

Semi-supervised protein classification using cluster kernels

Jason Weston^{1,*}, Christina Leslie², Eugene Ie², Dengyong Zhou³,
Andre Elisseeff³ and William Stafford Noble⁴

¹NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA, ²Center for Computational Learning Systems, Columbia University, Interchuch Center, 475 Riverside Drive, Mail Code 7717, New York, NY 10115, USA, ³Max-Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany and ⁴Department of Genome Sciences, University of Washington, 1705 NE Pacific Street, Seattle, WA 98195, USA

Received on January 2, 2005; revised on April 26, 2005; accepted on May 12, 2005

Advance Access publication May 19, 2005

ABSTRACT

Motivation: Building an accurate protein classification system depends critically upon choosing a good representation of the input sequences of amino acids. Recent work using string kernels for protein data has achieved state-of-the-art classification performance. However, such representations are based only on labeled data—examples with known 3D structures, organized into structural classes—whereas in practice, unlabeled data are far more plentiful.

Results: In this work, we develop simple and scalable cluster kernel techniques for incorporating unlabeled data into the representation of protein sequences. We show that our methods greatly improve the classification performance of string kernels and outperform standard approaches for using unlabeled data, such as adding close homologs of the positive examples to the training data. We achieve equal or superior performance to previously presented cluster kernel methods and at the same time achieving far greater computational efficiency.

Availability: Source code is available at www.kyb.tuebingen.mpg.de/bs/people/weston/semiprot. The Spider matlab package is available at www.kyb.tuebingen.mpg.de/bs/people/spider

Contact: jasonw@nec-labs.com

Supplementary information: www.kyb.tuebingen.mpg.de/bs/people/weston/semiprot

1 INTRODUCTION

A key problem in computational biology is the classification of proteins into functional and structural classes given their amino acid sequences. The major methods for homology detection can be split into three basic groups: pairwise sequence comparison algorithms (Altschul *et al.*, 1990; Smith and Waterman, 1981), generative models for protein families (Krogh *et al.*, 1994; Park *et al.*, 1998) and discriminative classifiers (Jaakkola *et al.*, 2000; Leslie *et al.*, 2002; Liao and Noble, 2002). Popular sequence comparison methods such as BLAST and Smith–Waterman (SW) are based on unsupervised alignment scores. Generative models such as profile hidden Markov models (HMMs) represent positive examples of a protein family, but they can be trained iteratively using both positively labeled and unlabeled examples by pulling in close homologs and adding them to the positive set. A compromise between these

methods is PSI-BLAST (Altschul *et al.*, 1997), which uses BLAST to iteratively build a probabilistic profile of a query sequence and obtain a more sensitive sequence comparison score. Finally, classifiers such as support vector machines (SVMs) use both positive and negative examples and provide state-of-the-art performance when used with appropriate distance metrics (i.e. appropriate kernels) (Jaakkola *et al.*, 2000; Leslie *et al.*, 2002; Liao and Noble, 2002; Saigo *et al.*, 2004). However, these classifiers still require an auxiliary method (such as PSI-BLAST) to handle unlabeled data; one generally adds predicted homologs of the positive training examples to the training set before training the classifier.

In practice, relatively few labeled data are available—~30 000 proteins with known 3D structure, some belonging to families and superfamilies with only a handful of labeled members—whereas there are close to one million sequenced proteins, providing abundant unlabeled data. New semi-supervised learning techniques should be able to make better use of this unlabeled data.

Recent work in semi-supervised learning has focused on changing the representation given to a classifier by taking into account the structure described by the unlabeled data (Zhu and Ghahramani, 2002; Chapelle *et al.*, 2002; Szummer and Jaakkola, 2001). These works can be viewed as cases of cluster kernels that produce similarity metrics based on the cluster assumption, i.e. two points in the same ‘cluster’ or region of high density should have a small distance between each other. In this work, we investigate the use of cluster kernels for protein classification by developing two simple and scalable methods for modifying a base kernel. The neighborhood kernel uses averaging over a neighborhood of sequences defined by a local sequence similarity measure, and the bagged kernel uses bagged clustering of the full sequence dataset to modify the base kernel. In both the semi-supervised and transductive settings, these techniques greatly improve the classification performance when used with mismatch string kernels, and the techniques achieve equal or superior results to all previously presented cluster kernel methods that we tried. Moreover, the neighborhood and bagged kernel approaches are far more computationally efficient than these competing methods.

The current work is an expanded version of a conference proceedings paper (Weston *et al.*, 2003). We have included new large scale experiments for the cluster kernel methods using the Swiss-Prot database as a source of unlabeled protein sequence data and comparison with the recent profile kernel method (Kuang *et al.*, 2004).

*To whom correspondence should be addressed.

2 REPRESENTATIONS AND KERNELS FOR PROTEIN SEQUENCES

Proteins can be represented as sequences of variable length, typically several hundred characters long, from the alphabet of 20 amino acids. In order to use learning algorithms that require vector inputs, we must first find a suitable feature vector representation, mapping sequence x into a vector space by $x \mapsto \Phi(x)$. Kernel methods, such as SVMs, need to compute only inner products, called kernels, $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$, for training and testing. Thus, we can accomplish the above mapping using a kernel for sequence data.

