IMPROVING PROTEIN REMOTE HOMOLOGY DETECTION USING SUPERVISED AND SEMI-SUPERVISED SUPPORT VECTOR MACHINES

By

ANUJ R. SHAH

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
Department of Computer Science

May 2008

Copyright by ANUJ R. SHAH, 2008
All Rights Reserved

To the Faculty of the Washington State University:

The members of the Committee appointed to examine the dissertation of ANUJ R. SHAH find it satisfactory and recommend that it be accepted.

Chair

ACKNOWLEDGEMENTS

I think writing the acknowledgements section of any document is the most difficult part. You don't want to miss out on thanking each and every individual who directly or indirectly contributed to your thesis. Words cannot express the guidance, the motivation and the inspiration that each person has provided during the course of this thesis. Nevertheless I make a sincere attempt to thank everyone who played a part in the completion of this thesis.

First, I would like to thank both Dr. Bobbie-Jo Webb-Robertson and Dr. Christopher Oehmen. It is to them that I owe an awful lot. They both have played a major role and did more than just explain things to me. They always had ideas and hints to improve my work. Not only did they have the time to help but were also willing to brainstorm at short notice. They taught me that there was a right way of doing things and a wrong way of doing things, not failing to show me the consequences of each approach. It is them who introduced me to the field of machine learning and high performance data intensive computing. Next on the list is Dr. John Miller. It was his enthusiasm and consistent support that helped me in pursuing and completing this thesis. His jovial nature and approach to life has got me through many difficult situations.

My acknowledgement section can never be complete without the mention of my wife, Dr. Mudita Singhal, who had a major role to play in this thesis. It was her diligence, more than mine, which got me through this. "Behind every successful man there is a woman", a cliché one would say, but couldn't be truer than in my case. Lastly I acknowledge the support of my family members, my dad, my mom, my in-laws and my sister, who have constantly kept asking me about my completion date.

I would also like to acknowledge my funding source, the Data-intensive Computing for Complex Biological Systems project funded by the Office of Advanced Scientific Computing Research, and the Laboratory Directed Research and Development Program at the Pacific Northwest National Laboratory, (PNNL). PNNL is a multi-program national laboratory operated by Battelle for the U.S. DOE under contract DE-AC06-76RL01830. This research was performed in part using the Molecular Science Computing Facility (MSCF) in the William R. Wiley Environmental Molecular Science Laboratory (EMSL), a national scientific user facility sponsored by the U.S. DOE, OBER and located at PNNL.

IMPROVING REMOTE PROTEIN HOMOLOGY DETECTION USING SUPERVISED AND SEMI-SUPERVISED SUPPORT VECTOR MACHINES

Abstract

Anuj R. Shah, Ph. D.
Washington State University
May 2008

Chair: John H. Miller

The understanding of protein functions and there-by characterization is essential to modeling complex biological systems; for example, developing drugs against a pathogen requires understanding the proteins whose function is relevant to virulent activity. To a degree, protein annotation is helped by the fact that protein sequences are evolutionarily related or homologous. Knowing that a protein sequence is homologous to an already characterized sequence, the properties of the latter can be propagated to the former. The task of detection of protein sequences that are similar (homologous) to a given query protein is termed homology detection. Computational homology detection is one of the oldest tasks in bioinformatics. Conventional methods of homology detection rely on the basic principle of sequence similarity. However these methods fail to identify distantly-related homologs that have highly diverged protein sequences.

This thesis aims at developing machine learning based models of groups of protein sequences to improve the sensitivity of remote homolog detection. This research makes two significant contributions to the field of remote homology detection by the development of additional algorithms. The first contribution is the algorithm SVM-SimLOC, which utilizes sub-cellular localization information of protein sequences. The second algorithm, SVM-HUSTLE, is the first of its kind semi-supervised method using support vector machines for remote homology detection. Lastly, we propose a systems biology-based solution to practical homology detection using a publicly available tool called the Bioinformatics Resource Manager.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

To my dad, for his one-conditional love, “Get a higher degree”.

CHAPTER ONE

1 INTRODUCTION

High-throughput genome sequencing methods are advancing on a daily basis, producing overwhelming amounts of sequence data. Although sequence data lead to knowledge of proteins in the cell they provide little information about the biological processes involving these proteins. Species differences arise not because of the actual components themselves but because of how these individual components interact with each other to perform a common cellular function. This realization has led to rapid growth in the field of proteomics, which aims at elucidating the structure, interactions, and functions of all the proteins within a cell of an organism. The interaction and similarity between proteins is fundamental to a broad spectrum of biological functions, including regulation of biological functions, metabolic pathways, progression through the cell cycle, and protein synthesis [1]. Although a complete understanding of protein functionality will require information on many levels such as knowledge of transcription, translational regulation, post-translational regulation, binding constants, and structures, answering basic questions, such as what proteins have similar functions? How they interact with each other? provides a foundation on which more complex regulatory information can be built [2]. The understanding of protein functions and there-by characterization is essential to modeling complex biological systems; for example, developing drugs against a pathogen requires understanding the proteins whose function is relevant to virulent activity.

Annotation of sequence data with functional and structural characterizations is one of the major challenges in bioinformatics. Experimental annotation methods are not only expensive but also have failed to keep pace with the ever-growing sequence information. One common solution to functional annotation of novel sequences is via structural classification. The traditional methods of determining the 3-dimensional structure of protein sequences are through NMR [3] and X-ray crystallography [4]. These methods are time-consuming, costly and currently infeasible for some protein families. As a result more and more computational methods are being developed to reduce the time to annotate these sequences; practically, on a regular basis.

To a degree, protein annotation is helped by the fact that protein sequences are evolutionarily related or homologous. Homology is defined as the relationship of any two characters that have descended, usually with divergence, from a common ancestral character [5]. Additionally if two sequences are deemed homologous, it is highly likely for them to preserve a common function. Inferring this common ancestry between similar sequences, termed homology detection, provides a kind of boot-strapping method by which newly sequences genomes can be characterized based on existing annotations. Knowing that a protein sequence is homologous to an already characterized sequence, the properties of the latter can be propagated to the former. The task of detection of protein sequences that are similar (homologous) to a given query protein is termed homology detection.

1.1 Motivation

Computational homology detection is one of the oldest tasks in bioinformatics. Conventional methods of homology detection rely on the basic principle of sequence similarity. If two sequences have a high degree of sequence similarity, it can be determined with high statistical confidence that the two sequences share a common ancestor, and therefore may be related in function. Homolog identification can be conducted through computationally matching a query sequence to similar sequences in a database, using any of the pairwise sequence similarity algorithms, such as Needleman-Wunsh [6], Smith-Waterman [7], FASTA (a fast approximation to Smith-Waterman) [8] or BLAST (A basic local alignment search tool) [9]. These tools are popularly used to produce a ranked list of protein sequences accompanied by a statistically calculated P-value or E-value that match a given query sequence that tests the hypothesis that the sequences are similar to each other. These algorithms are highly successful when the sequences have high sequence identity. However, the sensitivity of these algorithms deteriorates in cases of evolutionarily related sequences that diverge a fair-bit; and have sequence identity of less than 20% [10]. Also the residue matching process is not trivial as these proteins could have separated a long time ago in the evolutionary history. Thus homology detection purely based on sequence similarity may not result in the detection of distantly related proteins. The detection of distantly-related homologs with highly diverged protein sequences is termed as remote homology detection.

Discriminative machine-learning and support vector machines (SVM) based approaches have demonstrated increased sensitivity while detecting remote homologs [11-24]. Though limited in their application, these methods provide a foundation for further research in the field of remote homology detection. There is a need to develop novel methods for remote homology that enable the accurate detection of distantly related sequences and maintain a balance between required computation, the sensitivity of the approach and the time to solution.

The discriminative approaches to homology detection offer the capability to integrate additional information. For example, the inclusion of additional knowledge of biological properties such as hydrophobicity, molecular weight, structural orientations, sub-cellular localizations etc, may assist in the detection of remote homologs. However, there are few proteins with computed structural orientations and annotated localizations. Additionally, the computation of these properties may require large amounts of CPU cycles (in cases of structural calculations, etc) and may not be worth the additional effort. Apart from computational cost for calculations of additional properties, each of the discriminative methods has an inherent drawback as they rely heavily on a family-based training and classification of protein sequences. Proteins that do not fall into one of the selected families cannot be classified accurately. More importantly, new families of proteins can never be discovered with these approaches. This short-coming has limited the use of these approaches when detecting protein homologs.

1.2 Objectives and Contributions

The primary objective of this research is the development of new biosequence analysis methodologies that overcome the limitations of existing approaches and provide a more sensitive detection method that is computationally feasible and provides results in a competitive time-to-solution. This research aims to make two major contributions to the field of remote homology detection. The first is the development of family-based machine learning approach SVM-SimLoc that incorporates additional sub-cellular localization knowledge for individual protein sequences to reduce the number of false positives while improving the accuracy of detection. The second is the development of a new algorithm based on the principles of supervised and semi-supervised machine learning, SVM-HUSTLE.

In addition to the development of these algorithms, we discuss the design of an integrated system based on a systems biology based software solution called the Bioinformatics Resource Manager (BRM) [25] that presents the user with a highly intuitive interface to launch homology searches on custom databases.

1.2.1 SVM-SimLoc

SVM-SimLoc is the first algorithm developed that utilizes the predicted sub-cellular localization of a protein sequence in a SVM framework. The name for the algorithm signifies the integration of sequence similarity with sub-cellular localization feature vectors. The premise behind the use of sub-cellular localization is the elimination of a large number of false-positive relationships between protein functions that cannot arise due to location characteristics. The SVM-SimLoc algorithm uses data integration at the feature level to integrate a measure of sequence similarity with localization vectors. The sequence similarity is represented by Smith-Waterman E-scores between sequences while the location vectors are derived from the CELLO [26] localization prediction tool. This algorithm follows many other discriminative machine learning based algorithms in terms of a family-based setting. We report a statistically significant improvement of prediction accuracies using SVM-SimLOC on a previously published benchmarking dataset. We discuss the details of the SVM-SimLoc algorithm and compare its performance to other state of the art algorithms in Chapter 4. It is needless to say that a universal approach that can be easily applied across all sequences is required for remote homology detection.

1.2.2 SVM-HUSTLE

The **SVM**-based tool for **H**omology detection **U**sing i**T**erative **LE**arning (SVM-HUSTLE) is the first of its kind SVM-based algorithm for “pairwise” homology detection. SVM-HUSTLE combines the principles of supervised and semi-supervised machine learning to determine if two sequences are homologous to each other. Traditional SVM-based methodologies for remote homology detection implicitly rely on a family-based classification of protein sequences. Classifiers are built for a distinct set of families and sequences are classified as either a “member of” or “not a member of” a particular family. SVM-HUSTLE on the other hand deviates from the family-based SVM-based methods. Classifiers are built on-the-fly for a given query sequence using semi-supervised learning. SVM-HUSTLE outputs a list of protein sequences that are deemed homologous to the given query sequence based on a pairwise comparison. Our benchmarking results indicate that SVM-HUSTLE is comparable in performance and sensitivity to profile-profile based methods such as COMPASS [27], PROF_SIM [28] and HHSearch [29]. This is a very significant result, especially since profile-based methods require the computationally expensive generation of profiles built using multiple alignments. SVM-HUSTLE also significantly outperforms most existing state of the art remote homology detection methods on a stringent benchmarking dataset. SVM-HUSTLE is applicable in real world scenarios where there is no preconceived notion of families for a novel set of proteins. We discuss the algorithm in detail along with the configuration parameters in Chapter 5. An evaluation of the algorithm as compared to other existing methods is also provided.

1.2.3 Implementing a homology workflow for practical use

Using the BRM software and the high-performance data management capabilities of ScalaBLAST [30], we present the design of a workflow that allows researchers to search for homologous proteins within a database of their choice. Traditional homology-based websites provide a fixed set of databases that can be searched. This is largely because each database requires huge amounts of processing for e.g. computing of multiple alignments in cases of profile-profile based comparison algorithms. We present a practical user interface based solution on the front-end, coupled with high-performance clusters at the back-end, which can be asynchronously implemented as an additional workflow within the BRM toolkit.

CHAPTER TWO

2 BACKGROUND: MACHINE LEARNING METHODS

2.1 Introduction

Machine Learning originates from the field of Artificial Intelligence. The focus of machine learning research is "How to make machines capable of learning what humans can?" This could be learning a set of rules for classification, or identification of a set of patterns for character recognition or the categorization of a set of data points into distinct clusters. In other words, machine learning can be broadly considered as the programming of computers to either gain knowledge from data or the derivation of statistical models to make predictions in the future. Machine learning uses the theory of statistics and computer science at its core. The statistics help in building the models, as the core task is the inference from sample data and the computer sciences are used to solve optimization problems as well as process and store the massive amounts of data that are operated on. Additionally, the efficiency of learning algorithms, namely, their space and time complexity, is an important measure to assess the performance of these algorithms and/or models. Machine learning has demonstrated application in different fields such as medicine, telecommunications, networking, physics biology etc. The field of machine learning research can be broadly classified into three distinct disciplines, unsupervised learning, supervised learning and semi-supervised learning.

2.2 Unsupervised Learning

Unsupervised learning is a class of machine learning where the goal is to have the computer learn how to do something without us being told how to do it. Such a technique is commonly applied to a collection of samples that do not have any categorization associated with it. There are a number of reasons for interest in unsupervised learning. In the early stages of investigation, it may be a valuable exercise to reduce the amount of data to manageable chunks by using some sort of a clustering approach. The discovery of distinct subclasses or groups within a dataset may suggest interesting possibilities. In other cases, collecting and labelling a large set of sample patterns can be surprisingly expensive. However, if a classifier can be crudely designed based on the alphabet space of the problem, the labelling can be achieved in a significantly short time span. Unsupervised learning is also used to extract a “set of features” or pre-process data in certain cases.

There are a couple of approaches to unsupervised learning. The first is to teach the computer not by giving explicit classifications, but by using some sort of a reward system to indicate success. The other form is reinforcement learning, where the agent bases its actions on the previous rewards and punishments without necessarily learning any information about the exact ways its actions affect the world. This can essentially be equated to learning by trial and error and can be very time consuming. Unsupervised learning is based on the similarities and differences among the input patterns. It does not result directly in differences in overall behaviour because its outputs are almost internal representations. This type of learning plays a role in perceptual systems. Visual input is an example of such learning. Another form of unsupervised learning is the clustering of data into distinct categories. The term “clustering” and “unsupervised learning” are almost used synonymously in a number of settings. In the following sections we discuss some forms of unsupervised learning that are prevalent in literature and more popular than others.

2.2.1 Exploratory Data Analysis

When faced with data of high-dimensionality, researchers are generally looking for subsets of data that might be “interesting”, i.e. looking for anomalies or deviations from the normal behaviour within the group. The traditional approach to examine these high-dimensional data sets in an exploratory fashion is to reduce their dimensionality in the first place. This is achieved by either linear or non-linear mappings or projection strategies [31].

The idea of projection pursuit is to locate the projection or projections from high-to low dimensional space that reveal the most details about the structure of the data. Once a set of interesting projections has been found, the researcher can further investigate the data analyzing the interesting projections separately. The simplest form of a projection pursuit technique is a scatterplot: a two-dimensional display to indicate data characteristics across two selected dimensions. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis [32]. The main limitation of such a technique is the amount of time it takes to exhaustively explore a given data set. The higher the dimensions, a large number of scatterplots are to be generated.

Another common technique for finding patterns in high-dimensional data is the Principal Component Analysis (PCA) [33]. It has wide-ranging applications from compression [34], face recognition [35] to detection of epithelial cells in ovarian cancer patients [36]. Since patterns can be hard to find in high-dimensional data, PCA, an unsupervised learning technique, serves as a powerful tool for analysis. PCA is a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated orthogonal (mutually perpendicular) variables, which are called the principal components for that dataset. These components account for the maximum variability in the data in decreasing order, i.e. the first component accounts for as much of the variability in the data as possible and each following component accounts for the remaining. The PCA is a technique that seeks a projection that best represents the data with minimal loss of information. The PCA is most often used as an exploratory first step to identify if there are any distinct patterns in the dataset of interest.

2.2.2 Clustering

One of the other most commonly applied techniques on a large dataset is that of data clustering. Data clustering is a technique by which we make groups of objects that are somehow related or similar in characteristics. Clustering is often confused with classification. In clustering we are not pre-defining the classes and the number of classes is generally determined on-the-fly based on the patterns or characteristics of the data, while a classification technique relies more on the knowledge of a particular class and its properties before classification. In a classification technique, one decides whether a particular entity is a member of a particular class, while in a clustering technique, one determines what group of entities belong together based on their properties. A number of clustering techniques have been published in literature [37] . The most popular are the K-means clustering [38] and the hierarchical clustering [39].

The k-means algorithm is an algorithm that is used to cluster n items/objects into k clusters. The objective of this algorithm is to minimize the intra-cluster variance. In terms of performance, the algorithm is not guaranteed to return an optimal solution. As a result, a common practice is to run the algorithm multiple times and return the best clustering found.

Another drawback of this algorithm is that the number of clusters, k, is an input to the algorithm. An inappropriate choice of the number of clusters may yield poor results.

In contrast, the hierarchical clustering algorithm, tries to find an optimal clustering solution based on an $N * N$ distance matrix for a set of N items. A critical component of the hierarchical clustering algorithm is the distance measure. Common distance functions include the Euclidean distance, the Hamming distance, the dot product, etc. Based on the distance function, the hierarchical clustering algorithm either builds or breaks up a hierarchy of clusters. The traditional representation of such a clustering is a tree, with individual elements at one end and a cluster containing every other element at the other end. The algorithm progresses slower than heuristic k-means algorithms and is generally applied in cases where there is an expected hierarchy among the objects, for e.g. in categorical data.

2.3 Supervised Learning

Supervised learning is one extreme in the spectrum of machine learning algorithms. For every input in a supervised learning algorithm, there is an output. The output or target associated with each input is the desired response of the system. The learning algorithm adjusts its internal memory in such a way that it is more likely to produce the desired response given a defined set of inputs. Supervised learning is fairly common in classification problems because the goal is often to get the computer to learn a system that is already created. Examples of such systems include character and digit recognition systems. In such systems, the output is a binary yes/no answer. Another variant of supervised learning is called reinforcement learning. In reinforcement learning, there is no binary desired category but the only feedback is that the tentative category is right or wrong. To further illustrate the point, consider a game of chess, where the initial rules are very easy. However it is the combination of certain moves or strategies that make an effective game strategy. So the individual steps are of little importance, however when combined in a series or a strategy make either a winning or losing proposition. Reinforcement learning is generally applied in such game playing cases. In the following sections, we provide a brief introduction to some of the most popular techniques mentioned under supervised learning.

2.3.1 Artificial Neural Networks

An artificial neural network (ANN), more commonly referred to as a neural network, is a mathematical or computational model that is inspired by the way biological nervous systems, such as the human brain, process information. The key element of this system is the large number of interconnected nodes that represent the neurons in the brain. These neurons work together, learning by example, to solve a problem. ANNs can be used both in an unsupervised and a supervised setting. They have a remarkable ability to derive meaning from complicated or imprecise data and can be used to extract patterns and detect trends that are too complex for the human brain to process.

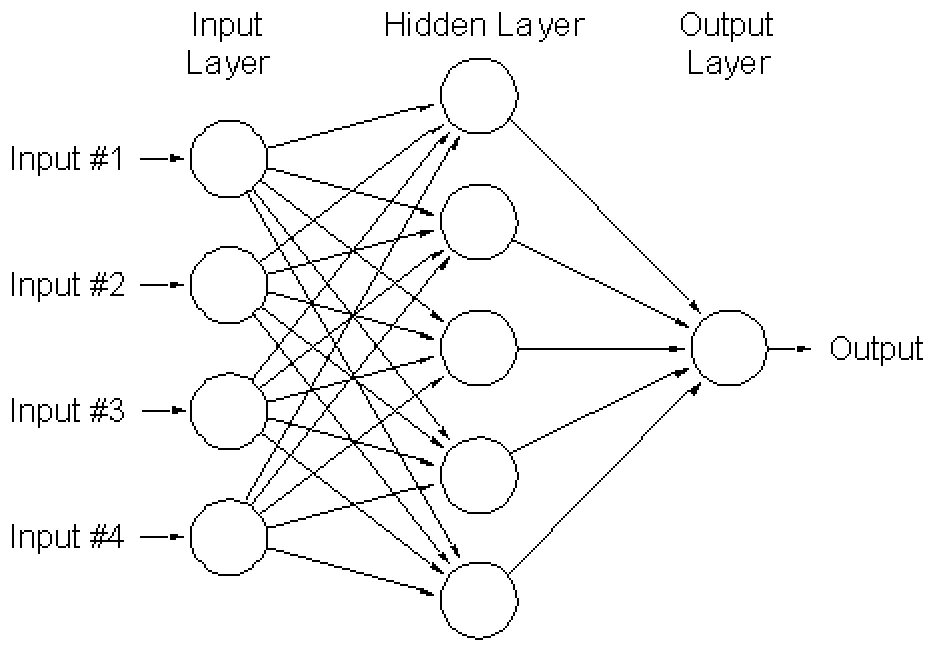


Figure 1 - Artificial Neural Network

The above figure, Figure 1, shows a type of neural network called a feed-forward neural network. In this type of network the processing units, or artificial neurons, or nodes are arranged in a layered configuration, containing one input layer, one or more hidden layers and one output layer. The input layer and the output layer can each have a number of nodes. Each node or neuron within the hidden layer processes all the inputs to that particular node in a weighted manner. This weighted sum is compared to a threshold and the activation signal is passed on to the next node within the layers. Optimization routines can be used to determine the ideal number of hidden layers as well as the number of nodes at each layer for an ANN. ANN's has been applied to a broad spectrum of data-intensive applications ranging from computational biology and bioinformatics [40, 41] to finance [42, 43], and pattern recognition [44].

