Mismatch string kernels for discriminative protein classification

Christina Leslie[1], Eleazar Eskin[1], Adiel Cohen[1], Jason Weston[3] and William Stafford Noble[2*]

[1]Department of Computer Science, Columbia University
[2]Max-Planck Institute, Tübingen, Germany
[3]Department of Genome Sciences, University of Washington

{cleslie,eeskin,adico}@cs.columbia.edu, weston@tuebingen.mpg.de
noble@gs.washington.edu

Abstract

Motivation Classification of proteins sequences into functional and structural families based on sequence homology is a central problem in computational biology. Discriminative supervised machine learning approaches provide good performance, but simplicity and computational efficiency of training and prediction are also important concerns.

Results We introduce a class of string kernels, called mismatch kernels, for use with support vector machines (SVMs) in a discriminative approach to the problem of protein classification and remote homology detection. These kernels measure sequence similarity based on shared occurrences of fixed-length patterns in the data, allowing for mutations between patterns. Thus the kernels provide a biologically well-motivated way to compare protein sequences without relying on family-based generative models such as hidden Markov models. We compute the kernels efficiently using a mismatch tree data structure, allowing us to calculate the contributions of all patterns occurring in the data in one pass while traversing the tree. When used with an SVM, the kernels enable fast prediction on test sequences. We report experiments on two benchmark SCOP data sets, where we show that the mismatch kernel used with an SVM classifier performs competitively with state-of-the-art methods for homology detection, particularly when very few training examples are available. Examination of the highest-weighted patterns learned by the SVM classifier recovers biologically important motifs in protein families and super-families.

Availability SVM software is publically available at http://microarray.cpmc.columbia.edu/gist. Mismatch kernel software is available upon request.

Contact cleslie@cs.columbia.edu

1 Introduction

One of the central problems in computational biology is the classification of protein sequences into functional and structural families based on sequence homology. Approaches based on pairwise similarity of sequences [27, 1, 2], profiles for protein families [11], consensus patterns using motifs [4, 3] and hidden Markov models [16, 9, 5] have all been used for this problem. Recent research suggests that the best-performing methods are discriminative: 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* a decision boundary between the different classes. In this category, several successful techniques [15, 19, 17] use protein sequences to train a support vector machine (SVM) [26, 8] classifier.

In this paper, we present a method for using SVMs for remote homology detection, based on a family of *kernel functions* called mismatch kernels. A kernel function measures the similarity between a pair of inputs, and defines an inner product in an implicit feature space for the SVM optimization problem. The features used by our mismatch kernel are the set of all possible subsequences of amino acids of a fixed length k. If two protein sequences contain many k-length subsequences that differ by at most m mismatches, then their inner product under the mismatch kernel will be large. More precisely, the mismatch kernel is calculated based on shared occurrences of (k, m)-patterns in the data, where the (k, m)-pattern generated by a fixed k-length subsequence consists of all k-length subsequences differing from it by at most m mismatches. Thus, the mismatch kernel extends the computationally simpler spectrum kernel presented in [17], adding the biologically important notion of mismatches.

We also describe the mismatch tree data structure that we use to compute the mismatch kernel. We can efficiently compute the contributions of all instances of (k, m)-patterns occurring in the data to all entries of the kernel matrix in one pass while traversing the tree. For (k, m)-parameters that are useful in applications, the compute time to generate the kernel is fast enough for practical use on real data sets. Moreover, when mismatch kernels are used with SVMs, we can implement the classification to make linear-time prediction on test sequences.

We report results for two sets of experiments over the SCOP database [21]. In the first set of experiments, we test our method on the benchmark dataset assembled by

---

[15], where SCOP sequences are augmented by domain homologs of positive training sequences in order to assist HMM-based methods. We show that our mismatch kernel, in conjunction with an SVM classifier, performs competitively with state-of-the-art methods such as the the SVM-Fisher method on this dataset. In the second set of experiments, we perform a similar set of experiments on a newer version of the SCOP database [19]; however, in this second test, we limit the training examples to proteins in the original SCOP dataset. In the absence of additional domain homologs for training the hidden Markov models, SVM-Fisher and other HMM-based approaches exhibit poorer performance than our mismatch-SVM approach. However, mismatch-SVM performs as well as SVM-pairwise, the best-performing method reported in [19] for this benchmark.

The current work is an expanded version of [18], which defined the mismatch kernel and presented results on the Jaakkola *et al.* dataset. Here, in addition to reporting experiments on the second benchmark dataset and compairing to the SVM-pairwise method, we present further results that give biological motivation for use of our method. First, we show that the mismatch kernel captures a biologically meaningful notion of sequence similarity by using the kernel values in an unsupervised setting for family-level and superfamily-level homology detection and for fold recognition. In an ROC analysis, we show that the mismatch kernels perform favorably with BLAST and Smith-Waterman for these tasks. Second, we describe how to extract from a trained mismatch-SVM classifier the most informative subsequences in the training set. These sequences typically correspond to highly conserved, motif regions in the positive sequence class. Thus, while we do not use a multiple alignment or HMM as an input to our method, we can recover conservation information as an output of our training.

2 SVMs and Kernels

Support Vector Machines (SVMs) are a class of supervised learning algorithms first introduced by Vapnik [6, 26]. Given a set of labeled training vectors $(\mathbf{x}_i, y_i)$, $i = 1 \ldots m$, where the $\mathbf{x}_i$ are real vectors and y_i are ± 1, training an SVM amounts to solving an optimization problem that determines a linear classification rule $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. A test example $\mathbf{x}$ is then classified as positive (negative) if $f(\mathbf{x}) > 0$ ($f(\mathbf{x}) < 0$). Such a classification rule corresponds to a hyperplane decision boundary between positive and negative points with normal vector $\mathbf{w}$ and bias term b.

