Rutgers Computer Science Technical Report RU-DCS-TR630
May 2008

Kernel Methods and Algorithms for General Sequence Analysis

by

Pavel P. Kuksa, Pai-Hsi Huang, Vladimir Pavlovic
Rutgers University
Piscataway, NJ 08854
pkuksa@cs.rutgers.edu

ABSTRACT

Problems of analysis and modeling of sequential data arise in many practical applications. In this work, we develop efficient algorithms and methods for general sequence analysis. In particular, we propose novel ways of modeling sequences under complex transformations (such as multiple insertions, deletions, mutations) and present a new family of similarity measures (kernels), spatial string kernels, that can be computed very efficiently and show state-of-the-art performance on a variety of distinct classification tasks. We also present new algorithms for approximate (e.g. with mismatches) string comparison that improve currently known time bounds for such tasks and show order-of-magnitude running time improvements. In an extensive set of experiments on many challenging classification problems, such as detecting homology (evolutionary similarity) of remotely related proteins, categorizing texts, and performing classification of music samples, proposed algorithms and measures display state-of-the-art classification performance and run substantially faster than existing methods. We solve these problems in both binary and multi-class settings, as well as apply our methods to large-scale datasets with partially labeled samples.

1 Introduction

Analysis of large scale sequential data has become an important task in data mining, inspired in part by numerous scientific and technological applications such as the biological sequence analysis or document and text mining. Advanced machine learning methods proposed in recent years ensure more accurate analytical results on these sets. One particular method to analyze sequential data is to consider the data as a *string* of symbols. In such framework, the goal may be to associate a class, or a label, with a string that has not been observed before. For example, protein sequences are strings of amino acids and the associated classification task is *functional annotation*. On the other hand, a text document can be considered *a string of words* with the associated classification task of assigning a *topic*, such as *sports* or *politics*, to a given text document.

Classification of data in sequential domains is made challenging by the variability in the sequence lengths, potential existence of important features on multiple scales, as well as the size of typical datasets. Over the last decade, discriminative learning methods that focus on capturing the differences among classes of sequences have shown significant promise [1, 7, 10, 16, 12]. A typical *discriminative* setting requires that each sequence be represented using a fixed-length descriptor. As a result, representations such as the string kernels for protein sequences and the bag-of-words for text documents have gained wide popularity. Such representations have led to increasingly accurate sequence classification algorithms. On the other hand, the present approaches have been plagued by either a simple representation's inability to model complex but revealing interactions of symbols in a sequence or a complex representation's computational intractability.

In this study, we propose a new family of string-based kernel classification methods, the sparse spatial sample kernels SSSK. The proposed kernels sample the sequences at different scales and capture local substring dependencies within a string. The feature spaces induced by SSSK kernels are low-dimensional and incur low computational cost. Such characteristics open the possibility of analyzing very large datasets under both fully-supervised and semi-supervised setting with modest computational resources.

Our main contributions in this study include the following:

- We propose a new class of string-based kernels, that can be efficiently computed and applied in different domains, such as the protein sequences, audio signal sequences, and text documents. We show improvement in classification accuracies and demonstrate state-of-the-art performance compared to previously published methods.

- We demonstrate the ability of the proposed kernels in the semi-supervised setting with large datasets. Due to the low computational cost incurred from our choice of features, the proposed feature set shows scalability and significant improvement in one particular domain (protein sequence analysis), which suggests similar improvement in other domains can potentially be acquired by leveraging the unlabeled datasets.

- We also show that the proposed feature set scales well with the size of the alphabet, across small (protein), medium (music) and large (document) alphabet sets.

2 Background

In a sequence classification problem, the data set usually comes in the form of N examples, each containing a series of observations made at various time or space intervals. In all examples, the length of the observation may vary. In this work we focus on the data with observations arising from a discrete alphabet Σ, such as the 20 amino acids in protein sequences, words in English dictionary or discretized representations of sound signal spectral energies.

The discrete-valued sequence representation gives rise to the problem of string classification. Over the past decade, various methods have been proposed as a solution to the string classification problem. Earlier methods involve use of *generative models*, such as *hidden Markov models* [14] for sequence classification. More recent methods focus on estimating the decision boundaries directly in the sequence space using *discriminative models*. Among the discriminative approaches, in many sequence analysis tasks, kernel-based [17] machine learning methods provide the most accurate results [7, 10, 16, 12].

