Contents

Abstract

This report compares a number of distance based learning classifiers for the task of assigning SCOP family level classification to a protein domain, given that the domain has already been assigned a superfamily SCOP classification.

The report begins by giving a basic introduction to the nature of biological sequences, delving into a little biology and biochemistry to give the computer scientist a quick primer on the relevant background to the problem domain. This was useful to the author of this report and it is hoped to be useful to others. It then goes on to frame the problem as a task that computer science is ideally suited to tackling, the report presents a narrative that ties the historical beginnings of computer science and the theory that has been developed in machine learning with the current research into protein classifications. It briefly describes many of the current and historic methods for machine learning in general and points out some key results in relation to computational biology. It then describes The Superfamily database, with information given on its development and the techniques currently used to classify protein sequences within it. Other state of the art protein classifying techniques are discussed.

Four different distance based methods are compared, Nearest Neighbour, Nearest Average, Nearest Centroid and Nearest Medoid. Nearest Neighbour was found to be the most effective and a discussion of the advantages of this methods is included.

This project is submitted as part of Master of Science Degree in Advance Computing – Machine Learning and Data Mining from The University of Bristol. The work will be completed in Summer and Autumn of 2012 by Sam Neaves BSc(Hons) Internet Communication Systems.

Introduction

Ever since Charles Darwin and Alfred Wallace first put forward the theory of evolution in 1859 (Darwin, 1859), the human race has been engaged in the enormous task of mapping out the tree of life. For a time the endeavour was hampered by the lack of understanding of genetics, it was not until Gregory Mendel's work on heredity in pea plants that the laws of heredity began to be unpicked (Mendel, 1866). Mendel's work was not widely recognised for decades after his death, it was rediscovered at the end of the century by Hugo de Vries (Vries, 1889) and Carl Correns (Correns, 1900), this understanding of the unit of natural selection, 'the gene' greatly enhanced the knowledge base, but the race was on to understand the actual mechanism of heredity. In 1953 James Watson and Francis Crick building on the work of others including Maurice Wilkins, Rosalind Franklin, A.R. Stokes, and H.R. Wilson, published in there seminal paper "A Structure for Deoxyribose Nucleic Acid" a description of the nature of DNA and for the first time how phenotypes (traits or observable characteristics) were passed on and varied was understood (Crick & Watson, 1953). The double helix DNA Molecule has become the modern symbol of biology.

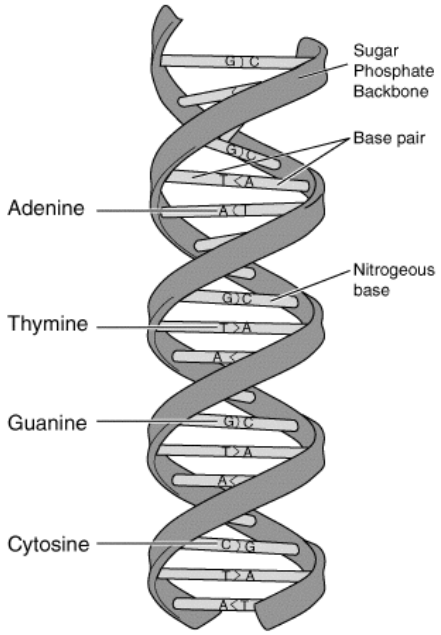


Figure 1 (WikiCommons, 2011)

In general a molecule is made up of a number of bonded atoms; atoms are the base elements that all matter in the universe is made up of. We now know that there are smaller more fundamental particles that make up atoms, including electrons, protons and neutrons, the chemical properties of atoms can be derived from the number and arrangement of these particles. In the centre of atoms are the protons and neutrons which together form the nucleus, around the nucleus are the orbital's (Not orbits) of electrons. These are best described using probability density functions. These functions can be divided up into shells, the first shell 'wants' to have two electrons, further shells 'want' to have eight electrons. The electrons in the outer most shell are the valance electrons (There are some exceptions in transitional metals where valance electrons can be found in inner shells). These valance electrons of an atom are available to be shared with other atoms to form covalent bonds and therefore molecules. Atoms where the outer shells are full are generally inert and do not form bonds, atoms close to a full shell are the most chemically reactive. If a molecule is formed then the electrons are shared between two atoms. Atoms can chain up to form larger molecules. (Jones, 2005)

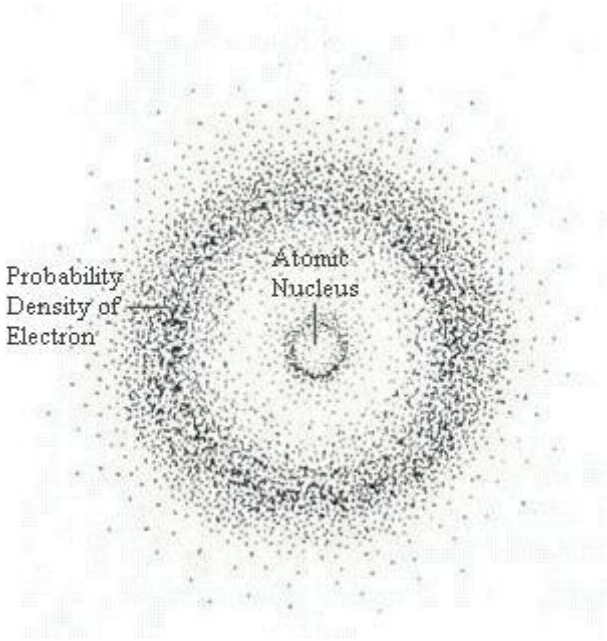


Figure 2 (Universe Review, 2011)

DNA molecules can either be found floating in cells in the case of prokaryotes (Two domains of life; the bacteria and the archaea) or in the nucleus of cells in the case of eukaryotes (All other domains of life including plants and animals). The DNA molecule is a large molecule formed of two long chains called polymers, made of the nucleotide molecules adenine and guanine, as well as thymine and cytosine. Adenine pairs with thymine, guanine pairs with cytosine. These are called base pairs Therefore a DNA sequence can be spelt out as a string, for example ATGCTGATCTTGGCCATCAATG which of course implies there will be a corresponding chain which in this case would be CATTGATGGCCAAGATCAGCAT along the other backbone of the helix. Therefore a DNA sequence can be read in two different directions. The Human genome has been shown to have three billion base pairs. Some parts of a DNA sequence codes for proteins, other parts of it have structural purposes and still more appears to be effectively 'junk'. (Ridley, 2000)

In segments of DNA that codes for proteins, the sequences are first transcribed to an RNA molecule which is similar to DNA but, adenine pairs with uracil instead of thymine. RNA forms a single string not a helical structure. So called Messenger RNA leaves a cell and is finally transcribed by ribosomes into proteins. Proteins are organic compounds; they have carbon atoms and make covalent bonds, C-C or C-H. If a compound does not have these bonds then it is not organic, proteins can generally be classified into structural or functional. Of the functional proteins some are regulatory known as hormones, others are transporters and some are catalysts known as enzymes. Antibodies are example proteins. Proteins are made up of smaller compounds known as amino acids, amino acids join together in long chains to form proteins. All amino acids (And therefore proteins) are composed of carbon, hydrogen, oxygen and nitrogen, some have iron, sulphur and phosphorous in addition. (Clayden, Greeves, Warren, & Wothers, 2001)

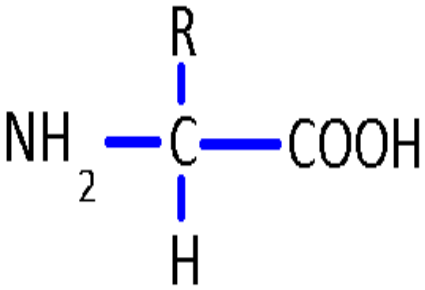


Figure 3 Amino Acid

The 4th covalent bond R represents a variable side chain. At one end of the protein will be a free NH2 molecule and at the other a COOH, these extremities are called the N and C terminus. The R variants lead to all 20 possible amino acids.

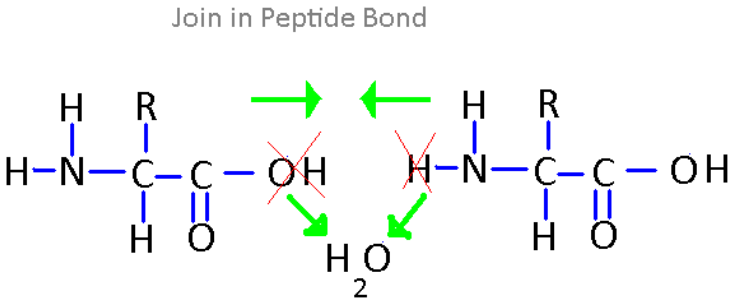


Figure 4 Peptide Bond

Two amino acids will join together in a process called dehydration synthesis. They form a peptide bond. The Messenger RNA molecule acts as a template for Transfer RNA, the amino acids want to bond with the right bit of Messenger RNA. The Amino acids pull up and then bond together, the MRNA detaches and the sequences folds up. The proteins twist or fold to make unique shapes, which are important to the function of a protein.

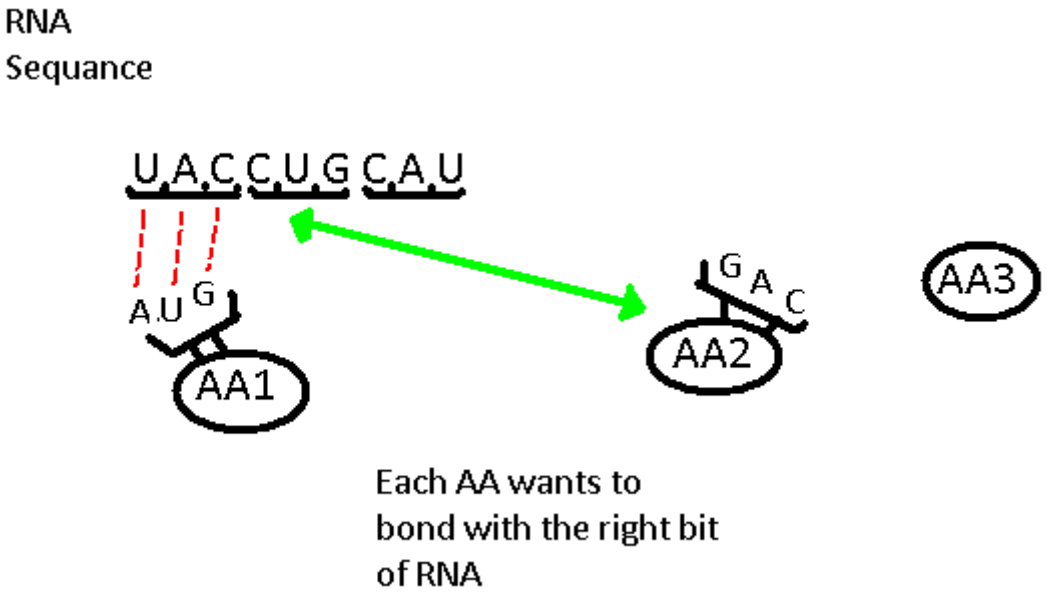


Figure 5 Amino Acids Floating

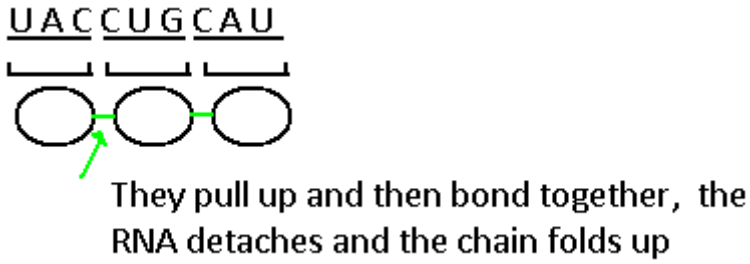


Figure 6 Amino Acid Joining up

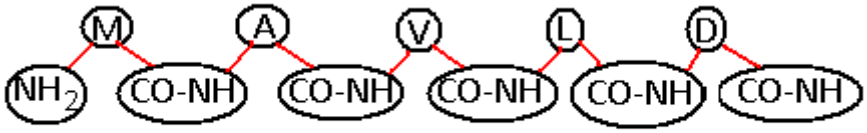


Figure 7 A short protein sequence

Three base pairs of DNA make up a codon which codes an amino acid. For example Insulin is 50 amino acids 50 * 3 = 150 pairs to code for insulin. This can be described as a gene. The original word allele which Mendal coined is now used to describe a variant of a gene.

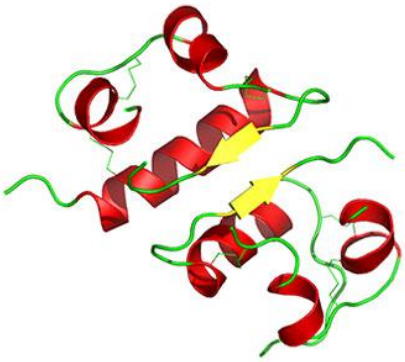


Figure 8 Insulin (Harrison, 2011)

Protein sequences that are similar have been shown to normally have similar folding patterns and therefore functions. It is of course more reliable to perform experiments to determine the structure and function of a biological molecule. However the sheer number of proteins makes this an impractical proposition. (Baldi, 2001).

Wetlaufer first suggested that protein sequences be divided into protein domains; these domains form independent stable three dimensional structures when they are folded. Each of these structures perform different functions in the running of a cell. They are often seen as the base units of evolution, they are the Lego bricks of biology. A domain can occur in one or many different proteins. A typical domain can have a length from 25 to over 500 amino acids.

The forces of natural selection are slow and as the giraffe's laryngeal nerve (As well as all other mammals) illustrates it is unable to invent '*de novo'* nature tinkers up the gradual slope to a peak of 'mount improbable' rather than leaping up sheer cliff faces. The nerve begins in the brain and travels to the voice box, via the heart, which in the case of the giraffe is a needless diversion of over 10ft! (Dawkins, 1997). Fortunately this curious property of evolution leads to detectable homology between any two DNA sequences. As DNA replicates a number of changes may take place, including substitution, insertion and deletion of sequence segments. (Clayden, Greeves, Warren, & Wothers, 2001).

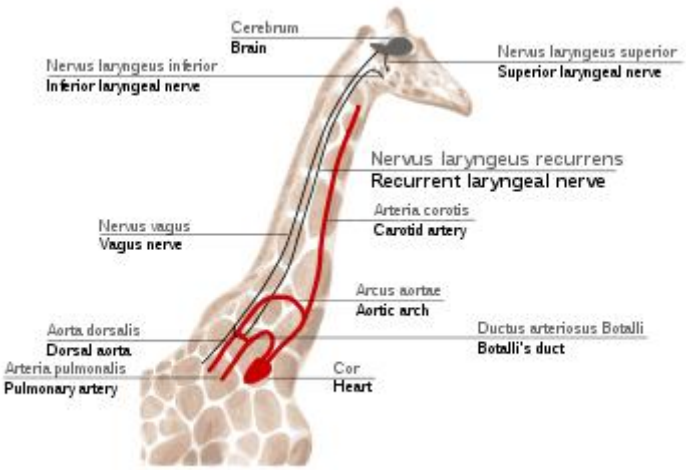


Figure 9 Evidence for gradual change in evolution, the laryngeal nerve (Source, 2011)

This leads to many of the problems in genomic analysis being essentially statistical. The forces of evolution; natural selection, chaotic random mutations and genetic drift create a large amount of noise in the data. One of the challenges of genomic analysis is to cut through this noise to detect the signal of common evolutionary heritage. (Baldi, 2001)

The first complete description of an organisms DNA was transcribed in 1977 by Fred Sanger, The Bacteriophage fX174 with 5,386 bases (Sanger, 1977). This was labelled a genome and the age of genomics was well and truly here, since then much progress has been made, a huge endeavour was undertaken to sequence the first full human genome, with many different teams racing to be first. The project was immense and many doubted it could be achievable and if it was even achievable many worried that the knowledge would in some way be a bad thing, with companies patenting genes and other predicted ethical calamities, the majority of people however saw great promise in this knowledge to fight and treat genetic diseases such as Huntington's disease, as well as the greater enterprise of sequencing all of life, to have powerful agricultural uses to increase crop yields and perhaps even aesthetic desires towards life forms including ourselves. Eventually in 2001 Craig Venters team announced that they had fully sequenced the human genome. (Venter, 2001) The 'shot gun' technologies that were used to do the sequencing had dramatically reduced the cost, and the speed of sequencing increased spectacularly, in fact it seemed to be faster even than the famous Moore's law of Gordon Moore of IBM, which has held in the world of computing for 50 years, Moors laws states that the number of transistors that can be placed on an integrated circuit doubles approximately every two years. The cost of DNA sequencing a human fell from $3billion dollars to under $5000 in 6 years. These efficiency increases have made biology into a science exploding with data, literally billions and billions of coded letters are being sequenced each year. It is the task of modern biologists to make sense of all this data and turn it into useful knowledge.