A key feature of the SVM optimization problem is that it depends only on the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ of the training vectors, allowing us to use *kernel techniques*. To introduce a kernel, we now suppose that our training data are simply labeled examples (x_i, y_i), where the x_i could belong to a vector space or to a space of discrete structures like sequences of characters from an alphabet. Given any feature map Φ from the input space into a (possibly high-dimensional) vector space called the *feature space*, we obtain a kernel 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 and efficiently compute the kernel values $K(x, y)$ without explicitly representing the feature vectors, we gain tremendous computational advantage for high-dimensional feature spaces.

3 The Mismatch Kernel

Below we introduce a class of *string kernels*, called mismatch kernels, that can be used on biological sequence data with various kernel-based machine learning methods. Our eventual goal is to use these kernels to train support vector machines for protein classification: the use of kernels in SVM training is outlined in Section 5, efficient computation of the kernel is explained in Section 4, and our application of mismatch kernels in an SVM approach to remote homology detection is described in Section 6.

In this section, we define the mismatch kernels and provide some intuition for why they capture meaningful sequence similarity information. For motivation, we interpret the mismatch value for a pair of protein sequences as a sequence similarity score, and as a simple application, we show that this score can be used directly for protein classification.

3.1 Mapping Protein Sequences to k-mer Feature Space

Recall that, given a number $k \geq 1$, the k-spectrum of a biological sequence is the set of all the k-length (contiguous) subsequences that it contains; we refer to such a k-length subsequence as a k-mer. In order to capture significant information from sequence data without first building a generative model, we represent the spectrum information in a sequence as a vector in "k-mer feature space".

If we are dealing with sequences of characters from an alphabet $\mathcal{A}$ of size $|\mathcal{A}| = l$ ($l = 20$ for the alphabet of amino acids), we represent the sequences as vectors in an l^k-dimensional vector space, or *feature space*, where the coordinates are indexed by the set of all possible k-mers. For a very simple feature map, we can assign to a sequence x a vector given as follows: for each k-mer α, the coordinate indexed by α will be the number of times α occurs in x. This gives the k-spectrum feature map defined in [17]:

$$\Phi_k(x) = (\phi_\alpha(x))_{\alpha \in \mathcal{A}^k} \qquad (1)$$

where $\phi_\alpha(x) = \#$ occurrences of α in x. Now the k-spectrum kernel $K(x, y)$ for two sequences x and y is obtained by taking the inner product in feature space:

$$K_k(x, y) = \langle \Phi_k(x), \Phi_k(y) \rangle \qquad (2)$$

Note that this kernel gives a simple notion of sequence similarity: two sequences will have a large k-spectrum kernel value if they share many of the same k-mers.

For a more sensitive and biologically realistic kernel, we want to allow some degree of mismatching in our feature map. That is, we want the kernel value between two sequences x and y to be large if they share many *similar* k-mers. For a fixed k-mer $\alpha = a_1 a_2 \ldots a_k$, with each a_i a character in $\mathcal{A}$, the (k, m)-pattern generated by α is the set of all k-length sequences β from $\mathcal{A}$ that differ from α by at most m mismatches. We denote this set by $N_{(k,m)}(\alpha)$, the "mismatch neighborhood" around α. Intuitively, when we see an instance of a k-mer α in our input sequence x, we would like it to contribute not only the α-coordinate in feature space but also to all coordinates corresponding to k-mers in the mismatch neighborhood of α.

We can now define our feature map into the l^k-dimensional feature space, indexed as before by the set of all possible k-mers. If α is a fixed k-mer, we first define the $\Phi_{(k,m)}$ on α by:

$$\Phi_{(k,m)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in \mathcal{A}^k} \qquad (3)$$

where $\phi_\beta(\alpha) = 1$ if β belongs to $N_{(k,m)}(\alpha)$ (or equivalently, if α belongs to $N_{(k,m)}(\beta)$), and otherwise $\phi_\beta(\alpha) = 0$.

We define the feature map on an input sequence x in $\mathcal{X}$ is as the sum of the feature vectors for the k-mers in x:

$$\Phi_{(k,m)}(x) = \sum_{k\text{-mers } \alpha \text{ in } x} \Phi_{(k,m)}(\alpha) \qquad (4)$$

The (k, m)-mismatch kernel is once again just the inner product in feature space[1]:

$$K_{(k,m)}(x, y) = \langle \Phi_{(k,m)}(x), \Phi_{(k,m)}(y) \rangle. \qquad (5)$$

Note that these kernels measure sequence similarity based on shared occurrences of (k, m)-patterns in the data: the kernel value $K_{(k,m)}(x, y)$ will be large if the sequences x and y share many k-length subsequences differing by at most m mismatches.

