ORIGINAL RESEARCH ARTICLE

Appl Bioinformatics 2005; 4 (1): 45-52
1175-5636/05/0001-0045/$34.95/0

Application of String Kernels in Protein Sequence Classification

Nazar M. Zaki,[1] *Safaai Deris*[2] and *Rosli Illias*[3]

1 College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
2 Artificial Intelligence and Bioinformatics Laboratory, Faculty of Computer Science and Information System, Universiti Teknologi, Skudai, Johor, Malaysia
3 Department of Bioprocess Engineering, Faculty of Chemical and Natural Resource Engineering, Universiti Teknologi, Skudai, Johor, Malaysia

Abstract

Introduction: The production of biological information has become much greater than its consumption. The key issue now is how to organise and manage the huge amount of novel information to facilitate access to this useful and important biological information. One core problem in classifying biological information is the annotation of new protein sequences with structural and functional features.

Method: This article introduces the application of string kernels in classifying protein sequences into homogeneous families. A string kernel approach used in conjunction with support vector machines has been shown to achieve good performance in text categorisation tasks. We evaluated and analysed the performance of this approach, and we present experimental results on three selected families from the SCOP (Structural Classification of Proteins) database. We then compared the overall performance of this method with the existing protein classification methods on benchmark SCOP datasets.

Results: According to the F1 performance measure and the rate of false positive (RFP) measure, the string kernel method performs well in classifying protein sequences. The method outperformed all the generative-based methods and is comparable with the SVM-Fisher method.

Discussion: Although the string kernel approach makes no use of prior biological knowledge, it still captures sufficient biological information to enable it to outperform some of the state-of-the-art methods.

Classification of protein sequences into functional and structural families based on sequence homology is a central problem in computational biology.[1] Many approaches have been applied, such as pairwise similarity of sequences,[2-4] profiles for protein families,[5] consensus patterns[6,7] and hidden Markov models (HMMs).[8-10] Another successful approach based on modelling the difference between positive and negative examples was introduced, and this approach obtains additional classification accuracy. The most prominent method that uses this approach is the SVM-Fisher method.[11] The SVM-Fisher method begins by training a generative HMM for a protein family and then using the model to extract a representation for each protein sequence in the form of *sufficient statistics*. Next, the sufficient statistics are treated to produce an analogous quantity known as the *Fisher scores*. Finally, support vector machines (SVMs) in conjunction with the Fisher scores are used to discriminate between protein families.

The SVM-Fisher method adds more accuracy to the problem of detecting remote protein homology, and it is also appealing because it combines the rich biological information encoded in a profile HMM with the discriminative power of the SVM classifiers.[1] Generally, large amounts of data or prior knowledge are required to train the HMM. Moreover, calculation of the feature vectors depends on dynamic programming, making it inefficient to compute the kernel matrix. We certainly need a more efficient method in terms of feature extraction, computational cost, and maintaining good classification abilities even when the training dataset is small.

In this article, we introduce the application of string kernels (SKs) in classifying protein sequences. The idea of designing a kernel function based on the string is not novel; however, applying a similar idea for solving more sensitive problems such as protein sequence classification is a great challenge. The string kernels approach[12] has been shown to achieve good performance on text categorisation tasks. The basic idea is to compare two protein

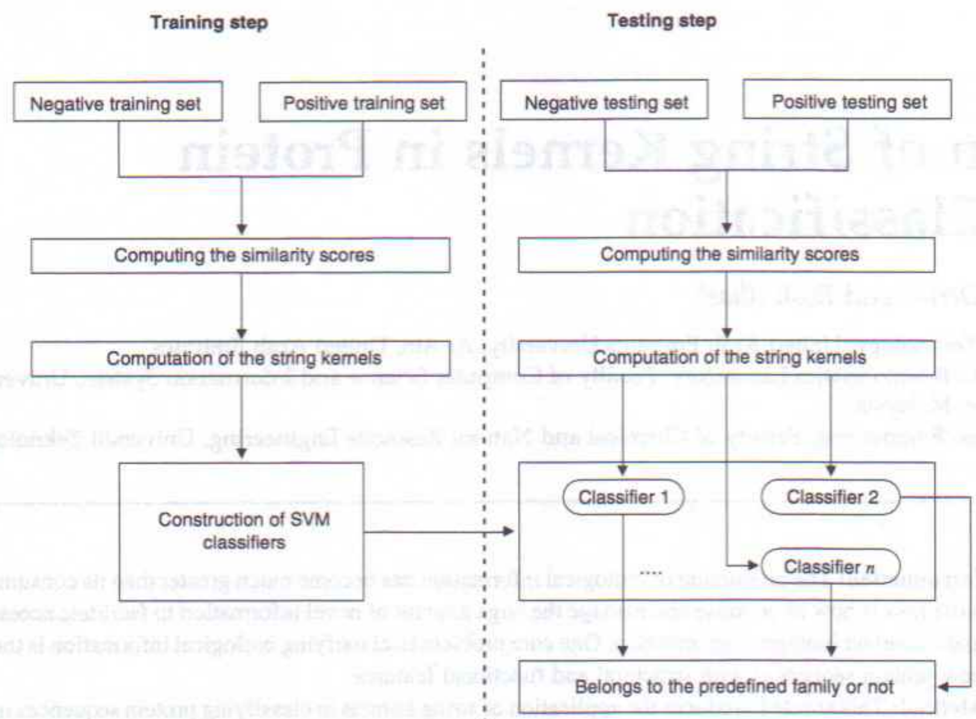
Training step
Testing step
Negative training set
Positive training set
Negative testing set
Positive testing set
Computing the similarity scores
Computing the similarity scores
Computation of the string kernels
Computation of the string kernels
Construction of SVM classifiers
Classifier 1
Classifier 2
....
Classifier *n*
Belongs to the predefined family or not

