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

**Steven Busuttil**[1] **John Abela**[2] **Gordon J. Pace**[2]

steven@cs.rhul.ac.uk jabel@cs.um.edu.mt gordon.pace@um.edu.mt

[1] Department of Computer Science, Royal Holloway, University of London, UK
[2] Department of Computer Science and Artificial Intelligence, University of Malta, Malta

**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.

**Keywords:** remote protein homology detection, support vector machine (SVM), kernel, profile

## 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 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 is known as protein homology detection. Traditional laboratory methods of protein homology detection are lengthy and expensive. Since using these methods 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 structural and functional details are already known — one at a time [17]. Each pair is given a score which indicates the amount by which they are related. This method gives good results for closely related proteins, however, remote homologies are not detected.

In the early 1990s a dramatic accuracy improvement was achieved by collecting the information inherent in a number of related proteins. This information is synthesised into a single model, to which new protein sequences are compared [5]. This technique is more sensitive to remote 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 [1, 11]. This process is performed iteratively and makes a model increasingly more accurate.

In 1999, Jaakkola *et al.* introduced a new computational technique for protein homology detection known as SVM-Fisher [9]. This technique couples a previous statistical protein homology detection method, known as a profile hidden Markov model (HMM), with a discriminator 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.

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 profile, and use an SVM as a discriminator.

# 2 Technical Background

## 2.1 Profiles

Profiles, or position specific scoring matrices (PSSMs) [5], 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. 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. For instance, sequences in the multiple alignment are given a weight which describes their informational value. This is important in cases where the multiple alignment contains a large set of closely related proteins. This set carries more or less the same amount of information as a single sequence, however, its size may allow it to overshadow other more divergent proteins. In addition, when calculating the character position probabilities a method is usually used that gives probabilities to unobserved amino acids based on their presumed association with those observed. 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 of the ratio between the available probabilities and the probability of a particular amino acid being found in nature.

## 2.2 Support Vector Machines

Although the study of support vector machines (SVMs) was started in the late 1970s by Vladimir Vapnik, it was not until the late 1990s that this work came to fruition and SVMs started to receive increasing attention [4, 19]. 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]. 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 [19]. Finding this hyperplane translates to solving a convex quadratic optimisation problem, which is formulated in such a way that all vectors appear inside a dot product.

Frequently, 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 operate in feature space without the inherent complexities. The simplest kernel is the usual dot product (known as the linear kernel), where the feature space is equal to the input space. A more powerful and commonly used kernel is the Gaussian 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 $\|\phi(\mathbf{x})\| = \sqrt{k_r(\mathbf{x}, \mathbf{x})} = 1$. The parameter $\sigma \in \mathbb{R}^+$ controls the amount of smoothing of the decision surface in input space.

An extension of the basic SVM algorithm is the soft margin classifier. Since in real-world data it is common that some erroneous elements are present, this extension allows some training examples to be misclassified, effectively making a trade-off between training accuracy and generalisation performance.

[1] Other uses of support vector machines apart from classification exist but these are beyond the scope of this paper.

# 3 Problem and Related Work

## 3.1 The Problem

Researchers simulate the problem of remote protein homology detection by withholding all members of a SCOP (Structural Classification of Proteins database) [15] family for testing, and training with the remaining members of the SCOP superfamily. In addition to these (positive) sequences, negative examples are taken from folds that are different from the one that the target family belongs to.

There are two semi-standard datasets that are used in the area. The first one was introduced in 1999 by Jaakkola *et al.* [9] and is based on SCOP 1.37. This dataset contains 33 experiments and adds homologous protein sequences to the positive training set. The second dataset, which was introduced in 2002 by Liao and Noble [14], is based on SCOP 1.53 and contains 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. This makes recognition more difficult.

## 3.2 Methods

Essentially, applying SVMs to a problem entails creating a new (problem specific) kernel. For homology detection, there usually is a choice of either (1) creating a new similarity measure between two protein sequences and proving that it is a (string) kernel, or (2) transforming sequences into fixed-length numerical vectors and using a standard kernel. By the closure of kernels [4], the combination of an explicit mapping to a feature space and a kernel results in a valid kernel.

The SVM-Fisher method [9] was introduced in 1999 by Jaakkola *et al.* Initially a number of HMMs are trained on different subsets of the positive training set, and then Fisher scores for each sequence are computed. The Fisher score for a sequence is a vector that measures the difference between it and an HMM. All the Fisher scores for a sequence are combined to give a vector that summarises the difference from a typical member of the target superfamily. An SVM is then trained using an RBF kernel. SVM-Fisher yields results that significantly outperform all previous techniques. In 2002, Liao and Noble proposed a simple but effective remote protein homology detection method, named SVM-pairwise [14]. This method builds an ordered list of training examples, to which each sequence is subsequently compared using a pairwise sequence comparison algorithm. This results in fixed-length vectors of pairwise scores. Such vectors are then used to train an SVM with an RBF kernel. On the Liao and Noble dataset, this method performs considerably better than SVM-Fisher.