2.3.2 Genetic Algorithms

Genetic Algorithms (GA) is a class of machine learning algorithms that can be used as a search technique to find the best-estimate solutions in optimization problems. GA's model the principles of evolutionary biology such as inheritance, mutation, recombination and natural selection. GA's are typically implemented as computer simulations in which a starting population of abstract representations evolves over time towards a better solution. The abstract representations, also referred to as chromosomes, are either string of 0/1 or can have more intricate encodings. The evolution starts from a random population and changes occur through a selection process over generations. In each generation, the fitness of the whole population is evaluated, and the most successful individuals are kept for the next generation. This selected group of individuals is supplemented with offspring that are obtained by processes of random mutation, recombination and/or crossovers. The fittest set from these parents and offspring becomes the starting population for the next generation or iteration of the algorithm.

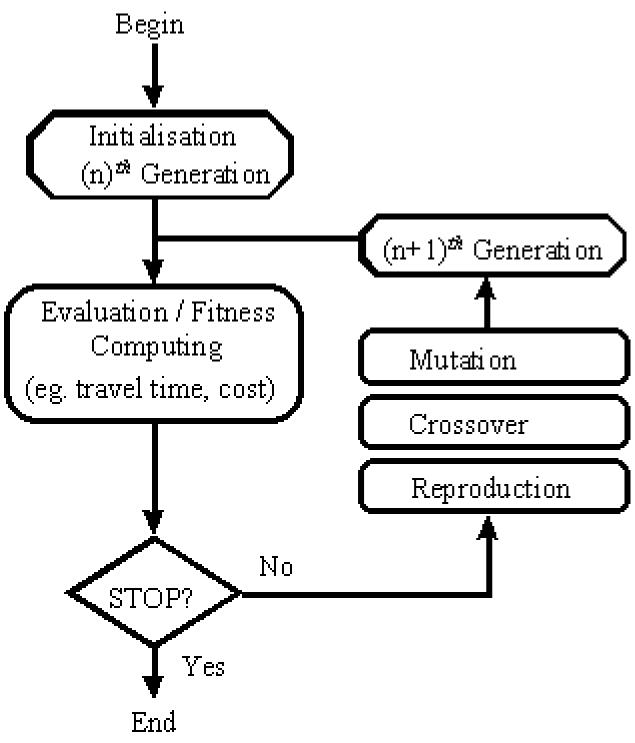


Figure 2 - Genetic Algorithm Flowchart

GA's are mainly applied to search and optimization problems [45]. However, there have been instances of their use in classification problems [46, 47]. In protein homology detection, genetic algorithms have been used mostly to either identify the most suitable parameters for an existing model [48] or refine existing models of protein homology [49]. Figure 2 shows the flowchart for a general Genetic Algorithm implementation.

2.3.3 Support Vector Machines

Vapnik and Chervonenkis developed the theory behind the SVM in the late nineties [50, 51]. Since then there's been a surge in its practical use. SVMs are machine learning algorithms designed with the intention of "generalizing better". The problem SVM algorithms address relates to the efficient learning of a classification rule from a set of exemplars. The goal is the classification of each candidate as being either 'a member' or 'not a member' of a given class. The fundamental difference between a GA and the other techniques is that a GA is more of an optimization approach for a pre-defined function, whereas for an SVM and ANN the function is defined based on the kernel (or topology of the network).

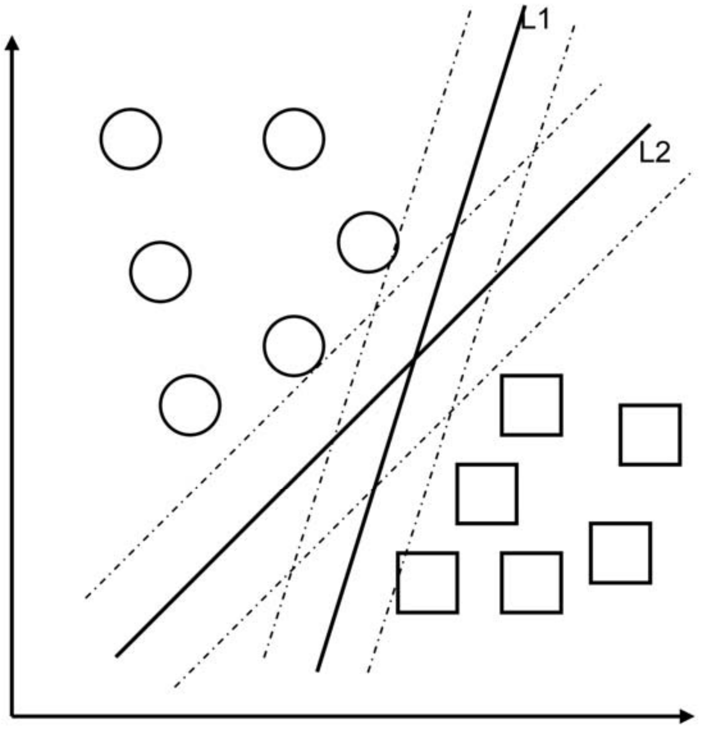


Figure 3 - Separating hyperplane example

To explain the operation of a support vector machine, consider Figure 3 where we represent two-class data, the circles and the squares. The objective of the classification task is to divide the two classes such that there is maximum separation between the two-class, implying that the distance of any element from the classifying plane is the most. There are many classification planes that can achieve this, however only one plane will ensure maximum separation (margin). By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. The goal of the SVM framework is to find this hyperplane in n-dimensional feature space, where n represents the length of the feature vector for the element.

A linear SVM has a decision function of the form

$$f(x) = wx + b \qquad (1)$$

where w - the weight vector

and b is a constant bias.

A data point is classified according to the sign of f. The choice of w and b is such that the separation between the positive and negative examples is the maximum for linearly separable data points.

To extend this description to n-dimensions, the SVM requires that the input data be represented as a collection of fixed-length vectors. These vectors are then transformed into a feature space of higher dimension. The objective of the algorithm is then to find an optimal hyper-plane (w, b) that separates the data points of the two classes as far as possible within this feature space (maximize margin γ). Such a hyper-plane is called the maximum margin hyper-plane. Statistical learning theory suggests that, for some classes of well-behaved data, the choice of the maximum margin hyper-plane will lead to maximal generalization when predicting the classification of previously unseen examples [51]. The internal transformation of the input fixed-length data vectors into a non-linear high-dimensional feature space can be accomplished by means of a kernel function. Any symmetric, positive semi-definite function is a valid kernel function, corresponding to an inner product in some feature space. The base kernel in an SVM is generally normalized forcing each vector to have a length of 1 in the feature space i.e.

$$K(X,Y) = \frac{X \cdot Y}{\sqrt{(X \cdot X)(Y \cdot Y)}} \tag{2}$$

Some of the most common kernel functions existing in literature are the linear, polynomial and radial basis function.

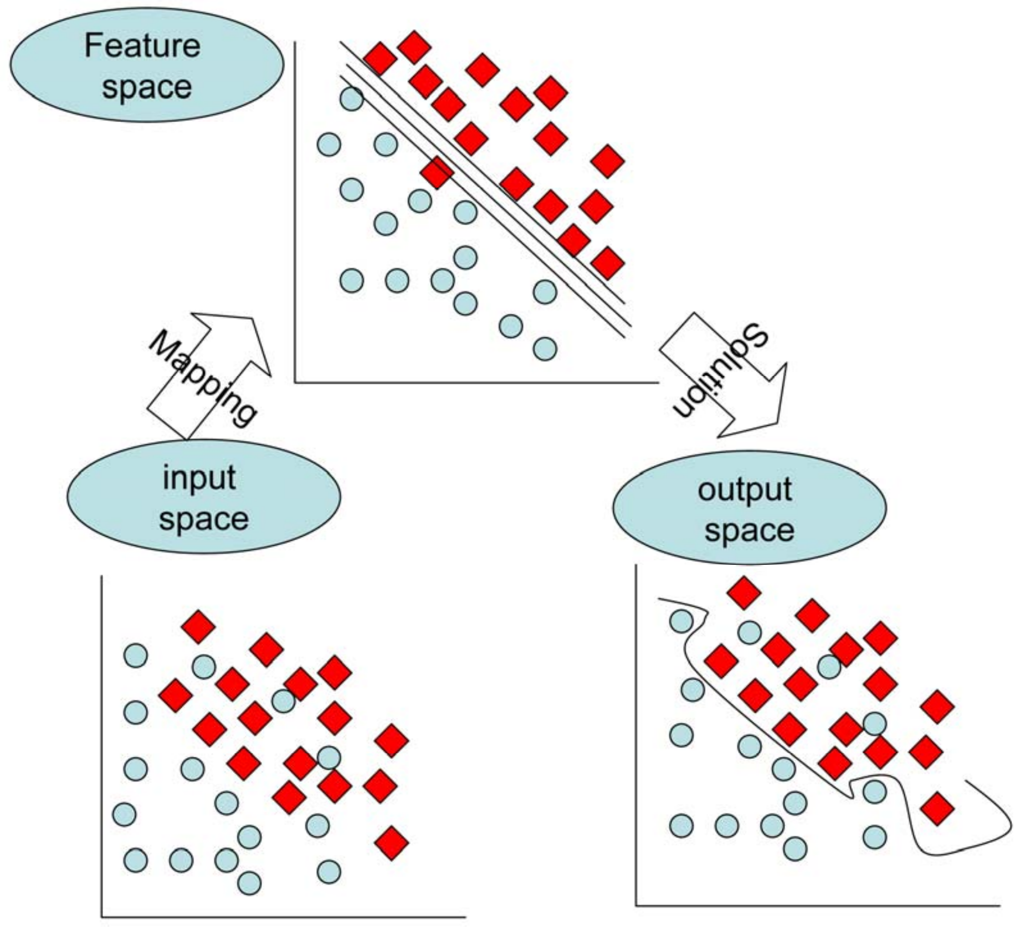


Figure 4 - SVM Workflow

In the parlance of SVM literature, a predictor variable is called an *attribute*, and a transformed attribute that is used to define the hyperplane is called a *feature*. The task of choosing the most suitable representation is known as *feature selection*. A set of features that describes one case (i.e., a row of predictor values) is called a *vector*. So the goal of SVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyperplane are the *support vectors*. Figure 4 presents an overview of the SVM process. The input space contains two-class data with an n-dimensional vector representing each element. The SVM transforms this vector into a feature space using a kernel function and finds a hyperplane in this high-dimensional space. The hyperplane and support vectors are now used in the output space to determine the correct classification for incoming data.

2.4 Semi-Supervised Learning

In supervised learning, the SVM 'learns' classification rules from both positive and negative exemplars. In unsupervised learning, patterns are formed based on clustering techniques. Semi-supervised learning is a training strategy that combines information from labeled data (i.e. data that is known to fall into one category or the other) and the unlabeled data which often clusters around labeled data. It is often expensive, time-consuming or difficult to collect labeled data, while unlabeled data maybe easier to collect in most cases. However using this unlabeled data in meaningful ways is often difficult. Semi-supervised learning is the class of algorithms that address this issue and help build better classifiers. It is important to note that using large amounts of unlabeled data is not a trivial task and a number of assumptions need to be made about the unlabeled data points. The assumptions help constrain the unlabeled data to the model being developed, resulting in a better classification model.

Semi-supervised functions are based on so-called “cluster” assumption: classification rules remain unchanged in areas of dense exemplars. Using clustering to associate unlabeled data with the labeled training data often improves coverage of the vector space improving the accuracy of classification. Figure 5 illustrates the potential difference between learning using a semi-supervised technique as opposed to a supervised technique.

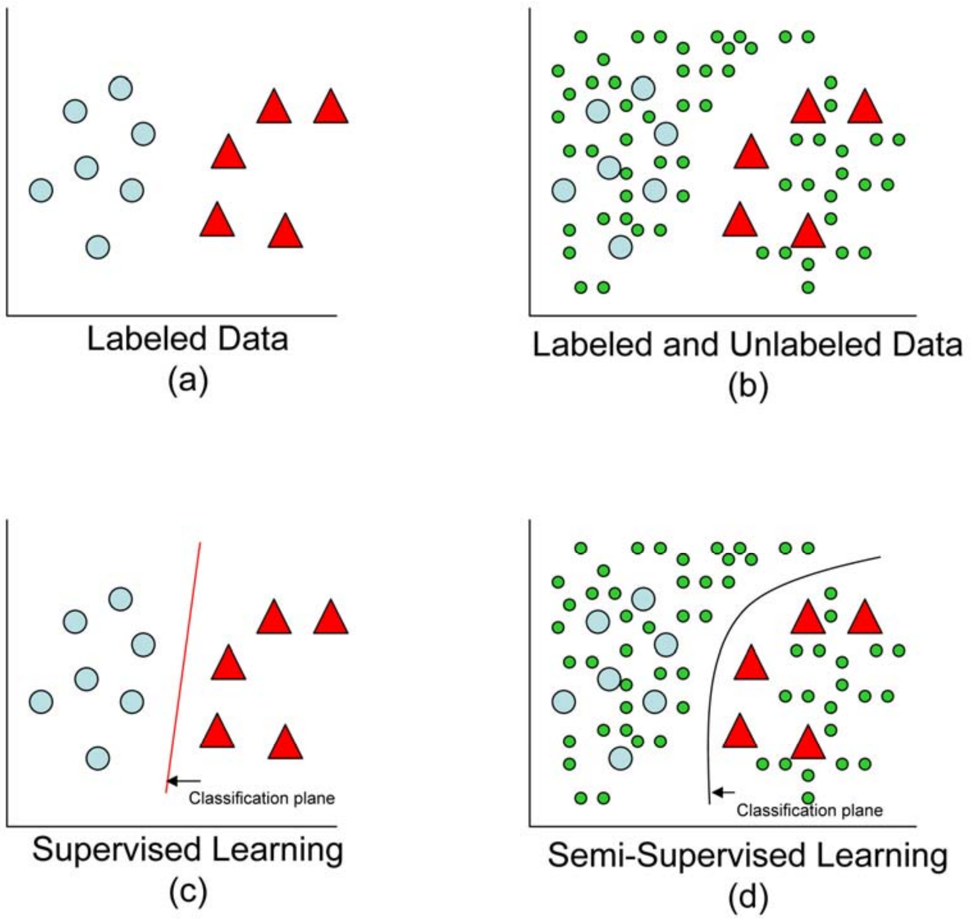


Figure 5 - Semi-Supervised Learning

The supervised algorithm may develop a classification rule that could cause incorrect classification of incoming exemplars when labeled data sparsely covers the vector space.

Adding unlabeled data refines the detail of the separating hyperplane placing the decision boundary in areas of low density generating a more accurate classifier.

Figure 5 illustrates the typical behavior expected of a semi-supervised learning methodology, however like with every rule there is an exception in these cases as well. The Baum-Welch algorithm is used in bioinformatics to find the optimal parameters for a hidden Markov model chain [52]. This algorithm classifies as a semi-supervised technique. It is noted that when a hidden Markov model is trained with unlabeled data, under certain initial conditions, the accuracy is reduced [53]. Similar arguments are made in [54]. Thus one needs to be cautious while employing a semi-supervised technique on large amounts of unlabeled data.

One of the most popular semi-supervised techniques is the Expectation Maximization algorithm, more commonly known as the EM algorithm [55]. An EM algorithm is used for finding the maximum likelihood estimates of parameters in probabilistic models. The model here depends on unobserved latent variables. An EM algorithm constitutes two steps, the first is an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and the second is a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the first step. These parameters are now used in the calculation of the next E step and the process continues until a terminating criterion is achieved.

Semi-supervised learning is generally applied when unlabeled data is abundant while labeled data is small e.g. web page classification. For a detailed literature survey of semi-supervised learning interested readers can refer to [56].

CHAPTER THREE

3 BACKGROUND: COMPUTATIONAL HOMOLOGY DETECTION

Computational homology detection is the cornerstone of bioinformatics. During the past 30 years, numerous efforts have been focused on computationally predicting the functions of novel proteins. A well-established practice is to compare a query sequence of unknown function with a well-characterized database of sequences. If two similar sequences are derived from a common ancestor, they often preserve a related function. Inferring this common ancestry between similar sequences, termed homology detection, provides a kind of boot-strapping method by which newly sequenced genomes can be characterized based on existing annotations.

The field of computational homology detection can be easily divided into four distinct phases: first, there were pairwise sequence similarity based algorithms; second, the sequence to profile comparison algorithms; the next phase saw the introduction of profile-profile comparison methods while alongside this phase was the development of discriminative machine learning based homology detection methods. Each phase presents novel findings and sets the stage for the next phase of detection algorithms. The following sections discuss the details of some of the most prominent algorithms and methods in the field of homology detection.

3.1 Pairwise Sequence Similarity

One method of quantifying the degree of homology is finding an optimal alignment between two sequences. The Needleman-Wunsch algorithm [6] was the first sequence similarity based algorithm to grace the field of bioinformatics. This is a dynamic programming based algorithm that finds the optimal global alignment of two sequences by maximizing the number of amino acid matches and minimizing the number of gaps necessary to align the two sequences. One of the major shortcomings of this algorithm is the time complexity ($O(M \otimes N)$) where M and N are the lengths of the two protein sequences. Also in practice, many proteins can be thought of as comprising a series of functional regions joined by natively disordered links [57]. Mutations in the functional regions would tend to degrade the behavior of a protein, whereas mutations in disordered regions may not impact the function at all. As a result, one would expect that homologous proteins can be more rapidly detected by aligning these smaller pieces or “local” regions.

Alignment methods such as the Smith-Waterman algorithm [7], the fast approximation to Smith-Waterman (FASTA) [8] and Basic Local Alignment Search Tool (BLAST) [9] take advantage of local sequence similarity in proteins. The Smith-Waterman is very similar to the Needleman-Wunsch algorithm, the only difference being that it’s a local alignment as opposed to the global alignment of Needleman-Wunsch. Instead of aligning the entire length of two proteins, this algorithm finds the region of highest similarity between the two sequences. The time complexity of this algorithm is $O(M \otimes N)$. In order to improve the time complexity of the Smith-Waterman algorithm, Lipman et al [58] devised the Fast Approximation to Smith-Waterman (FASTA) algorithm. Unlike the Needleman-Wunsch and Smith-Waterman algorithms, FASTA approximates the optimal alignment by searching and matching k-tuples, or subsequences of length k. The algorithm assumes that related proteins will have regions of identity, and by searching with k-tuples, the FASTA algorithm allows small regions of local identity to be found quickly. The FASTA algorithm is substantially faster than the Needleman-Wunsch or Smith-Waterman alignments and thus can be more easily used in database queries. The time complexity of the FASTA algorithm depends on the size of the k-tuple and is given as $O((M \otimes N) \div 20^k)$. Although the FASTA algorithm is faster than any of the previous algorithms discussed, it is not guaranteed to find the optimal alignment between two proteins. Because it uses k-tuples to speed up searches, it can miss smaller areas of similarity and thus there is a possibility of misaligning two sub sequences. Thus, it is clear that there is a tradeoff between speed and sensitivity.

To overcome the shortcomings of the above approaches in terms of speed of operation, Altschul et al [9] devised the Basic Local Alignment Search Tool, more popularly known as BLAST. Similar to FASTA, BLAST is a heuristic pairwise sequence similarity algorithm, however the basis of the BLAST algorithm is the use of words and high-scoring segment pairs instead of tuples. The BLAST algorithm begins by finding all words, or subsequences of length w (typically 3), which are present in the sequence. Using this list of subsequences it performs a database search for words in this neighborhood. The time complexity of BLAST is essentially the same as any of the similarity based algorithm, $O(M \otimes N)$, however the speed up is achieved due to the use of a hash table to store the high scoring segments and the smaller size of these segments. Also BLAST significantly lowers the number of segments which need to be extended using the statistically significant elimination of words. To date, BLAST is the most popular algorithm used for searching for a list of homologs against a large database.