3.2 A Simple Application: ROC Analysis of Protein Classification

As a simple application to demonstrate that our kernels capture useful biological information about sequence similarity, we use the spectrum kernel $K_k(x, y)$ and mismatch kernel $K_{(k,m)}(x, y)$ to derive distance measures between protein sequences x and y, and we use these kernel-derived distances for homology detection. A receiver operating characteristic (ROC) analysis shows favorable comparison with BLAST [2] for certain classification problems. Here our

---

[1]Note that we since we have defined $K_{(k,m)}$ from an explicit feature map, it is automatically a valid kernel function, and there is no need, for example, to show that the conditions of Mercer's Theorem apply.
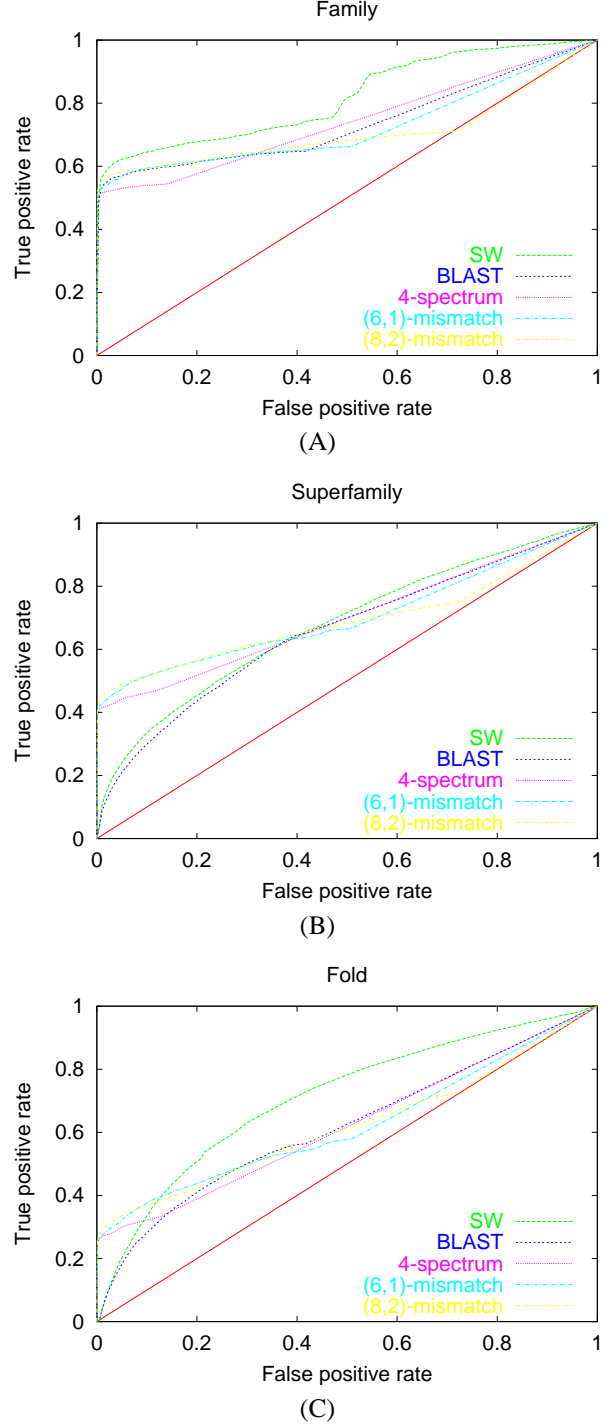

Figure 1: ROC curves comparing mismatch kernels with SW and BLAST for unsupervised homology detection. In each plot, "true" relationships are defined by SCOP (A) families, (B) superfamilies, and (C) folds.

purpose is simply to motivate our use of mismatch kernels in the SVM experiments described later in the paper; we do not claim that mismatch kernels are more appropriate or effective than BLAST for sequence similarity.

We first normalize the kernels as described in Section 6 and consider the induced distance $d(x,y) = \sqrt{K(x,x) - 2K(x,y) + K(y,y)}$ between pairs of protein sequences x and y. We wish to see whether the kernel-induced pairwise distance $d(x,y)$ can be used directly to predict whether x and y belong to the same protein family, superfamily, or fold.

We evaluate the performance of this distance for homology detection and compare with the performance of Smith-Waterman and BLAST over the same dataset. Sequences for these experiments were extracted from the Structural Classification of Proteins (SCOP) [21] version 1.53 using the Astral database (astral.stanford.edu, [7]), removing similar sequences using an E-value threshold of 10^{-25}. This procedure resulted in 4352 distinct sequences, grouped into families, superfamilies, and folds. All pairwise E-values are computed by querying the database of 4352 iteratively with each sequence.

We perform separate experiments to detect family-level and superfamily-level homology and fold-membership using Smith-Waterman, BLAST, the spectrum kernel with $k = 4$, and the mismatch kernels with the following (k, m) parameters $(6, 1)$ and $(8, 2)$. In experiments using kernel-derived distances, we compute ROC curves by varying the distance threshold: a pair of homologous proteins that is below the threshold is considered a true positive, while a pair of non-homologous proteins below the threshold is considered a false positive. We also compute an ROC curve for Smith-Waterman and BLAST using E-value thresholds.

Figure 1 shows ROC curves for all methods in the family, superfamily, and fold detection tasks. Table 1 summarizes the ROC scores for all methods at each task. The ROC score is the area under the ROC curve: perfect discrimination between positives and negatives gives an ROC score of 1, while a random classifier has an expected score close to 0.5. We first note that in this unsupervised setting, neither the kernel-induced distances nor the E-values from BLAST and Smith-Waterman give very strong performance in any of the homology/fold detection tasks. However, for family-level homology detection, the best mismatch kernel performs better than BLAST and almost as well as Smith-Waterman. For the superfamily homology and fold recognition tasks, the mismatch kernels outperform both BLAST and Smith-Waterman at low false positive rates, though Smith-Waterman is more sensitive at fold detection at higher false positive rates. In all tasks, the mismatch-induced distance gives stronger performance than the spectrum-induced distance at low false positive rates.