The mismatch kernel [13], introduced in 2003 by Leslie *et al.*, maps each protein to the feature space indexed by all possible subsequences of length $k$ (called $k$-grams). For a sequence $s$ this method generates a vector with feature $i$ corresponding to the number of times $k$-gram $i$ appears, with up to $m$ mismatches, in $s$. A linear kernel is used to measure the similarity of these vectors. The performance of the mismatch kernel is comparable to that of the SVM-Fisher and SVM-pairwise methods. 2003 also saw the introduction of the motif kernel [2] by Ben-Hur *et al.* This method uses an ordered list of discrete sequence motifs and vectorises a sequence by taking the number of times each motif appears in it. The linear kernel is then used to train an SVM. On a custom dataset, the performance of this method is better than some previous techniques.

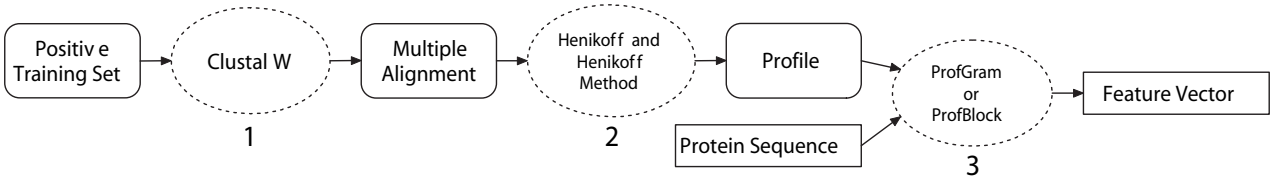
In 2004, Saigo *et al.* proposed a new string kernel for protein homology detection, called the local alignment kernel [16]. To measure the similarity between two protein sequences, this kernel takes the sum of the scores of all their possible local alignments. On the dataset by Liao and Noble, this method outperforms all previous SVM-based techniques. The profile kernel [12] introduced in 2004 by Kuang *et al.*, is related to the mismatch kernel. Initially, a profile is produced for each sequence, and then a kernel is defined on the profiles. Similarly to the mismatch kernel, the profile kernel maps sequences to the feature space indexed by all $k$-grams. This time, however, mismatches are only allowed for amino acids that have a profile column probability greater than a threshold. On a custom dataset, this method outperforms SVM-pairwise.

# 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.

## 4.1 Approach Overview

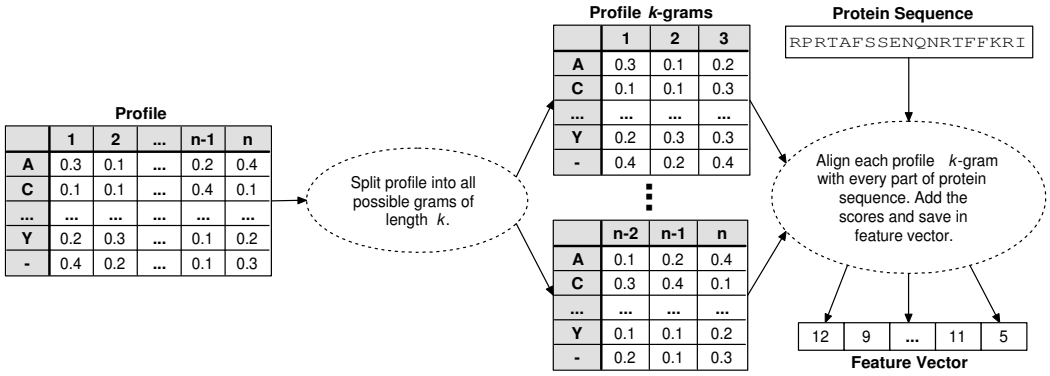
The following is a high level description of the steps involved in our vectorisation method (also illustrated in the figure below): (1) Compute a multiple alignment of the positive training set using ClustalW, a standard multiple alignment program by Thompson *et al.* [18]. (2) Build a profile from the multiple alignment using the Henikoff and Henikoff position-based method. Effectively, this results in a model that encapsulates what makes a protein part of the target superfamily. (3) Vectorise a sequence by comparing it to the profile to produce a vector that measures their similarity.




