the three types of glycosylation site. This is used to provide a weighting to each instance in the dataset. Secondly, the random forest is trained on the data and associated weights. Multiple random forests (ten in this work) are trained, with each voting to determine the class of each test instance. Each of the

random forests was trained using a data set comprising all positive instances from the cross validation fold and an equal number of randomly chosen negative instances, this dataset being generated from the training data (see section 4.2.3). We use multiple forests to allow for as complete as possible representation of negative instances in the training data without the negatives completely overwhelming the positives in the dataset. The pattern sets were created from the entire training data within a cross validation fold. This entire procedure is summarised in figure 4.2. The accuracy of the prediction was evaluated by cross-validation. The data were divided randomly into ten sections and the above training procedure was carried out using nine of these, the tenth providing a test set using all instances. This was repeated ten times on each occasion with a different section of the data acting as the test set. The measures of accuracy used to assess GPP are as follows. Sensitivity, expressed as a percentage, is calculated as $T_P/(T_P+F_N)$ x 100, where T_P is the number of true positive predictions and F_N is the number of false negative predictions. Specificity, expressed as a percentage, is calculated as $T_N/(F_P+T_N)$ x 100, where T_N is the number of true negative predictions and F_P is the number of false positive predictions. The number of correctly classified instances is given as a percentage. We use the Matthews correlation coefficient[37] (see equation 2.22) to compare the accuracy of our prediction program with that of the NetNglyc [http://www.cbs.dtu.dk/services/NetNGlyc/] and NetOglyc[11] glycosylation predictors. We use the number of correctly classified instances, the sensitivity and the specificity to compare our work with Oglyc[10]. In order to test the significance of the differences between the different methods of prediction, a paired *t* test[38] was conducted on 30 duplicate experiments for pairs of methods.

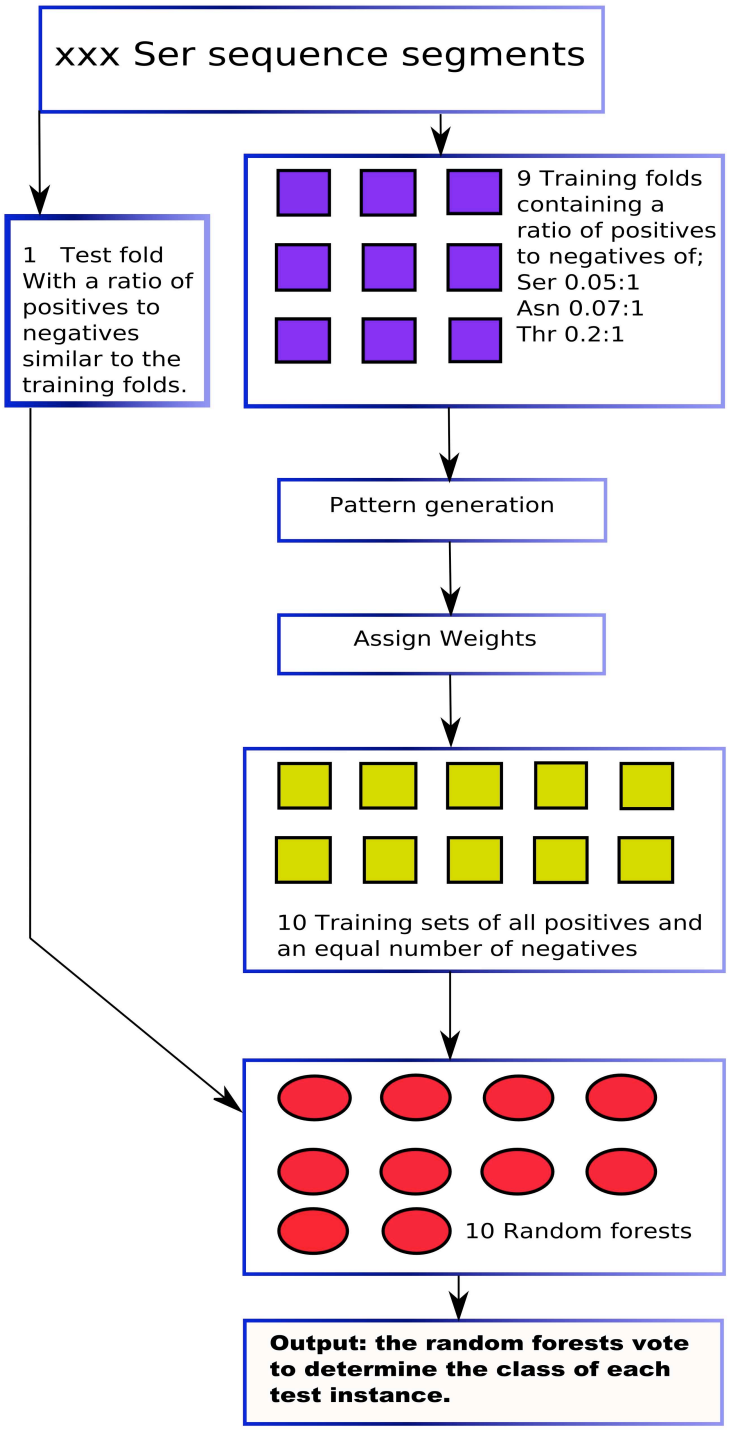


Figure 4.2. The cross-validation of the GPP prediction program, illustrated for the Ser dataset. This procedure is repeated 10 times with each fold in turn being used as the test set in order to conduct a cross validation. The 10 training sets are drawn from the sum of the 9 folds of training data and are used to train 10 random forests.

Given a set of results X_i from method A and a set of results Y_i from method B, each containing n data points, t is calculated as:

$$t = \left(\overline{X} - \overline{Y}\right) \sqrt{\frac{\left(n(n-1)\right)}{\sum_{i=1}^{n} \left(\hat{X}_i - \hat{Y}_i\right)^2}} \tag{4.3}$$

where $\overline{X}$ is the mean of X and $\overline{Y}$ is the mean of Y, $\hat{X}_i = \left(X_i - \overline{X}\right)$; $\hat{Y}_i = \left(Y_i - \overline{Y}\right)$ and p is the probability of obtaining a value as large or larger than the observed t. If p is below 0.05 then the difference of means is significant at the 5% level. The t test was calculated using the R statistics package[39].

For the purposes of comparison, we also conducted the above procedure substituting the naïve Bayes algorithm for random forest. The naïve Bayes algorithm is based on Bayes rule, which states that for a given input vector $x_1.....X_n$ the probability of observing a class M is:

$$P(M|x_1,.....,x_n) = P(x_1,.....,x_n|M)P(M)/P(x_1,.....,x_n) \tag{4.4}$$

Whilst it is theoretically possible to estimate the probability for each class M, in practice the conditional probabilities are not usually known and must be estimated from the data. For this reason, the naïve Bayes algorithm makes the assumption that the conditional probabilities are independent given the class in order to simplify equation 4.4 to:

$$P(x_i|M)=(P(x_1)),...,(P(x_n)) \quad (4.5)$$

Although this is a rough approximation of the probability for a given class, the naïve Bayes classifier has proven to be reasonably robust, because it only matters that the true class receives the highest probability, not that the probability itself is correct. We used the implementation of naïve Bayes in weka[23]. As a further comparison, we also carried out a basic pattern search using scansite[26], which classifies as positive all sites that have the consensus sequence. This was performed on the entirety of O-unique, since no training is required for scansite.

4.2.5 Extraction of Rules

Trepan is a method originally used to extract comprehensible rules from neural networks. Trepan uses an oracle function to represent the network and derives a decision tree from the classifications made by the oracle function. However, it can be used for rule extraction from any method that performs binary classification. We use here a modified version of trepan implemented in Matlab by Browne *et al.*[21], with GPP as the oracle function. Thus, we derive a decision tree based on the classification by GPP of the training data, and additional examples created by trepan. The additional examples are based on the distribution of the attributes in the training data and they ensure a pre-set minimum number of examples reach each node in the tree. The splitting test at each node is an *m* of *n* test. For each node in the tree there are *n* features. If *m* of these features are evident in a given instance, this instance is deemed to satisfy the *m* of *n* rule for this node. In practice, here we find rules only with $m = 1$ and $n < 3$, i.e., simple predicates involving one or two possibilities. Nodes of the tree are expanded based on a priority calculated as the number of examples misclassified

by the node. Those with highest priority are expanded first, since they have the most potential to increase the accuracy of the tree.

4.3 Results and Discussion

4.3.1 Frequency Analysis

We conduct the frequency analysis using the OGLYCBASE dataset. This was used, rather than O-unique, because it has a greater volume and range of sequences, allowing statistically significant differences to the background to be more visible. There is also a wider range of sequences than O-unique and it is useful to observe whether there are trends across the whole spectrum of glyco-proteins, i.e., is our method likely to be useful for predicting more than just the mammalian glycosylation sites found in O-unique? The consensus sequence for Asn glycosylation is clearly exhibited in the frequency table (Table 4.1). The only amino acids in evidence at the +2 position are Ser, Thr and Cys, with low numbers of Pro at the +1 position. At the -6 position there is an increase in Asp and at the -5 position there is a significant increase in Met. Met is hydrophobic in nature, and is the only such amino acid to be increased around glycosylated Asn residues. At the -2 position Gln is significantly increased. Cys is increased at the +3 position, indicating that Cys assists glycosylation at this position. There is an increase in Pro at the +4 position, which is perhaps surprising, as Pro disfavours glycosylation when found at +1 in almost all cases[5]. It may be that Pro helps create a structural conformation favourable for glycosylation when found at this position.