Fig. 1. Overview of the string kernel method in conjunction with support vector machines (SVM-SK).

sequences by looking at common subsequences of a fixed length. String kernels apparently make no use of prior knowledge and yet they have been used with considerable success.[13] The string kernel is built on the kernel method introduced by Haussler[14] and Watkins.[15] The kernel computes similarity scores between protein sequences without ever explicitly extracting the features. This technique does not use any biological knowledge, in the sense that it considers the protein dataset as just a long string of amino acids. A subsequence is any ordered sequence of k amino acids occurring in the protein sequence, and is not necessarily contiguous. The subsequences are weighted by an exponentially decaying factor of their full length in the sequence, hence emphasising those occurrences that are close to contiguous.

We evaluated and analysed the performance of the string kernel approach for classifying protein sequences, and experimental results on three selected families from the SCOP (Structural Classification of Proteins) database are presented in this article. Then, using benchmark SCOP datasets, we compared this method with the most successful remote homology detection methods for proteins, such as HMMER,[10] BLAST®,[3] SAM[16] and the SVM-Fisher method.[11]

System and Methods

Figure 1 is an overview of the proposed method, which we call SVM-SK (support vector machine-string kernel). It consists of two main steps: (a) the training step, in which the SVM classifiers are constructed; and (b) the testing step, which uses the SVM to determine if the protein belongs to the predefined family or not. Both steps require computing similarity scores between protein sequences without explicitly extracting the features.

Support Vector Machines

The SVM algorithm[17,18] addresses the general problem of learning to discriminate between positive and negative members of a given class of n-dimensional vectors. The algorithm operates by mapping the given training set into a possibly high-dimensional feature space, and attempting to learn a separating hyperplane between the positive and negative examples for possible maximisation of the margin between them. The margin corresponds to the distance between the points residing on the two edges of the hyperplane. Having found such a plane, the SVM can then predict the classification of an unlabelled example. In fact, much of the power of the SVM comes from its criterion for selecting a separating plane when many candidate planes exist: the SVM chooses the plane that maintains a maximum margin from any point in the training set.[19]

String Kernel Overview

The feature space is generated by all the subsequences of bounded length. In order to derive the string subsequence kernel (SSK), one begins from the features and then computes their inner product. Hence, the criterion of satisfying Mercer's condition (positive semi-definiteness) automatically applies here. It maps strings to a feature vector indexed by all k-tuples of amino acids. A k-tuple will have a non-zero entry if it occurs as a subsequence anywhere (not necessarily contiguous) in the string. The weighting of the feature will be the sum over the occurrences of the k-tuple of a decaying factor of the length of the occurrence.

Following Cristianini[18] and Lodhi et al.,[12] the string kernel can be defined as follows. Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s and t, we denote by $|s|$ the length of the string $s = s_1 \ldots s_{|s|}$, and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \ldots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, \ldots, i_{|u|})$, with $1 \leq i_1 < \ldots < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, for $j=1,\ldots,|u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$.

