2005

Fast protein superfamily classification using principal component null space analysis.

Leon French
University of Windsor

Fast Protein Superfamily Classification using Principal Component Null Space Analysis

by

Leon French

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2005

Your file Votre référence
ISBN: 0-494-09813-9
Our file Notre référence
ISBN: 0-494-09813-9

Abstract

The protein family classification problem, which consists of determining the family memberships of given unknown protein sequences, is very important for a biologist for many practical reasons, such as drug discovery, prediction of molecular functions and medical diagnosis. Neural networks and Bayesian methods have performed well on the protein classification problem, achieving accuracy ranging from 90% to 98% while running relatively slowly in the learning stage. In this thesis, we present a principal component null space analysis (PCNSA) linear classifier to the problem and report excellent results compared to those of neural networks and support vector machines. The two main parameters of PCNSA are linked to the high dimensionality of the dataset used, and were optimized in an exhaustive manner to maximize accuracy.

Acknowledgements

I owe many sincere thanks to Dr. Alioune Ngom, as a good friend and advisor for his expert guidance, sage advice and innovative suggestions. Dr. Ngom contributed the spark and foundation that got this thesis going, and provided constant support to keep it progressing. Without his support, knowledge, encouragement, and deadlines this thesis would be a much lesser work.

I acknowledge and am thankful for the support of Dr. Rueda my Co-Advisor for introducing and further developing my interest in pattern recognition. His expert teaching and guidance formed the skills necessary to complete this thesis. He contributed vast support and expert consultation in the area of pattern recognition.

I thank Dr. Kobti for his academic guidance, coaching and insight.

I thank Dr. Pandey for his enthusiastic teachings and for helping me get a better grasp of a Biologists perspective.

I thank my family for their constant support of my work, especially my Mother, Patricia French for her expert proofreading support, my Father, Doug French for encouraging my enrolment in graduate studies, Sara, and my brothers Nathan, Nils.

I acknowledge the support of my TA and GA supervisors, foremost Dr. Tsin and Dr. Lu for their support and intellectual stimulation by allowing me to help teach the most interesting classes on campus. I additionally thank Ms. Jin, Mr. Fortier, Dr. Ngom, Dr. Rueda and Dr. Mukhpadhyay for providing me with further support and enlightenment as a teaching assistant.

I thank Robert Maxwell, Sukhdeep Gill, Nikunj Modi, and Kelvin Zhang for friendship, academic support and guidance. I thank Adam Hartfie, Mike Beauchamp, and Maryam Yousif for their thought-provoking conversations and frisbee fun.

I thank Zeina El-Khaldi and Bernie Warren for teaching me the strengthening and stress relieving ways of Yoga and QiGong.

This research was also presented in short at the 18th Conference of the Canadian Society for Computational Studies of Intelligence in Victoria, BC (2005) as published by Springer Verlag. The work was titled "Fast Protein Superfamily Classification using Principal Component Null Space Analysis" with the authors of L. French, A. Ngom and L. Rueda.

This research has been partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Bioinformatics and Computational Biology

Recently the world of biology has gained the ability to decode or read large amounts of data from many organisms. This genetic data is a sort of blueprint or source code of an organism which is encoded with the discrete alphabets of RNA (ribonucleic acid), DNA(deoxyribonucleic acid) or amino acids. This data is legion in size, approaching gigabyte quantities with ease. To analyse and understand these data biologists have enlisted the power of computers. This new area of research has been named Bioinformatics, a mix of computer science and biology. The human genome project describes bioinformatics as "the science of managing and analyzing biological data using advanced computing techniques". This thesis adds to the area of bioinformatics by introducing a unique pattern recognition algorithm to the problem of protein sequence classification and splice site recognition.

Given this combination of Biology and Computer Science, a large amount of material needs to be introduced. To start, DNA and DNA splice sites are explained, then protein and protein superfamily classification. Next, the area of pattern recognition is briefly introduced followed by a short survey of past work in superfamily classification.

1.1.1. DNA

DNA or deoxyribonucleic acid is the core chemical structure for storing genetic information. DNA has four bases or structures which are combined in strands

like characters forming a word. These four bases are cytosine, guanine, adenine and thymine also known as C, G, A and T respectively. Reading of DNA strands to find the sequence of the bases is known as sequencing. State of the art sequencing methods reveal vast amounts of genetic data. Recently the human, rat, and mouse genomes have been sequenced and many more are in progress[1]. For example, Craig Venter is leading the Sorcerer II Expedition on a global quest to discover millions of new genes. His most recent results provided 1.2 million new genes (Venter, Remington et al. 2004). It is important to organize and annotate this massive amount of sequence data to maximize its usefulness – the primary goal of bioinformatics.

1.1.2. Splice Sites

A large amount of the sequenced DNA data is not directly or clearly related to the inner workings of the cell. They are non-coding DNA segments and are not instructions (code) for the production of protein. Inside the cell the coding and non-coding DNA segments are separated at splice sites.

The splice regions are windows of DNA around a splice site. A splice site separates an intron region from an exon region. Introns are known as junk or non-coding DNA segments, this DNA does not code for protein and its purpose is unknown. Conversely, the exon region is DNA that does code for the protein that the gene produces. When a gene is translated into a protein these regions are taken or spliced out, the goal is to recognize these splice sites computationally given a DNA sequence.

When the splice site recognition is combined with prediction of the gene start and

[1]National Human Genome Research Institute, http://www.genome.gov.

gene end signals a system can be created that can recognize complete genes. Finding these genes is of great interest to a bioinformatician or molecular biologist and is a difficult problem to solve with annotation accuracy.

Current work in gene prediction has revealed how small the human genome is. Estimates of the number of human genes stood at 100,000-140,000 genes before sequencing of the genome. After applying the gene recognition algorithms (which involve splice site recognition) the gene count has been put currently at around 30,000 which is less than three times the number of genes in a fruit fly. Research is still active in this area and it could be several years before an accurate gene count exists.

1.1.3. Protein

Protein sequences are strings hundreds of characters in length, where each letter represents one of 21 possible amino acids. All biological cells contain these sequences which are decoded from a DNA gene then converted to an RNA representation of the protein sequence. The RNA representation is then translated into a three dimensional molecule, a polypeptide or protein. These proteins are the gears, cogs and materials of a cell, like variables and functions in a computer program. Another analogy is strings or words that can be translated into sounds, similar to the 3 dimensional molecular translation of a protein sequence into a protein inside the cell. Figure 1 Central Dogma of Life, displays the conversion from DNA strands to protein.

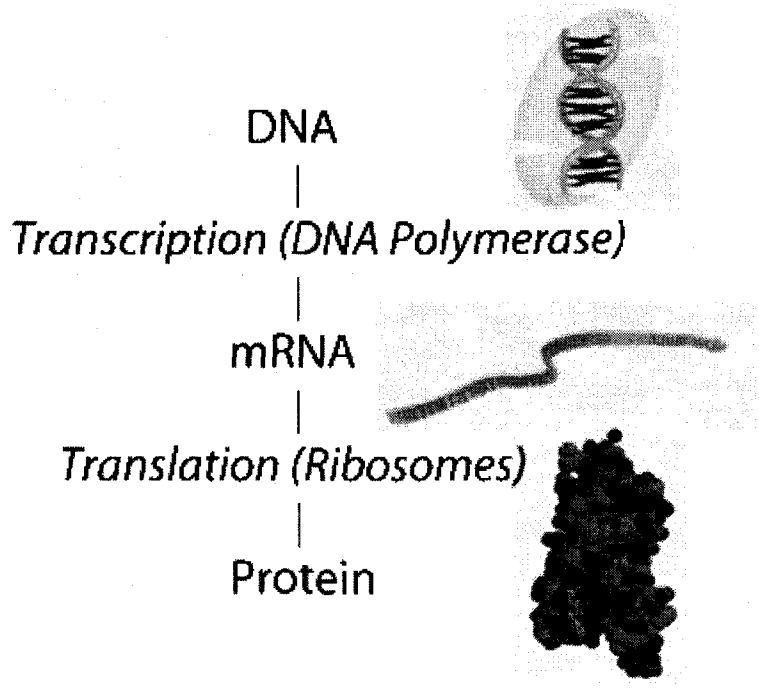


Figure 1 Central Dogma of Life

1.1.4. Protein Superfamilies and Protein Sequence Classification

The main source of the protein sequences is DNA sequencing projects. These projects produce new and unique protein sequences at an exponential rate. Annotation of these sequences greatly increases the research value of the sequences. One such annotation is organizing the protein sequences into superfamilies. A protein superfamily is a group of related proteins, the two important definitions are provided by the Protein Information Resource (PIR) and the Structural Classification of Proteins database (SCOP). The PIR database defines superfamilies as a set of sequences with similar global sequence similarity and having the same domain architecture(Wu, Yeh et al. 2003). The SCOP definition is:

> Families, whose proteins have low sequence identities but whose structures and, in many cases, functional features suggest that a common evolutionary origin is probable, are placed together in superfamilies; (Murzin, Brenner et al. 1995)

The SCOP database also groups proteins into Fold, Family, and Protein Domain levels, creating a hierarchical classification.

Protein sequence classification and remote homology detection are similar problems, both refer to the prediction of superfamily or other group membership of an unknown protein sequence. Protein sequence classification is an important problem in the area of bioinformatics. Protein sequence classification is used to organize the large amount of data produced by the genome sequencing projects. This organization aids in the finding of specific proteins for certain tasks, a researcher would be able to search for a certain type of protein to solve a very specific problem.

Once a protein is classified it becomes much more useful to the general research community. For example, traditionally sequences are converted to protein then biologists in a wet-lab examine the protein to annotate its structure and function. This wet-lab work is very time consuming in terms of time and resources. Analysis of the protein can be completed without lab work by comparing its sequence to other previously annotated protein and protein superfamilies –a strong match will suggest the function and structure annotation of the new protein sequence. Additionally, molecular evolution studies, protein function and structure prediction are examples in which knowledge of superfamily membership is valuable.

Stated more formally, protein sequence classification consists of determining a superfamily (or class) of an unknown sequence S given a known set of c superfamilies $\{\omega_1, \omega_2, ..., \omega_c\}$. A PIR-PSD superfamily is defined as a set of sequences with similar global sequence similarity and having the same domain architecture(Wu,

Yeh et al. 2003). Other approaches produce a probability that a given unknown protein is a member of a superfamily set.

Figure 2 From data to Information, the path of Bioinformatics provides an overview of the data, processes and resulting problems covered in the introduction. Splice site recognition and protein sequence classification displayed in boxes are the focus of this thesis. These problems are only subtasks within the problems of gene prediction and protein annotation. The arrows show a clear path from raw DNA data into valuable genetic information.

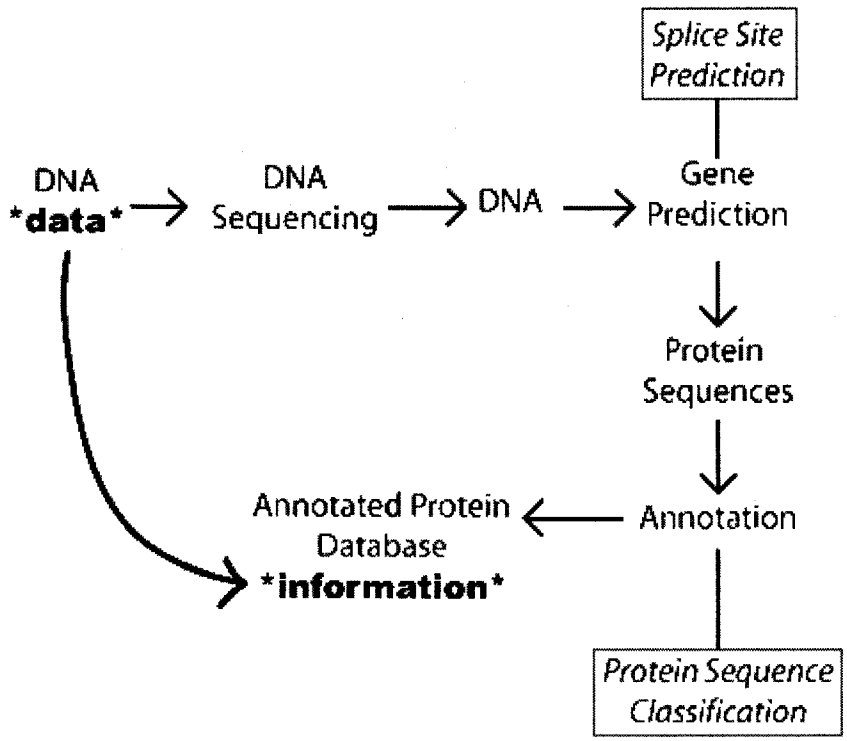


Figure 2 From data to Information, the path of Bioinformatics

1.2. Pattern Recognition

Pattern recognition is the process of recognizing patterns in data for the purposes of

classification, the area is related to artificial intelligence, machine learning and data mining. Pattern recognition has been applied to a wide variety of tasks from email filtering to face recognition. The goal is to determine correct class membership of unknown objects given past known examples and features. Many problems in bioinformatics can be viewed in this framework, two examples are splice site prediction and protein sequence classification. The work in this thesis attempts these problems and involves primarily feature extraction and linear discriminant analysis.

Normally a set of training samples are provided to an algorithm which are then used to learn the classes involved, next the algorithm can be tested on unknown samples. In Figure 3 Pattern recognition example on fruit, using a linear classifier, the process is described visually using the features of weight and size to classify object into classes of apples or oranges.

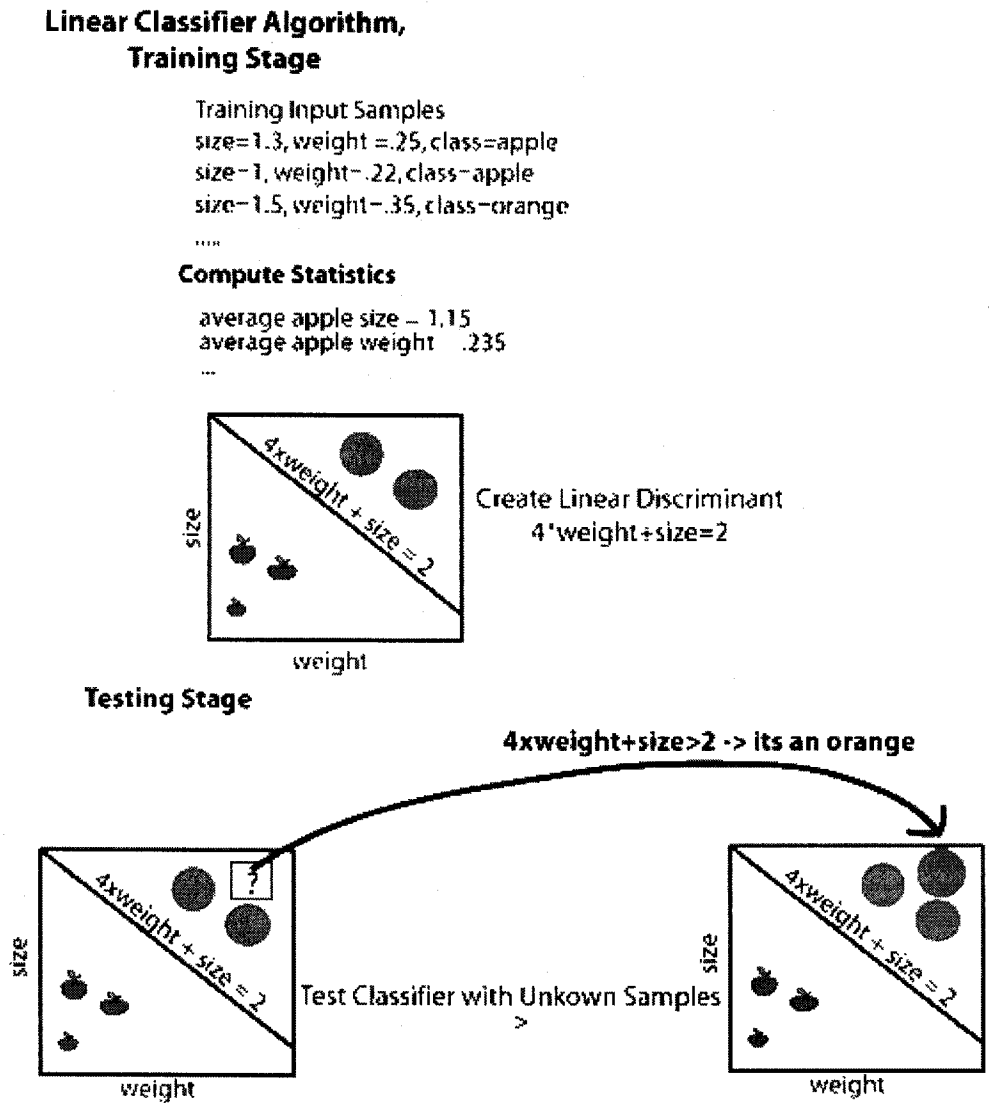


Figure 3 Pattern recognition example on fruit, using a linear classifier

1.2.1. Feature Extraction

Feature extraction is the process of converting the objects to be classified into data, or numbers. Once it's in this form a computer can process it, normally this is done with sensors or converters. To teach a computer to tell the difference between apples and oranges you need to provide a representation of the objects. For a human, it could be any of the five senses: taste, smell, touch, sound, or how it looks. For a machine any of the same could be used if a proper sensor could be created to extract

the features. For example a digital camera could be used and each pixel could represent a feature which would capture the color of a small area. This color could then be represented as a number or three, by quantifying the intensity of red, green, or blue. For a protein sequence consisting of characters a simple feature could be the length of the sequence which varies between protein types. Each object to be trained or tested with has a value for these features or attributes; these values can be numeric, boolean or nominal. A vector of these features forms the training input for a pattern recognition algorithm, with the expected output (class) for each vector.

1.2.2. Linear Discriminant Analysis

Linear discriminant analysis performs classification by creating a boundary between classes. This boundary is linear in the feature space, in a 2 dimensional feature space (each object has three features or attributes) this boundary would be a straight line, in a 3 dimensional space it would be a plane. This boundary is then used to discriminate between the classes. For example in the case of a line, the unknown points that fall below the line would be classified as class A, whilst the ones above would be classified as class B. Many advanced linear classifiers exist and perform at very high rates of accuracy, it is an active research area although other types of classifiers such as a non-linear support vector machine can outperform them.

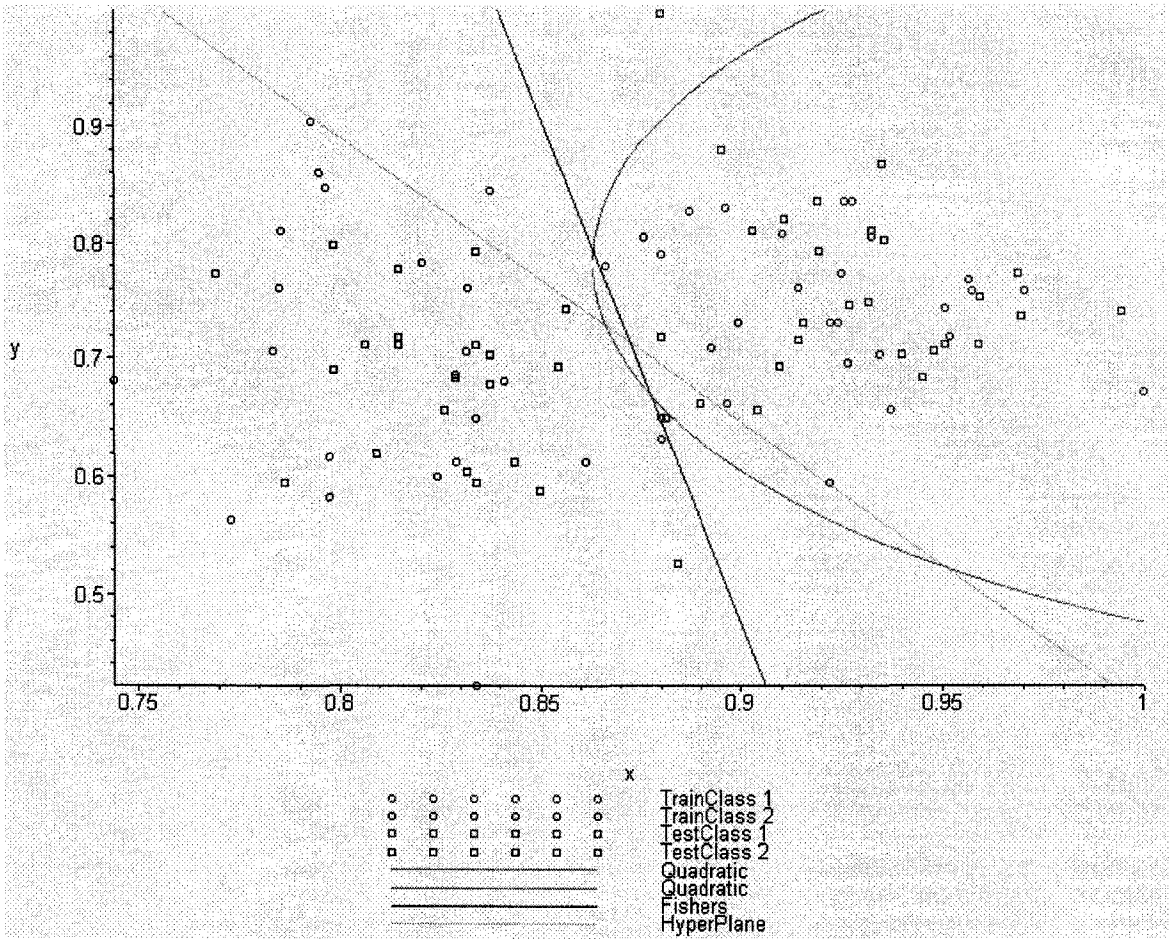


Figure 4 Example of two linear classifiers and one non-linear on a real dataset

In Figure 4 Example of two linear classifiers and one non-linear on a real dataset, two examples of a linear discriminant are shown by the two straight lines, the curved line of course is a non-linear discriminant function. Here, in one can see the separation the line creates between the blue and red points.

1.2.3. Supervised Learning

Supervised learning is an approach to pattern recognition that provides the learning mechanism with labeled known samples or objects. Given these known objects it is possible to train based on the class membership of the training objects.

For example, a parent showing a child an apple visually and then saying its name. Using this information statistics can be computed for the task of classification, such as the average colour of an orange.

Unsupervised learning only provides a set of input objects and no output result or classification. The goal in this case is to generate a model to represent the data, and possibly cluster it into groups.

1.2.4. Covariance Matrices

A covariance matrix is a statistic that measures the degree that two features change together across the dataset. Normal variance measures how closely the measurements are to the mean of the feature, or to be exact, the average squared distance of values from the mean. A covariance matrix measures the distance of one feature from its mean to another per sample. This matrix is very valuable in determining correlations in feature space. For example, a size of a fruit should be correlated to the weight –when height varies upwards, so does size. This implies a high positive value for the entries in the covariance matrix linked to the two features. Two unrelated and independent features should have a zero values in the covariance matrix.

The equation for computing the covariance matrix is provided below, n represents the number of samples, z represents the dataset, and $\hat{\mu}$ is the sample mean:

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{\mu})(z_i - \hat{\mu})^t .$$

1.3. Problem Statement

Three problems exist, stemming from the three main datasets:

A. PIR-PSD: determine the protein superfamily of an unknown sequence S given a known set of c superfamilies.

B. SCOP: determine the probability that a sequence from a previously unseen protein family is a member of a known protein superfamily. Determine the probability that a negative sample, from outside the known superfamily is part of the known protein superfamily.

C. HS3D: determine if a DNA sequence region, is a true or false human splice site, given a known set of true and false splice sites.

1.4. Contributions

Several contributions of note were achieved in this thesis:

- We document and provide results of applying a custom linear classifier based primarily upon Principal Component Null Space Analysis.
- This research demonstrates PCNSA's ability in an area it had not previously been applied, bioinformatics.
- It is shown that the two-gram sequence encoding method can still produce first class results on the protein sequence classification problem.
- PCNSA was compared against two different support vector machine designs, SMLight and SVM-Fisher. In both cases it proved itself to be at the same level of accuracy or better on the protein datasets.
- Multispace KL was also tested for its ability upon the protein sequence

based datasets.

1.5. Road Map

The rest of this thesis begins with a short survey of past attempts at the protein sequence classification problem. Next follows a section dedicated to PCNSA, then a short section explaining the lesser used Multispace KL algorithm. Then the underpinning of the empirical analysis is provided. Dataset descriptions are provided in section five, followed by Implementation details, then experimental designs in section six. The results of these experiments and implementations are documented in the Results section, which are then compared to other work in the Comparisons section. The work is then summarized in section ten, Conclusions.

2. Protein Sequence Classification

Many methods that deal with the protein classification problem have been proposed. Approaches used sequence alignment(Altschul, Madden et al. 1997) and hidden Markov modelling(Madera and Gough 2002). Sequence alignment is fast for two sequences but becomes very slow when aligning a sequence to an entire superfamily. Hidden Markov model approaches are tied to the quality of a time consuming task of multiple sequence alignment. Artificial neural networks have also been applied to the problem(Wu, Berry et al. 1995; Wang, Ma et al. 2001). Wang describes a classifier that combines the three aforementioned methods, and gives a good benchmark of the three methods (Wang, Ma et al. 2001). Recently SVMs making use of customized string kernels have been applied (Jaakkola, Diekhans et al. 2000; Leslie, Eskin et al. 2002). These past results have produced accuracies reaching the 99% range. In this paper, we present a novel approach for protein classification based on principal component null space analysis (PCNSA) (Vaswani 2002), a recently developed linear classifier. Our results show very high accuracy, in some cases misclassifying only seven samples of 2,500.

Two textbooks, "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids"(Durbin, Eddy et al. 1999) and "Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins"(Baxevanis and Ouellette 2004) are recommended for further reference on bioinformatics, superfamily classification and remote homology search.

3. Principal Component Null Space Analysis

3.1. Introduction

The main classification approach that we provide is based on Principal Component Null Space Analysis, PCNSA(Vaswani 2002; Vaswani and Chellappa 2004; Vaswani and Chellappa 2004). PCNSA is a linear classifier that takes a multiple perspective approach; each class is used to generate a view or perspective. This perspective is derived from filtering of the minor and major components of the covariance matrices. Removing minor components first reduces noise and dimensionality by performing *principal component analysis* (PCA) on the entire training dataset. The second step then finds a *null space* for each class. The null space is extracted by taking the dimensions with the *least* variance of each class using eigenvalue decomposition (removing the major components). The null space is a subspace of the feature space in which a given class has very little variance. Once theses spaces are computed then classification can be performed. The classification metric used to classify a sample is the euclidian distance at unclassified sample to the mean of each class inside the class null space. The classification rule is based on *Bayes'*; i.e. the unknown sample is assigned to the class that minimizes this distance. Given this rule, PCNSA can classify a dataset that contains more than two classes without additional complexity that other classifiers designed for binary classification would require. Additionally, the distance from unknown protein to the class average in its null space can be used as a score that represents how close the protein is to that class or superfamily.

In PCNSA, the PCA and the null space creation steps reduce the dimensionality of the dataset, or at least keep it the same. This dimension reduction allows for classification

of samples with many features, such as protein sequences. Most classifiers do not cope well with datasets of such dimensionality (also known as the "The Curse of Dimensionality"). The main disadvantages are a slow learning phase and overfitting when the entire feature space is used. PCNSA avoids these problems by carefully reducing the dimensionality.

