THE SPECTRUM KERNEL: A STRING KERNEL FOR SVM PROTEIN CLASSIFICATION

CHRISTINA LESLIE, ELEAZAR ESKIN, WILLIAM STAFFORD NOBLE [a]

{cleslie,eeskin,noble}@cs.columbia.edu

Department of Computer Science, Columbia University, New York, NY 10027

We introduce a new sequence-similarity kernel, the spectrum kernel, for use with support vector machines (SVMs) in a discriminative approach to the protein classification problem. Our kernel is conceptually simple and efficient to compute and, in experiments on the SCOP database, performs well in comparison with state-of-the-art methods for homology detection. Moreover, our method produces an SVM classifier that allows linear time classification of test sequences. Our experiments provide evidence that string-based kernels, in conjunction with SVMs, could offer a viable and computationally efficient alternative to other methods of protein classification and homology detection.

1 Introduction

Many approaches have been presented for the protein classification problem, including methods based on pairwise similarity of sequences [1,2,3], profiles for protein families [4], consensus patterns using motifs [5,6] and hidden Markov models [7,8,9]. Most of these methods are *generative* approaches: the methodology involves building a model for a single protein family and then evaluating each candidate sequence to see how well it fits the model. If the "fit" is above some threshold, then the protein is classified as belonging to the family. *Discriminative* approaches [10,11,12] take a different point of view: protein sequences are seen as a set of labeled examples – positive if they are in the family and negative otherwise – and a learning algorithm attempts to *learn* the distinction between the different classes. Both positive and negative examples are used in training for a discriminative approach, while generative approaches can only make use of positive training examples.

One of the most successful discriminative approaches to protein classification is the work of Jaakkola et al. [10,11] for detection of remote protein homologies. They begin by training a generative hidden Markov model (HMM) for a protein family. Then, using the model, they derive for each input sequence, positive or negative, a vector of features called Fisher scores that are assigned to the sequence. They then use a discriminative learning algorithm called a

---

[a] Formerly William Noble Grundy, see www.cs.columbia.edu/~noble/name-change.html

support vector machine (SVM) in conjunction with the feature vectors – in the form of a kernel function called the Fisher kernel – for protein family classification. A serious drawback of their approach is its computational expense – both for generating the kernel on the training set and for classifying test sequences – since the HMM is required for computing feature vectors both for training and test sequences. Training an HMM, or scoring a sequence with respect to an HMM, requires a dynamic programming algorithm that is roughly quadratic in the length of the sequence.

In this paper, we revisit the idea of using a discriminative approach, and in particular support vector machines, for protein classification. However, in place of the expensive Fisher kernel, we present a new string kernel (sequence-similarity kernel), the spectrum kernel, for use in the SVM. The kernel is designed to be very simple and efficient to compute and does not depend on any generative model, and we produce an SVM classifier that can classify test sequences in linear time. Moreover, the method is completely general in that it can be used for any sequence-based classification problem. In the experiments reported here, we do not incorporate prior biological information specific to protein classification, although we plan to use prior information in future research. We report results for experiments over the SCOP [13] database and show how our method performs surprisingly well given its generality.

When using a kernel in conjuction with an SVM, input sequences are implicitly mapped into a high-dimensional vector space where the coordinates are given by feature values. The SVM produces a linear decision boundary in this high-dimensional feature space, and test sequences are classified based on whether they map to the positive or negative side of the boundary. The features used by our spectrum kernel are the set of all possible subsequences of amino acids of a fixed length k. If two protein sequences contain many of the same k-length subsequences, their "inner product" under the k-spectrum kernel will be large. The notion of the spectrum of a biological sequence – that is, the k-length subsequence content of the sequence – has been used for applications such as sequencing by hybridization [14] and is conceptually related to Fourier-based sequence analysis techniques [15].

