TECHNICAL REPORT

Report No. CSAI2004-01
Date: August 2004

Support Vector Machines with Profile-Based Kernels for Remote Protein Homology Detection

Steven Busuttil
John Abela
Gordon J. Pace

Department of Computer Science & A.I.
University of Malta
Msida MSD 06
MALTA

Tel: +356-2340 2519
Fax: +356-2132 0539
reports@cs.um.edu.mt
http://www.cs.um.edu.mt/~reports

Support Vector Machines with Profile-Based Kernels for Remote Protein Homology Detection

Steven Busuttil
University of Malta, Malta.
sbusu@cs.um.edu.mt

John Abela
University of Malta, Malta.
jabel@cs.um.edu.mt

Gordon J. Pace
University of Malta, Malta
gordon.pace@um.edu.mt

Abstract: *Two new techniques for remote protein homology detection particulary suited for sparse data are introduced. These methods are based on position specific scoring matrices or profiles and use a support vector machine (SVM) for discrimination. The performance on standard benchmarks outperforms previous non-discriminative techniques and is comparable to that of other SVM-based methods while giving distinct advantages.*

Support Vector Machines with Profile-Based Kernels for Remote Protein Homology Detection

Steven Busuttil
University of Malta, Malta.
sbusu@cs.um.edu.mt

John Abela
University of Malta, Malta.
jabel@cs.um.edu.mt

Gordon J. Pace
University of Malta, Malta
gordon.pace@um.edu.mt

Abstract: *Two new techniques for remote protein homology detection particulary suited for sparse data are introduced. These methods are based on position specific scoring matrices or profiles and use a support vector machine (SVM) for discrimination. The performance on standard benchmarks outperforms previous non-discriminative techniques and is comparable to that of other SVM-based methods while giving distinct advantages.*

1 Introduction

In recent years, advances in molecular biology like large-scale sequencing and the human genome project, have yielded an unprecedented amount of new protein sequences. The resulting sequences describe a protein in terms of the amino acids that constitute it and no structural or functional protein information is available at this stage. To a degree, this information can be inferred by finding a relationship (or homology) between new sequences and proteins for which structural properties are already known. This technique is known as protein homology detection. Traditional laboratory methods of protein homology detection rely on lengthy and expensive procedures like x-ray crystallography and nuclear magnetic resonance (NMR). Since using these procedures is unpractical for the amount of data available, researchers are increasingly relying on computational techniques to automate the process.

Over the past quarter of a century, an array of successively more powerful computational methods for detecting protein homologies have been developed. Early attempts involved comparing a new protein to a set of diverse proteins whose struc-

tural and functional details are already known — one at a time [NW70, SW81, Pea85, AGM+90]. This results in a set of similarity scores for each pair made up of the new protein and a known protein. Such a pairwise score indicates the amount by which a pair is related. This method gives good results for proteins that are closely related, however, remote homologies are not detected.

In the early 1990s a threefold accuracy improvement was achieved by collecting the information inherent in a number of related proteins [PKB+98]. This information is synthesised into a single model, to which new protein sequences are compared [GLE90, BCHM94, KBS+94]. This technique is more sensitive to faint protein similarities and consequently, gives better results. Furthermore, even better accuracy was obtained in the late 1990s by supplementing these models with information present in large protein databases [AMS+97, KBH98]. This process is done iteratively and makes a model increasingly accurate.

In 1999, Tommi Jaakkola, Mark Diekhans and David Haussler introduced a new remote protein homology detection technique known as SVM-Fisher [JDH99, JDH00]. This technique couples a previous statistical protein homology detection method, known as a profile hidden Markov model (HMM), with a discriminator (or classifier) from the area of machine learning, known as a support vector machine (SVM). In empirical tests this technique significantly outperforms previous (non-discriminative) methods for protein homology detection. SVM-Fisher initiated a new stage in the development of computational techniques for protein homology detection since it was the first time discriminators were used in the area. Consequently, Jaakkola *et al.* received the "Best Paper" award at the annual Intelligent Systems for Molecular Biology conference.

In this paper we propose two new methods for protein homology detection. As is done in the SVM-Fisher method, our methods build on an existing technique in the area of protein sequence analysis, known as a position specific scoring matrix or profile, and use a support vector machine as a discriminator.

2 Technical Background

Since our methods rely on profiles and support vector machines we will now give an outline of these core technologies. The interested reader is encouraged to check the provided references for more details.

Pos:	1	2	3	4	5
s1	A	A	C	B	C
s2	A	–	C	B	B
s3	C	A	–	B	B
s4	–	A	C	C	B

Pos:	1	2	3	4	5
A	0.50	0.75	0	0	0
B	0	0	0	0.75	0.75
C	0.25	0	0.75	0.25	0.25
–	0.25	0.25	0.25	0	0

Figure 1: A multiple alignment (left) and its corresponding profile (using an alphabet consisting of A, B, C and –).

2.1 Profiles

Profiles, or position specific scoring matrices (PSSMs) [GLE90], were introduced in bioinformatics by Gribskov *et al.* in 1990. Given a multiple sequence alignment, a profile specifies, for every column, the frequency that each amino acid appears in that column [Gus97]. As an example, consider the multiple alignment and its corresponding profile in Figure 1. Once a profile is available, a new sequence can be aligned to it to see how well it fits the profile. This is similar to asking how much a sequence is similar to the proteins from which the profile was derived.

In practice, the construction of a profile is somewhat more involved than what is shown in Figure 1. For instance, sequences in the multiple alignment are given a weight which describes their informational value. Consider the case where there is a large set of closely related sequences. This set carries more or less the same amount of information as a single sequence, however, its size may allow it to 'outweigh' a small number of more divergent sequences. In addition, when calculating the character position probabilities, a method is usually used that gives probabilities to unobserved characters based on their presumed association with the observed characters. This is especially useful when the multiple alignment contains only a few sequences or the proteins to be classified are remotely related.

As an optional last step, it is common practice to convert profile probabilities to log-odds ratios to increase the selectivity of the model. This involves taking the logarithm (base is not important) of the ratio between the available probabilities and the probability of the particular amino acid being found in nature (also called the background probability).

2.2 Support Vector Machines

Although the study of support vector machines (SVMs) was started in the late 1970s by Vladimir Vapnik [Vap79], it was not until the late 1990s that this work came to

fruition and SVMs started to receive increasing attention [Vap95]. To date, SVMs and related techniques have been greatly developed and applied to many areas, including bioinformatics. In most cases, the performance of SVMs either matches or is significantly better than that of competing methods.

In essence, support vector machines are supervised learning machines based on statistical learning theory. SVMs take a vector of real numbers as input and, based on previous experience, return its classification[1] [CS00]. Support vector machines are based on linear learning machines and learn the hyperplane that separates two classes with the maximum margin of separation. This optimal hyperplane has been proven to guarantee the best generalisation performance [Vap95]. Finding this hyperplane translates to a convex quadratic optimisation problem, which is formulated in such a way that all vectors appear inside a dot product.

