Supervised Machine Learning in Bioinformatics: Protein Classification

Swansea University
Prifysgol Abertawe

MSc by Research

Department of Computer Science

Author:
Arron Lacey

Supervisor:
Dr. Xianghua Xie

June 14, 2014

Declaration

I, Arron Lacey, declare that this thesis titled, 'Supervised Machine Learning Techniques in Bioinformatics: Protein Classification' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Acknowledgements

I would like to thank my supervisor Dr Xianghua Xie for all the support and direction to guide me through the work documented in this thesis.

Abstract

Since the introduction of next generation sequencing (NGS) there has been demand for sophisticated methods to classify proteins based on sequence data. Protein sequences have traditionally been performed in the lab by biologists, however open source genetic datasets have grown exponentially since the inception of NGS, and multiple sequence alignment can yield statistical information about a group of proteins that may not be found in traditional analyses. The motivation behind this thesis is to investigate different approaches to protein classification through machine learning processes.

Two main approaches for this task are to use the raw sequence data and align them against other sequences, or to extract discrete high level features from the protein sequences and compare the features. Multiple sequence alignment of proteins within a known family has been a popular way to extract statistical information about a protein family that would not otherwise be observed to the human eye, and thus many algorithms focus on this method of protein classification. A more generative process would be to extract known biological features as a basis for classification.

Profile Hidden Markov Models are built from multiple alignment of raw sequence data and Random Forests are used to discriminate between two sets of proteins based on features such as functional amino acid groups properties extracted from the raw sequences. HMM's show that sequence alignment approaches determine small differences between similar proteins well, and Random Forest specializes in classifying proteins that do not share high sequence homology within the family as biological features are the basis for classification.

The conclusions of this thesis is to recognize the high classification accuracy of two relatively unused machine learning techniques in comparison to older techniques such as Artificial Neural Networks and Support Vector Machines. It is important to note that particular techniques are suited to particular problems, and it is important to use the right technique.

Contents

List of Figures

Introduction

Given the complexity and gigantic volume of biological data, the traditional computer science techniques and algorithms fail to solve complex biological problems of the real world. However, modern computational approaches such as machine learning can address the limitations of the traditional techniques. Machine learning is an adaptive process that enables computers to learn from experience, learn by example, and learn by analogy and their capabilities are essential tools to help us make sense of the biological data existing in exponentially expanding open source databanks. Analyzing protein data is the most comprehensive way to study an organism, and as such it is important to be able to classify individual proteins in an organisms' proteome. This thesis aims to give an overview of how it is possible to use machine learning techniques to classify proteins into known families by means of two different approaches - performing pattern recognition on raw protein sequences (Hidden Markov Models), and extracting high dimensional meta-features from protein data (Random Forest) to classify proteins.

Motivation

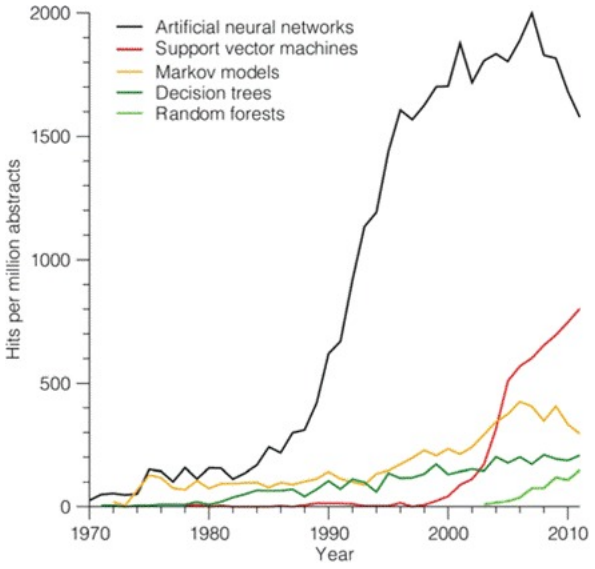
Machine learning has played a major role in establishing Bioinformatics as a field in it's own right over the last 30 years. A myriad of techniques ranging from randomized decision trees to neural networks, support vector machines and hidden Markov models have been applied successfully to solve problems in novel gene finding, evolutionary analysis, drug and agricultural research and protein classification. The motivation of this thesis is to show how successful machine learning can be when applied to biological data, as well as pointing out it's limitations and what technique is appropriate for a certain problem or dataset. The motivation behind this thesis was to learn how to apply machine learning techniques to biological data, whilst gaining knowledge of Bioinformatics and genetics. As such, the most important element

of the work carried out in this thesis was to learn what machine learning techniques work best for certain problems, rather than score different techniques against each other for the sake of competition. The particular focus is on protein classification - a discipline that guides many important aspects of drug discovery, agricultural solutions and assessing the effects of protein mutations on disease phenotypes. In the last 25 years, neural networks have been a popular choice in Bioinformatics solutions because of their simple implementation, protein classification is no different. The work in this thesis therefore focusses on two newer and less known techniques : randomized decision trees and hidden Markov models. Recent research has shown these two techniques, particularly hidden Markov models to achieve similar classification results, if not better than more traditional techniques and provide a powerful statistical approach to protein classification that can solve problems that other techniques either can't, or the implementation is not suited to the problem to be a desirable solution. By using these two techniques and comparing to others in the literature, the strengths and weaknesses of randomized decision trees and hidden Markov models are explored.

Overview

There are several biological domains where machine learning techniques are applied for knowledge extraction from data. The main areas of interest can be split into six different domains: genomics, proteomics, micro-arrays, systems biology, evolution and text mining. The greatest success stories have come mainly from supervised learning techniques. Artificial Neural Networks have been the leading and most popular technique to be used since the 70's, where Markov models were also used to a lesser degree. It was only from 2000 onwards that support vector machines found great popularity and saw the usage of AN-N's decrease by 21% since 2007.

In the late 90's Markov models were transformed into profile hidden



`http://www.ncbi.nlm.nih.gov/guide/data-software/databases`

Figure 1: Machine learning techniques used in bioinformatics and biology papers. Artificial Neural Networks have been a popular choice for Bioinformatics solutions as well as protein classification

Markov models [18] [27]. Using statistical information of aligning multiple sequences together, profile HMMs provide a unique classification and segmentation solution that is tailored to the data provided during training. The flexibility of such models became very popular and are used almost as much as support vector machines. As well as SVM's, another ensemble method using randomized decision trees called random forests [8], have gained a modest amount of popularity since 2003. Often random forest methods, when compared to other ensemble methods like SVM's equal or achieve higher classification accuracy. This thesis will study protein classification using profile Hidden Markov models and Random Forests.

Thesis layout

The thesis is set out as follows:

Chapter 1 - Background: A brief history of bioinformatics and biological data is presented, as well as the problems faced when the sheer size and depth of the data limits calls for sophisticated methods of analyzing the data. A literature review of popular machine learning techniques is presented.

Chapter 2 - Randomized Decision Trees - An ensemble based machine learning method that makes use of randomized decision trees is presented - forming the implementation of the random forest algorithm. The background and theory is explored, ending with an experiment to classify to very different types of protein families - transmembrane and antifreeze proteins.

Chapter 3 - Hidden Markov Models - A stochastic probalabistic model that utlizies statistical information from aligning multiples protein sequences together. Hidden Markov models are formulated from Markov chains and a unique implementation, profile Hidden Markov Models is explored to classify transmembrane and antifreeze proteins

Chapter 4 - Discussion - The last chapter of this thesis is a final discussion about what has been learned in the undertaking of the work in this

thesis.

Chapter 1

Background

In 1953 Watson and Crick first described the structure of DNA. 22 years later in 1975 F.Sanger, A.Maxam and W.Gilbert took the next step and described methods to sequence the DNA of organisms. Over the next 20 years their methods were used to sequence entire genomes such as the fruit fly, yeast, roundworm and the influenza virus, leading to a race to sequence the human genome in it's entirety. On 26^{th} June, 2000, President of the United States Bill Clinton and UK Prime Minister Tony Blair held a press conference to announce the completion of the first sequenced human genome. 3 billion base pairs of DNA collected over a decade, thanks to the efforts of the Human Genome Project. It was a landmark of scientific achievement that changed the face of biology and medicine. The efforts of the Sanger Institute, Celera, the National Human Genome Genome Research Institute and other organisations declared a tie, and the human genome was published. A new window had been opened to gene hunting,gene therapy and drug discovery, but had also provided a beacon for funding an effort to commercialize the sequencing of the human genome. This landmark ranked among the technological achievements that put man in space.

	Landmarks in the Human Genome Project
1953	Watson-Crick publish DNA structure
1975	F.Sanger, A.Maxam and W.Gilbert develop methods for sequencing DNA
1977	Bacteriophage $\Phi X - 174$ sequenced: first complete genome sequenced
1980	US Supreme court rules genetically modified bacteria are patentable
1981	Human mitochondrial DNA sequenced: 16569 base pairs
1984	Epstein-Barr virus genome sequenced: 172,281 base pairs
1990	International Human Genome Project launched
1991	J. Craig Venter identifies sequences of DNA complementary to messenger RNA
1992	Complete low resolution linkage map of the human genome
1992	*Caenorhabditis* sequencing project begins
1992	J. Craig Venter forms the Institute for Genome Research (TIGR)
1992	Wellcome Trust and UK Medical Research Council establish The Sanger Center for large-scale genomic sequencing
1995	First complete sequence of bacterial genome, *Haemophilus influenzae* by TIGR
1996	High resolution map of human genome
1996	Completion of yeast genome, first eukaryotic genome sequence
1996	Celera claims to finish sequencing human genome by 2001, Wellcome Trust respond by increasing funding to the Sanger Center
1998	*Caenorhabdtis elegans* genome published
1998	*Drosophila melanogaster* genome published
1999	Human Genome project says it will sequence human genome with 2 years
1999	Sequence of first human chromosome published
2000	Joint announcement of complete sequence of human genome

Up until the genome sequencing methods were successfully implemented

biology had been an observational science, relying mainly on the powers of deduction and came with varying degrees of precision, but genome sequencing has provided biologists with *discrete* data to determine th structure of an organism's DNA both *completely* and *entirely*. As such the precision of findings from genome sequenced data are staggeringly high. With data of this quality, the scope of biology and molecular genetics is vast:

- Understanding organisms on a macroscopic and microscopic scale through their DNA and amino acid sequences
- Inserting cell structure and function from DNA/protein sequences T
- Using data in an organism to deduce evolutionary relationships between other organisms D
- Supporting applications to medicine,agriculture and technology
- Curating databases and web servers that store and allow access to genes,proteins and nucleotide sequences found when new organisms are sequenced

1.1 The role of computer science in bioinformatics

Since the first human genome was published in June 2000, the last decade or so has seen the sharp rise of a field that merges biology,genetics and computer science: **Bioinformatics**. Bioinformatics is a discipline that has grown from the methods described back in 1975, but it's a discipline that would not be possible without both the expertise of biologists and without the advances in computer software and computer hardware. Bioinformatics can be thought of as an area that tackles two main issues that arose from whole genome sequencing.

1.2 Data - size,complexity and structure

The human genome spans 3 billion base pairs of nucleotides which amounts to roughly 3 Gb of data. Needless to say the data in genomes are *huge*, in fact so huge that the equivalent amounts of data that comprise human genomes are called ***huges***. Storing such data, quality checking and ensuring they can be accessed freely are truly mammoth tasks, yet there are many bioinformatics organizations that dedicate their work to doing just this. As of July 2013 only 189 non-bacterial genomes and 3762 bacterial have been sequenced, where databanks that store these nucleotides such as Genbank and The European Molecular Biology Laboratory (EMBL) databank [5],[47] have already surpassed 1×10^{12} base pair sequences. Given that current estimates of the amount of species there are on earth are at around 8.7 million [30], to sequence all of them appears to be incomprehensible. Nevertheless the bioinformatics community response has been rapid development of sequence databanks that store high precision data, and grow exponentially year by year.

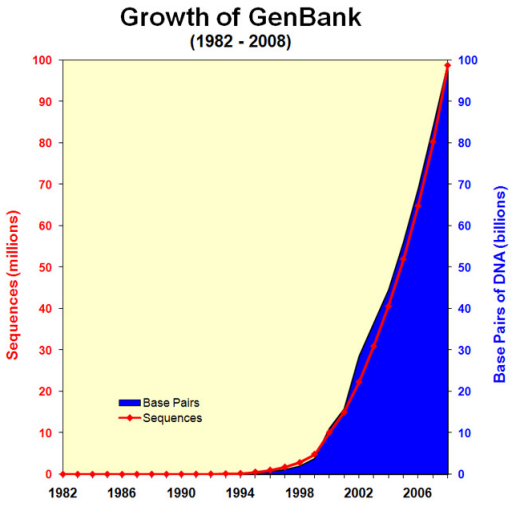


Figure 1.1: The growth of Genbank. Since the inception of the National Center for Biotechnology Information, over 100 billion base pairs of DNA have been sequenced. `http://www.ncbi.nlm.nih.gov`