	SW	BLAST	Spectrum k=4	Mismatch k=6,m=1	Mismatch k=8,m=2
fold	0.713	0.619	0.623	0.623	0.621
superfamily	0.679	0.660	0.702	0.704	0.695
family	0.820	0.737	0.741	0.784	0.708

Table 1: ROC scores for homology experiments comparing kernels to BLAST. The ROC score is the area under the ROC curve.

4 Efficient Computation of Kernel Matrix

Since our feature vectors are sparse vectors in a very high dimensional feature space, we do not compute and store the feature vectors for our input sequences. Instead, we directly and efficiently compute the kernel matrix using a data structure called a *mismatch tree*.

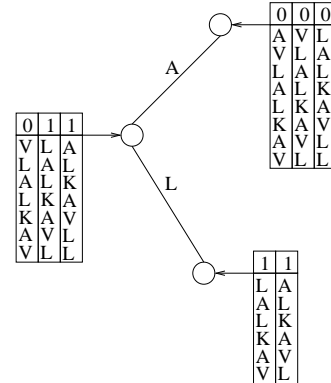## 4.1 Mismatch Tree Data Structure

Figure 2: An $(8, 1)$-mismatch tree for a sequence AVLALKAVLL used for computing the kernel matrix with k-mers of length 8 allowing 1 mismatch. The path from the root to the node is the "prefix" of a particular k-mer feature. The leaf nodes contain (i) the number of mismatches between the prefix of a substring in the sequence and the prefix of a feature and (ii) a pointer to the tail of the substring. The tree is in its initial state containing just the root node. The figure shows the tree after expanding the path AL. Many of the instances reach the maximum number of allowed mismatches and are not passed down to their child nodes.

The mismatch tree data structure is similar to a trie or suffix tree [12]. We use the mismatch tree to represent the feature space (the set of all k-mers) and to organize a lexical traversal of all instances of k-mers that occur (with mismatches) in the data. The entire kernel matrix is computed in one traversal of the tree. Our algorithm is similar to the approach presented in [24, 22] for finding subsequence patterns that occur with mismatches. A related data structure was also used for sparse prediction trees [10, 23].

A (k, m)-mismatch tree is a rooted tree of depth k where each internal node has 20 (more generally, l) branches, each

labeled with an amino acid (symbol from $\mathcal{A}$). A leaf node represents a fixed k-mer in our feature space, obtained by concatenating the branch symbols along the path from root to leaf. An internal node represents the prefix for those k-mer features which are its descendants in the tree. We perform a depth-first traversal of the data structure and store, at a node of depth d, n pointers to all substrings ("k-mer instances") from the sample data set whose d-length prefixes are within m mismatches from the d-length prefix represented by the path down from the root; this set of substrings represents the valid instances of the d-length prefix in the data. We also keep track, for each valid instance, how many mismatches it has when compared to the prefix. Note that the set of valid instances for a node is a subset of the set of valid instances for the parent of the node; when we descend from a parent to a child, each instance is either passed down (with 0 or 1 additional mismatch), or it is eliminated because it has exceeded m mismatches. When we encounter a node with an empty list of pointers (no valid occurrences of the current prefix), we do not need to search below it in the tree. When we reach a leaf node – corresponding to a particular feature k-mer α – we have pointers to all instance k-mers occurring in the source sequences that are up to m mismatches from α. Since for a source sequence x, the instances with mismatches of α in x — the k-mers in x belonging to $N_{(k,m)}(\alpha)$ — are exactly the ones that contribute to the α-coordinate of the feature vector $\Phi(x)$, we can now sum the contributions of all instances occurring in each source sequence and update the kernel matrix entries $K(x, y)$ for those source sequences x and y having non-zero α-features.

An example of the mismatch traversal is given in Figure 2.

4.2 Efficiency of kernel computation

During the kernel computation, we need only search down paths corresponding to k-mers that occur (with mismatches) in the data. The number of k-mers within m mismatches of any given fixed k-mer is $p(k, m, l) = \sum_{i=0}^{m} \binom{k}{i} (l-1)^i = O(k^m l^m)$. Thus the effective number of k-mer instances that we need to traverse grows as $O(Nk^m l^m)$, where N is the total length of the sample data. The running time of the algorithm is dominated by the kernel updates made at the leaf nodes. If exactly c input sequences contain valid instances of the current k-mer at a particular leaf node, one performs c^2 updates to the kernel matrix. One can then show that for M sequences each of length n (total length $N = nM$), the worst case for the kernel computation occurs when the M feature vectors are all equal and have the maximal number of non-zero entries, so that the worst case running time for the full kernel computation is $O(M^2 np(k, m, l)) = O(M^2 nk^m l^m)$. Thus, the kernel computation worst case running time grows with m, the number of mismatches, at the same rate as the mismatch

neighborhood and scales *linearly* with the length n of the input sequences. For the application to remote homology detection, small values of m are most useful, and the kernel calculations are quite inexpensive.

Another advantage of the mismatch algorithm is the efficient use of memory, which also leads to faster running time in practice. Since we perform a depth-first traversal, when we backtrack in the tree, we collapse the current node and expand the next node. Thus the only expanded nodes are along the current search path, there is a maximum of k stored nodes in the tree (counting the root node). In fact, the kernel computation can be achieved by a recursive function, without explicitly building and storing the tree.