We note that recently, Chris Watkins [16] and David Haussler [17] have defined a set of kernel functions over strings, and one of these string kernels has been implemented for a text classification problem [18]. However, the cost of computing each kernel entry is $O(n^2)$ in the length of the input sequences, making them too slow for most biological applications. Our spectrum kernel, with complexity $O(kn)$ to compute each k-spectrum kernel entry, is both conceptually simpler and computationally more efficient.

2 Overview of Support Vector Machines

Support Vector Machines (SVMs) are a class of supervised learning algorithms first introduced by Vapnik [19]. Given a set of labelled training vectors (positive and negative input examples), SVMs learn a linear decision boundary to discriminate between the two classes. The result is a linear classification rule that can be used to classify new test examples. SVMs have exhibited excellent generalization performance (accuracy on test sets) in practice and have strong theoretical motivation in statistical learning theory [19].

Suppose our training set $\mathcal{S}$ consists of labelled input vectors $(\mathbf{x}_i, y_i)$, $i = 1 \ldots m$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{\pm 1\}$. We can specify a linear classification rule f by a pair $(\mathbf{w}, b)$, where $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, via

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

where a point $\mathbf{x}$ is classified as positive (negative) if $f(\mathbf{x}) > 0$ ($f(\mathbf{x}) < 0$). Geometrically, the decision boundary is the hyperplane

$$\{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$$

where $\mathbf{w}$ is a normal vector to the hyperplane and b is the bias. If we further require that $|\mathbf{w}| = 1$, then the *geometric margin* of the classifier with respect to $\mathcal{S}$ is

$$m_{\mathcal{S}}^g(f) = \text{Min}_{\{\mathbf{x}_i \in \mathcal{S}\}} y_i f(\mathbf{x}_i).$$

In the case where the training data are linearly separable and a classifier f correctly classifies the training set, then $m_{\mathcal{S}}^g(f)$ is simply the distance from the decision hyperplane to the nearest training point(s).

The simplest kind of SVM is the maximal margin (or hard margin) classifier, which solves an optimization problem to find the linear rule f with maximal geometric margin. Thus, in the linearly separable case, the hard margin SVM finds the hyperplane that correctly separates the data and maximizes the distance to the nearest training points.

In pratice, training sets are usually not linearly separable, and we must modify the SVM optimization problem to incorporate a trade-off between maximizing geometric margin and minimizing some measure of classification error on the training set. See [20] for a precise formulation of various soft margin approaches.

3 Kernels in SVMs

A key feature of any SVM optimization problem is that it is equivalent to solving a dual quadratic programming problem. For example, in the linearly

separable case, the maximal margin classifier is found by solving for the optimal "weights" α_i, $i = 1 \ldots m$, in the dual problem:

$$\text{Maximize } \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

on the region: $\alpha_i \geq 0$ for all i

The parameters $(\mathbf{w}, b)$ of the classifier are then determined by the optimal α_i (and the training data). The dual optimization problems for various soft margin SVMs are similar.

The dual problem not only makes SVMs amenable to various efficient optimization algorithms, but also, since the dual problem depends only on the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, allows for the introduction of *kernel techniques.*

To introduce a kernel, we now suppose that our training data are simply labelled examples (x_i, y_i), where the x_i belong to an input space $\mathcal{X}$ which could be a vector space or a space of discrete structures like sequences of characters from an alphabet or trees. Given any feature map Φ from $\mathcal{X}$ into a (possibly high-dimensional) vector space called the *feature space*

$$\Phi : \mathcal{X} \to \mathbb{R}^N,$$

we obtain a kernel K on $\mathcal{X} \times \mathcal{X}$ defined by

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle.$$

By replacing $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ by $K(x_i, x_j)$ in the dual problem, we can use SVMs in feature space. Moreover, if we can directly compute the kernel values $K(x, y)$ without explicitly calculating the feature vectors, we gain tremendous computational advantage for high-dimensional feature spaces.