One of the major shortcomings of a sequence similarity based algorithm is that these methods often fail to detect common ancestry when proteins share low residue similarity, i.e. when proteins are remote homologs, which occurs commonly [10]. This shortcoming of sequence similarity based methods gave rise of the next phase of algorithms for remote homology detection.

3.2 Sequence - Profile comparison algorithms

The operation of a pairwise sequence alignment algorithm is essentially a one-step process. It takes as input two protein sequences and computes a score representing how closely related the two sequences are. The concept of protein families was introduced into the problem of remote homology detection as an approach to overcome the lack of sensitivity that arises with algorithms comparing sequences based on residue similarity. A protein family is a group of sequences that share a common evolutionary origin. A protein sequence profile (statistical/computational model) can be built for each of these families or super-families. A sequence profile is generated by aligning a group of closely related protein sequences, illustrating the probability of occurrence for each amino acid at each position of the multiple sequence alignments. A sequence profile better reflects a collection or family of proteins than a single sequence. A protein of unknown function can then be compared to each of the protein families producing a measure of likelihood of the sequence originating from this family. These family-based methods help computational biologists identify nearly three times the number of homology relationships as identified by simple pairwise alignments [59]. PSI-BLAST [60] and hidden Markov models (HMM) [61] are two most popular examples of such generative approaches.

PSI-BLAST builds an alignment-based statistical model from a set of proteins by defining a position specific scoring matrix (PSSM). It then adopts a semi-supervised algorithm to use this query-specific PSSM model to search the space to identify additional homologous sequences iteratively from the database, refining the model at each step. One important aspect of a PSSM is that it looks for ungapped alignments to a consensus. The challenge for sequence-profile based alignment resides in the selection of "closely" related protein sequences in the database for profile construction.

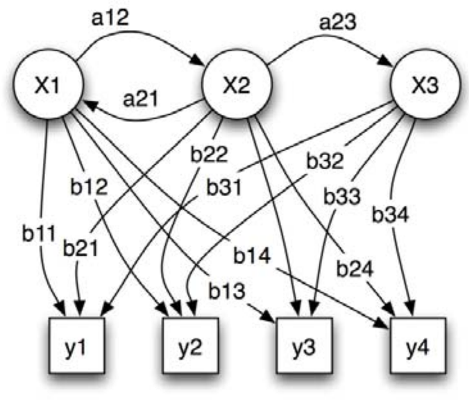


Figure 6 - Hidden Markov model example. In the above figure x — states, y — possible observations, a — state transition probabilities, b — output probabilities.

Hidden Markov Model (HMM) is a technique based on the probability distribution for a finite set of states. Each state has an associated transition or emission probability. The output is the joint probability for each state. Figure 6 gives an example of the states and the transition probabilities. The HMMs come in a number of varieties, or orders. HMM can be first order, second order and so on. The order defines the number of previous states that influence the current state, hence in a third order HMM, a state depends on the previous three states. Unlike multiple sequence alignments, an HMM based technique constructs a profile based on HMM, that contain a probability matrix for transitions and an emission matrix for amino acid changes. HMMs also allow an insert and delete state and so account for gapped alignments. Since HMM's construct an HMM based profile, the construction process is at least 10 times slower than PSI-BLAST [28]. However, the HMM based approaches offer a more sensitive remote homology detection method. Examples of such an approach are HMMER [62], The Sequence Alignment and Modeling (SAM) system [63], PFTools [64] and meta-MEME [65].

In the benchmarking experiments, the computational performance of SAM was observed as being slower than the HMMER however it performed better on measures of accuracy and sensitivity [63].

3.3 Profile-profile comparison

The comparison of profiles of a family of proteins against a database still fails to detect a large number of distant/weak homologs. Methods such as PROF_SIM [66], COMPASS [27] and HHSearch [29] take the approach of PSI-BLAST one step further. Instead of comparing a profile against individual sequences within a database, these methods compare two profiles in a pairwise setting, yielding significantly more homologs than PSI-BLAST, HMM or profile-based methods [29]. Both PROF_SIM and COMPASS allow for gaps and use the Smith-Waterman local alignment algorithm. PROF_SIM employs a column score based on Jensen-Shannon entropy. The statistical significance of the alignment is reported as a P-value that is calculated directly from the raw score. COMPASS, on the other hand, uses a column score based on the relative entropy between two amino acid distributions. It estimates the E-value analytically by generalizing the approach of PSI-BLAST to the profile-profile case. These methods are amongst the most sensitive remote homology detection methods.

To give a comparative scale of sensitivity performance, on a benchmark dataset with less than 20% sequence identity between any two pair of sequences, BLAST correctly identifies 2.2% of the total number of homologs. The corresponding numbers for PSI-BLAST and HMMER are 17.7% and 18.7% respectively while PROF_SIM and COMPASS find 24.9% and 34% of the total homologs respectively [29]. The most recently conceived method of HMM-HMM comparison by Soeding [29] is shown to outperform each of the above methods. Their method, HHSearch, is based on the generalization of log odd scores maximized in HMM-sequence alignment to the case of HMM-HMM alignment. The alignment of HMMs instead of simple sequence profiles improves both the sensitivity and the alignment quality significantly. However one major short-coming of these profile-profile comparison methods is the computationally expensive multiple alignments that needs to be calculated in order to generate profiles for a set of sequences. As the size of the target/search database grows it becomes increasingly difficult to generate multiple alignments for all the sequences. Alternative methods need to be developed that eliminate the need to align the database sequences.

3.4 Machine learning based discriminative methods

Along side the development of profile-based methods originated the application of machine learning methods to the problem of remote homology detection. The detection of homologs was now modeled as a binary classification problem, distinguishing the homologs from the non-homologs. The first successful application of a discriminative approach using support vector machines was demonstrated by Jaakkola et al [13]. Their technique, called SVM-Fisher, combines a profile HMM with the SVM for remote homology detection. In SVM-Fisher, sequences are represented as a fixed-length vector of Fisher scores extracted from profile HMM for a protein family. The SVM-Fisher approach is a combination of a generative HMM profile and a discriminative SVM. In empirical tests, SVM-Fisher significantly outperformed the other non-discriminative approaches [67]. To date, the most popular discriminative method for remote homology detection has been SVMs, spurring a series of efforts focused on devising more robust and biologically relevant representations of protein sequences as feature vectors. Building on the SVM-Fisher technique, Liao et al [19] used pairwise sequence similarity scores obtained from the Smith-Waterman local alignment algorithm, in a support vector machine framework. Their method, SVM-Pairwise [19] is one of the most sensitive discriminative methods. SVM-BALSA [21] advances the SVM-Pairwise methodology one step further by using similarity scores from a Bayesian Algorithm for Local Sequence Alignment (BALSA). The formulation of the problem as a Bayesian inference problem allows the data to be treated as random variables. By incorporating a series of scoring matrices, gap parameters, and all possible alignments in their forward algorithms, inferences can be made on these variables. SVM-BALSA is shown to be more sensitive than SVM-Pairwise [21]. Leslie et al devised the spectrum kernel [68] and the mismatch kernel [17]. The spectrum kernel is a very simple kernel, efficient to compute and does not depend on any generative model. A k-spectrum of an input sequence is defined as the set of all k-length subsequences that it contains. A kernel is calculated for this representation of a protein sequence. The efficiency in the calculation of subsequences is derived from the use of suffix trees. Suffix trees have been extensively used in string matching algorithms in the field of computer science. In benchmarking experiments, the spectrum kernel is observed to perform as well as the SVM-Fisher and SVM-Pairwise methodologies [68]. A number of other similar k-subsequence string matching kernels are devised by Leslie et al [16, 17] that have subtle variances. The mismatch kernels [16] are based on a trie data structure as opposed to a suffix tree structure for the spectrum kernel [68].

Each of the above described SVM-based methods is sequence based. They represent the nature of protein similarity, based on the amino acid composition in a machine learning framework. An alternative way of detecting protein similarity is via protein motifs. Motifs are usually constructed from multiple sequence alignments of related proteins. As a preliminary step, one extracts from an alignment 'blocks' which are ungapped regions of high sequence similarity. The premise behind a motif based detection of sequence similarity is that motifs for most catalytic sites and binding sites are conserved over much wider taxonomic distances and evolutionary time points than the sequences themselves. A number of protein motif databases exist today that represent the motif content of a given sequence. PROSITE was the first protein motif database. Others include BLOCKS, PRINTS, PFAM, ProDom, DOMO and InterPro. Ben-Hur devised the first of its kind sequence similarity measure based on the motif content of a protein sequence [11]. The motif-based kernel was shown to outperform kernels based on sequence similarity algorithms such as BLAST and Smith-Waterman. Also the performance of the kernel is significantly improved when coupled to an SVM-classifier rather than a nearest neighbor classifier [11]. Another aspect of the motif kernel method was the use of a sparse vector representing the occurrence of eMOTIF [69] patterns in a sequence. Another method that uses the motif content of a protein sequence was developed by Hou et. al [23]. The motif content in this case is based on local structural properties of a protein sequence. One of the most successful local structure prediction methods is by Bystroff [70]. The prediction method relies on a short sequence pattern library that correlates with protein three-dimensional structure. In this library, called I-sites, there are a total of 263 sequence-structure profiles each of which corresponds to a unique structure motif which is more specific than the three-state secondary structure. Using this structural content in a support vector machine, Hou et al [23] achieve performance comparable to the SVM-Pairwise and better than the SVM-Fisher and SAM methods. One of the advantages of SVM-I-Sites is the efficient computation of the structural content of a query sequence.

Zaki et al [67] and Saigo et al [71] have performed a comparative study of the prominent SVM-based remote homology detection methods. The study by Saigo et al is a more in depth analysis of the SVM-based as well as HMM based methods. On a single benchmarking dataset, extracted from the SCOP version 1.3 database, the results are titled in favor of SVM-Fisher. Their findings indicate that a combination of generative methods and a discriminative model are more efficient and reliable when it comes to homology detection. However since their study in 2003, there have been new methods that demonstrate better performance than SVM-Fisher or SVM-Pairwise [16-18, 23, 24, 68]. It would be an interesting exercise to do a follow up comparative study.

The above machine learning based studies all follow a similar pattern. The basic premise of these approaches remains the same:

1. Transform the protein sequences into a fixed length numeric vector representation
2. train a set of protein family specific classifier
3. Evaluate the approach using statistical validation.

The second phase essentially differs only in the software used for the SVM implementation and the selection of parameters used to generate the classifiers. The primary difference is in the first phase – the approach taken for vectorization of the sequences.

Despite improved sensitivity for the task of remote homology detection, there are caveats to a discriminative family-based approach: since the protein families used in family-based discriminative models are predetermined, only proteins that belong to a known and trained family can be classified. Most discriminative methods answer the question “Whether the given protein belongs to an existing family?” Thus new protein families or homologous sequence pairs outside these defined families, i.e. families for which models are pre-built, cannot be discovered. However, these approaches illustrate the power of applying machine learning methods to the problem of homology detection. The goal of this work is to investigate the possibility of improving upon existing family-based methods and develop the first non family-based SVM method for sensitive homology detection.

3.5 Protein ranking by network propagation

The searching of homologs for a protein sequence from a large database can be equated to a web search for a query string. In the context of a web-search a string is searched against a large network of pages that contains part of that string. The highly popular search engine GOOGLE is based on the PageRank algorithm [72] that takes advantage of the global network structure by exploiting the number of outgoing and incoming links into each indexed page. The output of the web query is a ranked list of web pages that are either similar to the given query or contain parts of the query. A protein homolog search is very similar in operation where a protein sequence is searched against a large database and a ranked list of proteins is produced with a score depicting either the probability of the match or its strength. Algorithms such as RankProp [73] and MotifProp [15] work on a similar principle. A protein database is represented as a weighted network where the links are weighted by a sequence similarity score between two protein nodes. Both algorithms exploit the global structure of this protein similarity network. The RankProp algorithm demonstrates that an advantage can be gained by including information about the global network structure in a protein sequence database search algorithm. In contrast to PSI-BLAST, which computes the local structure or a profile on the fly, RankProp begins from a pre-computed similarity network. The similarity network can be computed using any of the pairwise similarity algorithms such as Smith-Waterman, BLAST, etc. A ranked list of protein sequences is produced as output for a given query. The query is added to the database on the fly, and propagating the link information starting from the query sequence. The network propagation in RankProp is achieved by means of diffusion techniques [74], one similar to the field of machine learning. The RankProp algorithm achieves better rankings than the initial PSI-BLAST or BLAST rankings produced on the network (database) [73].

In a related effort, Kuang et. al [15] outline a graph-based algorithm called MotifProp, where the relationship between a protein and its motif content is represented in a protein-motif network. The MotifProp algorithm tries to exploit the structural aspect of a protein sequence by representing the local conserved regions (motifs) within the protein sequence in a network. Using a similar network propagation mechanism like RankProp, MotifProp represents the protein-motif relationships as a bipartite graph. A bipartite graph has two sets of nodes, one set representing the proteins and the other set representing the motifs within that protein list. A set of activation values are associated with each of the nodes and an alternating mechanism is used to propagate the activation in either directions. Similar to RankProp, MotifProp achieves good results on a benchmark dataset when compared against PSI-BLAST and RankProp.

Weston et. al [75] take the efforts of RankProp and MotifProp one step further and incorporate principles of semi-supervised learning. An adaptive RankProp algorithm is shown to outperform the original algorithm [75]. The semi-supervised learning is incorporated in this algorithm for the selection of model parameters. Both RankProp and MotifProp illustrate the application of network propagation to the problem of database searching. Incorporating semi-supervised learning improves the performance of these algorithms.

CHAPTER FOUR

4 SVM-SimLoc: Supervised Family-based Homology Detection

The integration of one or more features can provide valuable information and improve prediction accuracy in the case of homology detection. SVM-HMMSTR [24] is one such example that integrates sequence order dependent and independent features. Feature integration or Genomic Data Fusion has gained sufficient momentum in the fields of protein-protein interaction networks prediction and function prediction [76] among others. This data fusion can be done at two levels, the data level, or by fusing the transformed datasets (i.e. kernels) from two independent datasets- a method known as kernel fusion [77]. These methods however have only focused on one aspect of the sequence at a time, e.g., sequence similarity. Ben-Hur et al showed that using a motif-based representation of a protein sequence in a machine learning framework can provide results comparable to current discriminative approaches [11]. Apart from sequence-based homology information or motif content information, the sub-cellular localization of a protein sequence can provide valuable information for function prediction.

4.1 Introduction

As organelles and compartments within a cell are dependent on the type of organism, eukaryotic or prokaryotic, location prediction tools are developed catering to specific types. PSORT [78] is one of the first computer programs for predicting sub-cellular localization for plant sequences. Later versions of PSORT, such as PSORT II [79] and PSORTB [80], support eukaryotic and bacterial sequences. However, most location prediction tools are specific to a particular organism or species, for example, Target [81] can only be used for animal sequences while HSLPred [82] and PSLT [83] apply only to human proteins. Other tools such as LOCSVMPSI [84], ESLPred [85], PA-Sub [86], and LOCTree [87] are based on sequence homology to an extent. Still others such as SubLoc [88], P2SL [89], and pSLIP [90] do not classify sequences with ambiguous amino acids. The subCELlular LOcalization tool [91] popularly known as CELLO, on the other hand, is independent of species and characterizes sequences with ambiguous amino acid codes. The prediction accuracy of such computational tools has increased considerably over the last few years.

Feature integration or genomic data fusion has gained sufficient momentum in recent years; specific examples can be found in the fields of protein-protein interaction networks and prediction and function prediction [92, 93], among others. In the field of remote homology detection, SVM-HMMSTR [24] is one of the few examples that integrate sequence order-dependent and independent features. The basic premise is that combining multiple sources of low-resolution information will yield a higher-resolution solution. We hypothesize that integrating predicted sub-cellular localization of a protein sequence with a sequence similarity measure increases the ability of an SVM to accurately predict the correct family membership of proteins. The reasoning is that a homolog of a protein sequence in a particular organelle within a cell is likely to be present in the vicinity of that organelle. For example, a homolog for a membrane protein is likely to be located in the vicinity of the cell membrane across organisms.

4.2 Feature Representation

Due to SVMs robustness on large and noisy datasets, recent years have seen a surge in their use in computational biology [94]. However, SVMs require a multivariate representation of the data, which protein sequence data does not meet. Thus, protein sequences are transformed into numeric vector representation. SVM-SimLoc uses two transformations of the data, (1) sequence similarity scores and (2) subcellular localization predictions.

4.2.1 Sequence Similarity feature vectors

As discussed in Chapter 2, an SVM transforms a set of fixed-length feature vectors into kernels in high-dimensional feature space. However protein sequences by their very nature are variable length compositions of amino acids. The conversion to a fixed-length feature vector is achieved as follows. The training set of n proteins is used to transform a protein sequence into a sequence similarity vector representation. Specifically, each element in the vector is the sequence similarity score, or E-value, from the Smith-Waterman algorithm [7] of a query protein, X, against all n proteins. Thus, the feature vector corresponding to protein X is

$$F_X = \{f_{x1}, f_{x2}, \cdots, f_{xn}\}$$

where

- f_{x_i} is the E-value between the query sequence and the i^{th} sequence in the database

- n is the length of the training set.

Although we use Smith-Waterman E-values, any sequence similarity measure could be used in this manner.

4.2.2 Sub-cellular localization feature vectors

We hypothesize that using the predicted sub-cellular localization of a protein sequence increases the ability of an SVM to accurately predict the correct family membership of proteins. The reasoning is that a homolog of a protein sequence in a particular organelle within a cell is likely to be present in the vicinity of that organelle. For example a homolog for a membrane protein is expected to be located in the vicinity of the cell membrane across organisms. Once again we are faced with the problem of converting a set of variable length protein sequences to a set of fixed-length feature vectors representing their sub-cellular localization. For this task we use the CELLO prediction tool [91]. The generalized n-peptide composition of amino acids is used to predict protein folds with amino acids divided into four groups (charged, polar, aromatic and non-polar). CELLO lends itself to use in an SVM as it represents the localization of an eukaryotic protein sequence [26] as scores across the following 12 categories: (1) extra-cellular (2) plasma membrane (3) cytoplasmic (4) cytoskeleton (5) endoplasmic reticulum (6) Golgi apparatus (7) lysosome (8) mitochondria (9) chloroplast (10) peroxisome (11) vacuole and (12) nucleus. If we represent the above 12 locations with indices 1 to 12, then the location vector for a protein is given by

$$G_X = \{g_{x1}, g_{x2}, \cdots, g_{xn}\}$$

where

- g_{x_i} score given by CELLO for location i and

- $n = 12$.

4.2.3 Feature Integration

Each protein sequence is now represented as two feature vectors, one based on its sequence similarity (Smith-Waterman scores) with other sequences within and outside the family membership (as described in section 4.2.1) and the second is based on their sub-cellular localization (described in section 4.2.2). SVM-SimLoc fuses these features by appending each location vector G_X to the sequence vector F_X for all protein sequences. Using both sequence similarity and cellular localization prediction required sequences to be represented as $H_X = F_X + G_X$. To account for the heterogeneous scale of the new feature vectors, the feature vectors are row-wise normalized to have a mean of zero and unity variance.

4.3 Performance and Results

4.3.1 Comparison of SVM v/s non-SVM sequence similarity based methods

In our experiments we use a benchmark dataset that is a subset of the original dataset published by Liao et al [19]. The original benchmark dataset is derived from the Structural Classification Of Proteins (SCOP) [95] 1.53 database. The SCOP database is purged using the Astral website (http://astral.stanford.edu), removing similar sequences with an E-value threshold of 10^{-25}. This procedure results in a set of 4352 protein sequences classified into super-families and families based on the presence-absence of structural domains. The benchmark dataset consists of sequences across multiple organisms, bacteria and viruses. As the length of location prediction vectors is dependent on whether the sequence is Eukaryotic or Prokaryotic, we remove all the prokaryotes from this list of 4352 sequences. As a result, we have a subset of 2630 protein sequences classified into super-families and families. In our first experiment we revisit the comparison of the most commonly used sequence similarity methods, namely Smith-Waterman and PSI-BLAST.