5 Fast Prediction with SVMs

Mismatch kernels provide an additional advantage when used with SVMs: the particular form of the SVM solution combined with the definition of the mismatch feature map allow us to implement fast prediction on test sequences. The learned SVM classifier is given by

$$f(x) = \sum_{i=1}^{r} y_i a_i \langle \Phi_{(k,m)}(x_i), \Phi_{(k,m)}(x) \rangle + b$$

where x_i are the training sequences that map to support vectors, y_i are labels (± 1), and a_i are weights obtained from the dual SVM optimization problem. Note that the classification function evaluated on the test sequence x is the sum of classification "scores" $f(\alpha)$ for the k-mers α it contains. We can therefore pre-compute and store all the non-zero k-mer scores. Then the prediction $f(x)$ can be calculated in linear time by look-up of k-mer scores.

One way to compute the k-mer scores is to use two passes of the mismatch tree data structure. In the first pass, we compute the non-zero coordinates of the normal vector $w = \sum_{i=1}^{r} y_i a_i \Phi_{(k,m)}(x_i)$. We traverse the support sequences x_i, and at leaf node corresponding to k-mer β, we compute the weighted sum of valid instances to obtain the coordinate w_β of the normal vector. In the second pass, we use the normal coordinates w_β to obtain the k-mer scores. We traverse the set of k-mers β having non-zero w_β, and at leaf node corresponding to k-mer α, we compute the sum $\sum_{\beta \text{ in } N_{(k,m)}(\alpha)} w_\beta$ to obtain the score for α.

In practice, one usually wants to use a normalized feature map, so one would also need to compute the norm of the vector $\Phi_{(k,m)}(x)$, with running time $O(nk^m l^m)$ for a sequence of length n. Simple $O(1)$ normalization schemes, like dividing by sequence length, can also be used.
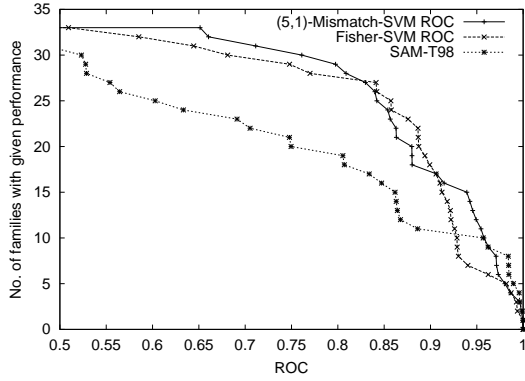
Figure 3: **Comparison of three homology detection methods for the SCOP 1.37 benchmark dataset.** The graph plots the total number of families for which a given method exceeds an ROC score threshold. Each series corresponds to one of the homology detection methods described in the text.
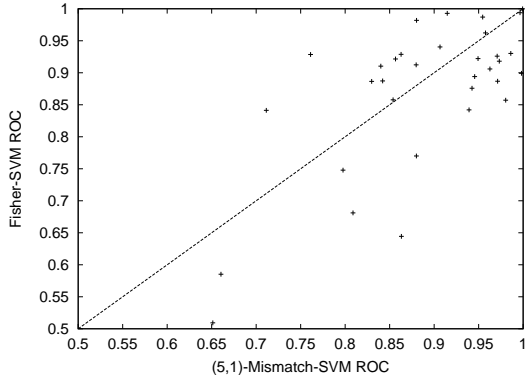

Figure 4: **Family-by-family comparison of mismatch and Fisher kernels for SCOP 1.37 benchmark dataset.** The coordinates of each point in the plots are the ROC scores for one SCOP family, obtained using the mismatch-SVM with $k = 5$, $m = 1$ (x-axis) and Fisher-SVM (y-axis). The dotted line is $y = x$.

6 Experiments: Protein Classification

6.1 SCOP Experiments with Domain Homologs

We first test the mismatch SVM method using the SCOP ([21], version 1.37) data sets designed by [15] for the remote homology detection problem. In this test, remote homology is simulated by holding out all members of a target SCOP 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 disjoint sets of folds 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.soe.ucsc.edu/research/compbio/
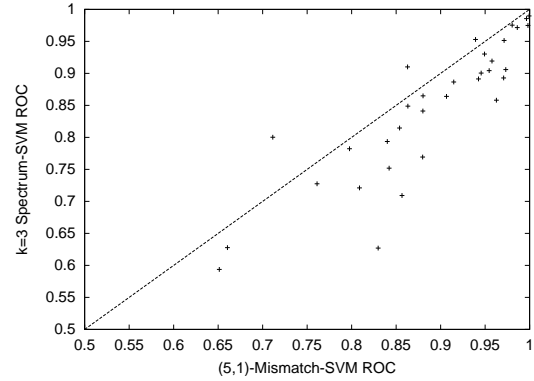

Figure 5: **Family-by-family comparisons mismatch and spectrum kernels for SCOP 1.37 benchmark dataset.** The coordinates of each point in the plots are the ROC scores for one SCOP family, obtained using mismatch-SVM with $k = 5$, $m = 1$ (x-axis) and spectrum-SVM with $k = 3$ (y-axis). The dotted line is $y = x$.

discriminative.

Because Jaakkola *et al.* needed to train hidden Markov models for the protein families represented in the positive training sets for these experiments, they used the SAM-T98 algorithm to pull in domain homologs from the non-redundant protein database. These additional domain homologs were added to the data set as positive examples in the experiments.

Because the test sets are designed for remote homology detection, we use small values of k. We tested $(k, m) = (5, 1)$ and $(6, 1)$, where we normalized the kernel via $K^{\mathrm{Norm}}(x, y) = \dfrac{K(x, y)}{\sqrt{K(x, x)}\sqrt{K(y, y)}}$. Our results show that $(k, m) = (5, 1)$ yields slightly better performance, though results for both choices were similar. (Data for $(k, m) = (6, 1)$ not shown.) We use a publicly available SVM software implementation (http://microarray.cpmc.columbia.edu/gist/), which implements the soft margin optimization algorithm described in [15]. Note that for this variant of the SVM optimization problem, the bias term b is fixed to 0.