4   The Spectrum Kernel

For our application to protein classification, we introduce a simple string kernel, which we call the spectrum kernel, on the input space $\mathcal{X}$ of all finite length sequences of characters from an alphabet $\mathcal{A}$, $|\mathcal{A}| = l$.

Recall that, given a number $k \geq 1$, the k-spectrum of an input sequence is the set of all the k-length (contiguous) subsequences that it contains. Our feature map is indexed by all possible subsequences a of length k from alphabet $\mathcal{A}$. We define a feature map from $\mathcal{X}$ to $\mathbb{R}^{l^k}$ by

$$\Phi_k(x) = (\phi_a(x))_{a \in \mathcal{A}^k}$$

4

where
$$\phi_a(x) = \text{number of times } a \text{ occurs in } x$$

Thus the image of a sequence x under the feature map is a weighted representation of its k-spectrum. The k-spectrum kernel is then

$$K_k(x, y) = \langle \Phi_k(x), \Phi_k(y) \rangle.$$

For another variant of the kernel, we can assign to the a-th coordinate a binary value of 0 if a does not occur in x, 1 if it does occur.

Note that while the feature space is large even for fairly small k, the feature vectors are sparse: the number of non-zero coordinates is bounded by $\text{length}(x) - k + 1$. This property allows various efficient approaches for computing kernel values.

A very efficient method for computing $K_k(x, y)$ is to build a suffix tree for the collection of k-length subsequences of x and y, obtained by moving a k-length sliding window across each of x and y. At each depth-k leaf node of the suffix tree, store two counts, one representing the number of times a k-length subsequence of x ends at the leaf, the other representing a similar count for y. Note that this suffix tree has $O(kn)$ nodes. Using a linear time construction algorithm for the suffix tree [21], we can build and annotate the suffix tree in $O(kn)$ time. Now we calculate the kernel value by traversing the suffix tree and computing the sum of the products of the counts stored at the depth-k nodes. The overall cost of calculating $K_k(x, y)$ is thus $O(kn)$. One can use a similar idea to build a suffix tree for all the input sequences at once and to compute all the kernel values in one traversal of the tree. This is essentially the method we use to compute our kernel matrices for our experiments, though we use a recursive function rather than explicitly constructing the suffix tree.

There is an alternative method for computing kernel values that is less efficient but very easy to implement. For simplicity of notation, we describe the binary-valued version of the feature map, though the count-valued version is similar. For each sequence x, collect the set of k-length subseqences into an array A_x and sort them. Now the inner product $K_k(x, y)$ can be computed in linear time as a function of $\text{length}(x) + \text{length}(y)$. Thus the overall complexity of computing the kernel value is $O(n \log(n))$ in the length of the input sequences using this method.

5   Linear Time Prediction

The output of the SVM is a set of weights α_i that solve the dual optimization problem, where $i = 1 \ldots m$ for a set of m training vectors. Training vectors

$\mathbf{x}_i$ for which the corresponding weight α_i is non-zero are called support vectors. The parameters $(\mathbf{w}, b)$ of the classifier are determined by the weights and support vectors. Specifically, we have

$$\mathbf{w} = \sum_{\text{support vectors } \mathbf{x}_i} \alpha_i y_i \Phi(\mathbf{x}_i),$$

and in general there is also an expression for b, though in our experiments, we use a version of the SVM algorithm for which $b = 0$. Thus, in the case $b = 0$, test examples are classified by the sign of the expression

$$f(\mathbf{x}) = \Phi(\mathbf{x}) \cdot \mathbf{w} = \sum_{\text{support vectors } \mathbf{x}_i} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i).$$

For our spectrum kernel K_k, the normal vector $\mathbf{w}$ is given by

$$\mathbf{w} = \left(\sum_{\text{support vectors } \mathbf{x}_i} \alpha_i y_i \phi_a(x_i) \right)_{a \in \mathcal{A}^k}.$$

Note that typically the number of support vectors is much smaller that m, so that the number of non-zero coefficients in the expression for $\mathbf{w}$ is much smaller that mn. We store these non-zero coefficients in a look-up table, associating to each contributing k-length subsequence a its coefficient in $\mathbf{w}$:

$$a \to \sum_{\text{support vectors } \mathbf{x}_i} \alpha_i y_i \{\# \text{ times } a \text{ occurs in } x_i\}.$$

Now to classify a test sequence x in linear time, move a k-length sliding window across x, look up the current k-length subsequence in the look-up table, and increment the classifier value $f(x)$ by the associated coefficient.