3.2. Example

An example of PCNSA can be described visually in a 2 dimensional space. Consider three classes of red, green and blue spread across three clusters (Figure 5):

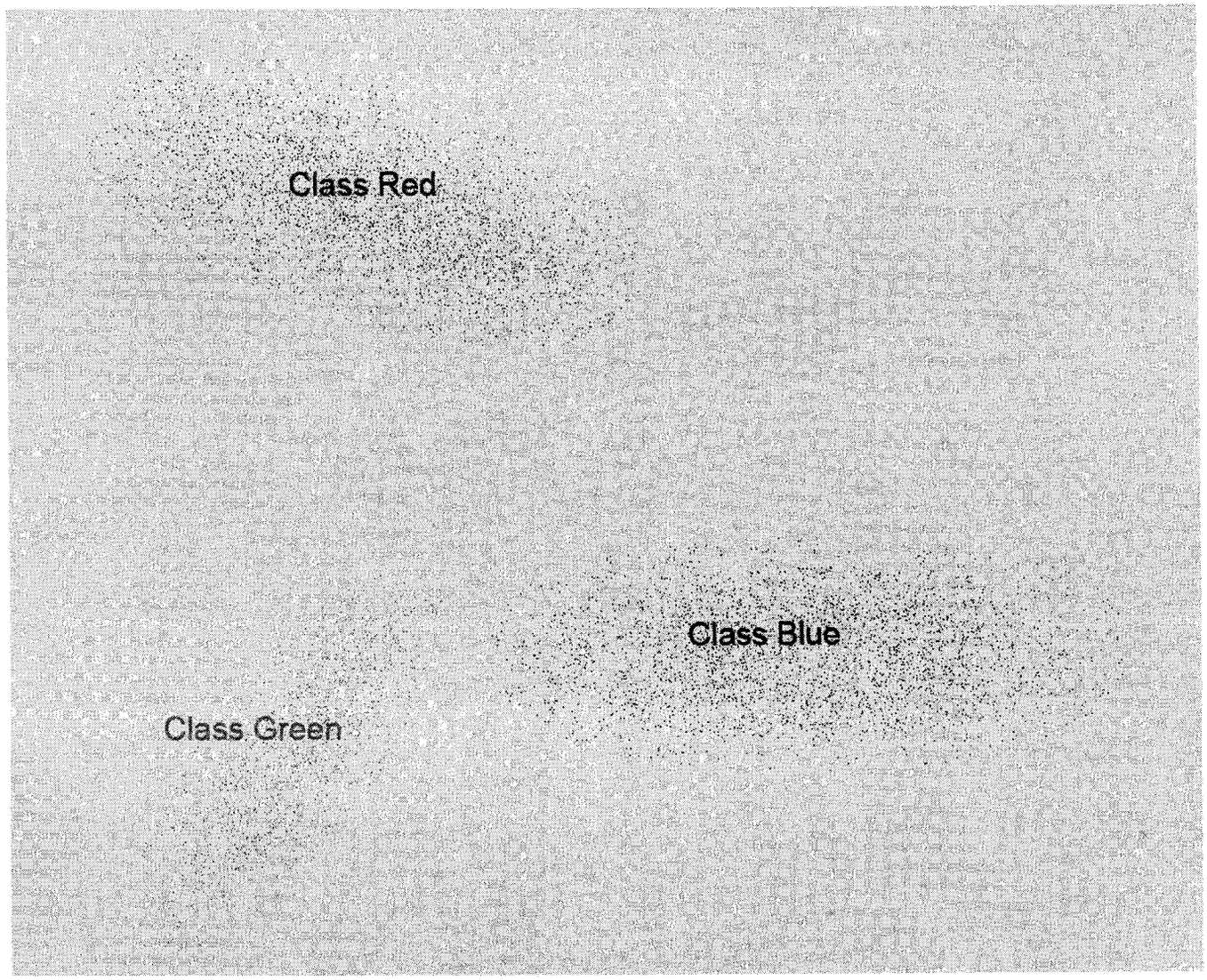


Figure 5 PCNSA example, dataset visualization

Next the three classes are treated as a whole and projected into a new space using PCA (Figure 6). This reduces noise and dimensionality, the size of this new space or the number of top eigen values to project into is a input parameter. In this example the 2D data is projected onto a one dimensional line, here the smaller principle component is discarded:

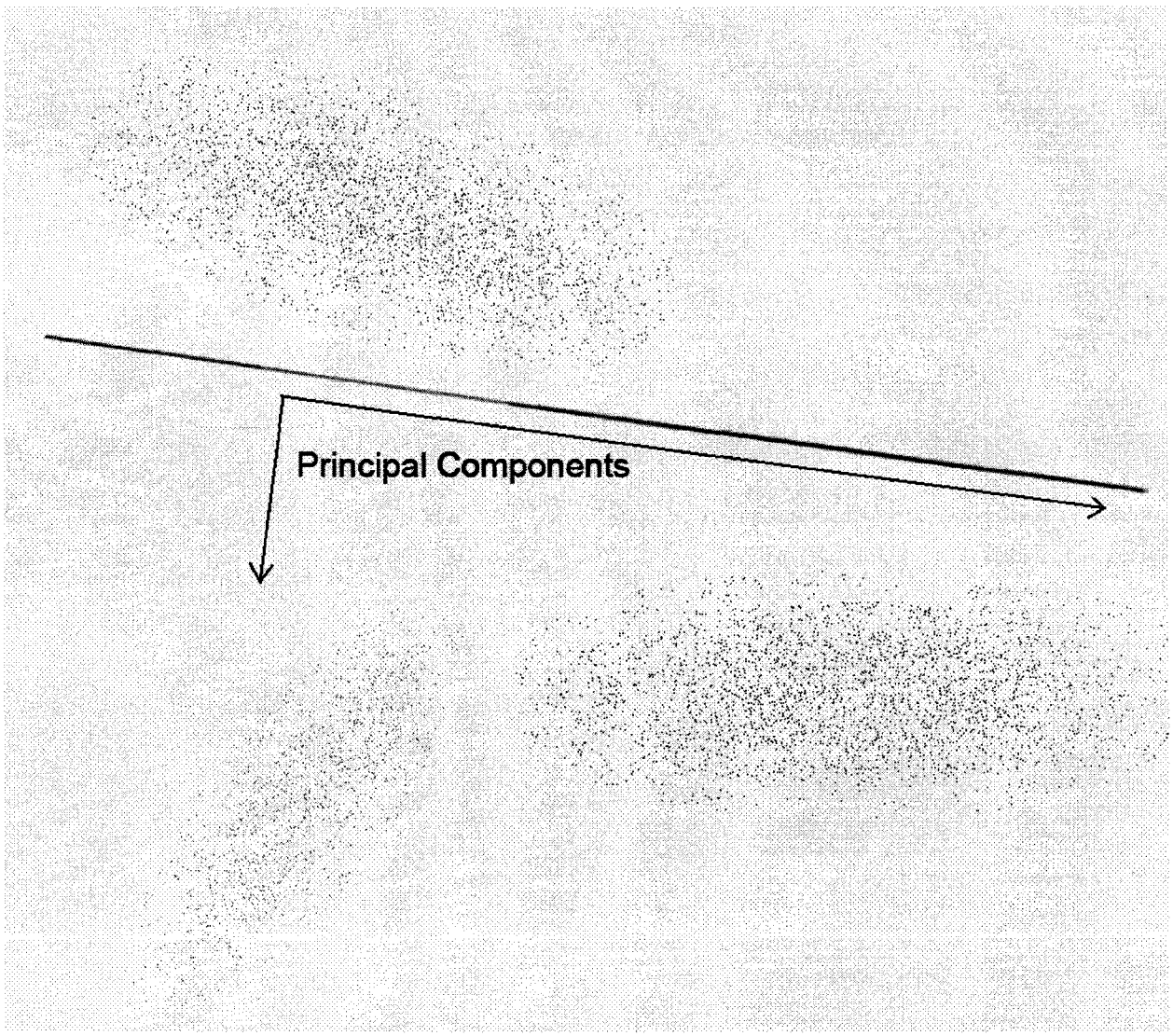


Figure 6 PCNSA example, PCA visualization

At this point in the example the data is one dimensional, which creates problems for

the null space creation stage. To solve this assume the PCA stage brought the data into the original 2D representation, not the single line seen in Figure 6 PCNSA example, PCA visualization.

Next the null spaces are computed for each class (supervised), this is very similar to the PCA stage (Figure 7). The differences are that it is supervised and the minor components are kept, not the major. Here again the dimensionality is brought down from two dimensions to one, so each null space consists only of one vector.

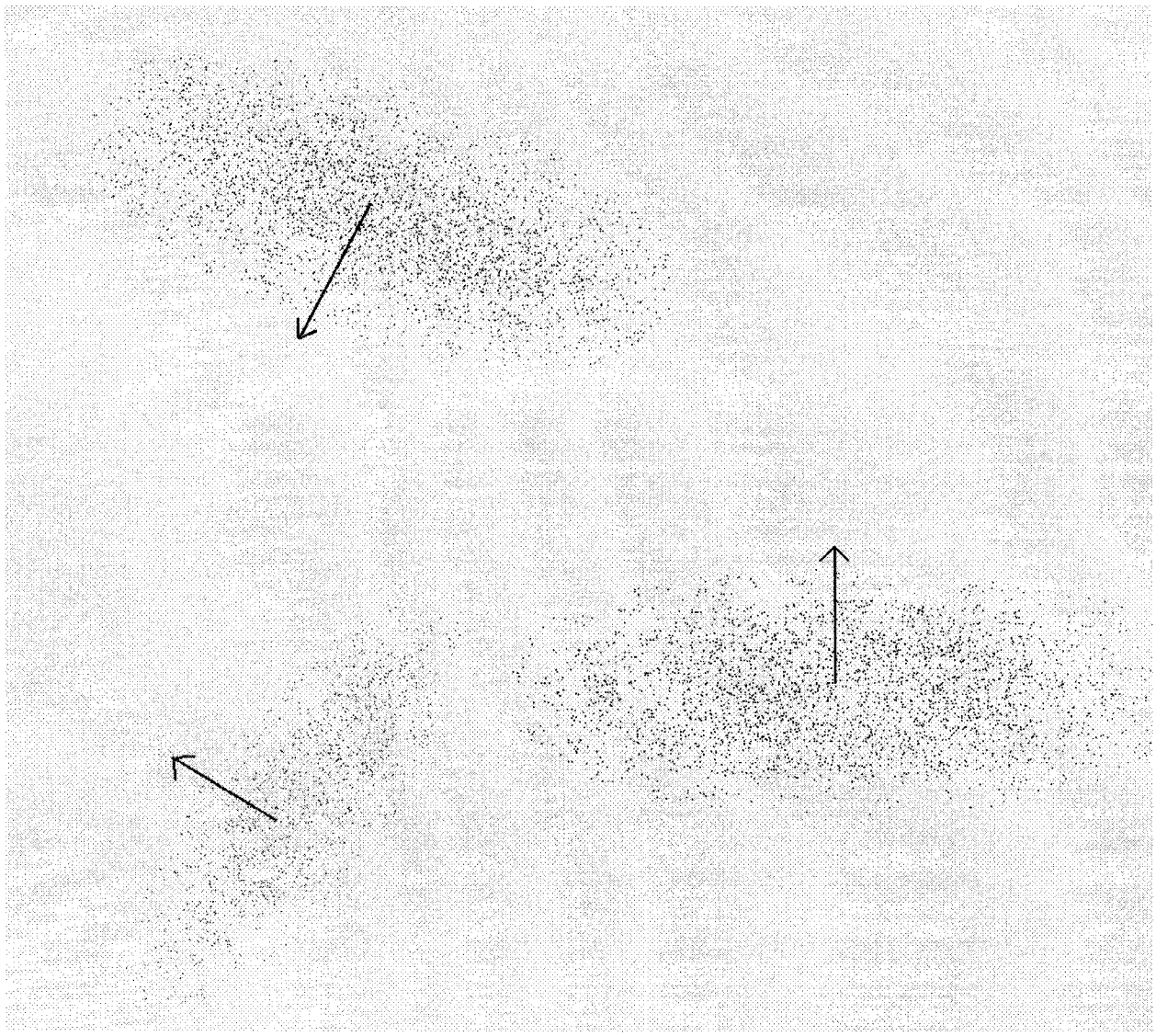


Figure 7 PCNSA example, null space visualization

Now that the training stage is complete, classification can be performed. An unknown sample displayed as an X, is projected onto each null space. In each space the

distance to the class mean of that null space is measured. Visually:

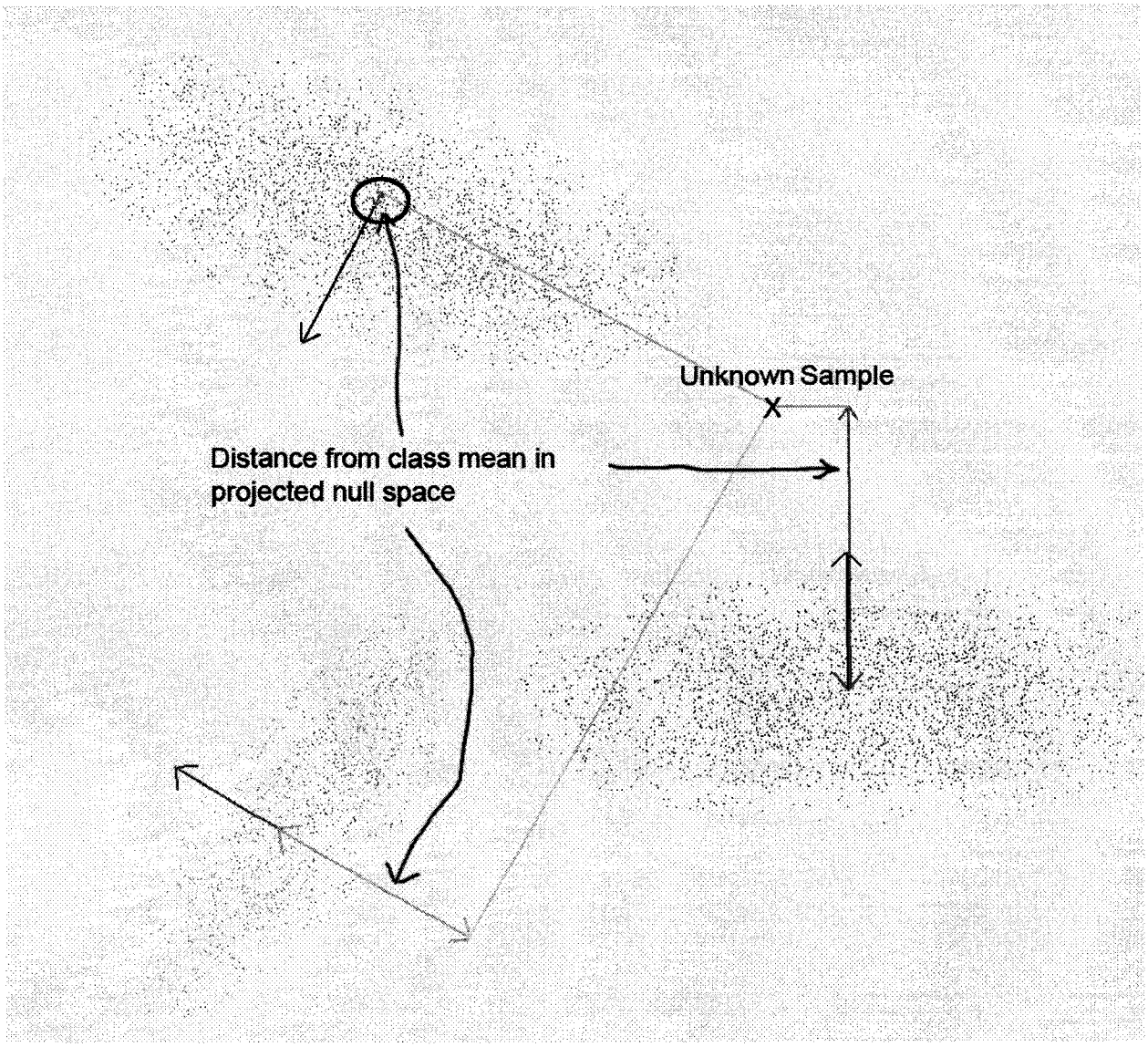


Figure 8 PCNSA example, visualization of null space distances

The class that measures farthest is green, followed by blue giving class red (circled) the predicted class of the unknown sample (Figure 9):

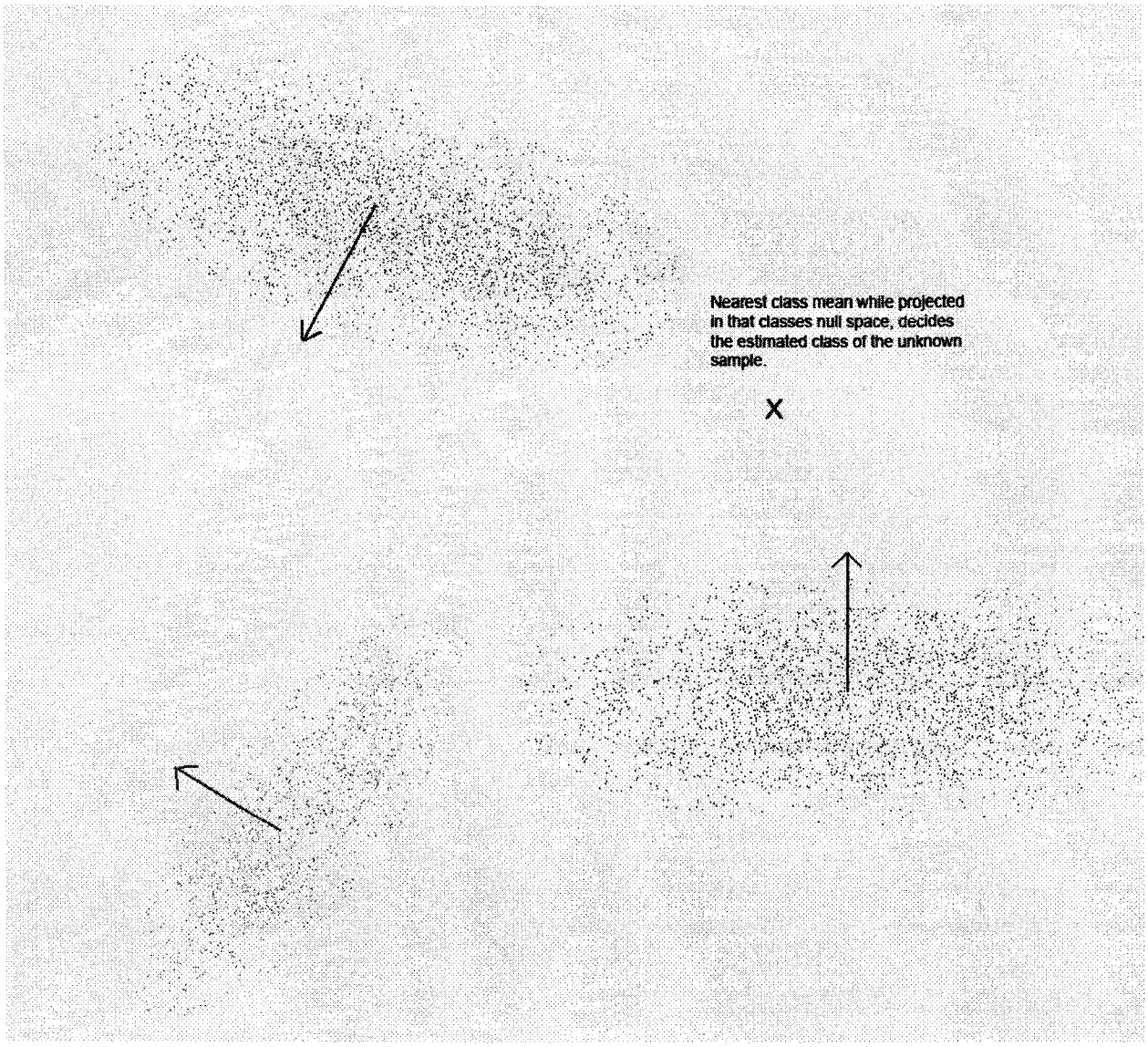


Figure 9 PCNSA example, classification

3.3. Theory

PCNSA has previously been only applied to image and video classification (Vaswani and Chellappa 2004). In this area, PCNSA has proven itself on

datasets that have quite different within-class covariance matrices, such as object recognition, action retrieval and abnormal activity detection (Vaswani and Chellappa 2004). Datasets of this type are referred to as "Apples vs. Oranges" problems, or stated in a different way: unequal and non-white noise covariance matrices. Datasets with similar matrices are referred to as "Apples vs. Apples". In this case, the resulting null spaces are very similar and should result in poor results. Additionally it is important and stated by Vaswani and Chellappa that a large amount of training samples are needed in order to compute accurate null space. Further theoretical analysis including error probability bounds are provided in the PCNSA references.

3.4. Algorithm

The algorithm that we propose is based directly on the PCNSA algorithm(Vaswani 2002); r and s are the two input parameters to our algorithm and specify the dimensionality of the PCA space and null space, respectively. Consider a dataset $D=\{x_1,...,x_n\}$ where $x_i=[x_i^{(1)},...,x_i^{(d)}]^t$ is a d-dimensional feature vector that represents a protein sequence. Our modified PCNSA algorithm proceeds as follows:

1. Normalize every data sample on a feature basis. For each feature/dimension and data sample, x_i, perform:

$$z_i^{(j)}=\frac{x_i^{(j)}-\min\limits_{1\le q\le n}\{x_q^{(j)}\}}{\max\limits_{1\le q\le n}\{x_q^{(j)}\}-\min\limits_{1\le q\le n}\{x_q^{(j)}\}}. \quad (1)$$

2. For the full dataset, compute the sample mean vector and covariance matrix as follows:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} z_i \text{ , and} \tag{2}$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (z_i - \hat{\mu})(z_i - \hat{\mu})^t . \tag{3}$$

3. Obtain the PCA projection matrix (an orthogonal matrix that is used in step 4), W, by taking the eigenvectors corresponding to the r largest eigenvalues of $\hat{\Sigma}$.

4. Project the training samples of each class into the PCA space as below:

$$y_i = W^t (z_i - \hat{\mu}), \tag{4}$$

obtaining a new dataset $D_y = \{y_1, ..., y_n\}$.

5. For each class, ω_k, compute the estimates for the class mean, $\hat{\mu}_k$, and the class covariance, $\hat{\Sigma}_k$ in the PCA space, using (2) and (3), and a data subset that contains the samples which just belong to ω_k.

6. Obtain the approximate null space $(N_k)_{r \times s}$ for each class ω_k as the s trailing eigenvectors of $\hat{\Sigma}_k$. The PCNSA classification matrix for class ω_k, W_k, is formed from these trailing eigenvectors.

7. Classify an unknown sample x, by projecting x into the PCA space as follows:

$$y = W^t (x - \hat{\mu}) \tag{5}$$

Then, assign x to class ω_k as per the following rule:

$$k = \min_{1 \le i \le c} \left\{ \| W_i^t (y - \hat{\mu}_i) \| \right\}. \tag{6}$$

There are various differences between our algorithm and the versions given by Vaswani and Chellappa (Vaswani 2002; Vaswani and Chellappa 2004; Vaswani and Chellappa 2004): two filters on the null space eigen vectors are not used, and the data is normalized. Although it is understood that both of these changes significantly undermine the assumptions and theoretical basis of PCNSA, the following reasons support our modifications.

Normalization was originally performed to aid in accurate tracking of feature weights. This normalization led to the null space failing to check for Σ_i having a high condition number or a large range of eigenvalues when computing the null space. Thus, the filter (eigenvalues $\lambda \leq 10^{-4}\lambda_{max}$) on the null space vectors was removed. We experimentally found that higher accuracy resulted from this change. A second check on the null space was also removed and a parameter was instead used to limit the number of null space dimensions, s, as seen in the original PCNSA paper (Vaswani 2002). These changes result in another variable(s) and removes a variable that was involved in the null space filtering. These changes made for a simpler, faster and more accurate classifier for the protein sequence dataset.

3.5. Time Complexity

The worst-case time complexity of our PCNSA algorithm is $O(nd^2 + d^3)$ where d is the dimension of the feature space and n is the number of samples in the training set. Classification is $O(d^2)$ per test. These time complexities can be lowered depending on algorithms used for matrix multiplication and eigenvalue

decomposition. Computation of a high ranged PCNSA parameter pair (320,297) takes approximately 8.5 minutes for a ten fold cross validation on a 2.0Ghz, 32bit AMD processor. This speed is mainly attributed to the linear nature of the algorithm allowing it to be very 'fast' when compared to other methods such as neural networks and support vector machines.

4. Multispace KL

Multispace KL or MKL is similar to PCA, it is used for dimension reduction and performs in an unsupervised manner. It finds PCA projections for disjoint subsets of the dataset that are best represented by the projections. In comparison PCA finds the principal components of the entire training dataset, MKL finds the k principal components best fitting several subsets of the entire dataset. Multispace KL was developed by R. Cappelli, D. Maio, and D. Malton in 1999. MKL has been applied primarily to fingerprint recognition. MKL can be combined with a classifier to achieve pattern classification. Figure 10 Example of Multispace KL, from Multispace KL for Pattern Representation and Classification(Cappelli, Maio et al. 2001) page 985 shows MKL solutions for cluster sizes of one (same as the PCA solution), two, and three data subsets using a k value of one dimension.

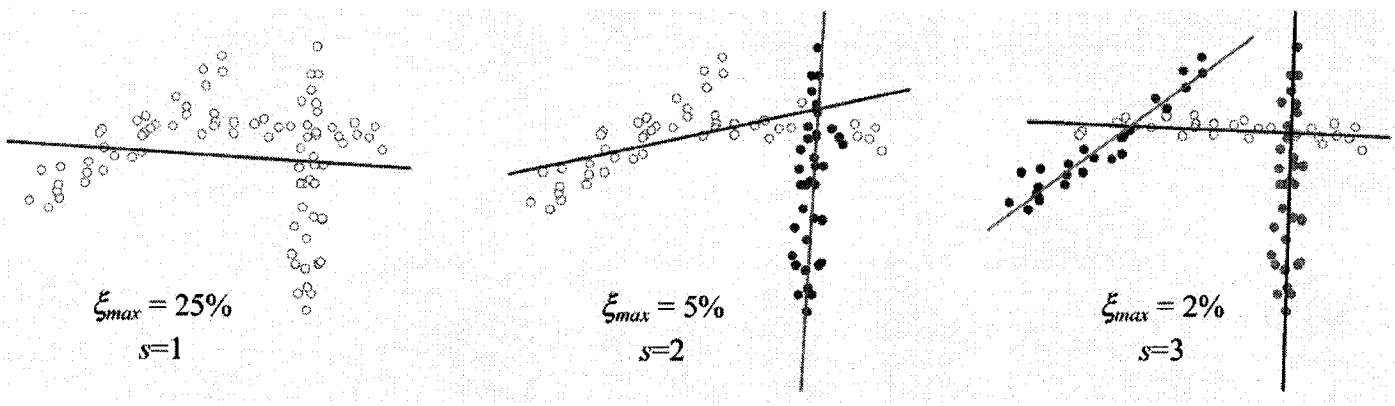


Figure 10 Example of Multispace KL, from Multispace KL for Pattern Representation and Classification(Cappelli, Maio et al. 2001) page 985

The MKL algorithm is similar to k-means clustering algorithms. The training dataset is randomly distributed into a preset number of clusters or data subsets, additionally a

parameter for the PCA space size is needed. PCA is performed on the data subsets and the accuracy to which it represents that set is measured. Points are moved between sets until the error rate falls below a threshold, this implies that data subsets are well represented by their PCA projections. Several rules exist for shifting points into the data subset where it is best represented; a simple measure is to measure the error when represented in the PCA projection for that data subset. This error is computed by projecting into the PCA space, then reverse projecting the data into its original feature space and computing the distance from its original location. Further information about MKL can be found in "Multispace KL for Pattern Representation and Classification"(Cappelli, Maio et al. 2001).

5. Datasets

5.1. Introduction

In this section the design of the several datasets used for empirical testing of the classifiers is given. All datasets consist only of numeric or boolean valued attributes/features. Most of the datasets have a large number of features, ranging from 400 to 700. None of the datasets contain unknown entries. All of the datasets are created via biosequence processing. These sequences usually vary in length, so certain steps are provided to convert a sequence into a fixed sized vector, a necessary requirement for the classifiers tested. First the two protein sequence datasets created from the PIR-PSD and SCOP dataset are provided, then information about the DNA splice site dataset created from the HS3D database is given.

5.2. WEKA Machine Learning Software

A valuable tool for working with the datasets is the WEKA 3 data mining software in Java(Witten and Eibe 2005). WEKA is an open source software suite containing facilities for most areas of pattern recognition and machine learning. For this work it was used primarily for dataset processing, visualization and execution of common classifiers. Several histograms will be provided for the datasets, all were generated by WEKA.