For comparison, we include the original experimental results from Jaakkola *et al.* for two methods: the SAM-T98 iterative HMM, and the SVM-Fisher method. Figure 3 illustrates the mismatch kernel's performance relative to these existing homology detection methods. The figure includes results for all 33 SCOP families, and each series corresponds to one homology detection method. Qualitatively, the curves for SVM-Fisher and mismatch-SVM are quite similar. When we compare the overall performance of two methods using a two-tailed signed rank test [14, 25], we find that almost none of the differences between methods are statistically significant. Using a p-value threshold of 0.05 and including a Bonferroni adjustment to account for multiple comparisons, we find only the following significant differences: Fisher-SVM and mismatch-SVM perform better

than SAM-T98 (with p-values 6.7e-03 and 1.3e-02, respectively). There is no statistically significant difference between the performance of Fisher-SVM and mismatch-SVM.

Figure 4 shows a family-by-family comparison of the ROC scores of the $(5, 1)$-mismatch-SVM and SVM-Fisher. The points fall approximately evenly above and below the diagonal, indicating little difference in performance between the two methods, though there are a handful of families on which SVM-Fisher has somewhat higher ROC scores.

Figure 5 shows the improvement provided by including mismatches in the SVM kernel. The figure plots ROC scores for two kernel SVM methods: using $k = 5$, $m = 1$ mismatch kernel, and using $k = 3$ spectrum kernel (no mismatches), the best-performing spectrum kernel reported in [17]. Most of the families perform better with mismatching than without, and in the cases where the 3-spectrum kernel wins over the $(5, 1)$-mismatch kernel, the difference is usually small.

6.2 SCOP Experiments without Domain Homologs

In order to investigate the performance of our method against competing methods in the more difficult setting of very limited positive training data, we completed a similar set of SCOP experiments described in [19]. Liao and Noble developed this benchmark dataset (available at www.cs.columbia.edu/compbio/svm-pairwise) to test their SVM-pairwise method, which uses explicit feature vectors based on Smith-Waterman alignment scores against the set of training sequences for SVM training. These experiments are based on SCOP version 1.53 and also test for remote homology detection, but the positive training data is limited to the original SCOP sequences. In particular, no method has the benefit of training on additional homologs of the positive training sequences pulled in from a large unlabeled protein database. For this more difficult discrimination problem, Liao and Noble found that their SVM-pairwise method outperformed SVM-Fisher, SAM-T98, and PSI-BLAST; these methods could not successfully perform in the presence of so little positive training data.

In our results on this dataset, we find that the our SVM-mismatch method performs as well as SVM-pairwise: Figure 7 shows the comparison of the (5,1)-mismatch kernel against SVM-pairwise. A signed-rank test confirms that there is no significant difference between performance. Note, however, that the mismatch method has the advantage that its kernel is fast and efficient to compute, while the SVM-pairwise method requires expensive computation of Smith-Waterman alignment scores for every pair of training sequences and between each training and test sequence (computation of each feature is quadratic in sequence length). Also, the computation of the kernel matrix from the Smith-Waterman matrix is $O(M^3)$ in the number of training examples. We did not test SVM-pairwise

on the Jaakkola dataset because the large number of experiments and training sequences make computation of the SVM-pairwise kernel prohibitively expensive.

One can argue that this set of experiments is an unrealistic comparison of methods, since one would generally not attempt to use hidden Markov models, either directly or in the SVM-Fisher approach, that had been trained on a limited and divergent set of positive examples. However, we view these experiments as a demonstration of the power of our model-independent discriminative approach in a difficult discrimination problem with limited training data.
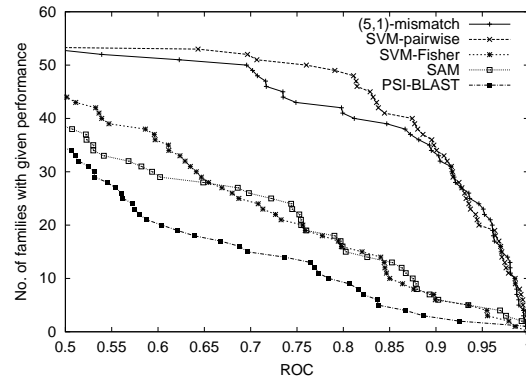
6.3 Extracting Explanations for SVM Predictions

Figure 6: **Comparison of homology detection methods for the SCOP 1.53 benchmark dataset.** The graph plots the total number of families for which a given method exceeds an ROC score threshold for the second set of remote homology detection experiments. Each series corresponds to one of the homology detection methods described in the text.
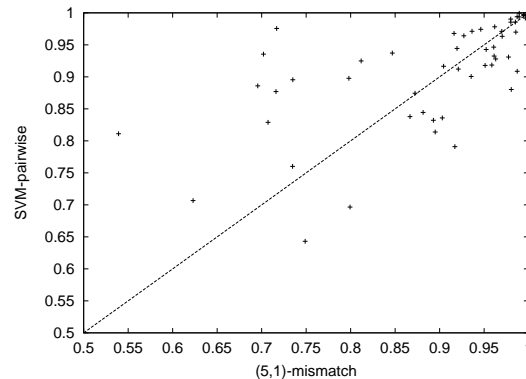

Figure 7: **Family-by-family comparison of** $(5, 1)$**-mismatch-SVM with SVM-pairwise for the SCOP 1.53 benchmark dataset.** The coordinates of each point in the plot are the ROC scores for one SCOP family, obtained using the SVM-mismatch with $k = 5$, $m = 1$ (x-axis) and SVM-pairwise (y-axis). The dotted line is $y = x$.