Position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
C	12	9	12	12	10	4	5	11	5	**17**	11	8	16	9
D	13	**23**	10	8	9	13	7	8	*0*	14	6	17	12	19
M	13	2	**10**	6	3	5	3	5	0	6	4	5	2	4
P	13	22	15	20	7	15	11	*1*	*0*	6	**31**	18	18	14
Q	9	10	10	16	15	**21**	7	8	*1*	11	13	16	9	11
S	20	22	25	16	16	14	24	23	**102**	32	23	28	11	32
T	14	26	15	23	17	17	15	17	**151**	16	16	13	19	22

Table 4.1. Frequencies of selected amino acids surrounding modified Asn residues. Frequency is reported as the number of occurrences in the set of 261 instances of modified Asn residues. Statistically significant increases over the expected frequencies are represented in bold; significant decreases are represented by italics. The full table containing all amino acids is included in Appendix B.

Around modified Ser residues there is known to be an abundance of Pro, Ser and Thr and the frequency analysis (Table 4.2) shows increases of Ser and Thr across the sequence window and increases in Pro at positions -6, -3, -1, 2, 3 and 4. Of those positions where Pro is increased, -1 and +3 present the greatest increases. There is an increase in Ala around the glycosylation site at position –1 perhaps suggesting small amino acids are preferred here. There is also a decrease in Phe at this position. Leu is decreased at –6, -2, +2, and +7, and Lys at +3 and +4. This suggests that these amino acids may have an unfavourable effect on glycosylation.

Position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
A	22	36	24	36	29	30	**37**	31	34	21	26	30	25	19
D	*6*	12	6	18	14	10	6	4	*3*	10	11	8	5	*5*
E	15	19	22	19	23	24	8	16	10	13	10	11	9	*6*
G	19	26	30	17	20	22	27	40	27	27	16	23	19	41
P	31	**38**	35	35	**41**	34	**46**	28	**40**	**51**	**42**	34	32	35
S	**56**	42	43	**54**	**53**	**56**	47	48	**60**	43	**56**	41	**48**	**49**
T	**58**	**43**	**46**	**41**	**48**	34	**62**	**61**	**48**	**50**	**58**	**51**	**51**	**47**

Table 4.2. Frequencies of selected amino acids surrounding modified Ser residues. Frequency is reported as the number of occurrences in the set of 388 instances of modified Ser residues. Statistically significant increases over the expected frequencies are represented in bold; significant decreases are represented by italics. The full table containing all amino acids is included in Appendix B.

Modified Thr residues (Table 4.3) exhibit elevations in Thr at all positions, except +7, and Pro at all odd numbered positions. There is an increase in Ser at the –1 position. This suggests that where Thr and Ser glycosylation sites are clustered together, they are almost always consecutive in sequence. Pro is particularly increased at the +3 position, suggesting this is important for glycosylation, as was shown by others[6].

There is a decrease in Ile at position –1 and an increase at –2. Gly is increased downstream at positions -5 and -2, and upstream at positions +1, +4 and +7. Gly is also decreased at –3 and +3. Gln is decreased at the –1 position, as is Lys, which is also decreased at –2, and +1, 2 and 3. There is a general decrease in Leu around the

glycosylation site, particularly at the –1 and +1 positions. Arg is decreased at –3, –1 and +3.

position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
G	52	40	**108**	42	*21*	**122**	32	**101**	40	*16*	**106**	30	37	**112**
I	20	16	16	33	23	**40**	*14*	17	17	13	18	29	14	18
P	**77**	61	**81**	67	**99**	62	**128**	**103**	68	**167**	59	**105**	59	**89**
R	33	22	17	28	*8*	*13*	21	*11*	25	9	15	30	30	16
S	81	67	65	68	74	70	**89**	67	76	74	63	59	52	70
T	**101**	**168**	**96**	**130**	**117**	**115**	**107**	**95**	**118**	**131**	**127**	**95**	**156**	87
W	2	10	4	*1*	2	4	2	2	5	2	3	*0*	*0*	*0*

Table 4.3. Frequencies of selected amino acids surrounding modified Thr residues. Frequency is reported as the number of occurrences in the set of 2010 instances of modified Thr residues. Statistically significant increases over the expected frequencies are represented in bold; significant decreases are represented by italics. The full table containing all amino acids is included in Appendix B.

4.3.2 Pairwise Patterns

The pairwise patterns for each residue type were ranked by weight to identify those most likely to be found around modified residues. These patterns have significant frequencies around unmodified residues, as well as around modified residues. The weights of some patterns are very similar, especially those for Ser, and statistical fluctuations due to the relatively small size of the dataset mean that the rank order of these patterns may not be exact.

Asn Pattern (weight)	Thr Pattern (weight)	Ser Pattern (weight)
.....N.S..... (4.78)	.....P.T.. (3.39)	.....S..P.... (0.98)
.....N.T..... (3.35)	...T.....P.... (2.17)	N.....S..... (0.90)
.....Q.N..... (1.78)	..T..P..... (2.14)	.S.....S..... (0.89)
.....N.....Q (1.27)	S..T..... (1.74)	.....S.....P (0.87)
.....N.....S (1.18)	.....S....P (1.57)	.....S.....I. (0.86)
....R..N..... (1.0)	.....T..I.... (1.43)	.....S....P.. (0.83)
.....AN..... (1.0)	.....T.....I. (1.39)	....P..S..... (0.82)
..I....N..... (1.08)	...T..P..... (1.25)	.....SA..... (0.80)
.....N.....F. (1.08)	.....T.....M (1.25)	.....S.....H. (0.80)
..S....N..... (1.05)	.....T.....P. (1.24)	.....ST..... (0.79)
S.....N..... (0.95)	.....TE..... (1.23)	.....S...V... (0.79)
...R...N..... (0.92)	....M..T..... (1.22)	.....S..T.... (0.77)
....F..N..... (0.92)	Q.....T..... (1.15)	.....S....R.. (0.77)
...P...N..... (0.89)	.....SP.... (1.11)	.....S.I..... (0.76)
.I.....N..... (0.88)	.M.....T..... (1.0)	.....ES..... (0.76)
.....N....A.. (0.88)	....P..T..... (1.0)	.....IS..... (0.74)
.....N.....L. (0.88)	.....AT..... (1.0)	.....S...P... (0.73)
.....R.N..... (0.86)	.....T..A.... (1.0)	.....S.....A. (0.73)
.....NV..... (0.82)	.....PT... (1.0)	...S...S..... (0.73)
.....N..S.... (0.81)	.....PS (1.0)	.P.....S..... (0.72)

Table 4.4. The 20 most significant patterns for glycosylated residues. The patterns are shown with amino acids represented by there single letter code and a '.' used to denote a position that may be occupied by any amino acid.

Around Asn residues (Table 4.4) the consensus sequence for Asn glycosylation was visible, with the patterns .....N.T..... (rank 02, weight 3.35) and .....N.S..... (rank 01, weight 4.78) as the top two patterns identified. Other patterns have substantially lower weights indicating the significance of the consensus sequence. Further patterns in the list indicate that Gln at –2 may be significant, as well as Ser, Ala and Arg at various positions. Gln at –2 is also increased in the frequency analysis above and so may be a

significant factor. However, there is no significant increase of Ser, Ala and Arg at corresponding positions in the frequency analysis, so it is possible this is only evident as part of a pairwise pattern.

The most significant pattern around Ser is Pro at the +3 position, which is in line with the frequency analysis. Other patterns include Pro, Ser, Ile and Thr at various positions indicating that these amino acids may play a prominent role when linked with either Ser or Thr. Many of the patterns around Ser residues have similar weights, although Pro at +3 is markedly more significant.

Whilst no consensus sequence has been shown for Thr, around Thr residues (Table 4.3) there are correlations between the patterns, which suggest one or more sequence motifs may enhance the propensity for glycosylation. The majority of the patterns in the top 20 contain one of Ile, Thr, Pro or Ser, suggesting that these amino acids favour glycosylation. Given the frequency, and the analysis above (Table 4.3) it is likely that at least one or more of these amino acids is required for Thr glycosylation. The most prominent pattern is of Pro and Thr at the +3 and +5 positions, respectively. This could indicate either a motif that encourages glycosylation or the importance of the clustering of Ser and Thr glycosylation sites together given the significance of Pro in the neighbourhood of both. There are also several patterns with high significance involving Glu always upstream of the glycosylation site, although no significant increase of this was found in the frequency analysis. It is evident however, that pairwise patterns in isolation are not sufficient for correctly predicting glycosylation sites. For example, the consensus sequence for Asn glycosylation is represented by

several pairwise patterns, but not all of the consensus sequences are glycosylated, so this alone is not sufficient to guarantee the presence of a glycosylation site. Only the synthesis of all the patterns gives sufficient information to predict whether glycosylation occurs. This is even more the case for O-linked glycosylation sites where a number of factors may be in effect as indicated by the frequency analysis. The complex nature of these overlapping patterns means that machine learning is necessary to utilise the information presented by the patterns in the form of a patterns weight and the sequence itself.