5.3. Protein Sequences

5.3.1. PIR-PSD Dataset

The PIR-PSD dataset is used in most of the classification tests performed. It was created from the protein sequence database (PSD) release 79.05 at the protein information resource (PIR) databank(Wu, Yeh et al. 2003). PSD provides fully annotated protein data in XML format for over 280,000 sequences. For this application, only the sequence, sequence type and superfamily of the entries were used. Some entries in the databank only have the sequence of a protein fragment, or are ambiguous in describing the sequence (e.g. "GLS(D.G.E)WXQL"). All complete non-ambiguous sequences of the four selected superfamily classes were processed. The four classes collected and their size are ras transforming proteins (455), kinase-related transforming proteins (517), globin proteins (672) and ribitol dehydrogenase proteins (868). Although the PIR-PSD database entries contain one or more superfamily classifications, none of the selected data subsets intersect. Two datasets were created: a two-class dataset containing kinase and ras transforming proteins (972), and a second multiclass dataset that includes all four classes mentioned above (2,512).

The string sequence data of each protein was processed to create an array of 465 numeric features plus the class label. At a high level, the features of a vector x that represents a sample are provided in Table 1. All of these features were generated directly from the sequence string. The pI and mass features are estimates based on the polypeptide encoded by the sequence string. Originally, the dataset contained only two-grams and exchange two-grams. As the research progressed, more data was added with the resulting accuracies increasing.

Table 1 Protein dataset feature/attribute list

$x^{(1)}$ = length of sequence

$x^{(2)}$ = isoelectric point (pI)

$x^{(3)}$ = mass

$x^{(4)}, x^{(5)}, ..., x^{(23)}$ = amino acid distribution (20)

$x^{(24)}, x^{(25)}, ..., x^{(423)}$ = two-grams (400)

$x^{(424)}, x^{(425)}, ..., x^{(429)}$ = exchange group distribution (6)

$x^{(430)}, x^{(431)}, ..., x^{(465)}$ = exchange group two-grams (36)

The two-gram features account for the majority of the attributes. They represent the frequencies or buckets of every consecutive "two-letter" sequence in the protein sequence. Two grams have the advantages of being length invariant, insertion/deletion invariant, not requiring motif finding and allowing classification based on local similarity(Wu, Whitson et al. 1992).

Exchange grams are similar but are based on a many-to-one translation of the amino acid alphabet into a six letter alphabet that represents six groups of amino acids, which represent high evolutionary similarity. Exchange groups used for this dataset are: e_1={H, R, K}, e_2={D, E, N, Q}, e_3={C}, e_4={S, T, P, A, G}, e_5={M, I, L, V} and e_6={F, Y, W}. The exchange groups are based on information from the point accepted mutations (PAM) matrix (Dayhoff, Schwartz et al. 1978), which statistically describes the probability of one amino acid replacing another over time.

Given an example sequence "GLALLA" the non-zero two-grams are GL=1, LA=2, AL=1 and LL=1. Translating "GLALLA" to an exchange group sequence results in

"$e_4e_5e_4e_5e_5e_4$", with the resulting exchange two-grams of e_4e_5=2, e_5e_4=2, and e_5e_5=1. The frequency of the amino acids and exchange groups are also added to the dataset entry, and result in G=1, L=3, A=2, e_4=3, and e_5=3. Next, the two-gram counts are converted to *probability estimates* by dividing by the total number of one-grams or two-grams of the sequence. For "GLALLA" the frequencies estimates are: G=$\frac{1}{6}$, L=$\frac{1}{2}$, A=$\frac{1}{3}$, e_4=$\frac{1}{2}$, and e_5=$\frac{1}{2}$, and the two-gram counts become: GL=$\frac{1}{5}$, LA=$\frac{2}{5}$, AL=$\frac{1}{5}$, LL=$\frac{1}{5}$, e_4e_5=$\frac{2}{5}$, e_5e_4=$\frac{2}{5}$, and e_5e_5=$\frac{1}{5}$.

Figure 11 visualizes the conversion of the PIR-PSD xml database into the final numeric dataset. The datatype is followed by an example data snippet, then the method used for conversion:

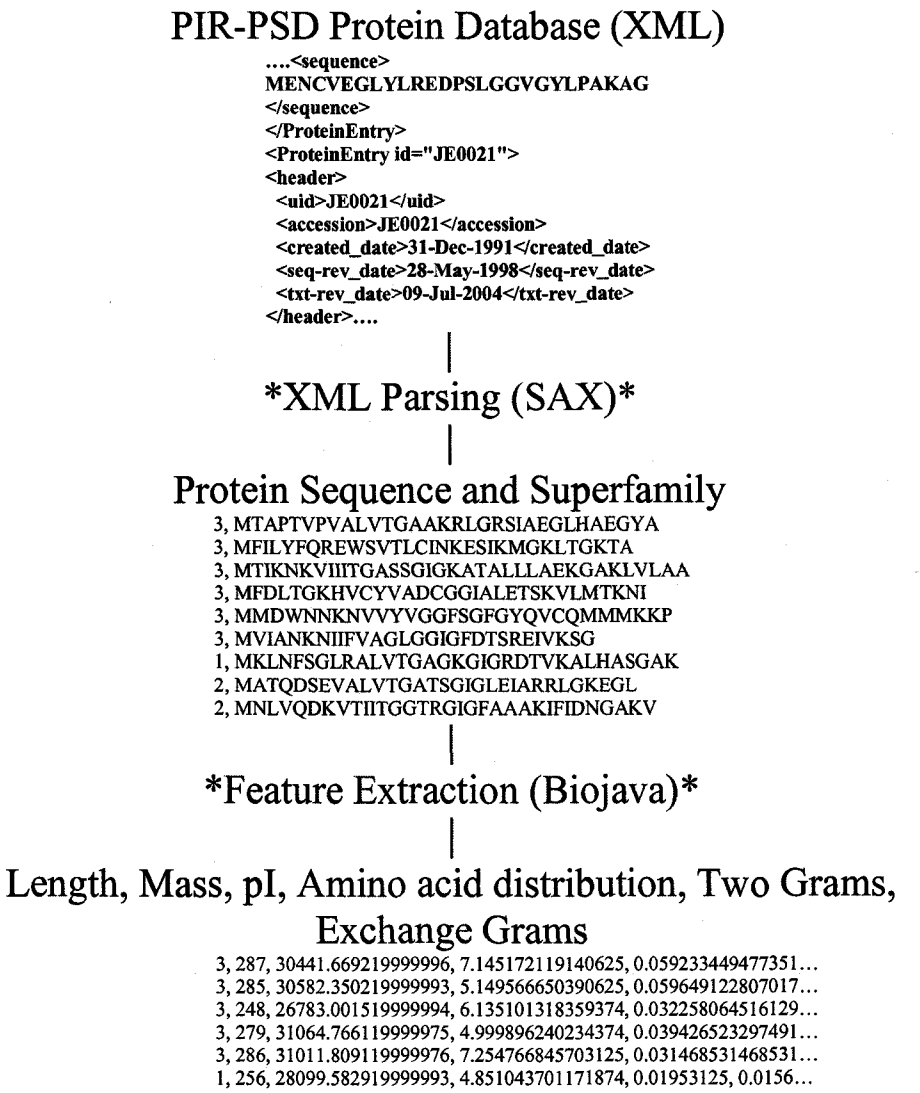


Figure 11 Path of dataset processing, from PIR-PSD XML database to final attribute relation file format.

The full two-gram encodings result in a very sparse dataset[2], with some features having a zero frequency value for over 85% of the instances. With the example of "GLALLA" there are $(20^2-4)+(6^2-3)=429$ zero-valued two-grams. The shortest

[2]the term "sparse" refers to a matrix with a large percentage of zero valued entries.

protein sequence in the dataset is 63 amino acids in length, which results in at most 62 non-zero two-grams out of 400, further demonstrating the sparseness of the dataset. These zero based entries suggest clues to why PCNSA performs so well on the n-gram protein sequence dataset. Past work has reduced the two-grams given to the classifier in order to decrease training time. In this thesis all of the described features are given as input to the PCNSA algorithm.

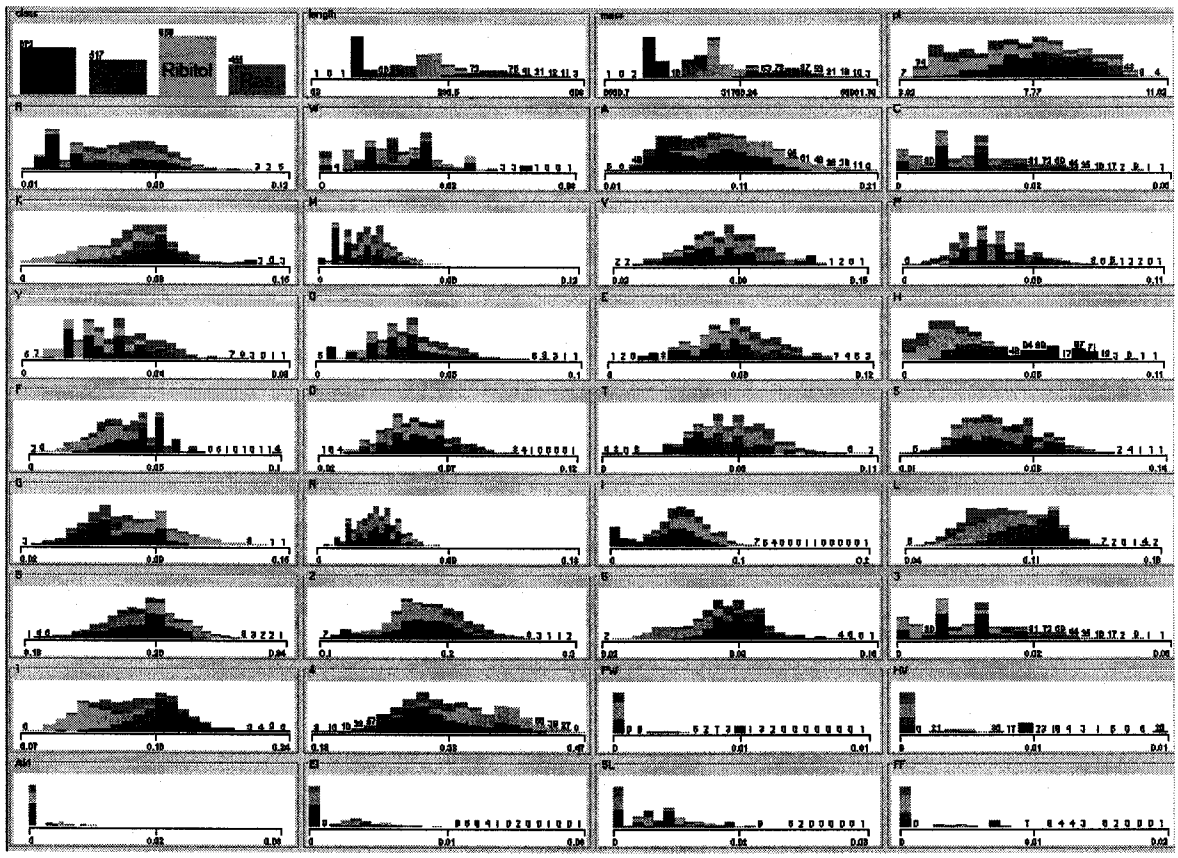


Figure 12 Histograms for several attributes of the PIR-PSD dataset

5.3.1.1. Rueda and Ngom Dataset

This dataset was provided by Luis Rueda and Alioune Ngom who previously used it for work on Fisher's classifier titled "An Empirical Evaluation of the Classification Error of Two Thresholding Methods for Fisher's Classifier"(Rueda and Ngom 2004). The dataset consists of a training set of size 731 and a test set of size

407 consisting of sequences from the kinase and ras superfamilies. The dimensionality is 50 features based on extracted two-grams, the full two gram dataset was reduced by filtering features according to a distance metric described in "New Techniques for Extracting Features from Protein Sequences"(Wang, Ma et al. 2001). The dataset was derived from PIR-PSD release 62.

5.3.1.2. Highest and Lowest Ten Dataset

The highest and lowest ten dataset consists of the four class PIR-PSD dataset based on the weights PCNSA gives to each feature. This was done to make the dataset more manageable for testing with other classifiers. This was done by extracting the highest weighted dimensions that PCNSA uses to classify. Additionally a contrasting dataset was created that took the lowest weighted dimensions used by PCNSA. This provides grounds for a hypothesis:

> The dataset based on the highest weighted attributes, deemed most important for classification by PCNSA will provide higher accuracy than the lower weighted attributes. The accuracies refer to the precision data produced when other classifiers are trained and tested with the two contrasting datasets.

Intuitively this makes sense, given PCNSA achieves 99.5% accuracy on this full dataset then its weights used in its computations must provide good discrimination. It is also hypothesized that all tested classifiers will achieve lower than 99.5% accuracy.

Extracting the weight values of PCNSA was not a simple task. First the dataset was normalized so that the weights would be in proper scale, next PCNSA was run to find the optimal choice for the PCA keep value and Null space size parameters. The

parameters of 200 and 150 were chosen to produce the weight vector, the highest ten weights in each null space for each class were added to a set as to remove duplicates and the same for the lowest ten sorted by absolute value. This resulted in 26 attributes on the high side, and 37 on the lower. This shows that there were many duplicates for the best attributes, while the lower attributes were more diverse. Below is the table of the attributes used for each dataset, in unsorted order.

Table 2 Highest and lowest weighted attributes, r=200 s=150

Highest Weighted	C, I, T, E, Y, N, A, 3, 2, 4, 1, NE, CW, SS, WQ, YE, WT, EY, KS, HH, MM, KK, QQ, AH, WW, NN
Lowest Weighted	IW, EM, ED, ER, QT, PD, AT, GF, YS, DP, NG, NP, MP, GQ, FR, YQ, TM, MG, QP, IY, VT, EH, PQ, QL, TV, SF, VI, SK, WS, FI, RF, FN, DI, MT, LP, ES, GD

Highest Ten Dataset

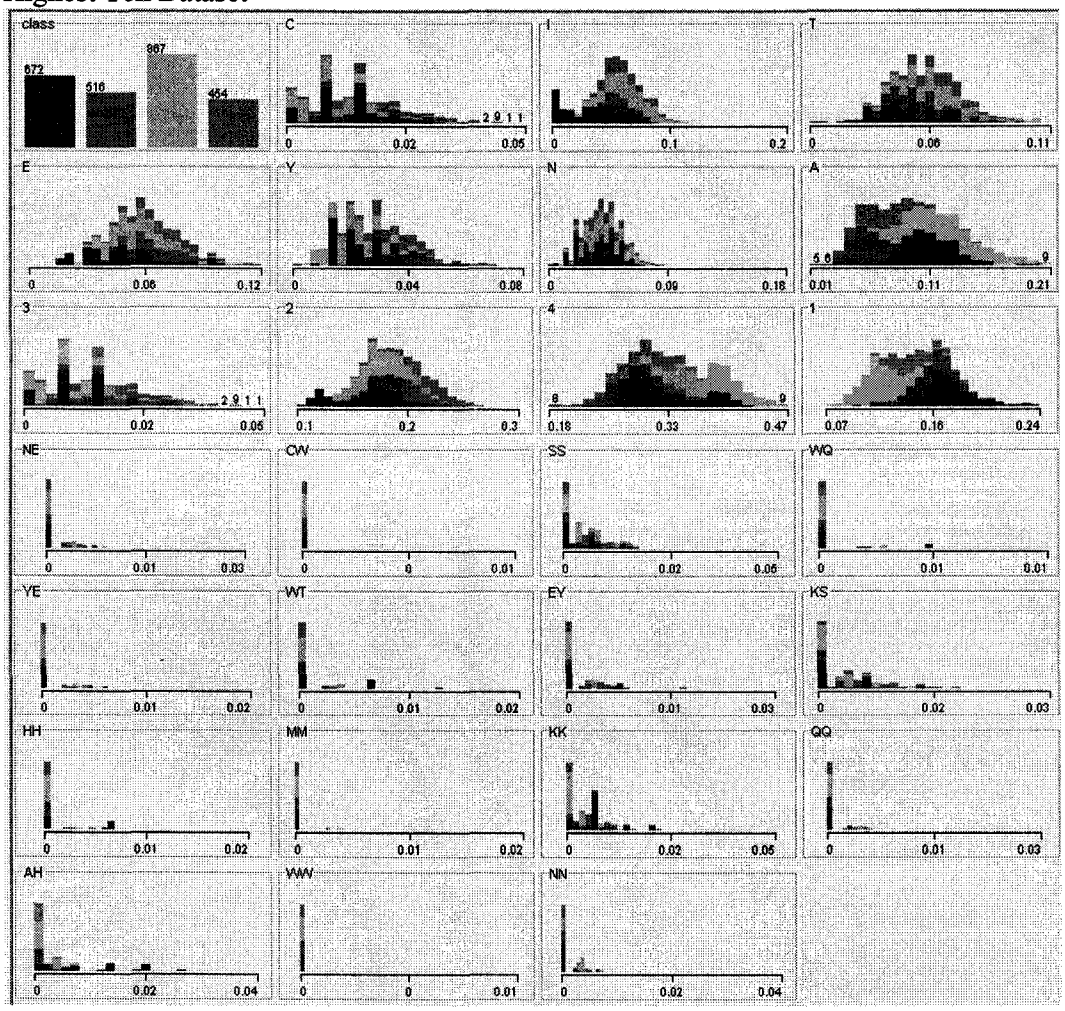


Figure 13 Histograms for all features in the highest ten dataset

Lowest Ten Dataset

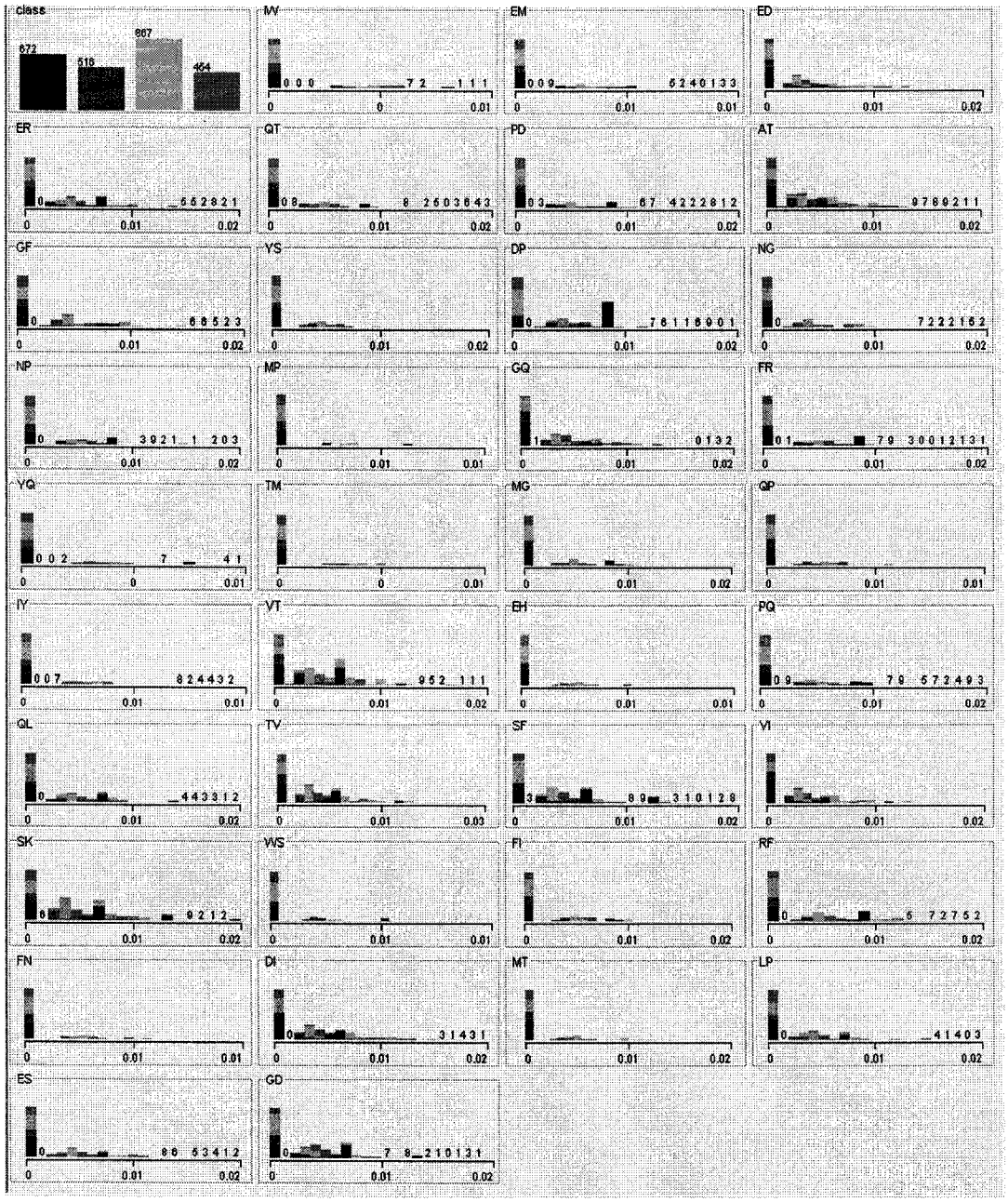


Figure 14 Histograms for all features in the lowest ten dataset

One significant note about both datasets is the large amount of zero's for most attributes; this suggests other approaches may be suitable for formatting the data. From the diagrams, the higher weighted attributes seem to suffer less from this high distribution of zeroes.

5.3.2. SCOP G Proteins Dataset

The Structural Classification of Proteins (SCOP) Database also provides a superfamily classification of proteins(Murzin, Brenner et al. 1995). SCOP varies from the PIR-PSD database in many ways. SCOP is based upon PDB90, PDB 90 is a subset of the Protein Databank where no two sequences have lower than 90% sequence similarity. This means that each entry is not a sequence but a homolog representing several similar protein sequences.

The SCOP G Proteins dataset is based upon the dataset used in "A Discriminative Framework for Detecting Remote Protein Homologies"(Jaakkola, Diekhans et al. 2000). The recreated version of the dataset used in this thesis contained only the training and test sets for the G Protein family. The G proteins family is one of 33 protein families used in original work by Jaakkola et al.

The dataset is unique in that it tests the ability of the classifier to recognize a previously unknown family; this is achieved by providing carefully selected training and test classes. The positive training classes do not include any sequences from the test family, only those in the same superfamily.

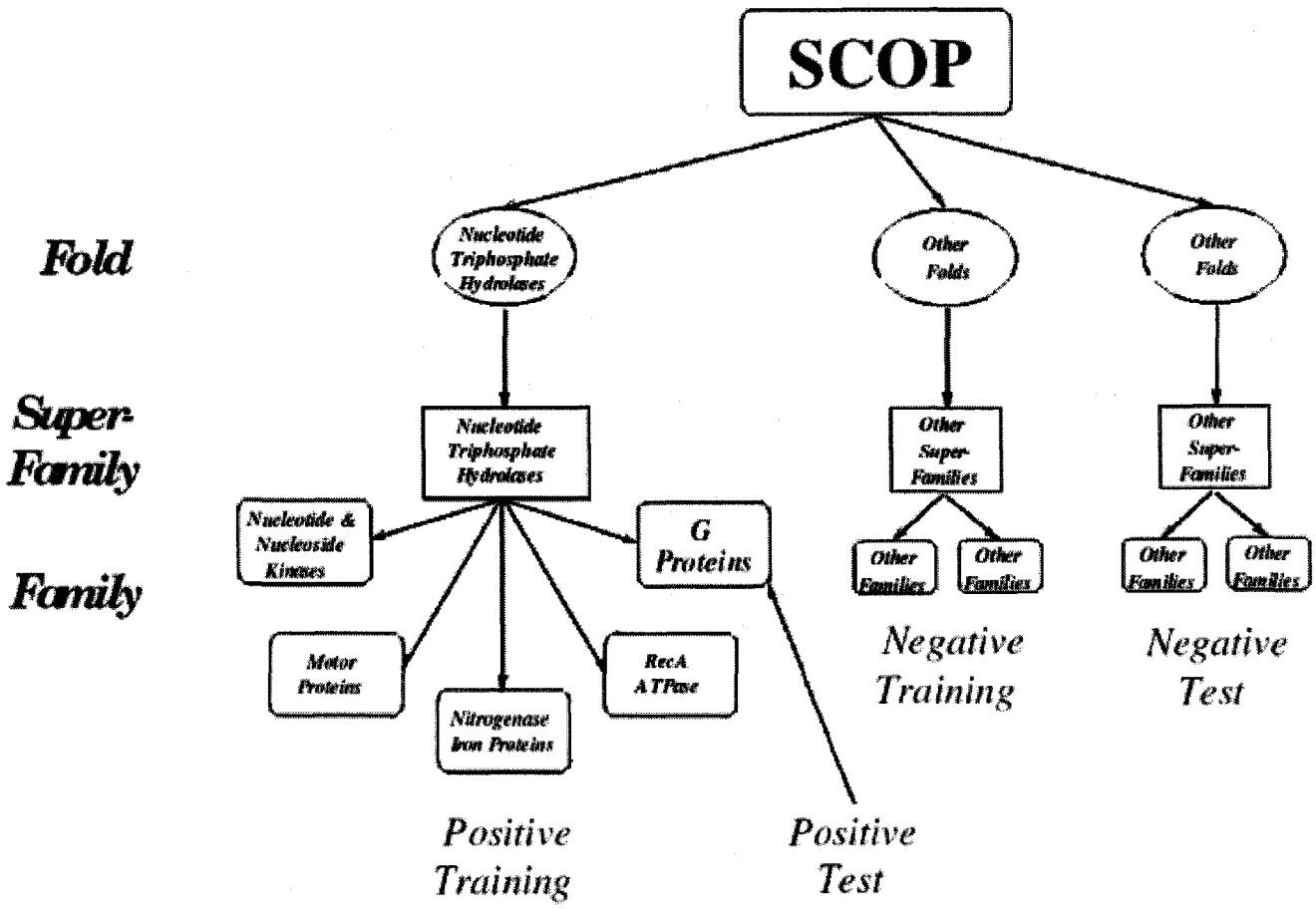


Figure 15 Training and Testing sets for G Proteins test, from page 105 of "A Discriminative Framework for Detecting Remote Protein Homologies" (Jaakkola, Diekhans et al. 2000)

The positive sets are very small, due to small size of the PDB90 database. The positive test set (G Proteins family) consists of eight sequences, and the G positive training set consists of nineteen in the SCOP only dataset. The SCOP training set can be extended by using homologs extracted from a second database. These homologs from the non redundant protein database (NRP) were found using SAM-T98(Jaakkola, Diekhans et al. 2000) and provided in the Jaakkola et al. dataset. Jaakkola et al. used these sequences to generate SAM-T98 hidden Markov models which were then used to create features(Jaakkola, Diekhans et al. 2000). These additional sequences form

the SCOP extended dataset, its only difference is that it has an additional 3816 positive training sequences. In the results section the normal SCOP dataset results are denoted as PCNSA, while the extended SCOP dataset is referenced as PCNSA-Hom.

The protein sequences are converted to numeric data using the same methods as the PIR-PSD protein sequence conversion. All "X" wildcard amino acid symbols were encountered and removed instead of removing the entire sequence to maximize the size of the dataset. Two negative sequences containing the non-standard amino acid symbol "Z" were removed, "Z" did not occur in the positive sets. Further details regarding this dataset are available in "A Discriminative Framework for Detecting Remote Protein Homologies"(Jaakkola, Diekhans et al. 2000).

5.4. HS3D Splice Sites

The Homo Sapiens Splice Sites Dataset (HS3D) was created by P.Pollastro and S.Rampone for gene recognition benchmarking purposes(Pollastro and Rampone 2003). HS3D provides samples of exon and intron splice regions from the human genome.

The splice regions are windows of DNA around a splice site. A splice site separates an intron region from an exon region. Introns are known as junk or non-coding DNA, this DNA does not code for protein and its purpose is unknown. Conversely, the exon region is DNA that does code for the protein that the gene produces. When a gene is translated into a protein these regions are taken or spliced out, the goal is to recognize

these splice sites.