We denote by Σ^n the set of all finite strings of length n, and by Σ^* the set of all strings (equation 1):

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n \qquad \text{(Eq. 1)}$$

We now define feature spaces $F_n = \Re^{\Sigma^n}$. The feature mapping φ for a string s is given by defining the u coordinate $\varphi_u(s)$ for each $u \in \Sigma^n$. We define (equation 2):

$$\varphi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \qquad \text{(Eq. 2)}$$

for some $\lambda \leq 1$. These features measure the number of occurrences of subsequences in the string s, weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to their frequency of occurrence and lengths (equation 3):

$$\begin{aligned} k_n(s,t) &= \sum_{u \in \Sigma^n} \left\langle \varphi_u(s) \cdot \varphi_u(t) \right\rangle \\ &= \sum_{u \in \Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{u \in \Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})} \end{aligned} \qquad \text{(Eq. 3)}$$

Computation of Similarity Scores and String Kernels

The SSK measures similarity among the protein sequences by summing the common substrings. In other words, sequences having more substrings in common will have a higher similarity score compared with the sequences that share few substrings. It is worth noting that these substrings can be non-contiguous, which makes these features distinguishable. Moreover, these substrings are weighted according to the degree of contiguity in a sequence. This idea can be illustrated by the simple examples given below.

Consider the following protein sequences:

>e1izb.2c 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
giveqcctsicslyqlenycn
>e1izb.2d 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
fvnqhlcgshlvqalylvcgergffytpkt
>e1sdb.1b 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
nqhlcgshlveqlylvcgergff

Assume that the substring 'gqlylc' (below in bold) occurs in all the above protein sequences, but is not always contiguous. Hence, this substring has a different weighting in each protein sequence, as shown below.

>e1izb.2c 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
give**q**cctsics**ly**q**l**eny**c**n
>e1izb.2d 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
fvnqhlc**g**shlv**q**a**lyl**v**c**gergffytpkt
>e1sdb.1b 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
nqhlc**g**shlve**qlyl**v**c**gergff

SSK generates an entirely different feature space. For each substring there is a dimension of feature space, and the value of these coordinates depends on how frequently and compactly this string is embedded in the protein sequences of interest.

We now illustrate SSKs by examples. Consider the following protein sequences:

>e1izb.2c 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
giveqcctsics**lyq**lenycn
>e1izb.2d 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
fvnqhlcgshlvq**aly**lvcgergffytpkt
>e1sdb.1b 7.1.1.1.3 Insulin {Pig (*Sus scrofa*)}
nqhlc**gsh**lveqlylvcgergff

From the above protein sequences, consider the highlighted strings (in bold). Let us first assume we are comparing the first two protein sequences, where there exists one string in each sequence. Let the first sequence comprise the string 'lyq' and the second sequence comprise the string 'aly'. For simplicity, we set the length of the substring to 2. In other words, these sequences are implicitly transformed into feature vectors, where each feature vector is indexed by the substrings of length 2. Table I presents the

Table I. Mapping two strings '*lyq*' and '*aly*' to a 5-dimensional feature space

Feature vectors	Strings				
	ly	*lq*	*yq*	*al*	*ay*
φ(*lyq*)	λ^2	λ^3	λ^2	0	0
φ(*aly*)	λ^2	0	0	λ^2	λ^3

λ = weighted decay factor; φ = feature mapping.

5-dimensional feature space, feature vectors $\varphi(lyq)$ and $\varphi(aly)$, and the corresponding kernel.

In this case, the un-normalised kernel $k(lyq,aly)$ can clearly be computed as λ^4, whereas computing the normalised kernel between '*lyq*' and '*aly*' requires computing $k(lyq,lyq)$ and $k(aly,aly)$.

Since (equation 4):

$$k(lyq, lyq) = 2\lambda^4 + \lambda^6 \qquad \text{(Eq. 4)}$$

hence, normalise (equation 5):

$$k(lyq, aly) = \frac{\lambda^4}{(2\lambda^4 + \lambda^6)} = \frac{1}{(2+\lambda^2)} \qquad \text{(Eq. 5)}$$

We can now consider the comparison among all the three strings in the protein sequences given above. Again considering $k = 2$, we obtain an 8-dimensional feature space, where strings are mapped in table II. Table II shows that the un-normalised kernel between '*lyq*' and '*aly*' is $k(lyq,aly) = 1/(2 + \lambda^2)$. For instance, if $k(lyq,gsh) = 0$, then it is clear that the kernel function is able to anticipate that there are no similarities between the two strings.

Datasets

The performance of the SVM-SK method is tested on the SCOP database version 1.37.[20] The use of SCOP datasets designed by Jaakkola et al.[11] allows direct comparison with the previous work on protein remote homology detection.[2-4,9,11] For the test, Jaakkola et al.[11] selected all SCOP families that contain at least five PDB90 sequences and have at least ten PDB90 sequences in the other families in their superfamily. This process results in 33 test families from 16 superfamilies. The *positive test* examples are simulated by members of a target SCOP family from a given superfamily. *Positive training* examples are chosen from the remaining families in the same superfamily. Whereas *negative test* and *negative training* examples are chosen from disjoint sets of folds outside the fold of the target family. The dataset is available at http://www.cse.ucsc.edu/research/compbio/discriminative.