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 [8], then the character position specific probabilities are calculated using the pseudo-count method by Henikoff and Henikoff [7]. 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. 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, giving a value that represents the similarity of the whole sequence to the $k$-gram. The resulting $k$-gram score is stored as an element of the feature vector. This process is illustrated below for $k = 3$ and is presented more formally subsequently.




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 profile score for character $s_j$ appearing in position $i$ 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[2]:

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

Equation 2 is thus used to calculate vector $\mathbf{v}$. An empirical test was carried out to determine a good candidate value for $k$. It seems that for some experiments a small width is better while for others a wider profile gram gives better results. We finally opted for $k = 4$ which gives a performance that is consistent throughout, even if 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.

#### 4.3.1 The Expected Matching Length

**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:

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

#### 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 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 $i$th 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 positions):

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

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 $\varepsilon$ 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 (more details and a full proof can be found in [3]):

[2]Recall that adding log-odds scores is equivalent to multiplying the underlying probability ratios.

**Lemma 4.1.**

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

Therefore, if the calculation of the expected matching length is stopped when $l(l-k)\prod_{j=1}^{k} p_j \leq \varepsilon$ Lemma 4.1 guarantees that the summation of all the other terms in the series is bound above by $\varepsilon$ and hence, $e_{ij}$ cannot increase by more than $\varepsilon$. This reduces the computation required to calculate (an approximation of) $e_{ij}$ considerably. Empirical analysis showed that setting $\varepsilon$ to be 0.001 takes only 14% of the time taken to calculate the exact expected matching length with a precision loss of just $1.2 \times 10^{-5}$.

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

| | | **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** | | ↓ | ↓ | ... | ↓ |
| | | $(g(\mathbf{t}_1),$ | $g(\mathbf{t}_2),$ | ... | $g(\mathbf{t}_n))'$ |

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 $\mathbf{t}_i$:

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

Note that the sum of squares is divided by the length of the sequence, $m$, to normalise the values across sequences of different lengths.

## 4.4 Time Complexity Analysis

In analysing the time complexity of our vectorisation methods, we will be taking the profile as given[3]. 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 grams). Therefore, the time complexity for ProfGram is $O(kmn)$. Note that in practice $k$ is usually a small constant.

[3]The calculation of a profile from a multiple alignment is a preprocessing step that is done just once for the dataset.

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, the length compared is usually just a small fraction of the maximum length possible. Empirical evidence indicates that this reduces the $\min(m, n)$ factor considerably.

## 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. 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. For our experiments, we normalise each vector to Euclidean length 1. Note that more information on the experiments performed and more detailed results can be found in [3].

In practice, the output of an SVM is a real number in the range $[-1, +1]$. To evaluate the predictions produced by an SVM, we employ two methods that are commonly used in bioinformatics. 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 and a smaller value indicates better performance. The *receiver operating characteristic (ROC)* [6] 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. Clearly, the ROC score is within the interval $[0, 1]$ and a higher score indicates better performance.

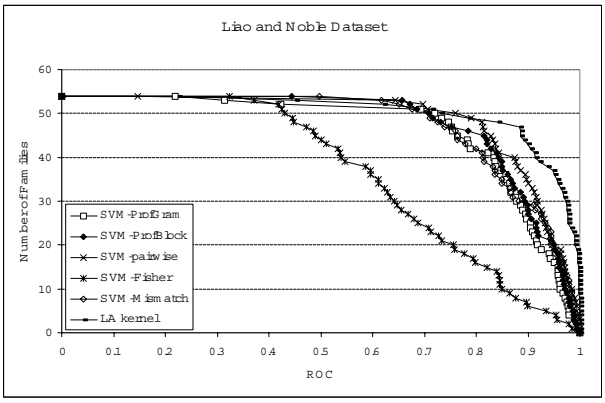
Through an empirical analysis it was found that our methods perform best when coupled with an RBF kernel (see Equation 1). Using this setup, two SVM parameters have to be chosen prior to training: $\sigma$, the RBF kernel smoothing parameter, and $C$ the (soft margin) SVM training error to generalisation performance trade-off parameter. Our goal is to identify a good $C$ and $\sigma$ pair such that the classifier accurately predicts unknown data. For this, we use a technique known as cross-validation. In $k$-fold cross-validation, the training set is split into $k$ subsets of equal size. For a particular $C$ and $\sigma$ configuration, 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 $\sigma$ over some ranges, a grid search using cross-validation is employed. In our experiments, the grid search performed is very coarse due to the size our datasets.