Biologically motivated sequence comparison scores, such as SW or BLAST, provide an appealing representation of sequence data. The SW algorithm (Smith and Waterman, 1981) uses dynamic programming to compute the optimal local gapped alignment score between two sequences, whereas BLAST (Altschul *et al.*, 1990) approximates SW by computing a heuristic alignment score. Both methods return empirically estimated E -values indicating the confidence of the score. These alignment-based scores do not define a positive definite kernel; however, one can use a feature representation based on the empirical kernel map

$$\Phi(x) = \langle d(x_1, x), \dots, d(x_m, x) \rangle,$$

where $d(x, y)$ is the pairwise score (or E -value) between x and y and $x_i, i = 1, \dots, m$, are the training sequences. Using E -values of SW algorithm in this fashion gives strong classification performance (Liao and Noble, 2002). Note, however, that the method is slow, both because computing each SW score is $O(|x|^2)$ and because computing each empirically mapped kernel value is $O(m)$.

Another appealing idea is to derive the feature representation from a generative model for a protein family. In the Fisher kernel method (Jaakkola *et al.*, 2000), one first builds a profile HMM for the positive training sequences, defining a log-likelihood function $\log P(x|\theta)$ for any protein sequence x . Then the gradient vector $\nabla_{\theta} \log P(x|\theta)|_{\theta=\theta_0}$, where θ_0 is the maximum-likelihood estimate for model parameters, defines an explicit vector of features, called Fisher scores, for x . This representation gives excellent classification results, but the Fisher scores must be computed by an $O(|x|^2)$ forward-backward algorithm, making the kernel tractable but slow.

It is possible to construct useful kernels directly without explicitly depending on generative models by using string kernels. For example, the mismatch kernel (Leslie *et al.*, 2002) is defined by a histogram-like feature map that uses mismatches to capture inexact string matching. The feature space is indexed by all possible k -length subsequences $\alpha = a_1, a_2, \dots, a_k$, where each a_i is a character in the alphabet \mathcal{A} of amino acids. The feature map is defined on k -gram α by $\Phi(\alpha) = (\phi_{\beta}(\alpha))_{\mathcal{A}^k}$, where $\phi_{\beta}(\alpha) = 1$ if α is within m mismatches of β , 0 otherwise, and is extended additively to longer sequences: $\Phi(x) = \sum_{k\text{-grams } \alpha \in x} \Phi(\alpha)$. The mismatch kernel can be computed efficiently using a trie data structure: the complexity of calculating $K(x, y)$ is $O(c_K(|x| + |y|))$, where $c_K = k^{m+1}|\mathcal{A}|^m$. For typical kernel parameters $k = 5$ and $m = 1$ (Leslie *et al.*, 2002), the mismatch kernel is fast, scalable and yields impressive performance.

Other direct string kernel methods include pair HMM and convolution kernels (Watkins, 1999; Haussler, 1999; Lodhi *et al.*, 2000), which are quite general but also have complexity $O(|x||y|)$; more recent and related string alignment kernels (Saigo *et al.*, 2004), also with complexity $O(|x||y|)$; and exact-matching string kernels built with suffix trees and suffix links, with complexity $O(|x| + |y|)$

(Vishwanathan and Smola, 2002). Inexact string matching models similar to the mismatch kernel but with complexity $O(c_K(|x| + |y|))$, with c_K independent of alphabet size, have also been presented (Leslie and Kuang, 2003). The motif kernel (Ben-Hur and Brutlag, 2003) uses features that are built from a fixed database of motifs; computing these features is linear in the length of the sequence. Finally, almost all these kernels can be constructed using the rational kernel framework of Cortes *et al.* (2002). We concentrate on the mismatch kernel representation for the current work.

3 SEMI-SUPERVISED KERNELS FOR PROTEIN SEQUENCES

In semi-supervised learning, one tries to improve a classifier trained on labeled data by exploiting a relatively large set of unlabeled data. An extensive review of techniques can be found in Seeger (2001). It has been shown experimentally that under certain conditions, the decision function can be estimated more accurately in a semi-supervised setting, yielding lower generalization error. The most common assumption one makes in this setting is called the ‘cluster assumption’, i.e. that the class does not change in regions of high density. Similarly, one assumes that the true decision boundary lies in regions of low density. In this section, we review previous semi-supervised kernel approaches that implement the cluster assumption.

3.1 Cluster kernels

We will focus on classifiers that re-represent the given data to reflect structure revealed by unlabeled data. The main idea is to change the distance metric so that the relative distance between two points is smaller if the points are in the same cluster. If one is using kernels, rather than explicit feature vectors, one can modify the kernel representation by constructing a cluster kernel.

Previous work of Chapelle *et al.* (2002) presented a general framework for producing cluster kernels by modifying the eigenspectrum of the kernel matrix. Two of the main methods presented are the random walk kernel and the spectral clustering kernel. The random walk kernel is a normalized and symmetrized version of a transition matrix corresponding to a t -step random walk. As described in Szummer and Jaakkola (2001), one can define a random walk representation by viewing an RBF kernel as a transition matrix of a random walk on a graph with vertices x_i . One then uses the eigendecomposition of the normalized transition matrix to compute the t -step random walk kernel. The spectral clustering kernel is a simple use of the representation derived from spectral clustering (Ng *et al.*, 2001) using the first k eigenvectors of a normalized affinity matrix.