Our experiments for observing the influence of the tunable parameters on the string kernels are performed on immunoglobulin C1 domain, immunoglobulin C2 domain and α-amylase N-terminal domain families (table III). The only reason for selecting these families is because they contain few total training and test sequences compared with the rest of the families. Note that we need to run the experiments many times on one family to achieve the required analysis.

Table II. Mapping three strings '*lyq*', '*aly*' and '*gsh*' to an 8-dimensional feature space

Feature vectors	Strings							
	ly	*lq*	*yq*	*al*	*ay*	*gs*	*gh*	*sh*
$\varphi(lyq)$	λ^2	λ^3	λ^2	0	0	0	0	0
$\varphi(aly)$	λ^2	0	0	λ^2	λ^3	0	0	0
$\varphi(gsh)$	0	0	0	0	0	λ^2	λ^3	λ^2

λ = weighted decay factor; φ = feature mapping.

Table III. Domain families included in the experiments

SCOP database ID	Domain family	Positive set		Negative set	
		training	testing	training	testing
2.1.1.2	Immunoglobulin C1 domain	174	103	950	1229
2.1.1.3	Immunoglobulin C2 domain	272	5	950	1229
3.1.1.1	α-Amylase N-terminal domain	57	13	1142	1227

SCOP = Structural Classification of Proteins.

Measures for Evaluating System Performance

The performance of the SVM-SK system is measured by how well it can assign a novel protein sequence to its correct family. A system can make errors by assigning the sequences to families to which they do not belong or failing to assign the sequences to families to which they do belong.

For evaluating the performance of the SVM-SK method, two evaluation measures were used:

1. F1 performance measure (gives equal weighting to both precision and recall): $F1 = 2P_{re}R_{ec}/(P_{re} + R_{ec})$, where P_{re} is the precision and R_{ec} is the recall. These are calculated as follows: $R_{ec} = tp/(tp + fn)$ and $P_{re} = tp/(tp + fp)$, where *tp* is true positive, *fp* false positive and *fn* false negative.
2. Rate of false positive (RFP): defined as the fraction of negative test sequences that score as high as or better than the positive sequence. $RFP = fp/(fp + tn)$ for $(fp + tn) > 0$, where *tn* is true negative.

Experiments and Results

Preceding sections describe the ability of string kernels to compute the similarity among protein sequences. This section discusses the experiments, of which the overall objective is to understand how the string kernel works in such sensitive and complicated classification tasks.

A protein classification method based on string kernels works by taking as input a set of amino acid sequences. The kernel returns a similarity score between the sequences without explicitly mapping them into the feature vectors. In this section we analyse the behaviour of the classifiers trained in a feature space generated by string kernels. The objective of the experiments is to observe the influence of varying the tunable parameters of λ (weight), C (soft-margin parameter) and k (length) on the classification performance. In order to accomplish this goal, we conducted a series of experiments on the benchmark datasets described in the section titled Measures for Evaluating System Performance.

Implementation

An important advantage we hope to obtain using string kernels is to avoid the computational cost of HMMs; however, the computational cost of string kernels is not cheap. SVM-SK requires some special properties in the software to provide better classification ability and a lower computational cost.

We used a simple gradient-based implementation of SVMs.[21] The algorithm called Adatron comes with theoretical guarantees of convergence to the optimal solution, and of a rate of convergence exponentially fast in terms of the number of iterations,[22] provided that the solution exists. This provides a procedure that emulates SVMs but does not need to use the quadratic programming toolboxes. The Adatron is an online algorithm for learning perceptrons that has an attractive fixed point corresponding to the maximal-margin consistent hyperplane, when this exists.

We applied different tricks in order to speed up the computation of the kernel matrix. First, the program reads a single file containing training and test sets. Second, we added two files containing the indexes of the training and test sets. The SVM-SK is a modified version of the software used by Lodhi et al.[12] for text classification.

Effectiveness of Varying Weighted Decay Factor

In this set of experiments, we analyse the effect of varying the weighted decay factor, λ, on the generalisation performance of a SVM learner that manipulates the information encoded in a SSK. SK weights the substrings according to their proximity in the protein sequence. This is the parameter that controls the penalisation of the interior gaps in the substrings.