Many databanks are curated and maintained by the open source commu-

nity, and as most things open source they reach a high degree of convergence and standards. Most databanks will generally include archives of structured information such as the raw nucleotide sequence and meta-data like gene locus location. Databanks also need to provide tools to access their data, usually via websites or ftp servers. The data also go further than just storing nucleotide sequences - proteins, secondary/tertiary structures, expression patterns and metabolic pathways all have their place in their dedicated databanks. Data curation across databanks maintain standards, beginning with the way in which raw sequences are stored ; usually in the FASTA file format (as well as other similar variations).

```
>gi|142864|gb|M10040.1|BACDNAE B.subtilis dnaE gene encoding DNA primase
GTACGACGGAGTGTTATAAGATGGGAAATCGGATACCAGATGAAATTGTGGATCAGGTGCAAAAGTCGGC
AGATATCGTTGAAGTCATAGGTGATTATGTTCAATTAAAGAAGCAAGGCCGAAACTACTTTGGACTCTGT
CCTTTTCATGGAGAAAGCACACCTTCGTTTTCCGTATCGCCCGACAAACAGATTTTTCATTGCTTTGGCT
GCGGAGCGGGCGGCAATGTTTTCTCTTTTTTAAGGCAGATGGAAGGCTATTCTTTTGCCGAGTCGGTTTC
TCACCTTGCTGACAAATACCAAATTGATTTTCCAGATGATATAACAGTCCATTCCGGAGCCCGGCCAGAG
TCTTCTGGAGAACAAAAAATGGCTGAGGCACATGAGCTCCTGAAGAAATTTTACCATCATTTGTTAATAA
ATACAAAAGAAGGTCAAGAGGCACTGGATTATCTGCTTTCTAGGGGCTTTACGAAAGAGCTGATTAATGA


>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Figure 1.2: Above a fasta file using DNA nucleotides.**Below** a fasta file using amino acid denotation. Fasta files are the basic file format for protein analysis, where many high level features can be extracted such as secondary structure, hydrophobicity and functional groups within the sequences.

The largest databank of nucleotide sequences, the International Nucleotide Sequence Database Collection is curated by collaborative effort of *EMBL Nucleotide Sequence Database* at the European Bioinformatics Institute (EBI), Hinxton UK, *Genbank* at the US National Center for Biotechnology Infor-

mation (NCBI), Maryland and *The Center for Information Biology and DNA Databank of Japan*. Similar the largest archive of amino acid sequences is a merger of *SWISS-PROT*,*The Protein Identification Resource (PIR)* and *Translated EMBL -TrEMBL* databases - forming the United Protein Database (UniProt). Some smaller databases may focus on certain organisms and become highly specialized. The 1000 Genomes Project [45] that was created in April 2007 and provides by the most comprehensive catalogue of human genetic variation, where up to October 2012 1092 human genomes had been sequenced and open access is available to these data. The Gene Expression Omnibus (GEO) [19] provides gene expression data, while specialized protein databases exist such the Transport Protein Database (TCDB) [42] provides details information on transmembrane proteins and is used later in this thesis.

All of these databases provide tools to access their data. Most databases will use an ftp server on their website to access files in their database, where some of the larger databases such as NCBI have developed retrieval systems (*ENTREZ*) that run on their website, or even mobile phone apps!

Another approach is to input data into the database and search for similar sequences. The Basic Local Alignment Search Tool (*BLAST*) [2] is one the most widely used bioinformatics tools. It is an alignment tool that takes DNA/amino acid sequences and aligns the sequences to similar sequences in the database by picking out short chunks of the input sequence that match other chunks of sequences in the database. Tools like ENTREZ and BLAST allow researchers to answer the following questions:

- Search *nucleotide* sequences in a databank that are similar to a sequence of interest. Finding novel genes usually starts with sequencing a cohorts genomes and sending potential novel sequences through a tool like *ENTREZ* or *BLAST* that will either match up with sequences in the databanks by some matching criteria, or if no matches are found it may be that a novel sequence has been found. Such matching al-

gorithms are discussed later, in which they vary from string-matching criteria to statistical inference algorithms.

- Search *protein* sequences in the database that are similar to a protein sequence of interest. Such matches may be made upon three dimensions - primary,secondary an tertiary structure of the protein, all of which have their own dedicated databases and algorithms to predict higher dimensions from the raw 1-D data.

- If one has a protein sequence of unknown structure, search the database to find proteins of similar structures and possibly classify the test protein into a known family of proteins. Some proteins are highly functional in terms of their secondary or tertiary structure. If a drug company was looking for proteins with a certain type of binding site determined by their protein structure, they may code up the binding site, and find proteins by searching the database on their mocked up protein structure.

- The reverse of the last point is that given a protein structure, find any sequences which relate to it in the databank. Some structures are highly dependant on specific amino acids being present in there sequences. This is rare however as protein structures that require certain amino acids are scarcely found in nature.

The scientific community puts a great deal of trust in databanks for their research and rely on the quality of not only sequencing precision, but of the annotative meta data that come with them. Each databank goes through peer review and have standard quality indices imposed on them to keep any errors to a minimum. The most common indices used are by reviewing the methodology that was used to create a dataset. Older databanks may have used peptide sequencing to determine the amino acid sequence, but as computing power increased and hardware used to sequence genome

improved, the raw DNA was sequenced and translated into amino acid sequences. Annotations and meta data will include information on how the data was retrieved, what organism the algorithm used to glean the information from the raw data and the authors that annotated it. If curators of data create their own meta data, they are mindful of being able to link their data in other databases, or would otherwise link into other databases the generate the relative meta data. Since the data available from genome sequencing is so huge, variables inferred from these data cannot be carried out by a specialist eye-balling the data, so many databanks generate their inferred data entirely from computer programs which will have published their methods before-hand, or used previously published methods in their analysis. Secondary structure of proteins for example can be predicted by *PSIPRED* [23] using raw amino acid sequence data, where it is understood that the prediction holds a certain accuracy. As one can imagine with such large datasets, annotation of nucleotide and amino acid sequences is a difficult task, which are almost entirely automated by computer programs, and with a demand for both *quantity* and *quality* of the databanks, it is only natural that bioinformatics has turned to more complicated methods than simple queries against databases e.g. machine learning.

1.3 Analyzing the data

The vast amount of data made available from genome sequencing is a welcome problem. Data of this magnitude can not be eye-balled by experts, and so it is the role of computer science that allows us to extract useful information from DNA/amino acid sequences. Many of the organizations that hold databanks have developed their own tools to extract useful data from user input, and will even use that information as criteria to search their own databases. Some areas of sequence analysis are:

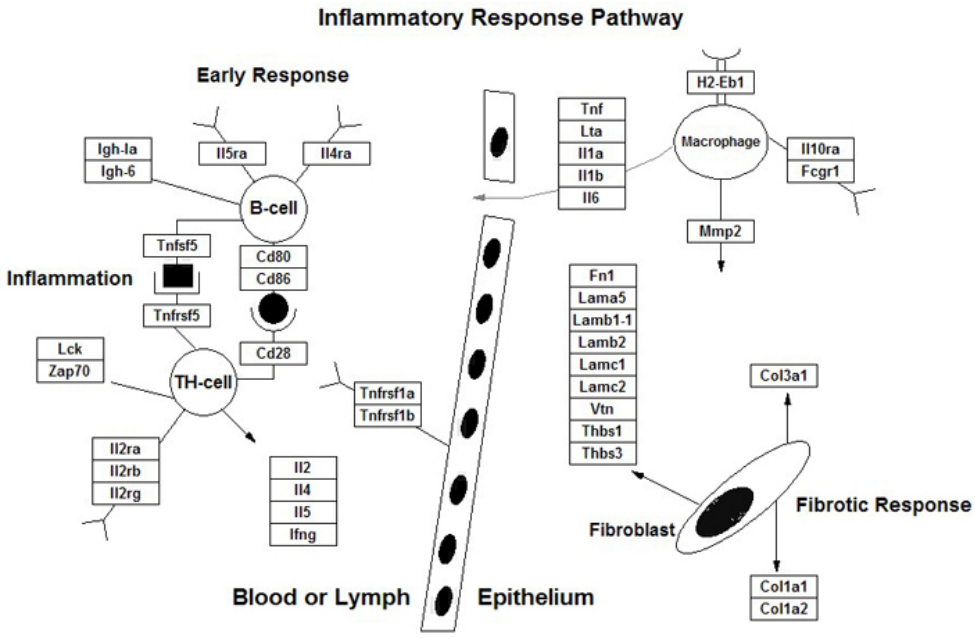
- Single Nucleotide Polymorphism analysis - analysing how mutated genes

impact phenotypes

- Phylogenetic analysis - using DNA to determine evolutionary relationships between organisms
- Protein structure analysis - use amino acid sequences to predict not only higher dimensional structure, but to also determine the secondary and tertiary structure against other proteins.
- Protein classification - classifying unknown proteins into existing families, or finding families that are similar in terms of both sequence and structure
- Gene expression - determining how genes are expressed in a genome and the effect they have on other genes.

Trying to understand how the genotype (genetic code) of a person affects their phenotype (the observable features of an organism) is a non-trivial problem. Gene mutations have to be expressed within the genome, usually meaning mutations should occur in highly conserved regions of an organisms genome passed down from the organism's ancestors. Tools like *MAPPFinder* [15], *Meta Gene Profiler (MetaGP)* [21] and *SeqExpress* [7] specialize in analyzing how genes interpret information in the organisms DNA and then how that information is used to create a phenotypic output, usually via mRNA or amino acid processing. Gene maps are built from the annotation data and show how they interact with eachother.

The effects of gene mutations or Single Nucleotide Polymorphisms (SNP) are extremely important, and fundamentally responsible for novel genetic disease mechanisms. It is usual that mutations occurring in genes that are found in highly conserved regions of DNA passed down by ancestors as these regions of DNA code for vital processes that make up an organism. Programs such as the Polymorphism Phenotyping tool, PolyPhen [1] and the Sorting Intolerant From Tolerant program, SIFT [34] predicts whether amino acid



`http://www.genmapp.org/helpv2/GenMAPP.htm`

Figure 1.3: Gene maps are convenient visualization tools for biologists and geneticists. They an be created in a variety of ways include ma- chine learning methods such as support vector machines and neural net- works. Training data will typically include gene labels and phenotypes associated with them, and then supervised (class prediction) or unsu- pervised (class discovery) can be carried out on new genes.

substitutions effect protein function. The general idea is that mutations are scored based on the position they occur at in a protein, as well as the type of amino acid substitution. The mutated sequence is then queried against similar proteins and uses the amino acids found in nature at the same position of the mutation to determine if the particular amino acid mutation is tolerated.

Algorithms such as SIFT and PolyPhen-2 rely on the amount of homologous sequences in the database to compare submitted amino acid sequences to. This is because if sequences in the database exist that are closely related to the test sequence then the algorithm will have a higher confidence

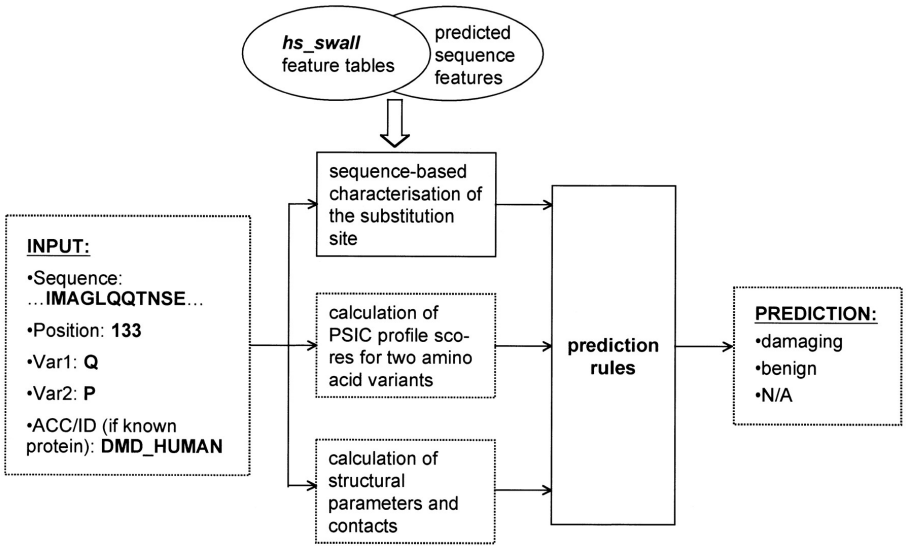


Figure 1.4: A generic PolyPhen pipeline process for determining the effects of amino acid substitutions in proteins. SNPs are inputted into the pipeline where prediction layers include variables such as secondary structure of sim- ilar amino acid sequences, known binding sites, molecular weight are determined to make a prediction on how the SNP alters the function of the protein. Source - `http://nar.oxfordjournals.org/content/30/17/3894/F1.expansion`

in the prediction. It also means that if there are many sequences that are similar, the test sequence would be part of a highly conserved amino acid sequence and therefore be more likely to predict functional change to the protein. This may lead to high rates of false positives, where SIFT and PolyPhen-2 will provide a measure for expecting false positives in the analysis by calculating the diversity of multiple sequences alignments when the test sequence is aligned to sequences in the database. A derived variable, *the median consveration value* is calculated from the diversity in the alignment. Predictions with a higher conservation value are more likely to yield false positives in the prediction result. Accordingly, thresholds for this measure can be imposed when deciding to trust the prediction. Classifying structural and functional features about proteins is a long standing tradition in biology, a tradition which has been kept and propelled forward in bioinformatics.

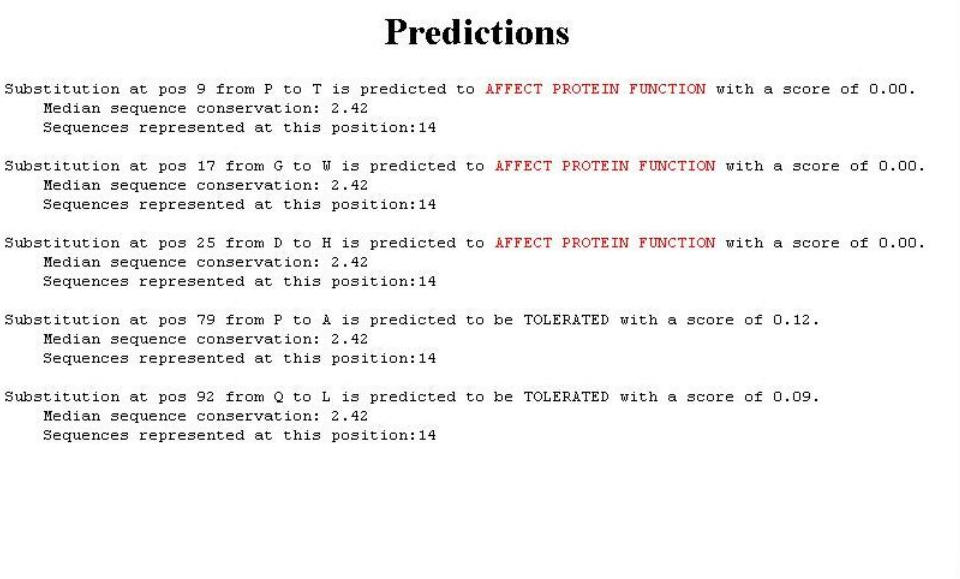


Figure 1.5: The SIFT format for tolerance predictions of amino acid substitutions. Substitutions with a score of less than 0.05 are predicted to affect protein function. Above is the result of a protein entered into the SIFT web application found at `http://sift.bii.a-star.edu.sg/SIFT.html`

The ultimate aim of bioinformatics in protein classification is to build on the manual classification by experts, using the experience that has been gained in terms of methodology and extend the scope of analysis via machine learning techniques to assist the experts. Structural classification of proteins derive primary, secondary and tertiary structures based on molecules that are built from the amino acid sequences. Primary structure classifications determine familial relationships between proteins by searching for similar homologues through various algorithms. One of the most widely used primary structure databases, *Protein Families database (PFAM)* [4] stores protein families and superfamilies. All of families are derived by using seed sequences of amino acids which are suspected to belong to the same family through sequence homology, and *Hidden Markov Models* build upon the prominent features found from the sequence alignments to further the classification process. The properties of HMM's make them useful not just for classifying protein families, but searching against them and learning what features determine what puts proteins into families other than sequence homology, and as such a whole chapter of this thesis id dedicated to using HMM's for protein classification. Secondary structure classification aims to predict the three-dimensional structure of local segments of amino acids. Many bioinformatics tools use only the raw amino acid sequences to predict where *helixes, coils* and *turns* occur in the corresponding protein. Before the data explosion of bioinformatics, secondary structure was predicted by experts using chemical properties and statistics of single sequences [11], but as multiple sequences per family have been made available, more sophisticated tools have been developed that automate the methodology. Prediction methods described in [31] and implemented by the *PSIPRED protein structure prediction server* [23] use neural networks and multiple sequence alignments to predict secondary structure.

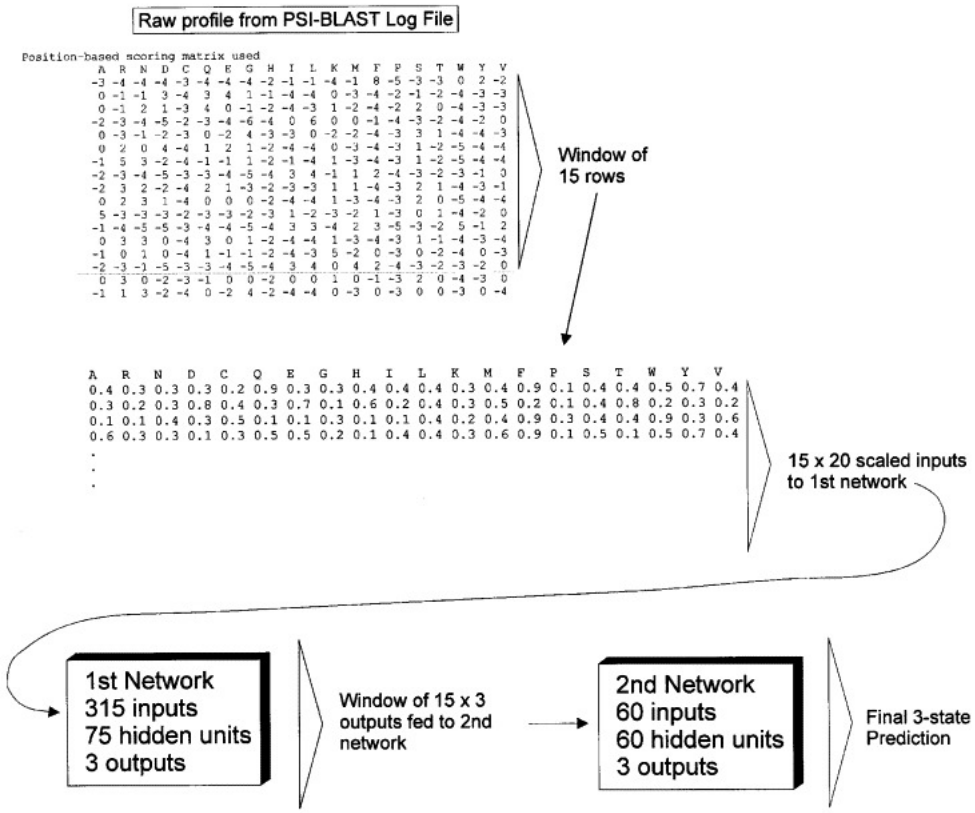


Figure 1.6: An outline of the PSIPRED method. The Position-Specific Iterative BLAST tool is used to create a non-redundant multiple sequence alignment database of 340,000 proteins. A scoring matrix of log-likelihoods for amino acids substitutions is defined from the multiple sequence alignment and used as input to the neural network

Tools such as PSIPRED allows users to submit primary amino acid sequence data and determine the secondary structure, classified into helixes,coils, turns and other features. In this thesis PSIPRED is used to determine secondary structure of transmembrane proteins, to be used as training features in random forest for primary protein classification. `http://bioinf.cs.ucl.ac.uk/psipred/`

1.4 Protein classification and machine learning

The scope of challenges and problems in bioinformatics are vast, and an overwhelming majority of the tools produced to answer them are built upon machine learning methods. Machine learning came directly from statistical model fitting, aiming to mine useful information from a wealth of data, and implementing them was a natural progression to fit the data explosion of genome sequencing. Not only are machine learning techniques a perfect choice to analyze large amounts of data, they can be used to build models that learn by observed examples and once trained, new data can be fed into the model and new predictions can be made. The aim is to build classifiers that describe complex, and often unknown relationships formed by millions of years of evolutionary changes in organisms. There are not enough biologists in the world that could churn out findings that keep up with the amount of data becoming available, so machine learning techniques provide a hypothesis into possible features of DNA/proteomic data that can be used to guide further experiments, either refuting or proving the hypothesis. Even if there were enough biologists in the world to keep up with the data, machine learning may still have one advantage over the experts, and that is being able to extract relationships that would not have been thought of by the experts - they can be purely data-driven as well as being used as an extension of the deductive powers of biologists. Typically, bioinformatics machine learning techniques can be split into three categories:

- *Supervised Learning* - the classifier has been given prior knowledge of the data, for example class labels.
- *Semi-supervised Learning* - only a small amount of information has been given to the classifier prior to learning
- *Unsupervised Learning* - no information has been given to the classifier

prior to learning

This thesis uses two examples of supervised learning so an overview of supervised learning follows.

1.5 Supervised learning

Supervised classification makes use of training data that has been divided into classes, labelled by perhaps protein family name. It is important in supervised learning that the labelled dataset is accurate. Features are provided along with the labels - these may be statistics about the class, or in the case of proteins they may be the sequences themselves. The point is that the features are usually chosen in advance such that the classifier will gain as much information as possible from the training data. Define a *class label* as $C \in \{0, 1\}$ and a feature vector as $X \in R$ where features in X is used to C. This classifier is expected to accurately sort data into classes based on any set of real feature vectors and generalize away from the training data so that unseen data may be classified into the class labels given in the training data. To produce reliable classification, the following steps should be taken:

- Select appropriate training data. This usually means use features that can be found in real life data, or some statistics about the data such as frequency of certain amino acids. Derived data such as secondary protein structure may also be used. The class labels also need to be accurate and the account for enough diversity within that class to avoid bias towards certain features.

- Select an appropriate learning algorithm. If the training data consists entirely of derived features or statistics of raw data, it might be appropriate to use algorithms that use decision based learning such as neural networks of decision trees. We will see later that alignments of raw amino acid sequences are best suited to hidden Markov models.

- Ensure that the predictions can be evaluated by use of a test dataset. Test datasets are useful to determine the accuracy of the classifier when adding new parameters to the training data.

Choosing which algorithm to use is a particularly interesting are of research. It is certain that one algorithm will not be the best classifier in all different types of training data. Algorithms of similar architecture are often compared, but ultimately the algorithm should be chosen to suit the data.

1.6 Popular supervised learning techniques

A variety of different techniques have been used in bioinformatics, and particularly in protein classification. They include Markov models, support vector machine, neural networks and decision trees.

Artificial Neural Networks: is a technique proposed by *Widrow et al* [40] in the 1950's, inspired by the central nervous system of animals. The proposed model was to mirror neurons being a basic information processing unit that feed into a central processing system. Each "neuron" measures certain properties, fire based on incoming information and send an output to the central processing unit. The state of the neuron and output can be represented mathematically, and as such each neuron is weighted, determining the effect of the incoming information. Neurons may be weighted positively (excitatory) or negatively (inhibitory) and the overall activation value for unit i at time t is given by

$$\eta_i(t) = \sum_j \omega_{ij} x_j + \beta_i \tag{1.1}$$

where w_j is the connection strength from unit i to j. The value unit bias is given by $_i$ and x_j is the output value of unit j. The neural network is set up such that the output of any unit directly effects that of another unit by either strengthening or weakening it's signal. The output of the neural

network is determined by the connections of a unit and represented by a non-linear function

$$\gamma_i(t) = \frac{1}{1 + e^{\frac{-\eta_i(t)}{T}}} \tag{1.2}$$

Wu *et al* [50] demonstrated an artificial neural network system, Protein Classification Artificial Neural System (ProCANS) in 1992 to classify 4 protein functional groups found in the Protein Information Resource (PIR). The architecture of the N-N consisted of a three layer feed-forward network; an input an output units, as well as a hidden layer to store a neural database of sequence meta-features, where a back propagation learning technique to adjust unit weighting and a feed-forward algorithm calculates the output of each unit. This produces a neural database for each protein superfamily that can be quereyed, albeit more effectively than a simple raw sequence database query and exploits the meta-features stored in the networks' hidden layer. Using 1656 training proteins and 492 test proteins, electron transfer proteins, tranferases, hydrolases, and lyases were classified to 90% accuracy. The ease of use, high accuracy and short computation time made N-N's popular for protein classification and has been used extensively with good success, in particular the classification of membrane and non-membrane proteins by [35] Pasquier *et al* and achieving classification accuracy of 97% by using only 11 proteins to train their neural network. This experiment provided a basis for training material in this thesis, where transmembrane proteins are classified between non-transmembrane proteins and sub-families within the superfamily.

Support Vector Machines introduced by *Vapnik et al* [12] is a supervised learning technique that takes data in some input space and constructs hyperplanes to split up the input space into classes. Define training points in input space as

$$D = \left\{(x_i, y_i)|x_i \in R^P, y_i \in \{-1, 1\}\right\} \tag{1.3}$$

where y_i indicates which class training point x_i belongs to. The SVM algorithm is designed to construct a hyper-plane that splits training points belonging to $y = 1$ from those belonging to $y = -1$. Such a hyper-plane can be computed as a hyper-plane which satisfies the equation