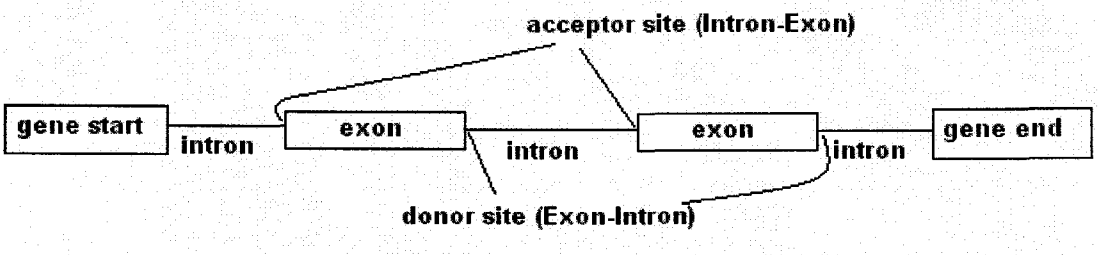


Figure 16 DNA strand showing splicing regions and sites

The dataset provides windows of DNA that are 140 nucleotides in length. A DNA nucleotide is just a letter A, G, C or T that represents the DNA molecule at that point in the strand. The windows are centered on the splice sites, all donor sites in this dataset are followed by GT and all acceptor sites are preceded by AG. For example:

TCCCATTGGTGGCAGCCAGTGCCACCATGCGCGCTCAGT*GTAAGTATCATTCCCTCTCACTGTCCTGGAGAGGAC
GTCCGTATCATATTAGGCGCTGTATGACAATCTCCATTC*GTAAGTACCTCTTGGTCATTTGGACACATTGTAGAT
GAGGCTGCTGCAGTTTGGGATCGTGGTCTATGTGGTAGG*GTAAGAGAGAAGAGCTTTTGGCCAGGCTGGAGGGGC

The stars above represent the donor splice site also known as EI junctions. Trailing GT's are displayed in bold, note that the first sequence above has 5 other occurrences of GT. Although there is also a trailing AAG in the above sequences this does not always occur. Also GT+AG rule is true in 99% of the splice sites found in nature but not all. Clearly the GT+AG rule cannot be used by itself to locate splice sites because not all GT+AG pairs indicate true sites, and they occur much more often than the splice sites given that the DNA alphabet is only four letters.

The dataset gives 2796 true donor splice sites based on proven annotation data from the GenBank database. Another 271,937 false donor splice sites are given. The large amount of false sites are given to properly represent the real life ratio of true to false sites based on the occurrences of the GT nucleotide pair which is < 0.015. The paper

titled "HS^{3}D, Homo Sapiens Splice Site Data Set"(Pollastro and Rampone 2003) is a good source for more information about this dataset. The dataset created for testing on PCNSA only contained ~1% (2881) of the false donor splice sites, this resulted in a 50/50 true/false split. This split, although not representative of the true problem, allowed for easier accuracy measurement and drastically reduced runtime.

The features for the dataset are similar to the one and two grams of the protein dataset, except the size of the grams are extended as high as five nucleotides in length. Since the DNA alphabet is only size four, a 4-gram feature set would only be 256 in size. Another added feature/attribute is the coding of a nucleotide at a specific position, this is feasible for the DNA sequences because the splice site regions are fixed in size, unlike the protein sequences. Four bit encoding was used to convert the DNA nucleotides into numeric data: A->1000, G->0100, C->0010, and T->0001. Using this method each position takes up four features and increases the number of zero valued entries. Several datasets were created from the splice sites by varying the features used. For example a dataset was created that consisted of 1, 2, 3, and 4 grams for the regions before and after the splice site, and position encoded features representing nucleotides from 69 to 75 for a total of 700 attributes. The configuration of these features was based on past research into splice sites, the three different datasets are described in Table 3 HS3D DNA dataset descriptions and features.

Table 3 HS3D DNA dataset descriptions and features

Dataset	Dimension	Features
EILetters	396	Nucleotide codings for positions 0-100
EIGramsLetters	340	1-4 grams of splice window from position 1-69 (before the splice site)
	28	Nucleotide encodings for positions 12-20
total=	**368**	
EIGramsHalfHalf	340	1-4 grams of splice window from position 1-69 (before the splice site)
	340	1-4 grams of splice window from position 72-120 (after the splice site)
	20	Nucleotide encodings for positions 69-75
total=	**700**	

6. Implementation

6.1. Introduction

Implementation was the most time consuming task of the thesis and produced a large amount of code. Software produced was designed for converting and creating five different data sources. Both the Multispace KL and PCNSA algorithm were fully implemented. Primarily the language used was the Java programming language versions 1.4 and 1.5 (5.0). Open source API's and libraries were used wherever possible. The Perl scripting language was used for simple conversions of datasets, execution of other classifiers and fold creation.

6.2. Biosequence Feature Extractor

The Protein Feature Extractor performs the conversion from string sequence data and its encapsulating format into the numeric feature based datasets described in section 5. The software for this purpose was actually written three times as features were added. The first version was completed in perl and was very simple and crude, it computed only one and two grams from input sequence strings.

The second version was written in Java and was more extensible. It took the main PIR-PSD formatted database in XML format as input and outputted a better structured output file in the form of WEKA's attribute-relation file format (arff). By producing the WEKA based file formats it was possible to visualize data and perform classification using WEKA's built in facilities. This version also made use of the Biojava API for bioinformatics. Biojava allowed the easy addition of the mass, and

isoelectric point features to the dataset. A detailed report documenting the creation of this version of the feature extractor is available in Appendix B.

The third version of the feature extractor is an improved design and was extendable to DNA sequences. Unlike previous versions it used primarily the Biojava API for computing the grams which reduced the size and complexity of the program. The flexibility of this version is demonstrated by its ability to generate n-grams, grams of any length. Its usage on the DNA splice regions made use of the n-grams. The DNA alphabet is much smaller, hence grams of size 4 or 5 can be used without drastically increasing the dimensionality of the dataset. Since the DNA splice regions are of constant size a position specific attribute was implemented. Additionally this third implementation was used to test and verify the output of the second java implementation.

The worst-case time complexity of these programs is $O(nl)$ where n is the number of sequences matching the selection criteria and l is the length of the longest sequence.

6.3. PCNSA

The PCNSA algorithm was originally implemented in Maple version 8 in order to quickly assess its ability on the protein dataset. The maple version proved PCNSA's ability to classify the protein dataset with its first accuracy score of 98.5%. The problem was that the maple implementation was very slow, it took over 24 hours to complete a single run of the algorithm. No clear speed optimizations in the Maple version existed, so the choice was made to move to another platform. The Java

platform was chosen for its ease of code reuse, portability, fast development cycle, and modest execution speed. Rewriting the algorithm in java helped to discover many small errors in the original Maple version, which increased the correctness of the second version. By moving to the java platform, the runtime was reduced to minutes.

The Colt high performance scientific and technical computing package for fast matrix operations was used extensively in the implementation of PCNSA (Hoschek 2000). Colt provided an API interface fast matrix operations used in the PCNSA algorithm, including eigen value decomposition (EVD).

In order to increase the performance of exhaustive searches in the PCNSA parameter space, a cache for covariance matrices was added to the PCNSA program. This cache simply checks if the matrices have been already computed for this dataset, fold and PCA parameter, and if so loads them from memory. This drastically increases the memory needed whilst increasing the speed of iterative runs with similar parameters.

Parts of the original PCNSA algorithm described by Vaswani and Chelleppa were implemented, and tested but not used extensively. These parts include new class detection and two null space filters. New class detection is designed to detect new classes from data instances that are far from all the null spaces computed for the classes. Two filters for the null space were implemented but not used, these are a condition that the eigen values had a certain scale and that the vectors were of a certain distance from other class means. Testing was computed with and without these extra abilities on, and it was found that they had little effect on the end result, while making the algorithm slower and more complex.

6.3.1. K-Nearest Neighbour

An extension to the PCNSA algorithm was explored that changed the classification rule. Instead of taking the class of closest mean in the null space, a nearest neighbour approach was designed. For each training sample, the distance from it to the unknown sample is measured in the null space for that class. These distances are then sorted (closest first) to find the majority class at k nearest neighbours. This function was originally derived in an effort to improve performance on the DNA dataset.

6.3.2. Attribute Tracker

In order to create the highest ten and lowest ten dataset an attribute tracker was created. This program computes and sorts the highest and lowest weighted attributes used by PCNSA for a given run. The computation is achieved by multiplying the two projection matrices together then multiplied by a vector of all ones. Functions are included for outputting the highest and lowest then weighted attributes.

6.3.3. Score Function

The G Proteins dataset and the experimental setup used in Jaakkola et al. required a classifier that computed a score value of a sequence. The lower the output score value the closer that protein sequence is to a positive target protein superfamily. This functionality was added and is the distance of the protein sequence to the class mean of the positive protein family in its null space.

6.4. *Multispace KL*

Multispace KL or MKL was implemented in Java according to the algorithm in "Multispace KL for Pattern Representation and Classification"(Cappelli, Maio et al. 2001). The 2D-Alignments method for cluster initializations was not implemented, two other methods were one random and Iterative-Removing. The use of Java 5.0 and code re-use from PCNSA allowed MKL to be developed in a short period of time.

6.5. *WEKA*

WEKA(Witten and Eibe 2005) was used extensively in this project as previously mentioned in the dataset section. WEKA became more useful for classification when combined with MKL and the highest and lowest ten database. Its open source nature allowed its functions to be called programmatically given its API .

WEKA's built in classifiers were used to perform experiments on the lowest and highest ten dataset. These classifiers and the descriptions from the WEKA API documentation are:

- BayesNet: "Bayes Network learning using various search algorithms and quality measures."
- NaiveBayes: "Class for a Naive Bayes classifier using estimator classes."
- REP Tree: "Fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting)."
- ID3 Tree: "Class for constructing an unpruned decision tree based on the ID3 algorithm. Can only deal with nominal attributes."

- MultiLayerPerceptron: "A Classifier that uses backpropagation to classify instances."
- IB1: "Uses a simple distance measure to find the training instance closest to the given test instance, and predicts the same class as this training instance.", also known as nearest neighbor.
- IBK: k-nearest neighbor

These built in classifiers were also combined with MKL to create an advanced classifier, see section 7.4 for further detail. WEKA's classifiers were needed because MKL is an unsupervised pattern representation algorithm by itself.

7. Experimental Designs

7.1. PIR-PSD Protein Dataset

The primary focus of this work was the two and four class PIR-PSD dataset. All experiments performed on the PIR-PSD dataset used ten fold cross validation, at least once and some of the results are computed using 5 runs or 10 runs of ten fold cross validation. Ten fold cross validation is performed by first shuffling the order of the entire dataset, then splitting it into ten folds. Ten runs of the classifier are performed using each fold as a test set, and the remaining folds as the training set. Using this setup each entry is tested once and used to train the classifier nine times. Each run has a 90%/10% training/testing split which provides a relatively large training set. Many results are also based on several runs of ten fold cross validation, this provides statistics regarding the stability of the algorithm. Obviously, this increases the time spent running the tests by as much as 100x when ten fold cross validation and ten runs are performed.

The main experimental parameters that were tested on PCNSA and all the other datasets was the value of r and s. These two parameters represent the size of the PCA space in terms of dimension and the size of each null space. Experiments were most often performed by ranging the r value from 5 to 465 by 5, and the s value from 5 to r by 5. This setup provides very complete survey of the parameter space as 465 is the dimensionality of the dataset and r is the maximum value for s ($465 \geq r \geq s \geq 1$). Many other experiments were performed on the PIR-PSD dataset, including:

- Adding and removing dataset features, for example using only 2-grams
- Skipping normalization step of the algorithm
- Enabling new class detection
- Disabling null space filters
- Testing k-Nearest Neighbour PCNSA based approach
- Enabling the attribute tracker add-on for information about attribute weights

Usually these experiments were performed in addition to searching the parameter space. By performing these experiments further insight into how PCNSA performed on the protein dataset was revealed.

7.1.1. Two Class and SVMLight

For comparison purposes a Support Vector Machine classifier was applied to the same dataset as given to PCNSA. The dataset used was the PIR-PSD four class dataset, as SVM cannot perform multiclass detection directly. For this comparison ten fold cross validation was used, the exact same folds were given to both classifiers to ensure both had the exact same training and testing sets. Several parameters of SVMLight were explored, results of the best performing choices are reported in section 8.2.1.

7.2. Highest and Lowest Ten Dataset

The highest and lowest ten dataset was tested on four classifiers implemented in WEKA:

- BayesNet: "Bayes Network learning using various search algorithms and quality measures."
- NaiveBayes: "Class for a Naive Bayes classifier using estimator classes."

- REP Tree: "Fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting)."
- ID3 Tree: "Class for constructing an unpruned decision tree based on the ID3 algorithm. Can only deal with nominal attributes."

Each classifier was tested with the same ten folds and with default classifier parameters. WEKA's Experimenter application which is a GUI based workflow, below is a screen shot of the Highest and Lowest ten experimental design:

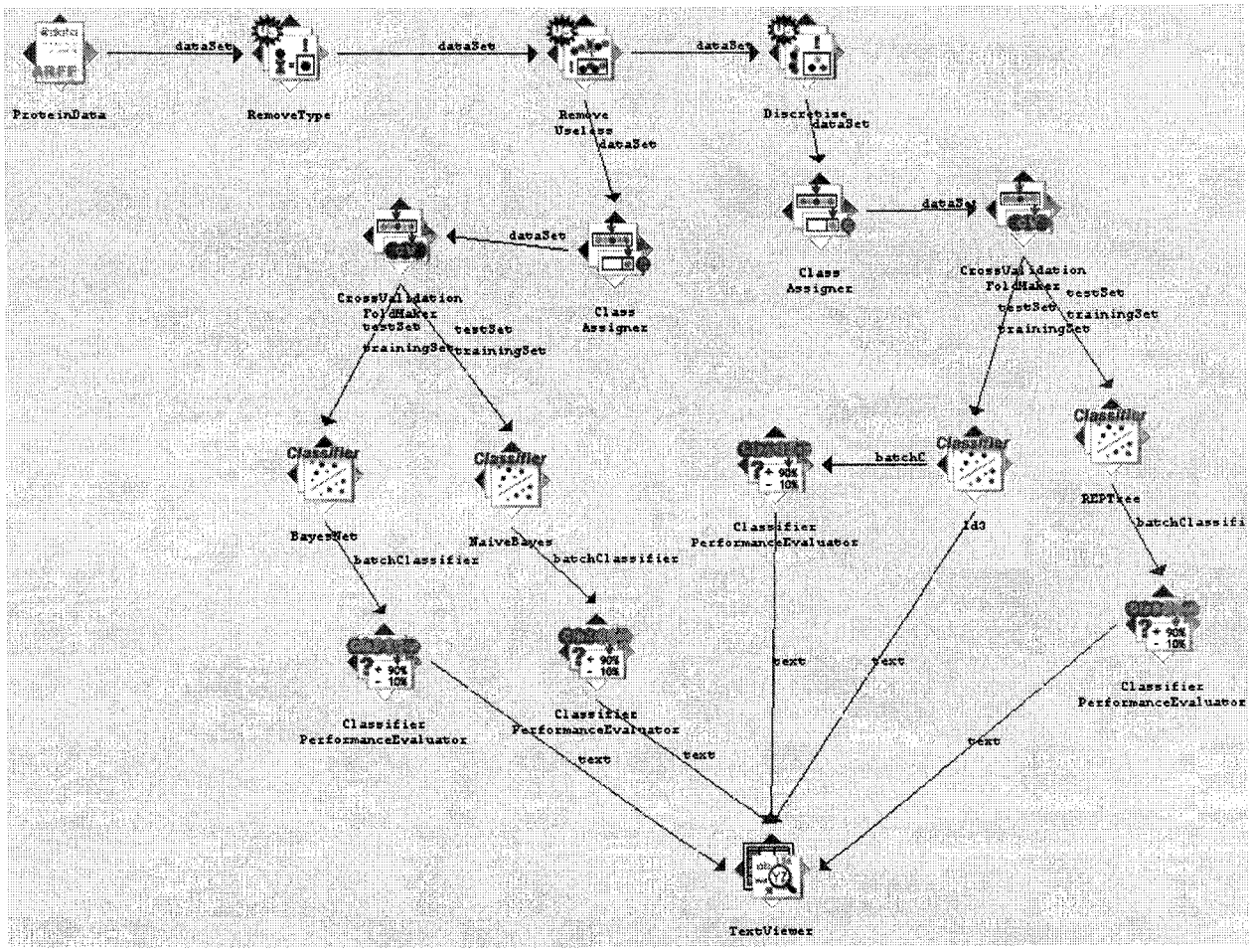


Figure 17 Experimental design of highest and lowest ten , presented as a workflow diagram

Here the process begins with the Protein dataset and the arrows show how the dataset

is cleaned of string and useless attributes. Next the dataset is forked into NaiveBayes and BayesNet using a ten fold cross validation. The other side of the fork is sent to a function that converts numeric data to discrete data for use in the Id3 and REPTree classifiers which only accept discrete data. The final results of all four classifiers are then sent to a TextViewer for recording.

7.3. SCOP G Proteins Dataset

The goal of the experiment as described by Jaakkola et al. is to test "The ability of the methods to distinguish the 8 PDB90 G proteins from 2439 sequences in other SCOP folds"(Jaakkola, Diekhans et al. 2000). The SCOP G Proteins experimental layout is very different from the other experiments. It is a two class problem (negative versus positive) where the samples are ranked according to a score, not classified. The performance metric measures how many negative samples scored above a positive sample also known as rate of false positives (RFP). The median and max RFP scores for a protein family are also used to measure performance. Only the G Protein superfamily was tested in this dataset, providing only small exploratory results. Cross validation is not used because the training and testing datasets are clearly defined, see subsection 5.3.2 for more details.

The r and s parameter space for PCNSA was searched for the best performance by increments of 5, similar to those performed on the PIR-PSD dataset. Both the SCOP dataset and the SCOP extended dataset were evaluated in this manner, where the only difference between the two was the size of the positive training set.

After training the PCNSA classifier with the negative and positive training samples, the classification of a single test positive protein datasample begins with its score given by its distance from the positive class null space. This score represents how closely related (distance) the sample is to the positive training class. Other methods for calculating a score with PCNSA were tested – inverted distance from negative class null space, and positive class null space distance minus negative class null space distance. The score is then inserted into a list that contains the scores of every negative sample in the dataset, this list is sorted in ascending order and the number of negative samples that score lower than the test sample is divided by the total number of negative sequences (2437). This division results in the RFP for that positive test sequence.

A secondary experimental design was evaluated on the SCOP extended dataset, this test iterated the (r,s) pairs of PCNSA on the training set, using 10 fold cross validation. This kept the eight G Protein sequences and half the negative test sequences unseen to the classifier. Then the RFP G proteins test was performed using the (r,s) score that classifies the positive training samples best. This experimental setup kept the G Proteins untrained only one (r,s) pair is given to PCNSA to run on the test samples, unlike the test that repeatedly tested these eight and selected the best (r,s) pair.

7.4. Multispace KL

Multispace KL was implemented to produce ARFF files for the WEKA machine learning. The input data used for MKL experiments was the PIR-PSD four

class protein dataset. Numerous ARRF output files were created by varying the amount of clusters (s), and the dimensionality of these clusters (k), the two main parameters to MKL. Three main runs were performed, the first ranged s from 1 to 30, while holding k=6, the second ranged s from 1 to 21 with k=18, and third ranged s from 1 to 11 with k=33. These output datasets were then fed into the following classifiers:

- MultiLayerPerceptron: "A Classifier that uses backpropagation to classify instances."
- IB1: "Uses a simple distance measure to find the training instance closest to the given test instance, and predicts the same class as this training instance.", also known as nearest neighbor.
- IBK: k-nearest neighbor
- BayesNet: "Bayes Network learning using various search algorithms and quality measures."
- NaiveBayes: "Class for a Naive Bayes classifier using estimator classes."

All classifications were performed using ten fold cross validation.

7.5. HS3D Dataset

The HS3D dataset experiments were performed using ten fold cross validation while searching the parameter space of r and s. Only single runs were performed on the HS3D dataset. Experiments were also performed using the PCNSA k-nearest neighbour based classifier. Several dataset configurations were tested by including various gram windows and position specific nucleotide features as shown in Table 3 HS3D DNA dataset descriptions and features.

8. Results

8.1. HS3D Dataset

Experiments performed on the three HS3D datasets resulted in poor results:

Table 4 HS3D Top Results

Dataset	Low (r,s)	High (r,s)	Dimension	Best Accuracy
EILetters	300,295	395,390	396	89.00%
EIGramsLetters	180,175	365,360	368	90.18%
EIGramsHalfHalf	225,220	565,560	700	89.65%

Table 4 shows the best accuracy of each dataset and the highest and lowest (r,s) pair that resulted in that accuracy. In this table it is seen that all datasets perform around 90% accuracy. It seems PCNSA performs best on this data when the null space is very large, relative to the PCA space, differing only by 5 in all the above cases suggesting no useful attributes were found with little variance. In other words, no smaller set of null space vectors provided better discrimination of the sites than all of the possible null space vectors. This suggests new dataset features are needed a better representation, or that the dataset contains too much noise.

8.2. PIR-PSD Protein Dataset

For the PIR-PSD Dataset the accuracy is computed as the number of correctly classified divided by total number of samples tested, averaged across the ten folds. When more than one run is performed, the accuracy is averaged across all runs and folds, plus or minus the unbiased standard deviation of the run accuracies.

8.2.1. Two Class and SVMLight

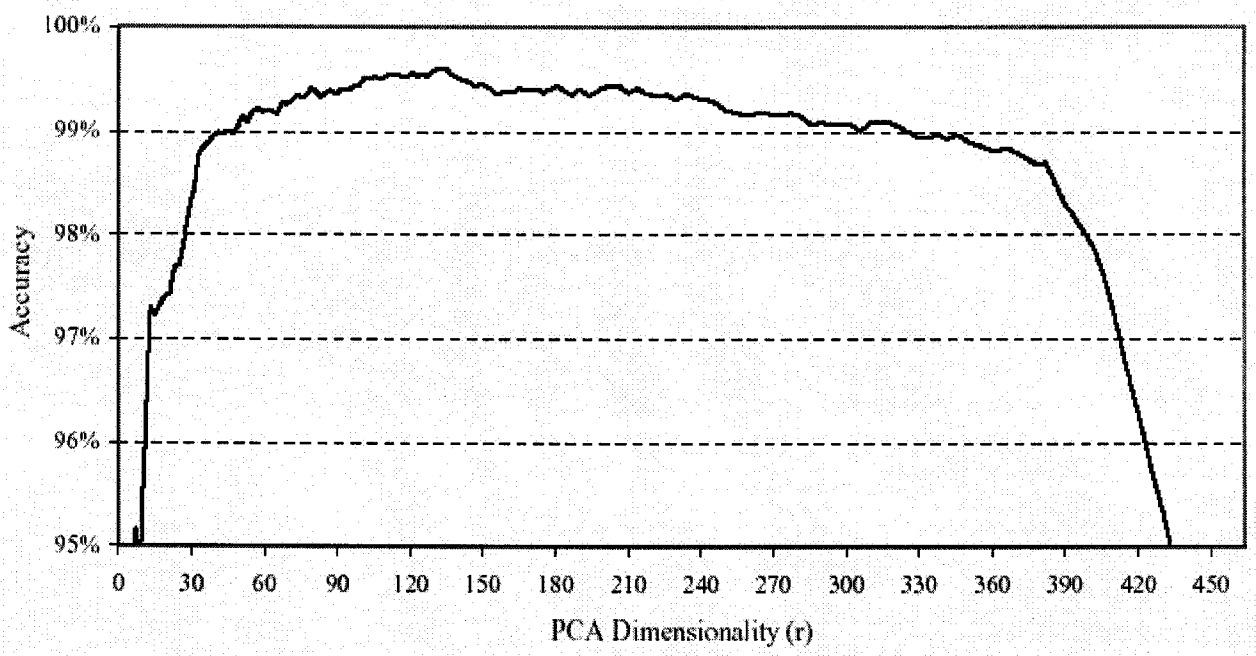


Figure 18 Effect of PCA space dimension, r, on two-class accuracy averaged across all s values, using ten fold cross validation for each (r,s) pair..

For the two-class case, Figure 18 shows the accuracy of PCNSA by varying the value of r. The accuracy displayed is an average of all possible values of s for that r value, where $465 \geq r \geq s \geq 1$. Again, ten fold cross validation was used for each test.

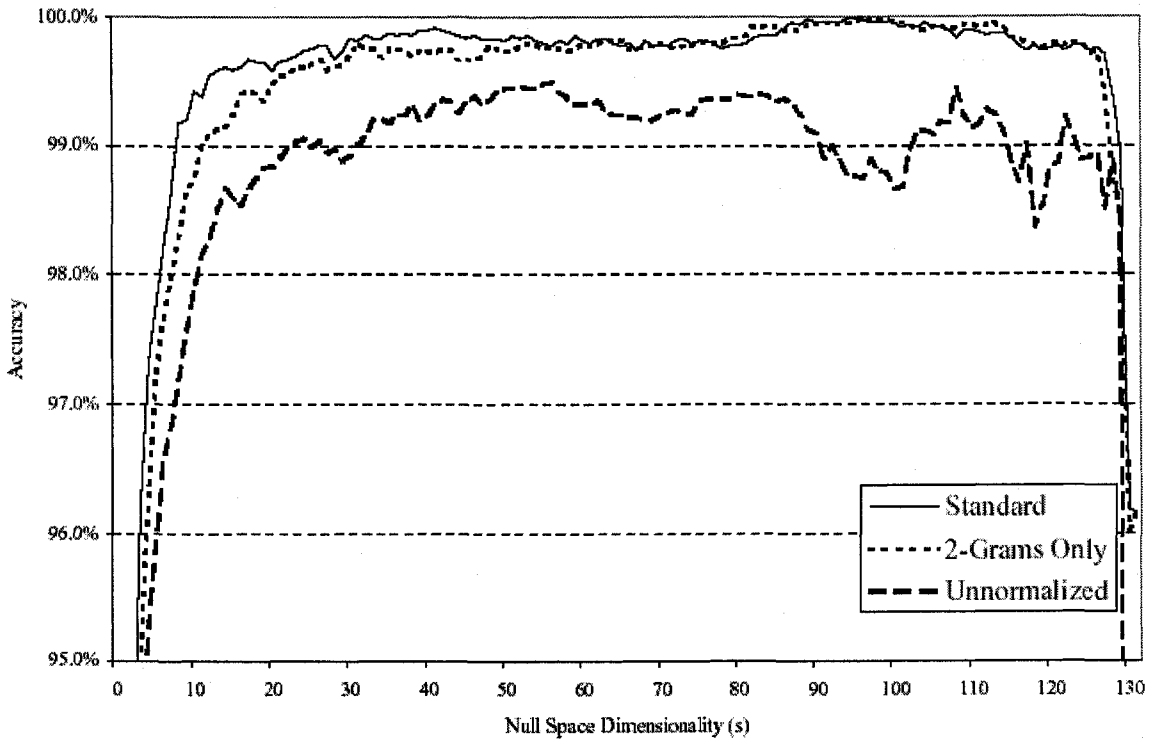


Figure 19 Effect of null space size and dataset type on two-class accuracy when r =133, based on 5 runs of ten fold cross validation.

Table 5 Comparison of a SVM to PCNSA, ten runs of ten fold cross validation

Method	Options	Ras	Kinase	Average
PCNSA	r=80, s=30	99.52% ± 0.20	99.96% ± 0.08	99.75% ± 0.11
PCNSA	r=133, s=97	99.98% ± 0.07	99.98% ± 0.06	99.98% ± 0.04
PCNSA	r=330, s=280	99.87% ± 0.15	99.94% ± 0.09	99.91% ± 0.10
SVMLight	Linear Kernel	99.49% ± 0.15	100% ± 0	99.76% ± 0.07
SVMLight	Polynomial Kernel degree 2	99.60% ± 0.09	100% ± 0	99.81% ± 0.04
SVMLight	Polynomial Kernel degree 3	99.60% ± 0.14	100% ± 0	99.81% ± 0.07
SVMLight	Polynomial Kernel degree 4	99.41% ± 0.15	100% ± 0	99.72% ± 0.07

A high scoring r value of 133 was obtained from results in the Figure 18. Figure 19 expands on that value by showing the effect of s on the accuracy. Additionally charted are the results of PCNSA given the dataset as two-grams plus exchange grams only, and unnormalized data. The "standard" line is the normal dataset setup, as described in the previous section. The unnormalized line skips the first step in the

PCNSA algorithm. Five runs were performed and averaged for each value of s. This graph supports earlier claims that normalizing the data reduces accuracy.

A SVM classifier was setup for comparison purposes. This was performed using SVM-Light support vector machine version 6.01 (Joachims, Schölkopf et al. 1999). The exact same datasets and folds were given to SVM-Light and PCNSA. Three top scoring parameter choices for PCNSA and four for SVM-Light are given. The only options provided to SVM-Light was the kernel function, all others were left as default. Radial basis function and Sigmoid kernels did not provide good results using the default kernel parameters. Table 5 Comparison of a SVM to PCNSA, ten runs of ten fold cross validation shows the resulting accuracies across ten runs of ten fold cross validation.