One of the innovations that begun development in 1995 when the tidal wave of biological data was just beginning to form was an effort to organise and catalogue protein data. This was setup by team at Cambridge university led by Muzin (Murzin AG, 1995) they set out The Structural Classifications Of Proteins (SCOP) database. The raw data for this is taken from The Protein Bank . They set out a hierarchical classification of proteins first into class, then fold, then superfamily, then family and finally domain. Proteins grouped at the superfamily level have similar shapes, but have little sequence similarity. Proteins having the same shape and some similarity of sequence are placed in families. It is assumed that they have a closer common ancestor and that the domains will have similar functions.

Cataloguing, labelling and understanding all of this data would have been an impossible task to even consider without computers, the sequences form an effective code that guard their secrets jealously fortunately there are many scientists and technology engineers working hard to work out methods and routines that allows us to decode life itself.

Codes and computer science

Detail from a photograph of World War I cryptographers trained by William and Elizebeth Friedman, Aurora, Illinois, early 1918. By facing either forward or sideways, the soldiers formed a coded phrase utilizing Francis Bacon's biliteral cipher. The intended message was the Baconian motto "Knowledge is power," but there were insufficient people to complete the *r* (and the *w* was compromised by one soldier looking the wrong way). (Sherman, 2011)

Figure 10

Code breaking has been central to the development of computer science, the above picture is an example of 'more than meets the eye' or stenography, the art of concealing messages in anything, The image was composed by William Friedman and he was inspired by the biliteral cipher of Sir Francis Bacon the farther and martyr to the scientific method. The a's and b's of Bacon's cipher were a form of binary encoding. (Sherman, 2011).

Binary encoding had first been properly explored by George Boole in the 1800's when he brought together the ideas of logic; truth, falseness and deduction, with mathematics' ones and zeros, to describe what is now called Boolean logic. (MacHale, 1985).

In the 1930's two disconnected men on either side of the Atlantic, Alan Turing and Claude Shannon were effectively building the foundations of information science; in 1937 Shannon wrote a master's thesis applying Boolean logic to electrical circuit design. (Shannon, 1937) Meanwhile Turing had proposed the idea of universal computing machines (Now known as Turing machines) in which any algorithm could be processed; this was in response to Entscheidungs problem, a kind of decision problem. In 1936 building on the mathematician Kurt Godel's incompleteness work which showed that some mathematical problems cannot be computed, Turing proved that some algorithms cannot be known if they will halt or not. (Turing, 1936). During the war both Turing and Shannon were conscripted to work on cryptology for their respective governments, attempting to break the axis powers encrypted codes, Turing at Bletchley Park in the United Kingdom and Shannon at The Bell laboratories in the United States. In 1942 Turing was sent on a mission to The Bell laboratories to share the work the British had been undertaking on decryption, the two finally met and Turing shared with Shannon his paper on universal computing machines. The work was highly complementary to Shannon's. (Alcatel-Lucent, 2006). As the war drew to a close In 1945 Shannon produced some of the first work on detecting signals from noisy data streams, this work had been prompted from work on fire control systems. He followed this in 1948 with the paper 'A Mathematical Theory of Communication' in which he introduced the ideas of information entropy (a measure for the uncertainty in a message) (Shannon, 1948.). After the war Turing added to the work he had produced in his code breaking efforts to manufacture an automatic code breaking machine to try and build the first universal computer. A number of machines can claim to be the first successful computer including the never completed machine proposed by Charles Babbage, Turing worked in Manchester on the ACE Machine, his work was hampered by war time secrecy rules but the team at Manchester eventually completed the Manchester 1 in 1949 which used the John von Neumann architecture; that

is using the same memory both to store programs and data. In 1950 Turing wrote a paper 'Computing machinery and intelligence' in which Turing addressed the problem of artificial intelligence (AI) (Turing, 1950). He proposed an experiment which became known as the Turing test, an attempt to define a standard for a machine to be called 'intelligent'. In the early 50's both Shannon and Turing separately worked on computer programs to play chess in some of the first attempts at programming software to behave in an intelligent manner. Turing probably frustrated at the restrictions on him drifted into different fields including computing theories of biology, until his untimely and awful convection for indecency and acts of homosexuality, which at the time was illegal. He was forced to undergo hormone treatments which likely lead to his subsequent tragic suicide. (Hodges, 1992).

As well as his chess work Shannon contributed another core first in AI research with his building of his maze navigating mechanical mouse, which can be said to be the one of the first examples of Machine Learning. (Alcatel-Lucent, 2006) Shannon's contributions did not end there! As he also began to apply his ideas of information entropy to natural language problems in his paper 'Prediction and Entropy of Printed English' this was the beginning of the research into Natural Language Processing (NLP) and computational linguistics (Shannon, 1951). Other developments in information theory came from the linguist Noam Chomsky in 1957, when he first described the Chomsky hierarchy of formal grammas. (Chomsky, 1957) This was extensively used in the early development of computer programming languages and later grammars were also applied to biological sequence analysis (Durbin & etal, 1998).

Shannon's chess work was also important in the field of game theory, which had really got started with the work of John von Newman's, Shannon is said to have made a lot of money in casinos in the 1950's with these mathematical ideas, many economists also took up the ideas and applied game theory to the worlds of business and markets. Game theory also became useful to the social scientists and much research was carried out in the second half of the 20th century. The biologist John Maynard Smith first applied some of the game theoretic ideas to evolutionary biology and ecology (Smith J. M., 1982). As well as biologist being inspired by computer scientists, the opposite was also true, with computer scientists being inspired by biology.

Neural Networks

As early as 1943 McCulloch and Pitts had proposed a model of neuron that corresponds to the now well known perceptron algorithm (McCulloch & Pitts, 1943) . Artificial Neural Networks (ANN) are built out of densely connected simple units, where each unit takes a number of inputs and outputs a single number, they can be used to find real valued, discrete valued and vector valued functions from examples. Biological research had shown that the brain must function in a massively parallel manner, ANN attempt to model this property. ANN learning has been shown to be an appropriate tool for learning when training data is noisy, typically from sensors such as microphones and cameras. As early as 1993 ANN was employed by Pomerleau's team to drive a car on public highways. (Mitchell, 1997)

The first functional ANN's were developed in the 1959 starting with the single layer perception models invented by Frank Rosenblatt (Rosenblatt, 1959) and further investigated by Widrow and Hoff in the 60's. (Widrow & Hoff, 1960) A perceptron works by inputting vectors of real numbers; it performs a calculation to combine these inputs and then out puts a Boolean value.

$$o(x_1, \dots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

Equation 1

Where w_i is a real valued constant or weight that determines the contribution of x_i . The threshold that must be surpassed for the perceptron to output a one is indicated by w_0

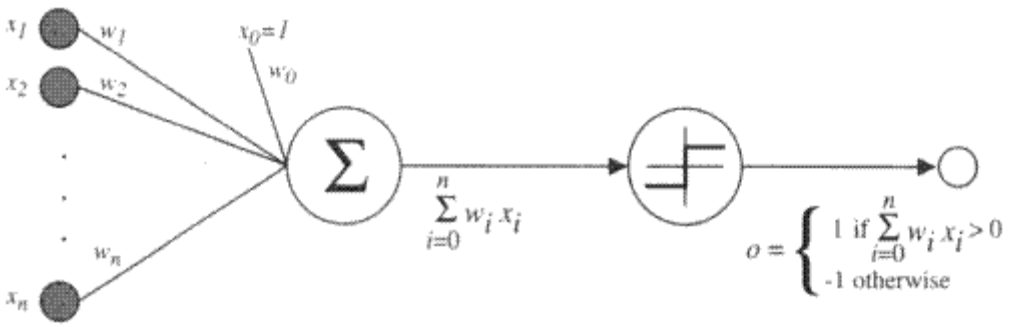


Figure 11

Therefore perceptrons can be used to represent most Boolean functions for example the AND function would be modelled by a perceptron with $w_0 = -0.8$ and $w_1 = w_2 = 0.5$ (Mitchell, 1997)

However it became apparent that Boolean functions such as exclusive or(one or the other but not both: XOR) could not be computed using single layer perceptrons as they are only able to represent linear decision surfaces and ANN fell out of fashion. The second graph here shows an XOR learning task, i.e. it is a learning task that is non-linearly separable.

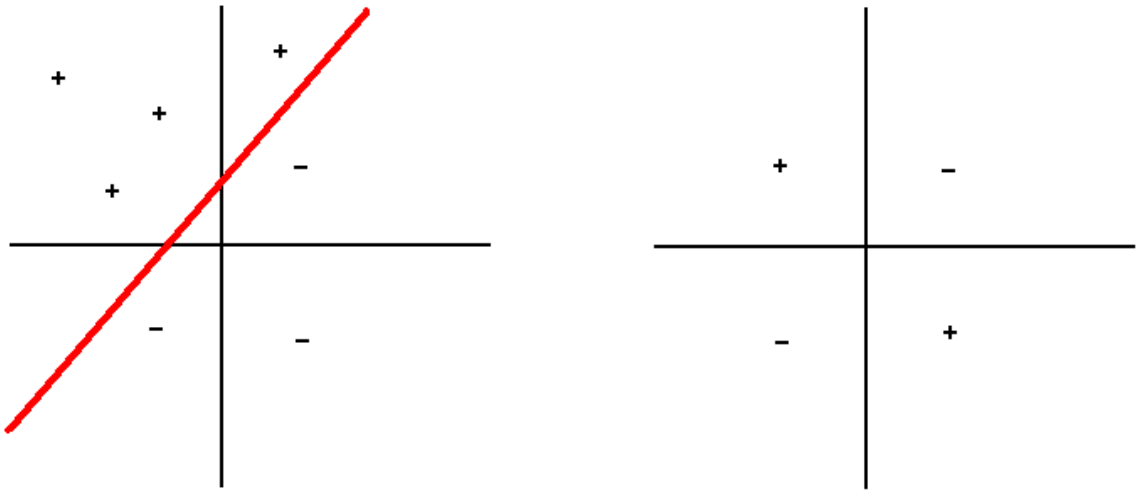


Figure 12

In a learning task, starting with an individual perceptron the aim is to learn the appropriate weight vector that delivers the correct +-1 output for each training example. One way to do this is called the perceptron training rule, which begins with random weights then iteratively modify the weights until it correctly classifies the example. This is then scaled to add many perceptrons; as long as the data is linearly separable this rule can be proven to converge in a finite number of steps to a vector that correctly classifies all training examples. (Minsky & Papert, 1969) This method converges on the thresholded part of the perceptron.

Another approach is gradient decent or delta rule, this converges on the unthresholded part of perceptron, i.e. the first stage of a perceptron without the threshold weight w_0. It will converge (perhaps requiring unbounded time) even if the examples are non-linearly separable.

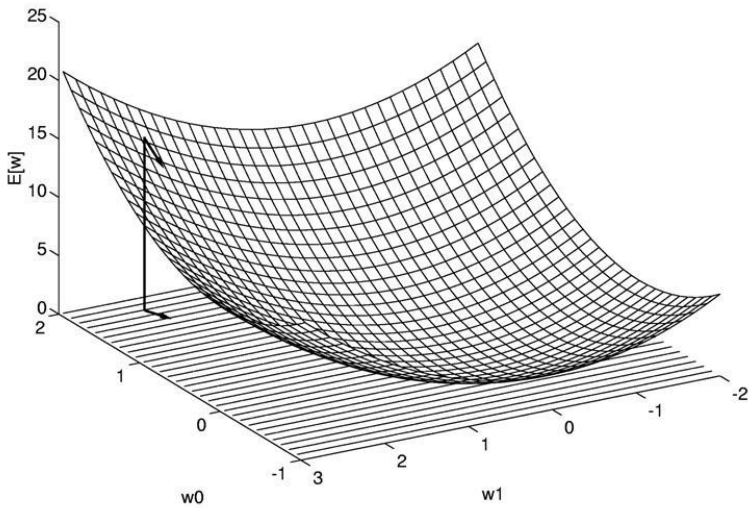


Figure 13

A function is used to calculate the steepest decent along the hypothesis surface using a derivative E with respect to each component of the vector $\vec{w}$. The steps need to be small in order not to leap over a local minimum. (Mitchell, 1997)

After the hiatus in ANN research throughout the 70's work resurged in the mid 1980's with the invention of Backpropagation algorithm in 1986, this uses a gradient decent approach (which was shown to scale to multi layer neural networks) to find locally optimum hypotheses. It is possible to represent information rich highly non linear decision surfaces. (Mitchell, 1997) One application to biological sequence analysis was to identify signal peptides. (Nielsen, Engelbrecht, Brunak, & von-Heijne, 1997)

A major difference to single layer networks is that the surface will not be a single minimum parabolic error surface (as depicted in the above diagram), and so the function may only find a local rather than global minimum.

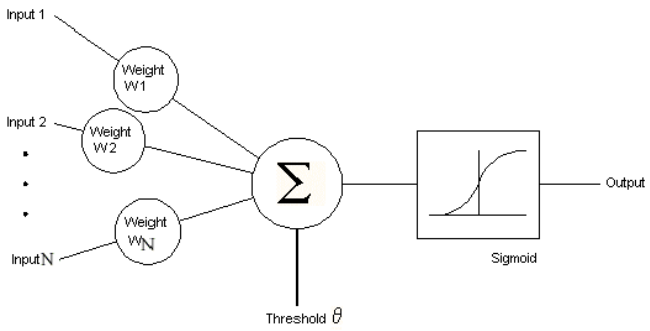


Figure 14

Neural networks have the advantages that long training times are generally acceptable as ANN then have very fast evaluations of learned target functions. A disadvantage of ANN's is that the functions composed can be difficult for humans to understand. (Mitchell, 1997). One application of neural networks to biological sequences was Chou and Zhang in 1995. (Chou & Zhang, 1995).

A different learning method: Decisions trees

Decisions tree learning began in 1966 with Hunt, Friedman and Breiman's work resulting in the cart system and Quinlans ID3 system. (Hunt, 1966), (Quinlan, 1986). Decision trees today are practical methods for inductive inference. They are robust to noisy data and capable of learning disjunctive expressions, they approximate discrete valued functions.

When a tree has been learnt it can be re represented as a set of 'if then' rules which aid human understanding. The idea behind a decision tree is to sort down a branch of a tree from the root which is normally depicted at the top of a diagram. Each branching point or node tests an example for a particular property. The lower terminals of the tree are called leaves and represent the classification of an example. A tree can be said to represent a disjunction of conjunctions of constraints on the attributes of instances. (Mitchell, 1997).

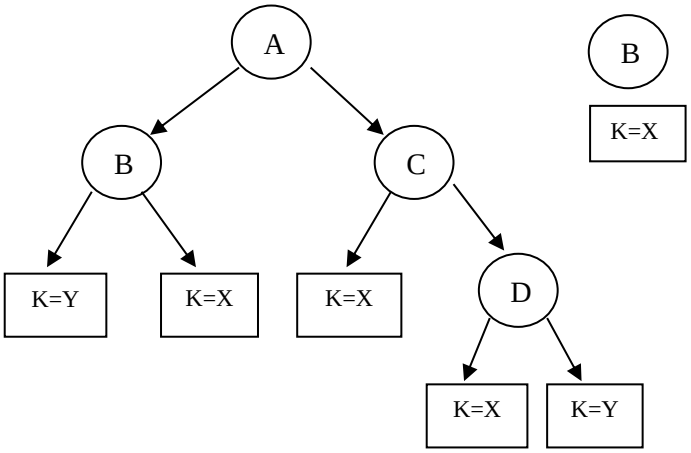


Figure 15

Decision trees are easily used when instances are represented by attribute value pairs or where each attribute can take on a small number of distinct values. E.G. Sex -> {M,F}. However it is possible to handle the input of real valued attributes as well. A decision tree requires the target function to have discrete output values; generally they are not suitable for out putting real values. Decisions trees are normally robust to errors and missing values, a number of techniques have been developed to overcome these scenarios. (Mitchell, 1997).