Researchers in the past [13, 16, 18, 19, 23] have compared their methods against PSI-BLAST scores. However, their experimental setup did not allow PSI-BLAST to run for more than a single iteration to build the model or use a large protein database to include distant homologs. To make a fair comparison, we repeated the PSI-BLAST/Smith-Waterman comparison while allowing PSI-BLAST to iterate for the full 20 iterations to ensure convergence to a query-specific scoring model. This position-specific sequence model (PSSM) is then used to score against the sequences in our benchmark data set. The primary statistic used for the statistical comparison of homology detection methods is a Receiver Operating Characteristic (ROC) [96, 97] curve. A ROC curve yields a measure of the rate of false positives versus the true positives i.e. it is a plot of the true positive rate against the false positive rate for the different possible cut-offs of a diagnostic test. The y co-ordinates are calculated as

$$Y = TP / NT \qquad (1)$$

and the x-coordinates are calculated as

$$X = FP / NF \qquad (2)$$

where TP = True Positive, FP = False Positive, NT = Total Number of trues, and NF = Total Number of Falses. The area under a ROC plot is 1 for an ideal classifier, indicating that one can infer all of the correct protein homologies (identified from the SCOP classification) without having to tolerate any false positives.

The ROC$_n$ score computes this score up to the nth false positive [96]. For ROC1, the area under the curve is calculated using the trapezoidal rule as

$$ROC_1 = \int \left[\left(X_{n+1} + X_n \right) \div 2 \right] \left(Y_{n=1} + Y_n \right) \qquad (3)$$

where values of X and Y are calculated using the above equations (1) and (2) at varying cut-offs.

```
Function calculate_ROC
Input:
        Parameter 1: SVM scores test sequences
        Parameter 2: True classification of test sequences
Output: ROC score

Step 1) Sort the SVM scores of the test sequences and get a sorted list of class labels (1 or -1) in a single column

tp=0 /* Initialize true positive */
fp=0 /* Initialize false positive */
roc=0 /* Initialize ROC score */

for each of the sorted label do
if (label=1) then
                tp=tp+1
else {
                fp=fp+1
                roc=roc+tp
                }

if (tp=0) then roc=0
else if (fp=0) then roc=1
else roc=roc/(tp*fp)
```

Figure 7 - Pseudo code for ROC score calculation

Figure 7 outlines the pseudo code to calculate the ROC score for a given set of test sequences from their true class labels and SVM classification scores. It is known that the kernel function used in training can have a significant affect on the results on the final classifier and thus two of the most popular kernels are evaluated, (1) quadratic and (2) radial basis function (rbf).

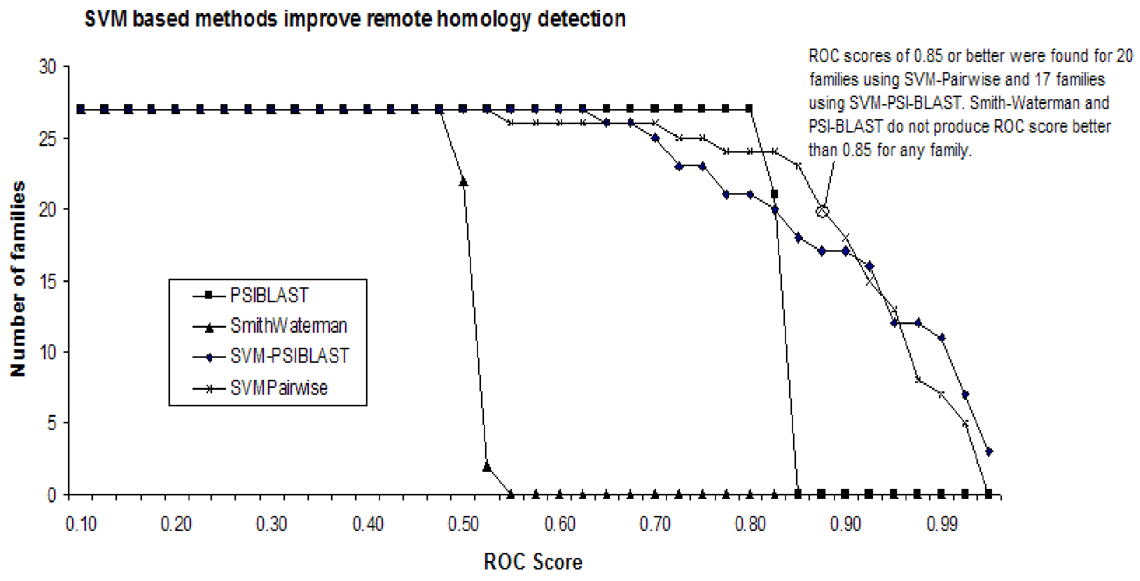


Figure 8 - Use of Support Vector Machines improves prediction accuracy for remote homology detection

Figure 8 plots the number of families on the Y-axis above a given threshold ROC score on the X-axis. The higher number of families present in the high ROC scores area, the better the classifier. Smith-Waterman scores, when used for remote homology detection, yield random results. PSI-BLAST scores, on the other hand, provide an accurate measure and yield an average ROC score of 0.829 per family. However, the introduction of SVMs drastically improves remote homology detection. There is no significant difference between SVM-Pairwise and SVM-PSI-BLAST.

We compare PSI-BLAST and Smith-Waterman scores using two methods. The first method involves computation of the ROC curves based solely on Smith-Waterman scores and PSI-BLAST scores, using family membership as a measure of homology. This gives maximum credit for the respective sequence similarity method in detecting homologs. The second method uses the very same sequence similarity scores in conjunction with a SVM. The methodology of using SVM in conjunction with Smith-Waterman scores was first published by Liao et al. [19] and is referred to as SVM-Pairwise. SVM-PSI-BLAST has never been reported to the best of our knowledge. Different SVM classifiers are built for each of the 27 families, resulting in 27 ROC curves (and ROC scores).

Statistical Method	Smith-Waterman	PSI-BLAST	SVM-Pairwise[1]	SVM-PSI-BLAST[2]
ROC Score	0.507	0.829	0.910	0.916

Table 1 - Average ROC area for each sequence similarity-based method with and without an SVM

1 - Methods using SVM outperform sequence similarity methods with a statistically significant p-value of < 1e-03

2 - There is no statistically significant difference between SVM-Pairwise and SVM-PSI-BLAST

Table 1 displays the average ROC score values for each of these methods, and Figure 8 plots the number of families for which a given method exceeds a ROC score threshold. A two-tailed signed rank test of the areas under the ROC curves is used to compare the methods. Figure 8 clearly shows that the SVM methods significantly outperform pair-wise similarity algorithms in the task of remote homology detection (p-value 5e-06 for Smith-Waterman and SVM-Pairwise; p-value 9e-04 for PSI-BLAST and SVM-PSI-BLAST). PSI-BLAST scores have not been utilized previously in conjunction with an SVM. It is demonstrated here that there is no significant different between SVM-Pairwise and SVM-PSI-BLAST when both methods are performing in optimal conditions.

4.3.2 Comparison of SVM-SimLOC

We report the results of combining location prediction and sequence similarity using SVM-SimLoc as compared to pairwise similarity based SVM-Pairwise and only location prediction based vectors. The above benchmark dataset is further profiled using true SCOP classifications to determine the number of sequences within each family. For each family, the protein domains within the family are considered positive test examples, and the protein domains outside the family but within the same super-family are taken as positive training examples. The data after sub-setting yields 27 families with at least 10 family members (positive test) and 5 superfamily members (positive train). Negative samples are taken from outside of the positive sequences' fold, and are randomly split into train and test sets in the same ration as the positive examples. Table 2 outlines the details about individual families and super-families as well as the total number of sequences within the positive train and test sets for each family.

			Positive		**Negative**	
Family number	**Class**	**Family**	**Test**	**Train**	**Test**	**Train**
1.27.1.1	All alpha proteins	Long chain cytokines	6	12	858	1754
1.27.1.2	All alpha proteins	Short chain cytokines	8	10	1183	1429
1.4.1.1	All alpha proteins	Homeodomain	23	16	1218	1347
1.4.1.3	All alpha proteins	Myb	9	30	447	2118
1.41.1.2	All alpha proteins	S100 proteins	6	36	379	2206
1.41.1.5	All alpha proteins	Calmodulin-like	25	17	1556	1029
2.1.1.1	All beta proteins	V set domains (antibody variable domain-like)	31	77	652	1825
2.1.1.2	All beta proteins	C1 set domains (antibody constant	22	86	445	2032

		domain-like)				
2.1.1.3	All beta proteins	V set domains (antibody variable domain-like)	8	100	158	2318
2.1.1.4	All beta proteins	I set domains	33	75	682	1795
2.1.1.5	All beta proteins	E set domains	14	94	546	1931
2.28.1.1	All beta proteins	Legume Lectins	44	11	1826	749
2.28.1.3	All beta proteins	Galectin (animal S-lectin)	6	49	247	2328
2.5.1.3	All beta proteins	Multidomain cupredoxins	8	5	1090	1527
2.52.1.2	All beta proteins	Phosphotyrosine-binding domain	5	12	778	1835
2.56.1.2	All beta proteins	Fatty acid binding protein-like	8	11	1112	1499
3.2.1.2	Alpha & beta proteins	Tyrosine-dependent oxidoreductases	8	13	780	1829
3.32.1.8	Alpha & beta proteins	G proteins	8	14	544	2064
3.42.1.5	Alpha & beta proteins	Glutathione S-transferases, N-terminal domain	10	16	889	1715
3.42.1.8	Alpha & beta	Glutathione peroxidase-	5	21	321	2283

	proteins	like				
7.3.10.1	Small proteins	EGF-type module	95	11	2125	229
7.3.5.2	Small proteins	Spider toxins	9	12	1007	1347
7.3.6.1		Long chain scorpion toxins	9	33	500	1854
7.3.6.2	Small proteins	Short-chain scorpion toxins	26	16	1481	873
7.3.6.4	Small proteins	Plant defensins	5	37	296	2058
7.39.1.2	Small proteins	Nuclear receptor	7	20	692	1911
7.39.1.3		LIM domain	14	13	1385	1218

Table 2 - SCOP families for Eukaryotic sequences.

Starting with datasets of vectors for each of the 2630 protein sequences, they were then separated in three vectorized sets. The first set contained only the sequence similarity feature vectors; the second contained only the location feature vectors while the third set was our integrated feature vectors generated by SVM-SimLoc. Since the integrated approach uses sequence similarity based on the Smith-Waterman algorithm, results for SVM-SimLoc are only compared to SVM-Pairwise.

SVM classifiers were built from the training sets for each of the 27 families. The area under the ROC curve is calculated for each of the 27 families for which the SVM classifier is constructed (Table 3) for both kernels. The resulting average ROC score over all 27 families is observed in Table 3.

Kernel	**SVM-Pairwise**	**Location**	**SVM-SimLoc**

RBF	0.912	0.78	0.935
Quadratic	0.910	0.779	0.927

Table 3 -Average ROC area for each method with different kernels

Kernel	**SVM-Pairwise v/s SVM-SimLoc**	**Location v/s SVM-SimLoc**
RBF	0.0057	2.4e-004
Quadratic	0.08	5.43e-005

Table 4 - Comparison of statistical significance (p-values) between methods using different kernel functions.

A two-tailed signed rank test [98] of the areas under the ROC curves is used to compare SVM-SimLoc with each of the independent datasets; SVM-Pairwise and Location. The p-values associated with these comparisons are given in Table 4. Overall, the rbf kernel returned higher average ROC curves for all methods and thus was selected for the overall analysis.

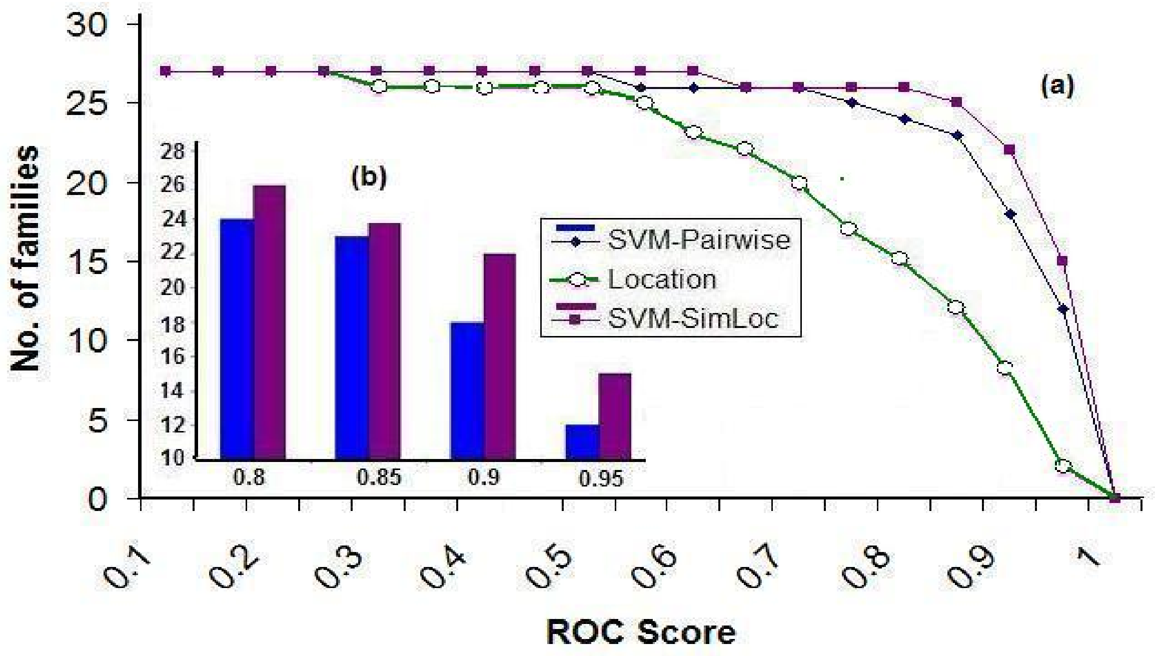


Figure 9 - Relative performance of SVM-SimLoc (a) SVM-SimLoc yields a better performance based on the ROC score across 27 families against SVM-Pairwise (p-value of 0.0057) and Location (p-value of 2.4e-004) in a two-tailed signed rank test. The Y-axis plots the number of families for which a given method exceeds a ROC score threshold (X-axis). (b) The graph plots the number of families on the Y-axis that achieve a ROC score of 0.8 or better on SVM-Pairwise and SVM-SimLoc. It can be observed that SVM-SimLoc results in a significantly higher number of families with high ROC scores than SVM-Pairwise. Location vectors are ignored in this figure for easy viewing.

A complimentary statistic used for comparison of remote homology methods is the mean rate of false positives (RFP). The median RFP is the proportion of non-homologous pairs that have a score higher than the median score of the homologous pairs. In the case of the median RFP, a score of 0 represents the best classifier. For comparative purposes with the ROC score this metrics is represented as one minus the median RFP, and thus similar to the ROC score, the closer to 1 the better the classifier. Figure 10 provides the average ROC and average complement median RFP scores across all 27 families considered in the experiment. In both cases higher ROC scores and complement median RFP are achieved by SVM-SimLoc.

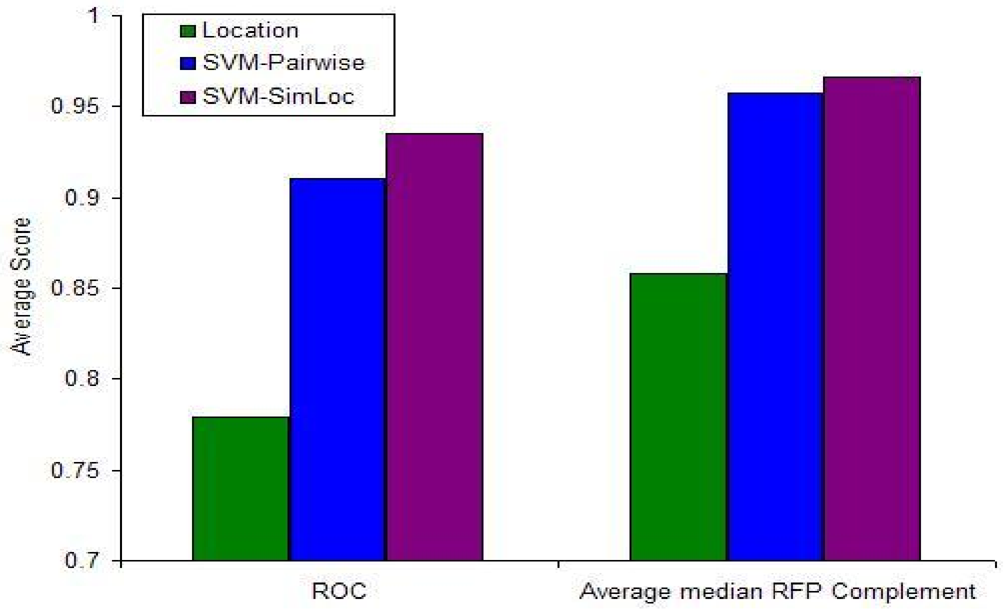


Figure 10 - Average ROC score and average median RFP complement. The average ROC scores are displayed in Table 3 and the median RFP values are 0.142, 0.043, and 0.034 for Location, SVM-Pairwise and SVM-SimLoc, respectively.

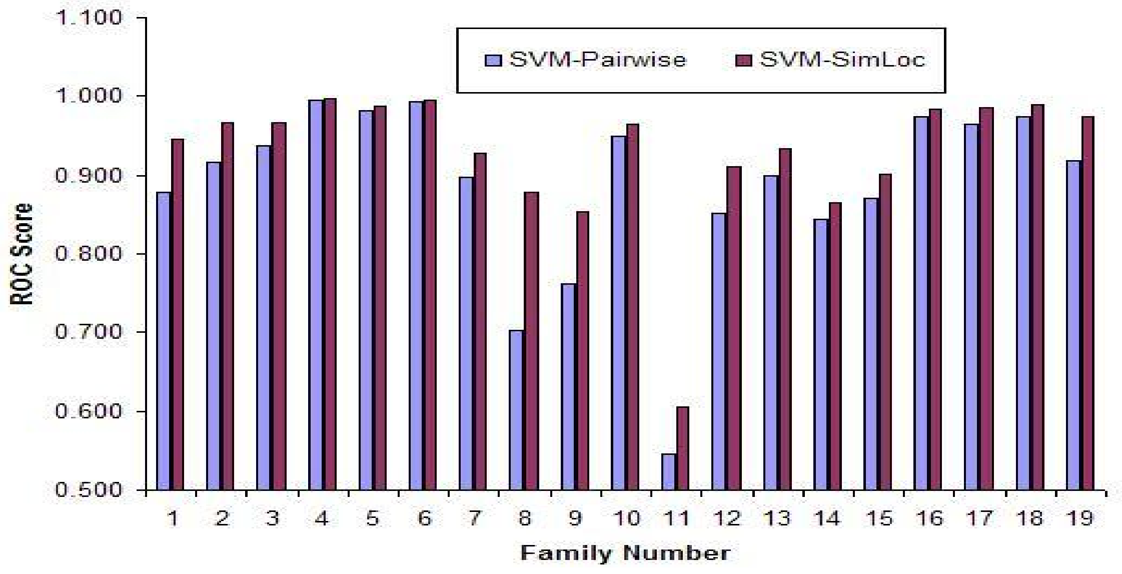


Figure 11 - Family wise comparison of average ROC scores

Number	Id	Class	Family
1	1.27.1.2	All alpha proteins	Long chain cytokines
2	1.4.1.1	All alpha proteins	Homeodomain
3	1.4.1.3	All alpha proteins	Myb
4	1.41.1.5	All alpha proteins	Calmodulin-like
5	2.1.1.1	All beta proteins	V set domains (antibody variable domain-like)
6	2.1.1.2	All beta proteins	C1 set domains (antibody constant domain-like)
7	2.1.1.5	All beta proteins	E set domains
8	2.28.1.1	All beta proteins	Legume Lectins
9	2.28.1.3	All beta proteins	Galectin (animal S-lectin)
10	2.5.1.3	All beta proteins	Multidomain cupredoxins
11	2.52.1.2	All beta proteins	Phosphotyrosine-binding domain
12	2.56.1.2	All beta proteins	Fatty acid binding protein-like
13	3.2.1.2	Alpha and beta proteins	Tyrosine-dependent oxireductases
14	3.42.1.5	Alpha and beta proteins	Glutathione S-transferases, N-terminal domain
15	3.42.1.8	Alpha and beta proteins	Glutathione peroxidase-like

16	7.3.10.1	Small proteins	EGF-type module
17	7.3.5.2	Small proteins	Spider toxins
18	7.3.6.2	Small proteins	Short-chain scorpion toxins
19	7.39.1.2	Small proteins	Nuclear receptor

Table 5 - Names of families with improved performance

SVM-SimLoc achieved higher ROC areas for 19 of the 27 families. Figure 11 provides a family by family comparison based on the ROC score between SVM-Pairwise and SVM-SimLoc. It can be observed that the scores are vastly improved on certain families using SVM-SimLoc e.g. 1, 2, 8, 9, 11 and 19. Table 5 captures the names of the families associated with Figure 11 for reference purposes.