In general, SVM classifiers do not easily yield explanations for the predictions that they make. In particular,

when a kernel function is used to map the data into a high-dimensional feature space, the hyperplane decision boundary located by the SVM training algorithm is only implicitly defined by dot products between images of data points in the feature space. Most kernel functions make it very difficult to compute explicitly the location of this hyperplane.

However, the mismatch kernel differs from most kernel functions in that the mismatch kernel's mapping into feature space is explicit (as in Equation 4). Therefore, for this kernel, the normal of the separating hyperplane $\mathbf{w}$ can be represented directly as $\mathbf{w} = \sum_i y_i a_i \Phi_{(k,m)}(x_i)$, where each x_i is a training example, y_i its training label, and a_i its corresponding weight obtained by the SVM algorithm. Note that $a_i > 0$ only for a subset of training examples, which are referred to as support vectors. The location of the hyperplane $\mathbf{w}$ is thus given by a set of weights, one for each k-mer. Using the mismatch data structure we can efficiently compute these k-mer weights by summing the support vector weights of all instances at each leaf node.

A sequence that contains many k-mers with large, positive weights is very likely to be a member of the positive class defined by the classifier. Therefore, in the case of protein classification, we expect these highly informative k-mers to correspond to conserved regions in the protein family alignment. Figures 8 and 9 illustrates this phenomenon. For a single protein superfamily (the EF-hand superfamily) from the experiments in Section 6.1, we computed the 200 k-mers that are assigned the highest weights by the (5,1)-mismatch SVM. Many of these k-mers are overlapping or slight variations of one another. The resulting set of 200 k-mers was then mapped onto the positive training set. Figure 9 illustrates the motif-like nature of these mappings. In Figure 9(A), five sequences from the multiple alignment are shown in detail, with the high-scoring k-mers indicated by uppercase letters. In these five sequences, the high-scoring k-mers all appear within a nine-letter region of the alignment. Note that this is the only nine-letter block within the alignment that does not contain any insertions or deletions. Figure 9(B) shows a larger portion of the same alignment, in schematic form. Here, the motif-like nature of the clusters of k-mer occurrences is apparent as vertical black bars in the alignment.

This analysis shows that, although the mismatch SVM does not provide sequence alignments per se, the classifier can be used to identify sequence regions that are highly conserved in the positive class.

7 Discussion

We have presented a class of string kernels that measure sequence similarity without requiring alignment and without depending upon a generative model. We have also presented a method for efficiently computing these kernels. For the remote homology detection problem, we are encouraged
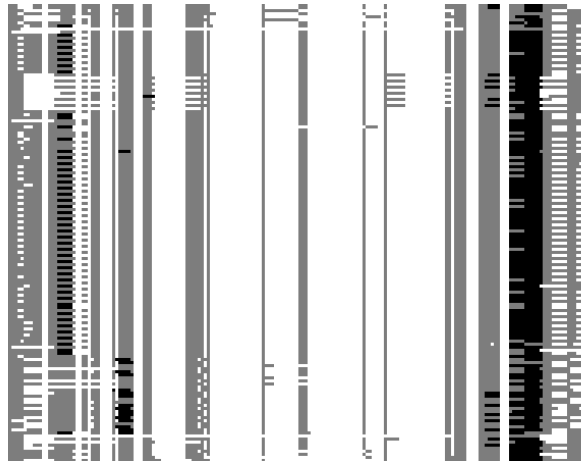

Figure 9: **Schematic of k-mers mapped onto a multiple alignment.** The figure shows the first 150 sequences from the same alignment as Figure 8, in schematic form. In the figure, insertions and deletions are white, unmatched amino acids are grey, and amino acids that map onto one of the top 200 k-mers are black. The entire alignment (not shown) contains 847 sequences.

that our discriminative approach — combining support vector machines with the mismatch kernel — performs competitively in the SCOP experiments when compared with state-of-the-art methods. When we make the discrimination problem more difficult by providing only limited training data (in particular, when no domain homologs of positive training sequences are used), our method performs as well SVM-pairwise, an approach that previously outperforms other known methods in this setting.

Our method has several advantages over other SVM approaches to protein classification, such as SVM-Fisher and SVM-pairwise. The SVM-Fisher method is appealing because it combines the rich biological information encoded in a profile hidden Markov model with the discriminative power of the support vector machine algorithm. However, one generally needs a lot of data or prior knowledge to train the hidden Markov model. In addition, because calculating the Fisher scores depends on dynamic programming (quadratic in sequence length for profile HMMs), in practice it is very expensive to compute the kernel matrix. Our mismatch kernel approach gives efficient kernel computation, linear time prediction, and maintains good performance even when there is little training data. Hence, our method could be advantageous in a number of settings: for training on large datasets, where kernel evaluations are the bottleneck; in applications where fast prediction is important, such as web-based tools; and in cases where positive examples are scarce and a good model is unavailable.