4.3.3 Prediction accuracy

To assess the improvement in accuracy of our balancing method, we carried out a prediction for Ser glycosylation sites using random under sampling and training with a single random forest. This resulted in an accuracy of 69.2% correctly classified instances, with a sensitivity of 67.9%, a specificity of 71.5%, and a Matthews correlation coefficient of 0.38. This is much lower than the results for Ser prediction given below. We do not consider other balancing methods for the reasons stated above in section 4.2.

We measured the prediction accuracy of GPP trained using the pattern weight and sequence only, and using additional structural information. For O-linked glycosylation sites the change in accuracy with additional information was minimal. For N-linked glycosylation an increase in accuracy was observed with the addition of predicted surface accessibility information. There was also a much smaller increase with the addition of predicted secondary structure information (Table 4.5). The prediction of Thr sites was more accurate than that of Ser sites. The Matthews correlation

coefficient, specificity and overall accuracy were higher.

	Random Forest				Naïve Bayes			
Dataset (size)	Correctly Classified Instances (%)	Sensitivity (%)	Specificity (%)	Matthews Correlation Coefficient	Correctly Classified Instances (%)	Sensitivity (%)	Specificity (%)	Matthews Correlation Coefficient
Ser	90.8	96.1	88.9	0.81	83.9	64.4	92.6	0.61
Ser + SA	91.1	95.5	89.6	0.82	82.3	60.5	92.3	0.58
Ser + Hydro	89.9	96.4	87.5	0.79	82.7	64.8	90.9	0.59
Ser + SS	91.7	96.3	90.1	0.83	82.4	62.9	91.3	0.58
Thr	92.0	93.6	92.4	0.84	86.8	74.8	93.3	0.70
Thr + SA	91.8	91.4	93.2	0.83	85.8	72.5	93.5	0.69
Thr + Hydro	91.1	91.8	92.2	0.82	85.9	73.0	93.3	0.69
Thr + SS	91.0	91.8	92.1	0.82	87.2	74.7	94.6	0.72
Asn	92.8	96.6	91.8	0.85	90.3	83.8	94.6	0.79
Asn + SA	94.0	95.7	94.3	0.88	89.3	81.9	94.5	0.77
Asn + Hydro	92.4	95.2	91.9	0.84	90.1	82.5	94.8	0.78
Asn + SS	93.2	96.4	92.4	0.86	89.3	79.8	94.9	0.76

Table 4.5. Accuracy of prediction of glycosylation sites with random forest and naïve Bayes algorithms. Hydro = Hydrophobicity data; SA = predicted surface accessibility; SS = predicted secondary structure.

However, the sensitivity was higher for the Ser site predictions. This was also the case for predictions of Ser and Thr carried out with additional information. In comparison to naïve Bayes, the prediction by random forest is superior. All predictions by naïve Bayes have a substantial loss in sensitivity and a much lower Matthews correlation

coefficient.

	GPP	NetOglyc	NetNglyc	Oglyc	CKSAAP[40]	EnsembleGly[41]	Scan Site
Ser CCI	90.8	91.8	N/A	N/R	83.1	N/R	N/A
Ser Sensitivity	96.1	66.7	N/A	N/R	80.7	N/R	N/A
Ser Specificity	88.9	95.3	N/A	N/R	85.6	N/R	N/A
Ser MCC	0.81	0.62	N/A	N/R	0.671	N/R	N/A
Thr CCI	92.0	84.9	N/A	N/R	81.4	N/R	N/A
Thr Sensitivity	93.6	81.5	N/A	N/R	80.3	N/R	N/A
Thr Specificity	92.4	89.5	N/A	N/R	82.5	N/R	N/A
Thr MCC	0.84	0.67	N/A	N/R	0.63	N/R	N/A
Asn CCI	92.8	N/A	76.7	N/A	N/A	95.0	79.8
Asn Sensitivity	96.6	N/A	43.9	N/A	N/A	98.0	72.7
Asn Specificity	91.8	N/A	95.7	N/A	N/A	77.0[b]	81.9
Asn MCC	0.85	N/A	0.49	N/A	N/A	0.84	0.54
Overall CCI	91.4[a]	88.6[a]	N/A	87.0[a]	N/A	89.0	N/A
Overall Sensitivity	94.9[a]	76.0[a]	N/A	92.0[a]	N/A	59.0	N/A
Overall Specificity	90.7[a]	92.8[a]	N/A	78.0[a]	N/A	68.0[b]	N/A
Overall MCC	0.83[a]	0.66[a]	N/A	0.71[a]	N/A	0.64	N/A

Table 4.6. A comparison of GPP and other glycosylation prediction programs.

a. combined accuracy for Ser and Thr

b. Specificity for EnsembleGly was calculated as T_p/ T_p+F_p. See text for comparison.

N/A=not applicable; N/R = not reported; CCI = % correctly classified instances; MCC = Matthews Correlation Coefficient

We first compare the results to the NetOglyc[8], Oglyc[10] and NetNglyc [http://www.cbs.dtu.dk/services/NetNGlyc/] prediction servers (Table 4.6). The

comparison with O-glycosylation predictors comes with the caveat that they may have been trained and tested with different data, which included differing ratios of positive and negative instances. We also had a slightly different focus than these predictors, in that we do not restrict ourselves to mucin glycosylation sites. For NetOglyc, we use data published in Julenius *et al.*[11] The accuracy measures reported did not include correctly classified instances; so we calculated this from the information published. No published results are available for NetNglyc; so we submitted the sequences in the O-unique dataset to the NetNglyc web server and calculated the accuracy measures described above. We also compare predictions for the Asn dataset to a basic pattern search for the consensus sequence carried out by scansite[26]. Li *et al.*[10] did not give the Matthews correlation coefficient for the Oglyc predictions. Therefore, we calculated it from the reported data and also use the measures of correctly classified instances, sensitivity and specificity for this comparison. We converted the values provided by Li *et al.* into percentages. Oglyc only report the combined accuracy; separate accuracy information for Ser and Thr was not available. The comparison with Oglyc was carried out against their dataset two, which produced the best results for their predictor. The GPP predictor has a higher correlation coefficient and sensitivity than NetNglyc. Scansite correctly predicts most positive instances of Asn glycosylation and has a higher sensitivity and specificity than NetNglyc. However, GPP is more accurate and has higher Matthews correlation coefficient, sensitivity and specificity. Our prediction of Thr sites is better in all measures than that of NetOglyc. For Ser prediction our overall accuracy is comparable, although we have a higher Matthews correlation coefficient. NetOglyc has a higher specificity and a lower sensitivity than GPP. There is a higher ratio of negatives to positives in the Ser data set compared to that for Asn and Thr. This affects the pattern weights, bringing them closer together

and making it more difficult for the random forest to discriminate between modified and unmodified residues. There are also more types of sugar in more equal proportions in the Ser dataset, creating a more difficult task for the random forest. The Asn dataset does not experience similar effects: its consensus sequence motif is easily picked out (and augmented) by the random forest algorithm. There are no data for separate Ser and Thr predictions available for Oglyc[10]. Their overall prediction accuracy of 87.4% (correctly classified instances) is less than the overall accuracy of GPP, and we also score better in sensitivity and specificity.

Two more recent predictions servers, EnsembleGly by Carega *et al*.[40] and CKSAAP by Chen *et al*.[41], were published during the completion of this work. Carega *et al.* use ensembles of SVMs to predict O- and N-linked glycosylation sites. Carega *et al*. calculate sensitivity as $S_n = T_p/ T_p+F_p$. We convert this measure into a percentage. Calculating this measure for GPP, for Asn prediction S_n = 87.17 for Ser S_n = 81.3 for Thr S_n = 7.53 and for the combined O-Linked predictions S_n = 84.4 GPP has a greater Matthews correlation coefficient for both N- and O-linked prediction (only an overall score for O-linked is given). For N-linked sites they have a greater accuracy and sensitivity, but GPP has greater specificity and Matthews correlation coefficient, indicating EnsembleGly has a greater number of false negative predictions. For O-linked sites, GPP scores better for sensitivity, specificity and Matthews correlation coefficient. Chen *et al*. predict mucin glycosylation sites using k-spaced pairwise patterns and SVMs. This method has some similarities with our own and the accuracy of the two methods is comparable. However, GPP is more accurate for both Ser and Thr predictions.

4.3.4 Rule extraction

Trepan identifies the consensus motif for Asn glycosylation (figure 4.3) as the most prominent rules in the decision tree. However, subsequent rules are somewhat misleading, as they allow glycosylation without the consensus sequence being present. This is probably an artefact of the generation of additional data by trepan. This approach is reliant on the distribution of the training data and will highlight patterns additional to the consensus sequence. The tree corresponding to Thr glycosylation (figure 4.4) shows features in line with the statistical data. Pro at residue +3 increases glycosylation when accompanied by a Ser or Thr.