Many times, the input data is not linearly separable in input space. SVMs handle this situation by substituting the said dot product with a function known as a kernel. A kernel takes two vectors and returns their dot product in some feature space. In this manner, SVMs find the optimal hyperplane in (a usually high dimensional) feature space. A commonly used kernel is the radial basis function (RBF) kernel:

$$k_r(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) \tag{1}$$

RBF kernels map the input space onto the surface of an infinite dimensional unit hypersphere, because by construction $\|\phi(\mathbf{x})\| = \sqrt{k_r(\mathbf{x}, \mathbf{x})} = 1$ for all $\mathbf{x} \in X$ [Her02]. The parameter $\sigma \in \mathbb{R}^+$ (the radius) controls the amount of smoothing of the decision surface in input space. Big values of σ lead to a very flat and smooth decision surface and conversely, small values lead to a very convoluted decision surface that fits tightly around the training points.

In real-world data it is not uncommon that some erroneous elements (noise) are present. An extension of the basic SVM algorithm is the soft margin classifier which allows some training examples to be misclassified, effectively making a trade-off between training accuracy and generalisation performance.

3 Problem and Related Work

In this section we introduce the problem and standard datasets used in the area. We then proceed to present the general method used and two prominent SVM-based computational methods for remote protein homology detection.

[1] Other uses of support vector machines apart from classification exist, including regression, where a real-valued function has to be learnt. However, this is beyond the scope of this report.

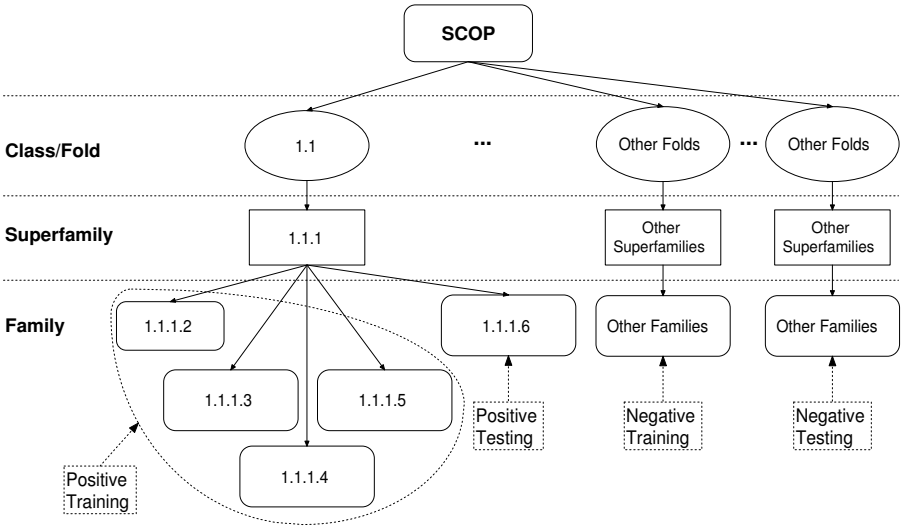

Figure 2: Illustration of a sample experiment from the SCOP dataset showing how sequences are chosen for training and testing. In this experiment, the task is to recognise the protein sequences from family 1.1.1.6 as being part of superfamily 1.1.1. The positive training set comprises the sequences from families 1.1.1.2, 1.1.1.3, 1.1.1.4 and 1.1.1.5. Negative examples are taken from different folds.

3.1 The Problem

Researchers simulate the problem of remote protein homology detection by withholding all members of a SCOP[2] (target) family and training with the remaining members of the SCOP superfamily. In addition to these (positive) training and testing sequences, negative examples are taken from folds that are different from the one that the target family belongs to. Figure 2 shows such a sample experiment, where the target family is 1.1.1.6. It is important to note that the SCOP database is constantly being updated and a number is assigned to every subsequent version. This means that there may be inconsistencies between one version and another, including unmatched family IDs.

The first such dataset was introduced in 1999 by Jaakkola *et al.*[3] [JDH99, JDH00] and is based on SCOP 1.37. This dataset contains 33 experiments and adds a number of homologous protein sequences retrieved from a database to the positive train-

[2]The Structural Classification of Proteins database [MBHC95]. `http://scop.mrc-lmb.cam.ac.uk/scop/`

[3]`http://www.cse.ucsc.edu/research/compbio/discriminative/`

ing set. In 2002 Liao and Noble introduced a new dataset[4] [LS02, LS03] based on SCOP 1.53, containing 54 experiments. This dataset is similar in principle to the one by Jaakkola *et al.*, however the positive training set does not include added homologues. As such, this makes the recognition tasks to perform more difficult.

3.2 Methods

Protein sequences, which are variable-length character strings, must be converted to fixed-length numerical vectors for input to an SVM. Therefore, all proposed solutions follow a simple method:

1. Vectorise the training and testing sequences.

2. Train the SVM on the training vectors using a standard kernel (usually the RBF kernel).

3. Test the performance of the SVM on a dataset.

The first (vectorisation) step is the crucial part in this method. In particular, note that it is actually part of the kernel, since it is an explicit mapping from input space to some feature space. By the closure of kernels, a function that maps its input to feature space coupled with a standard kernel gives a valid kernel.

As would be expected, the main difference between methods is the vectorisation step. Most methods aim to exploit prior knowledge of the protein homology detection domain to design a good mapping from input space to feature space. This, coupled with a standard kernel, gives a similarity measure between pairs of inputs in feature space that is then used by the SVM for discrimination.

3.2.1 The SVM-Fisher Method

The SVM-Fisher method [JDH99, JDH00] was introduced in 1999 by Jaakkola *et al.* This method couples iterative profile hidden Markov models (HMMs) with an SVM in such a way as to define a kernel. Initially, a number of HMMs are trained on different subsets of the positive training set using a standard training algorithm. In general, these subsets contain related proteins and other homologues pulled from a large protein database. After the HMM training, a vector known as the Fisher score

[4] `http://www1.cs.columbia.edu/compbio/svm-pairwise/`

is computed for any sequence with respect to each HMM. The Fisher score for a training sequence s with respect to the HMM H_i is the vector

$$\mathbf{u}_{si} = \frac{\delta \log \Pr(s|H_i(\theta))}{\delta\theta}$$

That is, the gradient of the log-likelihood of the sequence s with respect to the parameters (θ) of the HMM H_i. These gradients can be computed for every HMM parameter, however, the SVM-Fisher method uses only those that correspond to emission probabilities in the match states.

The vectors for sequence s corresponding to each model (which are of the same length) are subsequently combined into one vector (denoted $\mathbf{u}_s$) by taking their average. Effectively, a (combined) vector for sequence s summarises how different s is from a typical member of the superfamily the HMMs where trained on.

The last step in the SVM-Fisher method is to train an SVM using an RBF kernel. The resulting combined kernel is the dot product in feature space that is the image of the composition of two feature maps:

$$s \longmapsto \mathbf{u}_s \longmapsto \phi(\mathbf{u}_s)$$

where ϕ is the feature map associated with the RBF kernel.

SVM-Fisher yields results that significantly outperform previous (non-discriminative) state-of-the-art protein homology detection systems.