The central design decision when building a decision tree is the order in which attributes are tested to allow the most efficient classifications of examples. Most standard decisions tree algorithm beginning with Id3, and its successor C4.5 and higher versions use the principles of information entropy laid down by Shannon. (Shannon, 1948.)

If the target attribute can take *c* different values, then the entropy *S* relative to this *c*-wise classification is defined:

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

Equation 2

Where p_i is the proportion of *S* belonging to class *i*. This returns entropy measured in bits. This allows us to use a measure called information gain, which is the expected reduction in entropy to decide which attribute to chose next in building the decision tree.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Equation 3

Where *Values(A)* is the set of all possible values of an attribute A, and S_v is the subset of S for which attribute A has value *v* (i.e., $S_v = \{s \, \epsilon \, S | A(s) = v\}$). The information gain is recalculated for each branching point in the tree until each branch ends with a classification i.e. the entropy =0.

This algorithm is effectively searching through the hypothesis space of all possible trees starting with the simple empty tree and progressively adding complexity until the tree matches the hypothesis. Because of these features it is possible to say that the ID3 decision tree algorithm is bias towards smaller trees, this is an example of 'inductive bias' (which will be addressed later). This related to the heuristic proposed by William of Occam his so called 'Razor' states:

Prefer the simplest hypothesis that fits the data.

Most of science has this bias, i.e. science prefers simple or elegant explanations to complex ones. An argument put forward for this is statistical in nature, i.e. there are many more complicated ways of existing than there are simple, and therefore a simple solution that is found is more probably correlated with reality than a complex explanation. (Mitchell, 1997).

Because the learning search in trees can quickly grow to a very large number of trees a number of people developed algorithms that attempt to prune away decisions branches in order to make the computation feasible. (Mitchell, 1997).

Machine learning in general

Over the last 60 years machine learning has begun to mature as a field of research with a lot of published work not only on computational learning theory but also the practical applications of the developed techniques. Tom Mitchell gives a widely accepted definition:

A computer program solving task T learns from Experience E with respect to performance measure P, if its performance in task T as measured by P improves with experience E.

He also goes on to explain that a problem that requires learning needs to be well defined with proper performance metrics specified that give a clear idea of the learning task and what appropriate training data needs to be arranged. It is also important to identify how to measure and evaluate hypotheses and how to calculate the statistical confidence level as well as the acceptable margins of error. In machine learning it is also important to identify and evaluate the inductive learning bias of the learning technique utilised. It has been shown that in order to perform any kind of learning it is necessary to have a bias otherwise there is no rational reason for deciding which classification to assign to any instance. The basic tenet of machine learning is the inductive learning hypotheses which states:

Any hypotheses found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over unobserved examples. (Mitchell, 1997)

Some functions will include a target space of every possible hypotheses, others a subset. In very expressive hypothesis representations, it is impossible to generalise beyond the examples. This phenomenon is called 'Overfitting the data'. Any hypotheses that exactly match the training data will perform less well on the set of real data. Data which is especially noisy, i.e. a number of instances in the training set are wrongly labelled will lead to overfitting difficulties.

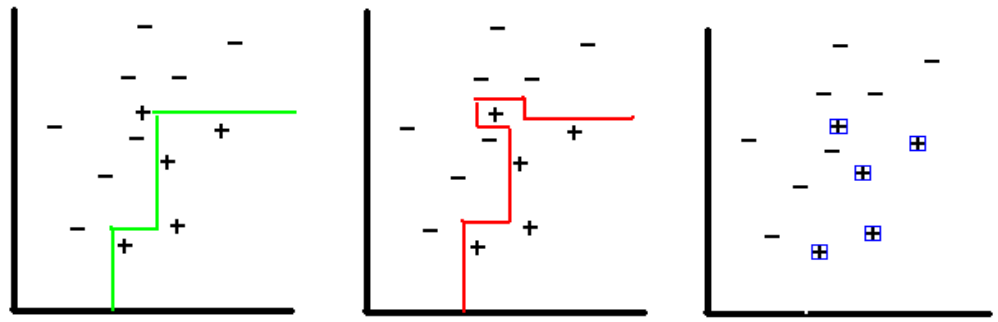


Figure 16

The first graph shows the simplest function for classifying positive examples, it makes one mistake on the training data, the second example does not make a mistake, it is a more complicated hypothesis representation and the third example is the most expressive hypotheses representation, only classifying the exact same examples. The projected relative performance of these three representations on 'real' data is shown below, which demonstrates that the simpler representation performs best.

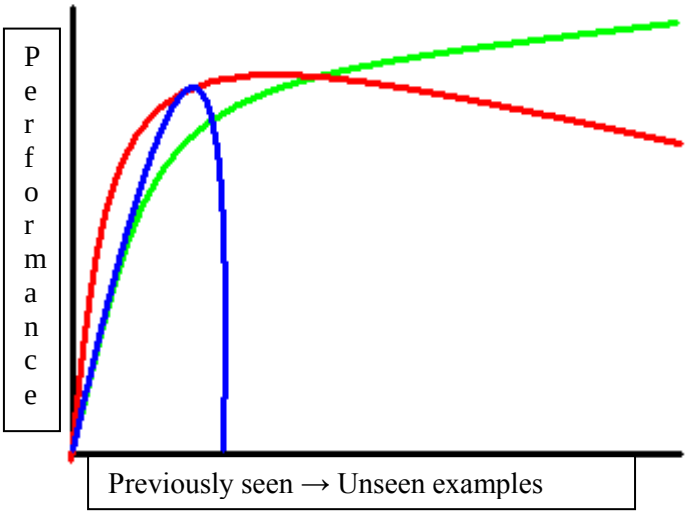


Figure 17

Different learning approaches can be classified by their inductive bias. It is preferable to make the inductive bias of learning algorithm explicit, as this can help explain the behaviour and performance of a learning algorithm at a particular task, as well as sometimes offering the ability to modify the parameters of the bias. Some algorithms were developed that attempt to learn their own inductive bias, these are called Meta learners. If an algorithm does not have an inductive bias it is a rote learner and it can only perform deduction from examples.

One of the most important factors in determining the success of a machine learning algorithm is the feature selection that an algorithm would be trained on, some argue that this is actually the essential problem of ML, as when a task is presented with large amount of data and each example is a large vector of attributes what you really want is an algorithm that detects which of these attributes are causal for the classification of instances desired. (Ray, 2011).

In computational learning theory The Probably Approximately Correct (PAC) model utilises algorithms that learn a target concept from a concept class C. It draws training example at random from a fixed yet unknown distribution, the learner will probably learn a hypothesis that is approximately correct with an error ε, as long as computational effort and training examples grow only polynomially . (Mitchell, 1997)

Within the setting of the PAC learning model, any consistent learner using a finite hypothesis space H where $C \supseteq H$ will with a probability $(1-\partial)$out put a hypothesis within error ε of the target concept after observing M randomly drawn training examples. This gives a bound to the number of training examples sufficient for successful learning under the PAC model. In the PAC model it is assumed that the learning model knows in advance that the learning concept belongs to the subset, an alternative approach is agnostic learning model which makes no assumptions of the class from which the target concept will be drawn, the learner will out put a H that has the least error over the training data.

The number of training examples required for successful learning is strongly influenced by the complexity of the hypothesis space considered by the learner; one method of measuring the complexity of the space is the Vapnik-Chervonenkis (VC) dimension, which is the largest subset that can be shattered by H. (Vapnik & Chervonenkis, 1971).

Another learning model is the mistake bound model it is used to analyse the number of training examples a learner will misclassify before it exactly learns the target concept. There is the halving

algorithm which will make $\log_2$ mistakes before exactly learning any target concept from H. The weighted majority algorithm combines the weighted votes of prediction algorithms to classify new instances. It learns weights for each of these prediction algorithms based on errors made over a sequences of examples.

As in any scientific discipline it is important to empirically measure the performance and accuracy of a hypothesis. If we measure the accuracy of a classifier on training data, the next step is to estimate the probability that this accuracy will hold on the actual instances. Another important task is when comparing the accuracy of two classifiers what is the probability that the result is down to chance rather than an actual improvement in classification? Finally what is the best way to use data to generate the most probable hypothesis? Evaluating the probability of hypothesis can also be used as an important metric for post pruning in decision trees.

In order to circumnavigate the positive bias a classifier gains from the training examples, the available data is normally split into training and testing data, the most thorough version of this would be to take K disjoint subsets and perform a leave one out test, as this also accounts for any variance in the available data by taking the average of K-tests. Central Limit Theorem which had important contributions from Turing before the war allows us to use the normal distribution to work out probabilities that these scores are typical of the underlying distributions (As long as $N \geq 30$). Further explanation of this method will come later.

William Gosset whose pen name was 'student of statistics' developed the now common approach to measure the sample error to true error values whilst working in the very noble cause of improving the quality of Guinness in 1908. The test is therefore called the t-test or student test. We can use t-tests to both measure the comparative accuracy of two hypotheses as well as two different learning methods. This then takes into account both estimation bias, and the variance in sampling.

This equation gives the confidence boundaries:

$$error_s(h) \pm z_n \sqrt{\frac{error_s(h)(1 - error_s(h))}{n}}$$

Equation 4

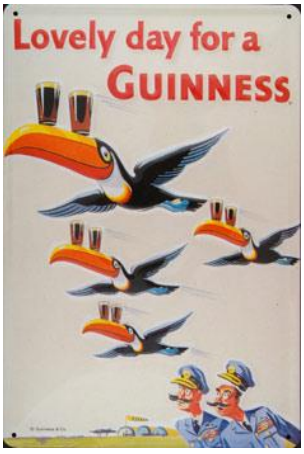


Figure 18

Further methods based on statistics and probability

As well as statistics and probability being useful to measure the effectiveness of ML algorithms some algorithms employ statistics in a central manner to perform induction. Bayesian methods are a basis for probabilistic learning, that require knowledge about prior probabilities, they allow assigning of posterior probabilities to determine the most probable hypothesis given the data, the maximum a posterior (MAP) hypothesis, this is the optimal hypothesis in the sense it is most likely, the most probable $h \in H$ given observed data D, the MAP hypotheses:

$$h_{ML} \equiv \underset{h \in H}{argmax}\ P(D|h)$$

Equation 5

These methods were built on the theory Reverend Thomas Bayes had developed in the 18th century.

Bayes Theorem :

$$p(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Equation 6

In Ml we are interested in $p(h|D)$ the probability that h holds given D, Bayes theorem allows us to calculate this when we have the probability $P(h)$ the probability of hypothesis, together with $P(D)$ the probability of data, and $P(D|h)$, the probability of data given a hypothesis.

The Bayes optimal classifier combines predictions to calculate the most probable calculation; this has the problem that lots of prior conditional probabilities are required to be known. Therefore the Naïve bayes classifier was developed in 1973 which made the naïve assumption that probabilities were conditionally independent which results in a more practical method, yet surprisingly shows little performance degradation compared to optimal methods. (Michie & etal, 1994)

The Naïve bayes classifier has been shown to perform well against other learning methods such as neural networks and decision trees (Michie & etal, 1994) Baysian methods incrementally adjust the probabilities when encountering new instances. Some hypothesis can therefore be completely eliminated from the search if no instances occur in training or alternatively Laplace correction can be instigated to give unrecorded hypotheses a small probability. All Bayesian methods can out put a probability of a hypothesis, for instance sequence A has an 80% chance of being related to sequence B. Bayes classification can incur significant computational costs, which makes it unsuitable for some tasks.

Naive Bayes Classifier:

$$v_{NB} = \underset{v_j \in V}{argmax}\ P(v_j) \prod_i P(a_i|v_j)$$

Equation 7

If the prior probabilities of data are unknown they estimated based on their occurrence in the training data, background knowledge can easily be incorporated as further prior probabilities.

A compromise between the optimal bayes classifier and the Naive bayes classifier is a Bayesian Belief Network, which allow conditional probabilities to be applied to subsets of attributes. A BBN

represents the joint probability distribution for a set of variables. Two things are needed to be known, one is the structure of the belief network dependencies and the other is the associated probabilities in the arcs of the network, if the network structures are known methods similar to neural nets can be used such as gradient decent to estimate the conditional probabilities, if the network structure is unknown then methods such as K2 (Cooper & Herskovits, 1992) or The EM algorithm can be used to build the network. (Dempster, Laird, & Rubin, 1977) The EM algorithm can also be used when the values of attributes are never directly observed.

Probabilistic methods for sequence alignment

In regards to biological sequences probabilistic methods are widely used for inferring similarities, detecting homologues and building phylogenetic trees, high quality alignments can also be built by knowledgeable experts but this is exorbitantly expensive and time consuming. If a set of sequences are homologous they are likely to result in proteins with similar features and functions. There are two key approaches, pair wise alignment and multiple sequence alignment.

Pair wise alignment is a basic sequence task used to decide if two sequences are related. This can be first attempted by trying to align two full sequences or just two sections of two sequences and then deciding whether the alignment is a product of chance or more likely because they are closely related. Of course many studies have shown that *all of life* is related and descends from a common ancestor that lived eons ago, therefore if comparing entire genomes it is matter of how closely related sequences are, however in the case of subsequences and protein sequences, two sequences could diverge not only from speciation but also within genome duplication. These two divergence phenomena are known as orthologues and paralogues respectively. (Durbin & etal, 1998)

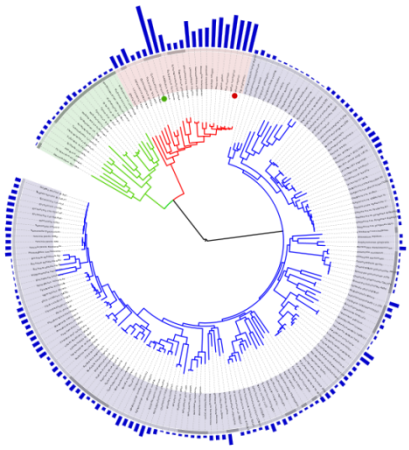


Figure 19 The tree of life

Figure 20 My (and yours!) nth cousin

In order to detect how closely related two sequences are, we decide what kind of alignment we are looking for and then we need to know how to rank an alignment by giving it a score that corresponds to the evidence they have diverged from a common ancestor by mutation and selection. Insertions and deletions are gaps, whiles substitutions are changes in residues in sequences.

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL

           G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL

HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```

Gaps are indicated by - , similarities by +.

How we score alignments is of critical importance, there are a number of different methods which attempt to incorporate biological knowledge. The scoring of a sequence comparison will be a sum of terms for alignments plus penalties for gaps. Gaps are penalised using either a linear score or an affine score. In a probabilistic interpretation this corresponds to the log of the relative likelihood. It is expected that large changes are less likely than small changes. This means that small changes should be more frequent than large changes. In an addictive scoring scheme each change is presumed to be non dependent, which of courses is not always the case. This assumption seems to hold well for DNA and protein sequences but nor RNA sequences. (Durbin & etal, 1998)

For protein sequence we would have an alphabet $\mathcal{A}${A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T,W,Y,V} of the 20 amino acids, X would be sequence 1 and Y sequence 2, i would be ith in X and j be jth in Y, X will have length n and Y length m.

In a simple random model R, a letter would occur independently with some frequency q_a. Therefore the probability of the two sequences is the product of each amino acid.

$$P(x, y|R) = \prod_i q_{x_i} \prod_j q_{y_j}$$

Equation 8

Of course in the *Match* model, aligned pairs will have a joint probability, which is the probability that a and b were both derived from an original residue c. Giving

$$P(x, y|M) = \prod_i p_{x_i y_i}$$

Equation 9

The ratio of both of these is called the odds ratio, to get an addivitive scoring we lake the log, which gives the log odds ratio

$$\frac{P(x, y|M)}{P(x, y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

$$S = \sum_i s(x_i, y_i) \text{ Where } s(a, b) = \log\left(\frac{p_{ab}}{q_a q_b}\right)$$

Equation 10

The $s(a, b)$ scores in the sum of individual scores of each aligned pair or residues. In proteins this can be arranged in a 20 by 20 matrix which is called a score matrix. Where d is the gap open penalty and e is the gap extension penalty. Once a scoring mechanism is in place, then an algorithm is needed, a large number of algorithms have been developed. If $n = m$, then there exist one optimum alignment, however if $n \neq m$ then there will be a very large number of global alignments. The algorithm for finding optimum global alignments was originally developed by Needleman and Wunsch in 1970 (Needleman & Wunsch, 1970)but a more efficient version was developed in 1982 by Gotoh. (Gotoh, 1982). It builds up optimum alignments by using previous solutions of small subsequence alignments. If we are looking to align two subsequances of *x* and *y* which is more common task due to it being a more sensitive way of detecting similarities from highly diverged sequences, (this is called finding the *local* alignment) then we can use the Smith and Waterman algorithm (Smith & Waterman, 1981) which is closely related to the Needleman and Wunsch algorithm, except an alignment can end anywhere in the matrix, and the ability for a cell to take on a zero value which corresponds to the start of a new alignment.