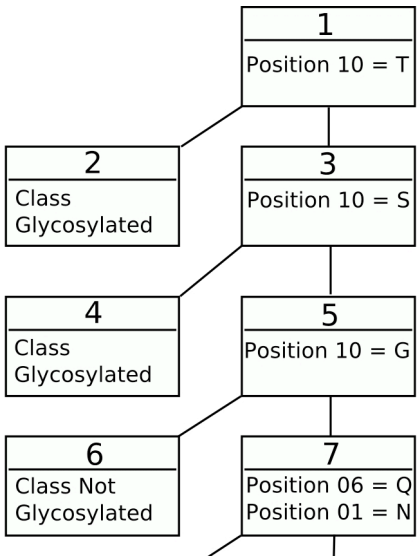


Figure 4.3. Asn glycosylation rules. A subset of the complete decision tree covering all the rules for Asn glycosylation (the complete tree is available in Appendix C). Each node is numbered in the order it was added to the tree. All rules are 1 of A, B… N so only the relevant features are shown. The amino acids are represented using the single letter code and the positions are indicated with respect to a sequence window of length 15, with the target residue at position 08.

The end of the sequence seems to be given undue importance. However, other rules are in line with the frequency analysis. Cys seems to strongly discourage

glycosylation, whilst Ser, Thr and Pro encourage it when accompanied by various other amino acids.

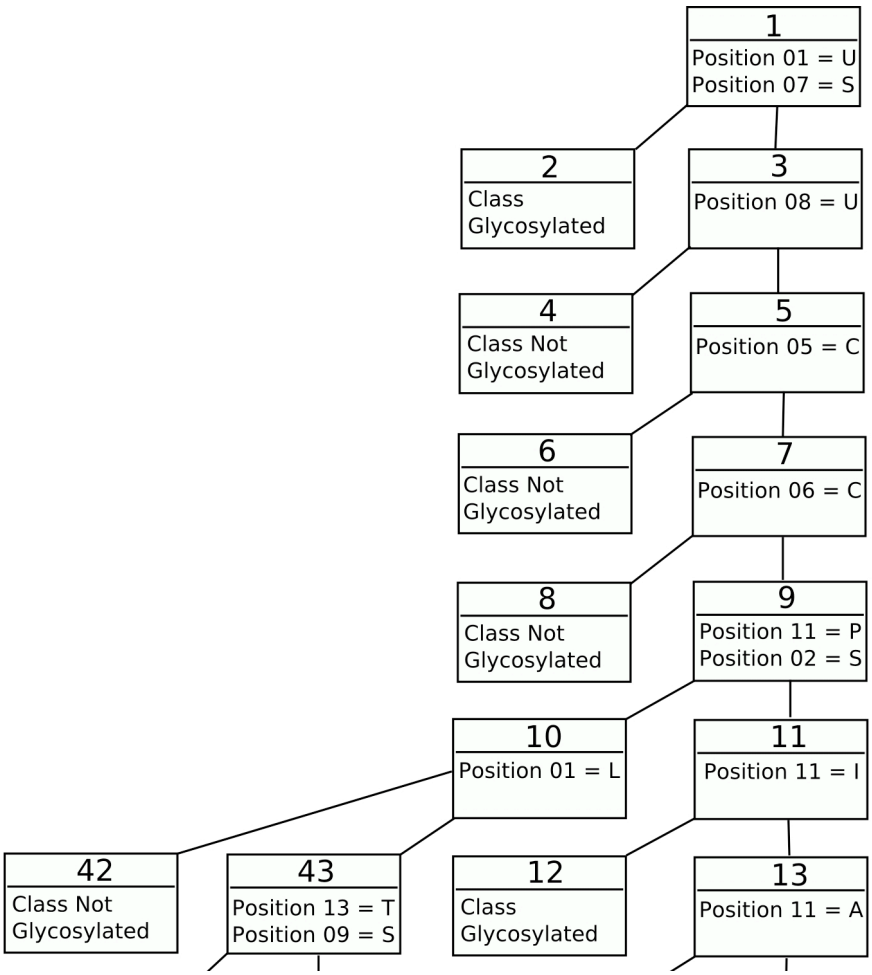


Figure 4.4. Thr glycosylation rules. A subset of the complete decision tree encompassing all the rules produced for Thr glycosylation (the complete tree is available in Appendix C). Each node is numbered in the order it was added to the tree. All rules are 1 of A, B… N so only the relevant features are shown. The amino acids are represented using the single letter code and the positions are indicated with respect to a sequence window of length 15, with the target residue at position 08.

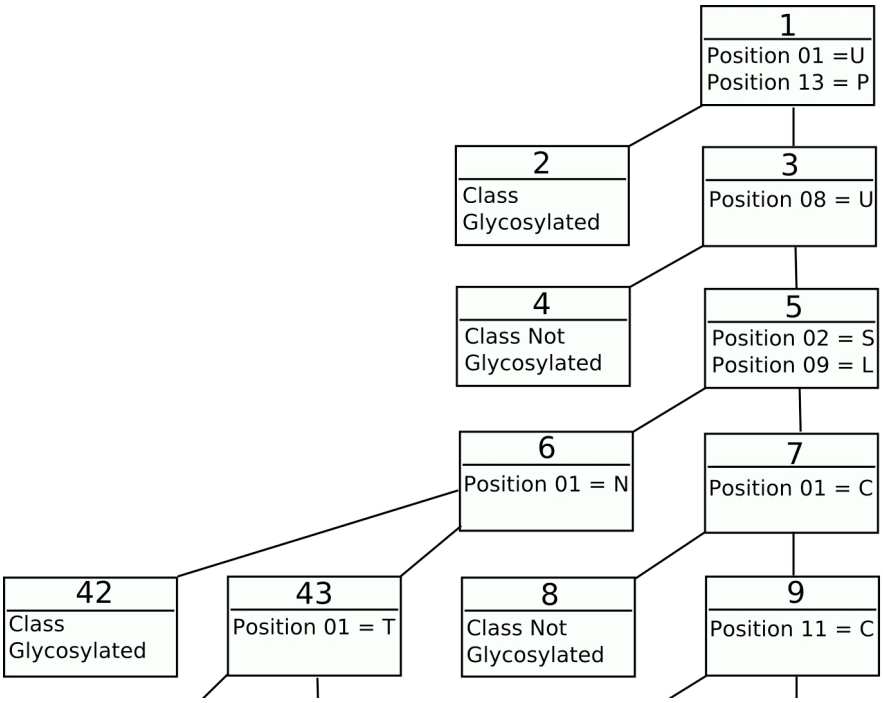


Figure 4.5. Ser glycosylation rules. A subset of the rules produced for Ser (the complete tree is available Appendix C showing the importance of the +2 position in glycosylation of Ser. Each node is numbered in the order it was added to the tree. All rules are 1 of A, B… N so only the relevant features are shown. The amino acids are represented using the single letter code and the positions are indicated with respect to a sequence window of length 15, with the target residue at position 08.

Some rules may be inexact, due to the limited data in O-unique that trepan can base its derived examples on. This is also true for the Ser tree (figure 4.5). The tree for Ser is similar to the one for Thr, although more complicated. Once again, the end of the sequence is implicated, as is the presence of Pro at various positions. Cys again seems to block glycosylation, whilst Ser, Thr, Glu, and Pro all encourage it when present at various positions along the sequence, especially in conjunction.

4.3.5 Sugar type

The approach to predicting glycosylation sites described above predicts all glycosylation sites, and does not differentiate with respect to the type of glycan attached, or the glycosidic linkage involved. As previously described, there are many

different sugars involved in glycosylation, and in most cases each of these has a different enzyme catalysing its attachment. Therefore, it is reasonable to assume that there are differences in the amino acid sequence associated with each type of glycan. To test this, we classify the data in OGLCYBASE by the sugar type that attaches the glycan to the protein (linkage sugar). We use OGLYCBASE, as it includes more sequences and presents a more realistic representation of the distribution of linkage sugars. Duplicate sequences were removed from the database. However, other ways of reducing sequence identity were not carried out, as it is expected that glycosylation sites for different types of sugars will be similar. Sequence representation was as described above for prediction and rule extraction. Since the number of classes is large, classification methods such as random forest or SVM are unlikely to distinguish between the various classes accurately. So we use clustering to distinguish between the different glycans.

For this work, we use farthest first clustering[42], which aims at separating the data optimally based on distance. The first point of k cluster centres is chosen at random (k is user defined). The second cluster centre is then the data point furthest away from the first point. The third cluster centre is the data point furthest from these two points. This process is continued until k points have been chosen to act as k initial cluster centres, with the k^{th} point being furthest from the k-1^{th} point. Each data point is assigned to the closest of these cluster centres using a distance metric. According to Hochbaum and Shmoys[42] this algorithm produces clusters, which are no more than twice the optimal k centre value. For farthest first clustering, we used $k = 7$ clusters for Ser residues and $k = 9$ for Thr residues. Experiments showed that these produced the clearest split in the data (data not shown). Although there are sugars with only a few

examples in the data set that do not have their own clusters, increasing the number of clusters did not separate these instances from the bulk of the data.

Cluster 1 (57)	Mannose 61.4%	GalNAc 33.3%	GlcNAc 3.5%	Xylose 1.8%
Cluster 2 (33)	GalNAc 69.7%	Mannose 24.2%	GlcNAc 6.0%	
Cluster 3 (19)	GalNAc 63.2%	Xylose 15.8%	Mannose 10.5%	GlcNAc 10.5%
Cluster 4 (16)	GalNAc 43.8%	Xylose 37.5%	Mannose 6.3%	GlcNAc, Galactose[a] 6.3%
Cluster 5 (64)	GalNAc 92.2%	Mannose 4.7%	Xylose 1.6%	GlcNAc 1.6%
Cluster 6 (29)	Mannose 48.3%	GalNAc 37.9	Xylose 10.3%	Gal-GalNAc 3.4%
Cluster 7 (46)	GalNAc 58.7%	GlcNAc 23.9%	Mannose 17.4%	

Table 4.7 Percentage membership of clusters generated by farthest first clustering of Ser glycosylation sites

a. both sugars have equal percentage so both have been included.