3.2.2 The SVM-pairwise Method

In 2002, Liao and Noble introduced a simple but effective remote protein homology detection method, named SVM-pairwise [LS02, LS03]. This method generates a kernel from scores produced by a pairwise sequence comparison algorithm (like the Smith-Waterman pairwise alignment algorithm [SW81]).

Let m be the number of training examples and n be the number of all (training and testing) examples. SVM-pairwise builds an $n \times m$ matrix of pairwise scores with element j of row i corresponding to the pairwise score of protein i when compared to protein j from the training set. Therefore, in the SVM-pairwise method, row i is a vector representation of protein i. Note that in this vectorisation step, all (negative and positive) protein sequences in the training set are used. As is done in other methods, these vectors are then combined with a standard RBF kernel and an SVM is trained (see Figure 3).

SVM-pairwise differs considerably from the SVM-Fisher method — in particular, both positive and negative examples are used in the vectorisation step. On the dataset

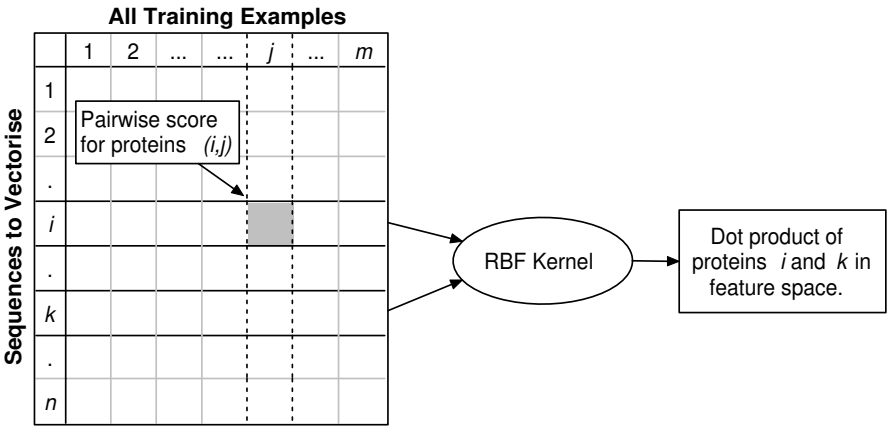

Figure 3: Illustration of the SVM-pairwise method.

by Liao and Noble, this method performs significantly better than the SVM-Fisher method. Running SVM-pairwise on the Jaakkola *et al.* dataset is not practical due to its computational complexity.

4 Solutions Proposed

Our aim is to vectorise protein sequences in such a way as to be accepted for input to an SVM. The vectors should incorporate prior knowledge from the area so that the discrimination of related and non-related proteins be made easier. For this purpose, we have developed two main methods: ProfGram and ProfBlock, which are described below.

4.1 Approach Overview

The following is a high level description of the steps involved in our vectorisation method (also illustrated in Figure 4):

1. Compute a multiple alignment of the positive training set using ClustalW, a popular multiple alignment program by Thompson *et al.* [THG94]. This is done so that conserved parts (which are typical of a superfamily) are emphasised.

2. Build a profile from the multiple alignment using the Henikoff and Henikoff position-based method. Effectively, this results in a model that encapsulates

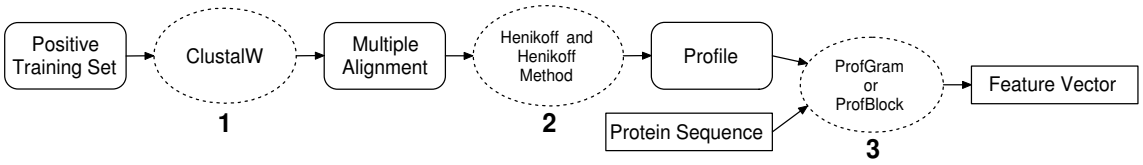

Figure 4: Overview of our protein sequence vectorisation technique.

what makes a protein part of the target superfamily[5].

3. Vectorise each sequence in the (positive and negative) training and testing sets by comparing it with the profile, using one of our methods: ProfGram or ProfBlock. This results in a vector that measures how similar the sequence is to the profile.

The construction of a profile from a multiple alignment (step 2) is a two step process. First, we use a modified version of the position-based sequence weights method by Henikoff and Henikoff [HH94]. In this method, sequence weights are based on the diversity observed at each position in the multiple alignment. Then the character position specific probabilities are calculated using the pseudo-count method by Henikoff and Henikoff [HH96], which has been shown to outperform similar methods in extensive empirical tests. For the vectorisation of a protein sequence (step 3) we propose two new methods, ProfGram and ProfBlock, which are described in detail in Section 4.2 and Section 4.3 respectively.

4.2 ProfGram

The ProfGram vectorisation method maps protein sequences to the feature space indexed by all possible (contiguous) profile subparts of length k, which we shall call profile k-grams (k is chosen empirically). For this method, the profile is converted to log-odds scores.

Each of these profile k-grams is slid across the sequence to be vectorised, taking the score of aligning each part of the sequence with the k-gram. All the scores for a particular k-gram are added together, representing how much the sequence as a whole matches the k-gram. The score for each k-gram is stored as an element of the feature vector. This process is presented more formally below and is illustrated in Figure 5.

[5]Recall that the positive training set is from the same superfamily as the positive testing set. Consequently, our problem is equivalent to discovering a new family of a known superfamily.

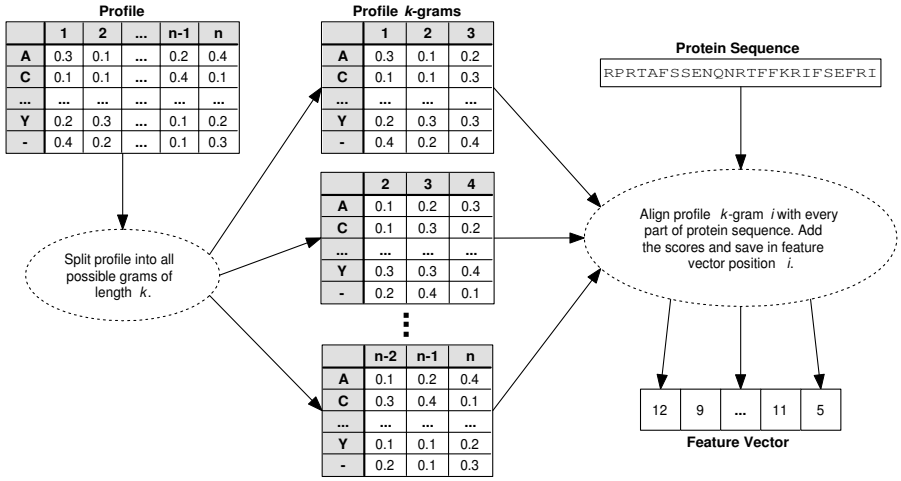

Figure 5: Illustration of the ProfGram vectorisation process for $k = 3$.