We evaluated the performance of this technique by averaging the results over runs of the algorithm. A series of experiments was conducted to study the performance of the protein sequence classification system based on SKs by widely varying λ. We describe the results of these experiments in table IV, where the relation between different values of λ and the corresponding influence of F1 are shown. The value of the subsequence length k was set to 2 and the soft-margin C was set to 1000. It was difficult choosing the subsequence length; however, the main aim of the experiments was to analyse the behaviour of the string kernels by varying the value of the decay factor. By using $k = 2$, we cannot guarantee better classification performance; however, it would enable us to record the outputs in less time. It is interesting to note that F1 peaks at a higher value (λ = 0.9) for all the three families.

In order to judge the classification ability of the SVM-SK method, we calculated the precision, recall and F1 based on a purely random classifier. Each protein sequence is classified true or false with a probability of 0.5 (table IV).

Effectiveness of Varying Soft-Margin Parameter

One of the significant parameters needed to tune our system is the soft-margin parameter or the capacity. The soft-margin parameter C allows us to control how much tolerance for error we allow in the classification of training samples. It therefore affects the generalisation ability of the SVM and prevents it from overfitting

Table IV. Performance of SVM-SK (support vector machine-string kernel) with a varying weighted decay factor (λ)

Domain family	λ	Precision	Recall	F1
Immunoglobulin C1 domain	0.01	0.200	0.107	0.139
	0.03	0.197	0.117	0.146
	0.05	0.228	0.126	0.162
	0.07	0.242	0.146	0.182
	0.09	0.263	0.146	0.188
	0.10	0.259	0.146	0.186
	0.30	0.357	0.194	0.252
	0.50	0.526	0.398	0.453
	0.70	0.618	0.660	0.638
	0.90	0.726	0.825	0.773
	0.99	0.440	0.359	0.396
Immunoglobulin C2 domain	0.01	0.061	0.600	0.111
	0.03	0.056	0.600	0.103
	0.05	0.070	0.600	0.125
	0.07	0.058	0.600	0.105
	0.09	0.055	0.600	0.100
	0.10	0.054	0.600	0.098
	0.30	0.063	0.600	0.113
	0.50	0.074	0.800	0.136
	0.70	0.067	0.600	0.120
	0.90	0.083	0.600	0.146
	0.99	0.051	0.600	0.094
α-Amylase N-terminal domain	0.01	0.059	0.231	0.094
	0.03	0.068	0.231	0.105
	0.05	0.064	0.231	0.100
	0.07	0.059	0.231	0.094
	0.09	0.042	0.154	0.066
	0.10	0.046	0.154	0.070
	0.30	0.060	0.231	0.095
	0.50	0.063	0.231	0.098
	0.70	0.125	0.385	0.189
	0.90	0.148	0.615	0.239
	0.99	0.122	0.462	0.194

Table V. Performance of SVM-SK (support vector machine-string kernel) with varying soft-margin parameter (*C*)

Domain family	*C*	Precision	Recall	F1
Immunoglobulin C1 domain	10	0.891	0.553	0.683
	20	0.861	0.602	0.709
	30	0.815	0.641	0.717
	40	0.773	0.66	0.712
	50	0.725	0.718	0.722
	60	0.724	0.816	0.760
	70	0.724	0.816	0.767
	80	0.741	0.835	0.785
	90	0.723	0.835	0.775
	100	0.726	0.825	0.773
	110	0.726	0.825	0.773
	120	0.726	0.825	0.773
	130	0.726	0.825	0.773
	140	0.726	0.825	0.773
	150	0.726	0.825	0.773
Immunoglobulin C2 domain	10	0.077	0.200	0.111
	20	0.083	0.400	0.138
	30	0.103	0.600	0.176
	40	0.103	0.600	0.176
	50	0.097	0.600	0.167
	60	0.094	0.600	0.162
	70	0.086	0.600	0.150
	80	0.083	0.600	0.146
	90	0.075	0.600	0.133
	100	0.083	0.600	0.146
	110	0.083	0.600	0.146
	120	0.083	0.600	0.146
	130	0.083	0.600	0.146
	140	0.083	0.600	0.146
	150	0.083	0.600	0.146
α-Amylase N-terminal domain	10	0.385	0.385	0.385
	20	0.385	0.385	0.385
	30	0.280	0.538	0.368
	40	0.250	0.538	0.341
	50	0.212	0.538	0.304
	60	0.200	0.538	0.258
	70	0.170	0.615	0.267
	80	0.170	0.615	0.267
	90	0.163	0.615	0.258
	100	0.157	0.615	0.250
	110	0.148	0.615	0.239
	120	0.145	0.615	0.235
	130	0.145	0.615	0.235
	140	0.145	0.615	0.235
	150	0.145	0.615	0.235

the training set. In this set of experiments, we analyse the effect of varying *C*. We again fix the values of subsequence length *k* and the weighted decay factor λ. We used values $k = 2$ and $\lambda = 0.9$. The results of this set of experiments are presented in table V. It is interesting to see that *C* peaks at a value <80. The classification results remain almost the same when the value of *C* is >100.