If using an affine gap penalty then the mathematics can be illustrated thus:

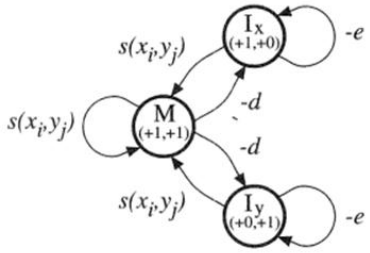


Figure 21

Which corresponds to using a Finite State Automation, each state is a node and the transitions carry a score. An alignment is a path through the states.

A major problem with these algorithms is that all though they output 'correct' results, they take an inordinate amount of time, to try an address this issue other heuristic methods have been developed. Two frequently used methods are BLAST and FASTA.

Blast was invented in 1990 by (Altschul & etal, 1990), it works with DNA and protein sequences, BLAST searches for short sections that score highly and uses these as seeds to start searching for longer alignments, BLAST keeps a record of short matches, by default for proteins three amino acids, which acts as cache, its scan a database for one of these 'words' and when it finds one begins a 'hit extension'. BLAST originally did not count gaps, but later versions added this functionality. FAST is another widely used method, but it has the advantage of being able to set the sensitivity, allowing it to be faster or more comprehensive. (Durbin & etal, 1998). When two sequences are optimally aligned then the significance can be tested with two methods either the Bayesian approach or the traditional hypothesis testing statistical approach.

Hidden Markov models

In the 1960's Hidden Markov models were first described, by Leonard E. Baum building on Markov and Newmans work on Monte Carlo random number generation. They are a special case of Andrey Markov's chains which he investigated in 1906 (Markov, 1906). In general a Markov model is a system that can have a number of states; each state has a probability of transitioning to another state independent of past states. The following diagram is a Markov model for DNA, where each letter corresponds to a state, and there is a probability parameter associated with each arrow. A sequence would also have a beginning and end state, in which case a Markov model then defines a probability distribution of all possible sequences. (Durbin & etal, 1998).

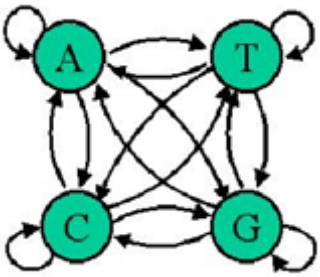


Figure 22

In a Hidden Markov model the state is not directly visible, but the output dependent on the state is visible, the sequence generated by a HMM gives information about the sequence of states, as illustrated in the diagram below.

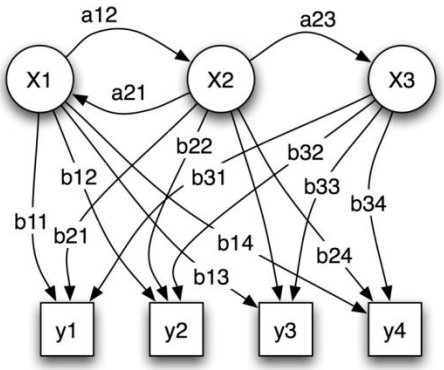


Figure 23

In the 1970's HMM's were applied to speech recognition with a lot of success, some of the early work on applying these techniques to biological sequence analysis came in the 1980's with Bishop and Thomson. HMM's can be used to ask if a sequence belongs to a certain family as well as it being able to tell us about a proteins internal structure.

The Viterbi algorithm is a method for finding the most probable path through a HMM

$$\pi^* = \underset{\pi}{\text{argmax}}\, P(x, \pi).$$

Equation 11

Where π is a path. The Viterbi algorithm does not allow us to answer all questions we may wish to ask, for instance what is the probability that observation x_i came from state *k* given the observed sequence. To calculate this we would use 'the backward' algorithm in which we calculate the probability of the entire sequence with the *ith* symbol being produced by state *k.*

A difficult problem when using HMMs is specifying the model, the states need be defined as well as the parameter transition and emission probabilities. To estimate parameter probabilities *Training sequences* are utilised, when paths are unknown this can done with Baum-Welch algorithm.

Using HMM has the advantage of being able to explore suboptimal alignments; we are able to score two sequences independently of any one alignment. For a pair HMM, there will be three core states, plus begin and end states, they are match and two states for corresponding inserts X and Y , this HMM will emit a pair wise alignment rather than a sequence.

An extension to pair alignment is to simultaneously align multiple sequences which can reveal further biological information, classically the alignment can be characterised by the frequency of symbols in positions, which can re represented by Position Specific Scoring Matrices (PSSM). However this method does work well when there are gaps of multiple lengths. Profile HMMs are an effective tool for this situation, they are able to detect if a sequence belongs to a family more reliably. Profile HMM's encode position specific information about frequency of amino acids, together with the frequency of insertions and deletions. There is a match state, an insertion state and a deletion state for each column in a multiple sequence alignment.

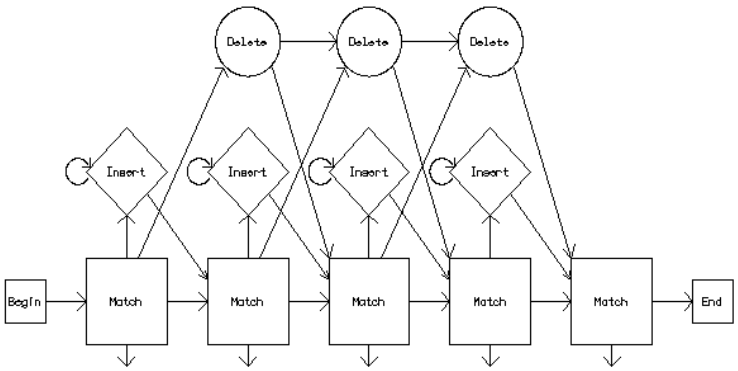


Figure 24

Two prominent packages which have been developed which use Profile HMMs are SAM and HMMER a review of these and some other motif tools was completed by (Eddy, 1998).

Genetic programming for machine learning

There is a certain symbolic irony and symmetrical satisfaction in using genetic algorithms to tackle problems in biological sequences, soon after Watson and Crick had discovered the structure of DNA computer scientists began researching genetic programming, programming influenced in an evolutionary manner. (Box, 1957) After all if nature and evolution could solve so many problems such as equipping organism to fly, to swim, and of course to act intelligently and to learn, then perhaps modelling this process will allow us to also solve these problems.

The methods that were developed are a type of dynamic programming, the search begins with a population of hypotheses, these could be a simple bit string or a complete computer program, these are then modified slightly randomly and the best approximations to the learning task are selected for, the process is then iterated. Therefore they are not searching general to specific or specific to general, but building on the best known current hypothesis. It is randomized beam search, which is less likely to fall into a local minima than a gradient decent algorithm. A threshold is defined at an appropriate level of fitness to terminate the algorithm. Genetic Algorithms' (GAs') can spawn plagues of similar solutions, the 'genetic diversity' converging and clustering, this can be mitigated by employing a fitness solution which penalises similar hypothesises over a threshold in order to gain greater genetic variation. (Mitchell, 1997).

In GA's there will be a fitness function is used to evaluate the current generation of hypothesis, some algorithms will perform crossover in analogy of sex, and of course unlike in the natural world this could be done with more than pairs of solutions in a computing environment. An advantage of GA's is that they are highly suitable to parallel processing.

The application of a fitness function to select for certain features means that most GA's are more analogous to artificial selection rather than natural selection as evolution does not aim to solve specific problems such as learning or flying, only to replicate, it just so happens that some replicators means of replicating is via flying or learning, all as a means to an end. Research into artificial life that creates a more natural selection environment includes Christopher Langtons work (Langton, 1998). Computer programming can take advantage of ideas that may be discredited in biological terms such Lamarckism as there is no theoretical reasons why this would not be a viable technique only that it is not how life works here on earth. Koza has showed how GA's can be used for classifying segments of protein molecules amongst other tasks. (Koza & etal, 1996).

Lazy Machine Learning Methods

In these methods, the generalising beyond the examples is postponed until a new instance is encountered. Training examples are simply stored; further instances may or may not then be incorporated into the model. They are known as lazy as no calculation is performed during training, conversely more computing power is required on greeting each instance. Typical methods such as K-nearest neighbour evaluate the relationship of an instance in Euclidian space to assign a target function. One key feature to be aware of in K nearest Neighbour algorithms is that they typically utilise all attributes, and do not in themselves distinguish between relevant attributes and irrelevant attributes. Therefore it is important to consider which attributes should be incorporated in a model. (Mitchell, 1997).

For K nearest neighbour all instances correspond to points in n-dimensional space where n is the attributes of an instance. The nearest neighbours of an instance are measured using standard Euclidian vectors.

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

Equation 12

K nearest neighbour can be refined by giving a weight to how far away an instance is, as well as utilising a collative vote or average from all instances, all though this can significantly increase computational cost. Taking a weighted average of vectors in this manner significantly improve performance on noisy data by smoothing incorrect instances out of the factoring. This is known as Sheppard's method (Shepard, 1968). The inductive bias of K- nearest neighbour is the assumption that the correct classification of an instance is most similar to the classification of other instances.

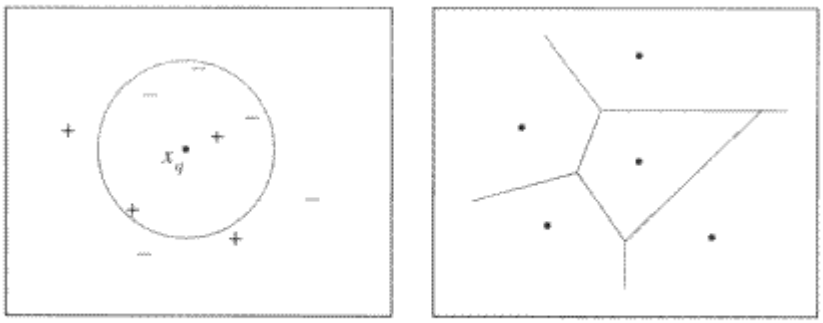


Figure 25 (Mitchell, 1997)

Support vector machines as an extension of perceptrons

A method that has been gaining ground for the last decade is Vladimir Vapnik's Support Vector Machines (SVM), these can be seen as an extension to the perceptron ANN approach and they also have similarities to K-Nearest Neighbour methods. SVMs create a plane that divides the training instances in Euclidian space, which maximises the margins from positive and negative classifications. When item sets are not linearly separable SVM's methods perform a transformation into a higher dimensionality space, to allow them to be linearly separable, SVMs in their standard form are only capable of binary classification, in order to do multiple class classification a problem needs to be broken into steps for dividing the classification into binary problems, one approach is to do this is 'All or nothing'. (Vapnik, 1995). SVMs learns a classification function from a set of positive examples $\mathcal{X}_+$ and negative examples $\mathcal{X}_-$. The function is of the form:

$$f(X) = \sum_{i:x_i \in \mathcal{X}_+} \lambda_i K(x, x_i) - \sum_{i:x_i \in \mathcal{X}_-} \lambda_i K(x, x_i)$$

The non negative weights λ_i are computed during training by maximizing a quadratic objective function, the so called Kernal function, any new instance is then predicated to be positive if the function $f(X)$ is positive.

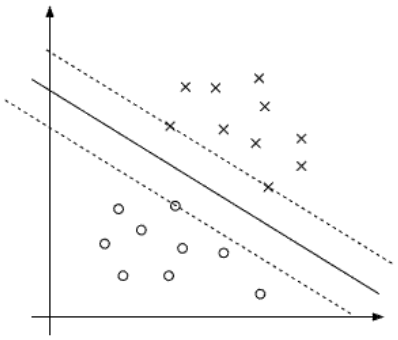


Figure 26

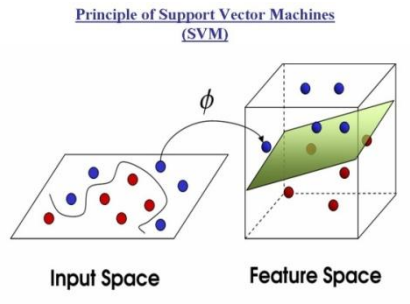


Figure 27 (Raghava, 2011)

Machine Learning Methods based on Logic

Another approach to machine learning is try to use the well understood rules of logic which have been studied by philosophers for many millennia, in common with many disciplines Epistemology the study of knowledge experienced great expansions in the 20th Century.

In regard to computer science and machine learning the aim is to build computing languages and methods which are more expressive and powerful than straight propositional logic. The English philosopher and polemist Bertrand Russell ventured in the early 1900's to produce solid axiomatic foundations of mathematics constructed from sound logical principles, he was not entirely successful yet the work he produced was very important to philosophy and logic, soon after Russell's early work the German Karl Popper made some strong arguments against induction in of itself, nevertheless computer scientists have had a number of success in taking inductive logic as a starting point regardless of how 'true' it is for human knowledge and learning. Of course decision trees and genetic programming create set of rules, yet they do not employ variables making them significantly less expressive, these techniques also do not employ sequential covering algorithms, which allow the logical rules to grow iteratively. In contrast first order rule sets or Horn clauses do allow these advantages, they allow precise recursive rules which are essential for describing some relationships between entities. The computer language Prolog invented in 1972 in Edinburgh utilises horn clauses and it is a fully Turing complete language while it employs negation as failure. (Mitchell, 1997).

To begin propositional logical rules (that are rules without variables) are constructed thus: The sequential rule algorithms such as CN2 , take an input of positive and negative instances, and create one rule, the rule is required to be accurate yet not necessarily giving high converge. This then iterates to learn a set of disjunctive rules. This is described as a greedy search which is not necessarily guaranteed to find the smallest set of rules consistent with the data. One approach developed was similar to ID3 decision tree algorithm, yet descends a single step at a time rather than growing an entire branch, this is a general to specific search through the space of possible rules, a beam search developed by Clark and Niblett in 1989 is used to reduce the possibility of making suboptimal choices, by recording K best candidates along a decision branch. (Clark & Niblett, 1989).

An advantage of searching general to specific is that there is a single most general hypothesis to begin the search from, where their maybe many equally specific hypotheses to begin searching from. This leads to an unproblematic and consistent manner of implementing a search. The tree that can be constructed as in ID3 can sometimes benefit from post pruning to avoid overfitting.

Many methods to learn first order Horn theories have been developed, in 1990 Quinlan developed FOIL, which extends the CN2 algorithm, to learn first order rules with two exceptions from strict Horn clauses, the rules are not permitted to contain literal function symbols, which reduce the complexity but are able to have negated body clauses, FOIL only searches for rules that are true and uses a simpler hill climbing rather than beam search. Foil also uses it own FOIL Gain function rather information gain based on entropy, FOIL has been shown to successfully learn recursive rules, being able to navigate the danger of infinite branches, It is possible to set FOIL parameters to balance accuracy, coverage and complexity to adequately handle noisy data. (Quinlan, Learning Logical definintions from relations, 1990).

There exists an alternative approach which supposes that induction can be modelled as the converse of deduction. This insight was first brought to attention by W.S. Jevons in the 19th century. Given some data D, and some background knowledge B, the task to generate a hypothesis that together with B explain D.

$$(\forall\langle x_i, f(x_i)\rangle \in D)(B \wedge h \wedge x_i) \vdash f(x_i)$$

Equation 13

This advantageously allows us to use the many algorithms developed for performing deduction, the idea is to design *inverse entailment operators* as their maybe many operators capable of doing this, a common heuristic employed is that of Minimum Description Length principle, another advantage is that incorporating background knowledge allows the search to be guided rather than a search amongst all syntactically correct hypotheses. This does however cause difficulties with noisy and inconsistent data, where it is no longer possible to distinguish truth values, other problems that arise is that the search space can become intractable due to the expressive nature of first order logic.