$$wx + \beta = 0 \tag{1.4}$$

where w is the normal vector to the hyper-plane and β is the intercept. There may be more hyper-planes if there can be more linear separations made on the data. The solution above assumes linear separation of the data, however if linear separation is not possible via a linear hyper plane, the solution may be obtained by forcing the hyper-plane to be non-linear via some transformation. The main idea behind SVM's is therefore to split a set of data into smaller sets by finding the largest distance between the data when plotted in kernel space.

Complex data sets can be split up in this way by means of a kernel trick to produce a non-linear classifier, merely by embedding the data into a different kernel space. The parameters in SVMs are few, and revolve around choosing which kernel to use e.g. a radial basis kernel, and yet despite such simple parameters and seemingly simple mathematical formulation SVM's produce an elegant solution to data mining for both classification and regression. Unlike artificial neural networks and other generative machine learning techniques, SVM's do not require to build a model for each class of data one are interested in, but allows a general classification process split across negative and

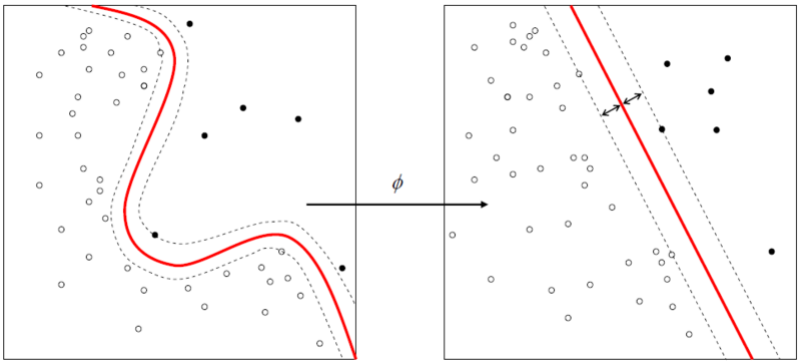


Figure 1.7: non-linear (left) and linear (right) support vector machine solutions. Non-linear solutions solve problems where the data is not linearly separable, in which a kernel trick is used that maps the data into a different space through an inner product, where linear separation is possible. Source `http://en.wikipedia.org/wiki/File:KernelMachine.png`

positive training data sets and a choice of high level feature vectors to be used as classification criteria over any training set. This flexibility yields a simple algorithm that allows the user to choose their own high level features to classify their data, rather than having an algorithm determine features in a black box approach. It means that the classification results can be described in terms of the high level features, rather than meta-features.

Many specific problems are solved by producing bespoke kernels, and protein classification is no exception. Leslie *et al* [29] developed the Spectrum Kernel used for discriminating between positive and negative training sets. High level vector features, such as hydrophobicity are embedded into co-ordinates in vector space in which the Spectrum Kernel produces a linear decision boundary in this vector space. The kernel is a string kernel in input space X of alphabet A where subsequences a of length k are observed in training and test sets. The feature map for the spectrum kernel is defined as

$$\Phi_k(x) = (\phi_a(x))_{a \in A^k} \tag{1.5}$$

where $\phi_a(x)$ is the amount of times a occurs in sequence x. The Spectrum

Kernel can then be defined as

$$K_k(x, y) = \langle \Phi_k(x), \Phi_k(y) \rangle \tag{1.6}$$

Subsequences of length k are collected by moving a sliding window across each of x and y. $k = 3$ was found to be the optimal subsequence length for training the spectrum kernel, where the authors report similar accuracy to psi-BLAST and the SAM-T98 HMM framework over 36 protein families from the SCOP database.

Naive Bayes Classifier is a simple probabilistic classifier based on *Bayes Theorem*. It assumes that the presence/absence of a feature in the data is independent of any other features. For example one might imagine two protein families containing a certain amino acid, but found more frequently in one family than the other. The probability model for a Naive Bayes classifier is given by

$$P(C|F_1, F_2, ..., F_n) \tag{1.7}$$

That is given the value of a feature F_i, what is the likelihood of belonging to class C_i? The training data fed into a naive Bayes classifier holds prior information of which feature values belong to each class. For example, imagine protein families A and B, where the amount of times amino acid C is observed in family A is 10/100, and 50/100 in family B. Clearly a new unclassified protein that contains amino acid C is more likely to belong to family B based on the prior knowledge of the training data, and the probability of belonging to a certain class is calculated as

$$P(C|F_1, F_2) = \frac{P(C)P(F_1, F_2|C)}{P(F_1, F_2|C)} \tag{1.8}$$

which translates as

$$probability = \frac{prior * likelihood}{observation} \tag{1.9}$$

Naive Bayes is a relatively simple generative classifier, but has been found to outperform other machine learning techniques. It's major advantage is that it doesn't require a lot of training data to learn the parameters for classification. Although Naive Bayes classifiers are not as widely used as N-N's and SVM's, there have been some successes in the literature, where Feng *et al* [20] use NB to classify phage virion proteins up to 79.5% accuracy, outperforming random forests

This thesis aims to demonstrate primary protein classification by implementing two other machine learning techniques - random forests and hidden Markov models. Their approach was to make a Naive Bayes prediction on each single feature vector expansion, where the feature with the highest accuracy is kept and a separate feature is added to it, and the highest accuracy of this feature subset is kept, and the process is iterated until all features have been added.

1.7 Summary

Machine learning has offered tools that can harness the knowledge of biology and make protein classification, as well as other bioinformatics problems solved accurately, quickly and effectively. For protein classification, many algorithms have a strong focus on learning the difference between classes and discriminating further test proteins from each other from using this learned knowledge. SVM's, Neural Networks and Decision Trees are all great examples of these and have been popular before next generation sequencing. This is mainly because such algorithms are designed to be flexible in terms of data input and can use meta data extracted from the sequences and tailored to the type of experiment to be carried out, however the black box approach to some of these algorithms may cause apprehension to researchers that do not understand the method being used. Other than discriminative processes, other more bespoke algorithms have also been designed, in particular alignment based algorithms that use raw sequence data and align proteins based on sequence position to obtain a direct comparison of proteins. The BLAST alignment search tool does this by comparing substrings of two sequences and scores them based on matches and indels (deletions or insertions) in any of the sequences. Later, a profile Hidden Markov Model is presented which makes use of multiple sequence alignments to learn amino acid emissions and transitions in certain protein families and scoring a test protein against the model is done by aligning the protein to the model and traversing through each position in the protein, noting how its own emissions and transitions compare to that of the family in question. Overall, machine learning offers something more than just an accurate classification tool, it offers classification that would not be possible by analyzing with the human eye, at least not on the scale needed. It is also important to understand, as this thesis aims to point out, that although one particular machine learning algorithm works well over lots of different applications and data sets, there may be another machine learning technique better suited to a certain problem.

Chapter 2

Protein classification using randomized decision trees

A decision tree is a directed graph comprised of levels (questions) where by with each question asked, multiple routes can be taken, of which one will be chosen depending on the answer. Each question asked (test) is represented by a node, and depending on the parameters of the test, the data sent through the node will be split and sent to further test nodes. Special nodes called a leaf nodes store the final answers to each test, and no further splits occur when the data has reached the respective leaf nodes. There is also a root node at the start of a tree, and is the only node where the entire data sample goes through. Collectively a tree is a collection of nodes, edges between nodes, leaf nodes and a root node.

Random Forest is an ensemble machine learning technique which builds decision trees at training time to output classes within the training set based on splitting the data at each node by a threshold. For each tree in the forest, the tree is trained and tested using bootstrapped samples (with replacement) of the dataset where the test data is referred to as "Out of Bag" data that is used estimate an OOB error. This OOB error tests the accuracy of the trees and the estimate is proven to be as accurate as using a test set which

is of equal size to the training set [9]. The classification accuracy the forest of tree is determined by summing the classification of each individual tree. A tree in the Random Forest is trained on a bootstrapped training subset by replacement from the original training data. Training data not used to train a tree is referred to as "*Out-Of-Bag*" (OOB) samples. Every tree in the random forest is both trained and tested independently from all other trees, allowing for training and testing to be carried out in parallel.

The Random Forest algorithm may be used for both regression and classification, and since this thesis describes methods that classify proteins into families via a machine learning approach, the latter will be discussed in detail. The aim of classification is to sort input data , v into discrete classes $c \in \{c_k\}$. The random forest is trained on some data, and with good generalization unseen data can be classified based on the initial training.

2.1 Randomized decision tree theory

Define data $v = x_1, x_2, ..., x_d$ where v is a dataset, such as proteins, and it's components x_d are some features about the dataset e.g. molecular weight or percentage of a certain amino acid. For a forest of decision trees, it is not necessary to load all features x_d per tree, in fact it is advantageous to randomly select and permute a sub-selection of features to be used in each tree to generalize the data.

The aim of training a decision tree is to optimize the parameters of nodes in the tree that contain each test or filter, which will in turn optimize the leaf nodes and generate accurate prediction. Denote training subsets (starting at the root node level 0) of training nodes by their node splits $S_1, S_2, ..., S_n$ in breadth-first order, where S_1^L, S_1^R are the child subsets splitting left and right from a parent node. The subset of any split node S_j in a tree may be defined

as

$$S_j = S_j^L \cup S_j^R, S_j^L \cap S_j^L = \emptyset, S_j^L = S_{2j+2}, S_j^R = S_{2j+2} \tag{2.1}$$

A function $\phi(v)$ selects random subsets of features from x_d to be used in each tree, where each split node S_j contains a test that evaluates feature x_i. To optimize the parameters, it is necessary to gain as much information about the sampled training data $S(v)$ as possible at each split node.

At each node split we want to maximize the amount of information gain, and to do this the data must be split such that the entropy of the data decreases. Entropy is a measure of disorder in a system, where lower entropy equals more disorder in the system. The information gain at each split node can be computed as

$$I = H(S) - \sum_{i \in 1,2} \frac{|S^i|}{|S|} H(S^i) \tag{2.2}$$

where $H(S)$ is the Shannon entropy and defined as

$$H(S) = -\sum_{c \in C} log(p(c)) \tag{2.3}$$

Where p is the probability of a character c appearing from a set of characters C. One can measure the amount of information gained directly by measuring the Shannon entropy, where the lower the entropy after subsequent splitting of the data, the more information gained.

The random forest is comprised of multiple trained decision trees, where each tree is trained independently with random sampling of training data without replacement. Each trained decision tree can be considered a "weak learner" that gains information from a bootstrapped sample of the training data v. Each leaf node in each tree are distinct predictors determined by the test at each node leading to the leaf node, and therefore store information that are used to classify test data. It is also important for the optimization of

the forest that the sample data fed into each tree, and therefore the predictors at the leaf nodes are determined randomly.

Since each split node j is a boolean test, each node split can be define as a binary function

$$h(v, \theta_j) \in \{0, 1\} \tag{2.4}$$

with 0 and 1 being the results of the boolean test. The data is then split into left and right child nodes based on where they exist in the binary set with the j^{th} feature having a probability of $p_{nj} \in (0, 1)$ of being selected. During training, data is split according to some threshold of the parameters held at node j to maximize the information gain I_j. Leaf nodes store the predictor data for a certain class, containing information such as which classes within the data are stored in the leaf node and the feature parameters that was used at nodes along its path through the tree. During testing the data is split at each node j according to some weak learner $h(v, j)$, defined as where ψ is a filter function that selects features from the entire dataset v, ϕ primitively separates the data at each node e.g. linear regression of the data, and τ stores the thresholds used in each binary test. A good weak learner will have optimized parameters, and optimal training parameters θ_j^*,at the j^{th} split node, must be computed by determining which training parameters split the data ad provide the most information possible at that node. To do this, the information gain is maximized as

$$\theta_j^* = \max \theta_j I_j \tag{2.5}$$

$$I_j = I(S_j, S_j^L, S_j^R, \theta_j) \tag{2.6}$$

S_j is the node before the split, and S_j^L and S_j^R are the child nodes after the split. Along with randomized sampling of the data, a random forest can be implemented by random optimization of the weak learner parameters.

There is also the question of how many split nodes are needed to gain adequate information, and when to stop tree growth to avoid over-fitting - which is referred to as tree pruning. Two common approaches that are common is to either limit the length of each tree by a maximum amount of levels,D, or to impose maximum information gain for each tree.

Generalization of the forest ensures higher accuracy of classifation over the entire dataset v. There little use in optimizing classification based on only a few features of the dataset, or even the strongest features as inevitably some test data fed into the forest will not exhibit certain features, or introduce noise into the forest which must be dealt with. Randomizing the way in which individual decision trees in the forest are trained ensures generalization by providing many permutations of predictions trees that are combined to provide an overview of the features that are important when splitting classes from a large superset of samples. In short, randomness is imposed in two ways: by randomizing the samples of v chosen to train the trees, and to randomize the way in which the weak learners θ are optimized, where both can be used at the same time. Randomizing the data sampling is trivial, so we will focus on randomized weak learner optimization.

If v is that entire dataset, and Π are all the parameters to be used for training, for the j^{th} node a subset of $\Pi_j \subset \Pi$ are made available for each node to choose from. The randomness is introduced by controlling $|\Pi_j \Pi$ and the node is optimized by

$$\theta_j^* = \arg\max_{\theta_j \in \Pi_j} I_j \tag{2.7}$$

where the optimal parameters θ_j^* for a given subset of parameters are those that maximize the information gain. The information gain will be at a maximum when the Shannon entropy is at its lowest. By optimizing the parameter choice at each node, the information passed to the leaf node serves as the best predictor possible given the parameters chosen in Π. When each tree is trained independently of each other, we have an ensemble of trees that all

carry their own predictions (leaf nodes) and these must be tested to determine which tree gives the best result for test data v.

A forest of containing weak learning trees $t \in \{1, ..., T\}$ each have predictor leaf nodes $p_t(c|v)$ and we are interested in combining the result of all predictors as a means of testing the forest. Testing involves v going through each tree, in parallel to increase efficiency, and combining the predictors by say an average over all predictors

$$p(c|v) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|v) \tag{2.8}$$

where classification given test data v is achieved by averaging each predictor p_t and normalizing over the number of trees T. The nature of the training process will produce trees that have gained more information (or decreased entropy) than others. The distribution of results from averaging over all trees will be representative of those trees that have gained more information than others, therefore the results are influenced by those trees that are the strongest weak learners.

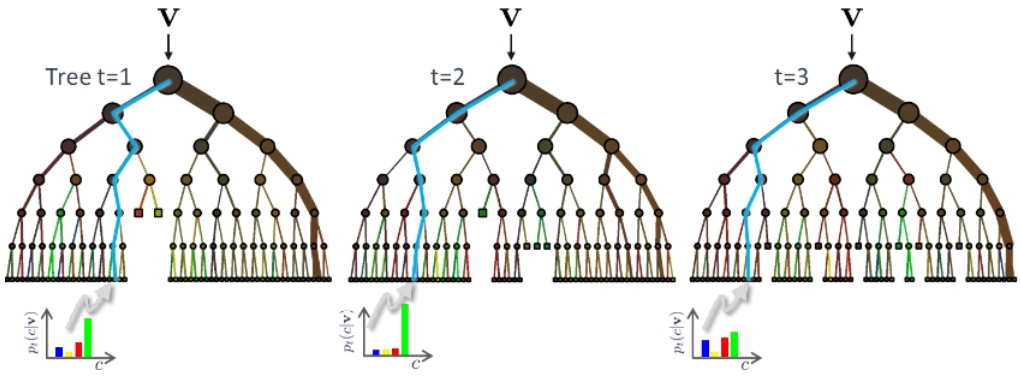


Figure 2.1: All test data moving through each tree reaches a leaf node based on the optimization of the parameters held at each node. The result of the leaf node is a weak learner in which the average of all trees are taken as the classification prediction [13]

The other advantage of doing this is to account for "noisy" trees, that may otherwise effect the result if selection was based solely on strongest weak learners. This is not the only ensemble method however,there are other ensemble methods that can be used such as user input to "hard" select certain trees.

2.2 Feature importance in random forest

The amount of data biologists are faced with, often with no prior knowledge of how to classify a protein or gene has lead to the need of machine learning techniques that not only classify data, but quantify how the data was classified for feature selection techniques [41]. Random Forest does both data classifation via feature extraction and determining which features best classified the data i.e. random forest performs feature selection in parallel to building classification rules. This is useful to biologists as more often than not, it is more important to know why a protein belongs to a certain family, or why a gene expresses a certain phenotype that to actually know why gene expresses that phenotype, or which family that protein belongs to. The measure used in this work to determine which features best split the data is the Gini index. The Gini index is essentially measured by calculation the level of impurity of the data at each node split found within the child nodes. At each node j, the impurity, or "Gini impurity" $G(j)$ is defined as:

$$G(j) = 1 - p_1^2 - p_2^2 \tag{2.9}$$

where $p_k = n_k/n$ is the fraction of n_k samples from class $k = 0, 1$ from n samples at the node j. The change in the Gini impurity as the data is split into two child nodes is

$$\delta G(j) = G(j) - p_L G(j_L) - p_R G(j_R) \tag{2.10}$$

where p_L and p_R are the respective sample fractions held in the child nodes. At each node, an exhaustive search over features and thresholds yields a pair Φ, τ that represents the maximum value of $G(j)$ that decreases with each node split, and for each node j in each tree T the Gini importance is the sum of all pairs yielding maximum $G(j)$

$$I_G(\Phi) = \sum_T \sum_j \delta G(j)_\Phi(j, T) \tag{2.11}$$

$I_G(\Phi)$ is a measure of how often feature Φ was used to split a node. If the Gini index decreases at each node, then clearly the larger the Gini importance for Φ, the more important that feature is in classifying the data.

At least for the findings in this work, feature importance is perhaps more interesting than simply comparing different algorithms to see which classifies best. That is just the first step, and the second is to ask why they belong to that family. It has to be remembered however that any classification and feature extraction of random forests are only relative to the training sets used - if one suspects a protein belongs to a family then the positive dataset is not so much of a problem, but the negative dataset is critical as one are discriminating against it. One would want to be in a position to say that the proteins are classified by features that are distinctly different from those in the negative training set, and so negative training sets should be large and randomly selected to represent background frequencies of any functional groups.

2.3 Summary of algorithm and parameters

The random forest algorithm contains multiple parameters that directly effect the ability to accurately carry out regression/classification by using an ensemble of decision trees, those being:

Parameter 1: Test/training data ,v, to be fed into the tree

Parameter 2: Forest size T

Parameter 3: Individual tree depth D

Parameter 4: The randomness parameter Π that controls the parameters that optimize each split node in a tree

Parameter 5: The data selection criteria ,θ, that determines how each weak learner is built

Decisions on how to set up these parameters will of course depend on the trade offs between accuracy and efficiency. Some of the literature on random forests show that accuracy increases with forest size [14],[43],[51], and some work points out that as the depth of a tree increases, over-fitting reduces the power of weak learners. Although using large amounts of training data may slow computation time, large training data can help inhibit problems such as over-fitting [44]. Breiman's random forest algorithm has shown the importance of introducing and optimizing randomness in the ensemble when acheiveing correlation and generalization of the predictors. The randomness parameter, Π, will be discussed later to show the effect of randomness on tree correlation. In general the algorithm is quick, with complexity $O(nm)$ where m is the tree depth and n are the amount of trees in the ensemble.

2.4 Protein classification using random forest

Random Forest as a classifier have shown accurate results in protein classification in the literature, although not nearly as widely used as popular techniques such as Support Vector Machines (SVMs) and Artificial Neural Networks (N-Ns). Most work using RF's has been done in protein-protein interaction and binding site studies, where proteins that bind together produce a functional output. RF used in this way look at the functional similarities of the binding proteins by looking at the tree intersections produced from