The composition of the clusters is presented in table 4.7. The four most prominent sugars in each cluster are shown, along with the percentage of the cluster they occupy and the total number of instances in each cluster. Residues where the sugar type is not known are marked as unknown. The majority of residues are GalNAc, which is the most common linkage sugar. The clustering does not differentiate between the type of linkage sugar.

Cluster 1 (124)	GalNAc 93.5%	Mannose 3.2%	Gal-GalNAc 0.8%	Galactosamine, HexNAc[a] 0.8%
Cluster 2 (40)	GalNAc 65.0%	Mannose 30%	GlcNAc 5.0%	
Cluster 3 (19)	GalNAc 78.9%	Mannose 15.8%	Xylose 5.263%	
Cluster 4 (12)	Fucose 50.0%	Mannose 33.3%	GalNAc 16.7%	
Cluster 5 (108)	Mannose 48.1%	GalNAc 46.3%	GlcNAc 4.6%	Xylose 0.9%
Cluster 6 (98)	GalNAc 84.7%	Mannose 10.2%	GlcNAc 5.1%	
Cluster 7 (86)	Hexose 87.2%	GalNAc 10.465	Mannose 2.3%	
Cluster 8 (47)	GalNAc 68.1%	GlcNAc 14.9%	Mannose 10.6%	HexNAc 4.3%
Cluster 9 (34)	GalNAc 70.5%	Mannose 23.5%	GlcNAc 5.9%	

Table 4.8 Percentage membership of clusters generated by farthest first clustering of Thr glycosylation sites

a. both sugars have equal percentage so both have been included.

There is not one single class per sugar and, in fact, GalNAc features prominently in all clusters, although mannose is clearly separated as the dominant sugar in two clusters. Increasing the number of clusters to ten does not alleviate this situation. This suggests that sequence alone is not sufficient to separate glycosylation types based on linkage sugar. It is likely that there are other factors involved, such as the structure of the protein and the properties of the protein surface, in determining which sugar is first attached to the protein at a given glycosylation site. The clustering of Thr glycosylation sites by linking sugar has much in common with Ser glycosylation sites.

The clusters are once again dominated by GalNAc, membership with this sugar featuring prominently in all the clusters. However, there are clusters for several of the significant sugars involved, such as hexose, mannose and fucose.

In general, no cluster for either Ser or Thr is solely occupied by one linkage sugar. This may be in part due to the fact that most glycosylation sites are linked with GalNAc, with other modifications being relatively rare. The clustering may, therefore, be improved by increasing the amount of data available for non-GalNAc glycosylation sites. It may also be that a different representation of the data may be more appropriate for clustering. These experiments use the same representation as the glycosylation prediction earlier in this chapter. Whilst this is appropriate when looking for pair-wise patterns, representing the data with PSSMs may well be better for clustering. However, the idea of clustering glycosylation sites shows some promise if the methodology were to be refined. This could be used in place of predicting individual glycosylation types using separate predictors if the accuracy is increased.

4.4 Conclusions

The random forest algorithm was used to predict glycosylation sites, based on pairwise sequence patterns and the amino acid sequence. The program improved over the best prediction programs currently available, with significant increases in accuracy for the prediction of Thr and Asn glycosylation sites. Neither the addition of structural data, hydrophobicity information, nor surface accessibility data improved the prediction accuracy of O-linked glycosylation, although N-linked glycosylation prediction is improved by the addition of surface accessibility data. However, it may be possible to improve prediction accuracy further through the inclusion of

information on protein disorder and information on the orientation of membrane proteins. It may also be possible to increase accuracy by extending the initial data set, or by considering separately proteins whose PTM is catalysed by the same enzyme. Another option would be to produce prediction programs for each specific glycan type, or to classify each glycosylation site by type of glycan after prediction. Our use of the trepan algorithm allows us to extract comprehensible rules describing features characteristic of a glycosylation site. Our use of clustering to identify the linkage sugar at a glycosylation site was less fruitful, but shows some promise. It may provide an alternative to training predictors for each individual glycosylation type.

4.5 References

1. Walsh CT, Garneau-Tsodikova S, Gatto Jr. JR. Protein posttranslational modifications: the chemistry of proteome diversifications. *Angew. Chem. Int. Ed.* 2005, **44**:7342-7372.
2. Hart GW. Glycosylation. *Curr. Opin. Cell Biol.* 1992, **4**:1017-1023.
3. Seitz O. Synthesis and the effects of glycosylation on protein structure and activity. *Chem. BioChem.* 2000, **1**:214-246
4. Spiro RG. Protein glycosylation: nature, distribution, enzymatic formation, and disease implications of glycopeptide bonds. *Glycobiology* 2002, **12**:43R-56R.
5. Blom N, Sicheritz-Ponten T, Gupta R, Gammeltoft S, Brunak S. Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence. *Proteomics* 2004, **4**:1633–1649.
6. Christlet THT, Veluraja K. Database analysis of O-glycosylation sites in proteins. *Biophysical J.* 2001, **80**:952-960.

7. Gupta R, Jung E, Gooley AA, Williams KL, Brunak S, Hansen J. Scanning the available Dictyostelium Discoideum proteome for O-linked GlcNAc glycosylation sites using neural networks. *Glycobiology* 1999, **9**:1009-1022.
8. Hansen JE, Lund O, Tolstrup N, Gooley AA, Williams KL, Brunak S. NetOglyc: prediction of mucin type O-glycosylation sites based on sequence context and surface accessibility. *Glycoconjugate J.* 1998, **15**:115–130.
9. Eisenhaber B, Bork P, Eisenhaber F. Prediction of potential GPI-modification sites in proprotein sequences. *J. Mol. Biol.* 1999, **292**:741-758.
10. Li S, Liu B, Zeng R, Cai Y, Li Y. Predicting O-glycosylation sites in mammalian proteins by using SVMs. *Comput. Biol. Chem.* 2006, **30**:203-208.
11. Julenius K, Mølgaard A, Gupta R, Brunak, S. Prediction, conservation, analysis, and structural characterization of mammalian mucin-type O-glycosylation sites. *Glycobiology* 2005, **15**:153-164.
12. Breiman L. Random Forests. *Machine Learning* 2001, **45**:5-32.
13. Chen X-W, Liu M. Prediction of protein-protein interactions using random decision forest framework. *Bioinformatics* 2005, **21**:4394-4400.
14. Qi Y, Bar-Joseph Z, Klein-Seetharaman J. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins: Struct., Funct. Bioinf.* 2006, **63**:490-500.
15. Diaz-Uriarte R, Alvarez de Andres S. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics* 2006, **7**:3**.**
16. Bureau A, Dupuis J, Falls K, Lunetta KL, Hayward B, Keith TP, van Erdewegh P. Identifying SNPs predictive of phenotype using random forests. *Genetic Epidemiology* 2005, **28**:171-182.
17. Bao L, Cui Y. Prediction of the phenotypic effects of non-synonymous single

nucleotide polymorphisms using structural and evolutionary information. *Bioinformatics* 2005, **21**:2185-2190

18. Arun K, Langmead CJ. Structure based chemical shift prediction using random forests. 2005, Article number: CMU-CS-05-163 School of Computer Science, Carnegie Mellon University.
19. Sander O, Sommer I, Lengauer T. Local protein structure prediction using discriminative models. *BMC Bioinformatics* 2006, **7**:14.
20. Craven MW, Shavlik JW. Extracting tree-Structured Representations of Trained Networks. In Advances in Neural Information Processing Systems volume 8. MIT Press, Cambridge, MA, 1996.
21. Browne A, Hudson BD, Whitley DC, Ford MG, Picton P. Biological data mining with neural networks: implementation and application of a flexible decision tree extraction algorithm to genomic problem domains. *Neurocomputing* 2004, **57**:275-293.
22. Gupta R, Birch H, Rapacki K, Brunak S, Hansen JE. O-GLYCBASE version 4.0 a revised database of O-Glycosylated proteins. *Nucleic Acids Res.* 1999, **27**:370-372.
23. Witten IH, and Frank E. Data mining: practical machine learning tools and techniques, 2nd edition. Morgan Kaufmann, San Francisco 2005:365-483.
24. Hirst JD, Vieth M, Skolnick J, Brooks CL III. Predicting leucine zipper structures from sequence. *Protein Engineering* 1996, **9**:657-662.
25. Gibrat JF, Garnier J, Robson B. Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs. *J. Mol. Biol.* 1987, **198**:425-443.
26. Obenauer JC, Cantley LC, Yaffe MB. Scansite 2.0: proteome-wide prediction

of cell signalling interactions using short sequence motifs. *Nucleic Acids Res.* 2003, **31**:3635-3641.

27. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 1999, **292**:195-202.
28. Adamczak R, Porollo A, Meller J. Accurate prediction of solvent accessibility using neural networks-based regression. *Proteins: Struct. Funct. Bioinf.* 2004, **56**:753-767.
29. Black SD, Mould DR. Development of hydrophobicity parameters to analyze proteins which bear post- or co-translational modifications. *Anal. Biochem.* 1991, **193**:72-81.
30. Kotsiantis S, Kanellopoulos D, and Pintelas P. Handling imballanced datasets: a review. *GESTS Int. Trans. Comp. Sci. Eng.* 2006, **30**:25-36.
31. Kotsiantis S, and Pintelas P. Mixture of expert agents for handling imbalanced data sets. *AMCT* 2003, **1**:46-55
32. Chawla NV, Bowyer KW, Hall LO, and Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. *J. Artificial Intelligence Res.* 2002, **16**:321-357.
33. Tomek I. Two modifications of CNN. *IEEE Transactions on Systems, Man and Communications* 1976, **6**:769-772.
34. Estabrooks A and Japkowicz N. A Mixture of experts framework for learning from imbalanced data sets, in Advances in intelligent data analysis. *Lecture Notes Comput. Sci.* 2001, 2189:34-43.
35. Chan PK and Stolfo SJ. Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In Proceedings of the fourth international conference on knowledge discovery and data mining

2001, pp164-168.

36. Yan R, Liu Y, Jin R and Hauptmann A. On predicting rare classes with SVM ensembles in science classification. In IEEE international conference on acoustics speech and signal processing, 2003.
37. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochem. Biophys. Acta* 1975, **405**:442-451.
38. Zar, J.H. BioStatistical Analysis. 4th edition. Prentice Hall, Upper Saddle River NJ; 1970:p. 633.
39. R Development Core Team (2007). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.
40. Carega C, Sinapov J, Silvescu A, Dobbs I, and Honaver V. Glycosylation site prediction using ensembles of support vector machines classifiers. *BMC Bioinformatics* 2007 **8**:438.
41. Chen YZ, Tang YR, Sheng ZY and Zhang Z. Prediction of mucin-type O-glycosylation sites in mammalian proteins using the composition of k-spaced amino acid pairs. *BMC Bioinformatics* 2008 **9**:101.
42. Hochbaum DS, and Shmoys DB. A Best possible heuristic for the k-center problem. *Math. Oper. Res.* 1985 **10**:180-184.

Chapter 5: Conclusions

The topics discussed in this thesis are at the centre of bioinformatics and computational biology. The prediction of dihedral angles has many potential uses in protein structure prediction and determination. Glycosylation is a key PTM, which is heavily involved in the biology of all organisms. It is involved in the regulation of various biological processes and is be important in signalling between cells, in the immune system and in many other aspects of biology.

Our hypothesis for the work in chapter 3 was that using the as yet untested method of SVR would improve prediction accuracy. We also wished to test the use of predicted secondary structure to enhance the prediction of dihedral angles. For this reason our initial experiments replicated the work of Wood and Hirst[1]. Reproduction of the Cascade Correlation networks predictor was based on incomplete information and was thus not entirely successful. We nevertheless obtained some potentially useful predictions, although the accuracy was well below that of state of the art methods such as PsiPred[2]. For the remainder of the chapter we focused on dihedral angle prediction, both with and without the secondary structure predictions obtained from CASCOR. Dihedral angles have potentially more use than secondary structure predictions. They give more information about the 3D structure than a three state secondary structure prediction, since it is possible to infer the orientation of the elements of the protein structure from the dihedral angles. It is also possible to use predicted dihedral angles as restraints for molecular dynamics and as a starting point for a complete 3D structure.

We began by discussing the choice of the Gaussian kernel in the SVR. The selection

of the kernel is a crucial aspect, as is borne out by the differing accuracy of the kernels. The Gaussian kernel was clearly better based on our results. Optimising of the user defined parameters of the SVR was also a crucial step to undertake. These parameters can potentially have a large effect on the results of a given experiment. The optimisation gave us a useful, but surprisingly small, improvement over un-optimised SVR. We used predicted secondary structure information to enhance the sequence representation and improve Ψ dihedral angle prediction. We also used normalization methods that have been previously employed by Real Spine 2[3]. Structural information only resulted in a small gain in accuracy. However, the normalisation method resulted in a large improvement. We predicted Φ using similar methods, again using Real Spine 2's normalisation method. No predicted secondary structure was added to the sequence representation for Φ prediction, since it does not have the same relevance. The best results are given by the normalised predictions in the case of Ψ with predicted structural information. In comparison to other work this method does not beat the state of the art methods such as Real Spine, though it is more accurate than the prediction of dihedral angles with Destruct[1]. However, the accuracy is far behind that of Real Spine XI[4], which was published during the writing of this thesis. The conclusion reached is that there is no improvement over the state of the art methods to be gained using SVR, although SVR is an improvement on Cascade correlation networks, and indeed there is also a loss of speed over other methods. In the wider context of protein structure, prediction of the dihedral angles would be extremely valuable, and is an important goal on the route to computational determination of a protein's structure. In the future the accuracy of dihedral angle prediction can still be improved upon and it is likely that this improvement will take the route of highly focused neural networks grouped into ensembles with each

predicting a part of the complete protein structure.

Also as part of chapter 3, we investigated a potential application of our dihedral angle predictions in NMR assignment, using Φ angles to assist in the solving of the Karplus equation. Such an application has not been attempted before, but would be a useful tool for the assigning of NMR spectra of proteins. This application still has the potential to be successful. However, this will require a much greater accuracy of the experimental measurements of 3 bond J couplings and of the prediction of dihedral angles, as our experiments here have shown. Currently, dihedral angles are assigned using the Talos software[5] from a relatively small subset of proteins. Using the Karplus equation would allow the assignment of dihedral angles without referring to an example dataset, but this depends on the ease of obtaining the experimental results.

In Chapter 4, we presented our research into predicting glycosylation sites. To test the feasibility of our initial hypothesis, we first looked for patterns in the amino acids surrounding glycosylation sites. Our frequency analysis shows that there are indeed amino acids, which are significantly increased or decreased in proximity of glycosylation sites. This has the implication that there are underlying motifs in the sequence that confer an increase in the likelihood of glycosylation. Pairwise patterns were generated within sequence windows. The frequency of occurrence of these pairs of amino acids shows that some patterns are much more likely to occur around glycosylated residues. This information was used to weight sequences, which were then used to train random forest. We balanced the dataset utilising an ensemble of random forests each trained on a balanced dataset created by under sampling from the training data. The sampling of the negative (majority class) was carried out without

replacement so the maximum information is retained. This method gave a large improvement in accuracy over both the unbalanced data and balancing the data using random under sampling alone. The resulting prediction program, GPP, produces accurate predictions of both N-linked and O-linked glycosylation sites. Comparing these predictions to other prediction methods it is clear that GPP is the best prediction method available. It scores higher in both accuracy and in Matthews correlation coefficient, indicating it makes significantly fewer false positives than other methods. The most recent prediction methods, are close to GPP in overall accuracy, but worse in at least some areas. In terms of interpretability all methods are equally black box in nature, making our work to interpret the random forest all the more important. In the wider context GPP is a useful tool for allowing the targeting of experiments, and for computational analysis of a large number of sequences. It is the best software available for these tasks.

The model generated by GPP is opaque, i.e. non-interpretable. To address this, the trepan algorithm was used to generate comprehensive rules in the form of a decision tree that represents the model generated by random forest. This showed some rules that were in agreement with experiment. There were, however, some details missed by trepan, possibly due to the nature of the *m* of *n* rules used therein, which did not seem to allow negative selection. So some sequence motifs that block glycosylation were not shown. There was also at least one rule that was puzzling in that it has no experimental evidence supporting it. Despite all this, the rules produce were able to provide some insight into the glycosylation process. These rules are potentially useful both for the understanding of the biological mechanism that dictates whether a residue is glycosylated or not, and to understand the weak points of the machine learning

method and suggest potential improvements. As a further experiment farthest first clustering was employed to classify the glycosylation site by linkage sugar type. This had little success and a different approach may be required in order to successfully determine the linkage sugar from sequence information. Such information would help to augment proteomics studies by indicating what may be attached to a given glycosylation site. However, the sugars attached may be tissue specific, so it may not be possible to determine the glycan attached to a glycosylation site without referring to factors in the surrounding environment.

The work discussed here gives scope for future studies. Of the work in chapter 4, both the novel balancing method and the pairwise pattern method could conceivably be used in other work. The general method of weighting pairwise patterns could be applied to many different problems involving the classification of sequences. Particularly interesting would be to apply this to the prediction of other PTMs for which it would be very much suited. It would also be interesting to see if such a method could be used to improve upon the dihedral angle prediction method of chapter 3, since pairs of residues might denote hydrogen bonds that are important for defining a particular type of secondary structure. The balancing method can be used to balance any set of imbalanced data, and it would be interesting to look into the benefits of this method in a wider range of situations. The future course of the research discussed here promises to lead to the use of accurate predictions of protein structure which can be used for design of proteins and for targeting experimental research, although this work leads the methods used away from SVR and suggests that the methods employed by Real Spine XI are a better bet. Future prediction of PTMs will yield both targeted experiments but also potential drug targets and a greater

understanding of some of the fundamental process in regulation of the cell and in intercellular signalling as well as of the diseases associated with these areas of biology.