Effectiveness of Varying Sequence Length

We studied the effect of varying sequence length *k*. We kept the values of the weighted decay parameter λ and the soft-margin parameter fixed and learned a classifier for different values of *k*. For these experiments, the value of λ was set to 0.9, the soft-margin parameter was 1000 and sequence length was varied. It was difficult choosing the weighted decay factor, and table V shows that a different family obtained the highest F1 value at different values of *C*. However, the main objective of the experiments described in this section was to analyse the behaviour of SSK by varying *k*.

The results of this set of experiments are given in table VI. From these results, we find that the performance of the classifier varies with varying sequence length. SVM-SK can be more effective for smaller or moderate length substrings compared with larger substrings. An optimal size of subsequence length can be found in a region that is not very large. For each family, F1 seems to peak at a subsequence length of 5–7 and, surprisingly, it looses the classification ability for a subsequence length >9. Generally, the optimal subsequence length is 5–7, and it seems that shorter or moderate non-contiguous substrings are able to capture the similarities better than the longer non-contiguous substrings. For subsequence length >9, SVM-SK is unable to capture similarities (F1 = 0).

Comparing SVM-SK with the Current Protein Classifiers

The performance of SKs in conjunction with SVMs was compared with the performance of the current homology detection methods HMMER, BLAST®, SAM-98 and Fisher-SVM, where HMMER, BLAST® and SAM-98 are purely generative models, and Fisher-SVM is a combination of generative and discriminative models. The probability scores produced using these methods are on different scales, meaning they cannot be easily compared. We reported the median RFP, which is defined as the fraction of negative test sequence that score as high as, or better than, the positive sequence. Figure 2 illustrates the overall performance in terms of median RFP for detecting protein remote homology in the 33 test families. Figure 2 also shows the overall performance of the above-mentioned methods. The results of the performance by BLAST®, SAM-98, and SVM-Fisher are taken from the report by Jaakkola et al.,[11] and the results of the HMMER performance are reported by Logan et al.[23] on the same datasets. The results show

Table VI. Performance of SVM-SK (support vector machine-string kernel) with varying length (*k*)

Domain family	*k*	Precision	Recall	F1
Immunoglobulin C1 domain	2	0.726	0.825	0.773
	3	0.830	0.806	0.818
	4	0.890	0.786	0.835
	5	0.952	0.767	0.849
	6	0.975	0.757	0.852
	7	0.985	0.631	0.769
	8	1.000	0.573	0.728
	9	1.000	0.495	0.662
Immunoglobulin C2 domain	2	0.103	0.600	0.176
	3	0.111	0.400	0.174
	4	0.177	0.200	0.188
	5	0.200	0.200	0.200
	6	0.380	0.200	0.262
	7	0.333	0.200	0.25
	8	0.300	0.200	0.24
	9	0.280	0.200	0.233
α-Amylase N-terminal domain	2	0.148	0.615	0.239
	3	0.250	0.538	0.269
	4	0.250	0.538	0.341
	5	0.318	0.538	0.400
	6	0.417	0.385	0.400
	7	0.500	0.308	0.381
	8	0.500	0.154	0.235
	9	0.500	0.140	0.218

that the SVM-SK approach delivers comparable performance in classifying protein sequences. The method outperformed all the generative-based methods and is comparable with the SVM-Fisher method, which is among the most accurate. Moreover, SVM-SK outperformed the SVM-Fisher method in classifying some of the protein families.

The performance of SVM-Fisher and SVM-SK in figure 2 is very close, so a more detailed comparison is made in figure 3. Family-by-family comparison of the performance of the SVM-Fisher and SVM-SK methods in terms of median RFP is shown in figure 3. From figure 3, the distribution of the families in the two superiority regions shows that SVM-Fisher is superior compared with SVM-SK; however, the performance difference is not significant. SVM-SK outperformed the SVM-Fisher method in classifying some of the protein families. We consider SVM-SK to be comparable with SVM-Fisher. Moreover, SVM-SK has many advantages over the SVM-Fisher method, as mentioned in the Discussion and Conclusion section.