6 Experiments: Protein Classification

We test the spectrum SVM method using an experimental design by Jaakkola et al. [10] for the remote homology detection problem. In this test, remote homology is simulated by holding out all members of a target SCOP [13] family from a given superfamily. Positive training examples are chosen from the remaining families in the same superfamily, and negative test and training examples are chosen from outside the target family's fold. The held-out family members serve as positive test examples. Details of the data sets are available at www.cse.ucsc.edu/research/compbio/discriminative.

Because the test sets are designed for remote homology detection, we use small values of k in our k-spectrum kernel. We tested $k = 3$ and $k = 4$, both for the unnormalized kernel K_k and for the normalized kernel given by

$$K_k^{\text{Norm}}(x, y) = \frac{K_k(x, y)}{\sqrt{K_k(x, x)}\sqrt{K_k(y, y)}}.$$

Our results show that a normalized kernel with $k = 3$ yields the best performance, although the differences among the four kernels are not large (data not shown).

We use a publicly available SVM software implemenation (www.cs. columbia.edu/compbio/svm), which implements the soft margin optimization algorithm described in [10]. Note that for this variant of the SVM optimization problem, the bias term b is fixed to 0. We did not attempt any fine-tuning of the soft margin SVM parameters.

We use ROC_{50} scores to compare the performance of different homology detection between methods. The ROC_{50} score is the area under the receiver operating characteristic curve – the plot of true positives as a function of false positives – up to the first 50 false positives [22]. A score of 1 indicates perfect separation of positives from negatives, whereas a score of 0 indicates that none of the top 50 sequences selected by the algorithm were positives.

For comparison, we include results from three other methods. These include the original experimental results from Jaakkola *et al.* for two methods: the SAM-T98 iterative HMM, and the Fisher-SVM method. We also test PSI-BLAST[3] on the same data. To approximate a family-based homology detection method, PSI-BLAST is run using a randomly selected training set sequence for one iteration, with the positive training set as a database and a very low E-value inclusion threshold. The resulting matrix is then used to search the test set for a maximum of 20 iterations using the default E-value inclusion threshold. The BLOSUM80 substitution matrix is used with PSI-BLAST.