Let s be a sequence of length m and let the character in position j of s be denoted by s_j. Let n be the length of the profile and $k \leq \min(m, n)$ be the predefined length of the profile grams. Let $w(i, s_j)$ be the score for character s_j appearing in position i of the profile and $\mathbf{v}$ be the feature vector of length $n - k + 1$. For a sequence s, an element of $\mathbf{v}$ (denoted v_h) is calculated as follows:

$$v_h \stackrel{df}{=} \sum_{i=1}^{m-k+1} \sum_{j=0}^{k-1} w(h + j, s_{i+j}) \tag{2}$$

Equation 2 is thus used to fill every position in $\mathbf{v}$ (for each profile k-gram)[6]. An empirical test was carried out to determine a good value for k. It seems that for some experiments a small width is better and for others a wider profile gram gives better results. We finally opted to choose $k = 4$ which gives a performance that is consistent throughout, although not always optimal.

4.3 ProfBlock

ProfBlock is a more sophisticated vectorisation method that uses the expected matching length of a sequence to every part of the profile. For this method the profile is not converted to log-odds scores, but is left at the stage where it contains raw probabilities.

[6]It is important to keep in mind that adding log-odds scores is equivalent to multiplying the underlying probability ratios, otherwise Equation 2 would not make mathematical sense.

4.3.1 The Expected Matching Length

Central to the ProfBlock method is a technique known as the expected matching length, defined below.

Definition 4.1 (Expected Matching Length). Given a profile p of length n, containing position-specific character probabilities and a sequence s of length m, the *expected matching length* of s to p starting at profile position $i \leq n$ and sequence position $j \leq m$, is denoted by e_{ij}. Let $p(i, c)$ be the profile probability of character c in position i, s_j be the character in position j of s and l be the maximum possible matching length, $l = \min(n - i + 1, m - j + 1)$. e_{ij} is defined to be the expected matching length as per standard probability theory:

$$e_{ij} \stackrel{df}{=} \sum_{k=1}^{l} k \left(\prod_{h=0}^{k-1} p(i+h, s_{j+h}) \right) (1 - p(i+k, s_{j+k})) \tag{3}$$

That is, a summation of the probabilities of matching *exactly* k positions, multiplied by k. As an example, consider the sequence CAD and its expected matching length with profile p of length $n \geq 3$, starting at the first position of both the sequence and profile (hence $l = 3$):

$$\begin{aligned} e_{11} &= 1 \times p(1, C) \times (1 - p(2, A)) \\ &+ 2 \times p(1, C) \times p(2, A) \times (1 - p(3, D)) \\ &+ 3 \times p(1, C) \times p(2, A) \times p(3, D) \times (1 - 0) \end{aligned}$$

4.3.2 Approximation of the Expected Matching Length

The expected matching length is a powerful way of finding the amount by which a part of a sequence matches a part of the profile, however it has the drawback of being very computational expensive. To alleviate this problem, we use a technique that enables us to compute only a small number of the iterations (summations) of Equation 3 while giving approximately the same result.

For notational clarity we will subsequently be ignoring the different profile and sequence positions and denote a profile probability as simply p_i, meaning the probability of the match of the ith sequence and profile positions pair. Once again, let l be the maximum possible matching length and let $k < l$. Equation 3 can be opened up by

splitting the summation into two parts (for the first profile and sequence positions):

$$\begin{aligned} e_{11} &= \sum_{i=1}^{k} i \left(\prod_{j=1}^{i} p_j \right) (1 - p_{i+1}) \\ &+ \sum_{i=k+1}^{l} i \left(\prod_{j=1}^{i} p_j \right) (1 - p_{i+1}) \end{aligned} \tag{4}$$

Since the probabilities product ($\prod_{j=1}^{i} p_j$) approaches zero as i increases, the second term of Equation 4 becomes progressively less significant as i increases. If e_{ij}^k is the summation of the first k terms (the first term in Equation 4), we would like to determine a value of k such that e_{ij} lies within the interval $[e_{ij}^k, e_{ij}^k + \varepsilon]$ where ε is a nonnegative real number. This would allow us to approximate e_{ij} by calculating only the first k terms. Moreover, we would like to determine when to stop based on information from the first k terms. Lemma 4.1 relates the values of the first k terms to the magnitude of the remaining terms:

Lemma 4.1.

$$l(l-k) \prod_{j=1}^{k} p_j \geq \sum_{i=k+1}^{l} i \left(\prod_{j=1}^{i} p_j \right) (1 - p_{i+1})$$

Proof

$$\begin{aligned} l(l-k) \prod_{j=1}^{k} p_j &= l \left(\sum_{i=k+1}^{l} 1 \right) \prod_{j=1}^{k} p_j \\ &\quad \{\text{since } 0 \geq p_i \geq 1 \text{ then } 1 \geq (\textstyle\prod_i p_i)\,(1 - p_j), \text{ therefore}\} \\ &\geq l \left(\sum_{i=k+1}^{l} \left(\prod_{j=k+1}^{i} p_j \right) (1 - p_{i+1}) \right) \prod_{j=1}^{k} p_j \\ &\quad \{\text{since } l \geq i \text{ then}\} \\ &\geq \left(\sum_{i=k+1}^{l} i \left(\prod_{j=k+1}^{i} p_j \right) (1 - p_{i+1}) \right) \prod_{j=1}^{k} p_j \\ &= \sum_{i=k+1}^{l} i \left(\prod_{j=1}^{i} p_j \right) (1 - p_{i+1}) \end{aligned}$$

Therefore, if the calculation of the expected matching length is stopped when

$$l(l-k) \prod_{j=1}^{k} p_j \leq \varepsilon \tag{5}$$

ε	Time (%)	Average Precision Loss
1	8%	4.8×10^{-3}
0.1	10%	1.2×10^{-3}
0.01	12%	1.1×10^{-4}
0.001	14%	1.2×10^{-5}
0.0001	15%	1.2×10^{-6}
0 (no approximation)	100%	0

Table 1: Performance of the ProfBlock method for different settings of ε.

Lemma 4.1 guarantees that the summation of all the other terms in the series is bound from above by ε and hence, e_{ij} cannot increase by more than ε. This reduces the computation required to calculate (an approximation of) e_{ij} considerably. Empirical analysis indicated that taking ε to be 0.001 gives a good time to precision loss trade-off (see Table 1).

4.3.3 Computing the Feature Vector

The expected matching length of a sequence of length m to a profile of length n, starting at profile position i and sequence position j, is denoted by e_{ij}. Scoring a whole sequence with a part of the profile starting at position i, results in the vector

$$\mathbf{t}_i = (e_{i1}, e_{i2}, \ldots, e_{im})'$$

The vector $\mathbf{t}_i$ represents how much the different positions of a sequence match the profile starting at position i. A function $g : \mathbb{R}^m \rightarrow \mathbb{R}$ is needed that takes a vector $\mathbf{t}_i$ as its argument and combines its elements to return a single real number. This number represents how much the sequence as a whole matches the profile starting at position i. The feature vector generated for a sequence is thus:

$$\mathbf{v} = (g(\mathbf{t}_1), g(\mathbf{t}_2), \ldots, g(\mathbf{t}_n))'$$

That is, a vector with elements representing how much a sequence matches the different positions of a profile. This process is illustrated in Figure 6.