5.1 References

1. Wood MJ, and Hirst JD. Protein secondary structure prediction with dihedral angles. *PROTEINS: Struct. Funct. Bioinf.* **59**:476-481.

2. Jones DT, Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 1999, **292**:195-202.

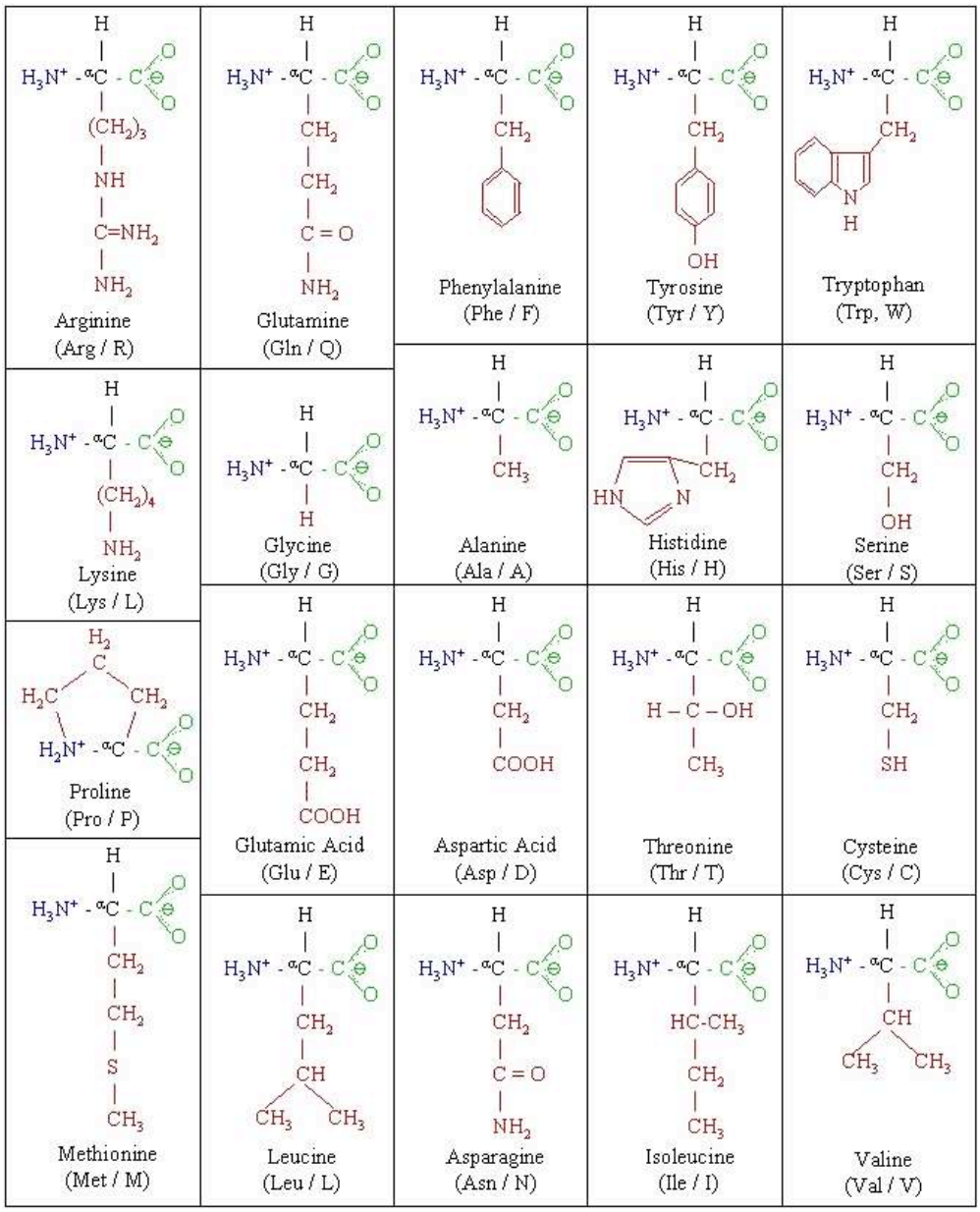
3. Dor O, and Zhou Y. Real-SPINE: An integrated system of neural networks for real-value prediction of protein structural properties. *PROTEINS: Struct. Funct. Bioinf.* 2007 **68**:76-81.

4. Faraggi E, Yang Y, Zhang S, and Zhou Y. Predicting consensus local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. *Structure* 2009 **17**:1515-1527

5. Cornilescu G, Delagio F, and Bax A. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *J. Biomol. NMR* 1999, **13**:289-302.

Appendix A

Structures of the 20 standard amino acids



Appendix B

The following tables contain the complete frequency statistics for glycosylated Ser, Thr and Asn residues. Significant increases are shown in bold significant decreases in italics. The results in these tables as well as the methods used to obtain them are discussed in chapter 4.

Table B.1 Frequency statistics for glycosylated Asn residues

Position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
A	19	15	18	17	20	18	15	23	*0*	15	21	22	15	16
C	12	9	12	12	10	4	5	11	5	**17**	11	8	16	9
D	13	**23**	10	8	9	13	7	8	*0*	14	6	17	12	19
E	19	11	11	12	21	13	19	7	*0*	15	21	18	19	27
F	6	12	10	6	16	13	10	12	*0*	7	7	5	15	4
G	19	19	15	18	14	9	20	30	*0*	9	16	17	19	12
H	6	7	6	8	9	8	5	8	0	7	9	7	10	6
I	10	13	15	11	10	6	7	19	*0*	12	10	4	13	6
K	10	6	11	9	9	14	11	12	*0*	17	10	9	9	17
L	30	18	23	26	27	28	22	25	*0*	22	21	18	23	13
M	13	2	**10**	6	3	5	3	5	0	6	4	5	2	4
N	12	7	14	13	9	18	20	12	*0*	8	8	12	9	16
P	13	22	15	20	7	15	11	*1*	*0*	6	**31**	18	18	14
Q	9	10	10	16	15	**21**	7	8	*1*	11	13	16	9	11
R	14	7	12	10	15	10	15	10	*0*	15	5	13	10	8
S	20	22	25	16	16	14	24	23	**102**	32	23	28	11	32
T	14	26	15	23	17	17	15	17	**151**	16	16	13	19	22
V	14	13	16	17	15	19	25	17	*0*	18	15	18	19	12
W	7	4	6	0	7	4	6	2	0	6	3	3	3	3
Y	6	14	6	12	12	12	13	10	*0*	6	9	8	7	5

Table B.2 Frequency statistics for glycosylated Ser residues

Position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
A	22	36	24	36	29	30	**37**	31	34	21	26	30	25	19
C	4	7	3	3	4	13	3	7	3	17	6	5	6	5
D	6	12	6	18	14	10	6	4	*3*	10	11	8	5	*5*
E	15	19	22	19	23	24	8	16	10	13	10	11	9	*6*
F	6	3	4	7	4	3	*4*	4	7	6	2	6	2	5
G	19	26	30	17	20	22	27	40	27	27	16	23	19	41
H	9	5	4	8	6	9	1	4	0	4	6	8	6	13
I	13	13	7	6	8	11	16	9	15	7	7	4	17	8
K	6	8	15	11	*2*	6	5	9	11	*3*	*4*	11	12	8
L	20	*12*	20	20	15	*11*	13	18	*12*	15	24	16	17	*15*
M	16	6	10	5	7	4	3	0	5	8	3	5	4	5
N	16	10	12	9	12	12	6	6	7	4	8	21	9	10
P	31	**38**	35	35	**41**	34	**46**	28	**40**	**51**	**42**	34	32	35
Q	8	11	11	9	14	12	9	11	6	9	14	16	11	8
R	12	5	12	11	8	7	4	11	8	8	*0*	4	11	8
S	**56**	42	43	**54**	**53**	**56**	47	48	**60**	43	**56**	41	**48**	**49**
T	**58**	**43**	**46**	**41**	**48**	34	**62**	**61**	**48**	**50**	**58**	**51**	**51**	**47**
V	17	24	24	20	19	23	28	21	26	24	26	18	20	19
W	3	9	0	4	4	3	2	0	1	1	1	0	5	2
Y	6	5	8	4	5	8	8	2	5	5	3	8	7	4

Table B.3 Frequency statistics for glycosylated Thr residues

position	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
A	59	54	**60**	**66**	48	52	**81**	51	**77**	62	47	41	41	47
C	11	3d	16	*0*	*8*	*3*	*3*	*13*	*3*	*2*	*4*	9	*5*	8
D	17	36	21	30	21	*10*	*7*	15	19	*10*	17	29	21	*14*
E	42	36	41	36	45	30	*12*	40	29	20	33	40	43	29
F	*6*	*10*	*4*	12	13	8	12	*8*	9	10	12	*4*	14	13
G	52	40	**108**	42	*21*	**122**	32	**101**	40	*16*	**106**	30	37	**112**
H	12	6	7	8	15	4	12	6	20	6	17	17	18	13
I	20	16	16	33	23	**40**	*14*	17	17	13	18	29	14	18
K	25	18	17	16	16	*12*	*8*	*7*	*13*	*7*	19	18	18	16
L	*19*	*25*	34	35	35	26	*14*	*15*	*27*	*20*	*26*	34	34	*17*
M	20	10	5	11	12	9	8	14	13	6	7	13	3	3
N	20	10	14	17	22	18	19	15	24	17	18	21	*14*	15
P	**77**	61	**81**	67	**99**	62	**128**	**103**	68	**167**	59	**105**	59	**89**
Q	22	27	15	25	30	14	19	22	15	28	17	20	24	17
R	33	22	17	28	*8*	*13*	21	*11*	25	*9*	15	30	30	16
S	81	67	65	68	74	70	**89**	67	76	74	63	59	52	70
T	**101**	**168**	**96**	**130**	**117**	**115**	**107**	**95**	**118**	**131**	**127**	**95**	**156**	87
V	47	40	37	33	50	46	69	51	42	35	30	42	32	36
W	2	10	4	1d	2	4	2	2	5	2	3	*0*	*0*	*0*
Y	11	6d	12	11	10	7	11	7	11	11	5	*4*	16	8

Appendix C

The following are the complete decision trees generated by trepan for Asn, Ser and Thr glycosylation. Full analysis and methods is given in chapter 4.