The results for all 33 SCOP families are summarized in Figure 1. Each series corresponds to one homology detection method. Qualitatively, the SAM-T98 and Fisher-SVM methods perform slightly better than PSI-BLAST and the spectrum SVM. However, if we evaluate the statistical significance of these differences using a two-tailed signed rank test [23,24] (including a Bonferroni adjustment for multiple comparisons), only the SVM-Fisher method does better than any other method: SVM-Fisher's performance is better than that of PSI-BLAST with a p-value of 0.000045 and is better than that of the spectrum SVM with a p-value of 0.042. These results suggest that the spectrum SVM performs comparably with some state-of-the-art homology detection methods. In particular, the signed-rank comparison of PSI-BLAST and the spectrum
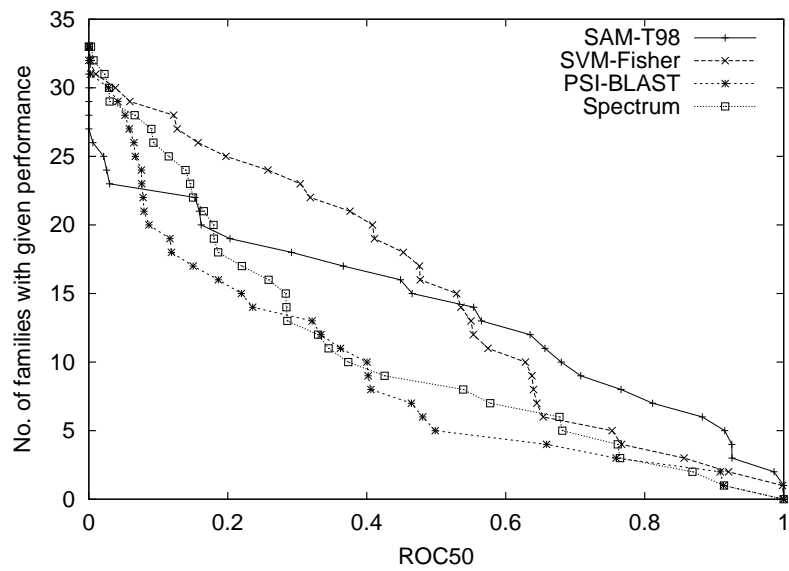
Figure 1: **Comparison of four homology detection methods.** The graph plots the total number of families for which a given method exceeds an ROC_{50} score threshold. Each series corresponds to one of the homology detection methods described in the text.
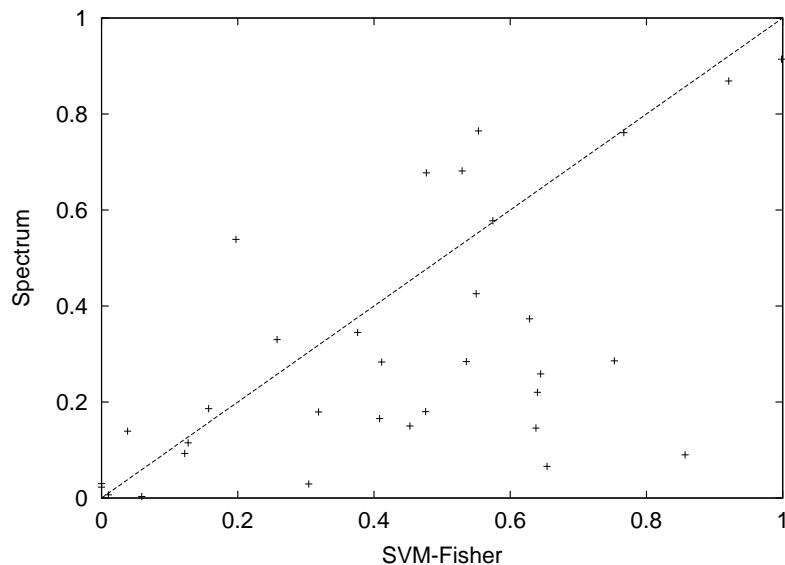
Figure 2: **Family-by-family comparison of the spectrum SVM and Fisher-SVM methods.** The coordinates of each point in the plot are the ROC_{50} scores for one SCOP family, obtained using the spectrum SVM and Fisher-SVM methods.

SVM gives a slight (though not significant) preference to the latter (unadjusted p-value of 0.16).

Figure 2 gives a more detailed view of the performance of the spectrum SVM with the Fisher-SVM method. Here we see clearly the optimization of the Fisher-SVM method for remote homology detection. For relatively easy-to-recognize families (i.e., families with high ROC_{50} scores), the two methods perform comparably; however, as the families become harder to recognize, the difference between the two methods becomes more extreme. Similar results are apparent in a comparison of Fisher-SVM and SAM-T98 (not shown), where SAM-T98 significantly out-performs Fisher-SVM for many of the easier families and vice versa.