functional features [10] [38]. However, as biological data grows there is a need to implement such techniques to learn what proteins belong to certain families and the reasons for doing so. Kandaswamy *et al.* [24] demonstrated RF to be a successful classifier for antifreeze proteins when using non-antifreeze proteins as a negative test set, achieving 84% accuracy, which was better than other methods used such as HMMs, SVM's and N-N's in their study. Clearly RF can be used and should be explored for other protein families to be established as a tool for future classification when new proteins are found. Another useful feature from the RF algorithm that has been explored in the literature is feature importance using measures such as Gini importance and permutative importance. Feature importance is an integral part of protein-protein interaction studies as it explains the relationships between protein bonds, and as this experiment shows, the Gini importance picks out features known in the literature to be essential features as part of transmembrane and antifreeze proteins that are used to split the proteins best over a range of other features.

This experiment uses the RF algorithm to classify two different types of protein families: ion channel transmembrane proteins from non transmembrane proteins and antifreeze from antifreeze-like proteins. Transmembrane proteins exist within the membranes of cells that transmembrane molecules and ions across the membrane to inside the cell. They are generally tightly packed with polar side groups on the outside to enhance their solubility in water, with non polar side groups folded to the inside to keep water from getting in and unfolding them. Transmembrane proteins show high structural homology across the family. In contrast, antifreeze proteins do not show high structural homology, specifically the type III antifreeze family. The type II clan consists of two sub groups - one being antifreeze and similar proteins such as flagellar and pilus proteins that provide a similar functional role, and the second being homologous proteins in terms of function, which for ease of use will be referred to as antifreeze and antifreeze-like proteins respectively.

Protein features	
Feature set	No. of features
Frequency of Amino acids	20
Frequency of functional groups	17
Physiochemical properties	6
Helix,strand and coil regions	60
Total	**93**

This particular family of antifreeze proteins have been chosen in contrast to the transmembrane proteins where structure is rigid and recognizable - antifreeze proteins do not have such well defined structure because they have convergantly evolved from various different types of organisms [32] and as such the high variance in structure of the each families' subtype constituents will be a good test for classification.

The transmembrane training set consisted of 337 ion transmembrane proteins and non-transmembrane proteins. Ion transmembrane proteins (positive dataset) were obtained from the PFAM database (PF00520) [46], while the non-transmembrane proteins were obtained from a random selection of non-transmembrane proteins in PFAM. The final dataset consisted of 297 transmembrane proteins (positive dataset) and 297 non-transmembrane proteins (negative dataset), in which 40 of each dataset were randomly as a test dataset, and the remaining 594 proteins used in the training dataset. A training set of 100 antifreeze proteins and 100 antifreeze-like proteins where taken from the PFAM database (CL0489), and a test set of 26 antifreeze proteins and 26 antifreeze-like proteins were used to validate the model. The feature vectors used were as follows

Frequency of amino acids: Frequencies of the 20 naturally occurring amino acids. *Frequency of functional groups:* 20 amino acids were categorized into 17 functional groups based on the presence of side chain chemical groups such as phenyl (F/W/Y), carboxyl (D/E), imidazole (H), primary amine (K), guanidino (R), thiol (C), sulfur (M), amido (Q/N), hydroxyl

(S/T) and non-polar (A/G/I/L/V/P) [37] The frequency of 17 functional groups (number of occurrences of functional group "X" divided by length of the protein) was computed for each sequence. *Physiochemical properties:* Physiochemical properties derived from AAINDEX database is used to compute the following properties: isoelectric point, aromaticity, grand average of hydropathicity index (GRAVY), secondary structure fraction, molecular weight and instability index [25]. For each sequence, physico-chemical property value was calculated as the sum of physico-chemical property value for all residues of the sequence, divided by the length of the sequence. These features were extracted using the biopython ProtParam module. *Helix,strand and coil regions:* SSE (helix: H, beta sheet: E and coil: C) information was assigned to all the sequences in the alignment using a secondary structure prediction program, PSIPRED [23]. The frequency of the 20 amino acids residing on helix(H), beta sheet(E) and coil (C) were calculated. With the test set of 40 a-type transmembrane proteins and 40 non-transport proteins, a baseline accuracy of 62.5% was achieved using only amino acid composition alone. Main causes for misclassification arose from false positives (22.5%) in classifying non-transmembrane proteins as transmembrane proteins. With the test set of 26 antifreeze proteins and 26 antifreeze-like proteins, a baseline accuracy of 86.9% was achieved using only amino acid composition alone. The accuracy for both tests increased marginally when more features were added into the training data set.

Threshold dependant parameters were used to evaluate the classification accuracy of the Random Forest algorithm, specifically sensitivity and specificity parameters defined as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.12}$$

which calculates the percentage of correctly predicted proteins from the pos-

itive test set, were TP are the true positives and FN are the false negatives

$$Specificity = \frac{TN}{TN + FP} \tag{2.13}$$

which calculates the percentage of correctly predicted proteins from the negative test set, where TN are the true negatives and FP are the false positives. Classification accuracy is then determined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

Cumulative classification accuracy		
Features	Transmembrane	Antifreeze
Frequencies of amino acids	62.5%	86.9%
Physiochemical frequencies	72.5%	89.1%
Frequency of functional groups	75%	89.1%
Secondary structure frequencies	85%	91.2%

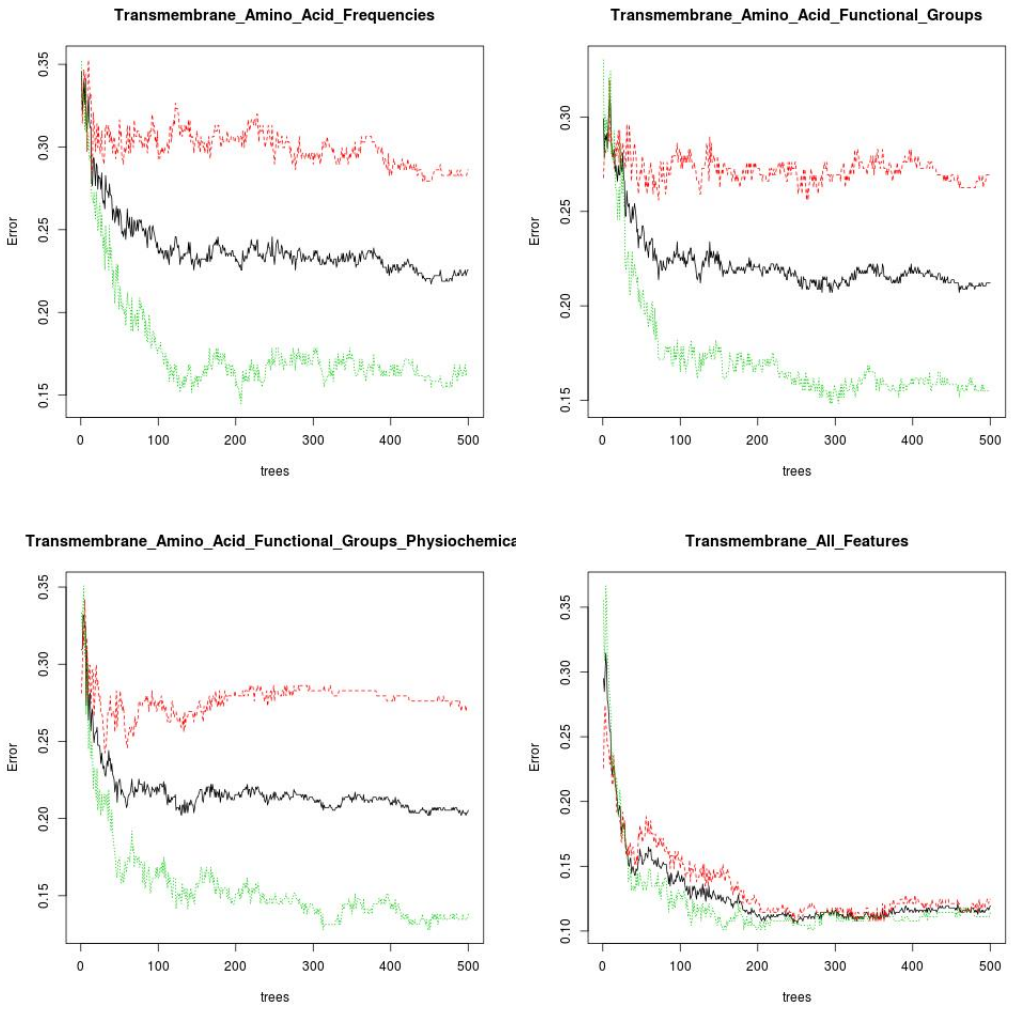


Figure 2.2: A plot of the random forest error for antifreeze proteins split cumulatively over the 4 sets of features. The top left graph is the error when using amino acid frequencies only, the top right is error when amino acid frequencies and physiochemical properties etc. The black line is the "Out of bag" error which represents the error of proteins used outside of the training process from both classes, and as such the OOB error is an average of both class errors. The green line denoted positive training datasets, where red are negative training sets.

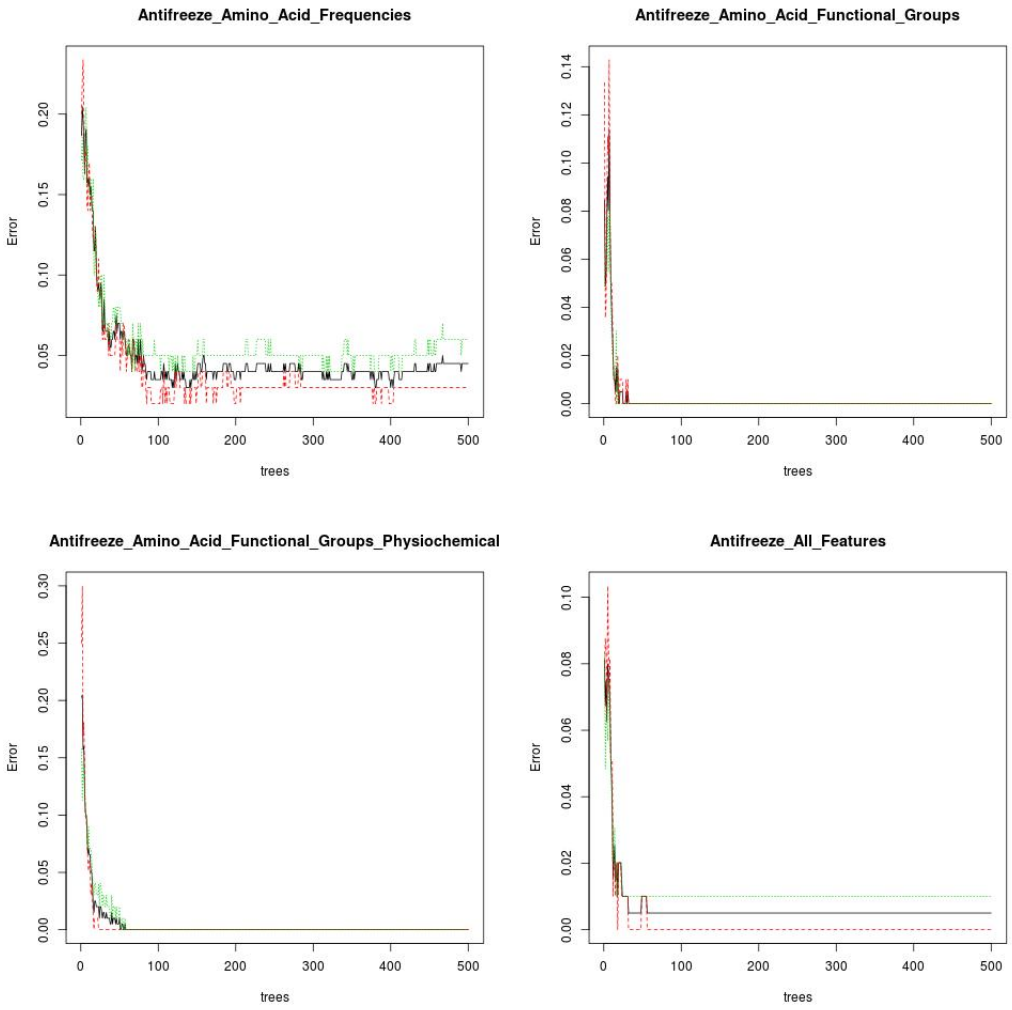


Figure 2.3: A plot of the random forest error for transmembrane proteins split cumulatively over the 4 sets of features. The top left graph is the error when using amino acid frequencies only, the top right is error when amino acid frequencies and physiochemical properties etc. The middle line is the "Out of bag" error which represents the error of proteins used outside of the training process from both classes, and as such the OOB error is an average of both class errors.

The measure used in this work to determine which features best split the

data is the Gini index. The Gini index is essentially measured by calculation the level of impurity of the data at each node split found within the child nodes. At each node j, the impurity, or "Gini impurity" $G(j)$ is defined as:

$$G(j) = 1 - p_1^2 - p_2^2 \tag{2.15}$$

where $p_k = n_k/n$ is the fraction of n_k samples from class $k = 0, 1$ from n samples at the node j. The change in the Gini impurity as the data is split into two child nodes is

$$\delta G(j) = G(j) - p_L G(j_L) - p_R G(j_R) \tag{2.16}$$

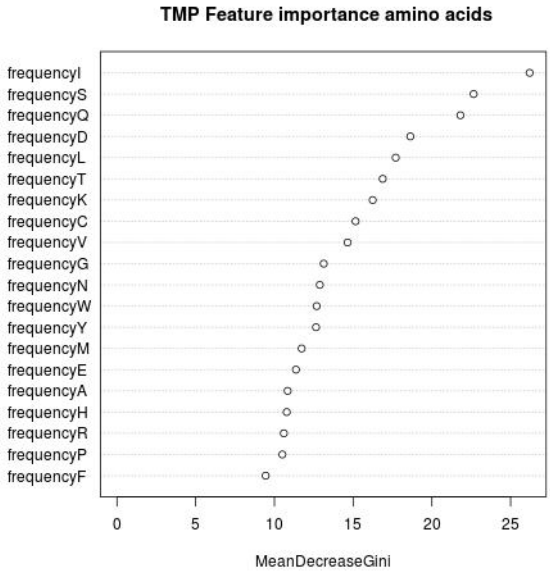
where p_L and p_R are the respective sample fractions held in the child nodes. At each node, an exhaustive search over features and thresholds yields a pair Φ, τ that represents the maximum value of $G(j)$ that decreases with each node split, and for each node j in each tree T the Gini importance is the sum of all pairs yielding maximum $G(j)$

$$I_G(\Phi) = \sum_T \sum_j \delta G(j)_\Phi(j, T) \tag{2.17}$$

$I_G(\Phi)$ is a measure of how often feature Φ was used to split a node. If the Gini index decreases at each node, then clearly the larger the Gini importance for Φ, the more important that feature is in classifying the data.

At least for the findings in this work, feature importance is perhaps more interesting than simply comparing different algorithms to see which classifies best. That is just the first step, and the second is to ask why they belong to that family. It has to be remembered however that any classification and feature extraction of random forests are only relative to the training sets used - if one suspect a protein belongs to a family then the positive dataset is not so much of a problem, but the negative dataset is critical as one are discriminating against it. You want to be in a position to say that

the proteins are classified by features that are distinctly different from those in the negative training set, and so negative training sets should be large and randomly selected to represent background frequencies of any functional groups.



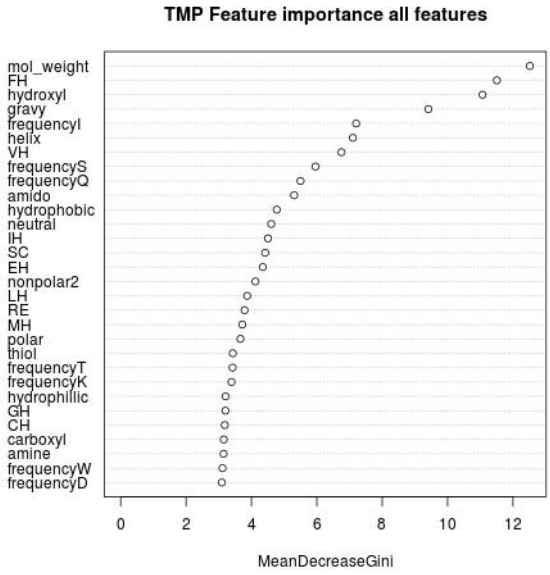
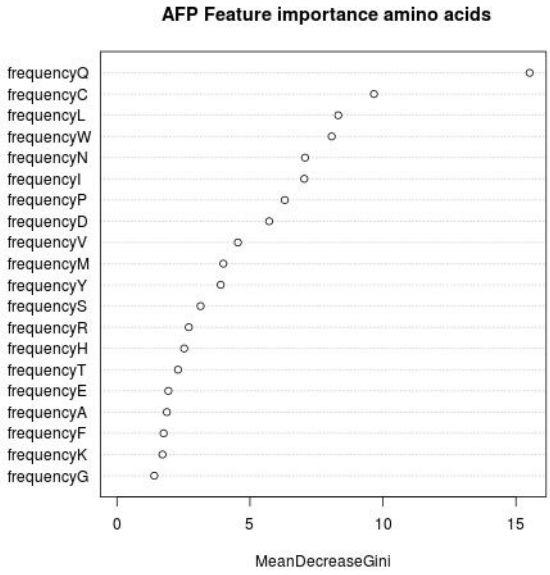


Figure 2.4: A comparison of the feature importance between the frequency of amino acids and the entire feature set in the transmembrane proteins. The frequency of phenylalanine found in helix positions (FH - frequency of helix), closely by molecular weight were found to be the two most important features for splitting the dataset as measured by the Gini index.

Phenyaline helices are integral in promoting folding of proteins to perform the functions that transmembrane proteins carry out in cells, and are also involved initiating interaction between other transmembrane proteins [49]. The role of molecular weight in any type of analysis between proteins is trivial in terms of function, yet useful for discriminatory measures. It is encouraging that the random forest process used the Grand Average of Hydropathy (GRAVY) to split the data. Kyte and Doolittle's work on Hydropathy [28] showed that transmembrane proteins will have a higher GRAVY score than other globular proteins, and I included the calculation of the GRAVY score beforehand to see if it was used to split the data. Other notable features of the data that would perhaps be expected to help determine between a transmembrane protein and a non-transmembrane protein that are also found to be of importance in this study are hydroxyl groups typically found in the form of glycerol that are found in cellular membranes, and the frequency of amino acids found in helices of each protein. Helices are responsible for the structure of transmembrane proteins where Bowie *et al* [6] documented "helix packing" in transmembrane proteins, but will fall to background frequencies in globular proteins.



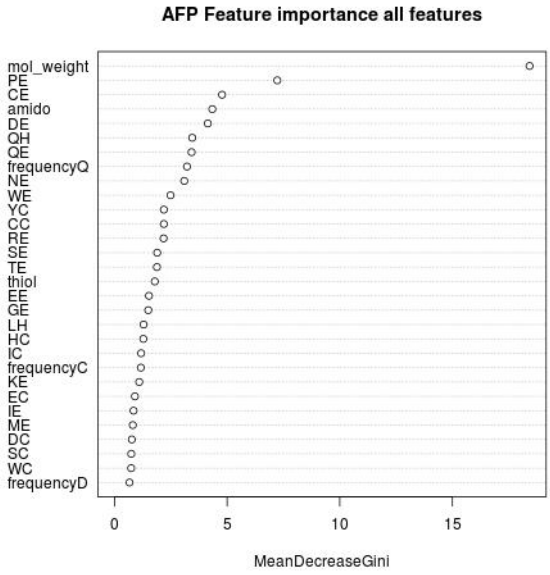


Figure 2.5: A comparison of the feature importance between the frequency of amino acids and the entire feature set in the antifreeze proteins. Molecular weight being the most important feature to split the data doesn't add any important hidden information in particular as on average the non homologous antifreeze "like" proteins are an average of longer sequence length than the antifreeze proteins.

Although random forest classification of antifreeze and non-antifreeze proteins have been reported by Kandaswamy *et al* [24] at just under 85% accuracy, a dataset of antifreeze and antifreeze-like proteins were chosen to test the discriminative power of random forests. It would be expected that it would be difficult to classify two sub families of proteins rather than classifying antifreeze from non-antifreeze proteins. On the face of the results in this experiment a baseline accuracy of 86% was achieved leading up to 92%. However looking at the importance as measured by the Gini index it is clear that molecular weight, and thus sequence length has played a big part in exceeding accuracy normally achieved in protein classification, and if sequence length were such a large factor in the classification process one would perhaps ask if a machine learning technique is needed (the antifreeze-like proteins are on average 2.5 times larger than the antifreeze proteins in terms of sequence length). However the baseline accuracy of 86% achieved with using the amino acid frequencies alone suggest that random forest can classify between the two well. This is not so surprising given that in each of the two sub families of antifreeze proteins, many of the individual proteins will have evolved from a large range of bacterial proteins - each with their own distribution of amino acids, of which through the evolution process some may not be used and get "deleted" from the proteins over time, as well as some proteins inserting amino acids over time. The evolution of proteins is not perfect, and the function of one antifreeze protein may have differed going back through it's evolutionary history, and to achieve it's functional purpose of being an antifreeze now, different paths would have had to be taken in terms of inserting and deleting amino acids from their sequence to cater end up producing their functional attributes. It is an interesting confirmation that functional groups are mainly absent from the Gini importance as the antifreeze and antifreeze-like proteins are near identical in function, yet as expected the structure of the proteins are the most important features, however helical secondary structure does not feature highly in terms of importance

as neither antifreeze groups have helical structure like the transmembrane proteins.

2.5 Summary

The results achieved in the two tests are similar to those found in the literature for other protein families, as well as comparing favorably to SVMs and NNs, where accuracy for protein classifation is usually reported from 80% upwards. It is noted in the literature that transmembrane proteins are known for their high structural homology, a finding reported in this experiment and was the second most important feature used for classification as determined by the Gini Index. A surprising finding is the high classification accuracy when trying to distinguish between antifreeze and antifreeze-like proteins. The two set of proteins were chosen because they carry out similar functions, with subtle differences being that antifreeze-like proteins may also carry out other functions. It is important to note that while length of sequence was not used a feature, a longer sequence has the capacity to contain more functional groups and is a factor in helping random forest distinguish between the two, as antifreeze-like proteins in general have longer sequences.

Random forests are flexible in that many different types of features can be used to classify proteins, with a small number of features such as amino acid frequencies needed to obtain accurate results. Some features however are harder to derive than others. Secondary structure prediction, despite being accurate to 90% [23] involves querying BLAST databases and for that reason is slow (computation time can be between 15-30 minutes per sequence). A database of predicted secondary structure that mirrors PFAM would perhaps be useful when wanting to use secondary structure features. The physiochemical properties were relatively straightforward to compute by using biopython modules, however the way in which they are calculated are complex compared to using simple statistics such as histograms and frequen-

cies.

Also what can be seen from these results is that often, optimum accuracy is achieved after using around 100 trees, so computation time is fast, but as previously mentioned, the data preparation is what takes the most time in running these experiments. The main strengths of using random forests is in its discriminative power and its ability to predict which features best classify the data. The classification however is highly dependant on the training set. Often biologists are faced with the task of classifying proteins from little to no prior knowledge about the protein, and if one is given a protein which one does not know or suspect at least a small group of families it belongs to, random forest will not be so useful as one have no target positive training sets to train with. Equally, one also has to make a choice of what negative training set might one use to discriminate from a family one have guessed the protein actually belongs to? This leads to having to process large negative training datasets to try and achieve a "background" representation of proteins. However if an educated guess or calculation from some other method points the protein into a small group of families, this is where random forest excels as one is able to use it as a magnifying glass to improve upon initial guesses from other methods, as well as ask *why* it might belong to that family by using feature importance measures such as the Gini index. What is perhaps counter intuitive is the fact that random forest appears to discriminate slightly better against two training sets that are similar, as demonstrated by the antifreeze and their non homologous antifreeze "like" proteins. This shows that the data splitting method used in random forest is sensitive to small and subtle changes between the two training sets. This is perhaps more relevant to the problems biologists face today, as it is more likely that one is trying to determine which sub group in a family a protein belongs to, rather than what super family it belongs to, where the negative training sets can be viewed as "competing" training sets. Even so, it must be kept in mind that the classifation has been made

in relation to the negative training set, and illustrates that choosing the training sets and the features extracted are absolutely critical if a thorough and robust analysis is to be carried out.

As we will see in the next section of this thesis, another machine learning technique called Hidden Markov Models are used to classify proteins from a positive training set alone, and as such provide a more standardized measure of protein classification that is dependant only on the family it is classified into.

Chapter 3

Protein classification using profile hidden Markov models

The previous chapter introduced a method for classifying proteins using randomized decision trees and showed high classification accuracy of two different structural protein groups: transmembrane proteins that have high structural homology shared across the family, and antifreeze proteins - a family of proteins that have evolved from many different types of bacteria in parallel and as such only show functional homology, rather than structural.

Implementing Hidden Markov Models for protein classification will take an entirely different approach. Firstly, no negative training set is required to classify proteins using Hidden Markov Models. Secondly, the raw sequence data will be used to classify the proteins, rather than meta data such as pre defined features. Also, rather than simply using one protein, an *alignment* of proteins belonging to one family will be used to describe the sequence parameters that classify the family, and build a model such that a test sequence can be aligned not just to one protein, but effectively to the entire protein set within the family. Multiple sequence alignments are used in algorithms such as BLAST [2] where raw proteins from the same family are aligned and position specific statistics at each column of the alignment are determined.

Another important feature of multiple sequence alignments are that they are able to take into account positions in the sequence alignment that are likely to contain deletions,insertions or substitutions of amino acids as the proteins in a family evolve. Certain amino acids can be substituted without compensating the function of the protein and as such when a certain amino acid is favored in the natural selection process, such substitutions occur. It is therefore important for a multiple sequence alignment model to be able account for such occurrences within a protein family, and HMMs are able to do this. It is therefore important to provide an overview of multiple sequences alignments before defining the HMM architecture, as will be beneficial to know how sequences are aligned and why they are important to HMMs.

3.1 Hidden Markov models from multiple sequence alignment

The previous chapter demonstrated a supervised learning technique that relied solely on descriptive statistics about the amino acid sequences, such as what is the percentage of thyamine in a sequence, or what percentage of the sequence is hydrophobic. In order to understand the strengths of Hidden Markov Models later in this thesis, it is important to understand that we can take a different approach, namely aligning sequences that are suspected to be similar and looking for position specific matches in amino acids between sequences. This way, descriptive statistics aren't needed by the researcher, because statistics about protein families are learnt from multiple sequence alignments of seed sequences in a certain protein family, where the seed sequences are those that best describe the features of a family. There are many elegant methods to align multiple sequences, and to understand what makes a good multiple sequence alignment it helps to think of pairwise alignments first.

Suppose one would want to align the following two sequences 'THI-

SISALINE' and 'THISISALIGNED'. One way to visualize this would be to draw a similarity matrix:

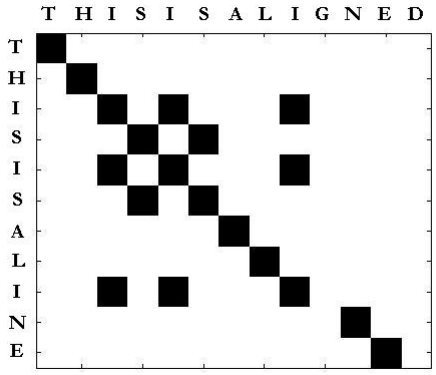


Figure 3.1: A similarity matrix aligning two sequences. Similar proteins that share amino acids in the same position of their proteins will show clear visual correlations of their similarity in a similarity matrix

By organizing one sequence on the x-axis, and the other on the y-axis, sequence similarity can be visualized quite easily. When an element of one sequence is the same is that of the other, a "dot" is drawn on the x-y plot. It is obvious that two identical sequences of same length will show a dotplot with a perfect diagonal line, where sequences of some local similarities will give diagonal regions outside of the main alignment. The following is a prion protein nucleotide sequence found in two sub-species of wild sheep; the mouflon and takin:

```
moufflon =

ATGGTGAAAAGCCACATAGGCAGTTGGATCCTGGTTCTCTTTGTGGCCATGTGGAGTGACGTGGGCCTC
TGCAAGAAGCGACCAAAACCTGGCGGAGGATGGAACACTGGGGGGAGCCGATACCCGGGACAGGGCAGT
CCTGGAGGCAACCGCTATCCACCTCAGGGAGGGGGTGGCTGGGGTCAGCCCCATGGAGGTGGCTGGGGC
CAACCTCATGGAGGTGGCTGGGGTCAGCCCCATGGTGGTGGCTGGGGACAGCCACATGGTGGTGGAGGC
TGGGGTCAAGGTGGTAGCCACAGTCAGTGGAACAAGCCCAGTAAGCCAAAAACCAACATGAAGCATGTG
GCAGGAGCTGCTGCAGCTGGAGCAGTGGTAGGGGGCCTTGGTGGCTACATGCTGGGAAGTGCCATGAGC
AGGCCTCTTATACATTTTGGCAATGACTATGAGGACCGTTACTATCGTGAAAACATGTACCGTTACCCC
AACCAAGTGTACTACAGACCAGTGGATCAGTATAGTAACCAGAACAACTTTGTGCATGACTGTGTCAAC
ATCACAGTCAAGCAACACACAGTCACCACCACCACCAAGGGGGAGAACTTCACCGAAACTGACATCAAG
ATAATGGAGCGAGTGGTGGAGCAAATGTGCATCACCCAGTACCAGAGAGAATCCCAGGCTTATTACCAA
AGGGGGGCAAGTGTGATCCTCTTTTCTTCCCCTCCTGTGATCCTCCTCATCTCTTTCCTCATTTTTCTC
ATAGTAGGATAG

takin =

ATGGTGAAAAGCCACATAGGCAGTTGGATCCTGGTTCTCTTTGTGGCCATGTGGAGTGACGTGGGCCTC
TGCAAGAAGCGACCAAAACCTGGCGGAGGATGGAACACTGGGGGGAGCCGATACCCGGGACAGGGCAGT
CCTGGAGGCAACCGCTATCCACCTCAGGGAGGGGGTGGCTGGGGTCAGCCCCATGGAGGTGGCTGGGGC
CAACCTCATGGAGGTGGCTGGGGTCAGCCCCATGGTGGTGGCTGGGGACAGCCACATGGTGGTGGAGGC
TGGGGTCAAGGTGGTAGCCACAGTCAGTGGAACAAGCCCAGTAAGCCAAAAACCAACATGAAGCATGTG
GCAGGAGCTGCTGCAGCTGGAGCAGTGGTAGGGGGCCTTGGTGGCTACATGCTGGGAAGTGCCATGAGC
AGGCCTCTTATACATTTTGGCAGTGACTATGAGGACCGTTACTATCGTGAAAACATGTACCGTTACCCC
AACCAAGTGTACTACAGACCAGTGGATCAGTATAGTAACCAGAACAACTTTGTGCATGACTGTGTCAAC
ATCACAGTCAAGCAACACACAGTCACCACCACCACCAAGGGGGAGAACTTCACTGAAACTGACATCAAG
ATAATGGAGCGAGTGGTGGAGCAAATGTGCATCACCCAGTACCAGAGAGAATCCCAGGCTTATTACCAA
AGGGGGGCAAGTGTGATCCTCTTTTCTTCCCCTCCTGTGATCCTCCTCATCTCTTTCCTCATTTTTCTC
ATAGTAGGATAG
```

Figure 3.2: The two prion protein nculeotide sequences are identical apart from one element in the sequence. How long does it take to spot this by eye?

The similarity matrix can be plotted as:

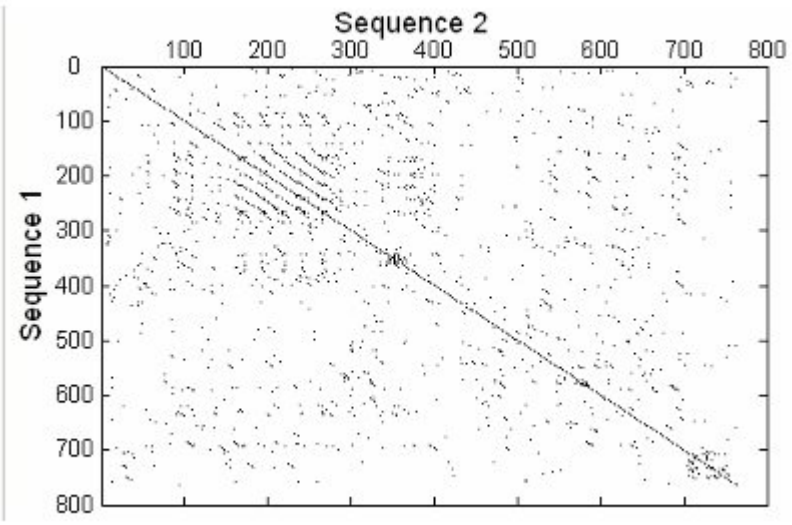


Figure 3.3: The dotplot show high homogeneity between the two sequences, made clear by the diagonal line through the center of the dotplot.

Aligning sequences like this has a major advantage over the descriptive features used in the random forest dataset. For example if one randomly orders the takin sequence while keeping the same percentages of each nucleotide, as well as maintaining the length of the sequence, aligning this to the mouflon sequence would yield a similarity matrix with no correlation between the two proteins

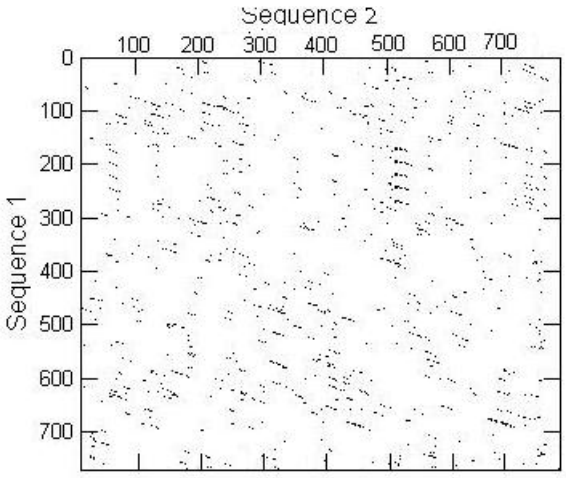


Figure 3.4: Despite the two proteins having the same frequency of amino acids, the order of the takin protein no longer correlates with the moufflon protein.

where no correlation can be found, and rightly so as the mouflon sequence is aligned to the fake sequence as any background sequence might. If the training data for random forest simply contained the percentage of amino acids per sequence, random forest would likely classify the fake sequence and the mouflon into the same family. Only adding more information in such as how many regions are hydrophobic would separate the two. However the usefulness of similarity matrices stop at being a visulaisation tool. They are one of the oldest methods for pairwise alignments, but if one look at any family of proteins there are always proteins that only share local regions of alignment, and so some algorithm must be constructed to align sequences and take into account local alignment.

3.2 Global and local alignments

The Needleman-Wunsch (NW) algorithm, introduced in 1970 [33] algorithm does not necessarily treat mismatches in a pairwise alignment as a "miss" in the alignment. This is because in nature it is acceptable to substitute

amino acids in a protein, but still keep the same protein function. The NW algorithm was designed to account for such substitutions by using a substitution matrix $S(a, b)$

$$S(a,b) = \begin{array}{c} \\ A \\ G \\ C \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & G & C & T \end{array} \\ \begin{pmatrix} 10 & -1 & -3 & -4 \\ -1 & 7 & -5 & -3 \\ -3 & -5 & 9 & 0 \\ 4 & -3 & 0 & 8 \end{pmatrix} \end{array} \tag{3.1}$$

where for any substitution between a sequence element a to a sequence element b, a score is given that either penalizes or strengthens the substitution. This scoring method allows for family members to be found in alignments as opposed to relying on a visual dotplot. There is also another factor that the NW algorithm, as well all algorithms after it, and that is it can deal with insertions or deletions in a sequence. As proteins evolve it is not just amino acid substitutions that account for any diversity, but amino acids that are not needed will be deleted, and amino acids that are needed for a protein to perform a task within the evolving organism will be added. One can imagine that the entire alignment would be shifted when an indel (insertion or deletion) is observed. This is extremely important as one could technically add in as many insertions in a sequence to achieve perfect alignment:

```
THISISALI-NE-
|||||||||  | |
THISISALIGNED
```

where the red letters indicate a match. One can notice here that every letter in the first sequence has conveniently found a match in the second sequence, but only because dashes have been added into the sequence to make a better alignment, perhaps indicating there were deletions in this sequence, and so an insertion may be added. A problem arises however if one is allowed to add as many insertions as one like to make as best alignment

as possible. A sequence with more insertions than amino acids, but with a good alignment score would actually be a terrible alignment, and so the NW algorithm arbitrarily penalizes the alignment score per indel, given by d. The NW algorithm for scoring substitutions and indels in pairwise alignment use three basic calculations in the algorithm

$$F_{0j} = d * j \tag{3.2}$$

$$F_{i0} = d * i \tag{3.3}$$

$$F_{ij} = max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d \tag{3.4}$$

F_{ij} is the optimal score that is updated with every iteration through the alignment and $S(A_i, B_j)$ is the substitution matrix for amino acid substitutions. The following shows how to obtain the optimum NW alignment for sequences 'TTCATA' and 'TGCTCGTA':

		T	G	C	T	C	G	T	A
	0	-6	-12	-18	-24	-30	-36	-42	-48
T	-6								
T	-12								
C	-18								
A	-24								
T	-30								
A	-36								

		T	G	C	T	C	G	T	A
	0	-6	-12	-18	-24	-30	-36	-42	-48
T	-6	5							
T	-12								
C	-18								
A	-24								
T	-30								
A	-36								

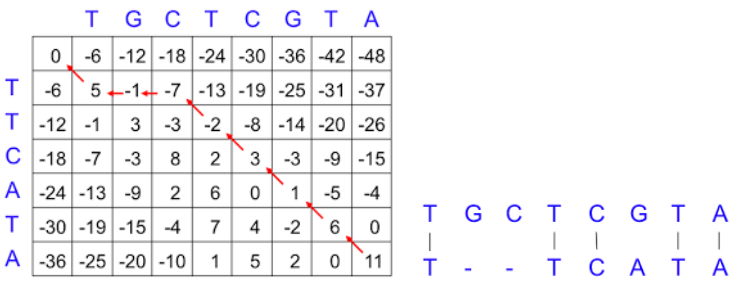


Figure 3.5: The NW-algorithm traversing through the sequence alignment space and calculating the optimal alignment. Starting from the last position in the alignment space, the optimal alignment is found by traversing back through cells in the alignment space to the $i-1$ cell that was used to calculate the value of cell i. The score of each cell is stored as the algorithm traverses back through the alignment space until it arrives at the start of the sequence alignment.

Because the NW-algorithm is a global algorithm, it forces alignment over the entire length of the longest sequence. This can also contribute to false negatives in finding possible family members by having criteria that are too strict.

Some protein sequences from the same family do deviate globally, but have strong regions of local alignment. Local alignment algorithms such as the Smith-Waterman algorithm introduced in 1981 [36] allow more flexibility in the sense that alignment isn't forced globally. The idea behind the S-W algorithm is to modify the N-W algorithm by providing thresholds where, wherever the global alignment score falls below the threshold,the path is abandoned and another local alignment begins from the next highest score in the alignment space.

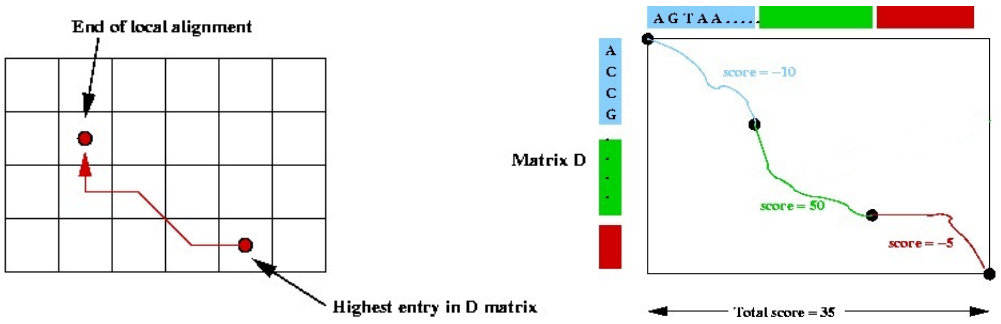


Figure 3.6: Here it is possible to see that a global alignment can be traded for multiple local alignment, where it is usual for smaller local alignments to have greater scores than a global alignment. `http://www.seas.gwu.edu/~simhaweb/cs151/lectures/module12/align.htm`

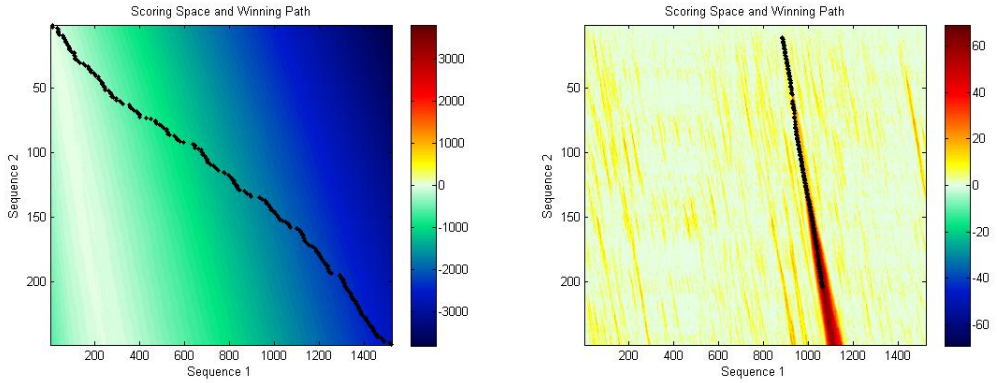


Figure 3.7: A comparison of the NW and SW alorgorithms aligning the Bai3 coupled protein receptor and the VIPR2 transmembrane receptor. The SW alignment has a far higher score than the NW alignment, suggesting the two proteins have similar function, but probably does not belong to the same family of proteins.

It becomes apparent that the NW and SW-algorithms should really be used for different situations. Since the NW-algorithm is forced globally, it is used best where the two sequences are known to be of some similarity to begin with, and really the NW-algorithm tests exactly *how* similar they are.

The SW-algorithm however finds regions of local alignments, so in the case where one do not know if two proteins are similar, using a SW alignment will tell the user if there are any areas where the two proteins are similar. Extremely good matches are then often passed onto the SW-algorithm.

The true power of HMM's lies in the fact that they can take advantage of aligning *multiple* sequences together and use statistics of that alignment to build a model that captures the sequence information. This means that one can effectively align a sequence to not just one sequence, but to multiple sequences by aligning to the HMM. This is a powerful tool that can both provide an architecture to store protein family statistics, as well as providing a tool to score proteins for potential family members. The next section introduces the learning algorithms used to build an HMM based on sequences and how to score a sequence against an HMM.

3.3 Hidden Markov model theory

Hidden Markov Models are stochastic, probabilistic model, whereby a finite set of states S discreet symbols O that are emitted from S, a probability transition matrix $T = t_{ji}$ to describe how to transition between different states in S, and a probability emission matrix $E = (e_{ix})$ to describe the probability of a symbol A being emitted at state S. For each state i in the system, there is a probability e_{ix} of emitting symbol x and a probability t_{ji} of transitioning to state j. Probability of an emission or a transition is determined by the current state only, as described by the first-order Markov principle, that is previous states and their emission and transition probabilities do not affect future states. A simple HMM might be observing the sequence of amino acids of a protein consisting of 4 types of emission A,T,C and G although the current amino acid will be T, the protein may be in part of a sequence which is rich with amino acids C and G, called CpG rich segments of a protein. The underlying segment of the sequence being described as CpG rich

is the hidden state, and the amino acid is the observable state. HMM's can generally be used to answer three types of questions. Suppose we observe the 10 amino acids of a protein sequence, what are the emission and transition parameters of our model going to be? If we have a HMM with emission and transition probabilities, then given a sequence of emissions, how likely is it that this sequence could be generated by walking through our HMM model? And finally, given a HMM model, what is the most likely sequence of emissions to be observed when walking through the model? We will need to use all three features of HMM's for protein classification to learn the parameters of a protein family, use those parameters to generate sequences belonging to that model, and we will also need to use the model to score an observed sequence of emissions and classify the protein into a certain family. The next section introduces the theory of how HMM's answer these three questions.

3.4 The forward,backward and viterbi algorithms

This section defines the algorithms used to determine the parameters of a HMM, the likelihood and most probable sequence of emissions and state transitions associated with an observed sequence - the forward, backward and viterbi algorithms. These algorithms are all dynamic programs and are all recursive, that propagate through the HMM architecture.

Lets consider how we can calculate the likelihood $P(A|w)$ of a sequence $A = X^1..X^2..X^N$ given a HMM model $M = M(w)$. The path π through a protein profile HMM M is defined as the sequence of states in M starting at a definitive *start* state and ending with an *end* state, where each emitting state in M emits symbols of a defined alphabet. If the likelihood of emitting

a symbol along path O is

$$P(O, \pi|w) = \prod_{start}^{end} t_{ji} \prod_{t=1}^{T} e_{iX^t} \tag{3.5}$$

where the first product denotes transitions along the path π from the start and end states, and the second product denotes the emissions states i along π, then an intuitive way to calculate the likelihood of emitting a sequence of emissions might be

$$P(O|w) = \sum_{\pi} P(O, \pi|w) \tag{3.6}$$

However, summing over all the paths along π to find the likelihood is typically exponential in nature, so we would be better off searching for a more efficient solution. One such way of doing this is described by $t$he forward algorithm. The forward algorithm, as well as the backward and viterbi algorithms iteratively propagate through π to calculate the likelihood of emitting a sequence, while avoiding calculating all emissions for all paths.

We can define the probability of being in state i at time t as

$$\alpha_i(t) = P(S^t = i, X^1...X^t|w) \tag{3.7}$$

having observed symbols $X^1...X^t$ from emission states in the HMM model $M(w)$. The start state can be defined as

$$\alpha_{start}(0) = 1 \tag{3.8}$$

since we always start at the start state. The likelihood of a sequence is being emitted from M then

$$P(O|w) = \alpha_{end}(T) \tag{3.9}$$

Therefore we must be able to compute the $\alpha_i(t)$ state and following states

$\alpha_i(t+1)$ for emitting recursively as we propagate through M. This can be defined as

$$\alpha_i(t+1) = \sum_{j \in S} \alpha_j(t) t_{ij} e_{iX^{t+1}} = \sum_{j \in N_{-i}} \alpha_j(t) t_{ij} e_{iX^{t+1}} \tag{3.10}$$

where N_{-i} is the neighborhood notation used to describe the previous neighbors in the model. Each iteration converges to a stable set of values $\alpha(t+1)$. The special case of a deletetion in a sequence as described when aligning two proteins calls for a simple modification in how to iterate through the model, that is

$$\alpha_i(t+1) = \sum_{j \in N_{-i}} \alpha_j(t+1) t_{ij} \tag{3.11}$$

where the contribution of the emission state is removed, and at most can be iterated N times to delete all emissions. Propagating through the model and calculating $\alpha_i(t+1)$ is done in both time T layers and M units in each layer per HMM state. In such a model, both T and M are computed in $O(N)$ so the forward algorithm is computed in $O(N^2)$ operations. The backwards algorithm propagates probabilities backwards through M, and is therefore the reverse of the forward algorithm. The backwards algorithm variables can be defined as

$$\beta_i(t) = P(X^{t+1} ... X^T | S^t = i, w) \tag{3.12}$$

which is the probability that the model is in state i and time t when observing the sequence X^{t+1}, where as in the start of the forward algorithm

$$\beta_{end}(T) = 1 \tag{3.13}$$

To propagate backwards through M we recursively compute $\beta_i(t)$ for emis-

sions as

$$\beta_i(t) = \sum_{j \in N_{+i}} \beta_j(t) t_{ji} e_{iX^{t+1}} \tag{3.14}$$

and for delete states as

$$\beta_i(t) = \sum_{j \in N_{+i}} \beta_j(t) t_{ji} \tag{3.15}$$

which when computing these variables at each state visited in M, the complexity is of the order $O(N^2)$ just as in the forward algorithm. Combining the forward and backward algorithms we can compute the probability of being in state i at time t when given an observed sequence (i.e. whether a fair or loaded dice is being thrown based on the observations). The probability of being in state i at time t is then

$$\gamma_i(t) = P(S^t = i|O, w) = \frac{\alpha_i(t)\beta_i(t)}{(P|O, w)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j \in S} \alpha_j(t)\beta_j(t)} \tag{3.16}$$

and the probability of transitioning from state $i \rightarrow j$ at time t to $t+1$ for an observed sequence,X is

$$\gamma_{ij}(t) = P(S^t = i, S^{t+1} = j|O, w) = \frac{\alpha_i(t)\alpha_{ij}\beta_i(t+1)\beta_j(X^{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i(t)\alpha_{ij}\beta_i(t+1)\beta_j(X^{t+1})} \tag{3.17}$$

The most likely states at time t can be found by maximizing $\gamma_i(t)$ and $\gamma_{ij}(t)$. By combining the forward and backward variables, it is possible to determine the most likely state at a given time, and to determine the most likely pathway of states in a particular model we can use the viterbi algorithm. By finding the most likely path through a model we will be able to align novel

sequences of symbols, such as proteins to an protein-HMM model and score each sequence.

The Viterbi algorithm gives the most probable path through a HMM model, given a set of observed symbols. This most probable path is defined as

$$\delta_i(t) = \max_{\pi_i(t)} P(\pi_i(t)|w) \tag{3.18}$$

where $\pi_i(t)$ is a predetermined path through the model characterized by sequence emissions $O = X^1...X^t$ up to state i. Like in the forward algorithm, the Viterbi algorithm updates the cumulative probability as it propagates through the states, where the probability of being in future state at $t+1$ is

$$\delta_i(t+1) = [\max_j \delta_j(t) t_{ij}] e_{iX^{t+1}} \tag{3.19}$$

for emitting states, and

$$\delta_i(t+1) = [\max_j \delta_j(t+1) t_{ij}] \tag{3.20}$$

for delete states. The Viterbi path is calculated by storing the previous optimal state, and as the algorithm iterates through O the most probable path is found. Notice that each time a delete state is found to be an optimal state in the Viterbi path, the probability of the entire path is decreased, meaning that a sequence consisting entirely of delete states will never be an optimal path.

3.5 Expectation of the hidden states

The forward,backward and Viterbi algorithms allow us to firstly determine the probability of being in a state at a given time, and also the most probable path through a model. Returning to the fair-loaded dice example, it is possible to calculate the probability distribution of hidden states $Q(\pi)$. To

do this we will need to compute the expectations of states,transitions and emissions in the HMM model. Define these three properties as

- $n(i, \pi, O)$ is the number of times state i is visited, given π and sequence O
- $n(i, X, \pi, O)$ is the number of times letter X is emitted from state i, given π and O.
- $n(j, i, \pi, O)$ is the number of times transition from state $i \to j$ occurs, given π and O

where the expectations for each are calculated as

$$n_i = \sum_{\pi} n(i, \pi, O) P(\pi|O, w) = \sum_{t=0}^{T} \gamma_i(t) \tag{3.21}$$

$$n_{iX} = \sum_{\pi} n(i, X, \pi, O) P(\pi|O, w) = \sum_{t=0}^{T} \gamma_i(t) \tag{3.22}$$

$$n_{ji} = \sum_{\pi} n(j, i, \pi, O) P(\pi|O, w) = \sum_{t=0}^{T} \gamma_{ji}(t) \tag{3.23}$$

where these properties are used to calculate the emission and transition parameters of a HMM model. There are various learning algorithms that use these properties, with the most common being the Baum-Welch algorithm being a widely used one.

3.6 Learning algorithms

When given a set of data, we would like to build an HMM architecture that encompasses all the features that define the data, so that the HMM can generate similar data based on the learned features. There are various

algorithms that can be used - maximum posterior parameter estimation using maximum likelihood,expectation maximization algorithms such as the Baum-Welch algorithm, the Viterbi algorithm which uses path calculation based on multiple local likely paths, rather than global, and a gradient descent approach which uses backward propagation to incorporate elements of neural networks into the HMM model.

To understand how expectation maximization works for the Baum-Welch algorithm, it is necessary to consider maximum likelihood estimation of emission and transition parameters. This is a low-level Bayesian inference that determines maximum a-priori parameters (MAP) in the HMM. As before, if the likelihood of a sequence is $P(O|w) = \sum_{\pi} P(O, \pi|w)$, then we can determine the dynamics of emissions in the system by minimizing the Lagrangian

$$\mathcal{L} = -logP(O|w) - \sum_{i \in E} \lambda_i (1 - \sum_{X} e_{iX}) - \sum_{i \in S} \mu_i (1 - \sum_{j} t_{ji}) \tag{3.24}$$

where μ and λ are Lagrange multipliers. The partial derivatives of the Lagrangian with respect to emissions is then

$$\frac{\partial P(O, \pi|w)}{\partial e_{iX}} = \frac{n(i, X, \pi, O)}{e_{iX}} P(O, \pi|w) \tag{3.25}$$

where $n(i, X, \pi, O)$ is the number of times i is visited given the path π. The Langragian is at the optimum when the partial derivatives are set to 0 and we obtain

$$\lambda_i e_{iX} = \sum_{\pi} n(i, X, \pi, O) Q(\pi) = n_{iX} \tag{3.26}$$

Which is the expectation value. If $Q = P(\pi, |O, w)$, the posterior probability,

then the expectation of the number of times state i is visited is

$$\lambda_i = \sum_{\pi} \sum_{X} n(i, X, \pi, O) Q(\pi) = \sum_{\pi} n(i, \pi, O) Q(\pi) = n_i \tag{3.27}$$

and at the optimum we obtain

$$e_{iX} = \frac{\sum\limits_{\pi} n(i, X, \pi, O) Q(\pi)}{\sum\limits_{\pi} n(i, \pi, O) Q(\pi)} = \frac{\sum\limits_{\pi} P(\pi|O, w) n(i, X, \pi, O)}{\sum\limits_{\pi} P(\pi|O, w) n(i, \pi, O)} \tag{3.28}$$

and thus the re-estimation equations for the number of times a symbol X is observed in state i,e_{iX}, or a transition is made from state $i \to j$, t_{ji} are

$$e_{iX}^{+} = \frac{\sum\limits_{\pi} n(i, X, \pi, O) Q(\pi)}{\sum\limits_{\pi} n(i, \pi, O) Q(\pi)} = \frac{\sum\limits_{t=0, X^t = X}^{T} \gamma_i(t)}{\sum\limits_{t=0}^{T} \gamma_i(t)} = \frac{n_{iX}}{n_i} \tag{3.29}$$

$$t_{ji}^{+} = \frac{\sum\limits_{\pi} n(i, X, \pi, O) Q(\pi)}{\sum\limits_{\pi} n(i, \pi, O) Q(\pi)} = \frac{\sum\limits_{t=0} \gamma_{ji}(t)}{\sum\limits_{t=0}^{T} \gamma_i(t)} = \frac{n_j}{n_i} \tag{3.30}$$

One might ask how to compute this directly without knowing the posterior distribution of the hidden variables Q as it depends on prior emission values e_{iX}. This is done by using the same assumption in computing expectation values, $Q(\pi) = P(\pi|O, w)$, so that if $P(O, \pi|w) = \prod\limits_{start}^{end} t_{ji} \prod\limits_{t=1}^{T} e_{iX^t}$ then $Q(\pi)$ by computed iteratively by updating e_{iX} in the previous equation. This iterative process of learning Q is exactly how the expectation maximization

algorithm works, and the Baum-Welch algorithm in particular calculates both e_{iX} and t_{ji} using the variables $\gamma_i(t)$ and $\gamma_{ji}(t)$ from the forward and backward algorithms - for that reason the Baum-Welch algorithm is often called the forward-backward algorithm. Each sequence entered into the model requires one forward and one backward iteration leading to a computation time of the order $O(KN^2)$.

The gradient descent method is similar to EM for learning parameters in HMMs in that it uses the forward-backward technique to determine e_{iX} and t_{ji}, however it differs in that a learning rate, η is specified. This learning rate helps to avoid large advances towards an incorrect convergence over single isololated sequences in on-line learning, a problem that the Baum-Welch algorithm can suffer from. Let us parameterize e_{iX} and t_{ji} by normalizing exponentially

$$e_{iX} = \frac{e^{w_{iX}}}{\sum_Y e^{w_{iY}}} \tag{3.31}$$

$$t_{ji} = \frac{t^{w_{ji}}}{\sum_Y e^{w_{ki}}} \tag{3.32}$$

where w_{iX} and w_{ji} are the parameterization variables. By taking the derivative with respect to these variables for gradient descent, the emission probabilities are computed as

$$\frac{\partial e_{iX}}{\partial w_{iX}} = e_{iX}(1 - e_{iX}) \tag{3.33}$$

$$\frac{\partial e_{iX}}{\partial w_{iY}} = e_{iX}e_{iY} \tag{3.34}$$

It can be seen that using the $w's$ in the parameterization process ensures that

neither emission or transition probabilities reach 0. Using the chain rule

$$\frac{\partial P(O|w)}{w_{iX}} = \sum_{Y} \frac{\partial P(O|w)}{e_{iX}} \frac{\partial e_{iY}}{w_{iX}} \tag{3.35}$$

where the gradient descent equations for the log-likelihood are

$$\delta w_{iX} = \eta(n_i X - n_i e_{iX}) \tag{3.36}$$

and

$$\delta w_{ji} = \eta(n_{ji} - n_i t_{ji}) \tag{3.37}$$

η is the learning rate, and $n_i X$ and n_{ji} are the expectation values from the forward-backward algorithm. The gradient descent algorithm also requires a forward and backward step, making compute time of the order $O(KN^2)$.

Viterbi learning, unlike the EM and gradient descent methods focusses on calculating expectations only along likely paths, rather than updated calculations based on all paths. It is often that for any sequence, only the most likely path π^* is used so that the emission count over that path is $n(i, X, \pi, O)$ and the counts of symbol X being emitted from state i is $\pi^*(O)$. Because only one path, the most likely path is considered, averaging count per state over one path would yield 0 or 1 for any particular emission state i, making a Viterbi EM algorithm redundant, but using parameterization in the gradient descent method yields more information from state emissions. For every step along a Viterbi gradient descent path, the parameters of states i in the most likely path π^* updated as

$$\delta w_{iX} = \eta(E_i X - e_{iX}) \tag{3.38}$$

and

$$\delta w_{ji} = \eta(T_{ji} - t_{ji}) \tag{3.39}$$

where E_{iX} and $T_{ji} = 1$ if X is emitted or a transitition from $i \rightarrow j$ is observed, but otherwise 0. The $w's$ are updated via the difference between training data frequency and probabilities e_{iX} and t_{ji} of the HMM. The trade-off for Viterbi learning is as expected speed vs accuracy. The compute time is reduced by not computing the variables in the backward propagation step, however optimization is only good for a well defined global likely path, but when subtle differences in sequences are due to multiple local optimized paths, or that there may be no optimized path, non-Viterbi algorithms will perform with higher accuracy.