8.2.2. Four Class

The four class problem contained proteins from the ras transforming protein (ras), kinase-related transforming protein(kinase), globin and ribitol dehydrogenase (ribitol) superfamilies. Figure 21 demonstrates the accuracy across all values of s, where $r = 233$. The value of 233 was chosen from an exhaustive search of all possible parameter choices. Again, we can see the results of unnormalized and the two-gram plus exchange gram datasets for 5 runs of ten folds. In this case the difference between these datasets is less clear and the two-gram plus exchange grams dataset actually has the highest scoring result of $99.61\% \pm .05$ accuracy. This lessens the hypothesis that the added attributes of mass, length, pI, amino acid and exchange gram frequencies increase accuracy. Additionally, it is seen that the unnormalized performs best for low null space size.

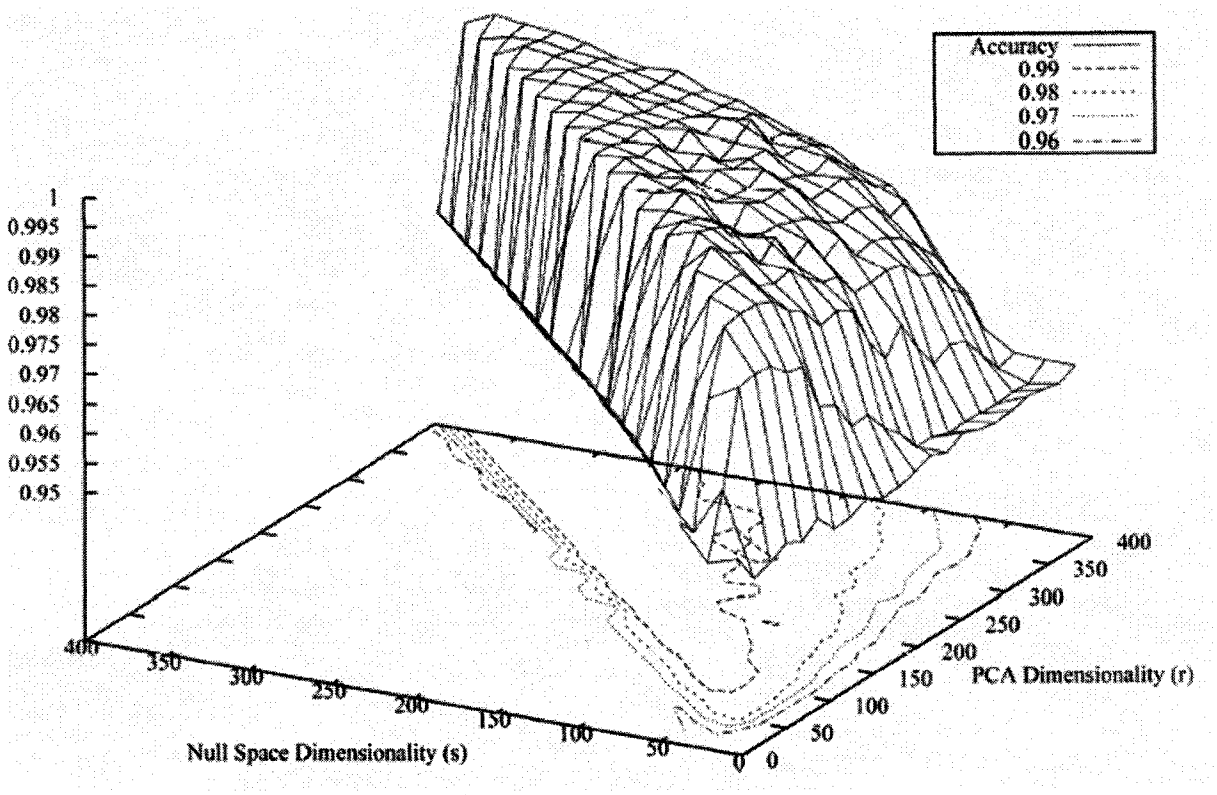


Figure 20 Accuracy of PCNSA displayed across all r and s values

The exhaustive search results were used to find three high scoring parameter combinations which were then further evaluated for ten runs to provide an accurate estimate of accuracy. The results of this test are provided in Table 6. Accuracy on a per class basis is also provided, it is important to note that the globin samples were classified perfectly on all ten runs and all three parameter pairs.

Table 6 Four class accuracies on three of the top r and s combinations, ten runs of ten fold cross validation

r	s	Ras	Kinase	Globin	Ribitol	Accuracy
185	150	97.95% ± 0.45	99.48% ± 0.23	100% ± 0	99.75% ± 0.09	99.43% ± 0.10
233	209	98.50% ± 0.17	99.42% ± 0.18	100% ± 0	99.77% ± 0.06	99.53% ± 0.06
320	297	98.54% ± 0.26	99.44% ± 0.14	100% ± 0	99.85% ± 0.10	99.57% ± 0.08

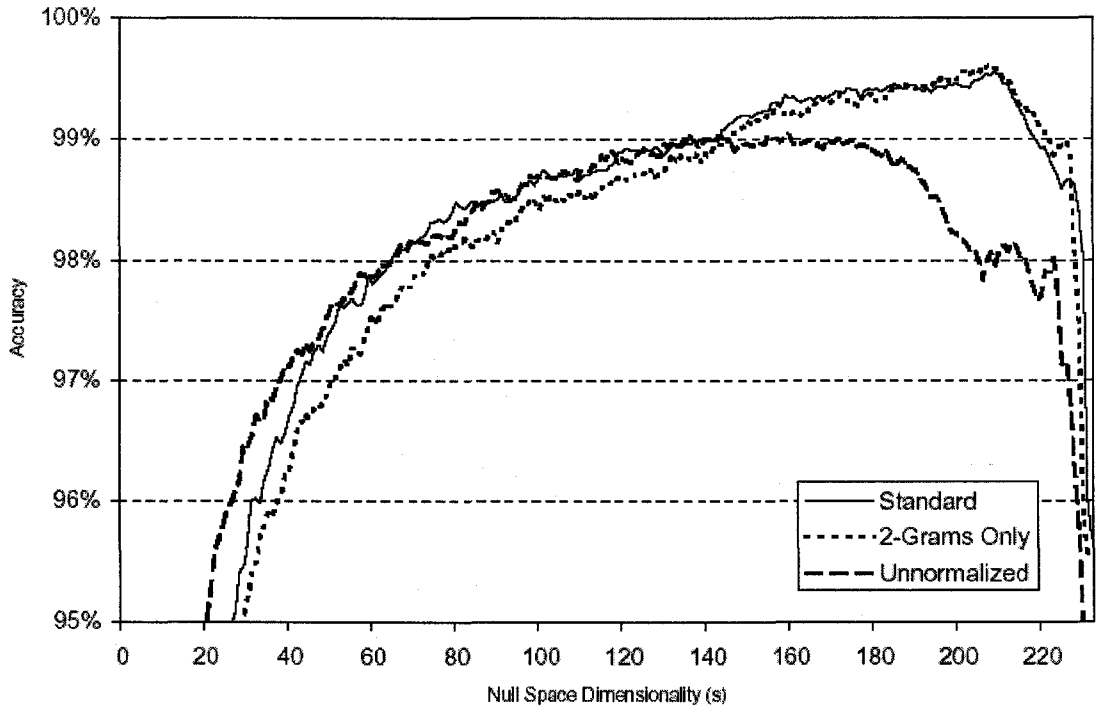


Figure 21 Effect of null space size and dataset type on four class accuracy when r=233, based on 5 runs of ten fold crossvalidation

In Table 7 it is possible to see how the algorithm accurately classifies data. This table gives the seven highest weighted attributes for each class, from a single run for r=320 and s=297. They are calculated using the PCA projection (W) and class null space projection (W_{class}) matrices. These are approximate weights because certain variables, such as class means, are not involved in the computations. Normalization of each attribute – step 1 of the algorithm, makes these weights more accurate. Every two-gram seen in the table occurs only once, demonstrating the uniqueness of the null spaces.

Table 7 Approximate highest weighted attributes, generated from a single ten fold cross validation using r =320 and s =297. Every entry is an amino acid two-gram.

Ras		Kinase		Globin		Ribitol	
VA	-7.36	RT	-9.24	SR	-8.42	PT	-10.07
AI	+7.13	IE	+9.24	TE	+7.80	WV	+7.85
MV	-6.96	DQ	+7.77	QK	+7.72	FN	-7.74
ED	-6.91	RE	-7.43	NF	+7.21	KA	+7.46
GE	-6.90	WD	+7.16	HP	+7.09	RP	-7.28
FM	+6.76	CM	-6.93	EG	-7.02	RL	+6.95
VT	+6.50	MF	+6.71	YY	-6.79	WL	-6.78

8.3. Highest and Lowest Ten Dataset

Below are the results of the highest and lowest ten dataset experiments, the parenthesized numbers represent the number of attributes/features in each dataset. The columns are ordered in expected accuracy, in ascending order.

Table 8 Results of highest and lowest ten dataset experiments

Classifier	Full Dataset (465)	Highest Ten (26)	Lowest Ten (37)
PCNSA	99.56%		
ID3	92.95%	74.73%	85.81%
REPTree	95.22%	78.72%	85.65%
NaiveBayes	94.62%	83.90%	77.68%
BayesNet	93.10%	95.22%	91.87%

In Table 8 two unexpected results can be seen. First BayesNet gained accuracy with the smallest dataset (Highest Ten). All classifiers performed considerably better on the full dataset compared to the other two, except BayesNet and all classifiers failed to beat PCNSA's accuracy. Second is that both trees performed better on the attributes that were weighted lower by PCNSA, which contradicts the hypothesis, an interesting result. One possible explanation is that the lower weighted dataset had 11 more

attributes to base classification on, hence giving it quantity over quality of the attributes.

8.4. SCOP G Proteins Dataset

The normal SCOP dataset, lacking the HMM extracted positive training sequences performed poorly. The best results in terms of lowest maximum rate of false positives (RFP) and lowest median RFP are provided in Table 9. Each row in a table represents one of the G Protein sequences and its RFP score. These results were taken from a exaustive search of the r, s parameter space. Classification accuracy was not recorded for these tests, only RFP statistics.

Table 9 RFP values for the eight test G Protein homologs, for top scoring (r,s) pairs

	Options	
Sequence	**r=16 s=7**	**r=31 s=28**
5p21	0.111	0.043
1guaA	0.322	0.090
1etu	0.014	0.102
1hurA	0.003	0.087
1eft(3)	0.023	0.060
1dar(2)	0.021	0.073
1tadA(2)	0.023	0.079
1gia(2)	0.172	0.101
median:	**0.023**	0.083
max:	0.322	**0.102**

The extended SCOP dataset which included the homolog sequences detected by SAM-T98 produced much better results. 22.6% of r and s pairs tested produced perfect (0) RFP scores for all the G Protein sequences. A method was created to determine an r and s pair for optimal classification without prior testing on the test sequences dataset. This second experimental design for the SCOP dataset used ten fold cross validation upon the extended training set while iterating r and s values by

10. This resulted in seven r and s pairs providing an accuracy of 99.98% (eight misclassified of 3802) on the positive samples and roughly 89.08% on negative samples. Every seven of these r and s pairs produced perfect RFP scores on the eight test sequences.

8.5. Multispace KL

Table 10 provides results for Nearest Nieghbour, Naïve Bayes and the Multilayer Perceptron classifiers when combined the MKL datasets. The K value represents the PCA size of each MKL cluster while s represents the number of clusters. The multilayer perceptron was too slow to run on the K=33 and K=18 sized datasets due to their size (363 dimensions at most). PCNSA was tested on the s=7, k=33 and the s=3, k=18 datasets under extended experiments and both yielded a best accuracy rate of 97.13%. In Table 10 it is seen that the Perceptron scores best followed closely by nearest neighbour then NaiveBayes performs the worst. The best overall accuracy is achieved by the Perceptron on the s=21, K=6 dataset, revealing an accuracy of 99.32%.

Table 10 Results of MKL combined with IB1, NaiveBayes and the MultilayerPerceptron classifiers

	K=33				K=6	
s	IB1/NN1	NaiveBayes	s	Perceptron	IB1/NN1	NaiveBayes
1	97.09%	**96.70%**	1	97.33%	96.22%	95.10%
2	98.73%	95.98%	2	97.77%	97.41%	**95.42%**
3	98.73%	96.26%	3	98.17%	97.57%	94.90%
4	98.65%	96.42%	4	98.37%	97.65%	94.86%
5	98.69%	96.38%	5	98.01%	97.25%	95.10%
6	98.73%	96.30%	6	98.65%	97.33%	95.26%
7	**98.93%**	96.62%	7	98.09%	97.61%	94.59%
8	98.53%	96.26%	8	98.69%	97.81%	94.94%
9	98.61%	96.10%	9	98.65%	97.73%	95.18%
10	98.89%	96.46%	10	98.93%	97.73%	94.86%
11	98.85%	96.26%	11	98.73%	97.73%	94.90%
			12	98.77%	97.77%	94.75%
	K=18		13	99.12%	98.01%	94.75%
1	97.01%	96.14%	14	98.93%	97.97%	94.82%
2	98.05%	96.14%	15	99.04%	97.53%	94.98%
3	98.25%	**96.34%**	16	99.04%	97.73%	94.67%
4	98.29%	95.58%	17	99.00%	**98.17%**	95.10%
5	98.09%	95.98%	18	99.12%	97.77%	94.90%
6	98.33%	95.78%	19	99.16%	97.73%	94.82%
7	98.25%	95.70%	20	99.04%	97.97%	94.98%
8	98.21%	95.50%	21	**99.32%**	98.09%	95.26%
9	98.33%	95.70%	22	99.00%	97.93%	95.02%
10	98.49%	95.98%	23	99.20%	98.17%	95.06%
11	98.53%	95.78%	24	98.89%	98.05%	95.02%
12	98.01%	95.98%	25	99.16%	97.89%	95.18%
13	98.25%	96.06%	26	99.04%	97.73%	94.90%
14	98.49%	96.02%	27	98.96%	97.97%	94.86%
15	98.21%	95.78%	28	99.04%	98.01%	94.90%
16	98.53%	96.10%	29	99.20%	97.93%	94.86%
17	**98.57%**	95.98%	30	99.08%	98.09%	95.02%
18	98.21%	96.14%				
19	98.41%	96.26%				
20	98.05%	96.18%				
21	98.53%	96.18%				

9. Comparisons

9.1. Thesis Based Comparisons

The below table summarizes experiments performed on the full PIR-PSD four class dataset using classifiers that were part of this thesis work. All classifiers were tested using ten fold cross validation, but not all classifiers were given the exact same fold distributions. It is seen that adding MKL and PCNSA bring the accuracies above common classifiers.

Table 11 Results of all classifiers tested directly on the PIR-PSD 4 class dataset

Classifier	Dataset	Dataset Size	Best Accuracy
PCNSA	Full	465	99.57%
MKL+Perceptron	MKL 21x6	126	99.32%
MKL+NearestNeighbour	MKL 33x7	231	98.93%
MKL+PCNSA	MKL 33x7	231	97.60%
MKL+NaiveBayes	MKL 33x1	33	96.70%
REP Tree	Full	465	95.22%
BayesNet	Highest Ten	26	95.22%
NaiveBayes	Full	465	94.62%
ID3 Tree	Full	465	92.95%

The SVMLight based classifier was not included in the four class table because it only directly performs binary classification. SVMLight was compared to PCNSA in section 8.2.1, the results demonstrated that PCNSA outperformed the basic SVM implementation.

9.2. Outside Research Comparisons

Table 12 provides a good comparison to other methods. All of the past work

was tested by the original authors on the PIR–PSD dataset but the class sizes and PSD version varied. All two-class cases used kinase versus ras, and the three class case added the globin superfamily.

Table 12 Comparison table of past and proposed classifiers on the PIR–PSD database

Method	PIR–PSD Release	Classes	Dataset Size	Accuracy
Fisher's (Rueda and Ngom 2004)	62	2	731	96.54%
PCNSA 2-Class (French, Ngom et al.)	62	2	731	98.00%
SVMLight (Joachims, Schölkopf et al. 1999)	79.05	2	972	99.81%
PCNSA 2-Class	79.05	2	972	99.98%
Multiclass NN (Zhang 2004)	N/A	3	3137	94.10%
Bayesian NN(Wang, Ma et al. 2001)[3]	62	4	1886	98.08% [4]
Combiner (Wang, Ma et al. 2001)[3]	62	4	1886	99.64% [4]
PCNSA 4-Class	79.05	4	2512	99.57%± 0.08

The Combiner method by Wang (Wang, Ma et al. 2001) provides the highest accuracy but there are several differences in the experimental design involved. Primarily, the problem definition used stated that a protein sequence can be classified into one or more superfamilies. This affected the dataset used, it contained 5 data subsets. The four sets corresponded to kinase, ras, globin, ribotol and a fifth set of 1650 negative sequences that did not contain any samples from the previous four sets. The classification took place in four binary experiments – a superfamily set (positive) versus the 1650 negative sequences. This suggests an easier-to-classify dataset than

[3] Binary classification performed for each of the superfamilies which is a very different experimental setup
[4] Computed from the average of four binary classification experiments, weighted by number of test sequences.

the one used in PCNSA experiments. Accuracy shown for Combiner and the Bayesian NN was computed from the average accuracy of the four binary classification experiments, weighted by number of test sequences.

It is important to note the complexity of the Combiner method(Wang, Ma et al. 2001). Combiner is based on the results of four classifiers: primarily, the Bayesian neural network, and then the results of classifiers based upon BLAST(Altschul, Madden et al. 1997), SAM(Hughey and Krogh 1996) and SAM-T99 (Karplus and Hu 2001). When compared to previous methods PCNSA is much simpler, faster and almost equal in accuracy.

Most of these competing methods used smaller training datasets and different experimental setups. Two of them provide the same experimental conditions under the 2-class case. First is the SVM using the above-described dataset and experimental setup. Second is Fisher's classifier, where a smaller training set was tested, as described in Rueda and Ngom (Rueda and Ngom 2004). This second experimental setup had a 60/40 train and test split with only 50 features. To assess PCNSA under similar conditions, we tested it using the *same* 60/40 training and testing datasets leading to 98% classification accuracy, and hence demonstrating its superiority over Fisher's classifier.

9.3. SCOP G Proteins Dataset

The results reported in section 7.3 are summarized and compared to first results on the dataset as reported by Jaakkola et al. in

Table 13. In this table the rate of false positives (RFP) is compared for each sequence in the G Proteins family, the lower the score the better. Here it is seen that PCNSA on the small dataset performs worse than SVM-Fisher(Jaakkola, Diekhans et al. 2000) and SAM-T98(Park, Karplus et al. 1998) on most sequences, the PCNSA scores are from the best scoring r and s pair in terms of median RFP (r=16, s=7). The BLAST (basic local alignment search tool) column represents the BLAST score on the small training dataset, and BLAST-Hom column represents the BLAST scoring when used on the extended dataset that includes the SAM-T98 homologs. The sequences 1etu and 1eft(3) are better classified using PCNSA than SAM-T98 and SVM-Fisher, additionally PCNSA using the small training set performs worse than all other methods, including BLAST for the sequences 5p21 and 1guaA. These mixed results demonstrate the uniqueness of the PCNSA algorithm, and also suggest that the small training set does not provide enough positive training samples to produce an accurate classifier.

Table 13 also presents the results of PCNSA when trained on the positive training set that included the homologs found by SAM-T98 denoted by PCNSA-Hom. Here as described previously the sequences are classified perfectly, this occurs with seven (r, s) pairs as determined by classification on the training set. These results clearly outperform all other methods at the time of the Jaakkola et al. publication in 1999. It is important to note that these results are comparing a small part of the dataset. The G Proteins family is only 1 family of 33 in the full dataset. Nonetheless these results combined with PIR-PSD results provide a good indication that the remaining 32 families will be well classified.

Table 13 Comparison to methods in Jaakkola et al. for each sequence in the G Proteins family, lower RFP score indicates better performance adopted from "A Discriminative Framework for Detecting Remote Protein Homologies" (Jaakkola, Diekhans et al. 2000) page 105

Sequence	BLAST	B-Hom	S-T98	SVM-F	PCNSA	PCNSA-Hom
5p21	0.043	0.01	0.001	0	0.111	0
1guaA	0.179	0.031	0	0	0.322	0
1etu	0.307	0.404	0.428	0.038	0.014	0
1hurA	0.378	0.007	0.007	0	0.003	0
1eft(3)	0.431	0.568	0.041	0.051	0.023	0
1dar(2)	0.565	0.391	0.289	0.019	0.021	0
1tadA(2)	0.797	0.33	0.004	0	0.023	0
1gia(2)	0.867	0.421	0.017	0	0.172	0

Table 14 provides more statistics that are also quoted from Jaakkola et al., these numbers are the median RFP and maximum RFP of the eight G Protein scores. Here it is seen that PCNSA on the small dataset performs in between SAM-T98 and SVM-Fisher, while PCNSA-Hom is the best with its perfect scores.

Table 14 Maximum and Median RFP scores for the G Proteins dataset adopted from "A Discriminative Framework for Detecting Remote Protein Homologies" (Jaakkola, Diekhans et al. 2000) page 106

Maximum RFP

BLAST	BLAST-Hom	SAM-T98	SVM-Fisher	PCNSA	PCNSA-Hom
0.867	0.568	0.428	0.051	0.083	0

Median RFP

BLAST	BLAST-Hom	SAM-T98	SVM-Fisher	PCNSA	PCNSA-Hom
0.378	0.33	0.007	0	0.023	0

10. Conclusion

10.1. Summary of Work Done

In this thesis a large amount of work was completed:

- Implementation of PCNSA algorithm, and many extensions to it
- Implementation of the Multisplace KL algorithm
- Implementation of dataset parsers and feature extractors for the SCOP, HS3D and PIR-PSD datasets
- Creation of several datasets in the standard format known as attribute relation file format
- Critical evaluation of PCNSA and Multispace KL under many different circumstances and with several datasets

10.2. Limitations

Several limitations exist on the current work presented:

- No fast way to determine r and s parameters for PCNSA, exhaustive search is currently required
- A full test suite for the SCOP dataset was not evaluated, only a small subset of it was evaluated
- Only three datasets were tested based upon the HS3D dataset, and resulted in poor performance
- A full empirical and theoretical study of the computational time and space complexity involved in PCNSA is not provided

10.3. Future Directions

Bioinformatics has a large amount of pattern classification problems. Microarray datasets are very very large in dimension and are often not fully analyzed. Microarray datasets are a promising match for PCNSA because of their propensity to very high noise and intuitively fit the description of an "Apples vs. Oranges" problem. Furthermore, the weight tracking module developed for PCNSA would allow output of the most heavily weighted genes or features in the microarray datasets.

Future work will involve testing this method on a larger dataset with sequences from SCOP (Murzin, Brenner et al. 1995) or PROSITE (Bairoch and Bucher 1994). The protein classification problem definition could be modified so that a protein sequence can be classified into zero or more superfamilies, which is a more biologically accurate model for the problem.

Another interesting avenue of research is "why does PCNSA perform so well?". This could be accomplished by reviewing error bound work in the original PCNSA paper by Vaswani. And conversely, "why does it work so poorly on the DNA dataset?". Additionally the feature tracker could be used more extensively to study the features PCNSA uses for classification and the statistics of them.

Additionally is clear that a good way to determine r and s parameters needs to be researched. Figure 20 Accuracy of PCNSA displayed across all r and s values, suggests a simple hill climbing or gradient descent algorithm could be successfully applied.

10.4. Conclusions

In this thesis, we present several approaches to the protein classification problem. Our primary method is based on the PCNSA linear classifier, which is slightly modified from its original version by introducing feature based normalization and removing two null space filters.

We have tested our method on four superfamilies for the PSD–PIR databank, and compared our results to previous methods. The empirical analysis presented shows that our method is superior to any previous results on the two-class problem, achieving an accuracy of 99.98%, with a standard deviation of 0.04. In the four-class case, our method performs at par to Combiner with $99.57\% \pm 0.08$ accuracy, while possessing the advantage of higher speed and lowered complexity.

Tests on the DNA splice site dataset, resulted in poor performance. The signal versus noise nature of the problem suggests justification for the results. Unlike the DNA dataset, the protein datasets were closer to the 'apples versus oranges' class.

Preliminary results suggest PCNSA will outperform the SVM-fisher method by Jaakkola et al. on the full SCOP based dataset. This finding and the SVMLight results contradict the current opinions in bioinformatics that the non-linear SVM's are the best classifiers for classifying biological data.

Bibliography

Altschul, S. F., T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nucleic Acids Res **25**(17): 3389-402.

Bairoch, A. and P. Bucher (1994). "PROSITE: recent developments." Nucleic Acids Res **22**(17): 3583-9.

Baxevanis, A. D. and B. F. F. Ouellette (2004). Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Wiley, John & Sons, Incorporated.

Cappelli, R., D. Maio and D. Maltoni (2001). "Multispace KL for Pattern Representation and Classification." IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(9): 977-996.

Dayhoff, M. O., R. M. Schwartz and B. C. Orcutt (1978). "A Model of Evolutionary Change in Proteins." Atlas of Protein Sequence and Structure **15**(Supplement 3): 345-358.

Durbin, R., S. R. Eddy, A. Krogh and G. Mitchison (1999). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press.

Hoschek, W. (2000). "Uniform, Versatile and Efficient Dense and Sparse Multi-Dimensional Arrays."

Hughey, R. and A. Krogh (1996). "Hidden Markov models for sequence analysis: extension and analysis of the basic method." Comput Appl Biosci **12**(2): 95-107.

Jaakkola, T., M. Diekhans and D. Haussler (2000). "A discriminative framework for

detecting remote protein homologies." J Comput Biol **7**(1-2): 95-114.

Joachims, T., B. Schölkopf and C. Burges (1999). Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, MIT-Press.

Karplus, K. and B. Hu (2001). "Evaluation of protein multiple alignments by SAM-T99 using the BAliBASE multiple alignment test set." Bioinformatics **17**(8): 713-720.

Leslie, C., E. Eskin and W. S. Noble (2002). "The spectrum kernel: a string kernel for SVM protein classification." Pac Symp Biocomput: 564-75.

Madera, M. and J. Gough (2002). "A comparison of profile hidden Markov model procedures for remote homology detection." Nucleic Acids Res **30**(19): 4321-8.

Murzin, A. G., S. E. Brenner, T. Hubbard and C. Chothia (1995). "SCOP: a structural classification of proteins database for the investigation of sequences and structures." J Mol Biol **247**(4): 536-40.

Park, J., K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard and C. Chothia (1998). "Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods." J Mol Biol **284**(4): 1201-10.

Pollastro, P. and S. Rampone (2003). "HS3D: Homo Sapiens Splice Site Data Set." Nucleic Acids Research(Annual Database Issue).

Rueda, L. and A. Ngom (2004). An Empirical Evaluation of the Classification Error of Two Thresholding Methods for Fisher's Classifier. International Conference on Artifical Intelligence & International Conference on Machine Learning; Models, Technologies & Applications, Las Vegas, Nevada, USA, CSREA Press.

Vaswani, N. (2002). "A Linear Classifier for Gaussian Class Conditional

Distributions with Unequal Covariance Matrices." Intl. Conference on Pattern Recognition (ICPR) **I**: 240.

Vaswani, N. and R. Chellappa (2004). Classification Probability Analysis of Principal Component Null Space Analysis. Intl. Conference on Pattern Recognition (ICPR), Cambridge, UK.

Vaswani, N. and R. Chellappa (2004). "Principal Component Null Space Analysis for Image and Video Classification." IEEE Transactions on Image Processing.

Venter, J. C., K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, D. E. Fouts, S. Levy, A. H. Knap, M. W. Lomas, K. Nealson, O. White, J. Peterson, J. Hoffman, R. Parsons, H. Baden-Tillson, C. Pfannkoch, Y.-H. Rogers and H. O. Smith (2004). "Environmental Genome Shotgun Sequencing of the Sargasso Sea." Science **304**(5667): 66-74.

Wang, J. T. L., Q. Ma, D. Shasha and C. H. Wu (2001). "New techniques for extracting features from protein sequences." IBM Systems Journal **40**(2).

Witten, I. and F. Eibe (2005). Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, San Francisco.