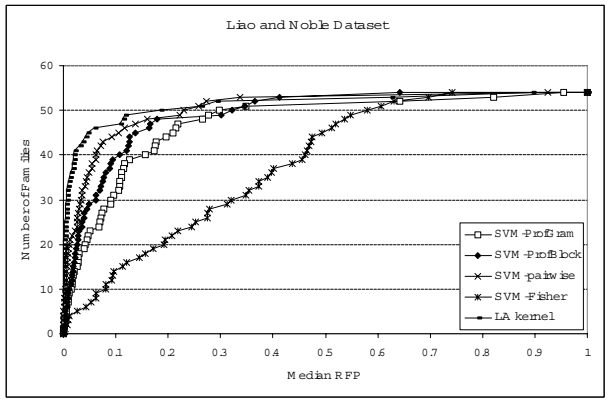
### 5.1 Results

For our experiments we use Joachims' implementation of a support vector machine, SVM*light* [10]. This implementation can handle large training sets and allows us to specify different cost models. We test our methods, SVM-ProfGram and SVM-ProfBlock, on both the Liao and Noble dataset and the Jaakkola *et al.* dataset. For comparison, we include the results obtained by SVM-Fisher, SVM-pairwise, the mismatch kernel (denoted SVM-Mismatch) and the local alignment kernel (denoted LA kernel)[4]. We were unable to include results for the motif kernel and the profile kernel since these were run on custom datasets. As a rough idea of the time taken by the experiments, consider that the 233,073 ProfBlock vectorisations required for the Liao and Noble dataset took approximately 15 hours.

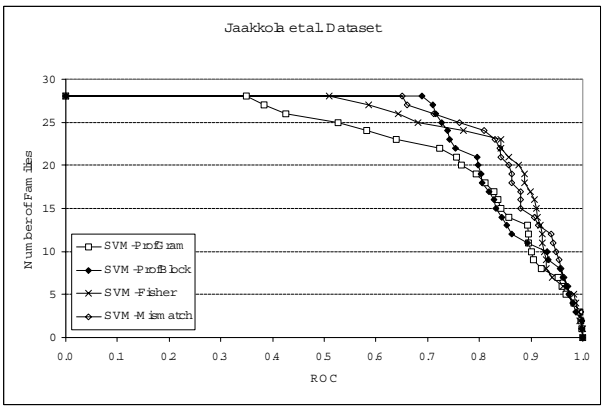
[4]The results of competing methods reported in this paper are those published in [14] and [9], except those of SVM-Mismatch and those of the LA kernel (from private communications with R. Kuang and H. Saigo respectively).

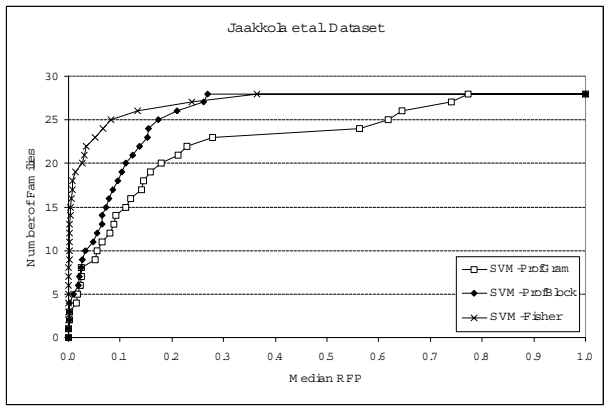
Our methods for remote protein homology detection were designed for the Liao and Noble dataset. For this dataset, the SVM parameters grid search was iterated over the following values: $C = \{2^{-3}, 2^{-2}, \ldots, 2^{5}\}$ and $\sigma = \{2^{-3}, 2^{-2}, \ldots, 2^{7}\}$. The SVM was trained with different 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. Negative training examples were left with the default misclassification cost factor of 1. Below is a summary of the results obtained on this dataset.




On the other hand, ProfGram and ProfBlock were not designed for datasets like the one by Jaakkola *et al.* where the positive training set contains a lot of extra homologous proteins. In addition, it was not possible to run our methods on 5 of the experiments (the Immunoglobulin families) because the size of the positive training sets prevented us from producing the corresponding multiple alignments. Consequently, we will have to leave out these experiments from our 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. Below are the results.