4.4 Discussion

The accurate annotation of newly sequenced proteins precedes much of the meaningful work that can be done in understanding molecular systems. For example, the analysis of microarray data and co-regulation in general, is much more meaningful when the genes are observed in a functional context. Yet this relies heavily on one's ability to accurately and sensitively identify homologous proteins which are of distant evolutionary origin and hence, have a low degree of sequence similarity. It only makes sense to incorporate as much information as possible to determine the family membership of a protein. The role of structure in remote homology detection has been well demonstrated, but the importance of sub-cellular localization has been largely ignored.

We demonstrate on a Eukaryotic based benchmark dataset that the use of computationally predicted localization information about a protein sequence can significantly improve remote homology detection when integrated with sequence similarity. This is achieved by basic feature integration, although integration at the kernel level could have been performed as well. It is observed that the improved sensitivity from the feature integration is robust to the kernel, significant improves are observed with SVM-SimLoc with the linear, quadratic and rbf kernel. Overall, the rbf kernel was discovered to yield the most accurate classifiers for both the integrated and independent datasets. In direct comparison to the independent dataset with the rbf kernel the sensitivity improvement was statistically significant at p-values of 0.0057 and 0.0002 for SVM-Pairwise and Location, respectively.

Our results support the ideas that modeling positive and negative examples in a machine learning framework and integration of information beyond sequence similarity adds significant value to the problem of remote homology detection. This work adds a novel dimension to the problem via the integration of predicted sub-cellular location allowing the problem to be further constrained by the biology of the protein. As systems biology continues to grow, the ability to accurately annotate all proteins in a genome will become vital, and thus improved methods of remote homology detection will become a necessity.

4.5 Computational Cost for SVM-SimLOC

The computational cost of SVM-SimLOC is a combination of the computational cost of calculating the sequence similarity vectors, the sub-cellular localization vectors and the hyperplane optimization calculation during the training phase. The sequence similarity vectors are calculated using ssearch34 program from the FASTA tools [99]. The complexity of the sequence similarity algorithm is $O(n \times m)$, where n and m are the lengths of the two sequences for which the score is computed. The sub-cellular localization scores are generated from the predictive CELLO tool. CELLO uses the one-against-one method for multiclass classification [100]. For 10 classes of localization, CELLO constructs 10(10-1)/2 = 45 SVMs. These SVM's are trained in parallel and the training cost of one SVM is a polynomial time complexity algorithm [101]. CELLO breaks down the protein sequence representation into four groups (charged, polar, aromatic and non-polar). This can be achieved in $O(n)$ time. As a result the time complexity of the SVM-SimLOC algorithm reduces to the time-complexity of the SVM optimization phase which is polynomial in nature. More importantly, since we do not have access to the CELLO prediction mechanism and server code, this process cannot be done programmatically. Manual intervention is required where the protein sequences are fed to the CELLO web server (http://cello.life.nctu.edu.tw/). Once the localization vectors are obtained, Python scripts are utilized to combine them with sub-cellular localization vectors.

4.6 Need for further research

Integrating sub-cellular localization feature vectors along with sequence similarity feature vectors significantly improve remote homology detection. However, generation of the sub-cellular localization vectors is not a trivial process. Additionally human intervention is required in the process of classification of protein sequences into respective families. One of the major shortcomings of SVM-SimLOC is its reliance on a family-based categorization of protein sequences. Like most other discriminative methods, SVM-SimLOC defines protein homology based on family membership of sequences.

In order to assist researchers perform homology searches against large databases, a paradigm shift is required. A classification model that deviates from the current family-based classification scheme needs to be developed and made accessible via intuitive user interfaces. More over the resources required to run high-throughput database searches may not be available to individual users and a mechanism needs to be created to avoid this obstacle.

CHAPTER FIVE

5 SVM-HUSTLE: Semi-Supervised Pairwise Homology Detection

As the amount of biological sequence data continues to grow exponentially we face the increasing challenge of assigning function to this enormous molecular 'parts list'. The most popular approaches to this challenge make use of the simplifying assumption that similar functional molecules, or proteins, sometimes have similar composition, or sequence. However, these algorithms often fail to identify remote homologs (proteins with similar function but dissimilar sequence) which often are a significant fraction of the total homolog collection for a given sequence. To allow detection of such remote homologs it is necessary to devise an approach that deviates from the family-based discriminative methods and at the same time provides sensitive remote homology detection in a competitive time-to-solution.

This chapter discusses a SVM-based tool to detect Homology Using Semi-supervised iTerative LEarning (SVM-HUSTLE) that identifies significantly more remote homologs than current state-of-the-art sequence or cluster-based methods. As opposed to building profiles or position specific scoring matrices, SVM-HUSTLE builds an SVM classifier for a query sequence by training on a collection of representative high confidence training sets, recruits additional sequences and assigns a statistical measure of homology between a pair of sequences. SVM-HUSTLE combines principles of semi-supervised learning theory with statistical sampling to create many concurrent classifiers to iteratively detect and refine, on-the-fly, patterns indicating homology.

5.1 Introduction

Most biologists and informaticists to date use the BLAST and the PSI-BLAST algorithms as their preferred method of homology detection. The primary reason for their allegiance to these algorithms is the speed at which the results are returned even while searching against large databases. Both these algorithms use sequence similarity to identify sequences from a database which are similar to a query sequence. The input query is searched against a large gene/protein database to determine closest matches and a ranked list of sequences is produced as output with a statistical score representing the strength of similarity. However one thing that most users of these algorithms neglect is the sensitivity of the results. It has been shown that the BLAST and PSI-BLAST algorithms produce the least sensitive results when looking for remote homologs against a database with low residue level similarity [29]. Novel algorithms and methods are needed that present the results in a competitive time-to-solution for the users and are more sensitive than the BLAST and PSI-BLAST algorithms on large datasets. Recent algorithms such as MotifProp [15] and RankProp [73] come close to reproducing a ranked list of protein sequences, more sensitive than BLAST.

5.2 SVM-HUSTLE: Combining semi-supervised learning and SVM Methodology

SVM-HUSTLE (A Support Vector Machine-based tool to detect Homology Using Semi-supervised Iterative Learning) is a new algorithm that employs a semi-supervised SVM model to iteratively identify homologs to a query sequence from a database of protein sequences. The methodology can be described by six steps details in the flow chart of the algorithm in the following figure.

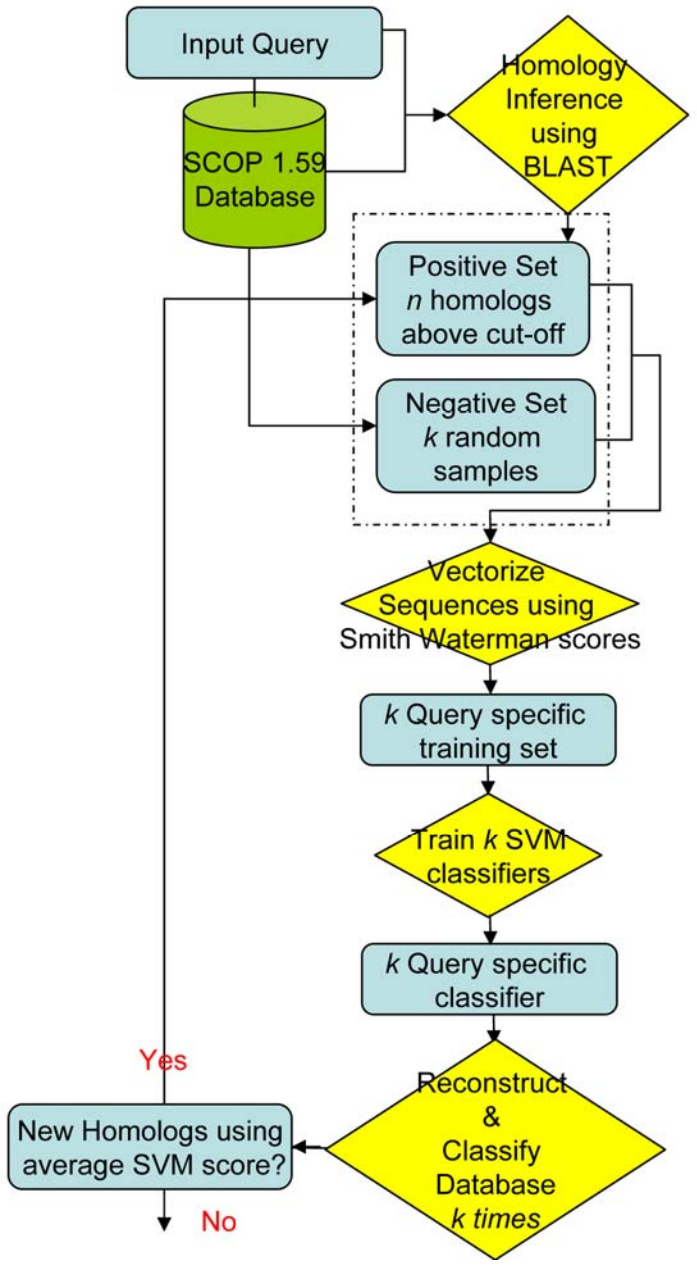

Input Query
SCOP 1.59 Database
Homology Inference using BLAST
Positive Set n homologs above cut-off
Negative Set k random samples
Vectorize Sequences using Smith Waterman scores
k Query specific training set
Train k SVM classifiers
k Query specific classifier
Reconstruct & Classify Database k times
New Homologs using average SVM score?
Yes
No

Figure 12 - SVM-HUSTLE algorithm

A novel component of this algorithm is the selection of the negative class (i.e. non-homologs) by sampling, to better cover the vector space of proteins sequences. In a general database query setting, the number of non-homologs is much larger than the homologous class and SVM-HUSTLE employs a random sampling algorithm to attain a representative negative class of proportionate size to the positive class.

The steps in SVM-HUSTLE can be explained as follows:

1. ***Identify Seed Positive Training Set.*** An initial set of n homologous proteins are identified using an auxiliary method (BLAST) against the database D. This produces a basis set of protein sequences that are similar to the original sequence with a high statistical confidence (E-value < 0.1). These n proteins form the initial positive training set (Pos) for SVM-HUSTLE. This is the only stage in the algorithm where sequence alignment scores are used.

2. ***Randomly sample k Negative Training Sets.*** Select *k* (number of concurrent SVM classifiers) random sets of proteins from *D* of size that is an integral multiple of *n* for *k* SVM constructions. These negative training examples are represented as an array *Neg[k]*. Proteins in the negative training sets must be 1) not in the high- or medium-confidence BLAST output, and 2) only in one negative training set.

3. ***Train SVM Classifiers.*** Train a set of *SVM[k]* classifiers where each training set has the same positive training examples (*Pos*) and a different set of negative training examples as defined by *Neg[k]*. The train and test set protein sequences are vectorized using the approach discussed in Section 5.4.

4. ***Vectorize the database.*** The database *D* is vectorized *k* times, once for each trained SVM. For each classifier, the proteins in *D* are converted to vector form using the positive train set and one of the negative train sets (*Neg[k]*). There will be k

different vectorized databases—one for each of the classifiers. Each instance of the vectorized database is different since they all have different negative sequences.

5. ***Classify protein pairs.*** Classify each protein in *D* and use the SVM discriminant score as a measure of classification confidence, select the top (best average score) *a*n* homologs for the given query protein q. The parameter '*a'* corresponds to the ratio of protein sequences selected as positive training examples per iteration.

6. ***Iterate.*** Repeat steps 2-5 until no new samples are classified as positive or for a maximum number of iterations (*m*).

SVM-HUSTLE uses semi-supervised learning to build multiple query-specific models of homology for the incoming query sequence. These models are then subjected to supervised learning using SVMs. The database is re-created using these query-specific models and re-classified using each distinct SVM. Finally an averaging mechanism is used that helps select the best possible matches for a protein sequence at each step. This combination of semi-supervised learning and supervised learning gives SVM-HUSTLE much of its classification power.

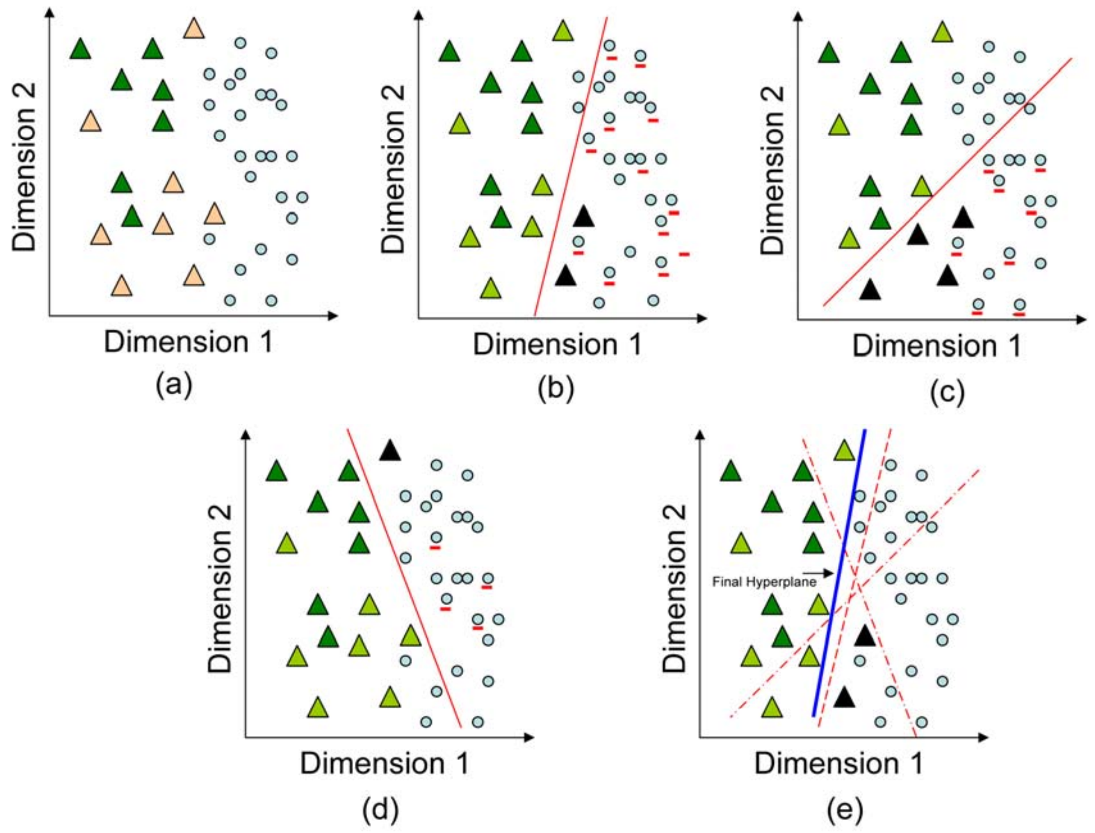


Figure 13 - Internal operation of SVM-HUSTLE. The triangles represent the homologs of a query sequence while the dots are the unlabeled data samples. Depending on the initial set of positive examples (dark green) and randomly selected negative examples (dots underlined in red) a hyperplane can be drawn in any of the three positions as illustrated in the figure. Figure (a) represents the data that is considered linearly separable in space after kernel transformations. Figures (b), (c) and (d) are examples of a hyperplane in two-dimensional space that separates the data based on separate random selection of the negative class. In each of these figures, the black triangles represent incorrectly classified data while the lime green triangles are correctly classified based on the hyperplane

selection. Figure (e) shows the final classification (hyperplane in dark blue) after taking the average of the other three hyperplanes illustrating the necessity of averaging the classifiers.

The above figure represents a schematic of the internal operation of SVM-HUSTLE after kernel transformations. The labeled and unlabeled examples can be laid out in two dimensions as shown. Constructing a single hyperplane based on the initial set of positive examples and randomly selected negative examples may not lead to correct classification in all cases. However constructing multiple hyperplanes k, would lead to an aggregate classification plane; one that will yield more accurate and sensitive results.

As its final output, SVM-HUSTLE produces a ranked listing of protein sequences that are homologous to a given query sequence with statistical scores signifying the similarity. SVM-HUSTLE does not require prior knowledge of protein family memberships for classification, so it can accurately and sensitively detect homologs which do not fall into any known protein family.

5.3 Feature Representation

In SVM-HUSTLE we use the traditional sequence similarity based representation for protein sequences. As discussed before in Chapter 4, the feature vector corresponding to a protein X is given by

$$Fx = [\, fx_1, fx_2\ , fx_3, \ldots.. fx_n \,]$$

where n is total number of proteins in the training set and f_{x_i} = E-value of the Smith-Waterman score between the sequence X and the i^{th} training set sequence [7, 19]. We use the

E-values as they are a statistical measure of similarity as opposed to raw Smith-Waterman scores between sequences which may introduce a bias based on the lengths of the individual proteins. The Smith-Waterman scores are attained using a scoring matrix of BLOSUM62, gap opening penalty of 11 and gap extension penalty of 1.

5.4 Performance and Results

5.4.1 Performance comparison with supervised and semi-supervised homology detection methods

We compare SVM-HUSTLE with PSI-BLAST, MotifProp, RankProp and their variants (k-mer MotifProp, E-Motif MotifProp etc). Comparisons between the pairwise approach of SVM-HUSTLE and family-based classification methods such as SVM-pairwise and its descendents cannot be performed as these methods rely on a family-based classification that identifies homologies within a set of pre-defined families. The comparison is made on a set of 7329 sequences, which were extracted from version 1.59 of the SCOP database, purged by using the website http://astral.berkeley.edu so that no pair of sequences shares more than 95% identity. This benchmark dataset has been used in a number of previous studies [15, 22, 73]. This SCOP dataset is divided into two portions: 379 super-families (4246 proteins) for training and 332 (3083 proteins) for testing. It is important to note that the training and testing sequences never belong to the same superfamily. The SCOP database is organized hierarchically into classes, folds, superfamilies and families. For the purpose of this experiment, two proteins that come from the same superfamily are assumed to be homologous while two proteins from different folds are treated as unrelated. For protein pairs that come from the same fold but different superfamilies are treated as unknown and not used in evaluating the algorithm. The dataset is divided into 4246 training sequences and 3083 test sequences. SVM-HUSTLE identifies remote homologs on-the-fly and as such does not require a pre-defined training set. The SCOP 1.59 training set of 4246 sequences has no overlap in superfamily membership with the 3083 test sequences and as such inclusion in the database D does not provide more positive examples, only negative examples. Similar to PSI-BLAST, the training data is included in the database used to build the model, but homology metrics are only generated for the test data to allow comparison with the current state-of-the-art. No prior knowledge of family or superfamily is used in the training of SVM-HUSTLE. Initially all data is considered unlabeled and the initial positive training set is constructed using BLAST results.

The performance of RANKPROP, MOTIFPROP and their variants were assessed by their ROC scores which were directly obtained from the authors of the algorithms. PSI-BLAST was allowed to run for a maximum of six iterations, using a default E-value threshold of 0.005 and the BLOSUM62 scoring matrix. As suggested by [102] this results in the most optimal performance of PSI-BLAST. Too few iterations results in an incomplete position-specific scoring matrix. Too many iterations result in the recruitment of large numbers of false positives, skewing the scoring matrix in favor of false positives.