Wu, C., G. Whitson, J. McLarty, A. Ermongkonchai and T. C. Chang (1992). "Protein classification artificial neural system." Protein Sci **1**(5): 667-77.

Wu, C. H., M. Berry, Y. S. Fung and J. McLarty (1995). "Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with Singular Value Decomposition." Machine Learning **21**: 177-193.

Wu, C. H., L. S. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, P. Kourtesis, R. S. Ledley, B. E. Suzek, C. R. Vinayaka, J. Zhang and W. C. Barker (2003). "The Protein Information Resource." Nucleic Acids Res **31**(1):

345-7.

Zhang, X. (2004). Protein Family Classification Using Multiple-Class Neural Networks. Faculty of Graduate Studies and Research. University of Windsor.

Appendix A: Survey of Protein Sequence Classification

A Survey on Remote Homology Detection and Protein Superfamily Classification

Instructor: Dr. Richard Frost
Supervisor: Dr. Alioune Ngom

August, 2005

By Leon French
leonfrench@gmail.com

Abstract

This report on the classification of protein sequences; presents methods, results and comparisons of various approaches to the problems of protein sequence classification and remote homology detection. A comprehensive study of all novel published techniques is provided to present a complete perspective of past research. Information concerning the experimental setup and sequence databases involved in the many tests of various methods is described and compared where possible. The evolution of the basic string comparison methods to the current advanced approaches using support vector machines are linked together to provide clear connections between the research.

Table of Contents

I Introduction

Two textbooks, "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids"(Durbin, Eddy et al. 1999) and "Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins"(Baxevanis and Ouellette 2004) are recommended for further reference on bioinformatics, superfamily classification and remote homology search.

The topic of this survey falls into the relatively new research area of bioinformatics. Bioinformatics is a mix of computer science and biology. The human genome project describes bioinformatics as "the science of managing and analyzing biological data using advanced computing techniques".

Research into the problems of protein sequence superfamily classification and remote homology detection sequence search are the focus of this survey. Protein sequence classification and remote homology detection are similar problems; both refer to the prediction of superfamily or other group membership of an unknown protein sequence. Commonly the known superfamily sequences are used as training data for a pattern recognition algorithm which then can be queried with a test sequence. The predicted superfamily will suggest the function and structure for the test sequence, providing annotation data without wet-lab work. The definition of the problem varies between experimental setups, definitions of a superfamily, and databases. For example the PIR database defines superfamilies as non-overlapping so exact membership can be predicted and accuracy tested (Wu, Yeh et al. 2003). Other approaches produce a probability that a given unknown protein is a member of an overlapping family or superfamily set and are tested with an ROC curve(Gribskov and Robinson 1996).

II Methods
Pairwise Sequence Similarity Based Methods

The first work comparing protein sequences was developed in 1970 by Needleman and Wunsch; they provided an $O(n^2)$ global pairwise alignment algorithm (Needleman and Wunsch 1970). In 1978, the Point Accepted Mutations (PAM) matrix was introduced (Dayhoff, Schwartz et al. 1978). This matrix provides probabilities of amino acid replacements between two sequences across an evolutionary distance. This matrix is commonly used in sequence comparison methods. Smith and Waterman describe an algorithm for local pairwise sequence alignment in 1981 (Smith and Waterman 1981). Described is a dynamic programming approach that allows for local pairwise alignment of two sequences. This algorithm is primarily used for benchmarking pairwise homology search methods but is not applied to the problem due to its $O(n^2)$ time requirements (Smith and Waterman 1981; Pearson 1991).

The next generation of sequence search methods are more applicable to fast database searches, they approximate the dynamic programming algorithms of Needleman, Wunsch, Smith, and Waterman. Fast database searches are a precursor problem to remote homology detection. A naïve remote homology detection method is: given a fast database search method one or more homologous sequences can be found and used to estimate the superfamily of the query sequence.

These fast search methods started with a 1985 paper by Lipman and Pearson which describes the FASTP algorithm for pairwise sequence comparisons (Lipman and Pearson 1985). This method is a based on lookup tables with entries for each amino acid or 2-gram and its position. A substitution matrix is used and insertions and deletions are not considered. Several test runs are provided with interpretation of the results. The authors state this method is computationally faster than other methods. Five years later, BLAST is introduced (Altschul, Gish et al. 1990). The Basic local alignment search tool or BLAST, addressed the problem of fast sequence comparison. BLAST uses local substrings of the sequence, combined with statistics to produce a hash table. BLAST finds maximum scoring pairs (identical length substrings with high similarity) between two sequences and then extends these pairs. BLAST was tested on the PIR protein database release 22.0. The authors claim an order of magnitude of speed increase relative to other heuristic methods. Unlike BLAST, dynamic programming algorithms for sequence comparison are too slow for comparisons against all sequences in a large database, BLAST approximates these algorithms. BLAST became a popular standard and general tool for bioinformatics. It is still popular today, but its remote homologue search abilities have been eclipsed (see section IV Direct Comparisons) by newer and more specialized methods as seen in this survey.

In 1991, FASTA and the Smith-Waterman algorithm are tested on

superfamily classification (Pearson 1990; Pearson 1991). Experiments were performed on the PIR-PSD database. Authors claim FASTA using ktup = 2 performed at par to the slower Smith-Waterman method. Further work on BLAST was announced in 1997 (Altschul, Madden et al. 1997). A major change to the original BLAST algorithm increases its speed by requiring two word pair hits, before extension takes place. Two new versions are developed, PSI-BLAST and Gapped BLAST. PSI-BLAST uses a profile (position specific score matrix) created from aligned BLAST results then searches the database using this profile to find more remote homologues - this is done in an iterative manner. Gapped BLAST allows for the high scoring pairs to contain gaps while sacrificing little performance and speed. PSI-BLAST performance is demonstrated on the BRCT protein superfamily. Results obtained show few false positives, but also demonstrated that distant sequences are not found. This paper allowed for faster and more sensitive searching for remote homologs using direct sequence comparison.

In 1997 a unique approach to homology detection named intermediate sequence search (ISS) was described (Park, Teichmann et al. 1997). This method approaches the problem of remote homology detection by using indirect pairwise searches. This method is implemented using two sets of FASTA (Pearson 1990) searches. Previous FASTA results are used again for another round of pairwise searching. Three sequences are involved and classification of a query sequence is based on a second intermediate sequence - if it and the query sequence matches well to a third sequence from the second round of FASTA results. Tests were performed on the PDB40-J database which contains only very remote homologies. The author claims an improvement of 70% over a traditional FASTA method.

Grundy examines and extends using BLAST for remote homology search in 1998 (Grundy 1998). This method is based on pairwise comparisons of the query sequence to each sequence in the training set but is extended for more than one query sequence. The implementation tested uses BLAST for the pairwise comparisons. This method is shown to perform well for difficult superfamilies that are very small in size (8-30 known sequences).

Profile Based Methods

In the first profile paper for homology detection a method for sequence to sequence family comparison is described (Gribskov, McLachlan et al. 1987). Unlike past pairwise methods, this method is based on combining a family of sequences into a profile that represents the probability of an amino acid occurring at a certain position. These profiles can then be aligned to single sequences to derive a score. High scoring sequences can be classified into the family that was used to generate the profile. Experiments are performed on the globin and immunoglobulin families from the PIR database. The authors claim it is an ideal method for binary classification. A similar work by Henikoff and Henikoff describes a database of locally aligned sequences (a block) which is used for protein family classification (Henikoff and Henikoff

1994; Henikoff, Pietrokovski et al. 1998). Searching is performed by aligning a query sequence against the profile of every block in the database. The authors give specific results of using this method.

Motif Based Methods

Bailey and Elkan are first to introduce motifs to biological sequences (Bailey and Elkan 1994). Specifically they address the problem of motif discovery using expectation maximization. From a set of unaligned sequences an algorithm named MM is given that produces several motifs. MM is based in the MEME algorithm for motif discovery which was also developed by Bailey and Elkan. These motifs can then be used for database searching. Experiments are performed on a DNA dataset. The MEME algorithm is further developed for discovering motifs using expectation maximization (Bailey and Elkan 1995). Results from several experiments are provided with demonstrate that MEME is more accurate (ROC measure) when given prior knowledge about the motifs in the sequences.

The paper titled "Score distributions for simultaneous matching to multiple motifs" introduces database searching using multiple motifs to increase accuracy of database searches(Bailey and Gribskov 1997). The main purpose of the paper is calculation of probability values for multiple motif scores hence a large amount of statistical reasoning is provided. Experiments use MEME for motif discovery and test on the SWISS-PROT database. The results support the claim that multiple motifs provide better accuracy (ROC50 values) than searches based on single motifs. Bailey and Gribskov provide empirical support for the previous theory based paper on the combination of multiple motif p-values(Bailey and Gribskov 1998). This addresses the problem of how to combine many p-values into a single value that represents probability that the sequence is of a class represented by the motifs. This work assumes that the p-values sources used are independent. An algorithm is provided and tested on SWISS-PROT 28.0, ROC50 statistics are provided. The authors claim increased sensitivity and selectivity for sequence homology searches.

Application of Hidden Markov Models

In 1994 a paper titled "Hidden Markov models in computational biology, Applications to protein modeling" was the first to introduce hidden Markov models to the problem of protein sequence classification(Krogh, Brown et al. 1994). Previous method involved profiles, motifs and single sequence based searches. Model parameters are learned from unaligned sequences using an expectation maximization algorithm. Tests were performed on globin, kinase and EF-hand proteins from the SWISS-PROT database. The authors claim HMM performs better than other methods available. Possible extensions for

unbiased sequence weighting schemes and incorporation of the PAM matrix are suggested. Another similar work was published around the same time titled "Hidden Markov models of biological primary sequence information" (Baldi, Chauvin et al. 1994).

Eddy, Mitcheson et al. extend the first work(Krogh, Brown et al. 1994) on the early applications of hidden Markov modeling for protein sequence classification(Eddy, Mitchison et al. 1995). In this paper the sequences used for the Markov models are weighted using the maximum discrimination scheme. They state this prevents many sequences of high similarity from biasing the model. The authors claim this allows for more sensitive remote homology searching compared to BLAST and other HMM approaches.

Hughey and Krogh further explore the application of hidden Markov modeling for protein sequence classification(Hughey and Krogh 1996). Several details of HMM's are examined – regularizers, dynamic model modification and free insertion modules. Theoretical and experimental results are provided. This paper describes the first implementation of the Sequence Alignment and Modeling (SAM) software, one of the two top performing HMM tools. SAM-T98 is introduced by Karplus, Barrett and Hughey as a new HMM based method named SAM-T98 (Karplus, Barrett et al. 1998). SAM-T98 iteratively generates the model starting with one sequence then uses this model to find homologs to merge into this model, a method similar to PSI-BLAST. This paper provides very detailed descriptions of the extended work into the hidden Markov model implementation – such as sequence weighting and the null model used. Tests were performed on the SCOP database and Person test set. The authors state that at all minimum-error points SAM-T98 performed best compared to competing methods.

In 2003, Griffiths-Jones and Bateman investigated the impact of multiple alignments used for the creation of HMMs(Griffiths-Jones and Bateman 2002). Experiments are performed on the Pfam database using HMMs derived from several multiple alignments based on structural or sequence data. The authors claim that using structural based alignments do not increase accuracy of derived HMMs on the problem of remote homology search. Also in 2003 a paper titled "Efficient estimation of emission probabilities in profile hidden Markov models" addresses the issue of HMM's giving equal weight to noisy amino acid positions, which causes overfitting (Ahola, Aittokallio et al. 2003). This paper extends work on the HMM's method by adding emission probability estimation. The new technique was tested against BLAST(Altschul, Gish et al. 1990) and HMMER(Eddy 1998) on TIM barrel sequences in the SWISS-PROT database. Results show similar accuracy but with a reduced false positive rate.

Consensus Based Approaches

Wang et al. developed a consensus classifier for the problem of protein

sequence classification in 2001 (Wang, Ma et al. 2001). Experiments are performed on four protein superfamilies from the PIR-PSD database. The consensus classifier classifies the protein as the majority vote of separate classifiers derived from BLAST(Altschul, Gish et al. 1990), SAM(Hughey and Krogh 1996), SAM-T99(Karplus and Hu 2001) and a Bayesian neural network. The accuracy of the combined classifier exceeds any of its parts on the datasets tested.

Can et al. address the problem of slow manual curation of proteins in the SCOP database(Can, Camoglu et al. 2004). This paper provides a consensus based framework for classification at the superfamily, family and fold levels. Classifiers used are three structure based methods and two sequence based – HMMER(Eddy 1998) and PSI-BLAST(Altschul, Madden et al. 1997). Classification is tested on new sequences added to the SCOP database. This paper shows that the consensus classifier outperforms its individual parts.

Application of the Support Vector Machine

Jaakkola et al. present the first full application of a support vector machine to the problem of remote homology search(Jaakkola, Diekhans et al. 1999). Protein sequences are encoded for a fixed dimensional space by hidden Markov models (SAM-T98), and then classified using the kernel Fisher classifier. Testing was performed on the SCOP database and compared to BLAST(Altschul, Gish et al. 1990) and SAM-T98(Karplus, Barrett et al. 1998) methods. The test dataset was novel in that it tested the ability of the classifier to predict an unseen family of a known superfamily. The authors claim significant improvement in ability of remote homolog detection.

In 2000 Liao and Noble combine pairwise alignment with a support vector machine(Liao and Noble 2002). This work uses Smith-Waterman(Smith and Waterman 1981) pairwise sequence comparison to create a feature vector. Each sequence is represented as a vector of scores from an alignment to every training sample. Tests were performed on the Astral/SCOP database and ROC50 results are provided. Authors claim significantly improved remote homology detection compared to current state-of-the-art methods while running slower by Omega(n) where n is training set size.

In "The spectrum kernel: a string kernel for SVM protein classification" the problem of remote homology detection is attempted with a novel kernel function(Leslie, Eskin et al. 2002). This paper is different from past SVM work because it uses a spectrum kernel function based on k-length subsequences or n-grams. The kernel function is computed by taking the dot product of vectors containing the k-length distributions corresponding to a sequence. The authors found that k=3 or 4 provides best results. This kernel function is faster and simpler to compute than the Fisher kernel. Tests were performed on the SCOP database and ROC50 results are provided. This general

method is shown by the authors to perform at par to state-of-the-art methods.

Hou et al. take another approach at remote homology detection with a support vector machine(Hou, Hsu et al. 2003). The approach is different from past SVM work because it uses a kernel function based on structure similarity instead of sequence similarity. Tests were performed on the SCOP database. ROC50 results are provided and compared to competing methods. This method is shown to perform at par to the SVM-pairwise (Liao and Noble 2002) method.

Ben-Hur and Brutlag provide another interesting application of a support vector machine in 2003 (Ben-Hur and Brutlag 2003). They describe a kernel function based on motif content scores. Tests were performed on the ASTRAL and SwissProt databases and ROC50 results are provided. This method is shown to perform better than kernels based on BLAST (Altschul, Gish et al. 1990) or Smith-Waterman (Smith and Waterman 1981) scores.

Hou et al. extend on past work (Hou, Hsu et al. 2003) on the usage of a SVM for remote homology detection(Hou, Hsu et al. 2004). Structure information is obtained for a global level by a hidden Markov model (HMMSTR) developed by Bystroff et, al (Bystroff, Thorsson et al. 2000). The authors test on the SCOP database and give ROC statistics. The authors claim excellent performance.

In "Mismatch string kernels for discriminative protein classification." the problem of remote homology detection is approached with a new string kernel(Leslie, Eskin et al. 2004). This work uses a mismatch kernel function that is similar to the spectrum kernel(Leslie, Eskin et al. 2002). The kernel function is computed by taking the dot product of vectors containing the k-length distributions with m mismatches, each of these vectors represents a sequence. The authors found that setting k=5 and m=1 provided best results. This kernel function is faster and simpler to compute than the Fisher kernel. Tests were performed on the SCOP database and ROC50 results are provided. Authors claim this method performs at par to the Fisher kernel on the SCOP dataset.

Unique Methods

In 1992, Wu et al. describes one of the first applications of a neural network to the problem of protein classification(Wu, Ermongkonchai et al. 1991). N-gram sequence encodings are used as input to a neural network. The Authors test the network on 620 superfamilies from the PIR-PSD database. The author's state accuracy of 90% is achieved.

Eskin et al. introduce method based on sparse Markov transducers for protein family classification (Eskin, Grundy et al. 2000). This method is based on probabilistic suffix trees and adds a mixture technique for placement of wildcards. In 2003 this work is extended to increase efficiency(Eskin, Noble et al. 2003). Experiments are performed on the Pfam and SCOP database and

ROC50 statistics are provided. Authors show using the ROC50 statistic that this method performs worse than the kernel Fisher method(Jaakkola, Diekhans et al. 1999) and in some cases BLAST(Altschul, Gish et al. 1990).

Wang extends previous work by Wu et al. on Neural Networks by applying a Bayesian neural network (BNN) and a consensus classifier to the problem of protein sequence classification (Wang, Ma et al. 2001). The BNN is trained and tested using 2-gram encodings and motif scores. Experiments are performed on four protein superfamilies from the PIR-PSD database. From comparisons given for BLAST (Altschul, Gish et al. 1990), SAM(Hughey and Krogh 1996) and SAM-T99(Karplus and Hu 2001) it is shown by the authors that the BNN performs at state-of-the-art levels.

A paper titled "Variations on probabilistic suffix trees: statistical modeling and prediction of protein families" introduces probabilistic suffix trees to the problem of protein family classification(Bejerano and Yona 2001). This method is based on `short memory' or the ability of the next amino acid to be predicted given the short subsequence preceding it. The authors state this method is simple to apply and does not require a multiple sequence alignment. A simple implementation of the method is evaluated on the Pfam database and compared to BLAST and HMM methods. The authors show the implementation outperforms a basic Gapped-BLAST pairwise (Altschul, Madden et al. 1997) method. Bejerano et al. describe another novel method that combines clustering, probabilistic suffix trees and a variable memory Markov models for the problem of protein family classification(Bejerano, Seldin et al. 2001). The authors perform experiments on protein domain detection.

French et al. apply a recently developed linear hyperplane classifier to protein superfamily classification(French, Ngom et al. 2005). Like previous work the 2-gram encoding method is used (Wang, Ma et al. 2001). This method involves projections into subspaces of the feature space. Experiments are performed on the PIR-PSD database and compared to past methods tested on this database. The authors claim accuracy similar to the consensus method described in Wang 2001 while being simpler and faster.

III Direct Comparisons

Madera and Gough test HMMER(Eddy, Mitchison et al. 1995), SAM(Hughey and Krogh 1996), SAM-T99(Karplus and Hu 2001), BLAST(Altschul, Gish et al. 1990) and PSI-BLAST(Altschul, Madden et al. 1997) on the remote homology problem (Madera and Gough 2002). Tests were performed on two families from the nrdb90 database and the entire SCOP database. The authors claim that SAM produced better models while HMMER is faster for model building. Additionally the authors concluded that SAM-T98 performs better than the other tested methods while sacrificing speed of classification.

Park et al. compared three multiple sequence remote homolog detection methods and compared them to pairwise methods(Park, Karplus et al. 1998). Methods tested were PSI-BLAST(Altschul, Madden et al. 1997), SAM-T98(Karplus, Barrett et al. 1998), and ISS(Park, Teichmann et al. 1997). The classifiers were evaluated using the PDB40-J database which contains only very remote homologies. The authors presented results that demonstrate SAM-T98 found more homologous relationships (35%) than PSI-BLAST(30%) or ISS (30%) which significantly outperform the pairwise methods of FASTA (17%) and Gapped BLAST(15%).

Performance comparisons are provided in most publications seen in this review, two diagrams and one table are given to provide an overview of the most popular and accurate methods. The table provides a comparison of methods performed on the PIR-PSD database:

Method	PIR–PSD Release	Classes	Dataset Size	Accuracy
Fisher's (Rueda and Ngom 2004)	62	2	731	96.54
Multiclass NN (Zhang 2004)	N/A	3	3137	94.10
Bayesian NN(Wang, Ma et al. 2001)[5]	62	4	1886	98.08 [2]
Combiner (Wang, Ma et al. 2001)[1]	62	4	1886	99.64 [6]
PCNSA (French, Ngom et al. 2005)	79.05	4	2512	99.57 ± 0.08

As seen in the table, comparisons on this database are difficult as versions, classes (number of superfamilies), dataset size, and experimental design differ greatly.

[5] Binary classification performed for each of the superfamilies which is a very different experimental setup
[6] Computed from the average of four binary classification experiments, weighted by number of test sequences.

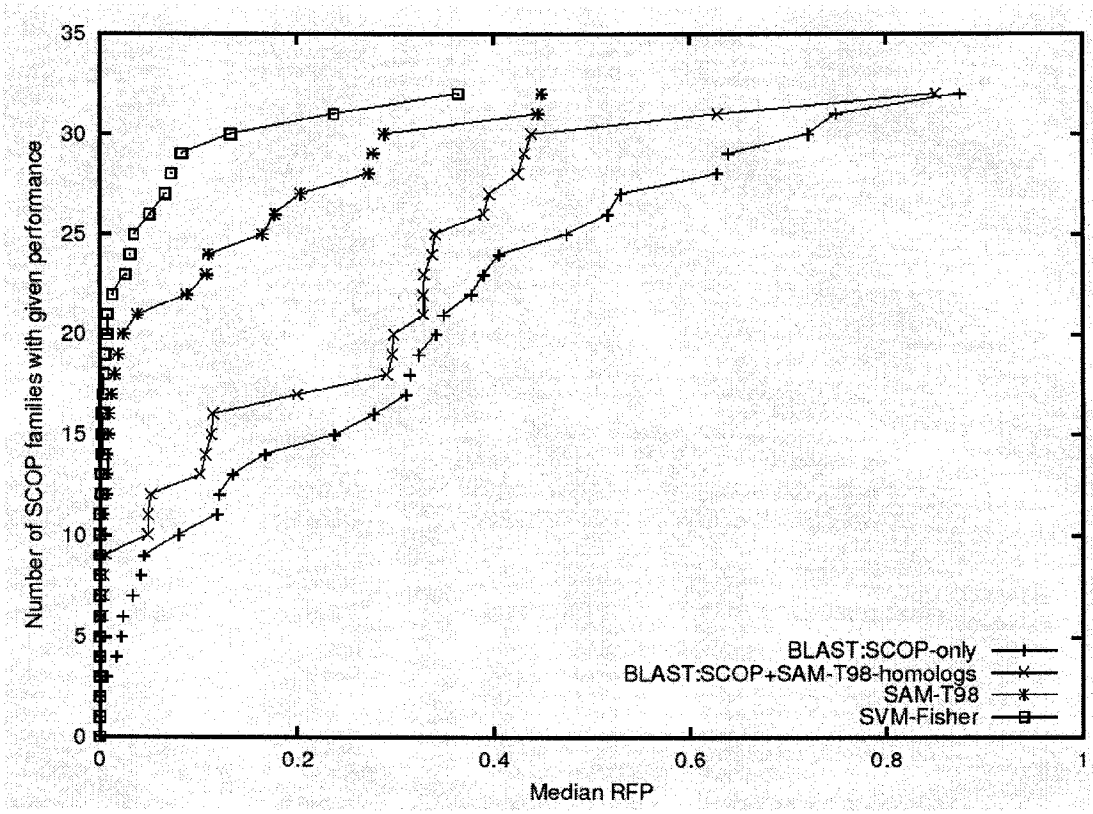


Figure 22 Number of families exceeding a median rate of false positive (RFP) score adopted from "A Discriminative Framework for Detecting Remote Protein Homologies" page 107 Fisher (Jaakkola, Diekhans et al. 1999).

Most researchers now test with the SCOP database. Figure 22 presents results of the SVM-Fisher method by Jaakkola et al., this first application of the support vector machine is compared against past techniques of pairwise (BLAST) and a hidden Markov model (SAM-T98). With this statistic, the faster the curve rises the better, a perfect result would have the line jump to the top left corner, then follow a straight line to the right border. In Figure 23, SVM-Fisher is compared against several of the most current SVM based methods, a hidden Markov model (SAM-T98), and a profile based method (PSI-BLAST). Both figures provide the same statistics, but the database versions differ. In the Figure 23 based experiments the SCOP dataset contains around 20 more families. In this newer figure the results show that the recent work with support vector machines has provided new levels of performance. It is seen that SVM-HMMSTR(Hou, Hsu et al. 2004) is top, followed by SVM-pairwise(Liao and Noble 2002), SVM-I-sites(Hou, Hsu et al. 2003), and SAM(Karplus, Barrett et al. 1998) is followed by SVM-Fisher(Jaakkola, Diekhans et al. 1999). It is important to note that since the publication of Figure 23 further work has pushed the levels of performance higher.

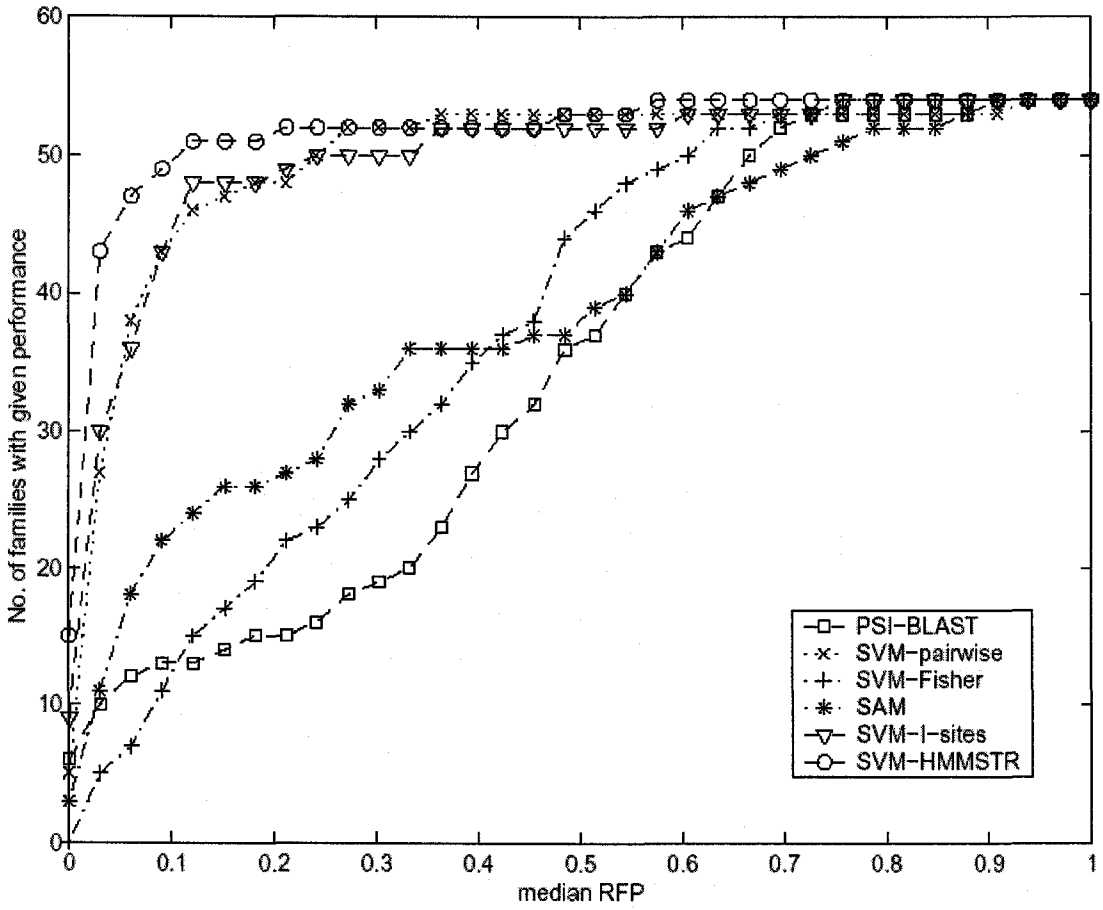


Figure 23 Number of families exceeding a median rate of false positive (RFP) score adopted from "Remote homolog detection using local sequence-structure correlations" page 527 (Hou, Hsu et al. 2004).

IV Summary of Advantages and Disadvantages of Competing Methods

The first naïve methods based on pairwise sequence methods performed database searches. These searches only made use of information from the individual positive samples, a generative approach. Compared to future methods it has the disadvantage of low accuracy, as related proteins can have very low sequence similarity. This method has the advantages of speed when implemented with sequence search tools such as BLAST.