A serious problem with these methods is that one must diagonalize a matrix of size m , where m is the number of labeled and unlabeled data, giving a complexity $O(m^3)$. Other methods of implementing the cluster assumption, such as transductive SVMs (Joachims, 1999), also suffer from computational efficiency issues. A second drawback is that these kernels are better suited to a transductive setting (where one is given both the unlabeled and test points in advance) rather than a semi-supervising setting. In order to estimate the kernel for a sequence not present during training, one is forced to solve a difficult regression problem (Chapelle *et al.*, 2002). In Sections 4 and 5, we will describe two simple methods to implement the cluster assumption that do not suffer from these issues.

3.2 Model-based semi-supervised kernels

Another approach for exploiting unlabeled data in kernel methods is to use semi-supervised training of probabilistic models and then define model-based kernels. For example, in the Fisher kernel approach (Jaakkola *et al.*, 2000), one can use an iterative training algorithm to alternately pull in homologs of the positive training examples from a large unlabeled sequence database and reestimate the profile HMM.

More recently, Kuang *et al.* (2004) introduced a semi-supervised profile-based string kernel for protein sequences. In the profile kernel approach, each sequence is represented by a profile estimated from a large sequence database (e.g. using PSI-BLAST), and each length k segment of the profile is used to define the local mutation neighborhood and a corresponding contribution to the feature vector in a k mer feature space. Unlike the Fisher kernel approach—where a single probabilistic model is used to define feature vectors for sequence examples—in the profile kernel, every example is represented by a probabilistic model in the form of a profile, and the kernel is defined on profile examples. The profile kernel method achieves state-of-the-art performance for the remote homology detection task, strongly outperforming all strictly supervised methods, including the mismatch kernel, the SVM-pairwise method, string alignment kernels (Saigo *et al.*, 2004) and other recent methods (Kuang *et al.*, 2005). We compare the new cluster kernel methods we define here to the profile kernel method in large-scale benchmark experiments in Section 6.3.

4 THE NEIGHBORHOOD MISMATCH KERNEL

In this section and section 5, we introduce two fast and general cluster kernels that leverage unlabeled data to improve a base kernel representation. Unlike other cluster kernel approaches, our kernels make use of two complementary (dis)similarity measures: (1) a base kernel representation that implicitly exploits features useful for discrimination between classes; and (2) a distance measure that describes how close examples are to each other. In our application to protein classification, we use the mismatch string kernel as the base kernel and standard sequence comparison metrics (such as BLAST or PSI-BLAST E -values) as the distance measure. We note that string kernels have proved to be powerful representations for SVM classification (Leslie *et al.*, 2002), but do not give sensitive pairwise similarity scores like the BLAST family methods; thus, the two sequence similarity measures play distinct roles in the kernel definition.

For our first cluster kernel, we use a standard sequence similarity measure like BLAST or PSI-BLAST to define a neighborhood for each input sequence. The neighborhood $\text{Nbd}(x)$ of sequence x is the set of sequences x' with similarity score to x below a fixed E -value threshold, together with x itself. Now given a fixed original feature representation, we represent x by the average of the feature vectors for members of its neighborhood:

$$\Phi_{\text{nbd}}(x) = \frac{1}{|\text{Nbd}(x)|} \sum_{x' \in \text{Nbd}(x)} \Phi_{\text{orig}}(x').$$

The neighborhood kernel is then defined by

$$K_{\text{nbd}}(x, y) = \frac{\sum_{x' \in \text{Nbd}(x), y' \in \text{Nbd}(y)} K_{\text{orig}}(x', y')}{|\text{Nbd}(x)| |\text{Nbd}(y)|}.$$

We will see in the experimental results that this simple neighborhood-averaging technique, used in a semi-supervised setting with the mismatch kernel, dramatically improves classification performance.

In general, computing each neighborhood kernel value is quadratic in neighborhood size, as is clear from the kernel expression given above. However, in the special case where we use the mismatch kernel as base kernel, we can modify the mismatch kernel algorithm by presenting each neighborhood set as a concatenation of the neighborhood sequences (keeping track of where the ends of sequences are located); using a trie data structure, the kernel computation is linear in sequence length, giving a complexity of $O(k^{m+1} |\mathcal{A}|^m (\sum x' + \sum y'))$ (i.e. linear in neighborhood size) to compute $K_{\text{nbd}}(x, y)$, where $|\mathcal{A}|$ is the size of the alphabet of amino acids (Leslie *et al.*, 2002).

To see how the neighborhood approach fits with the cluster assumption, consider a set of points in feature space that form a ‘cluster’ or dense region of the dataset, and consider the region R formed by the union of the convex hulls of the neighborhood point sets. If the dissimilarity measure is a true distance, the neighborhood averaged vector $\Phi_{\text{nbd}}(x)$ stays inside the convex hull of the vectors in its neighborhood, and all the neighborhood vectors stay within region R . In general, the cluster contracts inside R under the averaging operation. Thus, under the new representation, different clusters can become better separated from each other.