The ProfBlock scoring function was chosen after an empirical analysis of several candidate functions. It turns out that positive examples have longer expected matching lengths. The scoring function that was found to give the best performance is the sum of squares since it strengthens larger values present in the expected matching length vectors $\mathbf{t}_i$:

$$g(\mathbf{t}_i) = \frac{\sum_{j=1}^{m} e_{ij}^2}{m}$$

		Profile			
		1	**2**	...	**n**
Sequence	**1**	$(e_{11},$	$(e_{21},$	...	$(e_{n1},$
	2	$e_{12},$	$e_{22},$	...	$e_{n2},$
	⋮	⋮	⋮	⋱	⋮
	m	$e_{1m})$	$e_{2m})$	...	$e_{nm})$
		$= \mathbf{t}_1$	$= \mathbf{t}_2$	...	$= \mathbf{t}_n$
Feature Vector		$\downarrow$ $(g(\mathbf{t}_1),$	$\downarrow$ $g(\mathbf{t}_2),$	... ...	$\downarrow$ $g(\mathbf{t}_n))'$

Figure 6: Illustration of the ProfBlock vectorisation method.

Note that the sum of squares is divided by the length of the sequence, m, as a means to normalise the values between sequences of different lengths. If this is not done, long sequences would have values which are bigger than shorter ones simply due to their length.

4.4 Time Complexity Analysis

In analysing the time complexity of our vectorisation methods, we will be taking the profile as given[7]. Let n be the length of the longest profile and m be the length of the longest sequence. ProfGram compares each part of a sequence with each part of a profile, for up to k positions each time (k is the width of the profile k-grams). Therefore, the time complexity for ProfGram is $O(kmn)$. If k is chosen to be equal to the maximum length possible, then the time required will increase, but in practice k is usually a small number.

On the other hand, ProfBlock compares each position of a sequence (m possibilities) with each position of a profile (n possibilities), for up to the length of the sequence or profile ($\min(m, n)$ possibilities). The time complexity for the ProfBlock vectorisation method is therefore $O(\min(m, n)mn)$, making ProfBlock more complex than ProfGram by a factor of $\min(m, n)$. However, through our ProfBlock expected matching length approximation technique, we rarely compare a sequence and profile position for up to the maximum length possible. In fact, the length compared is usually just a small fraction of the full length. Empirical evidence indicates that this reduces the $\min(m, n)$ factor considerably (see Table 1).

[7]The calculation of a profile from a multiple alignment is a preprocessing step that is done just once for the dataset, unlike ProfGram and ProfBlock that have to be performed for each sequence.

5 Experiments and Results

Our remote protein homology detection method involves vectorising protein sequences using ProfGram or ProfBlock and then training and testing an SVM on the resulting vectors. In this section, we describe how the data is scaled and the SVM model selection prior to the actual SVM learning. Note that more detailed results can be found in [Bus04].

5.1 Scaling

Scaling vectors before applying the SVM algorithm is very important. This is mainly done to avoid vector elements in greater numeric ranges dominating those in smaller numeric ranges and to avoid numerical difficulties during the kernel calculations. Testing and training data must obviously be scaled using the same method.

For our experiments, we normalise each vector to Euclidean length 1 (the unit vector). Under this normalisation scheme, the normalised version of the 2-dimensional vector $\mathbf{v} = (v_1, v_2)'$, denoted $\hat{\mathbf{v}}$, is generated as follows

$$\hat{\mathbf{v}} = \left(\frac{v_1}{\|\mathbf{v}\|}, \frac{v_2}{\|\mathbf{v}\|} \right)'$$

where $\|\mathbf{v}\|$ is the norm of $\mathbf{v}$.

5.2 Scoring SVM Results

To evaluate the predictions produced by an SVM, we employ two different methods that are commonly used in the machine learning community: the median rate of false positives and the receiver operating characteristic (ROC). At this stage, it is important to note that in practice, the output of an SVM is a real number in the range $[-1, +1]$.

5.2.1 Median Rate of False Positives

The median rate of false positives (median RFP) is the fraction of negative examples that score as high or better than the median-scoring positive example. The median RFP is bounded by 0 and 1. The smaller the value, the better the performance.

5.2.2 Receiver Operating Characteristic

The receiver operating characteristic (ROC) [GR96] is a sophisticated technique that is used to evaluate the results of a prediction. A ROC curve is a graphical plot of the number of true positives as a function of the number of false positives for varying classification thresholds. The area under the ROC curve is called the ROC score and is commonly used as a summary statistic. The area under the curve is usually approximated by taking the area of the rectangles between data points, and clearly, is bounded by 0 and 1 (inclusive). A higher ROC score indicates better performance.

5.3 Model Selection

Through an empirical analysis it was found that our methods perform best when coupled with an RBF kernel (see Equation 1). Using this setup, two SVM parameters have to be chosen prior to training: σ, the RBF kernel smoothing parameter, and C the (soft margin) SVM training error to generalisation performance trade-off parameter. This is known as model selection.

The goal of model selection is to identify a good C and σ pair such that the classifier accurately predicts unknown data (testing data). Choosing a small value for C and a large value for σ results in a simple decision surface and vice-versa. Even if the data can be separated without error (and this is always possible with the RBF kernel), better results may be obtained with a simple decision surface since this can avoid overfitting.

A common technique used in the machine learning community for model selection is cross-validation. In k-fold cross-validation, the training set is split into k subsets (or folds) of equal size. For a particular configuration (in our case the C and σ parameters pair), one subset is tested using the classifier trained on the remaining $(k-1)$ subsets and scored. This is repeated for every subset and the average of the scores is taken. This average score is an approximation of the performance of the classifier on testing data for a particular configuration.

To find the best pair of C and σ over some ranges, a grid search using cross-validation is employed. This involves scoring each pair using cross-validation and the pair with the best score is chosen (see Table 2). In our experiments, the grid search performed is very coarse. Ideally, when good values for a parameter pair is found, a finer grid search is made in their vicinity. This, however, was not possible due to the size of the datasets and the amount of time required.

	C_1	C_2	...	C_n
σ_1	Cross-validation score using parameters C_1 and σ_1 — $s(C_1, \sigma_1)$.	$s(C_2, \sigma_1)$	...	$s(C_n, \sigma_1)$
σ_2	$s(C_1, \sigma_2)$	$s(C_2, \sigma_2)$	...	$s(C_n, \sigma_2)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
σ_m	$s(C_1, \sigma_m)$	$s(C_2, \sigma_m)$	...	$s(C_n, \sigma_m)$

Table 2: The cross-validation grid search.

5.4 Results

For our experiments, we use Joachims' implementation of a support vector machine, SVM*light*[8] [Joa99]. This implementation allows us to specify cost models for different classes of the classification task. Moreover, it can handle training sets containing many thousands of examples. The results obtained on the Liao and Noble dataset and the Jaakkola *et al.* dataset are presented in this section. Comparisons of our work to other methods are also included. We will be referring to our methods as SVM-ProfGram and SVM-ProfBlock to highlight the fact that we are running an SVM on sequences vectorised by a certain technique.