In 1965 Robinson devised a standard method for automated deduction called *resolution rule*, which was both sound and complete (Robinson, 1965), in 1988 Muggleton and Buntine successfully reversed the rule to create an inverse entailment operator which they named CIGOL (Muggleton & Buntine, 1988). CIGOL uses a sequential covering algorithm to generate candidate hypotheses h_i, that satisfy $(B \wedge h \wedge x_i) \vdash f(x_i)$ these are then added to the background knowledge and the process iterates. The process to generate first-order expressions is based on unifying two input clauses by means of a substitution , finding one Literal L_1 from clause C_1 and one literal L_2 from C_2, such that a unifying substitution θ can be found for $L_1 \rightharpoonup L_2$. The resolvent C can be found using:

$$c = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$$

Equation 14

Which leads to the inverse resolution rule:

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{\rightharpoonup L_1\theta_1\theta_2^{-1}\}$$

Equation 15

This may generate multiple hypotheses but they will all satisfy $(B \wedge h \wedge x_i) \vdash f(x_i)$ in contrast to FOIL which as it builds its rule sets may generate hypotheses that do not. A major problem with inverse entailment is that it often leads to a computational explosion, making many problems intractable. Between 1992 and 1995 Progol was developed by Muggleon, which just generates the single most specific hypothesis, which is conjoined with the background knowledge to act as bound for a general to specific search, with the principle of Minimum Description Length and a pruning algorithm that does not risk discarding the shortest hypothesis (Muggleton, 1995). This makes many problems more practical and Progol has successfully learnt how to find mutagenicity regions. (King, 1996).

Since the year 2000 Sato in Tokyo has been developing 'PRogramming In Statistical Modelling' or PRISM, which allows logic programs to be statistical models, it is built on B-Prolog and employs tabling of explanation sub graphs, to allow for practical computation, it remembers sub goals so there is no need to re-compute. PRISM uses a graphical EM(gEM), algorithm that has been shown to perform as well as specialised algorithms such as Baum-Welch algorithm and the Inside-Outside algorithm. By employing PRISM a user can build a system that models as effectively as Naïve Bayes classification or Hidden Markov models as well as many other probabilistic methods all in one place, without losing efficiency. (Kameya, 2008).

	OLDT	gEM	Specialized EM
HMMs	$O(N^2LT)$	$O(N^2LT)$	Baum-Welch
PCFGs	$O(N^3L^3T)$	$O(N^3L^3T)$	Inside-Outside
Singly connected Bayesian net	$O(\|V\|T)$	$O(\|V\|T)$	[Castillo et al. 97]
Pseudo PCSGs	$O(N^4L^3T)$	$O(N^4L^3T)$	[Charniak & Carroll 94]

N = #symbols, #states, L=sentence length, T = #data, |V| = #nodes

Table 1: Time complexity comparisons

Analytical Learning

Methods of machine learning such as decision trees and neural networks require a large number of training examples, a different approach similar to research into how humans learn is called explanation based learning (Qin & etal, 1992). It is useful when there are not many training examples available to learn from, the aim being to infer which example features are relevant. Explanation based learning uses background knowledge to reduce the hypothesis space to be searched, this can be termed the domain theory and would normally be given as a second set of Horn Clauses, an example algorithm is the Prolog EBG system produced by Kedar-Cabelli and McCarty in 1987, it is a sequential covering algorithm that guarantees to output a correct hypothesis when it is given a perfect domain theory (One that is both correct and complete), this hypothesis constitutes a formal proof. For non perfect domains it can be consider a plausible explanation. Prolog EBG will collect the relevant features that have an effect on the function; the output of the algorithm provides a complete justification for the classification of instances unlike inductive methods. These methods are useful when what we know or can compute in principle is different to what we know or can compute practically, for example in a closed world chess game, the domain theory would correspond to the rules of chess. An interesting property of the Prolog EBG method is that the inductive bias is first of all a preference for small sets of maximally general Horn clauses but more importantly it also corresponds to the background knowledge, which allows the possibility of simple adjustments to be made of the bias. This also means that the systems bias improves as it learns more about the domain. (Kedar-Cabelli & McCarty, 1987).

Work to combine logical and analytical methods with statistical methods began in the 1990's, the important questions is whether to give more weight to the background knowledge or the training examples. In 1990 Towell et al showed that Knowledge Based Artificial Neural Networks performed better than separate methods to recognise promoter regions in DNA segments, the KBANN method is limited to creating variable free Horn clauses. Another ANN method developed was the Explanation Based Neural Network, which uses a domain theory expressed as previously learned ANN's, EBNN is more accommodating of imperfect domain theories than systems such as Prolog EBG it also valuably generates a fixed size neural network rather than an ever growing set of Horn Clauses, this however does constrain complex representations. (Towell & Shavlik, 1989).

Ensembles

As the number of effective machine learning methods blossomed there have been efforts to try and combine the various approaches, tools, and techniques in order to build an ensemble of learners. The theory is perhaps a version of 'wisdom of crowds' but crowds of classifiers rather than people, these can be randomly constructed or using heuristics, a recent review of these methods was provided by Eastwood. (Eastwood, 2010).

Superfamily database

In the new millennium a project was set up to develop a database called Superfamily. The Superfamily database uses HMMs for remote homology detection, that is decide if the two proteins are related. Proteins with similar structure are likely to be related. It aims to combine sequence analysis with structural analysis of proteins, to classify all proteins into their structural domains. The Superfamily database is based on the SCOP classification of protein domains, which as described earlier was built to aid ontology. The SCOP is a standard way of doing this. The SCOP classifies proteins hierarchically, first into class, then fold, then superfamily, then family and finally domain. Proteins which are grouped together have structural, functional and sequence evidence for a common evolutionary ancestor. Therefore two sequences that are in the same lower level structural classification such as family are more likely to be homologs than two proteins in different structural classifications at the same level. It is at the superfamily level, as the name suggests, that the original Superfamily database operated. (Gough, 2001)

Two approaches to classifying protein sequences are generative and discriminative. In the generative approach we build a model for a single protein family and then evaluate each candidate protein sequence to see how well it fits the model. If the 'fit' is above some threshold then the protein is classified as belonging to the family. In contrast discriminative approaches treat proteins sequences as a set of labelled examples. Positive if they belong to the family and negative if they do not. A learning algorithm will use both positive and negative examples to learn how to discriminate the sequences whereas the generative approach can only make use of positive examples.

Thus far the Superfamily database has exclusively used a generative approach it uses multiple automatically generated overlapping models generated from a diverse set of sequences which Gough shows to perform better than a single model generated from an alignment of those sequences. (Gough, 2001).

A simple pairwise generative approach would be to use a sequence comparison program such as Blast which was described earlier, however structural analysis of proteins can detect further relationships then basic pairwise sequence alignments, this is because in a pairwise alignment differences have the same penalty no matter where they occur, in reality difference in the core structure of a protein are more significant as often the inner core of 3D protein structure will remain the same to maintain the integrity of a protein fold. Using Hidden Markov Models some information about the protein structural alignments can be inferred as the probabilities change along an alignment. A HMM can be used to generates a random sequence taking into account the parameters in the model. It is also possible to calculate the probability that a given sequence has been generated by a model by chance. A database of protein sequences can be searched by a model for members of the group by scoring all sequences in the database and selecting those with a significant score. A profile HMM is a model built from a multiple sequence alignment rather than just one sequence- this gives a better model of core underlying structures of a protein family.

The more complex generative approach of constructing profile HMM as performed for The Superfamily database can be broken down into three steps. The steps are:

1. A multiple sequence alignment is made of known members of a given profile superfamily. (The quality of the alignment and the number and diversity of the sequence it contains are critical)

2. A profile HMM of the superfamily is built from the multiple sequence alignment. The model building program uses information derived from the alignment along with prior knowledge of the nature of proteins.

3. A model scoring program is used to assign a score with respect to the model to a sequence of interest, the better the score, the higher the chance that the query sequence is a member of the protein superfamily represented by the model.

The two preeminent software tools for profile sequence alignment and modelling, SAM and HMMER were compared by the developers of Superfamily from the point of view of an informed but non expert user. An advantage found of SAM is that it contains a model builder T99, which uses Dirichlet mixture distributions to give background information on probability distributions of amino acids to eliminate any zero probabilities in columns on a multiple alignment. HMMER was shown to build better models from poor alignments then SAM, but SAM built better models from good alignments then HMMER. Comparing the time taken to run, the HMMER model calibration was slow, but both HMMER and SAM build models quickly, with SAM slightly faster. The trade off point was shown to be around 5000 sequences, after that HMMER will perform quicker than SAM. (Gough, 2002)

For the Superfamily database multiple models are built for each superfamily. Multiple models for each family are better as they are able to make more reliable fit. Bad models are detected and removed by hand.

The Superfamily database is useful for annotating data and improving the speed of genomic wide research. There were existing databases which use HMMs representing protein domains such as Pfam , SMART and others and there are also unifying databases which have several of these methods included, e.g. InterPro and CDD . The advantage of the Superfamily database is that offers whole genomes of structural protein knowledge.

The work on The Superfamily database has continued to be refined over the last decade. In 2004 Superfamily was upgraded to include domain architectures, which in analogues to protein sequences but with SCOP Superfamilies instead of amino acids (Gough, 2004) Other new developments included upgrading of blast and psi -blast.

In 2006 the Superfamily database was substantially upgraded to tackle the problem that many classification schemes for proteins and domains are hierarchical or semi hierarchical, yet many genome wide databases only provide assignment at one level of the hierarchy, including the original Superfamily database. To enable classification at the family level given the Superfamily a hybrid method between pair wise and profile methods was developed. The method developed performed better than either pair wise or HMMs on their own. A large advantage was that within the framework of the existing profile library that is provided by Superfamily database this method has very low extra computational costs. (Gough, 2006)

The Task

The aim is to improve the hybrid method in the task of assigning family level SCOP classifications to protein domains, given that they have already been assigned a superfamily. Although the hybrid method out performs simple alignment comparisons, it is appreciated that it does not perform optimally. It currently does not use any of the sophisticated classification methods which have been outlined in this report. It is envisaged that further work on developing the classification of this data will lead to much higher performance.

The hybrid method asks if a sequence is a member of subfamily A or a different subfamily, B, C etc. It takes the best pairwise score between the query domain and a sequence in family A. The following graph illustrate the current performance when compared to simple Blast alignment and the HMM curve here represents taking the family of the seed model to estimate the family level classification.

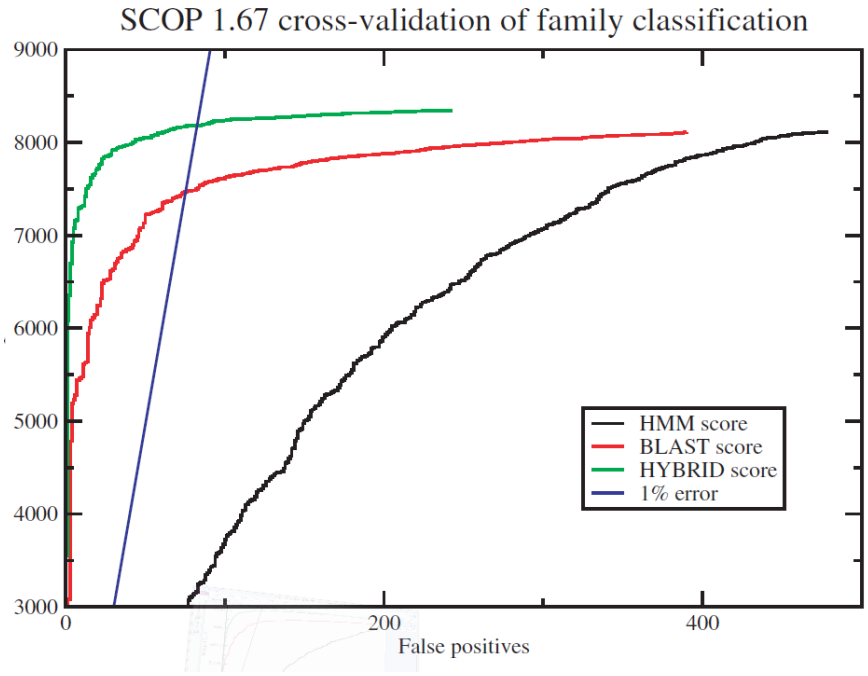


Figure 28 (Gough, 2006)How the current Hybrid method performs against a Blast alignment

Two approaches to this problem are possible, the first is to utilise the current variables as attribute vectors for an input to a multiclass classifier, this will have the advantageous potential of not increasing computational costs exorbitantly. However this method may constrain the hypothesis search to such an extent that a complete solution may not exist. The other method would involve more substantial computational costs, which would use the full available relational data from the sequence and structural information available to perform the hypothesis search, but this may prove computationally intractable, the best solution may lie somewhere along the spectrum between these two options.

Currently for the task of further identifying the proteins as belonging to a particular SCOP family given that you have already classified the sequence into the correct SCOP superfamily does not make use of all the models in superfamily database it just uses the top scoring model and the top scoring model for the next family as using multiple models increased the computational cost but with little improvement. (Gough, 2006)

Description of the data

The sequences of each domain in SCOP are obtained from the ASTRAL database, sharing no more than 90% sequence identity. (PDB 90%) This filter results in a data set consisting of proteins domains no two of which have 90% or more residue identity- this eliminates a large number of essentially redundant sequences from the ASTRAL database. The use of a domain database allows for an accurate determination of class removing the ambiguity that rises from searching whole chain protein databases.

In the SCOP classifications some superfamilies can be divided into many known families; other superfamiliess have just one or two families. Additionally in theory there could be unknown families in a superfamily that have yet to be defined. This can sometimes be inferred from a negative score on the hybrid method.

The following describes how the scores from a submitted protein domain, to domains in the database are generated:

The submitted sequence is aligned to all the models in the database. In the diagram below the HMM closest to the submitted sequence represents a high scoring alignment. This information is used to infer the domains present in the submitted sequences.

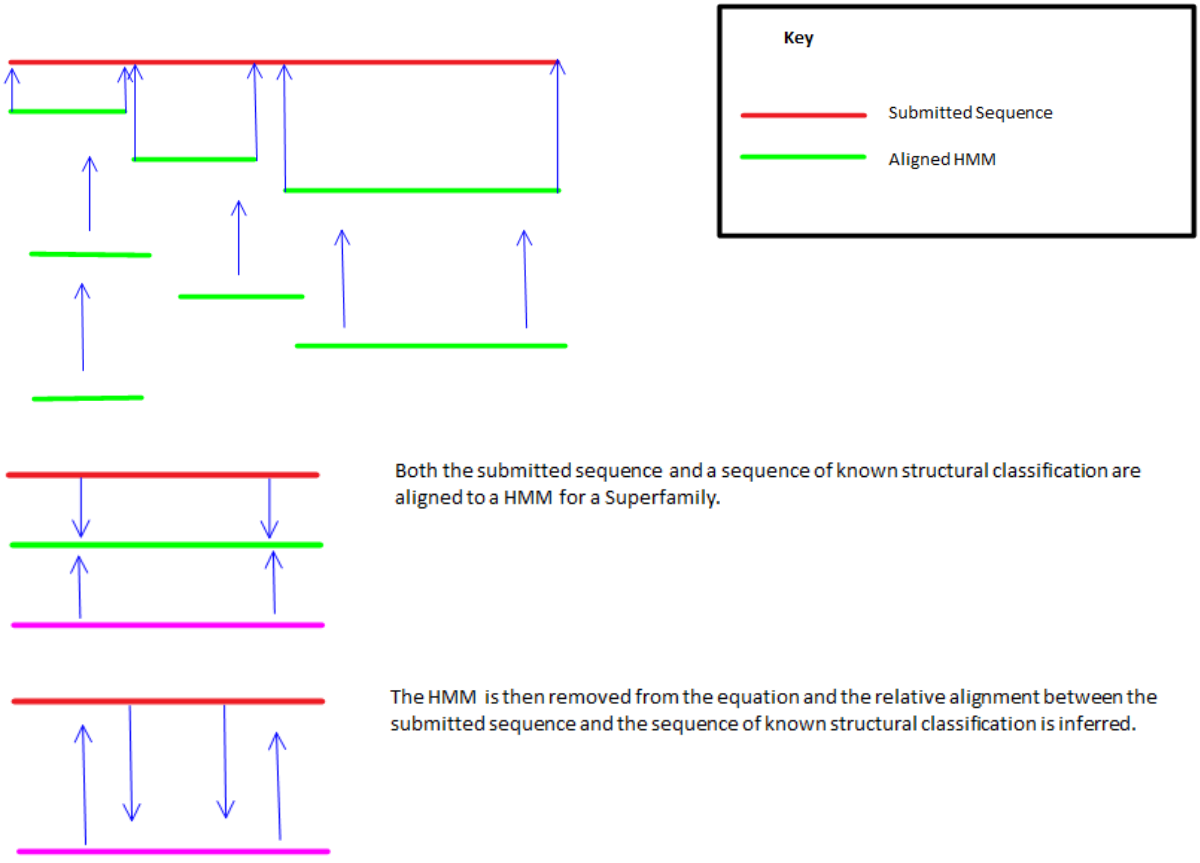


Figure 29

The scores for the alignment between the submitted domain and sequence in the database is then calculated using a substitution matrix, various substitution matrices were tried by Gough but the default BLOSUM62 matrix was found to be best. (Gough, 2006).