5 THE BAGGED MISMATCH KERNEL

A number of existing clustering techniques are much more efficient than the methods mentioned in Section 3. For example, the classical k -means algorithm is $O(rkmd)$, where m is the number of data points, d their dimensionality and r the number of iterations required. Empirically, this running time grows sublinearly with k , m and d . Therefore, in practice, it is computationally efficient to run k -means multiple times, which can be useful because k -means can converge to local minima. We therefore consider the following method:

- (1) Run k -means n times, giving $p = 1, \dots, n$ cluster assignments $c_p(x_i)$ for each i .
- (2) Build a bagged-clustering representation based on the fraction of times that x_i and x_j are in the same cluster

$$K_{\text{bag}}(x_i, x_j) = \frac{\sum_p [c_p(x_i) = c_p(x_j)]}{n}. \quad (1)$$

- (3) Take the product between the original and bagged kernel

$$K(x_i, x_j) = K_{\text{orig}}(x_i, x_j) \cdot K_{\text{bag}}(x_i, x_j).$$

Since k -means gives different solutions on each run, Step (1) will give different results; for other clustering algorithms one could subsample the data instead. Step (2) is a valid kernel because it is the inner product in an nk -dimensional space $\Phi(x_i) = \langle [c_p(x_i) = q] : p = 1, \dots, n, q = 1, \dots, k \rangle$, and products of kernels as in Step (3) are also valid kernels. The intuition behind the approach is that the original kernel is rescaled by the ‘probability’ that two points are in the same cluster, hence encoding the cluster assumption. To estimate the kernel on a test sequence x in a semi-supervised setting, one can assign x to the nearest cluster in each of the bagged runs to compute $K_{\text{bag}}(x, x_i)$. We apply the bagged kernel method with K_{orig} as the mismatch kernel and K_{bag} built by running k -means on the distances induced by PSI-BLAST.

6 EXPERIMENTS

We measure the recognition performance of cluster kernel methods by testing their ability to classify protein domains into superfamilies in the structural classification of proteins (SCOP) (Murzin *et al.*, 1995). For the purposes of this experiment, two domains that come from the same superfamily are assumed to be homologous, and two domains from different folds are assumed to be unrelated. For pairs of proteins in the same fold but different superfamilies, their relationship is uncertain, and so these pairs are not used in evaluating the algorithm. This labeling scheme has been used in several previous studies of remote homology detection algorithms (Jaakkola *et al.*, 2000; Liao and Noble, 2002). We use the same 54 target families and the same test and training set splits as in the remote homology experiments from Liao and Noble (2002). The sequences are 7329 SCOP domains obtained from version 1.59 of the database after purging with astral.stanford.edu so that no pair of sequences share >95% identity. Compared with Liao and Noble (2002), we reduce the number of available labeled training patterns roughly by one-third. Dataset sequences that were neither in the training nor test sets for experiments from Liao and Noble (2002) are included as unlabeled data.

All methods are evaluated using receiver operating characteristic (ROC) analysis (Hanley and McNeil, 1982). An ROC curve plots the rate of true positives as a function of the rate of false positives at varying decision thresholds. The ROC score is the area under this curve. A perfect classifier, which places all positive examples above all negative examples, receives an ROC score of 1, and a random classifier receives a score of ~ 0.5 . In addition to the ROC score, we compute the ROC₅₀ score, which is the ROC score computed only up to the first 50 false positives (Gribskov and Robinson, 1996). This score focuses on the top of the ranking, which in some applications is the most important.

In all experiments, we use an SVM classifier with a small soft margin parameter, set as $K_{ij} \leftarrow K_{ij} + \gamma$ where γ is 0.02 ρ times the median diagonal kernel entry, and ρ is the fraction of training set sequences that have the same label as the i -th sequence. The SVM computations are performed using the freely available Spider Matlab machine learning package.

6.1 Semi-supervised setting

Our first experiment shows that the neighborhood mismatch kernel makes better use of unlabeled data than the baseline method of ‘pulling in homologs’ prior to training the SVM classifier, i.e. simply finding close homologs of the positive training examples in the unlabeled set and adding them to the positive training set for the SVM. Homologs come from the unlabeled set (not the test set), and ‘neighbors’ for the neighborhood kernel come from the training plus unlabeled data. We compare the methods using the mismatch kernel representation with $k = 5$ and $m = 1$, as used in Leslie *et al.* (2002). Homologs are chosen via BLAST or PSI-BLAST as having a pairwise E -value < 0.05 [the default parameter setting (Altschul *et al.*, 1990)] with any of the positive training samples. The neighborhood mismatch kernel uses the same threshold to choose neighborhoods. For the neighborhood kernel, we normalize before and after the averaging operation via $K_{ij} \leftarrow K_{ij} / \sqrt{K_{ii}K_{jj}}$. The results are given in Figure 1 and Table 1.