5.4.1 Liao and Noble Dataset

Our methods for remote protein homology detection were designed for the Liao and Noble dataset. The SVM parameters grid search was iterated over the following values: $C = \{2^{-3}, 2^{-2}, \ldots, 2^5\}$ and $\sigma = \{2^{-3}, 2^{-2}, \ldots, 2^7\}$. For this dataset, the SVM was trained with different cost models for the two classes being classified since negative examples greatly outnumber positive examples in the training set. Positive training examples were given a misclassification cost factor of $\frac{N_-}{N_+}$ where N_- and N_+ are the number of negative and positive training examples respectively. The cost factor for misclassifying negative examples was left to the default of 1. The ROC results obtained by our methods and other competing methods on this dataset are presented in Table 3 and summarised in Figure 7[9].

[8] `http://svmlight.joachims.org/`

[9] The results of competing methods were taken from `http://www.cs.columbia.edu/compbio/svm-pairwise`.

Liao and Noble Dataset — ROC Scores															
No.	Family	PG	PB	PW	FS	SM	BL	No.	Family	PG	PB	PW	FS	SM	BL
1	1.27.1.1	0.977	0.951	0.971	0.511	0.530	0.545	28	2.9.1.4	0.957	0.979	0.918	0.431	0.485	0.365
2	1.27.1.2	0.963	0.969	0.918	0.629	0.433	0.388	29	3.1.8.1	0.958	0.972	0.963	0.323	0.426	0.580
3	1.36.1.2	0.960	0.898	0.935	0.845	0.879	0.434	30	3.1.8.3	0.919	0.950	0.931	0.445	0.799	0.514
4	1.36.1.5	0.905	0.709	0.976	0.641	0.294	0.443	31	3.2.1.2	0.867	0.850	0.838	0.371	0.348	0.532
5	1.4.1.1	0.997	0.988	0.968	0.708	0.490	0.453	32	3.2.1.3	0.834	0.821	0.898	0.611	0.530	0.460
6	1.4.1.2	0.926	0.959	0.814	0.795	0.744	0.315	33	3.2.1.4	0.908	0.891	0.964	0.847	0.686	0.821
7	1.4.1.3	0.971	0.961	0.944	0.635	0.754	0.588	34	3.2.1.5	0.897	0.968	0.932	0.597	0.594	0.506
8	1.41.1.2	0.962	0.957	0.999	0.956	0.992	0.867	35	3.2.1.6	0.838	0.849	0.912	0.624	0.797	0.769
9	1.41.1.5	0.976	0.900	0.998	0.935	0.893	0.925	36	3.2.1.7	0.905	0.920	0.909	0.536	0.366	0.773
10	1.45.1.2	0.315	0.672	0.971	0.547	0.522	0.481	37	3.3.1.2	0.859	0.867	0.937	0.733	0.722	0.736
11	2.1.1.1	0.754	0.839	0.978	0.840	0.878	0.491	38	3.3.1.5	0.848	0.915	0.917	0.448	0.568	0.806
12	2.1.1.2	0.958	0.958	0.994	0.756	0.875	0.575	39	3.32.1.1	0.876	0.900	0.946	0.777	0.852	0.817
13	2.1.1.3	0.890	0.967	0.985	0.844	0.755	0.531	40	3.32.1.11	0.916	0.965	0.880	0.899	0.969	0.667
14	2.1.1.4	0.864	0.886	0.974	0.876	0.976	0.561	41	3.32.1.13	0.752	0.821	0.836	0.727	0.760	0.278
15	2.1.1.5	0.785	0.732	0.832	0.647	0.602	0.488	42	3.32.1.8	0.857	0.918	0.901	0.759	0.867	0.688
16	2.28.1.1	0.764	0.658	0.815	0.490	0.392	0.639	43	3.42.1.1	0.836	0.861	0.886	0.687	0.790	0.784
17	2.28.1.3	0.733	0.829	0.829	0.596	0.375	0.476	44	3.42.1.5	0.721	0.784	0.811	0.586	0.523	0.562
18	2.38.4.1	0.789	0.752	0.697	0.501	0.282	0.423	45	3.42.1.8	0.885	0.916	0.760	0.425	0.463	0.396
19	2.38.4.3	0.746	0.816	0.707	0.419	0.385	0.436	46	7.3.10.1	0.950	0.981	0.986	0.898	0.507	0.604
20	2.38.4.5	0.866	0.874	0.877	0.539	0.417	0.573	47	7.3.5.2	0.944	0.992	0.996	0.850	0.698	0.886
21	2.44.1.2	0.219	0.444	0.146	0.533	0.444	0.525	48	7.3.6.1	0.979	0.975	0.998	0.985	0.999	1.000
22	2.5.1.1	0.896	0.894	0.925	0.680	0.752	0.696	49	7.3.6.2	0.968	0.978	0.994	0.955	0.826	0.837
23	2.5.1.3	0.878	0.907	0.896	0.669	0.803	0.764	50	7.3.6.4	0.992	0.989	0.992	0.864	0.935	0.838
24	2.52.1.2	0.694	0.716	0.643	0.472	0.541	0.232	51	7.39.1.2	0.978	0.988	0.928	0.713	0.378	0.621
25	2.56.1.2	0.823	0.851	0.844	0.612	0.492	0.511	52	7.39.1.3	0.783	0.871	0.990	0.820	0.902	0.406
26	2.9.1.2	0.940	0.944	0.874	0.485	0.582	0.554	53	7.41.5.1	0.913	0.844	0.791	0.798	0.649	0.471
27	2.9.1.3	0.998	0.995	0.970	0.655	0.745	0.311	54	7.41.5.2	0.422	0.686	0.943	0.979	0.863	0.488

Table 3: ROC scores obtained on the Liao and Noble dataset by various methods. PG = our ProfGram vectorisation method with an SVM; PB = our ProfBlock vectorisation method with an SVM; PW = Liao and Noble's SVM-pairwise [LS02, LS03]; FS = Jaakkola *et al.*'s SVM-Fisher method [JDH99, JDH00]; SM = Karplus *et al.*'s profile HMM-based SAM [KBH98]; BL = Altschul *et al.*'s PSI-BLAST [AMS+97].