Discussion and Conclusion

In this article, we presented the application of a new kernel for protein sequence classification. The performance of the string kernels in conjunction with SVMs was tested and analysed by applying this method to three different protein families from the SCOP database. Although the SVM-SK method makes no use of the prior biological knowledge that the structure of an amino acid within the protein sequence can give, it still captures sufficient biological information to enable it to outperform some of the state-of-the-art methods. The experiments indicated that the SVM-SK method can provide an effective alternative to more standard homology detection methods. The results on the SCOP dataset were, however, less encouraging. In most cases, the SVM-Fisher method outperformed SVM-SK. The SVM-SK method has several advantages over the SVM-Fisher method. The SVM-Fisher method requires a large amount of data or prior knowledge to train the HMM. In addition, for instance, calculating the Fisher scores depends on dynamic programming; in practice it is very expensive to compute the kernel matrix.[1] Our SVM-SK approach gives efficient kernel computation and maintains good performance. Another advantage of using SVM-SK is its use of the negative training set.

Based on the computational efficiency, both SVM-SK and SVM-Fisher take approximately $O(n^2)$ for the SVM optimisation, for n training examples. However, the computational cost in finding the hypothesis is significantly less with the kernel-Adatron algorithm used in SVM-SK.[21] SVM-Fisher requires training a profile HMM, whereas the SVM-SK method dispenses with this requirement. Moreover, the SVM-Fisher method requires computing the gradient vectors, which dominates the running time of $O(nmp)$, where n is the number of training examples, m is the

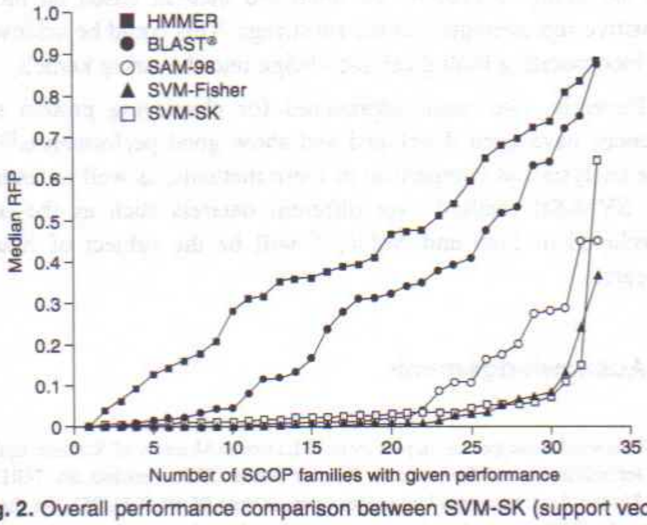

Fig. 2. Overall performance comparison between SVM-SK (support vector machine-string kernel) and the current protein remote homology detection methods HMMER, BLAST®, SAM-98 and Fisher-SVM. **RFP** = rate of false positive; **SCOP** = Structural Classification of Proteins database.

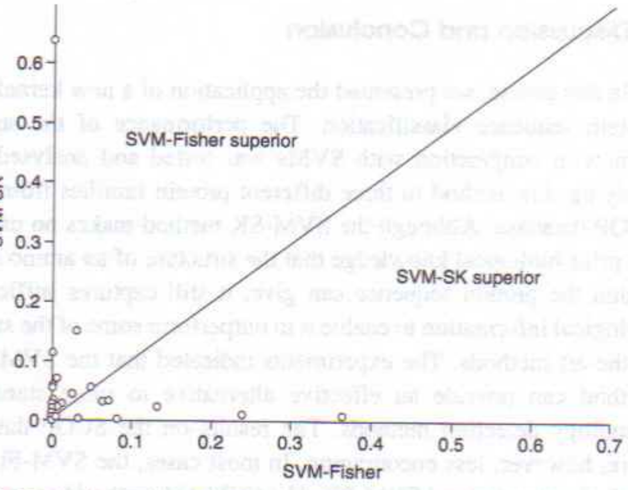

Fig. 3. Family-by-family performance comparison of SVM-SK (support vector machine-string kernel) against the SVM-Fisher method. Each family is plotted as a point (x,y) where x is the median rate of false positive (RFP) for the SVM-Fisher method and y is the median RFP for the SVM-SK method.

length of the longest training set sequence and p is the number of HMM parameters. The SVM-SK method takes roughly $O(n^2)$ for each kernel entry.

The success of applying the SVM-SK method in classifying protein sequences encouraged us to plan future directions such as studying weak areas of the approach that could be further improved. We suspect that by incorporating biological knowledge into the SVM-SK method we could achieve better classification ability. We need to find a more sensitive representation of the substrings. In the SVM-SK method, subsequences are presented using the weighted decay factor λ. However, from studying the effectiveness of varying the weighted decay factor, we noticed that the values of F1 react randomly and are not stable. In the future, we are going to propose an improved method based on more sensitive representation of the substrings. This could be achieved by incorporating biological knowledge into the string kernels.