Similarly to pairwise methods affine gap penalties are used. Gough determined that a gap open penalty of 3 and a gap extend penalty of 0.8 were good choices, it is to note however that the gap penalties are lower than the BLAST default values. This is due to the fact that the alignment is handed down from the superfamily based HMM. The comparative importance of each location in the HMM can be calculated and used as a weight, conserved sites in an HMM contribute much more to the HMM score than other more variable sites. The value from the substitution matrix for each position can then be multiplied by the weight for that position in the model to include this information. However Gough ultimately found that the optimal performance was achieved without weighting.

This procedure will result in data that can be illustrated thus:

(TC = True Classification, FC= family classification of hit)

Submitted Domain		Hit 1		Hit 2		Hit 3		Hit 4		Hit 5		Hit 6		Hit 7	
	TC	FC	Score	FC	Score	FC	Score	FC	Score	FC	Score	FC	Score	FC	Score
A	1	1	120	1	110	1	95	2	87	3	96	3	84	4	32
B	1	2	100	1	95	2	90	3	70	2	60	2	40	2	15
C	2	2	200	1	180	2	110	3	80	2	55	4	50	3	10
D	2	3	150	2	130	2	105	1	75	2	67	4	65	4	5
E	1	1	100	1	80	3	60	2	55	1	35	2	25	1	20
F	1	1	80	2	70	1	65	2	40	3	35	1	30	1	10
G	1	1	90	1	85	1	75	1	65	2	42	4	40	4	25

Table 2

Each of the submitted domains can belong to a different superfamily meaning that the family classification of 1-4 refer to different classifications for each data instance.

The data serves as a condensed measure of similarity between submitted domains and there corresponding hit domains. If the scores are inverted they can be interpreted as a distance measure from each of the submitted domains to the corresponding hit domains, in some unknown high dimensionality space.

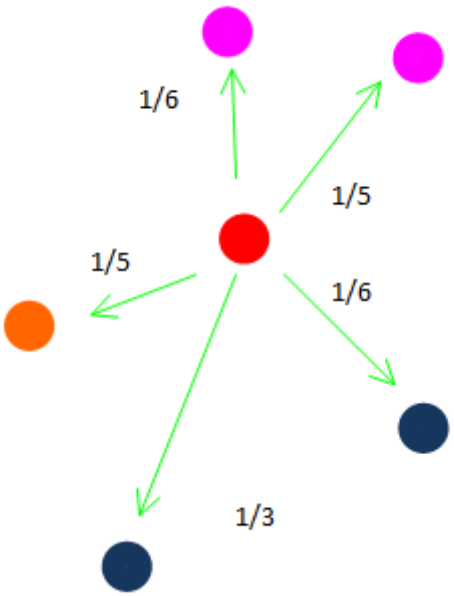


Figure 30

The advantage of viewing the scores in this manner, is that discriminative distance based learning techniques can be easily applied to classify the submitted domains into the most likely classification by taking into account the distance from the submitted domain to all of the hit domains. A lot of biological information is incorporated in one scalar value, that is already calculated as part of the model building process for the superfamily HMMs.

One way of thinking about this is to view the classification procedure as a two-step process. The first step in training the classifier is the initial alignment of the submitted domain to the superfamily HMM along with the scoring towards the domains in database. Parameters at this stage include the selection and building of the HMM, the substitution matrix used and the selection of the gap penalties. These will all influence the inductive bias of the classifier. Then we can use these values to train new discriminative distance based classifiers for each submitted domain. While each submitted domain from a unique superfamily can be seen as a separate multi class classification problem where the labels i.e the family level classifications are drawn from the corresponding set of families in superfamily. Alternatively the problem can be viewed as a binary classification problem where the labels simply correspond to correct incorrectly classified.

Evaluating Results

When evaluating the results of the classifiers, different techniques and issues will need to be taken into account depending on if the problem has been framed as a multiclass problem or as a binary classification problem. The first measure that could be used in either frame is to measure the accuracy:

For this we can represent classification results as a contingency matrix A with:

$$A_{ij}\ for\ i,j \in \vec{t} = \{t_1, \dots, t_k\}$$

Where k is the number of possible target values and $A_{\{ij\}}$ is the number of times that a data instancewith the true label t_i was classified as label t_j. Accuracy is then defined as:

$$Accuracy = \frac{\sum_i A_{ii}}{\sum_i \sum_j A_{ij}}$$

However accuracy is not a good measure when the balance of classes is out of proportion. If we had a classification task where 95% of the instances belong to one class then a classifier that simply always classified instances into this class would record 95% accuracy. This is unsatisfactory and inadequate.

This is commonly the case when the problem is more akin to detecting rather than classifying. As we have said the problem we are dealing with here can be framed in both ways. Classifying the domain sequences by their structure into family classifications or by detecting a shared evolutionary heritage between a submitted domain and the domains of known structure.

In the case of classifying with only two labels then we can draw on the large amount of work developed through the investigation of detection problems. We can use Receiver Operating Characteristics curves and F- measure. This is how the problem has been framed in previous work.

In order to plot a ROC curve we can construct a confusion matrix and calculate the precision and recall values for a classifier.

	actual class (expectation)	
predicted class (observation)	**tp** (true positive) Correct result	**fp** (false positive) Unexpected result
	fn (false negative) Missing result	**tn** (true negative) Correct absence of result

Precision and recall are then defined as:

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

Generally for a two label classification task, the confusion matrix is 2 by2 so the number of off diagonal cells is the same as the number of on-diagonal cells. So Precision and Recall give a good summary of the confusion matrix. In some settings such as this one we want to know if there are a lot of true negatives. Precision and recall are a better measure than accuracy when the classification frequencies are out of balance which is the case here.

The true positive rate is also known as sensitivity, and the false negative rate is also known as specificity. A ROC analysis investigates the relationship between these two values. The normal practise is to plot the sensitivity against the false positive rate which is 1- the true positive rate. A ROC curve can be used to address where a decision threshold should be to minimise the error rate or misclassification cost under a cost distribution (That is where specific error types are assigned different costs). They can also be used to identify regions where one classifier performs better than another or when a classifier performs worse than chance.

ROC curves were originally invented to determine a binary signal contaminated with random noise, however in machine learning scenarios it can sometimes be useful to reformulate the idea as the noisy signal coming from a mixture distribution consisting of two distributions with different means. The detection problem is then: given a received signal which distribution was it drawn from?

An important statistic used with ROC curves, is the Area Under Curve (AUC). The curve is located in a unit square AUC is ≤ 1 and ≥ 0. If AUC =1 then the classifier is perfect, if it is ½ then the classifier is normally effectively making a random guess, however it could also be the case that the costing of a classifier could result in this outcome. Any value less than a half can be inverted by simply taking the opposite value the classifier chooses. AUC can be interpreted as the expectation that a randomly drawn positive receives a higher score than a randomly drawn negative. (Flach, 2012)

The F-Measure combines precision and recall it is the harmonic mean of precision and recall. The Balanced F-score is calculated by:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In tasks with multiple classifications the confusion matrix expands so that there are many cells of the diagonal, each with a unique cost. Confusion matrices can be useful for diagnosing common confusion results.

In multiclass classification there is then a requirement to determine the relative importance that each point is classified correctly based on how it is classified and how damaging various incorrect classifications would be. This is essentially a task dependant problem as it would not be possible to build a consensus on the costs. In our case the relative damage of weighing incorrect classifications is of less importance than in other cases.

Other approaches to try and measure multiclass errors include The Balanced Error Rate, which is the average of errors on each class, another approach is to use mutual information. Mutual information is a quantity that measures the dependence of random variables measured in bits. For two variables this would be calculated by:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

However this approach has a critical problem in that perfectly bad predictions will have perfect mutual information exactly as a perfectly good prediction would.

Essentially most methods, not including the mutual information method work be evaluating the contingency table and weighting the predictions of class membership.

For our purposes here, it seems that framing the problem as a binary signal detection task will be of more utility. As a potentially a separate costing would likely need to be applied to each of the classification tasks and corresponding confusion matrices.

Validation

In order to be confident that the family level classification assigned by our candidate algorithms have a high likelihood of being correct and to evaluate the comparative quality of the candidate algorithms a number of properties and phenomenon need to be addressed.

Firstly as discussed earlier the intuitive idea of simply recording the performance of an algorithm on the available data is a poor way of comparing algorithms due to the tendencies to over fit the data on the training set. We need to take into account the statistical nature of the problem and make a number of assumptions. First of all we assume that the data we are learning from has been randomly sampled from the true underlying distribution.

Our data and loss functions

We have our labelled data D

$$D = ((x_1, y_1), \dots, (x_n, y_n))$$

And we have a finite set of models $m \in \{1, \dots, C\}$ for example m could be the k in k-nearest neighbour. If our models were not finite, then we could discretize them but this will only be effective for one parameter. More than one continues parameter will result in too many discretized models in a high dimensionality space.

We want to choose which model will be 'best' from a decision theoretic point of view, i.e. to minimize our expected loss. So we can have a loss function L and our error ε.

$$\varepsilon_m = E\, L(y, f_m(x))$$

The prediction function f is dependant both on the model and the data.

We assume that $(x, y) \in P$ are random distributed variables according to the probability distribution P where P is the true unknown distribution.

Validation

We assume that our data set was drawn IID from the true distribution P. We don't know the true distribution P so we try and estimate it.

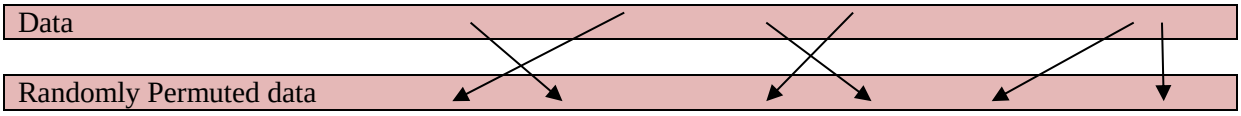
For validation we split our data set into training and validation sets. We train for each model m on the training set and then evaluate the performance on the validation set. And choose the m with the smallest error. However a model m could perform well or badly due to chance.

Cross validation

To counter for the fact that a model could perform well by chance, we repeat the splitting of the data set into training and validation sets multiple times.

The first step for cross validation is to randomly permute our data.

The second step is to split our data into K-folds. E.g. K is 5. So for example if n = 100 each fold would be a subset of 20 points.



Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

The data is then replicated into a series of rounds, with the number of rounds equal to the number of folds.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Validation				
	Validation			
		Validation		
			Validation	
				Validation

Figure 31

Each fold then takes its turn as the validation set, with the training set being made up of the remaining four folds in the round.

The process is then repeated for each m: $from\ 1:\ C$.

Then for each m we get an estimate of the error $\hat{\varepsilon}_m\{1\}$ would denote the error for mth model in the first round. We can then average the empirical error on each of the rounds with:

$$\hat{\varepsilon}_m = \frac{1}{k}\sum_{i=1}^{k}\hat{\varepsilon}_m\{i\}$$

The fourth step is to choose m to minimise $\hat{\varepsilon}_m$, we can then make the maximum use of our data by retraining m^* on all of D.

How do we choose the number of folds K? Often either 5 or 10 folds are chosen however we want to make the best use of our data and the number that we are training on is not prohibitive so we can use Leave-one-out-cross-validation this is K-fold cross validation where $K = n$. As we will then have n rounds, we don't need to permute our data. We train model m on $2 - n$ and test on datapoint 1.

So in our leave one out experimental methodology. The sequences of each SCOP family are taken as positive examples and the proteins outside the family but within the same superfamily are taken as negative training examples.

It is important to reiterate and make clear that there are two potential bias in our approach the first is that in this methodology due to the classification process being a two step process, the first part involving the alignment of models to generate the scores and the second part a distance based discriminate classifier. This will result that the training of the classifications is not entirely independent as the domains would have been used in the model generating process which are utilized in our step 1. This is probably a small problem, as one domain at this stage will have a minimal effect on the scores, however in order to fully satisfy the independence criteria all of the models would need to be regenerated multiple times, each time without the specific domain being tested occurring anywhere in the model generation process. This is an impractical proposition for this work at this time so it is to be noted as a caveat on our results.

The second important limitation is that in the leave one out methodology we are employing by its very nature will result in unbalanced data sets; negative instances will far outnumber positive instances which can cause bias in the classification process. The use of the ROC and F-measure will help to take into account this effect when choosing the best method.

Methods

The methods selected for trial were initially distance based methods that could make use of the pre calculated scores which when inverted were treated as distance metrics. Any method that made use of these would have a great computational advantage over any method that directly took the protein domain sequences as input vectors. A further advantage is that no new models would need to be stored and curated, this is a significant advantage as there would be an order of magnitude more models at the family level than at the superfamily level and the number of fully described sequences to build each model would be an order of a magnitude less due to the distribution of available data over the required models.

KNN-Classifier

In K-nearest neighbour the normal formulation of the data is of pairs $(x_i, y_i), \dots, (x_n, y_n)$ where x_i is a feature vector $x_i \in \mathbb{R}^d$ and the labels $y_i \in \{c_1, c_2, \dots, c_z\}$ where c_i represents a particular classification and z is the total number of classifications. The task then would be when presented with a new x what classification label should it be given. This is done by taking a majority vote amongst it's K-nearest neighbours. This would be done by using a distance metric normally Euclidian distance:

$$d(x_i, x_j)^2 = \|x_i - x_j\|^2 = \sum_{k=1}^{d} (x_{ik,} - x_{jk})^2$$

However in our case the distance metric has already been calculated in the form of the score, so this can just be used directly.

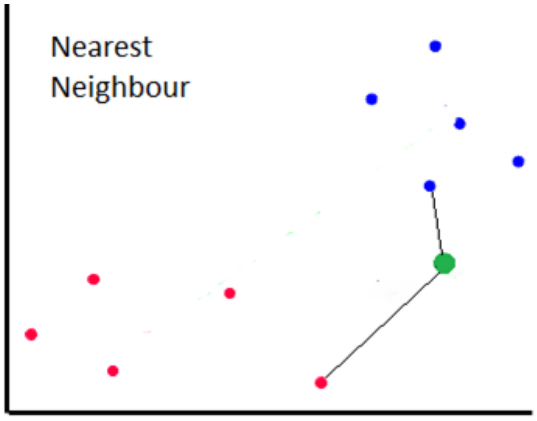


Figure32

One important property to note is that we do not know the dimensionality of the space we are working in. The scores contain a compressed version of this information. Another important thing to note is that the available classification labels are taken from set of labels in each training set for each superfamily.

The algorithm we used was adjusted so that if K was larger than the number of hits for a particular domain, then K would be reduced to the number of hits. Any ties were broken by a roulette wheel simulation.

The K-Nearest Neighbour algorithm can be given a probabilistic formulation as:

$Y \sim p \quad p(y) =$ fraction of points $x_i \; in \; N_k(x)$ with a class such that $y_i = y$.

This is sometimes written as $\hat{y} = \underset{y}{^{argmax}} p(y|x_j D)$ to show that it depends on the data and on x, but it is technically not a conditional probability as it is not defined on a joint distribution. Which is to say for each possible value of y compute the probability/fraction of points and choose the value of y for which this is maximised.

NAC Classifier

The next classifier that was developed, was The Nearest Average Classifier this was developed in order to take into account an approximate distance from some measure of the centre of all the families under one classification. It makes the assumption/simplification that all of the training points belonging to one class lie on a straight line directly away from the data point to be classified. This can be seen as part of its inductive bias.