The next section uses all aspects and tools that HMM's have to offer. Using protein sequence data, a HMM architecture will be trained to classify proteins into families, and given a trained HMM demonstrate that a given test sequence can be decoded against the model to determine if it belongs to that family. These special types of HMM's are called protein profile HMM's.

3.7 Protein profile HMMs

Proteins and nuceloetide sequences can be used as an input to a hidden Markov model to classify proteins into families and the organisms which have the protein. It has already been discussed that pairwise sequence alignment can show areas of a protein sequence that are conserved over time, as well as crucial parts of protein sequences that are inherited throughout a family, thus being able to help classify them into families. It is advantageous to use the nature of position-specific information of pairwise alignments to score how likely a sequence belongs to the same hierarchy as those in the alignment [22]. The statistics behind such alignments are sound and have proven to produce accurate results, in particular BLAST/PSI-BLAST [2], where powerful scoring matrices, known as BLOSUM matrices are produced to aid pairwise sequence alignment classification.

The introduction of HMMs allowed statistical analysis of protein 'profiles',

built from multiple sequence alignments. This is extremely useful as pairwise alignments assume that all positions in the two sequences are of equal importance in terms of protein homology, where profile models indicate that sequence conservation across protein sequences is non-uniform. Analyzing profiles of multiple sequence alignments causes problems ranging from the sheer amount of parameters that can contribute the classifying a family, to how to determine the best way to score indels in an alignment. As seen earlier between the Needleman-Wuncsh and Smith-Waterman alignments, ad-hoc scoring methods can be incorporated into alignment algorithms, however a well understood mathematical model is needed to analyze protein profiles.

Hausller and Krogh *et al* [27] first began to see the potential of using HMMs to analyze profiles of multiple sequences alignments, where many groups began work on defining the HMM architecture to incorporate protein profiles, with the aim of utilizing the power of HMM's to build a statistical model that could analyze profile of gapped (insertions and deletions) alignments. Eddy *et al* [16] defined a protein profile HMM (p-HMM) that is now widely used and implemented using the HMMER software [17] which builds profile HMM's for the PFAM database. Where the advantages of analysis of aligned sequences are shown in pairwise alignments algorithms like the Needleman-Wunsch algorithm, Eddy's p-HMM provided a way to use both unaligned and aligned sequences for analysis. Where the advantages of analyzing aligned sequences are shown in pairwise alignments algorithms like the Needleman-Wunsch algorithm, Krogh's and Eddy's p-HMMs provided a way to use both unaligned and aligned sequences for analysis. Although the majority of sequences used to build an aligned profile consist of match states across the columns of the alignment, two additional states are added to the match states to incorporate aligned profile. These are *delete* and *insert* states. A delete state allows for a deletion in an amino acid sequence across the columns of the alignment and emits nothing, i.e. no emission or transition, and an insert state allows for new amino acids being built into

the protein, where any amino acid can be emitted from an insert state, in which it transitions to itself, allowing multiple amino acids to be inserted while staying in the same insert state. Profile HMM's can also be trained from unaligned data, in which the deletion and insertion probabilities are determined statistically through Baum-Welch or Viterbi training, rather than a trial and error approach in pairwise alignment algorithms. The flexibility of p-HMMs is hugely popular while offering the same, if not better accuracy than programs such as BLAST, and the ability to handle deletions and inserts in amino acids is a crucial part of finding highly divergent proteins that have evolved over time.

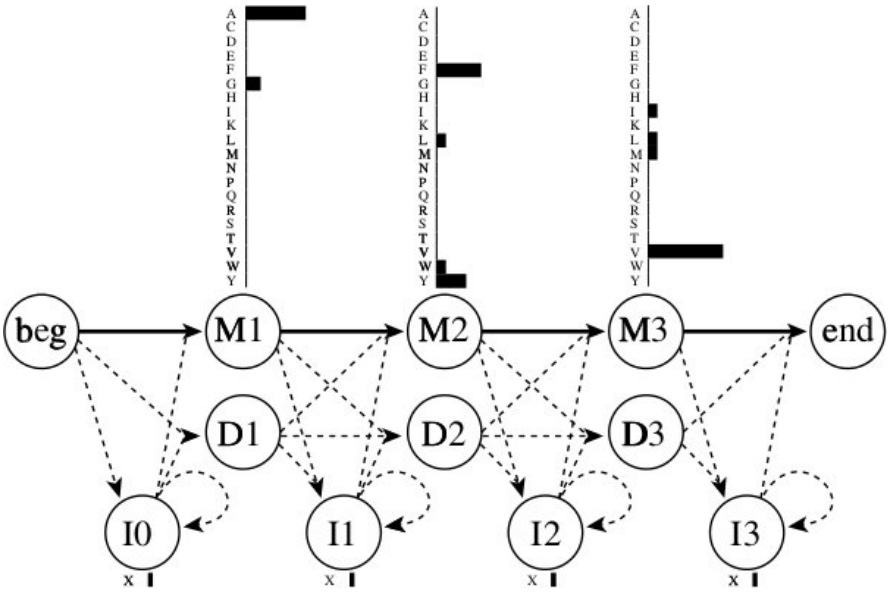


Figure 3.8: An example of a profile HMM. States B and E are beginning and end states. Match states are a normal part of both aligned and unaligned sequences, but delete/insert states are needed to create a p-HMM based on aligned data. Delete states do not emit anything from the model, and insert states can emit any amino acid, and can either transition back to itself, or to a match state [16]

The score produced by aligning sequences to a p-HMM are traditionally given a log-odds ratio for match states, where if the probability of a match state being emitted is residue x is p_x and the probability of a background/null

sequence is p_n then the log-odds score for that residue in a match state is

$$S = \frac{p_x}{p_n} \tag{3.40}$$

where S is comparable to scores given by FASTA or BLAST for unaligned sequences. Deletions and insertions are handled differently from those of pairwise alignments, where ad-hoc gap penalties are used. If x is the amount of residues being emitted from an insert state, gap penalties can be split into gap open penalties a - entering and leaving an insert state, and gap extend penalties b - the amount of residues emitted as a result of being in an insert state. To penalize insertions in a sequence, the probability of transitioning from a match state to a transition state tMI, emitting a residue while in an insert state tII and transitioning from an insert state to a match state tIM are used where the costs are $log(tMI)$,$log(tII)$ and $log(tIM)$ respectively so that the total insert cost is calculated as

$$C = a + b(x-1) = log(tMI.tIM) + log(tII)(x-1) \tag{3.41}$$

where the difference compared to traditional pairwise alignments is that the gap penalties calculated by HMM's are non-arbitrary. Gap penalties for pairwise alignments are ad-hoc and each insert or delete penalisation are scored the same as the next insert or delete, without taking into account which match state came before it. In fact, the gap penalties for traditional pairwise alignments may use global penalty parameters that do not reflect the sequences being compared. Insertions in sequences are dependant on the match states around them, and that is exactly what a p-HMM takes into account - the probability of an insert occurring in a sequence is actually linked to the probability of transitioning from a match state to an insert state. It is worth noting that even transitioning from a match state to a match state has a cost tMM, depending on which amino acid is next emitted. The probabilities tMM and tMI are the tradeoff probabilities that

determine the score of an entire sequence against a HMM - increasing the probability of tMI decreases the probability of tMM. Inserting amino acids into a protein has in the past been either thought to have a background probability, or the methodology had not been developed to take into account that some amino acids are more likely to be inserted into certain parts of the sequence - for instance surface loops on transmembrane proteins are often inserted into the protein sequence, in which the amino acids will reflect runs of hydrophilic structures. Clearly the background frequency of amino acids is not sufficient to model amino acid emission probabilities at insert states for these situations. By modelling a p-HMM on multiple alignment profiles, insertion emission probabilities can be calculated and linked to match-insert and insert-insert states.

Profile HMMs are different from other multiple alignment methods in that the match, insert and deletion parameters are not specified, indeed even known for the particular set of sequences the multiple alignment is based on. Unlike BLOSUM matrices describing global scoring methods for matched amino acids and tolerating amino acid substitutions, p-HMMs require no prior scoring matrices as they are learned and tailored to the multiple alignment. This is not to say that p-HMMs cannot use prior information during the training phase, emission, transition and insertion parameters can all be biased towards certain amino acids, for classifying hydrophobic proteins and biasing the amino acids to suit hydrophobic chains. The flexible manner of learning how to score insertions and deletions often sees p-HMMs outscoring other methods when the multiple alignment contains many gaps [18] meaning p-HMMs are often better for confirming/refuting distant homologues, and using biasing techniques can cut down the size of the proteins used for multiple alignment for accurate classification [3]. Profile HMMs are now the choice algorithms used to classify proteins into domains and families, where pairwise alignment algorithms like BLAST specialize in scoring single sequences against any other sequence.

The rise of p-HMMs is largely responsible for the mass classification of proteins into families. The Pfam database,developed by Erik Sonhammer is a special database - not only is it the leading database that holds protein family domains, but each family is constructed on the basis of p-HMM scores and those included in the family score above a given threshold in Pfam for that given family.

3.8 Protein classification using profile hidden Markov models

Profile Hidden Markov models have been used extensively for protein classification, where two reputable databases PFAM [46] and PANTHER [48] curate protein families based on p-HMMs. The databases are typically divided into clans, families and sub-families and a large amount of studies using protein data use databases such as PFAM and PANTHER, in which Eddy's software, hmmer is used to classify all families to similar, if not better accuracies achieved with other machine learning techniques, including software suggest as BLAST. Profile HMM's work particularly well at detecting remote and distant homologs [26] by using sequence alignments alone and can incorporate other features such as secondary structure and evolutionary relationships from Phylogenetic trees [39]. Classification of such homologs using HMM's rely on the fact that sequences in a certain homolog have similar sequence data when aligned to each-other, rather than from a functional or structural point of view, and as such is excellent at detecting proteins which have evolved from an "ancestor" protein even through different organisms and species. This is where some machine learning techniques would struggle to find a true positives / negatives when querying a protein homology. For example the Random Forest algorithm in the previous chapter did not rely on sequence alignment, but rather on selected features such as frequency of amino acids, secondary structure and functional groups, where some proteins

not within the same homology will contain similar features and structures, especially if two proteins carry out a similar function. It is easy to see how an algorithm like Random Forest may be bias towards the features used in classification and falsely identify a remote homolog, where the sequence alignment would not be so good and therefore not a homolog at all. On the contrary, not all protein groups or families are defined by a common ancestor. Antifreeze proteins for instance are a classic example of parallel evolution, in which a group of protein ancestors not related to each-other have evolved to converge on a functional purpose, and hence the proteins in a family of antifreeze proteins do not share aligned sequential homology. Random Forest would in this case be expected to discriminate between different protein groups that have evolved in parallel better than a machine learning method that uses multiple sequence alignments. This experiment uses antifreeze proteins as before, as well as transmembrane proteins. As already discussed, antifreeze proteins contain many different proteins evolved from multiple "protein ancestors", where transmembrane proteins evolves from a very small group of proteins and hence show both functional and structural homology.

Ligand gated ion channel or Ionotropic Glutamate receptors are part of the ion channel transmembrane clan (*CL0030*). They are synaptic receptors attached to the membranes of neuronal cells. Glutamate receptors are implicated in a number of neurological conditions. Their central role in the central nervous system is linked to a variety of neurodegenerative conditions and diseases such as autism, Parkinson's disease, Huntington's disease and multiple sclerosis. Transmembrane proteins are designed so that they allow different molecules to pass through their structure, and as such they have distinct structures not only from non-transmembrane proteins, but also from other transmembrane proteins. Various machine learning techniques have been applied to the classification of transmembrane proteins. Pasquier *et al* [35] used a neural network to classify transmembrane proteins from non-transmembrane proteins. Using only 11 different types of transmembrane

proteins were used to train the neural network and used a positive test set of 995 globular membrane proteins to test the NN. They falsely classified only 23 proteins, acheiveing an accuracy of 97%. Saha *et al* [35] used support vector machines to classify voltage-gated ion channel proteins, as well as classifying 4 sub-types of transmembrane proteins families, acheiveing accuracy of 82.89% and 96.89% respectively. The work in the previous chapter on random forest compares comparatively well, to these two papers, especially where acheiveing slightly higher accuracy than using the SVM approach.

The aim of this experiment is to show that Hidden Markov Models in the form of p-HMMs can classify transmembrane proteins (in this instance voltage gated ion channel proteins) from non-transmembrane proteins and other sub-types, typically up to a good threshold of 85%.

Three tests were devised for classification of various voltage gated ion (VGC) proteins:

- Classify VCG proteins from non-VGC proteins. 400 VGC proteins (300 potassium, 60 calcium and 60 sodium) were sourced from SWISSPROT and proteins with more than 90 percent sequence similarity were filtered out. 280 (200 potassium, 40 calcium and 40 sodium) were used as a training set for the p-HMM and the remaining 120 as a positive test set. 300 non-VGC proteins were also sourced from SWISSPROT as a negative test-set.

- Classify VGC subtype protein from non-VGC proteins. 44 seed sequences for Ligand-gated ion channel proteins in PFAM (ID PF00060) were used as a positive training set to train a p-HMM. The remaining 3228 ligand sequences in the PFAM ligand HMM family were used as a positive test set, and 1445 non-VGC proteins selected randomly from PFAM were used as negative test set.

- Classify VGC-subtypes from other VGC-subtypes The ligand p-HMM was used to classify ligand proteins from other VGC proteins. The

other two VGC subtypes used as negative test sets were Potassium-transmembrane ATPase A subunit proteins (ID PF03814, 2239 proteins) and Inward rectifier potassium channel (ID PF01007, 1452 proteins), as well as the same ligand test set in test 2 as a positive test set.

Each HMM model was trained using using the following process:

Step 1: Obtain seed sequences to build antifreeze profile HMM.

Step 2: Create a multiple sequence alignment of sequences T.

Step 3: Initialize a profile HMM using the length of the multiple sequence alignment D.

Step 4: Build profile HMM from the seed sequences.

Step 5: Estimate transition and emission parameters.

Step 6: Score the test sequences against the profile HMM.

Step 7: Repeat steps 4-6 until test scores converge and do not change. This is the fully trained model.

3 transmembrane (ligand, potassium, rectifier), and two antifreeze (antifreeze and antifreeze -like) models were created.

Table of results

Test	Classification accuracy
Trans vs non-Trans	80%
Ligand vs non-VGC	92.4%
Ligand vs Potassium/Rectifier	94.7%

The second test to discriminate between ligand transmembrane proteins and two other sub families of ion channel transmembranes yielded similar accuracy of 94.7% for positive ligand classification.

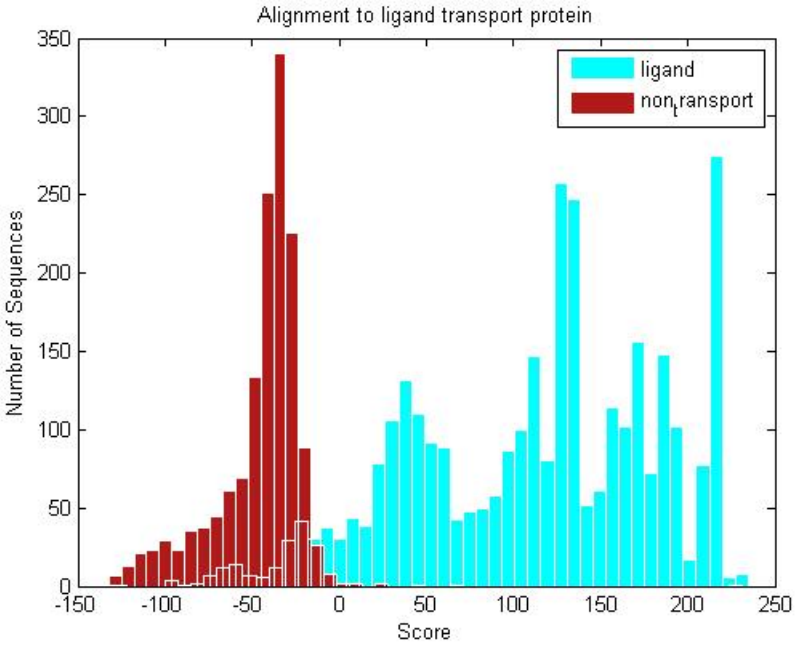


Figure 3.9: Ligand transmembranes and non-transmembrane proteins were scored against the ligand profile HMM. The distribution clearly shows the trained ligand HMM able to discriminate between the two groups of proteins.

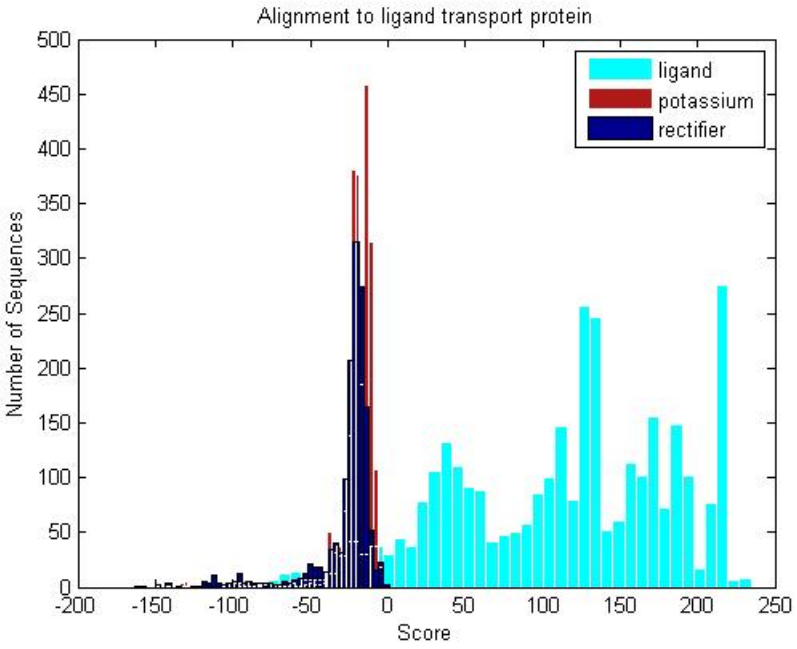


Figure 3.10: Three transmembrane sub-groups (ligand,potassium and rectifier) were scored against the ligand profile HMM, in which despite stark similarities in structure, the ligand HMM scores significantly higher for ligand proteins than the other transmembranes.

Antifreeze proteins (AFPs) are produced by organisms often inhabiting environments below freezing temperature, where such proteins can prevent the organisms cells from freezing. AFPs are present in numerous organisms, including bacteria, plants, vertebrates and invertebrates (split over 2605 species). Despite AFPs having more or less the same functional purposes, sequence and structure (28 different structures over 4935 different sequences) varies greatly, where classification on sequence alone does no discriminate AFP's from other proteins very well. Kandaswamy *et al* [24] show 83% prediction accuracy with random forest to classify antifreeze proteins from non-antifreeze proteins, giving higher accuracy than standard profile hidden Markov models. The following example aims to highlight the difficulties that are incurred when trying to classify antifreeze proteins using p-HMMs.

An antifreeze profile HMM was built from 169 seed antifreeze proteins from the PFAM database, where a test set of 4935 antifreeze proteins, 4935 fake proteins generated from amino acid distributions of the antifreeze proteins, and 5000 randomly selected proteins from the PFAM database were scored against the antifreeze HMM. A second test, taking 1927 "antifreeze-like" proteins from PFAM were also scored against the antifreeze HMM.

The antifreeze profile HMM easily classified antifreeze proteins, and discriminated against the fake proteins and randomly selected proteins. The "antifreeze-like" proteins however, scored within the same range as the antifreeze proteins themselves.

It is important to note in the fake protein example that using not only profile HMM's, but raw sequence alignment analysis in general would always produce a correct estimation, where random forest would not be able to discriminate between the fake proteins and antifreeze proteins based on amino acid frequency alone, as the fake sequence has used the same composition of amino acids.

While antifreeze proteins are known to contain above average amounts of certain functional groups and amino acids, the sequence structure is not

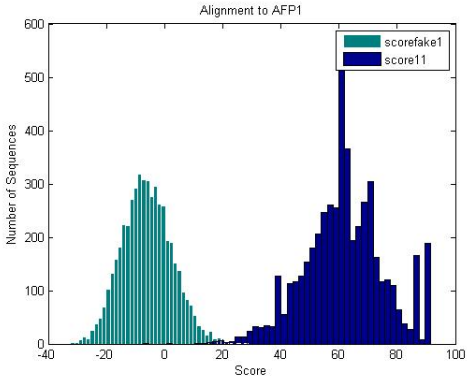


Figure 3.11: Fake proteins generated with the same amino acid frequency as the antifreeze proteins scored against the antifreeze HMM was significantly less than that of the antifreeze proteins

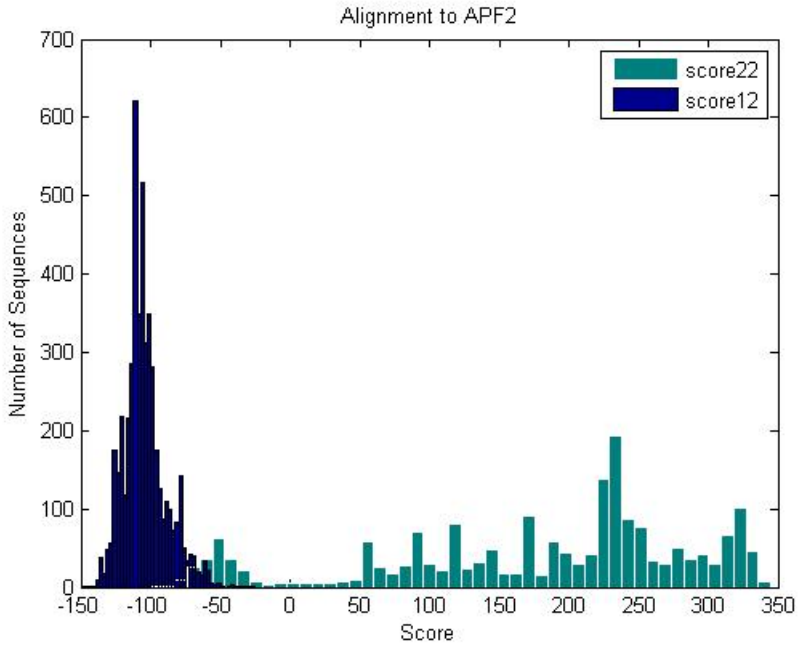


Figure 3.12: Antifreeze-like (green) and antifreeze proteins (blue) scored against an antifreeze-like HMM. Antifreeze proteins do not score well because antifreeze-like proteins have high variance in their sequence data, hence providing various other functional properties.

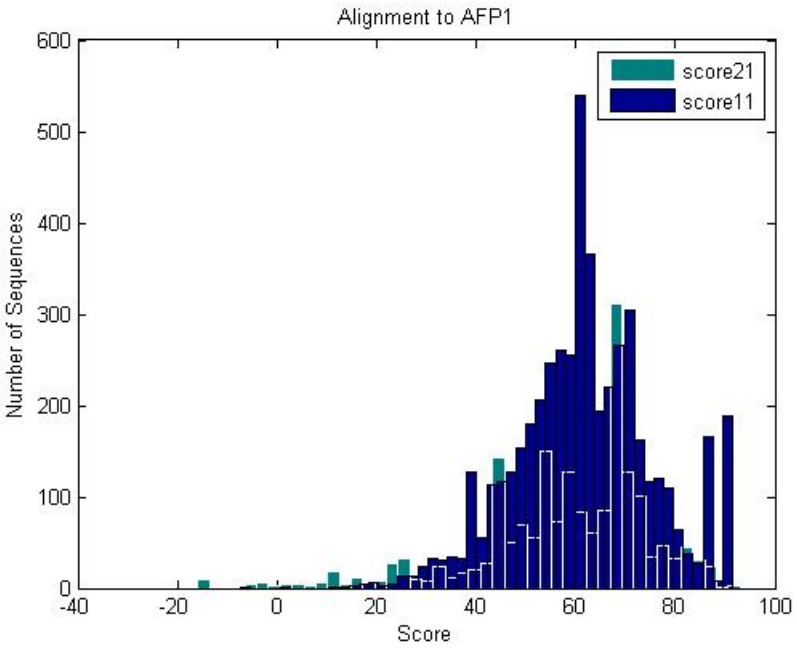


Figure 3.13: Antifreeze-like (green) and antifreeze proteins (blue) scored against an antifreeze HMM. Both score well against the HMM, where antifreeze-like proteins contain segments of sequence similarity to antifreeze proteins.

know for high homology within the family. This is reflected in the results from the HMM that find it difficult to distinguish between antifreeze and antifreeze-like proteins

It is clear that p-HMM's perform just as well as in the literature, and excel at picking out subtle differences in proteins of similar sequence structure. This is illustrated by the high classification accuracy obtained determining between different types of proteins within the same protein family (92% accuracy). p-HMM's are able to discriminate at this level because the alignment of proteins to a p-HMM scores the alignment based on conserved regions of sequence, and hence are able to track the subtle difference in proteins that occur over evolutionary processes. Based on the multiple sequence alignment to a p-HMM model it is possible to explore regions of the sequences that are conserved to see why the proteins are related on a functional or structural level.