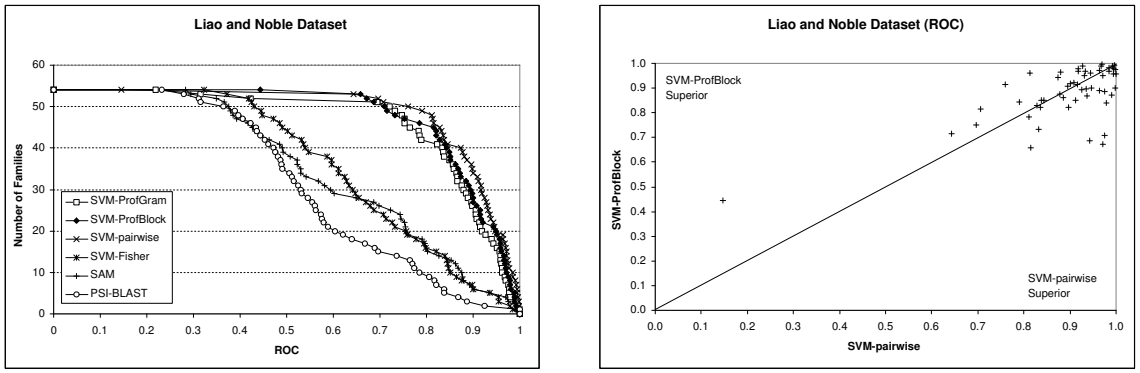

Figure 7: Comparison of several methods (left) and a family-by-family comparison of SVM-ProfBlock with SVM-pairwise on the Liao and Noble dataset.

5.4.2 Jaakkola *et al.* Dataset

ProfGram and ProfBlock were not designed for the Jaakkola *et al.* dataset. This is because the positive training set contains a lot of proteins that are related to particular superfamily protein sequences. These protein sequences may be too distantly related between themselves to generate a good profile. We believe that with some minor modifications to handle this situation, the performance of our methods can be improved. Unfortunately, it was not possible to run our methods on the Immunoglobulin (2.1.1.x) families (5 experiments) because the size of the positive training sets prevented us from producing the corresponding multiple alignments using ClustalW. Consequently, we will be ignoring these families for the comparisons.

For this dataset, the two SVM parameters were iterated over the following values: $C = \{2^1, 2^2, \ldots, 2^5\}$ and $\sigma = \{2^6, 2^7, \ldots, 2^{12}\}$. There was no need to train the SVM with different cost models since the positive and negative training sets are roughly balanced. The median RFP results obtained by our methods and other competing methods on this dataset are presented in Table 4 and summarised in Figure 8[10].

5.5 Analysis of Results

Our ProfGram and ProfBlock vectorisation methods are designed take advantage of the fact that related proteins share some common conserved areas. The profile itself emphasises these important features of a collection of related proteins. In essence,

[10]The results of competing methods were taken from [JDH00].

Jaakkola *et al.* Dataset — Median RFP													
No.	Family	PG	PB	FS	SM	BL	No.	Family	PG	PB	FS	SM	BL
1	1.1.1.2	0.562	0.262	0.364	0.450	0.342	18	2.5.1.3	0.180	0.125	0.002	0.003	0.116
2	1.25.1.1	0.016	0.002	0.035	0.446	0.397	19	2.8.1.2	0.064	0.079	0.133	0.088	0.391
3	1.25.1.2	0.002	0.009	0.002	0.109	0.114	20	2.8.1.4	0.025	0.019	0.066	0.204	0.630
4	1.25.1.3	0.026	0.049	0.004	0.289	0.440	21	3.1.1.1	0.052	0.020	0.000	0.007	0.851
5	1.34.1.4	0.002	0.002	0.000	0.000	0.000	22	3.1.1.3	0.144	0.055	0.008	0.009	0.338
6	1.34.1.5	0.001	0.000	0.000	0.000	0.000	23	3.1.1.5	0.024	0.024	0.031	0.110	0.426
7	2.1.1.1	–	–	0.000	0.000	0.000	24	3.19.1.1	0.055	0.066	0.008	0.019	0.004
8	2.1.1.2	–	–	0.000	0.000	0.000	25	3.19.1.3	0.080	0.154	0.002	0.009	0.102
9	2.1.1.3	–	–	0.000	0.000	0.006	26	3.19.1.4	0.229	0.064	0.002	0.001	0.049
10	2.1.1.4	–	–	0.000	0.000	0.004	27	3.19.1.5	0.119	0.026	0.002	0.024	0.330
11	2.1.1.5	–	–	0.073	0.178	0.329	28	3.25.1.1	0.142	0.110	0.005	0.015	0.299
12	2.19.1.1	0.740	0.211	0.083	0.278	0.298	29	3.25.1.3	0.092	0.174	0.000	0.007	0.330
13	2.31.1.1	0.111	0.137	0.000	0.000	0.002	30	3.33.1.1	0.018	0.002	0.000	0.000	0.000
14	2.31.1.2	0.157	0.152	0.000	0.000	0.000	31	3.33.1.5	0.212	0.270	0.238	0.273	0.201
15	2.34.1.1	0.773	0.103	0.003	0.012	0.108	32	3.50.1.7	0.087	0.086	0.000	0.000	0.053
16	2.41.1.1	0.618	0.032	0.051	0.165	0.293	33	3.73.1.2	0.644	0.072	0.026	0.007	0.433
17	2.5.1.1	0.278	0.095	0.013	0.039	0.049							

Table 4: Median RFP results obtained on the Jaakkola *et al.* dataset by various methods. PG = our ProfGram vectorisation method with an SVM; PB = our ProfBlock vectorisation method with an SVM; FS = Jaakkola *et al.*'s SVM-Fisher method [JDH99, JDH00]; SM = Karplus *et al.*'s profile HMM-based SAM [KBH98]; BL = Altschul *et al.*'s BLAST [AGM+90].

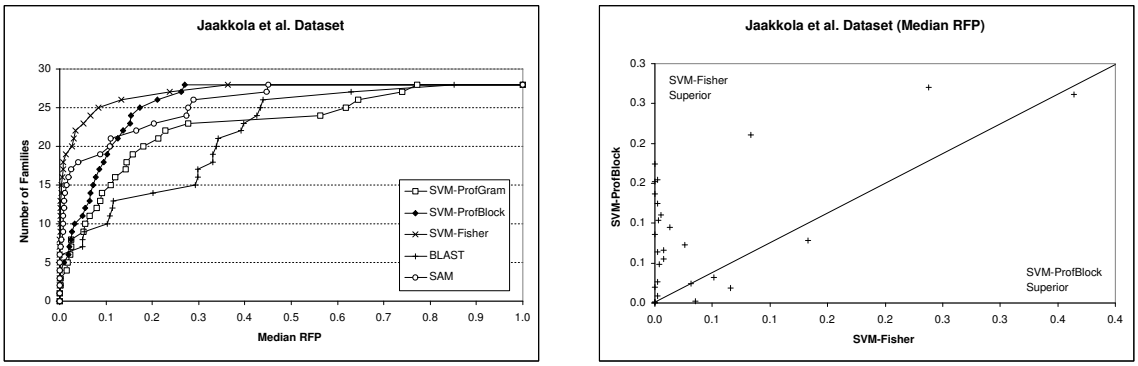

Figure 8: Comparison of several methods (left) and a family-by-family comparison of SVM-ProfBlock with SVM-Fisher (right) on the Jaakkola *et al.* dataset.

a profile can be seen as describing a typical protein in the superfamily from which the proteins were taken. Both our methods compare a sequence with each part of a profile. If a sequence has a high similarity with part i of the profile, it receives a high score for element i of the feature vector. ProfGram uses a simple scoring mechanism, where the profile contains log-odd scores. On the other hand, ProfBlock uses a more sophisticated technique based on the expected matching length of a sequence with a part of the profile. In this case, the profile contains the probabilities derived from the multiple alignment of related proteins. A vector created with one of our methods should therefore represent how much a sequence matches a profile as a whole.