Profile methods were soon developed, these methods combined sequence information from a set of homologous sequences. The profile method has the disadvantages of being position specific and generative. Its advantage of superior accuracy compared to pairwise methods stems from its use of multiple sequences. Motif based methods extend on the profile idea by focusing on local profiles or motifs. Motifs are constructed from regions of local similarity obtained from alignments of multiple sequences, these local regions often represent functionally important features such as binding sites. Again, Motifs provide better accuracy than pairwise methods, primarily due to the use of multiple training sequences.

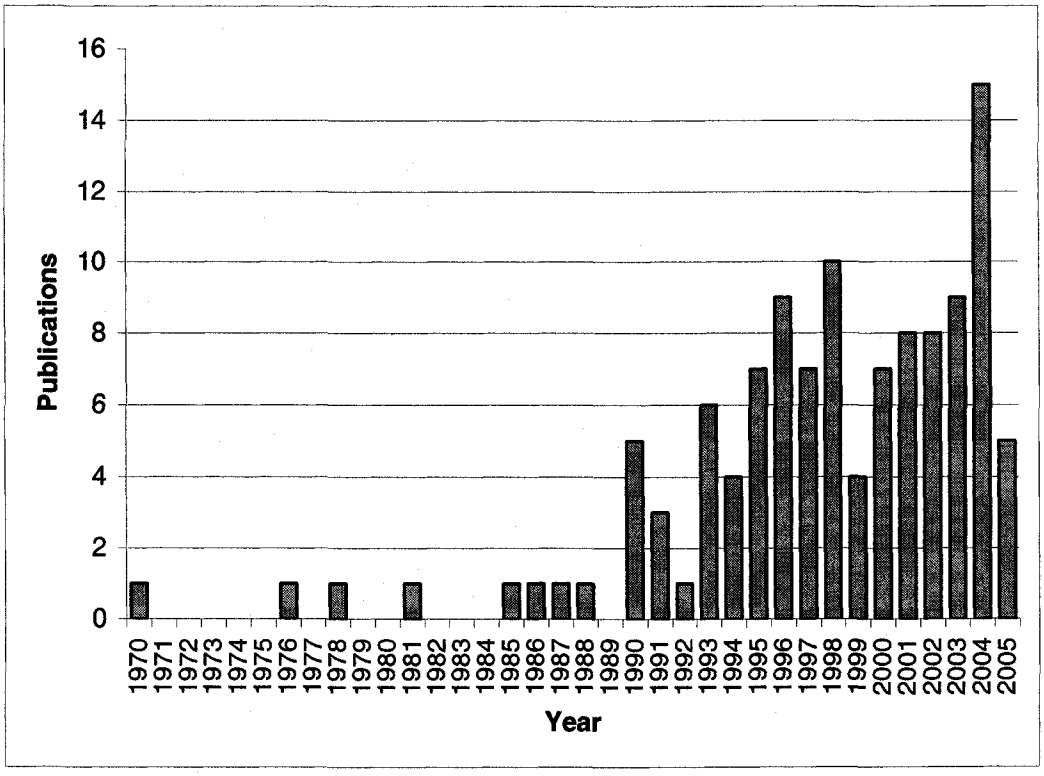
Hidden Markov models extend the work on profiles and motifs by adding an advanced statistical model based upon probabilities. This mathematical underpinning gives HMM's a strong advantage over profiles and motifs. Hidden Markov model approaches have the disadvantages of being generative and slow - they require a multiple sequence alignment on training examples.

Consensus based methods, using more than one classifier to determine if a sequence is homologous trade speed and complexity for slightly better accuracy. Only a few examples of this method exist.

Support vector machines are the most accurate method of remote homology detection. One disadvantage is that they require feature extraction from the training sequences, along with the choice or development of a kernel function. This disadvantage is often offset by combining the previously mentioned methods to produce feature vectors. This combination of previously honed generative models (HMM's, motifs, pairwise) and a state of the art statistical discriminative classifier (SVM) yields very high accuracies. A support vector machine is discriminative because it discriminates between both positive and negative training examples, unlike generative methods. Disadvantages of SVM methods are that they are hard to implement and only perform binary classifications.

V Concluding Comments

The problem of detecting remote homologous sequences and protein superfamily classification has received a tremendous amount of attention. Practically all methods of machine learning have been applied to the problem. Recently research in the area has increased with the application of support vector machines and growth in bioinformatics. The chart below displays this increase with a histogram of references in the final bibliography per year:



This tremendous effort to solve the problem has achieved excellent results. Currently the methods described in this survey are being used in annotation systems, with the help of manual annotation(Wu, Nikolskaya et al. 2004).

The first attempts at the problem involved simple sequence to sequence comparisons. These pairwise searches found few remote homologies, due to the small scope of the search. Further work combined sequences into statistical representations to better represent a protein family or superfamily; this was implemented with motifs, profiles and hidden Markov models. It is important to note that these methods did not incorporate information from negative sequence sets. These newer multiple sequence based techniques produced a three fold increase in accuracy when compared to pairwise methods(Park, Karplus et al. 1998).

Several innovative approaches to the problem were developed. Some examples are: neural networks, linear classifiers, intermediate sequence

searches, and probabilistic suffix trees.

The most recent major innovation introduced the discriminative power of support vector machines to the problem(Jaakkola, Diekhans et al. 1999). This advanced classifier continues to be combined with previous sequence processing techniques to form feature vectors and kernels; these include pairwise, motif, profile and hidden Markov model techniques. Given these sequence statistics and unlabeled data, the SVM based research is still producing new levels of performance.

References

Ahola, V., T. Aittokallio, E. Uusipaikka and M. Vihinen (2003). "Efficient estimation of emission probabilities in profile hidden Markov models." Bioinformatics **19**(18): 2359-2368.

Altschul, S. F., W. Gish, W. Miller, E. W. Myers and D. J. Lipman (1990). "Basic local alignment search tool." J Mol Biol **215**(3): 403-10.

Altschul, S. F., T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nucleic Acids Res **25**(17): 3389-402.

Bailey, T. L. and C. Elkan (1994). "Fitting a mixture model by expectation maximization to discover motifs in biopolymers." Proc Int Conf Intell Syst Mol Biol **2**: 28-36.

Bailey, T. L. and C. Elkan (1995). "The value of prior knowledge in discovering motifs with MEME." Proc Int Conf Intell Syst Mol Biol **3**: 21-9.

Bailey, T. L. and M. Gribskov (1997). "Score distributions for simultaneous matching to multiple motifs." J Comput Biol **4**(1): 45-59.

Bailey, T. L. and M. Gribskov (1998). "Combining evidence using p-values: application to sequence homology searches." Bioinformatics **14**(1): 48-54.

Bairoch, A. and P. Bucher (1994). "PROSITE: recent developments." Nucleic Acids Res **22**(17): 3583-9.

Baldi, P., Y. Chauvin, T. Hunkapiller and M. A. McClure (1994). "Hidden Markov models of biological primary sequence information." Proc Natl Acad Sci U S A **91**(3): 1059-63.

Baxevanis, A. D. and B. F. F. Ouellette (2004). Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Wiley, John & Sons, Incorporated.

Bejerano, G., Y. Seldin, H. Margalit and N. Tishby (2001). "Markovian domain fingerprinting: statistical segmentation of protein sequences." Bioinformatics **17**(10): 927-34.

Bejerano, G. and G. Yona (2001). "Variations on probabilistic suffix trees: statistical modeling and prediction of protein families." Bioinformatics **17**(1): 23-43.

Ben-Hur, A. and D. Brutlag (2003). "Remote homology detection: a motif based

approach." Bioinformatics **19**(90001): 26i-33.

Bystroff, C., V. Thorsson and D. Baker (2000). "HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins." J Mol Biol **301**(1): 173-90.

Can, T., O. Camoglu, K. Singh and Y. Wang (2004). Automated Protein Classification Using Consensus Decision. Computational Systems Bioinformatics, Bioinformatics Conference, Standford, CA.

Cappelli, R., D. Maio and D. Maltoni (2001). "Multispace KL for Pattern Representation and Classification." IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(9): 977-996.

Dayhoff, M. O., R. M. Schwartz and B. C. Orcutt (1978). "A Model of Evolutionary Change in Proteins." Atlas of Protein Sequence and Structure **15**(Supplement 3): 345-358.

Durbin, R., S. R. Eddy, A. Krogh and G. Mitchison (1999). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press.

Eddy, S. R. (1998). "Profile hidden Markov models." Bioinformatics **14**(9): 755-63.

Eddy, S. R., G. Mitchison and R. Durbin (1995). "Maximum discrimination hidden Markov models of sequence consensus." J Comput Biol **2**(1): 9-23.

Eskin, E., W. N. Grundy and Y. Singer (2000). "Protein family classification using sparse Markov transducers." Proc Int Conf Intell Syst Mol Biol **8**: 134-45.

Eskin, E., W. S. Noble and Y. Singer (2003). "Protein family classification using sparse markov transducers." J Comput Biol **10**(2): 187-213.

French, L., A. Ngom and L. Rueda (2005). Fast Protein Superfamily Classification using Principal Component Null Space Analysis. 18th Conference of the Canadian Society for Computational Studies of Intelligence, Victoria, Canada, Springer-Verlag.

Gribskov, M., A. D. McLachlan and D. Eisenberg (1987). "Profile analysis: detection of distantly related proteins." Proc Natl Acad Sci U S A **84**(13): 4355-8.

Gribskov, M. and N. L. Robinson (1996). "The Use of Receiver Operating Characteristic (ROC) Analysis to Evaluate Sequence Matching." Computers and Chemistry(20): 25-34.

Griffiths-Jones, S. and A. Bateman (2002). "The use of structure information to increase alignment accuracy does not aid homologue detection with profile HMMs." Bioinformatics **18**(9): 1243-1249.

Grundy, W. N. (1998). "Homology detection via family pairwise search." J Comput Biol **5**(3): 479-91.

Henikoff, S. and J. G. Henikoff (1994). "Protein family classification based on searching a database of blocks." Genomics **19**(1): 97-107.

Henikoff, S., S. Pietrokovski and J. G. Henikoff (1998). "Superior performance in protein homology detection with the Blocks Database servers." Nucleic Acids Res **26**(1): 309-12.

Hoschek, W. (2000). "Uniform, Versatile and Efficient Dense and Sparse Multi-Dimensional Arrays."

Hou, Y., W. Hsu, M. L. Lee and C. Bystroff (2003). "Efficient remote homology detection using local structure." Bioinformatics **19**(17): 2294-2301.

Hou, Y., W. Hsu, M. L. Lee and C. Bystroff (2004). "Remote homolog detection using local sequence-structure correlations." Proteins **57**(3): 518-30.

Hughey, R. and A. Krogh (1996). "Hidden Markov models for sequence analysis: extension and analysis of the basic method." Comput Appl Biosci **12**(2): 95-107.

Jaakkola, T., M. Diekhans and D. Haussler (1999). "Using the Fisher kernel method to detect remote protein homologies." Proc Int Conf Intell Syst Mol Biol: 149-58.

Jaakkola, T., M. Diekhans and D. Haussler (2000). "A discriminative framework for detecting remote protein homologies." J Comput Biol **7**(1-2): 95-114.

Joachims, T., B. Schölkopf and C. Burges (1999). Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, MIT-Press.

Karplus, K., C. Barrett and R. Hughey (1998). "Hidden Markov models for detecting remote protein homologies." Bioinformatics **14**(10): 846-56.

Karplus, K. and B. Hu (2001). "Evaluation of protein multiple alignments by SAM-T99 using the BAliBASE multiple alignment test set." Bioinformatics **17**(8): 713-720.

Krogh, A., M. Brown, I. S. Mian, K. Sjolander and D. Haussler (1994). "Hidden Markov models in computational biology. Applications to protein modeling." J

Mol Biol **235**(5): 1501-31.

Leslie, C., E. Eskin and W. S. Noble (2002). "The spectrum kernel: a string kernel for SVM protein classification." Pac Symp Biocomput: 564-75.

Leslie, C. S., E. Eskin, A. Cohen, J. Weston and W. S. Noble (2004). "Mismatch string kernels for discriminative protein classification." Bioinformatics **20**(4): 467-476.

Liao, L. and W. S. Noble (2002). Combining pairwise sequence similarity and support vector machines for remote protein homology detection. Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology.

Lipman, D. J. and W. R. Pearson (1985). "Rapid and sensitive protein similarity searches." Science **227**(4693): 1435-41.

Madera, M. and J. Gough (2002). "A comparison of profile hidden Markov model procedures for remote homology detection." Nucleic Acids Res **30**(19): 4321-8.

Murzin, A. G., S. E. Brenner, T. Hubbard and C. Chothia (1995). "SCOP: a structural classification of proteins database for the investigation of sequences and structures." J Mol Biol **247**(4): 536-40.

Needleman, S. B. and C. D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." J Mol Biol **48**(3): 443-53.

Park, J., K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard and C. Chothia (1998). "Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods." J Mol Biol **284**(4): 1201-10.

Park, J., S. A. Teichmann, T. Hubbard and C. Chothia (1997). "Intermediate sequences increase the detection of homology between sequences." J Mol Biol **273**(1): 349-54.

Pearson, W. R. (1990). "Rapid and sensitive sequence comparison with FASTP and FASTA." Methods Enzymol **183**: 63-98.

Pearson, W. R. (1991). "Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms." Genomics **11**(3): 635-50.

Pollastro, P. and S. Rampone (2003). "HS3D: Homo Sapiens Splice Site Data Set."

Nucleic Acids Research(Annual Database Issue).

Rueda, L. and A. Ngom (2004). An Empirical Evaluation of the Classification Error of Two Thresholding Methods for Fisher's Classifier. International Conference on Artifical Intelligence & International Conference on Machine Learning; Models, Technologies & Applications, Las Vegas, Nevada, USA, CSREA Press.

Smith, T. F. and M. S. Waterman (1981). "Identification of common molecular subsequences." J Mol Biol **147**(1): 195-7.

Vaswani, N. (2002). "A Linear Classifier for Gaussian Class Conditional Distributions with Unequal Covariance Matrices." Intl. Conference on Pattern Recognition (ICPR) **I**: 240.

Vaswani, N. and R. Chellappa (2004). Classification Probability Analysis of Principal Component Null Space Analysis. Intl. Conference on Pattern Recognition (ICPR), Cambridge, UK.

Vaswani, N. and R. Chellappa (2004). "Principal Component Null Space Analysis for Image and Video Classification." IEEE Transactions on Image Processing.

Venter, J. C., K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, D. E. Fouts, S. Levy, A. H. Knap, M. W. Lomas, K. Nealson, O. White, J. Peterson, J. Hoffman, R. Parsons, H. Baden-Tillson, C. Pfannkoch, Y.-H. Rogers and H. O. Smith (2004). "Environmental Genome Shotgun Sequencing of the Sargasso Sea." Science **304**(5667): 66-74.

Wang, J. T. L., Q. Ma, D. Shasha and C. H. Wu (2001). "New techniques for extracting features from protein sequences." IBM Systems Journal **40**(2).

Witten, I. and F. Eibe (2005). Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, San Francisco.

Wu, C., A. Ermongkonchai and T. C. Chang (1991). Protein Classification Using a Neural Network Protein Database (NNPDB) System. Proc. Anal. Neural Net Appl. Conf.

Wu, C., G. Whitson, J. McLarty, A. Ermongkonchai and T. C. Chang (1992). "Protein classification artificial neural system." Protein Sci **1**(5): 667-77.

Wu, C. H., M. Berry, Y. S. Fung and J. McLarty (1995). "Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with Singular Value Decomposition." Machine Learning **21**: 177-193.

Wu, C. H., A. Nikolskaya, H. Huang, L. S. Yeh, D. A. Natale, C. R. Vinayaka, Z. Z. Hu, R. Mazumder, S. Kumar, P. Kourtesis, R. S. Ledley, B. E. Suzek, L. Arminski, Y. Chen, J. Zhang, J. L. Cardenas, S. Chung, J. Castro-Alvear, G. Dinkov and W. C. Barker (2004). "PIRSF: family classification system at the Protein Information Resource." Nucleic Acids Res **32**(Database issue): D112-4.

Wu, C. H., L. S. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, P. Kourtesis, R. S. Ledley, B. E. Suzek, C. R. Vinayaka, J. Zhang and W. C. Barker (2003). "The Protein Information Resource." Nucleic Acids Res **31**(1): 345-7.

Zhang, X. (2004). Protein Family Classification Using Multiple-Class Neural Networks. Faculty of Graduate Studies and Research. University of Windsor.

Appendix B: Tool Report for Biojava

60-520
Tool Report on BioJava
By Leon French

Fall, 2004

Introduction

Recently the world of biology has gained the ability decode or read large amounts of data from many organisms. This genetic data is a sort of blueprint or source code of an organism which is encoded with the discrete alphabets of RNA, DNA or amino acids. This data is legion in size; approaching gigabyte quantities with ease, to analyse and understand this data biologists have enlisted the power of computers. This new area of research has been named Bioinformatics. Several toolkits have been created to aid the creation Bioinformatics software such as Bioperl, Biopython and BioJava. BioJava which is a java based bioinformatics API will be the focus of this report.

Characteristics of the Tool

BioJava is a bioinformatics based API for the java language. The official BioJava website describes the tool very well[1]:
BioJava is an open-source project dedicated to providing a Java framework for processing biological data. It include objects for manipulating sequences, file parsers, DAS client and server support, access to BioSQL and Ensembl databases, and powerful analysis and statistical routines including a dynamic programming toolkit.

BioJava was founded by Matthew Pocock and Thomas Down in 1998; other main contributors are Micheal Heuer, David Huen, and Mark Schreiber. The current version of BioJava contains over three thousand files and is growing with the help of constant contributions. The project is licensed under the Lesser GPL, so its code can be used in non-free software.

The version used for this report is 1.4 pre-release 1, which is designed for java 1.4 SDK. A newer version is currently under development that will make use of the java 1.5/5.0 SDK features.

Detailed Case Study

Problem and Requirements

Create an application that reads in a PIR-International Protein Sequence Database (PSD) XML file and creates an input dataset for a pattern recognition algorithm. This requires filtering the dataset for proteins of specific protein superfamilies. Only identifiers and sequence data are to be extracted. The sequence data for each protein will be processed in several ways to produce the output data. Output shall be formatted in Attribute-Relation File Format or ARFF for use with the WEKA machine learning software[2].

The application will be used to create a dataset from the protein sequence database (PSD) release 79.05 at the protein information resource (PIR) databank. PSD provides fully annotated protein data in XML format for over 280,000 sequences. For this application, only the identifier, sequence, sequence type and superfamily of the entries were used. Some entries in the databank only have the sequence of a protein fragment, or are ambiguous in describing the sequence (e.g. GLS(D.G.E)WXQL). All complete non-ambiguous sequences of the four selected superfamily classes were processed.

The four classes to be collected and their size are ras transforming proteins (455), kinase-related transforming proteins (517), globin proteins (672) and ribitol dehydrogenase proteins (868). Although the PIR-PSD database entries contain one or more superfamily classifications, none of the selected data subsets intersect. Two datasets were created: a two-class dataset containing kinase and ras transforming proteins (972), and a second multiclass dataset that includes all four classes mentioned above (2,512).

The string sequence data of each protein was processed to create an array of 465

numeric features plus the class label. At a high level, the features that represent a sample are:

Frequency of amino acids and exchange groups (1-gram)

- 436 Amino Acid 2-gram attributes[3]
- 49 Exchange Group 2-gram attributes
- Length of the sequence
- Mass of the peptide encoded by the sequence
- pI of the peptide encoded by the sequence
- Ability to randomly separate dataset in to two parts

All of these features were generated directly from the sequence string. The pI and mass features are estimates based on the polypeptide encoded by the sequence. Originally, the dataset contained only two-grams and exchange two-grams. As the work progressed, more data was added with the resulting accuracies increasing.

The two-gram features account for the majority of the attributes. They represent the frequencies of every consecutive ``two-letter'' sequence in the protein sequence. Two grams have the advantages of being length invariant, insertion/deletion invariant, not requiring motif finding and allowing classification based on local similarity.

Exchange grams are similar but are based on a many-to-one translation of the amino acid alphabet into a six letter alphabet that represents six groups of amino acids, which represent high evolutionary similarity. Exchange groups used for this dataset are: $e1$={H, R, K}, $e2$={D, E, N, Q}, $e3$={C}, $e4$={S, T, P, A, G}, $e5$={M, I, L, V} and $e6$={F, Y, W}. The exchange groups are based on information from the point accepted mutations (PAM) matrix, which statistically describes the probability of one amino acid replacing another over time.

Given an example sequence "GLALLA" the non-zero two-grams are GL=1, LA=2, AL=1 and LL=1. Translating "GLALLA" to an exchange group sequence results in "*e*4*e*5*e*4*e*5*e*5*e*4", with the resulting exchange two-grams of *e*4*e*5=2, *e*5*e*4=2, and *e*5*e*5=1. The frequency of the amino acids and exchange groups are also added to the

dataset entry, and result in G=1, L=3, A=2, *e*4=3, and *e*5=3.

The program should be easily extended so that more attributes can be added to the dataset. Performance of the application is not a priority as dataset creation will only be performed occasionally. Usability is also not a priority because the program only to be used occasionally and by an expert user. The program should be written to handle certain special cases and anomalies in the input dataset and do so gracefully. All the sequences outputted must not be fragments or contain the X amino acid symbol. Any non conforming proteins are to be excluded from the output dataset.

Analysis

From the problem definition it is seen the task is a simple file in and file out type setup. All of the data output is based on a single independent instance of the input data this suggests the data can be processed on the fly. A fast java XML parser will be required along with BioJava's sequence processing abilities. The Simple API for XML (org.xml.sax) was chosen for this purpose, primarily for its speed. Speed is not required but since the input file is 776Mb in size, it will help to have a fast XML parser.

Design

The design should be loosely object oriented because it is a simple data in, data out setup. This suggests procedural programming approach. On the other hand java lends itself to an object oriented approach and the problem definition requires extensibility.

Class Main: a simple main class that gets the process started with necessary parameters

Methods:

Main – initializes the XML handler and calls parseXMLFile

parseXMLFile – executes the XML parsing and handles exceptions

Class PSDXMLHandler:

Methods:

PSDXMLHandler – constructor method, primarily writes header into the output file

outputLine – outputs one data output instance, the file it writes to depends on a randomly generated number

finish – called when parsing is complete, closes output files

startElement – called when an XML element is initially encountered

endElement – called when an XML element is finished reading

characters – called when element data is read, this is where the on the fly processing is done, characters of the sequence are read here and then the sequence is processed and written out to file.

Class Gram: an abstract class that is extended for two gram and exchange gram classes

Methods:

compute – generates a hash table of grams as keys and the normalized frequency of the gram as the value, then outputs a string representing the hash table.

expandHash – called by compute, converts a hash table of grams into a coma separated string of frequency values

Class TwoGram: this class is small as most of its work is done in the Gram class

Methods:

Constructor – initializes size of gram and alphabet (protein) used.

getFormatS – returns the format string, or the attribute sequence it returns when compute is called.

Class ExchangeGram: this class is similar to TwoGram except its constructor is more complex because a custom alphabet is used.

Methods:

Constructor – creates a custom alphabet that defines the exchange groups and translates the input protein sequence into the new alphabet.

getFormatS – returns the format string, or the attribute sequence it returns when

compute is called.

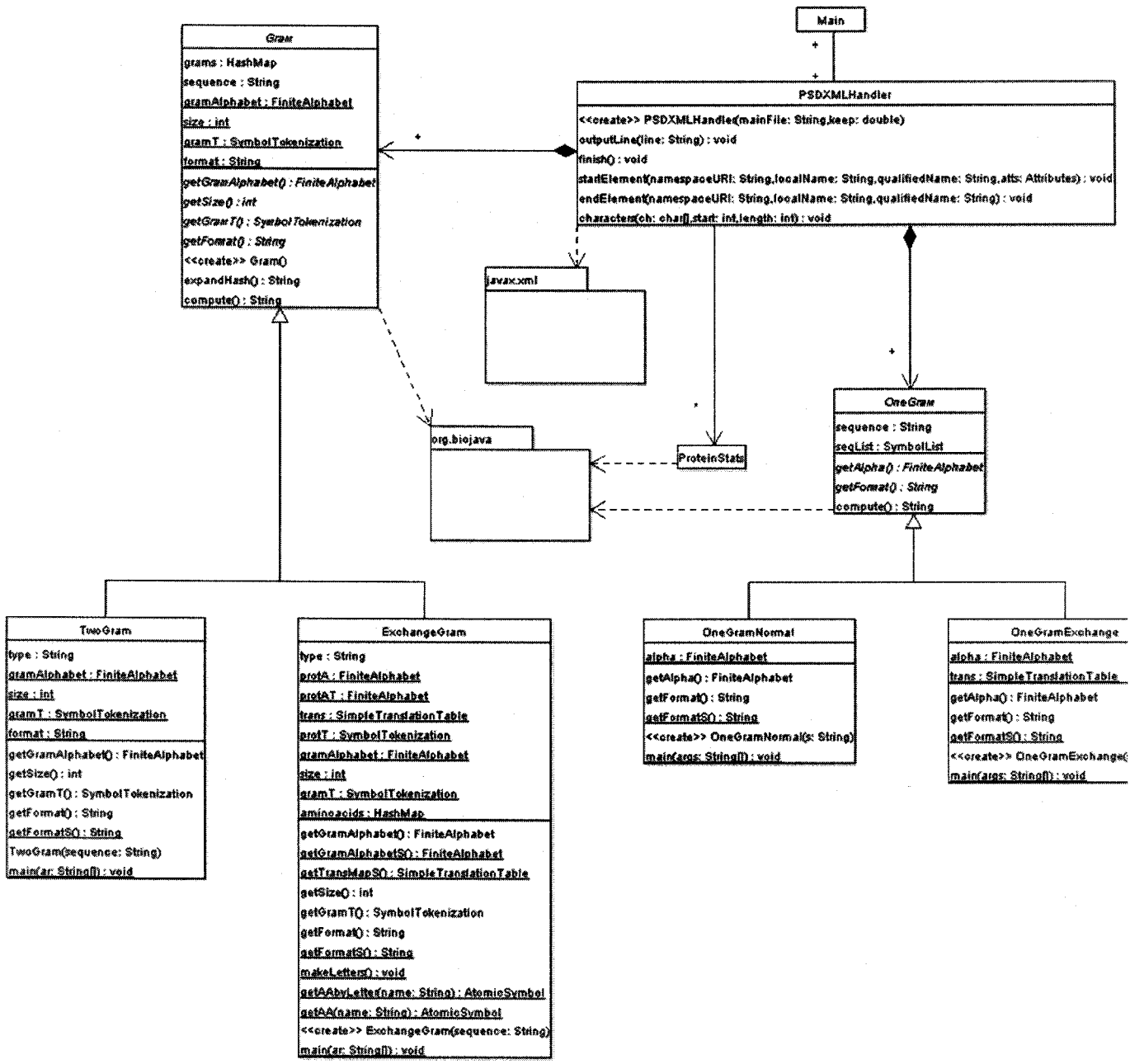
Class ProteinStats: same basic setup as Exchange and TwoGram

Methods:

compute – returns length of sequence, and the mass and isoelectric point (pI) of the peptide.

getFormatS – returns the format string, or the attribute sequence it returns when compute is called.

To visualize these class relationships an UML diagram is provided on the next page.



Development and Implementation

Programming of the application was spread across several months and took on an iterative development path. As the work continued the application was refactored and many features were added as the problem developed. Although the code was written from the start to be extendable some refactoring was required to accommodate additional features. For example the original problem did not require a dataset split or arff output format. This iterative process is evident in the final code as on reflection it can be seen that the ProteinStats and Gram classes should belong to a super class since they all output one or more attributes based on sequence data.

Implementation platform used was the Textpad[4] text editor running under Windows XP. Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_01-b06) was used for the java virtual machine, and the BioJava version used was 1.4pre1 development release dated May 31, 2004.

Programming with the API was relatively easy but tricky at times. There is not a large amount of help available for BioJava and inexperienced java programmers may have problems starting out. The learning curve is a bit steep due to the advanced design methods used, for example the symbolist representation of a sequence instead of a String or char array.