It must be pointed out however that in an example such as antifreeze proteins where the proteins in the family are mainly based on functional purpose rather than sequence similarity, discriminating between sub-types does not work as well as the transmembrane proteins. This is not surprising given that the proteins have evolved from multiple different types of proteins to converge on a similar function. It would be necessary to used a technique that could use functional groups within the protein to classify them, such as random forest.

The scoring system from the p-HMM however are standardized by the fact training is undertaken by using a positive datasetset only. A negative dataset is not required, and the result of aligning a protein to a p-HMM is not relative to a negative dataset. This puts the alignment in firm context of what is the main goal of p-HMM's, and that is to determine if the protein belongs to a certain family based on the statistics of the multiple sequence alignment of sequences in that family. p-HMM's as well as any other technique that rely of sequence alignment is perhaps the only way of determining

true distant homologs without supplementing them with Phylogenetic data from evolutionary processes. It therefore has to be said that based on the nature of the training and scoring system, p-HMM's are the choice of technique to use when a standardized measure of what family a protein belongs to is needed.

3.9 Summary

Profile HMM's offer a way of generalizing protein sequences belonging to a certain family by producing a statistical model of the sequences when aligning all sequences in the family. They were introduced by [27] and further developed by [16] to provide a method of aligning multiple sequences for comparison. Proteins are able to be aligned to the profile HMM by the Viterbi algorithm in which the proteins are scored against the model by means of a log-odds ratio. Emissions and transitions of amino acids in specific positions of the sequence are scored, as well as deletions/insertions of amino acids from sequence to sequence within the family. This produces an accurate method of aligning sequences that effectively allows alignment to all of the proteins in the family. There is also no need to use a negative training set as the HMM is not strictly discriminatory as random forest or SVMs. The score is therefore an absolute prediction of how a protein might belong to a family, as opposed to being weighted by a negative dataset. What hasn't been included in the HMM experiment is the ability for the HMM to produce sequences based on the statistical features of the model. However this could be useful for protein synthesis, especially in drug discovery.

Many applications of HMM's focus on determining hidden states of a system, such as a loaded dice in a casino based on a sequence of rolls. HMM's can be used in this way in protein analysis, for example segmenting ares of sequence that are rich in certain functional groups, profile HMM's are a special case where the statistical model built from a family of sequences are

used to score a protein by comparing the amino acids sequentially compared to the model, where no hidden states are factored into this scoring system. Their specialty is in detecting remote protein homologs that have typically evolved away from an ancestor family of proteins, and as this experiment has shown, it is possible to distinguish between very similar groups of proteins such as transmembrane proteins.

Chapter 4

Discussion and future work

The aim of this thesis was to provide an insight of machine learning in the context of protein classification, in particular the random forest and hidden Markov model algorithms. Random Forests and HMMs take completely different approaches - random forest uses meta data extracted from protein sequences to split the data into user-defined classes, where HMMs build a statistical model from directly aligning protein sequences of known homology, and new sequences are then aligned the model. There is a large focus in bioinformatics on how various machine learning algorithms compare to eachother in terms of classification accuracy, but just as was illustrated in the differences between local and global alignments of sequences (Needleman-Wunsch and Smith-Waterman algorithms), each machine learning algorithm should also be viewed on it's individual merits and what they can offer.

Hidden Markov models not only classify proteins, but can determine underlying features such as frequency of certain amino acids in match states of a profile HMM, or by segmenting the model to determine hydrophobicity or isoelectric profiles. The fact that profile HMMs are built from directly aligning sequences to a model built by aligning *many* sequences provides a classification tool that specializes in finding the local segments of protein sequences that are conserved through evolution or show high homology, and

thus provide a way to find distant homologues that have diverged through evolutionary processes. They also take advantage of the statistical information obtained from proteins inserting amino acids into their sequence to achieve a new functional purpose, and when proteins divergence it is often difficult to see this unless inserts and deletions in data are accounted for, where two proteins clearly show they are of similar origin, however both proteins have their own amino acids inserted and deleted which only become apparent when aligning them in this way. Another aspect of HMMs is that a log-odds scoring system (converted into bit-scores for more readable purposes) provide a metric that is far more expansive than simple and discrete yes/no labels used in random forest. This can provide biologists a way of focussing on proteins which do not classify as well and explore through other methods to discriminate them against a family. Random forests on the other hand excel at discriminating between two groups of proteins and are not constrained to classifying on raw sequence data. The ability to extract and train on a vast array of features, ranging from simple frequencies and purely mathematical features to recognized physiochemical and functional properties of the proteins allows for customization that can tailor the process based on prior knowledge of the protein groups. The features used to best split the data are computed via the gini index (or permutation importance is an alternative method) and so provide an insight into what features were used to classify the data. This is an incredibly useful and intuitive feature of random forests and can not only be used to verify that important features *expected* to be found within certain families are part of the classifier, but also could potentially provide a platform for investigating features picked out by random forest that were not necessarily thought to be important.

This thesis implements both algorithms on transmembrane and antifreeze proteins through a series of different comparisons. Transmembrane proteins have typically evolved from very few ancestors and as such have retained high sequence and structural homology, designed to carry out very specific

tasks. Small nuances in localized parts of the sequences and catalyst sites are usually the source of difference between sub groups. Antifreeze proteins on the other hand are a classic example of parallel evolution where many groups of proteins have evolved convergentially and independently to perform the same function, namely controlling the thermal hysteresis of cells in an organism. Because of the parallel evolution, the sequences and structure are highly divergent within their own sub groups, but all the while demonstrating common functional and physiochemical features between them. As such there are proteins which purely act as antifreeze proteins (which are distinctly different between different types of organisms such as fish and insects), and antifreeze-like proteins where the proteins perform in the same way as antifreeze proteins, but may also contain sequence data that allows them to perform other functions as well, or functions which are now redundant (like the common Christmas tree containing huge amounts of junk DNA which isn't used any longer, but once performed some function in its evolutionary history). To really test the power of random forest and HMM, not only were transmembrane protein groups compared against non-transmembrane proteins, but this thesis tried to classify between type III antifreeze proteins and their homologous counterparts.

The results of using both techniques yielded accuracy achieved among many other techniques across various different protein groups (accuracy typically from 85% upwards). Both techniques coped well with classifying between transmembrane proteins and non-transmembrane proteins, as well as classifying between their sub groups. Random forest required a lot of features in the training and test datasets to achieve similar accuracy, including predicted secondary structure data which is time consuming to process. Random forest will also rely heavily on the negative training set to classify the data - something which requires a huge dataset that can represent "background" protein distributions. However, as was seen in the classification of antifreeze proteins from their homolgously similar counterparts, because random forest

can take into account secondary structure and functional properties as a feature vectors, classification between the two groups is possible, however due to the fact that the raw sequences have high variation even between proteins of the same sub group, profile HMMs struggle to tell the difference between the two sub groups when aligning both groups to a profile HMM of the antifreeze proteins. This is because both groups only small local segments of the protein that are similar, where antifreeze proteins do not contain certain segments of data found in flagellar basal-body an pilus assembly proteins, however viewed from the other way around the antifreeze-like proteins will contain more segments found in antifreeze proteins in order to perform their function just as antifreeze proteins do. It is interesting to note that the PFAM database which has defined the two sub families of the type III antifreeze proteins only discriminates well between the two when aligning both families to the antifreeze-like model, meaning that parts of the sequence and structure in the antifreeze-like proteins that are not used for antifreeze functionality play a major part in discriminating between the two. It is also interesting that random forest classifies between these two groups better on a whole when using the seed training data that was used specifically to build the type III antifreeze HMM in PFAM - which if we remember all families were built from HMM software.

A case could be made that random forest and profile HMMs can be used side by side given their methodological difference and outputs. Because profile-HMMs only require a positive dataset to both train and test on, they would often be the first port of call to ask the simple question "which family does this protein belong to?" as it picks up on sequence similarities which explicitly show how it could belong to a certain family. Random forest can then be used to look deeper into which features are important within a family. On the other hand, because some protein families are defined by functional purpose alone, such as antifreeze proteins, random forest can be tailored to look only at functional features to classify proteins. Once classified using

random forest, these proteins with their functional importance known, can be alingined against profile-HMMs models and potentially identify protein families with similar functional purpose, something which drug companies would benefit from knowing. The main realization from this thesis is that, it is obviously important to have techniques such as random forest and HMM's that are known to classify proteins well, however it important to know *why* certain proteins belong to certain families. Both techniques in this thesis demonstrate that and go some way to refuting the idea that machine learning techniques are merely black box processes giving no insight into what was important when classifying the data.

Future work will include using predicted secondary structure as the primary input to profile-HMM's. If HMMs can built on secondary structure, it would be possible to enhance search capability for proteins that share similar structure. Just as aligning to profile-HMMs is effectively like aligning to 100's of sequences rather than just one, it would be possible to search for structures based on families rather than individual proteins which be vary within it's own family. It would be much quicker to align to a model, find a homologous family in terms of structure and take a small subset of proteins to use in experiments. Not only does this speed up an otherwise exhaustive search, but it is also more powerful. Considering the most time consuming part of this thesis was to get predicted secondary structure of the proteins, having a profile-HMM of secondary structures and the ability to generate sequences from a model, as long as the structure is well defined and recognized to belong to the intended family, secondary structure generation could be possible to help populate secondary structure training sets for random forest. In the interest of trying to combine the merits of each technique, it would also be entirely feasible to use the score of aligning a proteins to a profile-HMM as input to a random forest,especially if there are multiple competing families to discriminate between. Perhaps a more ambitious metric to take from p-HMMs into random forests would be to segment the align-

ments to a HMM into sections such as regions of high conservation within the alignment, or even something as simple as percentiles of the alignment and how well each percentile scores as a local alignment. This would allow a way to inject ordered sequence data into the random forest, something which random forest lacks due to its reliance on meta data.

Bibliography

[1] Ivan A Adzhubei, Steffen Schmidt, Leonid Peshkin, Vasily E Ramensky, Anna Gerasimova, Peer Bork, Alexey S Kondrashov, and Shamil R Sunyaev. A method and server for predicting damaging missense mutations. *Nature methods*, 7(4):248–249, 2010.

[2] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[3] Pierre Baldi. Substitution matrices and hidden markov models. *Journal of Computational Biology*, 2(3):487–491, 1995.

[4] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl 1):D138–D141, 2004.

[5] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic acids research*, 38(suppl 1):D46–D51, 2010.

[6] James U Bowie. Helix packing in membrane proteins. *Journal of molecular biology*, 272(5):780–789, 1997.

[7] John Boyle. Seqexpress: desktop analysis and visualization tool for gene expression experiments. *Bioinformatics*, 20(10):1649–1650, 2004.

[8] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[10] Xue-Wen Chen and Mei Liu. Prediction of protein–protein interactions using random decision forest framework. *Bioinformatics*, 21(24):4394–4400, 2005.

[11] Peter Y Chou and Gerald D Fasman. Conformational parameters for amino acids in helical, β-sheet, and random coil regions calculated from proteins. *Biochemistry*, 13(2):211–222, 1974.

[12] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[13] A Criminisi, J Shotton, and E Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, 5(6):12, 2011.

[14] Antonio Criminisi, Jamie Shotton, Duncan Robertson, and Ender Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. pages 106–117, 2011.

[15] Scott W Doniger, Nathan Salomonis, Kam D Dahlquist, Karen Vranizan, Steven C Lawlor, Bruce R Conklin, et al. Mappfinder: using gene ontology and genmapp to create a global gene-expression profile from microarray data. *Genome biol*, 4(1):R7, 2003.

[16] Sean R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[17] Sean R Eddy. {HMMER: Profile hidden Markov models for biological sequence analysis}. 2001.

[18] Sean R Eddy et al. Multiple alignment using hidden markov models. 3:114–120, 1995.

[19] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.

[20] Peng-Mian Feng, Hui Ding, Wei Chen, and Hao Lin. Naïve bayes classifier with feature selection to identify phage virion proteins. *Computational and mathematical methods in medicine*, 2013, 2013.

[21] André Fujita, João R Sato, Humberto M Garay-Malpartida, Rui Yamaguchi, Satoru Miyano, Mari C Sogayar, and Carlos E Ferreira. Modeling gene expression regulatory networks with the sparse vector autoregressive model. *BMC Systems Biology*, 1(1):39, 2007.

[22] Steven Henikoff. Scores for sequence searches and alignments. *Current opinion in structural biology*, 6(3):353–360, 1996.

[23] McGuffin LJ Jones DT, Bryson K. The psipred protein structure prediction server. *Bioinformatics*, 4:404–405, 2000.

[24] Krishna Kumar Kandaswamy, Kuo-Chen Chou, Thomas Martinetz, Steffen Möller, PN Suganthan, S Sridharan, and Ganesan Pugalenthi. Afp-pred: A random forest approach for predicting antifreeze proteins from sequence-derived properties. *Journal of theoretical biology*, 270(1):56–62, 2011.

[25] Kanehisa and Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28:27–30, 2000.

[26] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[27] Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of molecular biology*, 235(5):1501–1531, 1994.

[28] Jack Kyte and Russell F Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, 157(1):105–132, 1982.

[29] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575. World Scientific, 2002.

[30] Camilo Mora, Derek P Tittensor, Sina Adl, Alastair GB Simpson, and Boris Worm. How many species are there on earth and in the ocean? *PLoS biology*, 9(8):e1001127, 2011.

[31] John Moult, Tim Hubbard, Krzysztof Fidelis, and Jan T Pedersen. Critical assessment of methods of protein structure prediction (casp): round iii. *Proteins: Structure, Function, and Bioinformatics*, 37(S3):2–6, 1999.

[32] Abhigyan Nath, Radha Chaube, and S Karthikeyan. An insight into the molecular basis for convergent evolution in fish antifreeze proteins. *Computers in biology and medicine*, 2013.

[33] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[34] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[35] Claude Pasquier and SJ Hamodrakas. An hierarchical artificial neural network system for the classification of transmembrane proteins. *Protein engineering*, 12(8):631–634, 1999.

[36] William R Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11(3):635–650, 1991.

[37] Ganesan Pugalenthi, K Krishna Kumar, PN Suganthan, and Rajeev Gangal. Identification of catalytic residues from protein structure using support vector machine with sequence and structural features. *Biochemical and biophysical research communications*, 367(3):630–634, 2008.

[38] Yanjun Qi, Judith Klein-Seetharaman, Ziv Bar-Joseph, Yanjun Qi, and Ziv Bar-joseph. Random forest similarity for protein-protein interaction prediction. In *Pac Symp Biocomput*. Citeseer, 2005.

[39] Bin Qian and Richard A Goldstein. Detecting distant homologs using phylogenetic tree-based hmms. *Proteins: Structure, Function, and Bioinformatics*, 52(3):446–453, 2003.

[40] David E Rumelhart, Bernard Widrow, and Michael A Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, 1994.

[41] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[42] Milton H Saier, Can V Tran, and Ravi D Barabote. Tcdb: the transporter classification database for membrane transport protein analyses and information. *Nucleic acids research*, 34(suppl 1):D181–D186, 2006.

[43] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. pages 1–8, 2008.

[44] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[45] Nayanah Siva. 1000 genomes project. *Nature biotechnology*, 26(3):256–256, 2008.

[46] Erik LL Sonnhammer, Sean R Eddy, Richard Durbin, et al. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins-Structure Function and Genetics*, 28(3):405–420, 1997.

[47] Guenter Stoesser, Wendy Baker, Alexandra van den Broek, Evelyn Camon, Maria Garcia-Pastor, Carola Kanz, Tamara Kulikova, Rasko Leinonen, Quan Lin, Vincent Lombard, et al. The embl nucleotide sequence database. *Nucleic acids research*, 30(1):21–26, 2002.

[48] Paul D Thomas, Michael J Campbell, Anish Kejariwal, Huaiyu Mi, Brian Karlak, Robin Daverman, Karen Diemer, Anushya Muruganujan, and Apurva Narechania. Panther: a library of protein families and subfamilies indexed by function. *Genome research*, 13(9):2129–2141, 2003.

[49] Stephanie Unterreitmeier, Angelika Fuchs, Teresa Schäffler, Roland G Heym, Dmitrij Frishman, and Dieter Langosch. Phenylalanine promotes interaction of transmembrane domains¡ i¿ via¡/i¿ gxxxg motifs. *Journal of molecular biology*, 374(3):705–718, 2007.

[50] Cathy Wu, George Whitson, Jerry McLarty, Adisorn Ermongkonchai, and Tzu-Chung Chang. Protein classification artificial neural system. *Protein Science*, 1(5):667–677, 1992.

[51] Pei Yin, Antonio Criminisi, John Winn, and M Essa. Tree-based classifiers for bilayer video segmentation. pages 1–8, 2007.