The support vector machine has to determine (through training) which features make a vector belong to the positive class and vice-versa. As described in Section 2.2, SVMs are particularly good at choosing the hyperplane that best separates these two classes. Moreover, the RBF kernel maps the vectors to a space where the important features are emphasised, making the resulting feature vectors linearly separable (possibly with some error).

As expected, SVM-ProfBlock performs better than SVM-ProfGram in all of the experiments, even though the difference is not very pronounced. Moreover, the performance of our methods is always better than traditional, non-SVM methods and comparable to the state-of-the art SVM methods. What follows in this section is an analysis of the results achieved with our methods on the Liao and Noble dataset and the Jaakkola *et al.* dataset. It is important to keep in mind that our methods were designed for the Liao and Noble dataset. In fact, our methods' performance on this dataset is very satisfactory. On the other hand, our methods could not be run on five experiments of the Jaakkola *et al.* dataset, due to the huge amount of protein sequences in their positive training sets.

5.5.1 Performance on the Liao and Noble Dataset

Of particular interest is the performance of the SVM-ProfBlock method on the Liao and Noble Dataset. For almost half of the families in the dataset our method outperforms SVM-pairwise, which is the best performing method on this dataset (recall that the dataset was created by the authors of this method). In general, however, SVM-pairwise performs slightly better than SVM-ProfBlock.

The time complexity of the vectorisation step of SVM-pairwise for a single sequence is $O(m^2 l)$ where m is the length of the sequence and l is the amount of sequences in the training set. Recall that the time complexity of ProfBlock for the vectorisation of a sequence is $O(\min(m, n)mn)$, where n is the length of the profile. Typically, $l \gg \max(m, n)$, which makes our method more efficient. Moreover, through our

expected matching length approximation technique the time required in practice is reduced further (see Table 1 for an empirical result).

Our methods are similar in spirit to SVM-Fisher since both build a model (a profile and an HMM respectively) from the positive training set and then vectorise a sequence in relation to this model. On this dataset, both SVM-ProfBlock and SVM-ProfGram significantly outperform the SVM-Fisher method. This may be because the HMMs used in the SVM-Fisher method are undertrained due to limited training examples. The fact that our methods perform so well on limited positive training examples is commendable.

5.5.2 Performance on the Jaakkola *et al.* Dataset

The performance of SVM-ProfGram and SVM-ProfBlock is also satisfactory on the Jaakkola *et al.* dataset, although not as good as on the Liao and Noble dataset. However, it is important to keep in mind that five families (2.1.1.x) of the dataset were left out from these benchmarks, therefore the results here are somewhat incomplete.

Once again, in this dataset, SVM-ProfBlock achieves better classification on almost half of the experiments that were performed than the state-of-the-art method, SVM-Fisher. However, the difference between our method and SVM-Fisher is much more pronounced than that with SVM-pairwise on the Liao and Noble dataset (recall that this dataset was created by the authors of SVM-Fisher).

The time complexity of the vectorisation step of SVM-Fisher for a single sequence is $O(mp)$ where m is the length of the sequence and p is the number of HMM parameters. SVM-Fisher also includes the training of profile HMMs as a preprocessing step, however, this will be ignored since it is done only once prior to the actual vectorisations (this is similar to our profile creation step). Recall that the time complexities of ProfGram and ProfBlock for the vectorisation of a sequence are $O(kmn)$ and $O(\min(m,n)mn)$ respectively, where n is the length of the profile. Since $n \approx p$, the ProfGram vectorisation method takes approximately the same amount of time as the SVM-Fisher vecorisation. On the other hand, ProfBlock takes approximately $\min(m,n)$ times as long as the SVM-Fisher method (although through our expected matching length approximation technique the time required is much less). However, it is important to note that for this dataset, the SVM-Fisher vectorisation of a sequence is usually done relative to several HMMs to produce feature vectors which are subsequently combined.

The performance of our methods on this dataset is satisfactory but does not match that of SVM-Fisher. We suspect that this is due to the excessive amount of positive training examples from which a profile is built. Some of these examples may be too

distantly related, resulting in a profile that is too general. Unfortunately, it was not possible to verify this hypothesis due to time constraints.

6 Conclusion

In general, the results obtained with our methods significantly outperform previous non-discriminative methods of protein homology detection and are comparable to the state-of-the-art SVM-based methods. In addition, it is interesting to note that our techniques perform well even when only a few positive training examples are available (as in the Liao and Noble dataset).

On the other hand, the performance of our methods on the Jaakkola *et al.* dataset suffers, presumably because the generated profile is too general. To handle this situation, Jaakkola *et al.* in the original SVM-Fisher experiments, created a number of models (in the form of HMMs) for different sets of homologous proteins. We suspect that if we create a similar setup, where the HMMs are replaced by our profiles, better results would be obtained. In this setup, our vectorisation methods would have to combine the scores from the different profiles. The combination of scores is an object of research, but preliminary functions could be the mean or the maximum of all the scores.

References

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[BCHM94] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov Models of biological primary sequence information. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 91, pages 1059–1063, 1994.

[Bus04] S. Busuttil. Support vector machines for detecting remote protein homologies. Master's dissertation, University of Malta, July 2004.

[CS00] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[GLE90] M. Gribskov, R. Luthy, and D. Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1990.

[GR96] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

[Gus97] D. Gusfield. *Algorithms on strings, trees and sequences*. Cambridge University Press, USA, 1997.

[Her02] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. The MIT Press, 2002.

[HH94] S. Henikoff and J. G. Henikoff. Position-based sequence weights. *Journal of Molecular Biology*, 243:574–578, 1994.

[HH96] J. G. Henikoff and S. Henikoff. Using substitution probabilities to improve position-specific scoring matrices. *Computer Applications in the Biosciences*, 12:135–143, 1996.

[JDH99] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh Annual International Conference on Intelligent Systems for Molecular Biology*, pages 149–1158, Menlo Park, CA, 1999. AAAI Press.

[JDH00] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal on Computational Biology*, 7(1–2):95–114, 2000.

[Joa99] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT-Press, 1999.

[KBH98] K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[KBS+94] A. Krogh, M. Brown, I. Saira Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[LS02] L. Liao and W. Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the sixth annual international conference on Computational biology*, pages 225–232. ACM Press, 2002.

[LS03] L. Liao and W. Stafford Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[MBHC95] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarity in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[Pea85] W. R. Pearson. Rapid and sensitive sequence comparisons with FASTP and FASTA. *Methods in Enzymology*, 183:63–98, 1985.

[PKB+98] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210, 1998.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[THG94] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[Vap79] V. Vapnik. *Estimation of dependences based on empirical data [in Russian]*. Nauka, Moscow, 1979. English translation: Springer-Verlag, New York, 1982.

[Vap95] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, 1995.