## 5.2 Analysis of Results

In general and as was expected, SVM-ProfBlock performs better than SVM-ProfGram, even though the difference is not very pronounced. Of particular interest is the performance of our methods on the Liao and Noble dataset since this was the dataset they were designed for. Although our methods are similar to SVM-Fisher in that they use a model of the positive training set, they still clearly outperform SVM-Fisher. The fact that our methods perform so well on limited positive training

examples is commendable. SVM-ProfGram performs equally well as SVM-Mismatch, while SVM-ProfBlock performs comparably to SVM-pairwise (although the latter has a slight advantage). The current state-of-the-art technique for this dataset, the local alignment kernel, is significantly better than any of our methods. The performance of SVM-ProfGram and SVM-ProfBlock is also satisfactory on the Jaakkola *et al.* dataset, even though they were not designed for it. It is important to keep in mind, however, that five experiments were left out of the benchmark, therefore the results here are somewhat incomplete. On this dataset, SVM-Fisher and SVM-Mismatch perform better than our methods. 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.

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 $k$ is a small constant, $m$ is the length of the sequence, and $n$ is the length of the profile[5]. The time complexity of the vectorisation step of SVM-pairwise for a single sequence is $O(m^2 l)$ where $l$ is the amount of sequences in the training set. Typically, $l \gg \max(m,n)$, making both our methods more efficient. The time complexity of the vectorisation step of SVM-Fisher for a single sequence is $O(mp)$ where $p$ is the number of HMM parameters[6]. 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. However, it is important to note that for the Jaakkola *et al.* dataset, the SVM-Fisher vectorisation of a sequence is done relative to several HMMs to produce feature vectors which are subsequently combined. This means that for this dataset several $O(mp)$ operations have to be performed. The implementations of the mismatch kernel and the local alignment kernel do not explicitly vectorise sequences, but give their similarity measure immediately. This means that one of their operations is equivalent to two vectorisations. The time complexities of the mismatch kernel and the local alignment kernel are $O(2m)$ and $O(m^2)$ respectively, making our methods more expensive.

## 6 Conclusion

In general, the results obtained with our profile-based methods significantly outperform previous non-discriminative methods of protein homology detection and are better or comparable to competing SVM-based methods. The only competing method that performs significantly better than both our methods is the local alignment kernel. ProfGram performs better than ProfBlock in terms of its computational complexity, whereas ProfBlock has a slight performance advantage. It is also interesting to note that our techniques perform well even when only a few positive training examples are available.

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 general, we feel that designing new techniques that combine profiles and SVMs for remote protein homology detection is still an open research area.

## References

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

[5] The $\min(m,n)$ term of the ProfBlock time complexity is reduced considerably through our approximation technique.

[6] 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.

[2] Ben-Hur, A. and Brutlag, D., Remote homology detection: A motif based approach, *Bioinformatics*, 19:i26–i33, 2003.

[3] Busuttil, S., Abela, J., and Pace, G.J., Support vector machines with profile-based kernels for remote protein homology detection, *CSAI Technical Reports*, CSAI2004-01, University of Malta, 2004. `http://www.cs.um.edu.mt/~reports/archive/CSAI2004-01.pdf`

[4] Cristianini, N. and Shawe-Taylor, J., *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[5] Gribskov, M., Luthy, R., and Eisenberg, D., Profile Analysis, *Methods Enzym.*, 183:146–159, 1990.

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

[7] Henikoff, J.G. and Henikoff, S., Using substitution probabilities to improve position-specific scoring matrices, *Comput. Appl. Biosci.*, 12:135–143, 1996.

[8] Henikoff, S. and Henikoff, J.G., Position-based sequence weights, *J. Mol. Biol.*, 243:574–578, 1994.

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

[10] Joachims, T., Making large-Scale SVM Learning Practical, *Advances in Kernel Methods: Support Vector Learning*, Schölkopf, B., Burges, C., and Smola, A., editors, MIT-Press, 1999.

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

[12] Kuang, R., Ie, E., Wang, K., Wang, K., Siddiqi, M., Freund, Y., and Leslie, C., Profile-based string kernels for remote protein homology detection and motif extraction, *Proc. Comput. Systems Bioinfo.*, 2004.

[13] Leslie, C., Eskin, E., Weston, J., and Stafford Noble, W., Mismatch string kernels for SVM protein classification, *Advances in Neural Information Processing Systems*, S., Becker, S., Thrun, K., Obermayer, editors, MIT Press, 2003.

[14] Liao, L., and Stafford Noble, W., Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships, *J. Comput. Biol.*, 10(6):857–868, 2003.

[15] Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C., SCOP: A structural classification of proteins database for the investigation of sequences and structures, *J. Mol. Biol.*, 247:536–540, 1995.

[16] Saigo, H., Vert, J-P., Ueda, N., and Akutsu, T., Protein homology detection using string alignment kernels, *Bioinformatics*, 20:1682–1689, 2004.

[17] Smith, T.F. and Waterman, M.S., Identification of common molecular subsequences, *J. Mol. Biol.*, 147:195–197, 1981.

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

[19] Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.