Figure 1 plots the number of families achieving a given ROC₅₀ score. A signed rank test shows that the neighborhood mismatch

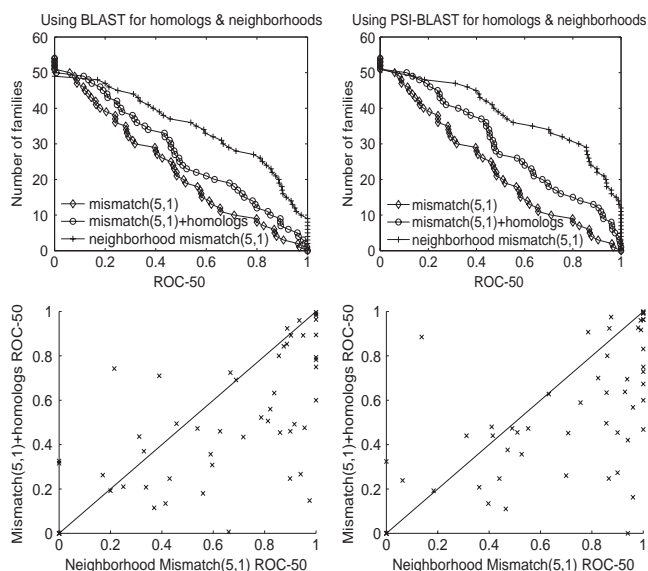


Fig. 1. Comparison of protein representations and classifiers using unlabeled data. The mismatch kernel is used to represent proteins, with close homologs being pulled in from the unlabeled set with BLAST (left) or PSI-BLAST (right). Building a neighborhood with the neighborhood mismatch kernel in both cases improves over the baseline of pulling in homologs. Note: We also pull in homologs during the SVM training for the neighborhood kernel.

Table 1. Mean ROC₅₀ and ROC scores over 54 target families for semi-supervised experiments, using BLAST and PSI-BLAST for adding homologs and defining the neighborhood kernel

	BLAST ROC ₅₀	ROC	PSI-BLAST ROC ₅₀	ROC
Mismatch kernel	0.416	0.870	0.416	0.870
Mismatch kernel + homologs	0.480	0.900	0.550	0.910
Neighborhood mismatch kernel	0.639	0.922	0.699	0.923

kernel yields significant improvement over adding homologs (P -value = 3.9×10^{-5}). Note that the PSI-BLAST scores in these experiments are built using the whole database of 7329 sequences (i.e. test sequences in a given experiment are also available to the PSI-BLAST algorithm); so these results are slightly optimistic. However, the comparison of methods in a truly inductive setting using BLAST shows the same improvement of the neighborhood mismatch kernel over adding homologs (P -value = 8.4×10^{-5}).

The improvement from the neighborhood kernel does not come from the BLAST and PSI-BLAST representations alone. The mean ROC₅₀ score for these representations using an empirical map (see the transductive setting for a description) are 0.368 and 0.533 without pulling in homologs, and 0.448 and 0.595 with pulled-in homologs. Moreover, simply adding the BLAST and mismatch kernels together (using an empirical map) without using homologs yields a mean ROC₅₀ of 0.3943. Therefore the improvement is not because the methods give independent information about the targets which can be easily combined.

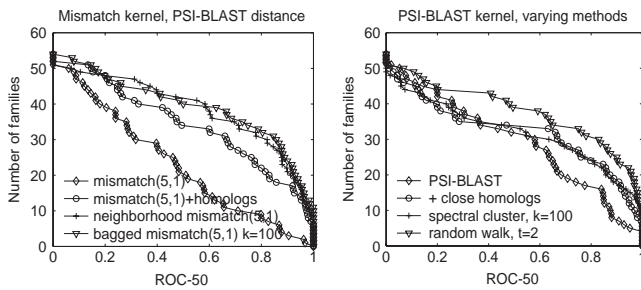


Fig. 2. Comparison of protein representations and classifiers using unlabeled data in a transductive setting. Neighborhood and bagged mismatch kernels outperform pulling in close homologs (left panel) and equal or outperform previous semi-supervised methods (right panel). Note: We also pull in homologs during the SVM training for the neighborhood and bagged kernels.

6.2 Transductive setting

In the following experiments, we consider a transductive setting, in which the test points are given to the methods in advance as unlabeled data, giving slightly improved results over the last section. Although this setting is unrealistic for a real protein classification system, it enables comparison with random walk and spectral clustering kernels, which do not easily work in the semi-supervised setting. In Figure 2 (left panel), we again show the mismatch kernel compared with pulling in homologs and the neighborhood kernel. This time we also compared with the bagged mismatch kernel using bagged k -means with $k = 100$ and $n = 100$ runs, which gave the best results. We found the method quite insensitive to k . The result for $k = 400$ is also given in Table 2. We then compare these methods with random walk and spectral clustering kernels. Neither of the two methods works well for the mismatch kernel (Supplementary data), perhaps because the feature vectors are so orthogonal. However, for a PSI-BLAST representation via empirical kernel map, the random walk outperforms pulling in homologs. We take the empirical map with $\Phi(x) = (\exp(-\lambda d(x_1, x)), \dots, \exp(-\lambda d(x_m, x)))$, where $d(x, y)$ are PSI-BLAST E -values and $\lambda = 1/1000$, which improves over a linear map. We report the results for the best parameter choices, $t = 2$ for the random walk and $k = 200$ for spectral clustering. We found the latter quite brittle with respect to the parameter choice; the results for other parameters can be found on the Supplementary website. For pulling in close homologs, we take the empirical kernel map only for points in the training set and the chosen close homologs. Finally, we also run transductive SVMs. The results are given in Table 2 and right panel of Figure 2. A signed rank test (with adjusted P -value cutoff of 0.05) finds no significant difference between the neighborhood kernel, the bagged kernel ($k = 100$) and the random walk kernel in this transductive setting. Thus, the new techniques are comparable with random walk, but are feasible to calculate on full scale problems.