The SVM-pairwise method has a different computational problem, in that the dimension of the feature space scales with the number of training sequences used, which typi-

```
1cb1_4:78            saq..---ksp..aelksifek..yaakeg....dpnqlsk...eel...........kqliqaef..................p...........
SWN:CA22_HUMAN_116:179  snklh------.----fafrl..yd-ldk....d-ekisr...del...........lqvlrmmv.................gvnisdeqlgsi
SWN:S106_CHICK_4:82  maa..pldqai..gllvatfhk..ysgkeg....dknslsk...gel...........keliqkelti...............g...........
SW:BTV3_BETVE_140:197  sfggf------.----kvfd-..---edg....d-gyisa...rel...........qmvlgk-l.................g...........
SW:CA22_RAT_116:179  snklh------.----fafrl..yd-ldk....d-dkisr...del...........lqvlrmmv.................gvnisdeqlgsi


                                                                                          *********
1cb1_4:78            sll.................k......g...................prtlddl....fqeldkn...gdgevSFEEFQvlvkkisq......
SWN:CA22_HUMAN_116:179  ---.................-......-...................---adrt....iqeadqd...gdsaiSFTEFVkvl-----ekvdv.
SWN:S106_CHICK_4:82  pkl.................k......d...................ae-iagl....medldrn...kdqeVNFQEyvtflgalamiynea.
SW:BTV3_BETVE_140:197  ---................-fsegs.e...................idrvekm....ivsvdsn...rdGRVDFFEFkdmm-----rsvlv.
SW:CA22_RAT_116:179  ---.................-......-...................---adrt....iqeadqd...gdsaiSFTEFVkvl-----ekvdv.
```

Figure 8: **Mapping high-scoring k-mers onto a multiple alignment.** The figure shows the first five sequences from a profile HMM multiple alignment of the EF-hand superfamily and its homologs. The 200 k-mers that receive the highest weights from a (5,1)-mismatch SVM have been mapped onto the multiple alignment. Occurrences of these k-mers are indicated by uppercase letters. Periods (".") in the alignment correspond to insertions, and hyphens ("-") correspond to deletions with respect to the HMM. The location of a motif-like region of clustered k-mer occurrences is indicated by asterisks.

cally would be very large. (Each kernel computation also depends quadratically on sequence length when pairwise Smith-Waterman scores are used as features.) For example, computation of the kernel matrix is cubic in the number of sequences. It is possible that a smaller sample of features could be used effectively in this method. However, it is difficult to gain any biological insight about protein sequences from the SVM-pairwise classifier. By contrast, one can extract high-scoring k-mers from a trained SVM-mismatch classifier in order to look for discriminative motif regions in the positive sequence family. Therefore, by using mismatch kernels, we can relate the SVM classification rule to conserved subsequences in remotely related protein sequences.

An important characteristic of any protein classification algorithm is its computational efficiency. Recently, Chris Watkins [28] and David Haussler [13] have defined a set of kernel functions over strings, and one of these string kernels has been implemented for a text classification problem [20]. 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. The (k, m)-mismatch kernel for a pair of length n sequences can be computed in the worst case in $O(nk^m l^m)$, where l is the size of the alphabet. Since small values of m, the number of mismatches, are most useful in biological applications, the worst case running time is feasible. In the SCOP experiments, we found choices of (k, m) that allowed for both reasonable compute times and good discriminative performance when used with an SVM.

In future work, we will revisit the way in which we enforce a threshold on instance k-mers that contribute to a particular feature. Currently, we use the notion of a (k, m)-pattern generated by a feature k-mer α, but alternatively, we could use features with wildcards or establish a probabilistic threshold by introducing a substitution probability score relative to each feature k-mer. We will investigate how to establish a threshold that guarantees reasonable compute times.

Secondly, in certain biological applications, the k-length subsequence features that are "most significant" for discrimination can themselves be of biological interest, as shown in Section 6.3. 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 hope to exploit the mismatch tree data structure to perform meaningful feature selection for SVM discrimination.

References

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

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

[3] T. K. Attwood, M. E. Beck, D. R. Flower, P. Scordis, and J. N Selley. The prints protein fingerprint database in its fifth year. *Nucleic Acids Research*, 26(1):304–308, 1998.

[4] A. Bairoch. The prosite database, its status in 1995. *Nucleic Acids Research*, 24:189–196, 1995.

[5] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden markov models of biological primary sequence information. *PNAS*, 91(3):1059–1063, 1994.

[6] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[7] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

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

[9] S. R. Eddy. Multiple alignment using hidden markov models. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 114–120. AAAI Press, 1995.

[10] E. Eskin, W. N. Grundy, and Y. Singer. Protein family classification using sparse Markov transducers. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 134–145, Menlo Park, CA, 2000. AAAI Press.

[11] Michael Gribskov, Andrew D. McLachlan, and David Eisenberg. Profile analysis: Detection of distantly related proteins. *PNAS*, pages 4355–4358, 1987.

[12] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, 1997.

[13] D. Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.

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

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

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

[17] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Proceedings of the Pacific Biocomputing Symposium*, 2002.

[18] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. *Neural Information Processing Systems*, 2002.

[19] C. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, 2002. To appear.

[20] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. Preprint.

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

[22] G. Pavesi, G. Mauri, and G. Pesole. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 17:S207–S214, July 2001. Proceedings of the Ninth International Conference on Intelligent Systems for Molecular Biology.

[23] F. Pereira and Y. Singer. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36:183–199, 1999.

[24] M. Sagot. Spelling approximate or repeated motifs using a suffix tree. *Lecture Notes in Computer Science*, 1380:111–127, 1998.

[25] S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:371–328, 1997.

[26] V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.

[27] M. S. Waterman, J. Joyce, and M. Eggert. *Computer alignment of sequences*, chapter Phylogenetic Analysis of DNA Sequences. Oxford, 1991.

[28] C. Watkins. Dynamic alignment kernels. Technical report, UL Royal Holloway, 1999.