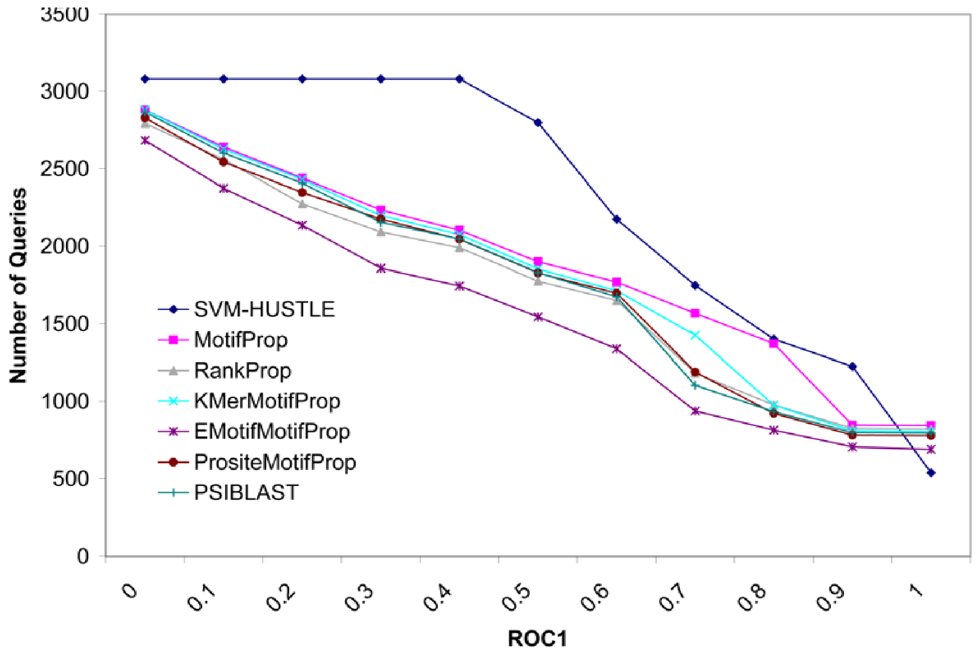
In order to compare the performance of SVM-HUSTLE against the above methods, SVM-HUSTLE was run on each of the 3083 test set protein sequences on a LINUX cluster using 4 nodes with 2 processors per node with different values of parameters defined in Section 5.3 namely:

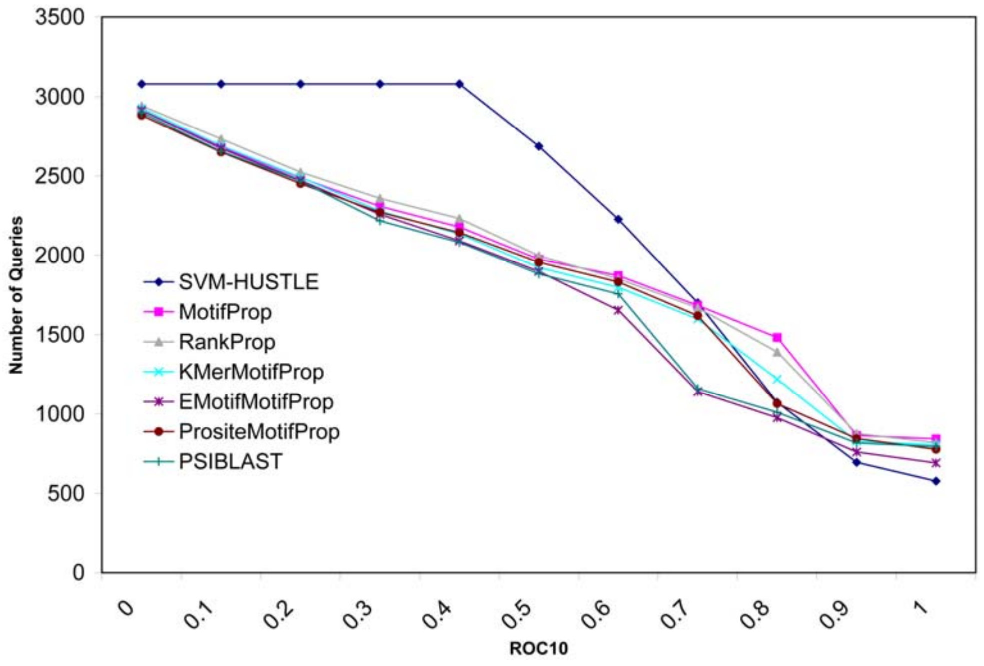
k - number of concurrent SVM classifiers,

a – ratio of protein sequences selected as positive training examples at each iteration and

m - maximum number of iterations.

SVM-HUSTLE could not be run on 3 sequences out of the total 3083 sequences in the test set as BLAST failed to produce homolog hits with E-VALUES better than 0.10. These three sequences were eliminated from all methods for comparison with SVM-HUSTLE. The network propagation methods overcome this issue by building a network despite the discovery of only a self-homolog. However, a single positive example of a sequence being homologous to itself is not adequate to train a SVM.


3500
3000
2500
2000
1500
1000
500
0
Number of Queries
SVM-HUSTLE
MotifProp
RankProp
KMerMotifProp
EMotifMotifProp
PrositeMotifProp
PSIBLAST
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
ROC1


3500
3000
2500
2000
1500
1000
500
0
Number of Queries
SVM-HUSTLE
MotifProp
RankProp
KMerMotifProp
EMotifMotifProp
PrositeMotifProp
PSIBLAST
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
ROC10

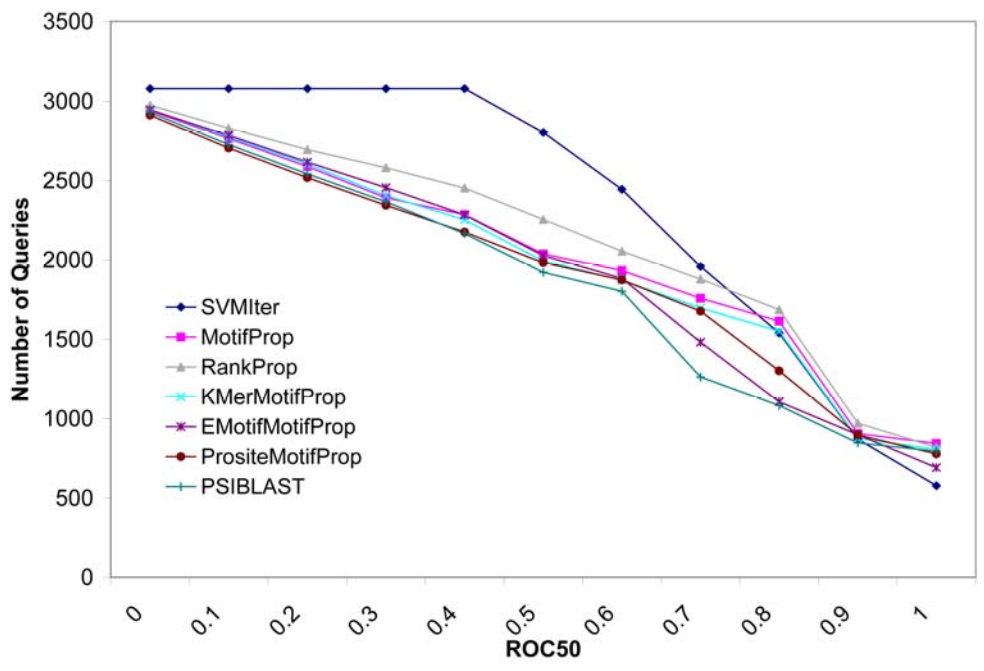


Figure 14 - SVM-HUSTLE Performance Comparison

Above figure shows the comparison of SVM-HUSTLE with the other approaches using the ROC_n curves (n = 1, 10 and 50). The graph plots the total number of queries for which a given method exceeds a ROC score threshold. As shown by Figure 14, our results suggest that SVM-HUSTLE outperforms each of the algorithms with p-values less than 1e-20 in a two-tailed signed rank test [98].

Method	SVM-HUSTLE	MotifProp	RankProp	PSI-BLAST
ROC 1	0.7445	0.6405	0.5927	0.5978
ROC 10	0.7818	0.6629	0.6674	0.6172
ROC 50	0.8122	0.6876	0.7249	0.6411

Table 6 - ROCn (n = 1, 10, 50) scores averaged over all queries

Table 6 reports the corresponding average ROC_n scores across all test set queries for a given method. All results of SVM-HUSTLE are reported for $k = 100$, $a = 1$, $m = 10$ and a BLAST cut-off of 0.10 for initial homolog set. The rationale behind selection of a generous cut-off for the BLAST search is to allow more diverse sequences to be recruited in the initial positive train set.

5.4.2 Performance comparison with profile-profile comparison methods

In this experiment we compare SVM-HUSTLE against profile-profile based remote homology detection methods such as COMPASS [27] , PROF_SIM [66] and HHSearch [29]. These programs are shown to be significantly more sensitive than PSI-BLAST and have been applied to identify novel homologies.

To compare the results sequences were extracted from the SCOP 1.63 database using the website http://astral.berkeley.edu so that no pair of sequences shares more than 20% identity (SCOP-20). This results in a total of 3691 sequences, the homologies for which are obtained from their SCOP classifications. This benchmark dataset, previously used in [29], has only 41505 true positive homolog pairs while there are 1.08 x 107 false positives. The results for the above profile-profile comparison algorithms are extracted from [29] while SVM-HUSTLE was set up to run each of the 3691 sequences with varying parameters. We report the results of SVM-HUSTLE for $k = 60$, $a = 1$ and $m = 10$ for this experiment. To include sufficient number of homologs in the initial training set, the BLAST cut-off was increased to 1.0 since this dataset has hardly any sequence similarity. In order to recruit sequences in a semi-supervised technique, we augment the 3691 protein sequences with sequences from the previous 7329 benchmark dataset. This results in a total of 8272 unique sequences. However, the test statistics are calculated only on the original 3691 SCOP-20 dataset.

As previously report in [29] at a 10% error rate (FP /(TP + FP) = 10%) BLAST detects only 908 homologous pairs which is 2.2% of the total number of homologs. PSI-BLAST performs slightly better and results in a correct identification of 17.7% while HMMER finds 18.7%. PROF_SIM and COMPASS find 24.9% and 34% respectively. HHSearch algorithms perform much better than any of the existing methods identifying 40% to 50% of the homologs, based on usage of additional knowledge such as the correlation score and predicted secondary structure. SVM-HUSTLE achieves an identification rate of 48.5%, correctly identifying 20,153 of the 41,505 homologs. Modification of the configuration parameters may result in higher percentage of homologs being found, for e.g. increasing the BLAST cut-off from 1 to 10 and increasing the number of concurrent SVM's results in an additional 5% increase in homolog detection. These results are comparable to HHSearch while they outperform both PROF_SIM and COMPASS yielding almost twice the number of homolog detection.

Our results indicate the importance of using semi-supervised learning in combination with a supervised approach. Using the SCOP 7329 dataset as a source of unlabeled data (in a semi-supervised setting) increases the homology detection capability of SVM-HUSTLE. Secondly, SVM-HUSTLE can be easily reproduced and does not require the complicated profile-profile alignments or the alignments of a large number of variable length sequences in order to generate HMM profiles.

5.5 SVM-HUSTLE Implementation

SVM-HUSTLE is implemented using MPI libraries for parallel programming and ANSI C. A version of SVM-HUSTLE is compiled for both 32 bit and 64 bit Red Hat Linux systems and is available for download. The source code cannot be released due to intellectual property concerns. Much of the framework to set up and parallelize the problem is developed here at the Pacific Northwest National Laboratory while the support vector machine implementation is using the source code from SVMLight [103]. Our preliminary experiments were conducted by using a file-based communication with SVMLight, where training and testing files were written to storage media and SVMLight was invoked on these files. The results file is later parsed by SVM-HUSTLE into memory for further processing. One of the major disadvantages of such a communication model was the latency in file writing and the increased kernel-based calls. Improving on the initial implementation, SVM-HUSTLE directly hooks into SVM-Light creating data structures and objects in memory. These data structures directly communicate with the SVMLight API (Application Programming Interface) to avoid the file writing and kernel calls.

5.6 Computational cost for SVM-HUSTLE and Scalability

Our results clearly demonstrate the improved performance characteristics of SVM-HUSTLE when detecting remote homologs. This increased sensitivity in homology detection does however come with increase in computational cost with respect to PSI-BLAST. A query specific training set is generated with similar positive exemplars and randomly sampled multiple negative exemplars. Since a query specific model is built for each incoming query, the target database *D* has to be reconstructed (re-vectorized) using this model, for each concurrent SVM classifier. The reconstruction cost of the database is the biggest bottleneck for SVM-HUSTLE.

The database reconstruction cost is directly impacted by the parameter k; the number of concurrent SVM classifiers, which linearly increases the time-to-solution for a given query. Increasing the number of concurrent classifiers introduces greater variability in the negative set that is randomly selected. As a result, we can better represent the space of negative exemplars in the training space while a smaller value of k only covers part of the database. We experimented with multiple values of this parameter and report results for $k = 60$. Our evaluation indicates a statistically significant increase in accuracy values when moving from $k=4$ to $k=100$. The transition from $k=40$ to $k=80$ is significant and as a result we settle for the average case where $k=60$. A formal estimation of the best value of k would be part of future work on SVM-HUSTLE.

Depending on configuration parameters for SVM-HUSTLE, the database would be reconstructed in the worst case (complete 10 iterations, 60 samples) for m (number of iterations) x k (concurrent SVM classifiers) times, 600 in our case. The best case scenario for SVM-HUSTLE would be when $m = 1$ which would amount to k reconstructions. If a computing facility is chosen such that there are multiple processors and the processor to classifier ratio is unity, the computation can be effectively expected to complete in time proportional to the size of the database. As the database grows in size (number of sequences), the database reconstruction cost can be prohibitive in nature and can slow down classification. Currently, SVM-Hustle requires specialized hardware to scale to large databases, such as nr. An additional challenge with large datasets is that in the case where a query has a large number of homologs, e.g., over 10,000, the size of the SVM training set would be prohibitive. However this is easily circumvented by setting a maximum limit on the initial positive train set as well as the BLAST cut-off for selection of training examples. We are currently developing a web service that would allow searches against various large sequence databases, public or user-defined. This would offer access to the specialized hardware required to run the algorithm on larger datasets. Additionally, an executable that can be used on datasets of less than 10,000 proteins is available at www.sysbio.org/sysbio/networkbio/svm_hustle.stm.

The average time to solution for a single query in both experiments using SVM-HUSTLE is in seconds once the initial setup is completed. A large number of the queries from the test set (85%) complete well within the average time however queries that have a large number of training examples require more SVM optimization cycles and take around 1-2 minutes or more for convergence. Depending on the size of the database, users looking for a small set of homologs may be able to retrieve their results in a very short time.

As mentioned, the primary computational bottleneck is in creating the protein vectors on the fly using the Smith-Waterman algorithm to reconstruct the database. An alternative and efficient way of building the SVM classifier would be to use a protein representation that is independent of the training set sequences. We are currently investigating the potential of using fixed-length motif representation of protein sequences based on motif content [11, 23]. These motif-based representations would allow SVM-HUSTLE to produce rank-lists of proteins within seconds without requiring the reconstruction phase.

5.7 Conclusions

The accurate annotation of newly sequenced proteins precedes much of the meaningful work that can be done in understanding molecular systems. For example, the analysis of microarray data and co-regulation in general, is much more meaningful when the genes are observed in a functional context. This relies heavily on one’s ability to accurately and sensitively identify homologous proteins which are of distant evolutionary origin and hence, have a low degree of sequence similarity. Our approach to incorporate semi-supervised learning coupled with concurrent training of SVM classifiers, though computationally expensive compared to PSI-BLAST, improves the sensitivity of remote homology detection by a significant margin using stricter thresholds of the ROC measure. More importantly SVM-HUSTLE does not rely on knowing the superfamily classifications of query sequences making it possible to find homologs of proteins which do not fit into any known family. SVM-HUSTLE is a pairwise sequence homology detection algorithm that builds models from representative high-confidence training sequences on the fly in a semi-supervised fashion. A typical setting for the use of SVM-HUSTLE would be to search for homologous proteins from a database for a given query. SVM-HUSTLE yields significantly better results than network propagation algorithms such as the RANKPROP and MOTIFPROP on our benchmark datasets. The performance is also comparable to HHSearch which requires the computation of multiple alignments and/or structural models of the database.

CHAPTER SIX

6 Conclusions and Future Work

6.1 Summary

Annotation of sequence data with functional and structural characterizations is one of the major challenges in bioinformatics. High-throughput sequencing experiments and methods have made the use of experimental annotation methods prohibitive due to the cost and time requirements. As a result more and more computational methods are being developed to reduce the time to annotate these sequences; practically, on a regular basis.

Protein annotation is helped to an extent by the fact that protein sequences are evolutionarily related or homologous. Homology is defined as the relationship of any two characters that have descended, usually with divergence, from a common ancestral character [5]. Additionally if two sequences are deemed homologous, it is highly likely for them to preserve a common function. Inferring this common ancestry between similar sequences, termed homology detection, provides a kind of boot-strapping method by which newly sequences genomes can be characterized based on existing annotations. Knowing that a protein sequence is homologous to an already characterized sequence, the properties of the latter can be propagated to the former. The task of detection of protein sequences that are similar (homologous) to a given query protein is termed homology detection. Pairs of protein sequences that share a high percentage of identical residues are deemed homologous due to their sequences being highly similar. However there are a number of proteins which have highly divergent sequences but are functionally homologous. The detection of such protein sequences with very low residue level similarity is termed as remote homology detection. It is obvious that pairwise sequence comparison methods are highly suitable for detection of homologs with high levels of sequence similarity. However, the sensitivity and accuracy of sequence similarity based methods decreases when detecting distant homologs.

Computational methods of remote homology detection have been a subject of study for the past several years. The machine learning based methods use a discriminative supervised framework to identify remote homologs. The premise of family-based models of homology is prevalent in these approaches. Classification models are devised for individual families of protein sequences. An incoming query sequence is tested for membership within individual families and sequences belonging to individual families are identified as being homologous to all other family members. To date, the most sensitive methods for remote homology detection are methods that rely on a profile-profile comparison rather than a sequence-based comparison. A distinct signature/profile is developed for a set of protein sequences based on the frequency of amino acids and their positions. A profile is also built for an input query sequence and algorithms are employed that compare the two profiles to determine the nature of relationship.

Current computational methods have their limitations and short comings that make them difficult to deploy in real-world scenarios. While the discriminative methods are inherently family-based and practically infeasible, the profile-profile comparison based methods rely on the generation of a multiple alignment amongst all the sequences in the target database. In this thesis, we have outlined the development of two new algorithms; SVM-SimLOC and SVM-HUSTLE that aim at improving the sensitivity of remote homology detection and at the same time overcome the practical limitations of current approaches.

SVM-SimLOC follows the family-based discriminative machine learning based approach and statistically significantly outperforms other discriminative methods [104]. However the approach has only been validated on Eukaryotics organisms and still does not lend itself to being implemented as a real time application. The SVM-Hustle algorithm, on the other hand, not only deviates from the family-based modeling but also succeeds at providing the results in a competitive time to solution. Additionally, SVM-HUSTLE outperforms each of the sequence similarity based and the network propagation based algorithms and achieves the same performance results as the profile-profile comparison based methods and does not require multiple alignments to be calculated.

6.2 Future Work

Life science researchers often require an exhaustive list of protein sequences similar to their query protein. Ever since the BLAST [9] algorithm came into existence, homology search has become one of the most executed bioinformatics tasks. The speed at which results are retrieved against a large database of millions of proteins is one of the motivating factors for its use, not to mention the simplicity with which the results can be obtained. Additional tools such as PatternHunter [105] and websites like CATMA (http://www.catma.org/index.html) are examples of software that provide fast homology searches for researchers against processed databases. However, there is a shortage of software tools that allow researchers to search for homologs against user-created databases. Often while looking for homologs, users wish to search against their custom databases. It is increasingly difficult to find software tools that allow usage of custom databases and in certain cases like HHSearch [29], the process is not trivial. As future work to this thesis, we discuss the design of an integrated system which will not only facilitate public access to homology searches but also provide the ability to upload and process custom databases with minimal effort.

The proposed system uses existing freely available systems and components that have been developed at the Pacific Northwest National Laboratory (PNNL). This system uses the Bioinformatics Resource Manager (BRM) [25], the MeDICi workflow specification framework [106] and the ScalaBLAST high performance BLAST software [30]. The BRM environment serves as the client interface in this workflow scenario while the MeDICi and ScalaBLAST software operate behind the scenes and without the users' knowledge. A brief background on the three public systems is provided in the following sections. We then discuss the design of the integrated system followed by an envisaged description of its usage.

6.3 Background

6.3.1 Bioinformatics Resource Manager

The Bioinformatics Resource Manager (BRM) [25] is a systems biology toolkit that provides users with data management, data retrieval and data maintenance capabilities directly at the users desktop. BRM is designed in collaboration with biologists and caters to their mundane analysis and data integration task. It serves as a one-stop shop for annotation and integration of biological data and connects to a number of public databases and sources to retrieve annotation as well as cross reference information. BRM is implemented using a multi-tier architecture where the client software is installed on the users' desktop system. The client, on startup, connects to a remote server which hosts a series of database tables to provide cross linkage information and data integration capabilities. Amongst its other functions, BRM also provides the capability to use regular expression-based processing on datasets, launch additional analysis tools using the GAGGLE software [107] and launch analysis workflows in the background.

6.3.2 ScalaBLAST

One of the major challenges associated with the practical implementation of SVM-HUSTLE is the calculation of the initial $n \times n$ BLAST score matrix for a database with n sequences. Traditional BLAST searches are extremely effective when generating scores for a small number of sequences against a large database, however when n is in the order of thousands, a serial BLAST run may require months or even longer to compute. Another most common approach would be to write scripts that run in parallel on multiple copies of the database. While this can speed up the matrix generation, it places additional load on the resources required in terms of storage space. To avoid these and other related problems, we utilize the services of ScalaBLAST [30], a high-performance scalable implementation of BLAST that presents an entirely software-based solution. ScalaBLAST is implemented using the MPI parallel programming libraries and the Global Array software for shared memory management. Amongst its' other techniques, ScalaBLAST employs distributed memory management, latency hiding through pre-fetching database sequences, parallel I/O and multi-level parallelism to achieve higher performance and scalability. ScalaBLAST has been shown to scale linearly with respect to the number of processors.