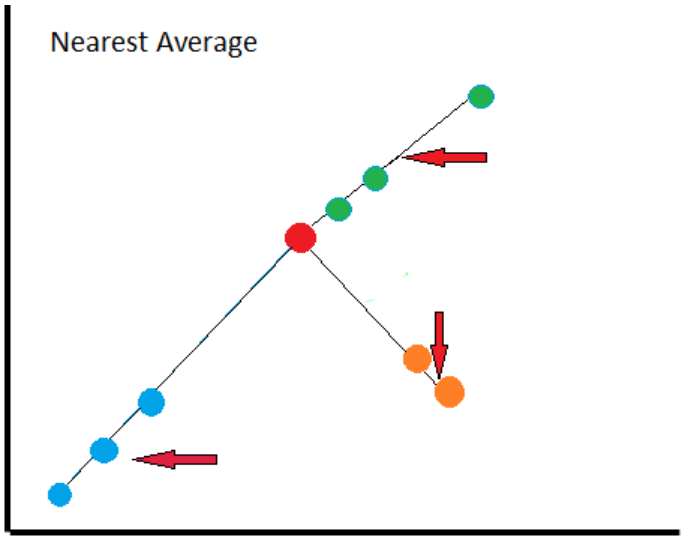


Figure 33

For each x with label y,

$$Y = \frac{1}{n}\sum_{i=1}^{n} \|x_i - x_j\|$$

$$\hat{y} = \underset{Y}{^{argmin}}$$

We then calculate the distance to each mean centre from the domain being classified. The shortest distance is the then the label assigned to the domain being classified.

Multidimensional Scaling

It was appreciated that the assumption made for the Nearest Average Classifier that all the testing data points sat on a straight line was unlikely to be the case. However to get a better approximation of where a central point might lie for a family group and then to work out the distance from our domain to be classified to that central point, we would need to ascertain where our data points lie in a space. We don't want to use the full domain sequences as input vectors as this will result in an extremely high dimensionality space, with none of the information gained from the alignments to the superfamily Hidden Markov Models. Instead we convert the distance matrix for each superfamily into a set of co-ordinates in a dimensionality of our choosing, using the statistical method multidimensional scaling, this will result in a higher dimensionality representation of our data that takes into account the information gained from the superfamily HMM alignments but is not of so high a dimensional space that the distance based methods fail due to the incorporation of uninformative features.

For each superfamily we have a dissimilarity matrix:

$$\Delta := \begin{pmatrix} \delta_{1,1} & \cdots & \delta_{1,I} \\ \vdots & \ddots & \vdots \\ \delta_{I,1} & \cdots & \delta_{I,I} \end{pmatrix}$$

The aim of MDS is given Δ, to find I vectors $x_i, \dots, x_I \mathbb{R}^N$ such that $\|x_i - x_j\| \approx \delta_{i,j}$ for all $i, j \in I$

The MDS was implemented using the Matlab function cmdscale (Mathworks, 2012).

The main problem with MDS is that depending on the implication it can be computationally costly with some algorithms taking $O(n^3)$ time. However in our case n <100 in all cases, and in the majority of superfamilies' n<10. So this is not an impossible task.

Nearest Centroid Classifier

Once we have a set of co-ordinates for our data points we can then calculate the centroid of all points belonging to a particular family classification, this will give a better measure of the central point than the Nearest Average Classifier.

The centroid of a finite set of k points $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ in $\mathbb{R}^n$ is:

$$\boldsymbol{C} = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_k}{k}$$

By calculating all of the centroids for each of the families in a superfamily, we can then measure the distance from our submitted domains to each centroid and assign the family classification to the nearest one.

Nearest Medoid Classifier

The principle behind the Nearest Medoid Classifier, is similar to that of the nearest centroid except a medoid is taken as the central point rather than the centroid. A medoid is defined as a representative object of a cluster of data, whose average dissimilarity to all the objects in the cluster is minimal. Medoids are always members of the data set. It is possible for there to be more than one medoid in which case the closest medoid is used to represent the cluster.

Results

The following plots first show the performance of different values of K in K-Nearest Neighbour , and then a comparison of the best performing 1-NN to the NAC, NCC and NMC classifiers.

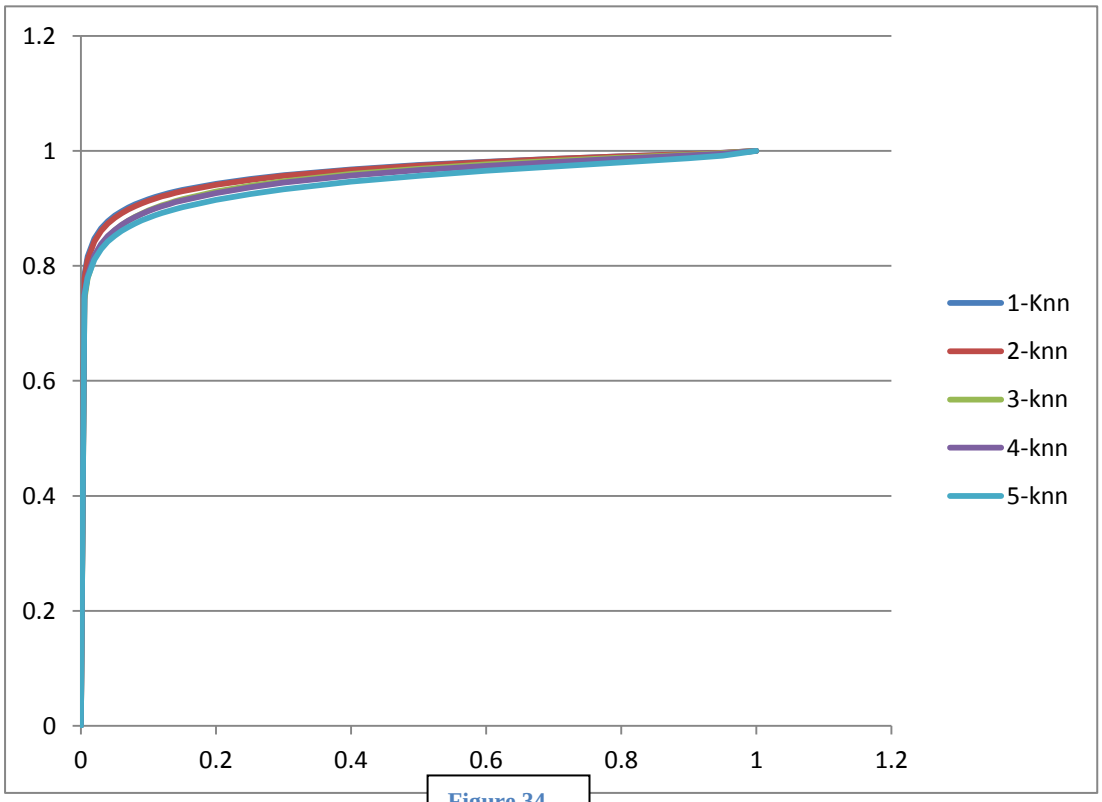


Figure 34

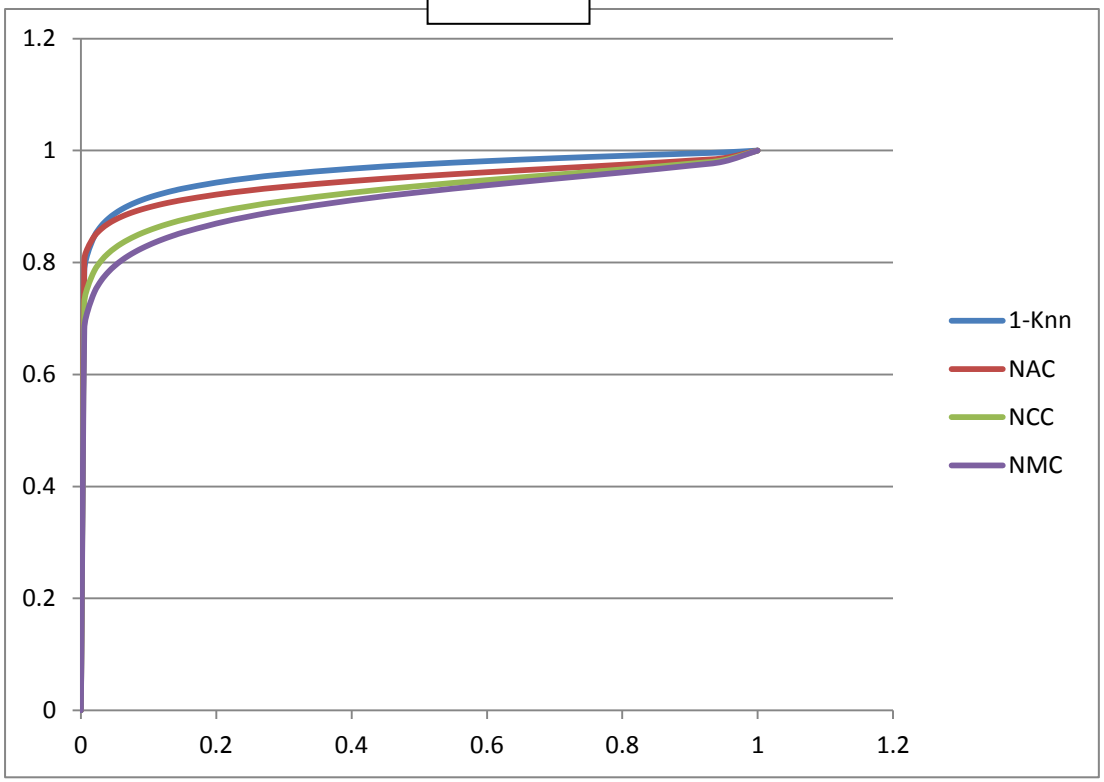


Figure 34

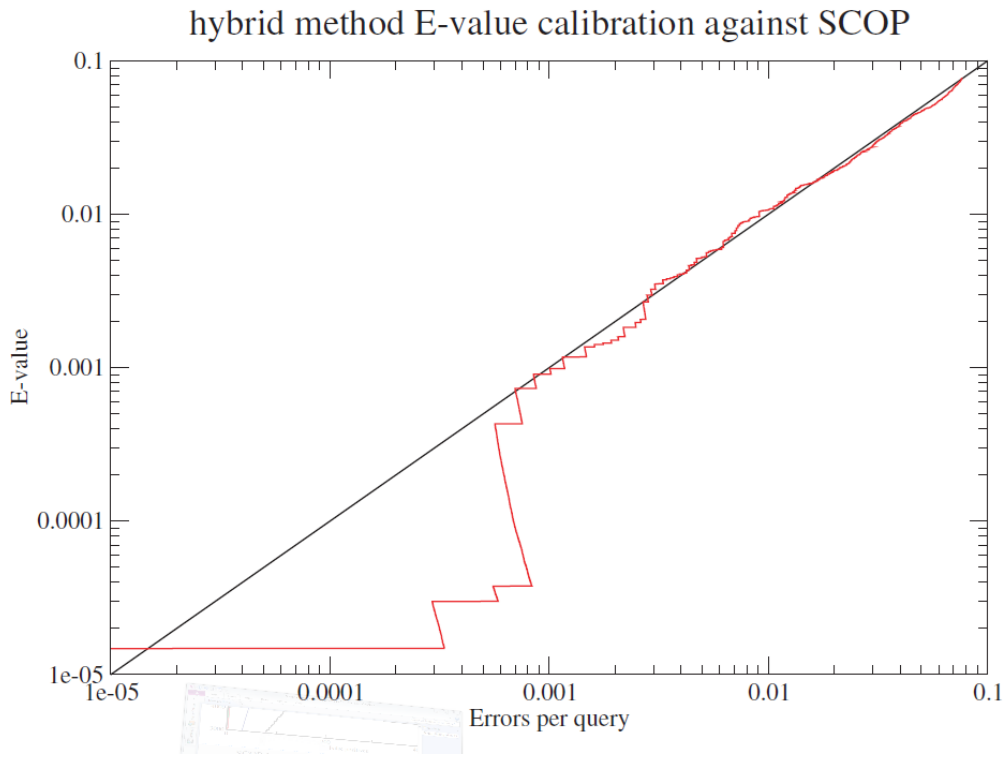
The following table summarises the results for the average values taken from the leave one out tests: The Null Hypothesis in each case is that the differences come from a normal distribution with mean 0 and unknown standard deviation. $\alpha = 0.05$.

Method	Accuracy	Sensitivity	Specificity	AUC	F-Measure	Significant
1-NN	90.2%	87.9%	94.4%	0.963	0.910341	No
2-NN	90.0%	87.5%	94.4%	0.962	0.908191	No
3-NN	88.2%	87.5%	89.5%	0.954	0.884887	No
4-NN	87.3%	88.2%	85.0%	0.952	0.865704	Yes
5-NN	86.3%	88.6%	81.3%	0.944	0.847932	Yes
NAC	89.7%	86.8%	95.0%	0.946	0.907151	No
NCC	86.3%	81.8%	94.4%	0.925	0.876495	Yes
NMC	85.7%	81.5%	93.3%	0.911	0.870017	Yes

Table 3

E-value calibration

In order to be useful to biologists the classifications need to be given an E-value, as the best performing classifier was the simple nearest neighbour results, the E-Value calibration can be the same as from Goughs earlier work. (Gough, 2006).



$$E - \text{value} = \frac{K}{1 + \mathrm{e}^{(\ln(n_2 e^{\lambda S_2}) - \ln(n_1\, \mathrm{e}^{\lambda S_1}))}}$$

Equation 16

Where K and λ are coefficients and n is the length of the target sequence, S is the raw score and subscripts 1 and 2 refer to the top scoring hit and the next highest scoring hit out side of the top scoring family. If there are no values for n_2 then it is set to 180 which is the average domain length and S_2 is set to 0 for no alignment. (Gough, 2006).

Discussion/analysis

The first graph illustrates the performance of the k-nearest neighbour algorithm through values of $k = 1$ to 5 averaged over the leave one out validated data sets.

It is clear the simplest nearest neighbour algorithm performs at a superior rate, any attempt to smooth out jitteriness by increasing k lead to inferior performance, although this is only significant when $K >= 4$ there appears to be a clear trend downwards. The curves show a uniform deterioration and at no point does a curve with a higher K value exceed the performance of a curve from a classifier with a lower k-value. This means that there would be no benefit from attempting to combine the classifiers for different decision thresholds. This coupled with the increased complexity of taking into account the values of further neighbours and resolving any ties, would lead to a recommendation not to use larger values of K.

The second graph illustrates the performance of the 1-NN when combined with methods that measure the distance to a centre point of the data clusters. This shows that the NAC had a similar performance to the 1-NN, but that again the added complexity of performing the multidimensional scaling and then performing either the NAC to NMC were ineffective in comparison to the simple NN. The added complexity again does not increase performance in any measure.

All the results show a higher specificity rate over the sensitivity rate, with the NAC recording the highest Specificity rate and the 1-NN recording the highest sensitivity rate. The highest F-score and the Highest AUC were both recorded for the 1-NN.

One of the main problems for these methods is the lack of training data for each classification task. Some superfamilies' will only have a couple of family classifications appear in the alignment process, some of these methods may improve as available data increases.

An analysis of the errors occurring in the 1-NN was performed this found that of 1332 errors made by the nearest neighbour algorithm it was found that in only 396 of these errors was the correct label classification included in the set of available labels. This meant that the submitted test domains were not getting any hits of domains in the correct family. All of the methods here would fail on these 936 cases. In order for a classifier to classify a data point it needs to have some training data of all the possible classes, it could be possible to incorporate a method such as laplace correction which is sometimes used in Bayesian methods, but this would at best be a guess for these potential classes and it would not be possible to increase performance without more information content being inputted into the training of the classifiers.

Alternative methods

An alternative server to Superfamily, which also uses profile HMMs is HHpred system, which offers a method of sequence searching similar to Blast at the more sensitive level than plain HMM's provide, this systems connects directly to databases such as SCOP, Pfam, Cog and CDD. The authors of this project have compared their HHSearch method to HMMER and PSI-Blast, and found that their method performs at a superior rate. (Soding, Biegert, & Lupas, 2005). The systems initially generate PSI-Blast searches to build a Profile HMM, which is then used to make HMM –HMM comparisons.

Recently a number of researchers to the problem of protein classification have increasingly converged on using machine learning techniques such as Support Vector Machines to achieve results that are not possible with traditional methods, Jaakkola and Haussler first introduced Fisher Kernals (Named after the biologist Sir Ronald Fisher) which incorporates generative probability models into discriminative classifiers in 1999. They added Diekhans to their team and applied the Fisher Kernal method to detecting remote protein homologies with some success. (Jaakkola, Diekhans, & Haussler, 2000).