The sequence representation in BioJava involves FiniteAlphabets, Sequences, Symbols, AtomicSymbols, SymbolTokenizations, and SymbolLists. This complex design took awhile at first to grasp, but once the object oriented relations are understood it becomes clear. This is in stark contrast to other toolkits which would just use a standard String as a sequence representation instead of the several classes above or just an object that encapsulates a string. This method has its advantages and disadvantages. A primary disadvantage is that each residue will take 4 bytes in memory, as opposed to 1 byte for a character in a string. The 256 possibilities a byte provides is already far beyond the protein alphabet of 21 characters. An advantage of the 4 byte object representation is for equality testing, and ambiguity symbol

representation. For example an 'a' in a DNA sequence is not equal to an 'a' in a Protein sequence, a BioJava sequence will maintain this property, but a simple character in a string will not. Here we can see BioJava's complex object oriented design styles can lead to a steep learning curve for the programmer. Matthew Pocock, co-founder of BioJava agrees with this criticism and provides information that BioJava version two will contain new approach to sequences similar to the CharSequence class. He also noted that "string representations can't handle tuples of symbols (e.g. a codon)", such tuple representation is common and was needed in this report for the gram generation.

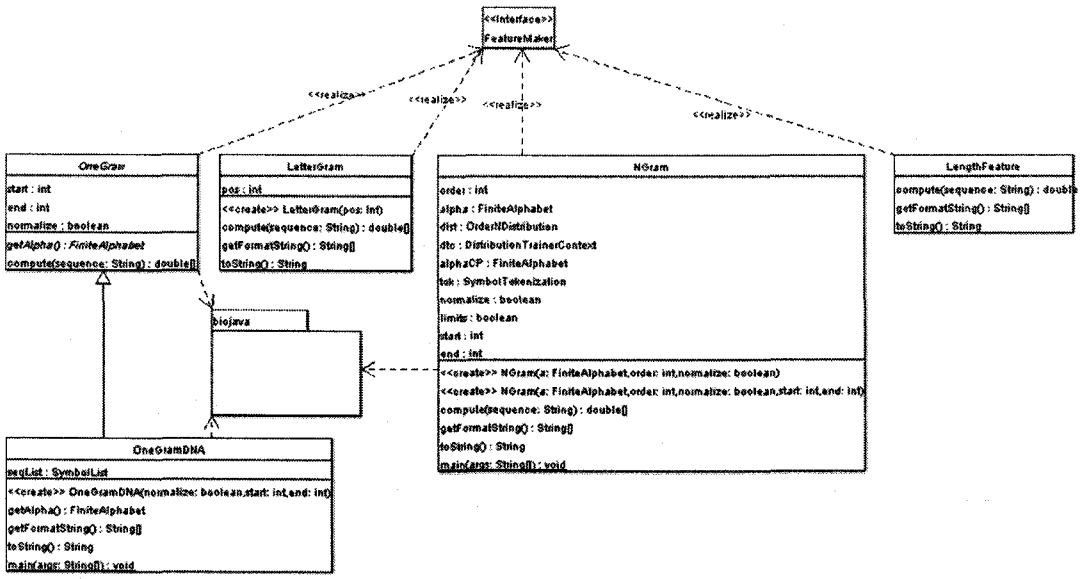
The most useful resource I found was "BioJava in Anger"[5] by Mark Schreiber. BioJava in Anger is similar to the java almanac. It provides sample code solutions to common problems. Like all java packages the API is essential for BioJava programming, the BioJava API is complete but very sparse in some areas. Additional help with programming the application in BioJava was provided by the IRC channel for BioJava (#BioJava) which is hosted on the freenode IRC network.

Several times I had problems finding the method I needed and ended up coding my own and later finding the class I needed in the API. The first example is when I needed amino acid symbol objects for each of the 21 acids, this is coded in for the DNA acid bases in the DNATools class, but was absent in ProteinTools. I later discovered the symbols had been brought into a newer version of BioJava after I wrote my own code to produce the symbols. The second case involved code for the twogram generation, where I needed to create a count and distribution based on the alphabet cross product. After doing a quick implementation myself I found BioJava has coded such classes already, specifically the OrderNDistribution class. This mistake was mainly due to user error as BioJava in Anger contained sample code for such a problem.

Testing and Verification

Several verification and testing procedures were performed on the application and its output. Primarily a second implementation designed for the same task using

DNA this was adapted to work with the protein alphabet. This second implementation was written by the same author and relies more heavily upon the biojava API, but its design is different than in this report as seen in the UML diagram below. Given the difference in design and biojava usage the probability that both implementations have the same bugs is very low.



The second implementation was used to test against the main implementation for the ExchangeGram and TwoGram classes. Using this second implementation only a small rounding difference was discovered, in the Exchange gram class. This difference occurred only at the least significant digit and was considered negligible; to further verify it did not occur with un-normalized output. Along with checking the output matched between implementations other checks were performed. For example output was checked to ensure the produced numbers summed up to one when processing normalized gram/distribution values.

The ProteinStats and OneGram classes were briefly tested manually with unit tests in there main methods, the results were verified manually. It is important to note that these classes perform simple tasks and are direct API calls to the biojava package. They do not need exhaustive testing as the biojava test cases have already covered these API calls.

To summarize:

OneGram – summation test, manual unit testing, biojava tested (API calls).
TwoGram – second implementation comparison, summation test, manual unit testing, biojava tested (API calls).
ExchangeGram – second implementation comparison, summation test, manual unit testing, biojava tested (API calls).
ProteinStats – mass compared to measured, manual unit testing, biojava tested (API calls).

Above a unit level, global scope is was tested by visually inspecting the data and comparing the classification results. Several other algorithms have used this dataset for classification and made the results (accuracy) available, these accuracies compare as expected to classification results performed with the input data created by this application. Additionally exhaustive holistic testing is not needed because each instance of the XML data is transformed into one line in the output file, no significant interaction takes place beyond this unit level. The main program basically consists of a for loop and the XML parser. This explains the large amount of unit testing that was performed.

Comparison with Other Tools in the Same Category

BioJava is compared to BioPerl and BioPython below, they are written for the Perl and Python languages respectively. These are both regarded as scripting languages while java is a full programming language so many differences exist. These differences between languages will not be compared here, just the differences of the API itself. A more detailed comparison that considers the underlying languages can be found in "The Bio* toolkits – a brief overview by Harry Mangalam[6]. BioRuby is another Bioinformatics toolkit; it is very small and currently not very popular. BioRuby has been left out of this comparison for briefness.

BioPerl
Older and more established, larger codebase
More popular and most commonly used

Most documented of the toolkits due to the above points

Interfaces to outside programs like the EMBOSS suite

Scripting language so code is faster to write

Smaller residue data representation than BioJava (1 byte versus 4)

BioPython

Martel library for fast format parser creation

Largest library of format parsers

Started around the same time as BioJava

More documentation than BioJava

Modules for working with Structural and Microarray data

Smaller residue data representation than BioJava (1 byte versus 4)

Discussion of Advantages and Disadvantages of BioJava

Advantages

- XML integration
- Advanced design and structure (eg. ChangeListeners and byte-code generation) of classes and packages to help a programmer create extensible and maintainable code.
- Advanced packages that implement complex algorithms or concepts: a Support Vector Machine, dynamic programming and hidden Markov models.
- Event based format parsers and other advanced designs
- Large amount of useful biology based GUI classes
- Several advantages of the java language are inherited – portability, standardized, large library, very strongly typed, and object oriented syntax.

Disadvantages

- Lacking end-user code

- Lacks interfaces to independent applications such as BLAST, Clustalw or the EMBOSS suite
- Lacking example or tutorial based documentation
- Lacking certain parts of a full bioinformatics toolkit - limited support for Structural and Microarray data.
- Several disadvantages of java are inherited such as large footprint and syntax of the language. Regarding syntax, a biologist may find it easier to learn Perl or Python than the java language.

The Future of BioJava

Currently BioJava version two is in the works, it is based on Java 5.0 (previously 1.5). Java 5.0 provides a very large extension of java, allowing BioJava to also extend its ability and design in new and better ways. A quote from Matthew Pocock explains:

Java <5 doesn't have parametric types, so you can't have a SymbolList over FiniteAlphabet, or have an ambiguity symbol over BasisSymbol. So the complexity propagates at each level :/ . BJv2 uses parametric types (by abusing generics) to hide most of this mess from the end-user, so that most of the time they don't notice any of this nastiness. (Matthew Pocock, #BioJava IRC channel)

The nastiness is explained in further detail in the Development and Implementation section of this report. BioJava version two is a complete rewrite of BioJava with refactoring at the byte-code generation level.

The BioJava community is also currently working on Microarray and extended structure support. These areas are currently lacking in the current BioJava release, but are an important part of a comprehensive bioinformatics API.

Conclusion

BioJava is an advanced bioinformatics toolkit suitable for experienced java programmers. BioJava is still in its infancy and lacks end user documentation, except 'BioJava in anger' which was created because of this problem, hence the 'anger'. BioJava provides a comprehensive javadoc which experienced programers expect and will find useful. BioJava uses advanced design methodologies that make the code more maintainable and extensible but in return can make it daunting for a beginner programmer.

BioJava is considered one of the top three bioinformatics toolkits along with Bioperl and Biopython. Bioperl still gains in popularity and size. Biopython has a main advantages are maintainability and end user friendliness. BioJava gains ground with its solid java base and intelligent design.

BioJava is recommended for an experienced java programmer that needs to write large and complex bioinformatics programs. Such an application was written for this report and was found to be a suitable tool to use for the job. Implementation of the application went smooth once the appropriate classes were found and a clear understanding of the API was acquired.

References

1 Official website for the BioJava Project, http://www.BioJava.org

2 Attribute-Relation File Format (ARFF) definition, http://www.cs.waikato.ac.nz/~ml/weka/arff.html

3 J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. IBM Systems Journal (Special Issue on Deep Computing for the Life Sciences) 40(2), 426--441, 2001

4 TextPad Homepage, http://www.textpad.com

5 Mark Schreiber, BioJava in Anger, http://www.BioJava.org/docs/bj_in_anger/

6 Mangalam H. The Bio* toolkits--a brief overview. *Brief Bioinform. 2002 Sep;3(3):296-302*

Appendix C: Assignment based on Highest and Lowest Ten dataset

Dataset

The dataset to be classified in this assignment is data generated from protein sequences. From the view of a computer scientist a protein sequence is just a string made up of a 21 letter alphabet (amino acids). Each protein used has a superfamily classification determined by biologists; the goal is to correctly classify protein instances into the correct superfamilies.

Since the sequences are variable in length using them directly (each attribute is a letter in the sequence) as data is impractical because this would lead to missing data and improperly aligned data as each attribute would line up to different spots in the protein. A successful way of converting the protein sequences into data instances of constant length known as 2-grams has been developed[1]. Using the frequency of the amino acids, or letters for analysis is a method that has been used since the discovery of DNA. The two grams method extends this to 2 consecutive letters, so instead of a discrete distribution of 20 letters, it is a much larger distribution across 400 combinations of two letters/amino acids. These two grams have been extended further into exchange grams which take the evolutionary similarity between the letters into account; this lessens the letters in the alphabet to 6, and 36 for the order of two. These two grams, and one grams are combined to produce the input dataset to the classifiers.

Each instance in the processed dataset I will be using consists of:

- superfamily class which is either ras transforming protein, Kinase-related transforming protein, Globin or Ribitol dehydrogenase.
- Length of the sequence
- Mass of the peptide produced by the sequence, this is computed given the mass of each amino acid.
- pI: isoelectric point of the peptide produced by the sequence.

- amino acid frequency: distribution of the 20 amino acids, normalized.
- exchange group frequencies: distribution of the 6 exchange groups, normalized.
- amino acid twogram frequencies: distribution of the amino acid twograms, normalized.
- exchange group twogram frequencies: distribution of the twogram exchange groups, normalized.

PCNSA

The protein sequence dataset as described above was originally created for evaluation of the Principal Component Null Space Analysis (PCNSA)[2] classifier. PCNSA involves first reducing noise and dimensionality by performing PCA on all class data. The second step then finds a null space for each class, the null space is extracted by taking the dimensions with the least variance of each class using eigen value decomposition. A simple distance from the unclassified sample to the mean of each class inside the class null space is the classification metric; the class that minimizes this distance is the predicted class. Both the PCA and the null space dimension extract will reduce the dimensionality of the dataset, or at least keep it the same. This dimension reduction allows for classification of datasets with many attributes, like the protein dataset. Other classifiers normally do not cope well with datasets of such dimensionality – "The Curse of Dimensionality"[3].

PCNSA was used make the dataset more manageable for the tested classifiers. This was done by extracting the highest weighted dimensions that PCNSA uses to classify. Additionally a contrasting dataset was created that took the lowest weighted dimensions used by PCNSA. This provides grounds for a hypothesis:

> The dataset based on the highest weighted attributes, deemed most important for classification by PCNSA will provide higher accuracy than the lower weighted attributes. The accuracies refer to the precision data produced when other classifiers are trained and tested with the two contrasting datasets.

Intuitively this makes sense, given PCNSA achieves 99.5% accuracy on this full dataset then its weights used in its computations must provide good discrimination. It is also hypothesized that all tested classifiers will achieve lower than 99.5% accuracy.

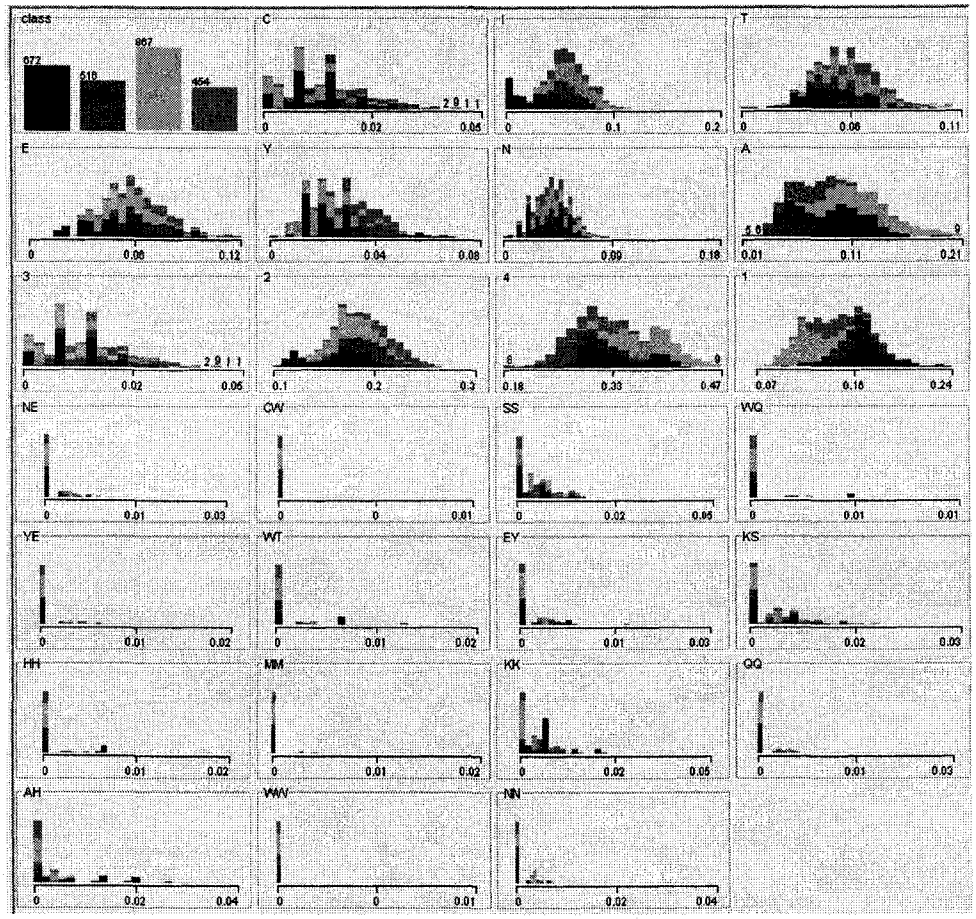
Extracting the weight values of PCNSA was not a simple task. First the dataset was normalized so that the weights would be in proper scale, next PCNSA was ran to find the optimal choice for the PCA keep value and Null space size parameters. The parameters of 200 and 150 were chosen to produce the weight vector, the highest ten weights in each null space for each class were added to a set as to remove duplicates and the same for the lowest ten sorted by absolute value. This resulted in 26 attributes on the high side, and 37 on the lower. This shows that there were many duplicates for the best attributes, while the lower attributes were more diverse. Below is the table of the attributes used for each dataset, in unsorted order.

Highest Weighted	Lowest Weighted
C	IW
I	EM
T	ED
E	ER
Y	QT
N	PD
A	AT
3	GF
2	YS
4	DP
1	NG
NE	NP
CW	MP
SS	GQ
WQ	FR
YE	YQ
WT	TM

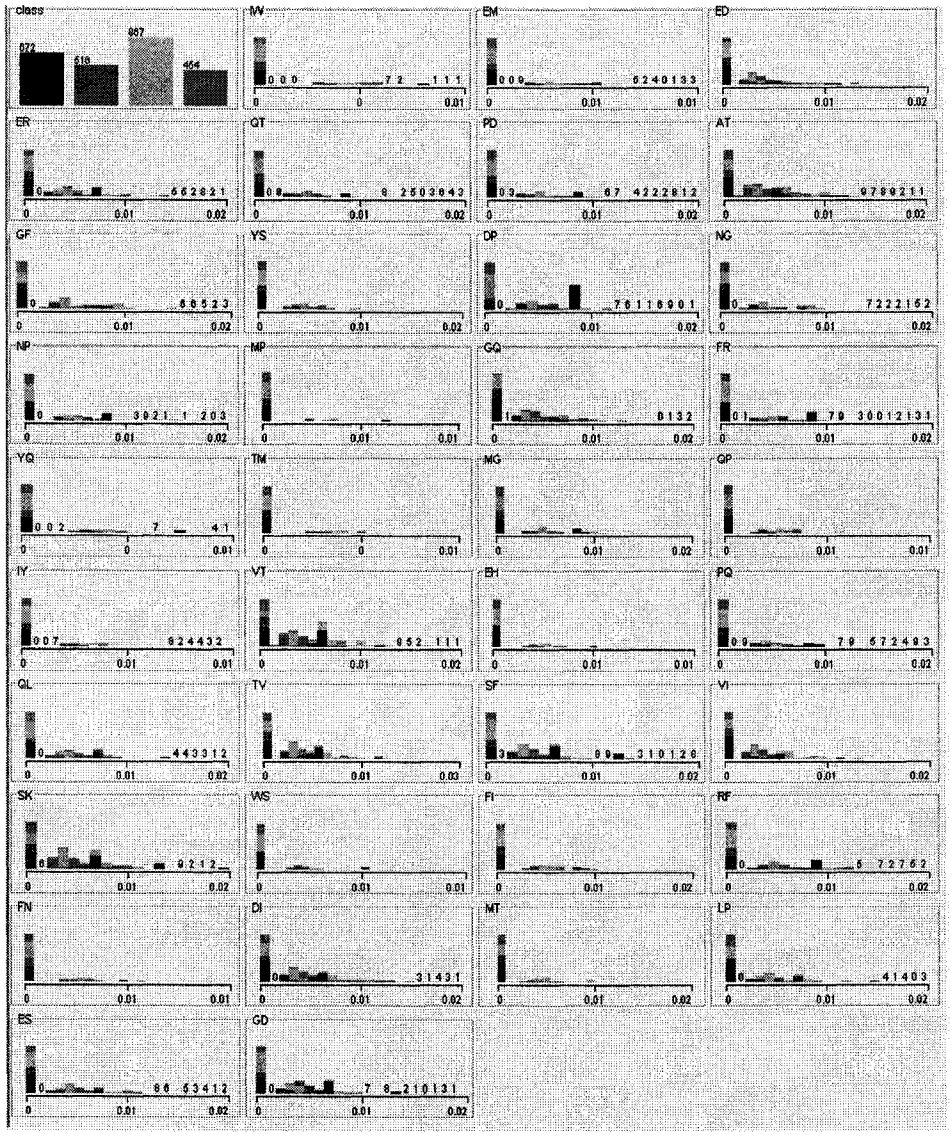
EY	MG
KS	QP
HH	IY
MM	VT
KK	EH
QQ	PQ
AH	QL
WW	TV
NN	SF
	VI
	SK
	WS
	FI
	RF
	FN
	DI
	MT
	LP
	ES
	GD

Histogram diagrams provided by WEKA[4] show the distributions of each attribute in each set:

Highest Ten Dataset



Lowest Ten Dataset



One significant note about both datasets is the large amount of zero's for most attributes; this suggests other approaches maybe suitable for formatting the data. From the diagrams, the higher weighted attributes seem to suffer less from this high distribution of zeroes.

Classification

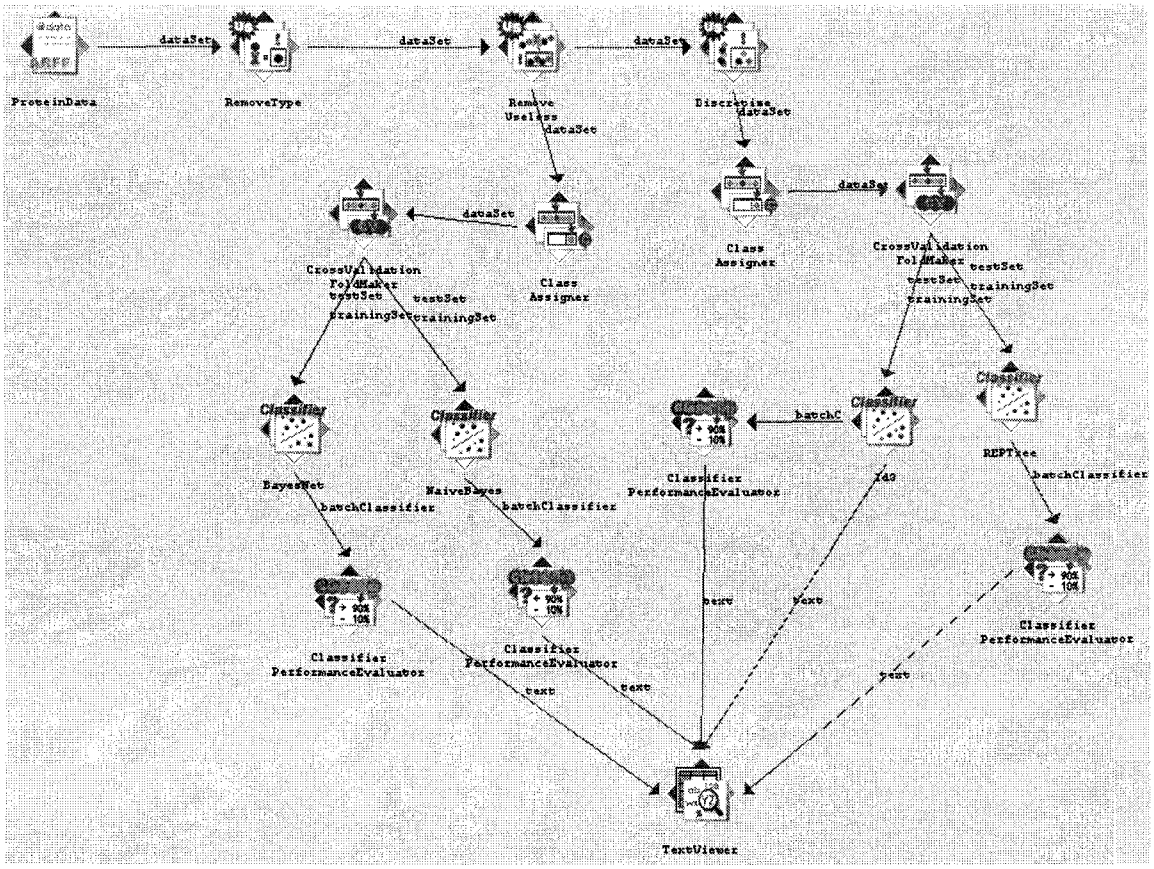
The classifiers used in this experiment were BayesNet, NaiveBayes, REPTree and ID3 tree. All of these classifiers were previously implemented in WEKA. WEKA

describes each classifier as:

- BayesNet: "Bayes Network learning using various search algorithms and quality measures."
- NaiveBayes: "Class for a Naive Bayes classifier using estimator classes."
- REP Tree: "Fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting)."
- ID3 Tree: "Class for constructing an unpruned decision tree based on the ID3 algorithm. Can only deal with nominal attributes."

Both BayesNet and NaiveBayes were given the continuous numeric dataset whilst the two tree based classifiers were tested with a discretized version of the datasets. The conversion into discrete dataset was accomplished using WEKA's Discretise filter with bucket size set to 15. All classifiers were executed in WEKA with default parameters.

The experiments were implemented using the knowledge flow interface of WEKA. All results were collected from one run of ten fold cross validation of the entire dataset of size 2509 instances. Below is a screen shot of the setup used:



Results

Below are the results of the experiments, the parenthesized numbers representing the number of attributes in each dataset, not counting the class attribute. The columns are ordered in excepted accuracy, in ascending order.

Classifier	Full Dataset (465)	Highest Ten (26)	Lowest Ten (37)
PCNSA	99.56%		
ID3	92.95%	74.73%	85.81%
REPTree	95.22%	78.72%	85.65%
NaiveBayes	94.62%	83.90%	77.68%
BayesNet	93.10%	95.22%	91.87%

In this table two unexpected results can be seen. First BayesNet gained accuracy by

going to the smaller dataset (Highest Ten). All classifiers performed considerably better on the full dataset compared to the other two, except BayesNet and all classifiers failed to beat PCNSA's accuracy. Second is that both trees performed better on the attributes that were weighted lower by PCNSA, which contradicts the hypothesis, an interesting result. One possible explanation is that the lower weighted dataset had 11 more attributes to base classification on, hence giving it quantity over quality of the attributes.

The next set of results is for the tree's created by ID3 and REP algorithms, reported is the maximum depth across all folds, and approximate size of the tree in nodes.

Full Dataset (465)

Classifier	**Max Depth**	**Size**
ID3	3	300-400
REP	4	100-130

Highest Ten Dataset (26)

Classifier	**Max Depth**	**Size**
ID3	5	>1000
REP	5	310-350

Lowest Ten Dataset (37)

Classifier	Max Depth	Size
ID3	6	350-650
REP	7	350-400

Remember REP is a pruned tree whilst ID3 is not, this explains the differences in size. The depth tells that only a partial amount of the attribute set was used at any one path in the tree, at most 7 or the 37 for the lowest ten dataset, this weakens the theory that the accuracy was higher on the lowest ten dataset because of its amount of attributes compared to the highest ten dataset and accuracy. Another experiment could be run in the future that equalizes the amount of attributes in each sub dataset. From the high size for the ID3 on the highest dataset it can be inferred that the algorithm is over

specializing on that data, hence the low accuracy in contrast to the lower size on the lowest ten dataset. This specialization maybe due to the higher quality of the attributes thus allowing the algorithm to extract alot of information gain with little depth and high specialization.

Conclusion

WEKA provided a fast and efficient way to examine, visualize and create datasets. WEKA provided the needed classifiers for this experiment, but the ID3 Tree implementation lacked non-text tree visualization ability. WEKA was fast, user friendly and no bugs were encountered.

The results show that the protein dataset is very hard to classify, none of the tested classifiers achieved higher than 95% accuracy. The attribute selection based on PCNSA's classification test allowed for some interesting results, for it was seen that BayesNet achieved better accuracy on 27 attributes than the full set of 465 attributes. NiaveBayes was the only classifier to perform as expected. Both tree based classifiers performed better on the lower weighted attributes, these attributes were expected to result in low accuracy when given as a dataset. The suspected reasoning is that the lower weighted dataset allowed less information for over specialization. Overall BayesNet achieved the best accuracy when averaged across the three datasets.

References

[1] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. IBM Systems Journal (Special Issue on Deep Computing for the Life Sciences) 40(2), 426--441, 2001

[2] N. Vaswani, "A linear classifier for gaussian class conditional distributions with unequal covariance matrices," in *International Conference on Pattern Recognition*, 2002.

[3] Mario Koeppen, The Curse of Dimensionality, 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), held on the internet, September 4-18, 2000, WSC5 Homepage,

4 "Data Mining: Practical machine learning tools with Java implementations," by Ian H. Witten and Eibe Frank, Morgan Kaufmann, San Francisco, 2000.

VITA AUCTORIS

NAME:	Leon French
PLACE OF BIRTH:	Thunder Bay, Ontario
YEAR OF BIRTH:	1980
EDUCATION:	Tilbury District High School, Tilbury 1994-1999
	University of Windsor, Windsor, Ontario 1999-2003 B.Sc. Honours
	University of Windsor, Windsor, Ontario 2003-2005 M.Sc.