6.3.3 MEDICI

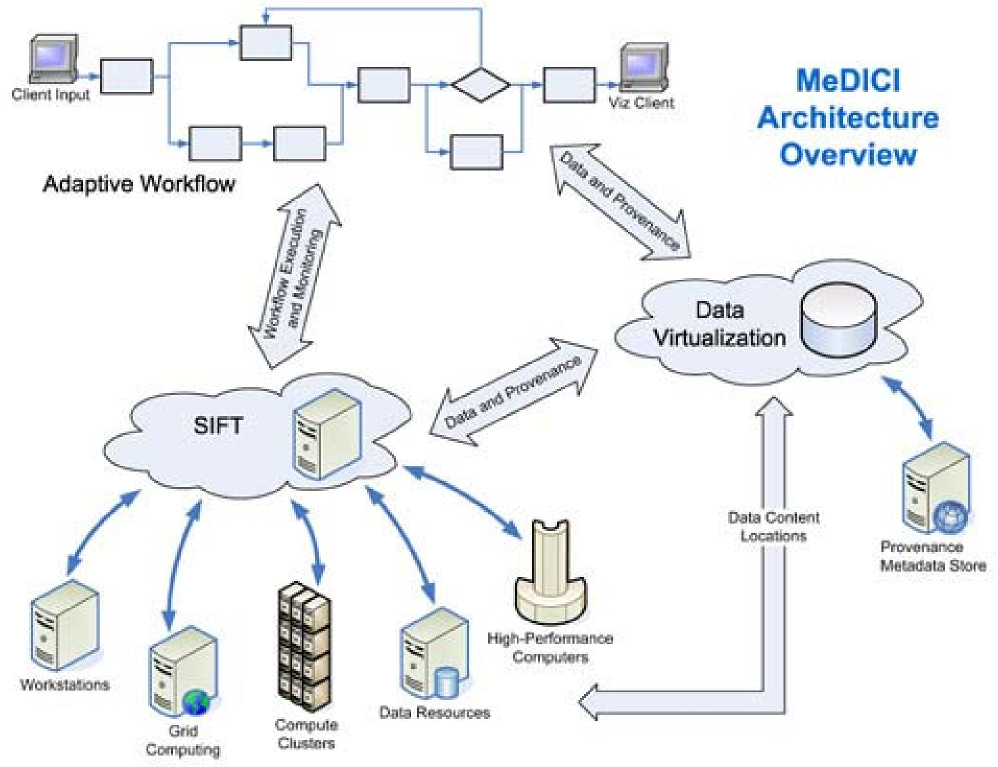


Figure 15 – MEDICI Architecture – Image courtesy Ian Gorton

The MEDICI integration framework [106] provides a component-based messaging and integration architecture for creating analytical pipelines for real-time high-throughput data analysis tasks. Engineering such real-time systems is a challenging task for a number of reasons:

- High throughput experimentation generates large amounts of data in matter of seconds. The processing of such high data arrival rates require a scalable infrastructure to support low latency inter-component communications
- Metadata and provenance information should be captured during these processes to enable reproducibility as well as tracking information

- The integration of analysis components must be relatively low cost.

The MeDICi architecture, leverages open source middleware technologies and imposes a platform agnostic programming model. Components are wrapped in a common structure to communicate with the pipeline and provide the necessary data processing or transfer functions. The MeDICi architecture has been used in a variety of applications that support high-throughput processing.

6.4 System Design

BRM is implemented in Java^{TM} and is based on 3-tier client server architecture. Users install a client implemented using Java Swing^{TM} on their desktop computer that provides interfaces to create and manage projects along with datasets, retrieve external data from public sources and launch analysis tools. Figure 16 displays the architecture of the BRM software along with the technologies used at each tier. The middle tier consists of the BRM server developed using the Enterprise Java Beans 2.0 (EJB) technology and residing in a JBOSS application server. Connections to external data sources are pre-configured within the BRM Server via the Java DataBase Connectivity Application Programming Interface (API). The BRM server is responsible for user authentication, data retrieval from external data sources, and management of user sessions while the client is used only as a front-end for all such requests. The third tier of the BRM architecture is a PostgreSQL (www.postgresql.org) database system.

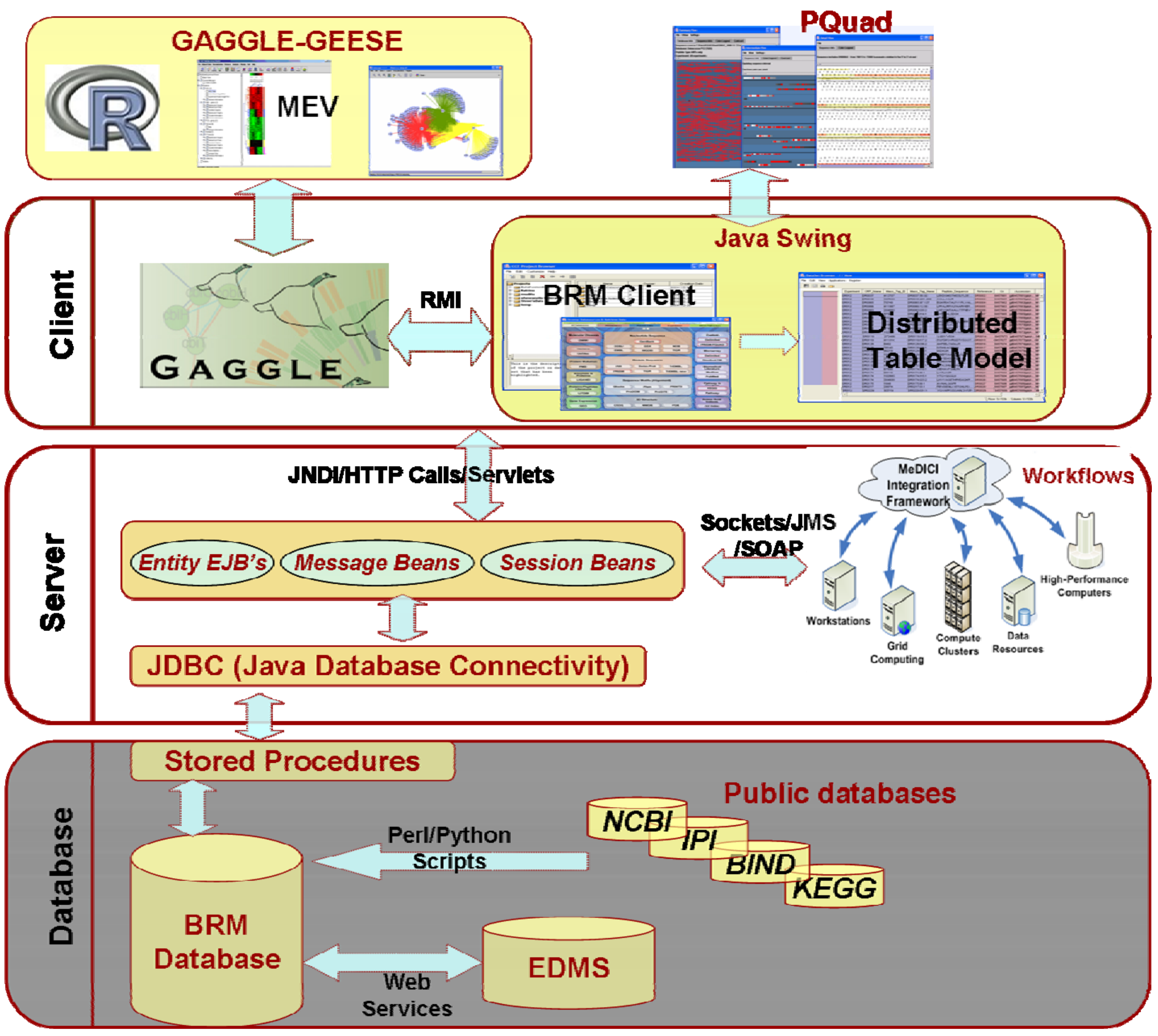


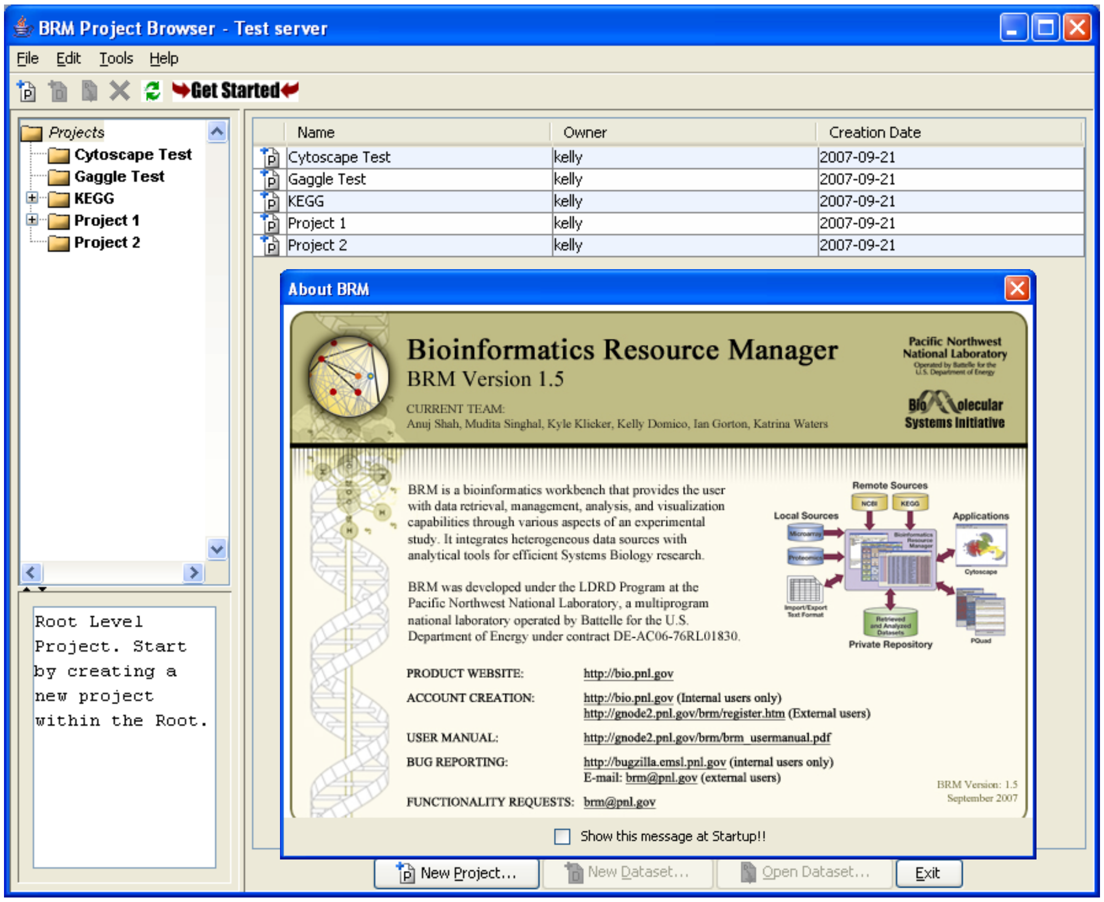
Figure 16 – BRM System Architecture

Tables representing users, projects, data sets and data-sources are accessed via the middle tier. All externally retrieved data is also stored along with metadata information in this persistent storage. Data transfer between the client-server and the individual server components is kept to a minimum to counter Internet bandwidth limitations and communication overhead common to integrative environments. Communication is achieved via short messages (Java Messaging API) containing metadata. Each server component can then retrieve data based on its metadata for further processing. A distributed caching data model is implemented where clients receive data in chunks of 2000 rows. Client-server systems require the client to be in-sync with the server for optimal operation. BRM provides an automated version checking mechanism at startup. A pop-up message informs the user of version mismatches and directs them to BRM's web page for updates.

6.4.1 User Interface Design

One of the major advantages of using the BRM client for homology searches is the ability to upload a set of custom databases that can be used as the target of a search query. These databases or datasets can be stored on the BRM server for future access. BRM serves as the data management application for future reuse of databases and retrieval of query results.

Figure 17 shows the main BRM application window, called the BRM Project Browser. The Project Browser shows the "projects" (working collections of data) and the datasets that you create and organize while working with the BRM. The first time you log in and until you create your own projects and datasets, the Project Browser will be "empty," displaying only an empty **Projects** folder in the top left pane of the window.


BRM Project Browser - Test server
File
Edit
Tools
Help
Get Started
Projects
Cytoscape Test
Gaggle Test
KEGG
Project 1
Project 2
About BRM
Remote Sources
NCBI
KEGG
Local Sources
Microarray
Proteomics
Import/Export Text Format
Bioinformatics Resource Manager
Retrieved and Analyzed Datasets
Private Repository
Applications
Cytoscape
PQuad
Pacific Northwest National Laboratory
Operated by Battelle for the U.S. Department of Energy
BioMolecular Systems Initiative
New Project...
New Dataset...
Open Dataset...
Exit

Name	Owner	Creation Date
Cytoscape Test	kelly	2007-09-21
Gaggle Test	kelly	2007-09-21
KEGG	kelly	2007-09-21
Project 1	kelly	2007-09-21
Project 2	kelly	2007-09-21

Root Level Project. Start by creating a new project within the Root.

Bioinformatics Resource Manager
BRM Version 1.5

CURRENT TEAM:
Anuj Shah, Mudita Singhal, Kyle Klicker, Kelly Domico, Ian Gorton, Katrina Waters

BRM is a bioinformatics workbench that provides the user with data retrieval, management, analysis, and visualization capabilities through various aspects of an experimental study. It integrates heterogeneous data sources with analytical tools for efficient Systems Biology research.

BRM was developed under the LDRD Program at the Pacific Northwest National Laboratory, a multiprogram national laboratory operated by Battelle for the U.S. Department of Energy under contract DE-AC06-76RL01830.

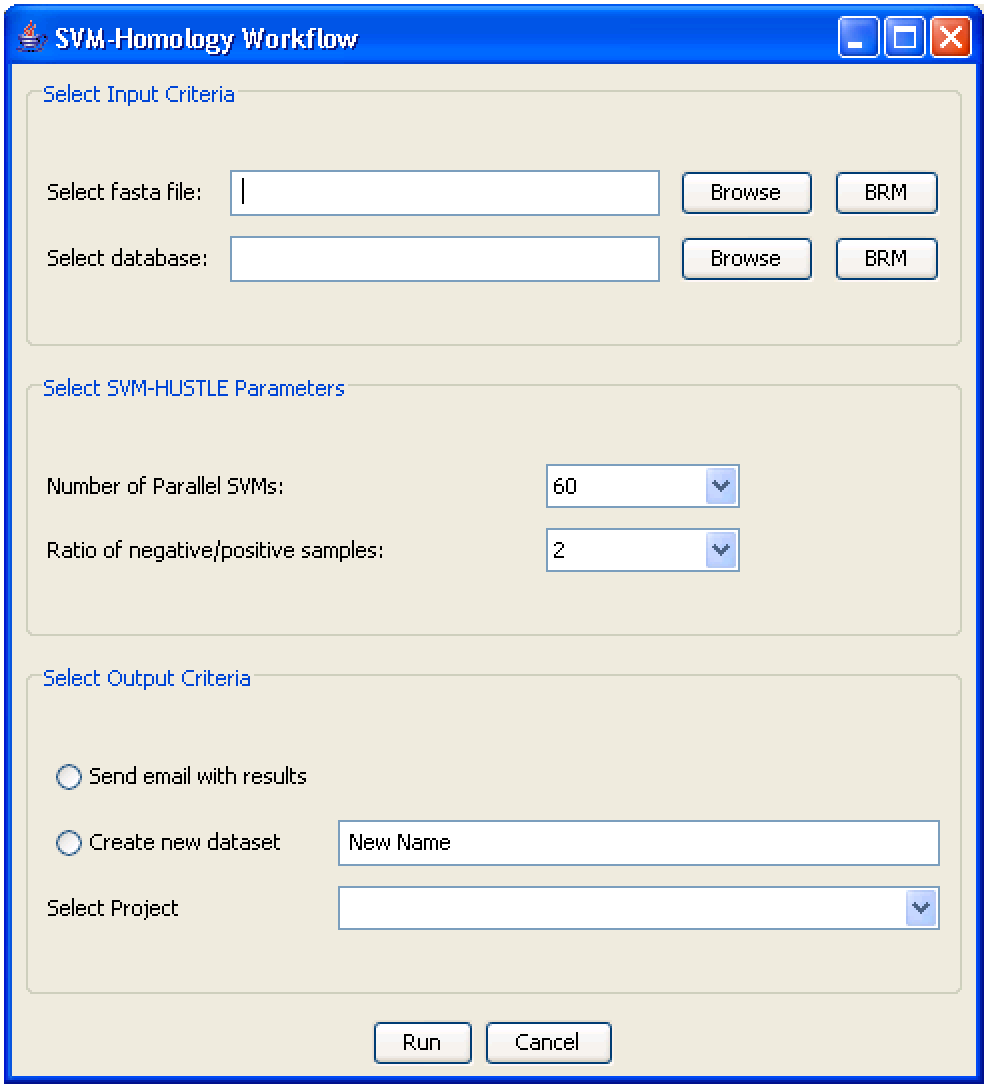
PRODUCT WEBSITE:	http://bio.pnl.gov
ACCOUNT CREATION:	http://bio.pnl.gov (Internal users only) http://gnode2.pnl.gov/brm/register.htm (External users)
USER MANUAL:	http://gnode2.pnl.gov/brm/brm_usermanual.pdf
BUG REPORTING:	http://bugzilla.emsl.pnl.gov (internal users only) E-mail: brm@pnl.gov (external users)
FUNCTIONALITY REQUESTS:	brm@pnl.gov

BRM Version: 1.5
September 2007

☐ Show this message at Startup!!

Figure 17 – Project browser and welcome dialog box

The Tools menu item on the Project Browser window will be used to launch the SVM-based homology workflow interface. The user interface for the workflow launching capability will be organized into into three sections as shown in Figure 18. In the first part of the user interface, the user specifies the location of the query FASTA file and the FASTA file that is to be used as the database/target for the search. The BRM button on the interface allows the user to select an existing dataset file from the BRM environment, while the Browse buttons enable a user to select a file from the local file system.



Select Input Criteria

Select fasta file:

Select database:

Select SVM-HUSTLE Parameters

Number of Parallel SVMs: 60

Ratio of negative/positive samples: 2

Select Output Criteria

Send email with results

Create new dataset: New Name

Select Project

Figure 18 - SVM Homology workflow interface

The second section allows the user to specify the parameters for the SVM-HUSTLE algorithm that will be run behind the scenes to complete the homology searches. The user interface allows the user to specify in particular the number of parallel SVM's that will be run. Our results, as outlined in the previous chapter, indicate that running 60 SVM's in parallel provides reasonable performance in terms of accuracy and speed. The number of processors that will be used to execute the software is not provided as a user configurable option. This is determined behind the scenes based on the availability of nodes on the cluster. Users can also specify the ratio of negative to positive examples for each individual SVM. A ratio of 2, where the number of negative training examples is double the number of positive training examples is optimal in most respects. It is to be noted that having a large number of negative samples and a small number of positive samples may hamper the ability of the SVM to identify true positives.

Finally the bottom section of the interface allows the user to specify how the results should be delivered. Currently, there are two options for the user; to receive the results via email or have a new dataset created with the results within the user environment in BRM. Depending on user selection, the user receives an email containing a link where they can download their results from.

The final step in launching the homology workflow would be to click the “Run” button at the bottom of the interface. A series of steps are followed behind the scenes in order to execute this workflow in an asynchronous manner. It is a challenge to present the user with the results in real-time and as a result, the results are presented at a later stage using an asynchronous mechanism.