7 Conclusions and Future Work

We have presented a conceptually simple, computationally efficient and very general approach to sequence-based classification problems. For the remote homology detection problem, we are encouraged that our discriminative approach

9

– combining support vector machines with the spectrum kernel – performed remarkably well in the SCOP experiments when compared with state-of-the-art methods, even though we used no prior biological knowledge specific to protein classication in our kernel. We believe that our experiments provide evidence that string-based kernels, in conjunction with SVMs, could offer a simple, effective and computationally efficient alternative to other methods of protein classification and homology detection.

There are several directions we plan to take this work. For improved performance in remote homology detection as well as for other discrimination problems – for example, classification problems involving DNA sequences – it should be advantageous to use larger values of k (longer subsequences) and incorporate some notion of mismatching. That is, we might want to change our kernel so that two k-length subsequences that are the same except for a small number of mismatched characters will, when mapped into feature space, have non-zero inner product. For protein classification, we would likely incorporate BLOSUM matrix information [25] into our mismatch kernel. We plan to implement an efficient data structure to enable us to calculate kernel values for a spectrum kernel that incorporates mismatching.

Secondly, in certain biological applications, the k-length subsequence features that are "most significant" for discrimination can themselves be of biological interest. For such problems, it would be interesting to perform feature selection on the set of k-spectrum features, so that we identify a feature subset that both allows for accurate discrimination and gives biologically interesting information about the spectrum differences between positive and negative examples. We are studying feature selection techniques in the context of SVMs, and we hope eventually to apply such techniques to the k-spectrum features.

1. MS Waterman, J Joyce, and M Eggert. *Computer alignment of sequences*, chapter Phylogenetic Analysis of DNA Sequences. Oxford, 1991.
2. SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. A basic local alignment search tool. *JMB*, 215:403–410, 1990.
3. SF Altschul, TL Madden, AA Schaffer, J Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.
4. M Gribskov, AD McLachlan, and D Eisenberg. Profile analysis: Detection of distantly related proteins. *PNAS*, pages 4355–4358, 1987.

5. A Bairoch. PROSITE: A dictionary of sites and patterns in proteins. *Nucleic Acids Research*, 19:2241–2245, 1991.

6. TK Attwood, ME Beck, DR Flower, P Scordis, and JN Selley. The prints protein fingerprint database in its fifth year. *Nucleic Acids Research*, 26(1):304–308, 1998.

7. A Krogh, M Brown, I Mian, K Sjolander, and D Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *JMB*, 235:1501–1531, 1994.

8. SR Eddy. Multiple alignment using hidden Markov models. In *ISMB*, pages 114–120. AAAI Press, 1995.

9. P Baldi, Y Chauvin, T Hunkapiller, and MA McClure. Hidden Markov models of biological primary sequence information. *PNAS*, 91(3):1059–1063, 1994.

10. T Jaakkola, M Diekhans, and D Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 2000.

11. T Jaakkola, M Diekhans, and D Haussler. Using the fisher kernel method to detect remote protein homologies. In *ISMB*, pages 149–158. AAAI Press, 1999.

12. CHQ Ding and I Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

13. AG Murzin, SE Brenner, T Hubbard, and C Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *JMB*, 247:536–540, 1995.

14. I Pe'er and R Shamir. Spectrum alignment: Efficient resequencing by hybridization. In *ISMB*, pages 260–268. AAAI Press, 2000.

15. D Anastassiou. Frequency-domain analysis of biomolecular sequences. *Bioinformatics*, 16:1073–1081, 2000.

16. C Watkins. Dynamic alignment kernels. Technical report, UL Royal Holloway, 1999.

17. D Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.

18. H Lodhi, J Shawe-Taylor, N Cristianini, and C Watkins. Text classification using string kernels. Preprint.

19. VN Vapnik. *Statistical Learning Theory*. Springer, 1998.

20. N Cristianini and J Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge, 2000.

21. E Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

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

23. S Henikoff and JG Henikoff. Embedding strategies for effective use of information from multiple sequence alignments. *Protein Science*, 6(3):698–705, 1997.

24. SL Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:371–328, 1997.

25. S Henikoff and JG Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89:10915–10919, 1992.