In 2002 Liao and Noble researched a pairwise method that combined the Smith and Waterman algorithm with Fisher SVM's, which revealed previously unseen families in the SCOP database. (Liao & Noble, 2002) The work at Columbia University continued and in 2004 mismatch string kernals with SVM's were investigated. These measure sequence similarity based on shared occurrences of fixed length patterns, allowing for mutations in the patterns. This appears to have a better biological justification than the probabilistic family based HMM's. This system works particularly well when there are few training examples available, another advantage is that examination of the highly weighted patterns learned by the SVM reveals important motifs in the proteins. (Leslie, Eskin, Cohen, Weston, & Noble, 2004). Other work on string kernals was carried out in 2002 in New York, where Leslie et al produced a classifier that runs in linear time with performance that compared well to the state of the art. (Leslie, Eskin, & Noble, 2002). Further work on remote homology detection using discrete sequence motifs and SVM's occurred in 2003 at Standford University. This method successfully discovered unseen SCOP families and the authors also used the technique to predict enzyme classes. (Ben-Hur & Brutlag, 2003).

In 2004 Saigo et al used local alignment kernals, which measure the similarity between two sequences by summing up scores obtained from local alignments with gaps of the sequences. Their method again improved on existing methods whilst the kernals have sound biological motivations. (Saigo, Vert, Ueda, & Akutsu, 2004).

Work in Glasgow in 2008 combines further available features such as the amino acid composition, the predicted secondary structure, hydrophobicity, van der Walls volume, polarity and polarizability with attributes derived from local sequence alignment such as Smith-Waterman scores. They build a single multi-class kernel machine that informatively combines the feature groups with accurate weightings. This aims not only to detect further homologues but also the three dimensional structure of the proteins. This method reduces computational complexity while maintaining a solid Baysian formalism which allows greater inferences to be made of the feature spaces.

SVM's do not naturally work over relational data and of course biological data is naturally relational, they also do not provide any insight concerning the reasons for classification, to this end some very recent work at the University of Trier has focussed on the "twilight zone" between multiple sequence analysis and full sequence analyses. In such case only motifs are conserved, these authors introduce a logical representation for the known physicochemical properties which allows them to use Inductive Logic Programming to find the frequent patterns which are then used to train propositional models, which could be decision trees or SVMs, but in this case they use SVMs. Thus far they have not incorporated structural or phylogenetic tree information, which could be an obvious extension to their approach. Their method was used to evaluate protein recognition within the same superfamily, the performance compared well with the other published methods, but with the advantage that their

method provided comprehensible sets of logical rules that help ascertain the functions of proteins. (Bernades, Carbone, & Zaverucha, 2011).

An interesting and innovative approach to the sequence alignment problem has been implemented by a team at McGill University, their approach is to try and harness the collective power of human intelligence and tendency to enjoy games and puzzles. To this end they have created an online game that allows people to optimise the alignments that have been generated by their algorithms. The game is implemented in flash and players align up to nine sequences at once. (Kawrykow & Roumanis, 2010). Computing methods like these sprung out of efforts such as SETI at home, which was a screen saver designed to use people's spare computing power. Over the last decade it has dawned on humanity that the real spare computing power resides in people. To this end a number of teams have instigated a large number of successful projects, from the open source computing movement through the success of Wikipedia and can even be seen in people sharing video information on Youtube, this is sometimes known as crowdsourcing.

Conclusions and Summary

The project has evaluated and reviewed a number of machine learning techniques applied to the task of assigning SCOP family level classifications to protein domains by using existing Superfamily Hidden Markov Models in order to save on the creation, storage and maintenance of a large number of family level models.

Scores derived from aligning domains of known family classification via the Superfamily model to a new unclassified domain are calculated and the scoring to the different known family classification is then used as an input to various machine learning classifiers to infer the most likely family level classification of the unclassified submitted domain.

The experimental design was adapted from existing work; it uses a leave one out evaluation method comparing true classifications to false classifications making use of data from the ASTRAL database filtered at 90% similarity. It uses this method as the importance and costs of measuring between the different incorrect classifications is not as important in a signal detection problem such as this. The signal being detected is whether two domains show evidence of a recently sharing a common evolutionary ancestor.

The scores of the alignment were treated as representations of the inverted distances between the sequences in some unknown dimensionality sequence space. This meant that a high score from one sequence to another could be considered as two sequences being in some sense ‘close’ and therefore potentially a more likely candidate to be homologues.

As data was available giving the distances to domains of more than one family and the observed noisiness of this data needed to be taken into account for the classifications to have high accuracy, different distance based machine learning techniques that attempt to smooth out jitteriness were evaluated. This first included various incarnations of the K-nearest neighbour algorithm and then increasingly complex attempts to derive measures of central tendency first with a nearest mean classifier, then transforming the distance matrices into co-ordinate grids in high dimensionality space using multidimensional scaling. These co-ordinate matrices were then used to calculate nearest centroid and nearest medoid points from the unclassified domains. However results from these attempts to smooth out and take into account the distance scores to a greater number of known domains proved disappointing and certainly did not justify the extra complexity and resources they required.

An evaluation of the errors from the simple distance based classifier was then undertaken, this showed that these methods could only theoretically achieve 95% accuracy over the available testing data set, due to 5% of domains used for testing not resulting in any hits of the correct family furthermore of the 4% of errors made that at least have a correct hit the correct hit was never the most numerous and no discernible way was evident in order to discriminate the correct classification without importing further information to use as features from the sequences themselves, however this would counteract the aim of the project of using existing superfamily Hidden Markov Models as the prime data feature for classification.

Therefore although the hybrid method of using superfamily profile Hidden Markov models and combining these with alignments to sequences of domains of known family classification can be effectively used to assign family level classifications to unclassified domain sequences attempts to improve this by smoothing out the jitter using central distance learning methods proved complex, resource intensive and did not deliver improvements in performance. In order to improve the performance of family level classification beyond these scores new information content will have to

be inputted into the characteristics of different family domains, whether this can be done without the need to build a full library of family level models remains the subject of future work.

The work on protein classification and homology detection is a very active area of research as it remains a challenging multifeature, multi class problem, methods continue to be improved and it is hoped that this project will have made a contribution to these improvements.

Bibliography

Alcatel-Lucent. (2006). *Bell Labs Advances Intelligent Networks*. Retrieved May 04, 2011, from alcatel-lucent: http://www.alcatel-lucent.com/wps/portal/!ut/p/kcxml/04_Sj9SPykssy0xPLMnMz0vM0Y_QjzKLd4w39w3RL8h2VAQAGOJBYA!!?LMSG_CABINET=Bell_Labs&LMSG_CONTENT_FILE=News_Features/News_Feature_Detail_000025

Altschul, & etal. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215. 403-410.

Baldi, P. (2001). *Bioinformatics: The Machine Learning Approach.* Cambridge (USA): MIT Press; 2nd Revised edition edition .

Ben-Hur, A., & Brutlag, D. (2003). Remote Homology detection: a motif based approach. *Bioinformatics*, Vlo 19 Suppl 1 i26-i33.

Bernades, J., Carbone, A., & Zaverucha, G. (2011). A discriminative method for family-based protein remote homology detection that combines inductive logic programming and propositional models. *BMC Bioinformatics*, 12:83.

Box, G. (1957). Evolutionary operation: a method for increasing industrial productivity. *Journal of the royal statistical society*, 62, 81-101.

Chomsky, N. (1957). *Syntactic Structures.* Mouton & Co.

Chou, K., & Zhang, C. (1995). Prediction of protein structural classes. *Critical review Biochem mol biol*, 30: 275-349.

Christina, S., Leslie, & Eskin, E. (2004). Mimatch string Kernels for discriminative protein classification. *Bioinformatics*, Vol 20, 467-476.

Clark, R., & Niblett, R. (1989). The CN2 induction algorithm. *Machine Learning*, 3, 261-284.

Clayden, J., Greeves, N., Warren, S., & Wothers, P. (2001). *Organic Chemistry*. Oxford: OUP Oxford.

Cooper, G., & Herskovits, e. (1992). A Baysian method for the induction of probabilistic networks from data. *Machine Learning*, 9,309-407.

Correns, C. (1900). *G.Mendel's Law Concerning the Behavior of the Progeny of Racial Hybrids.*

Crick, J. D., & Watson, F. (1953). *A Structure for Deoxyribose Nucleic Acid.* Nature.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* UK: John Murray.

Dawkins, R. (1997). *Climbing Mount Improbable* . Penguin.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likilihood from incomplete data via the EM algorithm. *Journal or the Royal Statistical society*, 39, 1-38.

Durbin, R., & etal. (1998). *Biological sequence analysis.* Cambridge Uk: Cambridge University Press.

Eastwood, M. (2010). *Building well performing classifier ensembles, model and decison level combination.* Bournemouth University Thesis.

Eddy, S. (1998). Profile Hidden Markov Models. *Bioinformatics*.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162: 705-708.

Gough. (2002). A comparison of hidden Markov model procedures for remote homology detection. *Nucl. Acids* , 30(19), 4321-8.

Gough. (2004). The superfamily database in 2004 : additions and improvements. *Nucl. Acids Res*, 32(1), D235-9.

Gough. (2006). Genomic scale sub-family assignment of protein domains. *Nucl. Acids Res*, 34(13) 3625-3633.

Gough, K. H. (2001). Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *J Mol Biol*, Nov 2;313(4):903-19.

Harrison, K. (2011). Retrieved May 03, 2011, from 3dchem: http://www.3dchem.com/molecules.asp?ID=196

Hodges, A. (1992). *The Alan Turing: The Enigma.* Vintage; New edition edition.

Hunt, E. (1966). *Experiments in induction.* New York: Academic Press.

Jaakkola, T., Diekhans, M., & Haussler, D. (2000). A Discrimative framework for detecting remote protein Homologies. *Journal of computational biology*, 95-114.

Jones, A. (2005). *Chemistry - An introduction for medical and health sciences.* John Wiley and Sons.

Kameya, S. (2008). *New advances in logic-based probabilistic modeling by PRISM.* Springer.

Kawrykow, A., & Roumanis, G. (2010). *Phylo*. Retrieved May 08, 2011, from McGill University: phylo.cs.mcgill.ca

Kedar-Cabelli, & McCarty. (1987). Explanation based generalisation as resolution theorem proving. *Proceedings of the the fourth international workshop on machine learning*, 383-389.

King, M. (1996). Structure activity relatioships derived by machine learning: The use of atoms and their bond conectivities to predict mutagenicity by inductive logic programming. *proceedings of the national academy of sciences*, 93, 438-442.

Koza, & etal. (1996). Four problems for which a computer program evoloved by gentic programming is competitive with human performance. *Proceedings of the 1996 IEEE International conference on Evolutionary Computation*, 1-10.

Langton, C. (1998). *Artifical life - an overview.* Cambridge UsA: MIT Press.

Leslie, C., Eskin, E., & Noble, W. (2002). The spectrum Kernel: A string Kernal for SVM protein classification. *Pacific Symposium of Biocomputing*, 7: 566-575.

Leslie, C., Eskin, E., Cohen, A., Weston, j., & Noble, W. (2004). Mismatch String Kernels for discriminative protein classification. *Bioinformatics*, Vol 20 467-476.

Liao, L., & Noble, W. (2002). Combining pairwise sequance similarity and support vector machines for remote homology detection. *Proceedings of the sixth Annual international conference on computational molecular biology*, 225-232.

MacHale, D. (1985). *George Boole: His Life and Work.* Boole Press Ltd.

Markov, A. (1906). Extension of the law of large numbers to dependent quantities (In Russian). *Izvestiya Fiziko-Matematicheskogo Obschestva pri Kanzoanskom Universitete*, 135-156.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathmatical Biophysics* , 5, 115-133.

Mendel, G. (1866). Experiments on Plant Hybridization). *Verhandlungen des naturforschenden Vereins Brünn*.

Michie, & etal. (1994). *Machine Learning, neural and statistical classification.* New York: Ellis Horwood.

Minsky, M., & Papert, S. (1969). *Perceptrons.* Cambridge USA: MIT press.

Mitchell, T. (1997). *Machine Learning.* Singapore: McGraw Hill.

Muggleton, S. (1995). Inverse entailment and progol. *new generation computing*, 13, 245-286.

Muggleton, S., & Buntine, W. (1988). Machine invention of first order predicates by inverting the resolution. *proceedings of the fith international machine learning confernce*, 339-352.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequance of two proteins. *Journal of Molecular Biology*, 48, 443-453.

Nielsen, H., Engelbrecht, J., Brunak, S., & von-Heijne, J. (1997). Nielsen, H., Engelbrecht, J., Brunak, S., and vIdentification of prokaryotic and eukaryotic signal peptides and prediction oftheir cleavage sites. *Protein Eng.*, 10, 1– 6.

Qin, & etal. (1992). Using explanation based generalisation to simulate human learning from examples and learning by doing. *proceedings of the florida AI research symposium*, 235-239.

Quinlan, J. (1986). Induction of Decsion Trees. *Machine Learning*, 1. 81-106.

Quinlan, J. (1990). Learning Logical definintions from relations. *Machine Learning*, 5, 239-266.

Raghava, D. G. (2011). *algorithm*. Retrieved from imtech: http://www.imtech.res.in/raghava/rbpred/algorithm.html

Ray, O. (2011). *Learning from structured data*. Retrieved May 04, 2011, from Bristol University - : http://www.cs.bris.ac.uk/Teaching/Resources/COMSM0301/

Ridley, M. (2000). *Genome: The Autobiography of a Species in 23 Chapters.* Fourth Estate; New Ed edition .

Robinson, R. (1965). A machine oriented logic based on the resolution principle. *Journal of the ACM*, 12, 23-41.

Rosenblatt, F. (1959). The perceptron: a probabilsitic model for information storage and organisation in the brain. *Psychological review*, 65. 386-408.

Saigo, G., Vert, J., Ueda, N., & Akutsu, T. (2004). Protein homology detection using string allignment kernals. *Bioinformatics*, Vol 20 no 11 1682-1689.

Sanger, F. (1977). Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 687-695.

Shannon, C. (1937). *A Symbolic Analysis of Relay and Switching Circuits.* Cambridge (USA): MIT.

Shannon, C. (1948.). A Mathematical Theory of Communication. *The Bell System Technical Journal,*, Vol. 27, pp. 379–423, 623–656, July, October, .

Shannon, C. (1951). Prediction and entropy of printed English. *The Bell System Technical Journal*, 30:50-64, January .

Shepard. (1968). A two dimensional interpolation function for irregularly spaced data. *proceedings of the 23rd national conference of the ACM*, 517-523.

Sherman, W. H. (2011). *How to Make Anything Signify Anything*. Retrieved May 04, 2011, from www.cabinetmagazine.org: http://www.cabinetmagazine.org/issues/40/sherman.php?utm_medium=referral&utm_source=pulsenews

Smith, J. M. (1982). *Evolution and the Theory of Games.* Cambridge UK: Cambridge University Press.

Smith, T. F., & Waterman, M. S. (1981). Identifaction of common molecular subsequences. *Journal of molecular biology*, 147: 195-197.

Soding, J., Biegert, A., & Lupas, A. (2005). The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Research*, Vol33 244-248.

Source, U. O. (2011). *Youdebate*. Retrieved May 04, 2011, from http://www.youdebate.com/cgi-bin/scarecrow/topic.cgi?forum=3&topic=94081

Towell, G., & Shavlik, J. (1989). An apporach to combining explanation based and neural learning algorithms. *connection science*, 233-255.

Turing, A. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, 59, 433-460.

Universe Review. (2011). *Atoms*. Retrieved May 03, 2011, from Universe Review: http://universe-review.ca/F13-atom.htm

Vapnik. (1995). Support-vector networks . *Machine Learning*, 20, 273-297.

Vapnik, V., & Chervonenkis, A. (1971). On the uniform covergence of relative freuencies of events to their probabilities. *Theroy of probability and its applications*, 16, 264-280.

Venter, C. (2001). The Sequence of the Human Genome. *Science*, 1304.

Vries, H. d. (1889). *Intracellulare pangenesis.* Jena.

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. *IRE Wescon convention record*, 4, 96-104.

WikiCommons. (2011, May 03). *http://en.wikipedia.org/wiki/File:DNA-structure-and-bases.png*. Retrieved from www.wikipedia.org: http://en.wikipedia.org/wiki/File:DNA-structure-and-bases.png