6.4.2 Server Components Design

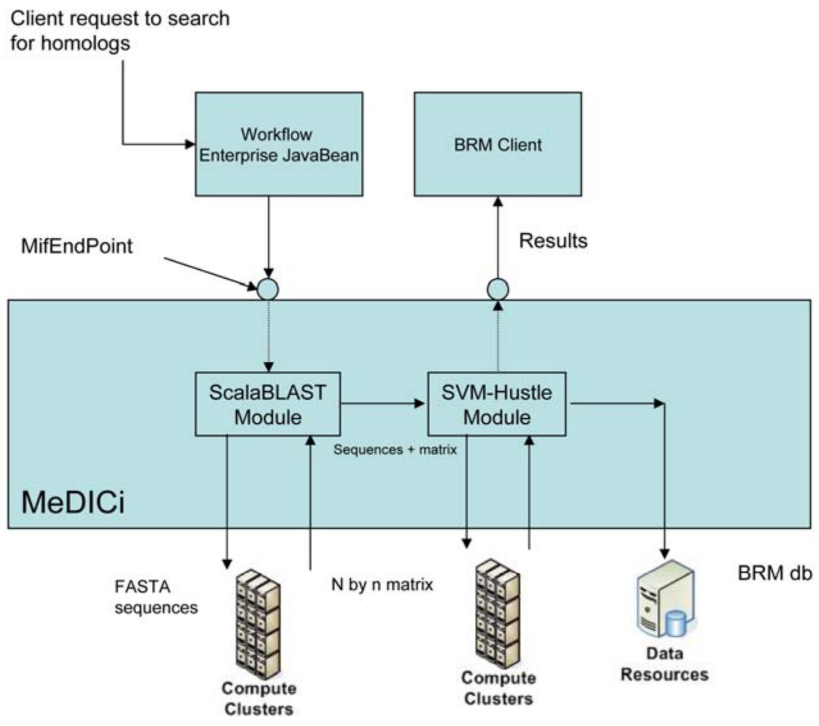


Figure 19 - Schematic of server components

Figure 19 provides a schematic of the design for the server side components that enable the homology workflow. A workflow EJB residing within the BRM server will be responsible for intercepting the client request. The MEDICI architecture is used for the resource management underlying the workflow. The FASTA database and query files are transported to the respective compute servers for use within the ScalaBLAST and SVM-HUSTLE software codes. The ScalaBLAST module is responsible for generating an N x N matrix (N number for sequences in database) which is later transported to the compute clusters running SVM-Hustle. This transfer of high-throughput data along with other data managements issues are tackled by the MEDICI architecture behind the scenes. The SVM-Hustle module can then run on the input files and the results forwarded to the BRM database. If the user requests that the results be presented as a new dataset, then the results can be stored within the user's environment in the database. If an email is requested then the files can be stored on the file system on the BRM server machine and a link sent to the user via email. The users can then retrieve those files at their own convenience.

6.5 Conclusion

Today's homology based searches require the support of high-end computing infrastructure as well as a performance guarantee in terms of time-to-solution. Most websites and software tools allow users to search for homologs against pre-existing databases and the process to search for homologs against custom databases is not trivial. Using the BRM software toolkit in conjunction with MEDICI and the high-performance BLAST implementation, ScalaBLAST, we strive to fill in this gap. Biologists and researchers are now successfully able to launch homology searches against their own databases and obtain the results at their convenience.

References

1. Alberts, B., et al., *Macromolecules: structure, shape, and function.* Molecular Biology of the Cell. 2nd edn., Garland, New York, 1989.
2. Tucker, C.L., J.F. Gera, and P. Uetz, *Towards an understanding of complex protein networks.* Trends Cell Biol, 2001. **11**(3): p. 102-6.
3. Rabi, I.I., et al., *A New Method of Measuring Nuclear Magnetic Moment.* Physical Review, 1938. **53**: p. 318-318.
4. Barlow, W., *Probable nature of the internal symmetry of crystals.* Nature, 1883. **29**: p. 186-?
5. Fitch, W.M., *Homology a personal view on some of the problems.* Trends in Genetics, 2000. **16**(5): p. 227-231.
6. Needleman, S.B. and C.D. Wunsch, *A general method applicable to the search for similarities in the amino acid seqeunces of two proteins.* Journal of Molecular Biology, 1970. **48**: p. 443-453.
7. Smith, T. and M. Waterman, *Identification of common molecular subsequences.* Journal of Molecular Biology, 1981. **147**: p. 195-197.
8. Pearson, W.R., *Rapid and sensitive sequence comparisons with FASTP and FASTA.* Methods Enzymol, 1985. **183**: p. 63-98.
9. Altschul, S.F., et al., *A basic local alignment search tool.* Journal of Molecular Biology, 1990. **215**: p. 403-410.
10. Rost, B., *Twilight zone of protein sequence alignments.* Protein Engineering, 1999. **12**(2): p. 85-94.

11. Ben-Hur, A. and D. Brutlag, *Remote homology detection: a motif based approach.* Bioinformatics, 2003. **19**(1): p. i26-i33.

12. Busuttil, S., J. Abela, and G.J. Pace, *Support Vector Machines with Profile-based Kernels for Remote Protein Homology Detection.* Genome Informatics, 2004. **15**(2): p. 191-200.

13. Jaakkola, T., M. Diekhans, and D. Haussler, *A discriminative framework for detecting remote protein homologies.* Journal of Computational Biology, 2000. **7**: p. 95-114.

14. Kuang, R., et al., *Profile-based string kernels for remote homology detection and motif extraction.* Journal of Bioinformatics and Computational Biology, 2005. **3**(3): p. 527-550.

15. Kuang, R., et al., *Motif-based protein ranking by network propagation.* Bioinformatics, 2005. **21**(19): p. 3711-3718.

16. Leslie, C., et al., *Mismatch string kernels for discriminative protein classification.* Bioinformatics, 2003. **1**(1): p. 1-10.

17. Leslie, C., et al. *Mismatch string kernels for SVM protein classification*. in *Neural Information Processing Systems*. 2003. British Columbia, Canada.

18. Leslie, C. and R. Kuang. *Fast Kernals for Inexact String matching*. in *Sixteenth Annual Conference on Learning Theory and Seventh Kernel Workshop*. 2003.

19. Liao, L. and W.S. Noble, *Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships.* Journal of Computational Biology, 2003. **10**(6): p. 857-868.

20. Tang, C., et al., *On the role of Structural Information in Remote Homology Detection and Sequence Alignment: New Methods using Hybrid Sequence Profiles.* Journal of Molecular Biology, 2003: p. 1043-1062.

21. Webb-Robertson, B.-J.M., C. Oehmen, and M. Matzke, *SVM-BALSA: Remote homology detection based on Bayesian sequence alignment.* Computational Biology and Chemistry, 2005. **29**: p. 440-443.

22. Wetson, J., et al., *Semi-supervised protein classification using cluster kernels.* Bioinformatics, 2005. **21**(15): p. 3241-3247.

23. Yuna Hou, et al., *Efficient remote homology detection using local structure.* Bioinformatics, 2003. **19**: p. 2294-2301.

24. Yuna Hou, et al., *Remote Homology Detection using Local Sequence-Structure Correlations.* PROTEINS: Structure, Function and Bioinformatics, 2004. **57**: p. 518-530.

25. Shah, A.R., et al., *Enabling high-throughput data management for systems biology: The Bioinformatics Resource Manager.* Bioinformatics, 2007. **23**(7): p. 906-909.

26. Chou, K.-C. and D.W. Elrod, *Protein subcellular location prediction.* Protein engineering design and selection, 1999. **12**(2): p. 107-118.

27. Sadreyev, R.I., D. Baker, and N.V. Grishin, *Profile-profile comparisons by COMPASS predict intricate homologies between protein families.* Protein Science, 2003. **12**: p. 2262-2272.

28. Madera, M. and J. Gough, *A comparison of profile hidden Markov model procedures for remote homology detection.* Nucleic Acid Research, 2002. **30**(19): p. 4321-4328.

29. Soeding, J., *Protein homology detection by HMM-HMM comparison.* Bioinformatics, 2005. **21**(7): p. 951-960.

30. Oehmen, C.S. and J. Nieplocha, *ScalaBLAST: A Scalable Implementation of BLAST for High-Performance Data Intensive Bioinformatics Analysis.* IEEE Transactions on Parallel and Distributed Systems, 2006. **17**(8): p. 740-749.

31. Crawford, S.L. and T.C. Fall. *Projection pursuit techniques for visualizing high-dimensional data sets*. in *Visualization in Scientific Computing*. 1990: IEEE Computer Society Press.

32. Utts, J.M., *Seeing Through Statistics*. 3rd ed. 2005: Thomson Brooks/Cole. 166-167.

33. Jolliffe, I.T., *Principal Component Analysis*. 2nd ed. Springer Series in Statistics, ed. Springer, NY: Springet. 487.

34. Ho, P.-M., et al., *PCA-based compression for image-based relighting*, in *Multimedia and Expo. ICME '03. Proceedings. International*. 2003.

35. Moon, H. and P.J. Phillips, *Computational and performance aspects of PCA-based face-recognition algorithms.* Perception, 2001. **30**(3): p. 303-321.

36. Odunsi, K., et al., *Detection of epithelial ovarian cancer using H-NMR-based metabonomics.* International Journal of Cancer, 2004. **113**(5): p. 782-788.

37. Jain, A.K., *Algorithms for Clustering Data*. Computer Science, ed. P.H.A.R. Series. 1988: Prentice Hall.

38. Hartigan, J.A. and M.A. Wong, *A K-Means Clustering Algorithm.* Applied Statistics, 1979. **28**(1): p. 100-108.

39. Johnson, S.C., *Hierarchical Clustering Schemes.* Psychometrika, 1967. **2**: p. 241-254.

40. Bohr, H., et al., *Protein secondary structure and homology by neural networks.* FEBS Letters, 1988. **241**(1-2): p. 223-228.

41. Reinhardt, A. and T. Hubbard, *Using neural networks for prediction of the subcellular location of proteins.* Nucleic Acid Research, 1998. **26**(9): p. 2230-2236.

42. McNeils, P.D., *Neural Networks in Finance*. AP Advanced Finance. 2004: Elsevier.

43. Skabar, A. and I. Cloete. *Neural Networks, Financial Trading and the Efficient Markets Hypothesis*. in *Twenty-Fifth Australasian Computer Science Conference*. 2002. Melbourne, Australia.

44. Carpenter, G.A. and S. Grossberg, *The ART of Adaptive Pattern Recognition by a Self-Organising Neural Network.* IEEE Computer, 1988. **21**: p. 77-88.

45. Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning*. 1 ed. 1989: Addison-Wesley Professional. 432.

46. Pei, M., et al., *Genetic Algorithms for classification and feature extraction*, in *Classification Soceity of North America*. 1995: Denver, Colorado, USA.

47. Huang, J., Y. Cai, and X. Zu, *A hybrid genetic algorithm for feature selection wrapper based on mutual information.* Pattern Recognition Letters, 2007. **28**(13): p. 1825-1844.

48. Ginalski, K., et al., *ORFeus: detection of distant homology using sequence profiles and predicted secondary structure.* Nucleic Acid Research, 2003. **31**(13): p. 3804-3807.

49. Mareuil, F., et al., *A simple genetic algorithm for the optimization of multidomain protein homology models driven by NMR residual dipolar coupling and small angle X-ray scattering data.* European Biophysics Journal, 2007. **37**(1): p. 95-104.

50. Vapnik, V.N., *The nature of Statistical Learning Theory*, ed. Springer. 1995, New York: Springer.

51. Vapnik, V.N., *Statistical Learning Theory. Adaptive and learning systems for signal processing, communications, and control*, ed. Wiley. 1998, New york: Wiley.

52. Baum, L.E., et al., *A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains.* The Annals of Mathematical Statistics, 1970. **41**(1): p. 164-171.

53. Elworthy, D. *Does Baum-Welch Re-estimation Help Taggers?* in *Proceedings of the 4th ACL Conference on Applied Natural Language Processing*. 1994. Stuttgart.

54. Cozman, F.G., I. Cohen, and M.C. Cirelo. *Semi-Supervised Learning of Mixture Models and Bayesian Networks*. in *Twentieth International Conference of Machine Learning*. 2003.

55. Dempster, A., N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, 1977. **39**(1): p. 1-38.

56. Zhu, X., *Semi-Supervised Learning Literature Survey*. 2006, University of Wisconsin: Madison.

57. Dunker, A.K., et al., *Intrinsic disorder and protein function.* Biochemistry, 2002. **41**: p. 6573-6582.

58. Lipman, D.J. and W.R. Pearson, *Rapid and Sensitive Protein Similarity Searches.* Science, 1985. **227**: p. 1435-1441.

59. Park, J., et al., *Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods.* Journal of Molecular Biology, 1998. **284**(4): p. 1202-1210.

60. Altschul, S.F., et al., *Gapped BLAST and PSI-BLAST: A new generation of protein database search programs.* Nucleic Acid Research, 1997. **25**: p. 3389-3402.

61. Baldi, P., et al., *Hidden Markov models of biological primary sequence information.* Proc. Natl Acad. Sci, 1994. **91**(3): p. 1059-1063.

62. Eddy, S.R., *Profile hidden Markov models.* Bioinformatics, 1998. **14**: p. 755-763.

63. Karplus, K., C. Barrett, and R. Hughey, *Hidden Markov models for detecting remote protein homologies.* Bioinformatics, 1998. **14**(10): p. 846-856.

64. Bucher, P., et al., *A flexible motif search technique based on generalized profiles.* Computational Chemistry, 1996. **20**: p. 3-23.

65. Grundy, W.N., et al., *meta-MEME: Motif based hidden Markov models of protein families.* Bioinformatics, 1998. **13**(4): p. 397-406.

66. Yona, G. and M. Levitt, *Within the twilight zone: a sensitive profile-profile comparison tool based on information theory.* Journal of Molecular Biology, 2002. **315**: p. 1257-1275.

67. Zaki, N.M., S. Deris, and R.M. Illias, *A Comparative Analysis of Protein Homology Detection Methods.* Journal of Theoretics, 2003. **5**(4).

68. Leslie, C., E. Eskin, and W.S. Noble, *The spectrum kernel: a string kernel for SVM protein classification*, in *Pacific Symposium on Biocomputing*. 2002, Altman, R.B., Dunker, A.K, Hunter, L. Lauerdale, K. and Klein, T.E.: Hawaii.

69. Huang, J.Y. and D. Brutlag, *The eMOTIF Database.* Nucleic Acid Research, 2001. **29**(1): p. 202-204.

70. Bystroff, C. and D. Baker, *Prediction of local structure in proteins using a library of sequence-structure motifs.* Journal of Molecular Biology, 1998. **281**(3): p. 565-577.

71. Saigo, H., et al., *Protein homology detection using string alignment kernels.* Bioinformatics, 2004. **20**(11): p. 1682-1689.

72. Sergey Brin and L. Page. *The anatomy of a large scale hypertextual web search engine*. in *Seventh International World Wide Web Conference*. 1998.

73. Wetson, J., et al., *Protein ranking: from local to global structure in the protein similarity network.* Proc. Natl Acad Sci. , 2004(101): p. 6559-6563.

74. Zhu, X., Z. Ghahramani, and J. Lafferty. *Semi-supervised Learning Using Gaussian Fields and Harmonic Functions*. in *International Conference on Machine Learning, ICML-2003*. 2003: AAAI Press.

75. Wetson, J., et al., *Protein Ranking by Semi-Supervised Network Propagation.* BMC Bioinformatics, 2006. **7**(Suppl I): p. S10.

76. Lanckriet;, G.R.G., et al. *Kernel-based Data Fusion and its application to protein function prediction in Yeast*. in *Systems Biology*. 2004.

77. Lanckriet;, G.R.G., et al., *A statistical framework for genomic data fusion.* Bioinformatics, 2004. **20**(16): p. 2626-2635.

78. Nakai, K. and M. Kanehisa, *Expert system for predicting protein localization sites in gram-negative bacteria.* Proteins, 1991. **11**: p. 95-110.

79. Horton, P. and K. Nakai, *Better prediction of protein cellular location sites with the k nearest neighbors classifier.* Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology 1997. **5**: p. 147-152.

80. Gardy, J.L., et al., *PSORTb v.2.0: Expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis* Bioinformatics, 2005. **21**(5): p. 617-623.

81. Guda, C. and S. Subramaniam, *pTARGET a new method for predicting protein subcellular localization in eukaryotes.* Bioinformatics, 2005. **21**(21): p. 3963-3969.

82. Garg, A., M. Bhasin, and G.P. Raghava, *Support vector machine-based method for subcellular localization of human proteins using amino acid compositions, their order and similarity search.* Journal of Biological Chemistry, 2005. **280**(15): p. 14427-14432.

83. Scott, M.S., D.Y. Thomas, and M.T. Hallett, *Predicting subcellular localization via protein motif co-occurrence.* Genome Research, 2004. **10A**: p. 1957-1966.

84. Xie, D., et al., *LOCSVMPSI: a web server for subcellular localization of eukaryotic proteins using SVM and profile of PSI-BLASt.* Nucleic Acids Research, 2005. **33**(Web Server Issue): p. W105-W110.

85. Bhasin, M. and G.P. Raghava, *ESLpred: SVM-based method for subcellular localization of eukaryotic proteins using dipeptide composition and PSI-BLAST.* Nucleic Acids Research, 2004. **32**(Web Server Issue): p. 414-419.

86. Lu, Z., et al., *Predicting Subcellular Localization of Proteins using Machine-Learned Classifiers.* Bioinformatics, 2004. **20**(4): p. 547-556.

87. Nair, R. and B. Rost, *Mimicking cellular sorting improves prediction of subcellular localization.* Journal of Molecular Biology, 2005. **348**(1): p. 85-100.

88. Hua, S. and Z. Sun, *Support vector machine approach for protein subcellular localization predition.* Bioinformatics, 2001. **17**(8): p. 721-728.

89. Atalay, V. and R. Cetin-Atalay, *Implicit motif distribution based hybrid computational kernel for sequence classification.* Bioinformatics, 2005. **21**(8): p. 1429-1436.

90. Sarda, D., et al., *pSLIP: SVM based protein subcellular localization prediction using multiple physicochemical properties.* BMC Bioinformatics, 2005. **6**(152).

91. Yu, C.-S., C.-J. Lin, and J.-K. Hwang, *Predicting subcellular localization of protein for Gram-negative bacteria by support vector machines based on n-peptide compositions.* Protein Science, 2004. **13**: p. 1402-1406.

92. Lanckriet, G.R.G., et al., *A statistical framework for genomic data fusion.* Bioinformatics, 2004. **20**(16): p. 2626-2635.

93. Lanckriet, G.R.G., et al. *Kernel-based Data Fusion and its application to protein function prediction in Yeast*. in *Systems Biology*. 2004.

94. Scholkopf, B., *Kernel Methods in Computational Biology (computational Molecular Biology)*. 2004, Cambridge: MIT Press.

95. Murzin, A.G., et al., *SCOP: a structural classification of proteins database for the invesitgation of sequences and structures.* Journal of Molecular Biology, 1995. **247**: p. 536-540.

96. Gribskov, M. and N.L. Robinson, *Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching.* Computer and Chemistry, 1996. **20**(1): p. 25-33.

97. Hanley, J.A. and B.J. McNeil, *The meaning and use of the area under a receiver operating characteristic (ROC) curve.* Radiology, 1982. **143**(1): p. 29-36.

98. Salzberg, S.L., *On comparing classifiers: Pitfalls to avoid and recommended approach.* Data Mining Knowledge Discovery, 1997. **1**: p. 317-328.

99. Pearson, W.R. and D.J. Lipman, *Improved tools for biological sequence comparison.* Proceedings of the National Academy of Sciences, 1988. **85**(8): p. 2444-2448.

100. Yu, C.-S., et al., *Fine-grained protein fold assignment by support vector machines using generalized n-peptide coding schemes and jury voting from multiple-parameter sets.* Proteins: Structure, Function and Genetics, 2003. **50**(4): p. 531-536.

101. Joachims, T., *Making Large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, ed. B. Burges, C.J. C., and A.J. Smola. 1999, Cambridge: MIT Press. 169-184.

102. Schaffer, A.A., et al., *Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements.* Nucleic Acid Research, 2001. **29**(14): p. 2994-3005.

103. Joachims, T., *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods, ed. B. Scholkopf, C. Burges, and A. Smola. 1999: MIT-Press.

104. Shah, A.R., et al., *Integrating Subcellular Location for Improving Machine Learning Models of Remote Homology Detection in Eukaryotic Organisms.* Computational Biology and Chemistry 2007. **31**(2): p. 138-142.

105. Kisman, D., et al., *tPatternHunter: gapped, fast and sensitive translated homology search.* Bioinformatics, 2005. **21**(4): p. 542-544.

106. Gorton, I., et al. *The MeDICi Integration Framework: A Platform for High Performance Data Streaming Applications.* in *7th IEEE/IFIP Working Conference on Software Architecture*. 2008. Vancouver, Canada: IEEE Computer Society.

107. Shannon, P.T., et al., *The Gaggle: an open-source software systems for integrating bioinformatics software and data sources.* BMC Bioinformatics, 2006. **7**: p. 176-188.