Recently, two more approaches for classifying protein sequences have been developed and show good performance.[1,19] The analysis and comparison of these methods, as well as testing the SVM-SK method over different datasets such as the one developed in Liao and Noble,[19] will be the subject of future research.

Acknowledgements

This work is supported in part by the Malaysian Ministry of Science, under the Intensification of Research in Priority Areas (IRPA, project no. 74017), and United Arab Emirates University (project vote 05-01-9-11/05). We thank Huma Lodhi for providing information and codes related to his work.

The authors have provided no information on conflicts of interest directly relevant to the content of this article.

References

1. Leslie C, Eskin E, Weston J, et al. Mismatch string kernels for discriminative protein classification. Bioinformatics 2004; 20 (4): 67-76
2. Smith T, Waterman M. Identification of common molecular subsequence. J Mol Biol 1981; 147: 195-7
3. Altschul SF, Gish W, Miller W, et al. Basic local alignment search tool. J Mol Biol 1990; 215: 403-10
4. Altschul SF, Madden TL, Schaffer AA, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997; 25: 3389-402
5. Gribskov M, Lüthy R, Eisenberg D. Profile analysis. Methods Enzymol 1990; 183: 146-59
6. Attwood TK, Beck ME, Bleasby AJ, et al. Prints: a database of protein motif fingerprints. Nucleic Acids Res 1994; 22 (17): 3590-6
7. Bairoch A, Apweiler R. The SWISS-PROT protein sequence data bank and its new supplement TREMBL. Nucleic Acids Res 1996; 24: 21-5
8. Baldi P, Chauvin Y, Hunkapiller T, et al. Hidden Markov models of biological primary sequence information. Proc Natl Acad Sci U S A 1994; 91 (3): 1059-63
9. Krogh A, Brown M, Mian IS, et al. Hidden Markov models in computational biology: applications to protein modeling. J Mol Biol 1994; 235: 1501-31
10. Eddy SR. Multiple alignment using hidden Markov models. Proc Int Conf Intell Syst Mol Biol 1995; 3: 114-20
11. Jaakkola T, Diekhans M, Haussler D. A discriminative framework for detecting remote protein homologies. J Comput Biol 2000; 7 (1-2): 95-114
12. Lodhi H, Saunders C, Shawe-Taylor J, et al. Text classification using string kernels. J Mach Learn Res 2002; 2: 419-44
13. Saunders C, Shawe-Taylor J, Vinokourov A. String kernels, Fisher kernels and finite state automata. In: Becker S, Thrun S, Obermayer A, editors. Proceedings of Neural Information Processing Systems 15. 2003
14. Haussler D. Convolution kernels on discrete structures [technical report UCSC-CRL- 99-10]. Santa Cruz (CA): Computer Science Department, University of California in Santa Cruz, 1999
15. Watkins C. Dynamic alignment kernels: advances in large margin classifiers. Cambridge (MA): MIT Press, 2000: 39-50
16. Karplus K, Barrett C, Hughey R. Hidden Markov models for detecting remote protein homologies. Bioinformatics 1998; 14 (10): 846-56
17. Vapnik VN. The nature of statistical learning theory. New York: Springer Verlag, 1995
18. Cristianini N, Shawe-Taylor J. An introduction to support vector machines. Cambridge (UK): Cambridge University Press, 2000
19. Liao L, Noble WS. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. J Comput Biol 2003; 10: 857-68
20. Murzin AG, Brenner SE, Hubbard T, et al. SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol. 1995; 247: 536-40
21. Friess T, Cristianini N, Campbell C. The kernel-adatron: a fast and simple learning procedure for support vector machines. In: Shavlik J, editor. Proceedings of the Fifteenth International Conference on Machine Learning (ICML). Bonn: Morgan Kaufmann, 1998: 188-96
22. Opper M. Learning times of neural networks: exact solution for a perceptron algorithm. Phys Rev A 1998; 38 (7): 3824-6
23. Logan B, Moreno P, Suzek B, et al. A study of remote homology detection [CRL technical report 2001/5]. Cambridge (MA): Cambridge Research Laboratory, 2001

Correspondence and offprints: Dr *Nazar M. Zaki*, College of Information Technology, United Arab Emirate University, PO Box 17555, Al Ain, United Arab Emirates.
E-mail: nzaki@uaeu.ac.ae