6.3 Large-scale experiments

Semi-supervised and transductive methods are most interesting and potentially give the greatest benefit in the realistic setting where a large amount of unlabeled data is used. We therefore test our cluster kernel methods in large-scale experiments, using 101 602 Swiss-Prot protein sequences as additional unlabeled data. For simplicity, we first give results for both the neighborhood and bagged kernels

Table 2. Mean ROC_{50} and ROC scores over 54 target families for transductive experiments

	ROC_{50}	ROC
Mismatch kernel	0.416	0.875
Mismatch kernel + homologs	0.625	0.924
Neighborhood mismatch kernel	0.704	0.917
Bagged mismatch kernel ($k = 100$)	0.719	0.943
Bagged mismatch kernel ($k = 400$)	0.671	0.935
PSI-BLAST kernel	0.533	0.866
PSI-BLAST+homologs kernel	0.585	0.873
Spectral clustering kernel	0.581	0.861
Random walk kernel	0.691	0.915
Transductive SVM	0.637	0.874

in the transductive setting, i.e. in the case where test sequences are available as additional unlabeled examples in all the experiments. Then, for a clean comparison against the profile kernel, we test the neighborhood kernel and the profile kernel in a semi-supervised setting, where the Swiss-Prot database alone is used as the source of unlabeled data.

For the large-scale neighborhood mismatch kernel experiments, we first compute the entire SCOP plus Swiss-Prot kernel (108931×108931) matrix with mismatch kernel parameters $k = 5$ and $m = 1$. We then apply the neighborhood averaging operation to produce the 7329×7329 kernel matrix for SCOP sequences needed for SVM training. We normalize the kernel matrix before and after the neighborhood averaging operation. Results in Table 3 clearly show that the inclusion of a large amount of additional unlabeled data from Swiss-Prot significantly improves classification performance. Moreover, the neighborhood kernel again outperforms the baseline method of adding homologs of the positive training sequences to the training set.

For the large-scale bagged mismatch kernel experiments, the fact that many of the sequences in the Swiss-Prot database are multi-domain protein sequences complicates the clustering step. Since the PSI-BLAST E -values used as the dissimilarity metric are based on local alignment, a multi-domain sequence can be similar to many unrelated single-domain sequences, and hence the clustering algorithm may fail to converge. As an approximate remedy, we only use Swiss-Prot protein sequences with maximal length of 250 for the large-scale k -means clustering, reasoning that most multi-domain sequences would be eliminated by this length constraint. We randomly sample 30 000 protein sequences from the set of Swiss-Prot with length 250 or less to use as unlabeled data for clustering. Since the method mainly depends on the quality of the clusters containing the labeled points, we terminate the k -means clustering algorithm once there are no more changes in the label assignment for the SCOP sequences. It is worth noting that a small number of two-domain sequences may have length below our cutoff, but we observe that the k -means clustering algorithm still behaves relatively stably. We use the same mismatch kernel parameters for the bagged kernel as the ones we use for the small-scale bagged kernel experiments. A comparison of results is shown in Table 3. Again, bagged kernel performance significantly improves when a large amount of unlabeled data is provided to the clustering algorithm.

Table 3. Mean ROC₅₀ and ROC scores over 54 target families for large-scale transductive experiments

	Without Swiss-Prot		With Swiss-Prot	
	ROC ₅₀	ROC	ROC ₅₀	ROC
Mismatch kernel + homologs	0.625	0.924	0.706	0.945
Neighborhood mismatch kernel	0.704	0.917	0.871	0.971
Bagged mismatch kernel ($k = 100$)	0.719	0.943	0.803	0.953
Bagged mismatch kernel ($k = 400$)	0.671	0.935	0.775	0.955

We include homologs from the unlabeled set and the test set (SCOP + Swiss-Prot) for the training of all our SVMs.

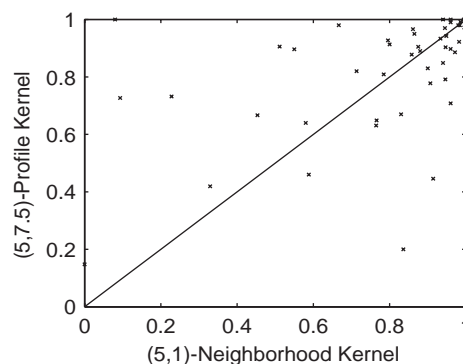
Table 4. Mean ROC₅₀ and ROC scores over 54 target families for large-scale semi-supervised experiments

	ROC ₅₀	ROC
Neighborhood mismatch kernel	0.810	0.955
Profile kernel	0.842	0.980

We do not include homologs from the unlabeled set (Swiss-Prot) for the training of our SVMs in these experiments.