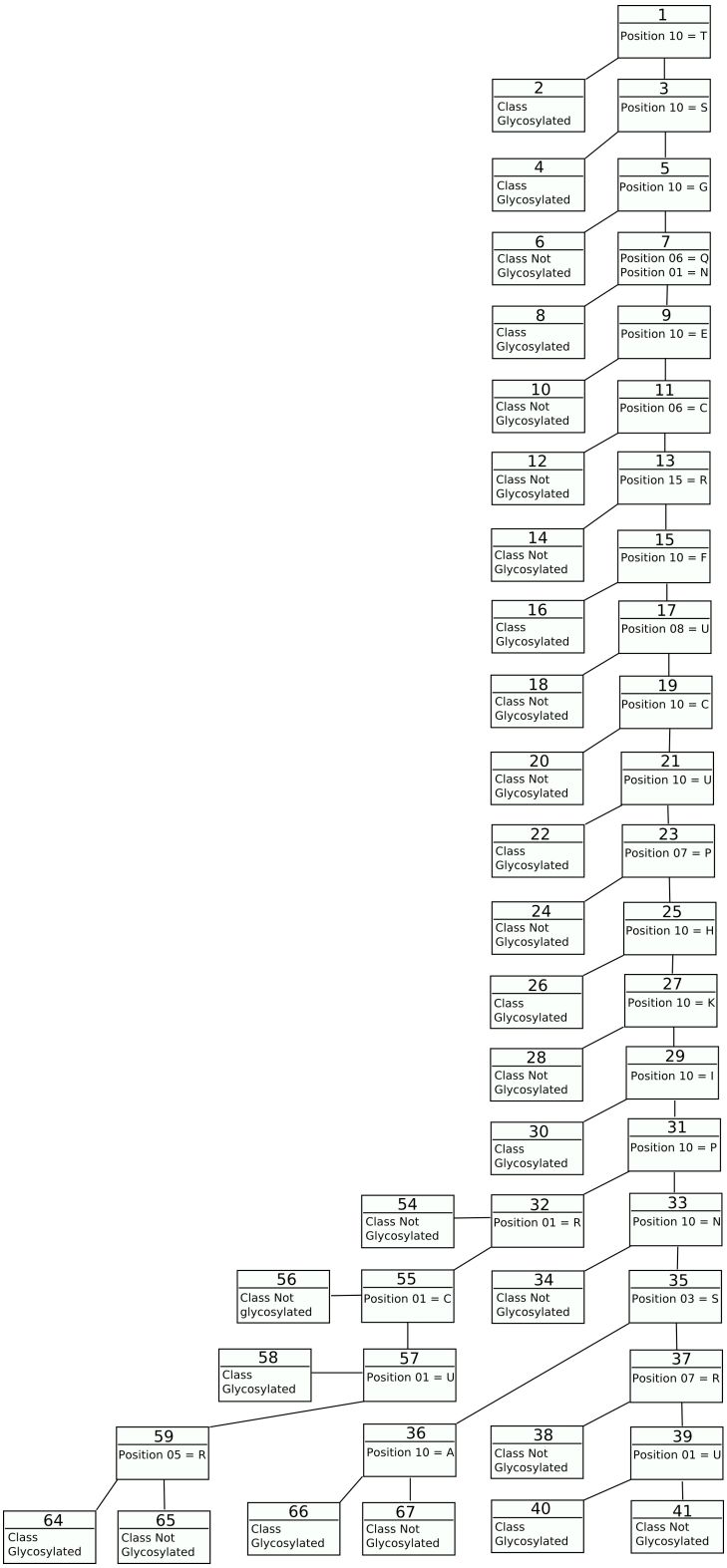


Figure C.1 Complete decision tree for Asn glycosylation sites

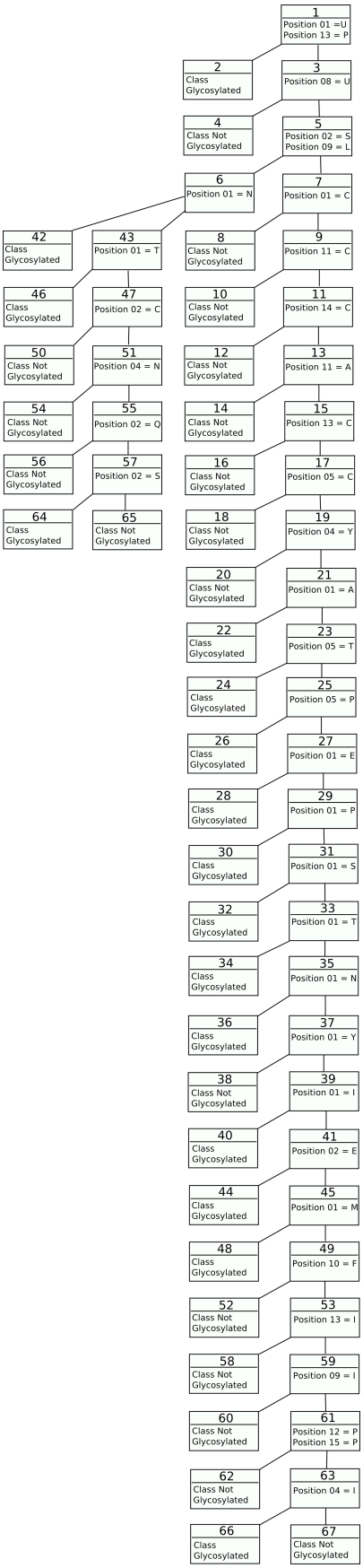


Figure C.2 Complete decision tree for Ser glycosylation sites

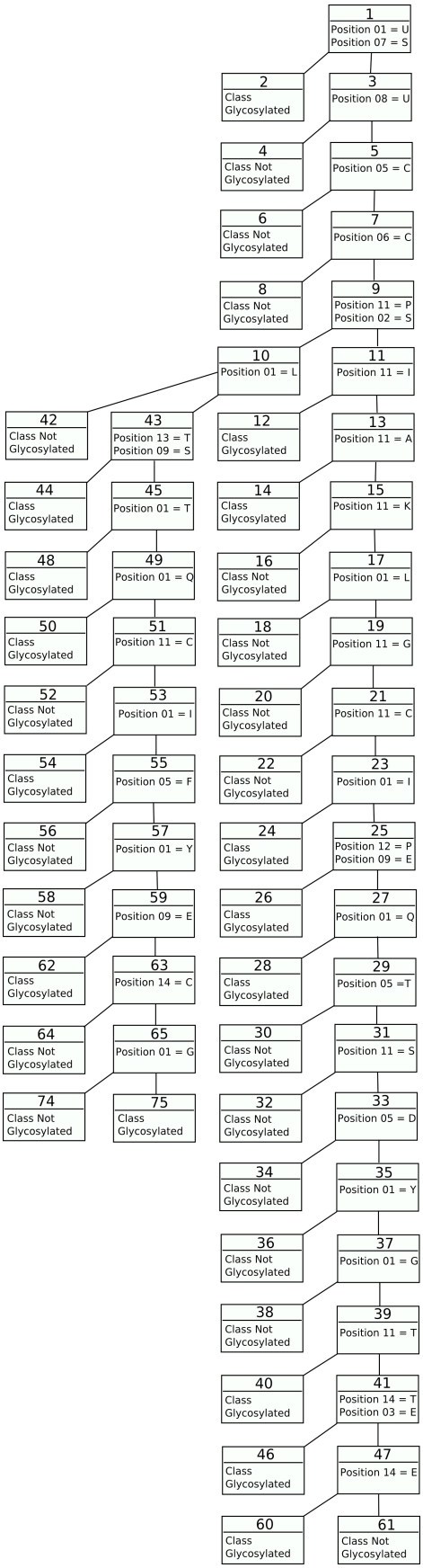


Figure C.3. Complete decision tree for Thr glycosylation sites

Appendix D

Listed below are the commonly found monosaccharides encountered in this thesis

Monosaccharides

Name	Abbreviation
Galactose	Gal
D-Glucose	Glc
D-Mannose	Man
L-Fucose	Fuc
D-Xylose	Xyl
D-Glucuronic acid	GlcA
N-Acetyl-D-galactoseamine	GalNAc
N-Acetyl-D-glucoseamine	GlcNAc
N-Acetyl-Neuraminic acid	NeuAc