Finally, we compare the performance of the stronger of the cluster kernels, the neighborhood kernel, to a state-of-the-art semi-supervised kernel method, the profile kernel. The profile kernel representation depends on estimating sequence profiles for each input sequence using a large sequence database, and therefore, we only present results in the large-scale setting. In these experiments, we use a semi-supervised training set-up: the Swiss-Prot database alone is used as the source of unlabeled data for estimating PSI-BLAST profiles and defining sequence neighborhoods; SCOP sequences are not used for profile learning or for neighborhood averaging. For the cleanest comparison, we do not add SCOP homologs to the positive training set before training the SVMs. Mean ROC₅₀ and ROC results are given in Table 4, and a comparison of ROC₅₀ results over all experiments is given in Figure 3. Although the cluster kernel method does not outperform the profile kernel on average, results of the two methods are similar (20 wins, 25 losses, 9 ties for the cluster kernel); a signed rank test with a P -value threshold of 0.05 finds no significant difference in performance between the two methods.

The computational cost of the profile kernel depends on the parameter σ controlling the size of the local mutation neighborhoods; the mutation neighborhood of each length k window of the profile consists of k -mers whose negative log-likelihood given the profile is $< \sigma$. If M_σ is the maximum size of any local mutation neighborhood in the input sequences, the complexity of computing the profile kernel for profiles of sequences x and y can be bounded by $O(kM_\sigma(|x| + |y|))$. In practice, in the large-scale experiments reported, the profile kernel takes less time to compute than the neighborhood kernel, since we do not limit the number of sequences represented in the neighborhoods and hence the neighborhood representation is much less compact than the profile kernel representation. Various schemes could make the neighborhood kernel running time comparable or faster than the profile kernel, e.g. sampling from the neighborhood rather than using

**Fig. 3.** Comparison of neighborhood kernel and profile kernel ROC₅₀ performance for large-scale semi-supervised experiments. No homologs were added to the training set for the purpose of training the SVMs.

all sequences; one would have to investigate how much sampling is needed to retain classification performance.

Although the neighborhood and profile kernels have a similar number of wins in this set of experiments, performance on individual experiments can be quite different. We speculate that the profile kernel might perform best when the training profiles themselves are not too distant from the test sequences (as measured, e.g. by PSI-BLAST E -value), whereas the neighborhood kernel might perform better in the more difficult situation where the test sequences are poorly described by every positive training profile. (Note that, in both situations, the discriminative profile kernel SVM outperforms the PSI-BLAST algorithm used directly as a ranking method.) To test this hypothesis, we consider the top 10 wins of the profile kernel method and of the neighborhood kernel method, as ranked by the difference in ROC₅₀ performance, and for each positive test sequence in these experiments, we find the positive training profile that detected the test sequence with minimum PSI-BLAST E -value. We then rank the test sequences by this minimum E -value and record the median E -value in this list. We find that in the top 10 profile kernel wins, 8 of the experiments has median minimum E -value < 0.05 , and 9 are < 0.1 , whereas in the neighborhood kernel wins, only 2 experiments have median minimum E -value < 0.05 , and only 3 are < 0.1 . We conclude that the neighborhood kernel may have an advantage over the profile kernel in situations where the training profiles are very distant from the remote homolog sequences to be detected.

7 DISCUSSION

Two of the most important issues in protein classification are representation of sequences and handling unlabeled data. Two developments in recent kernel methods research, string kernels and cluster kernels, address these issues separately. We have described two kernels, the neighborhood mismatch kernel and the bagged mismatch kernel, which combine both approaches and yield state-of-the-art performance in protein classification. These approaches, used with an efficient string kernel, are fast and scalable cluster kernels for sequence data and do not require diagonalization of the kernel matrix as in other cluster kernel methods. A potential direction for improvement in the neighborhood kernel would be to extract only those segments of ‘neighboring’ sequences that correspond to the

local alignment-based *E*-value score; when we use the entire multi-domain Swiss-Prot sequences as neighbors of a single-domain SCOP sequence, these neighbor sequences may include long regions that are unrelated to the SCOP domain, and hence we introduce noise in the neighborhood averaging operation.

Although we have motivated our kernels by earlier work on cluster kernels and the cluster assumption, one can also view the neighborhood and bagged kernels as using unlabeled data locally (from nearby sequences or the local cluster) for smoothing the kernel representation. Related work using probabilistic models instead of unlabeled data for smoothing includes the recently introduced Bhattacharyya kernel (Jebara *et al.*, 2004), which assigns a probability distribution to each example and defines a kernel on these distributions.

We also compared with the profile-based string kernels of Kuang *et al.* (2004), which are also based on a semi-supervised learning paradigm. These string kernels are also scalable and achieve very high classification accuracy; in our experiments, the neighborhood kernel performs similar to the profile kernel. However, the profile kernel method requires producing a profile for each query sequence, which is necessarily tied to alignment. In contrast, the cluster kernels that we present here are more general, in that any dissimilarity measure can be used for neighborhood averaging or bagging and any base kernel chosen for the initial representation. These kernels may, therefore, be applicable to a wider range of problems. For example, one could use the expression coherence in a set of microarray experiments as a measure of functional similarity of genes combined with a base kernel to define cluster kernels for functional gene classification. One could also hope to further improve the performance for the protein classification task by using a more powerful base kernel than the mismatch kernel [e.g. the string alignment kernel of Saigo *et al.* (2004)], although the computational expense of the improved base kernel representation may become a concern.

ACKNOWLEDGEMENTS

We would like to thank Eleazar Eskin for discussions that contributed to the neighborhood kernel, Rui Kuang for providing the profile kernel experimental results, and Olivier Chapelle and Navin Lal for the helpful discussions. W.S.N. is an Alfred P. Sloan Foundation Research Fellow. This work is supported by an Award in Informatics from the PhRMA Foundation and NSF grant EIA-0312706.

Conflict of Interest: none declared.

REFERENCES

- Altschul,S.F. *et al.* (1990) A basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
- Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
- Ben-Hur,A. and Brutlag,D. (2003) Remote homology detection: a motif based approach. *Bioinformatics*, **19**, i26–i33.
- Chapelle,O. *et al.* (2002) Cluster kernels for semi-supervised learning. *Adv. Neural Inf. Process. Syst.*, **15**, 601–608.
- Cortes,C. *et al.* (2002) Rational kernels. *Adv. Neural Inf. Process. Syst.*, **15**, 617–624.
- Gribskov,M. and Robinson,N.L. (1996) Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Comput. Chem.*, **20**, 25–33.
- Hanley,J.A. and McNeil,B.J. (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**, 29–36.
- Hausler,D. (1999) Convolution kernels on discrete structures. *Technical report UCSC-CRL-99-10*, University of California, Santa Cruz.
- Jaakkola,T. *et al.* (2000) A discriminative framework for detecting remote protein homologies. *J. Comput. Biol.*, **7**, 95–114.
- Jebara,T. *et al.* (2004) Probability product kernels. *J. Mach. Learn.*, **5**, 819–844.
- Joachims,T. (1999) Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia, pp. 200–209.
- Krogh,A. *et al.* (1994) Hidden Markov models in computational biology: applications to protein modeling. *J. Mol. Biol.*, **235**, 1501–1531.
- Kuang,R., Ie,E., Wang,K., Wang,K., Siddiqi,M., Freund,Y. and Leslie,C. (2004) Profile-based string kernels for remote homology detection and motif extraction. In *3rd International IEEE Computer Society Computational Systems Bioinformatics Conference*, Stanford, CA. IEEE Computer Society, pp. 152–160.
- Kuang,R. *et al.* (2005) Profile kernels for detecting remote protein homologs and discriminative motifs. *J. Bioinform. Comput. Biol.*, **3**, 1–23.
- Leslie,C. and Kuang,R. (2003) Fast kernels for inexact string matching. In *Proceedings of the Sixteenth Annual Conference on Learning Theory and Seventh Kernel Workshop*, Washington, DC, pp. 114–128.
- Leslie,C. *et al.* (2002) Mismatch string kernels for SVM protein classification. *Adv. Neural Inf. Process. Syst.*, **15**, 1441–1448.
- Liao,C. and Noble,W.S. (2002) Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, Washington, DC, pp. 225–232.
- Lodhi,H. *et al.* (2000) Text classification using string kernels. *Adv. Neural Inf. Process. Syst.*, **13**, 563–569.
- Murzin,A.G. *et al.* (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.
- Ng,A. *et al.* (2001) On spectral clustering: analysis and an algorithm. *Adv. Neural Process. Inform. Syst.*, **14**, 849–856.
- Park,J. *et al.* (1998) Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods. *J. Mol. Biol.*, **284**, 1201–1210.
- Saigo,H. *et al.* (2004) Protein homology detection using string alignment kernels. *Bioinformatics*, **20**, 1682–1689.
- Seeger,M. (2001) Learning with labeled and unlabeled data. *Technical report*, University of Edinburgh, UK.
- Smith,T. and Waterman,M. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.
- Szummer,M. and Jaakkola,T. (2001) Partially labeled classification with Markov random walks. *Adv. Neural Inf. Process. Syst.*, **14**, 945–952.
- Vishwanathan,S.V.N. and Smola,A. (2002) Fast kernels for string and tree matching. *Adv. Neural Inf. Process. Syst.*, **15**, 585–592.
- Watkins,C. (1999) Dynamic alignment kernels. *Advances in Large Margin Classifiers*, 39–50. *Technical report*, Royal Holloway, University of London, UK.
- Weston,J. *et al.* (2003) Cluster kernels for semi-supervised protein classification. *Adv. Neural Inf. Process. Syst.*, **16**, 595–602.
- Zhu,X. and Ghahramani,Z. (2002) Learning from labeled and unlabeled data with label propagation. *Technical report CMUCALD-02-107*, Carnegie Mellon University, Pittsburgh, PA.