In our study, we will focus on the set of kernels that directly operate on the observed string. These kernels were originally motivated by analysis of biological sequences but also have applications in other research areas. We will also discuss a simple semi-supervised learning technique that can be easily implemented under the kernel-based learning framework; we aim to show that the features, together with the simple algorithm that we propose for evaluating the kernel matrix, scales well both in the size of the alphabet set and in the size of the datasets at hand.

In the following subsections, we will briefly review some standard string-based kernels in supervised and semi-supervised settings.

2.1 Supervised Methods

Given a sequence, X, the *spectrum-k* kernel [9] first *implicitly* maps X to a d-dimensional vector, where $d = |\Sigma|^k$:

$$\Phi^{spectrum-k}(X) \quad = \quad \left(\sum_{\alpha \in X} I(\alpha = \gamma) \right)_{\gamma \in \Sigma^k}, \tag{1}$$

where α denotes a *k-mer* in X, γ is a member in the set of all *k-mers* induced by Σ, and $I(.)$ is the indicator function. Next, the kernel between X and Y is defined as

$$K(X, Y) \quad = \quad \Phi(X)^T \Phi(Y), \tag{2}$$

measuring the agreement between the k-mer spectra of the two sequences. The $mismatch(k, m)$ kernel [10] relaxes the exact string matching of the spectrum kernel by allowing up to m mismatches, or substitutions, between α and γ. Defining $N(\gamma, m)$ as the set of all possible k-mers that contains up to m mismatches with γ, the representation of a sequence X induced by the mismatch(k,m) kernel is

$$\Phi^{mismatch(k,m)}(X) \quad = \quad \left(\sum_{\alpha \in X} I(\alpha \in N(\gamma, m)) \right)_{\gamma \in \Sigma^k}. \tag{3}$$

Explicit inclusion of the substitution process allows the mismatch kernel to significantly outperform the spectrum kernel [10]. Such substitution are commonly observed in protein sequences as mutation or in text documents as use of synonyms. However, the main drawback of the mismatch kernel is the exponential size of the induced feature space and the presence of mismatches, both of which, when combined, incur high computational cost. These computational considerations in both kernels restrict the affinity measure to relatively short and dense k-mers, leading to a loss of the longer-term dependency information.

2.2 Semi-supervised Methods

Under the mismatch kernel all substitutions, such as mutation in residues for protein sequences and change in words for text documents, are treated as equally likely. However, in many practical applications, such assumption may not always be true. For example, due to the physical and chemical affinities, amino acids that belong to the same group are more strongly affiliated with one another than with those of another group. Likewise, in text documents, substitution by synonyms is more likely to occur than substitution by an irrelevant word. The profile kernel [6] takes such constraints into consideration: given a sequence X and a profile P_X, Kuang et al. [6, 5] define the profile features of X as

$$\Phi^{profile(k,\sigma)} = \left(\sum_{i=1\cdots(T_{P_X}-k+1)} I(P_X(i,\gamma) < \sigma) \right)_{\gamma \in \Sigma^k}, \tag{4}$$

where σ is a pre-defined threshold, T_{P_X} denotes the length of the profile and $P_X(i,\gamma)$ the cost of *locally* aligning the k-mer γ to the k-length segment starting at i^{th} position of P_X. A profile is a $|\Sigma|$ by T_X table, where each row represents a symbol in Σ and each column a position in the sequence. The entry $P_X(i,j)$ denotes the probability of observing symbol σ_i at position j. Typically, estimation of such profiles requires the use of an *unlabeled dataset* to avoid overfitting.

Construction of such a profile for each sequence may not be practical in some application domains. For biosequences, $|\Sigma|$ is 20. However, for text classification, $|\Sigma|$ can potentially be very large, on the order of tens of thousands of symbols. In this case, a very simple semi-supervised learning method, the *neighborhood kernel*, can be employed. Neighborhood kernels take advantage of the unlabeled data using the process of neighborhood-induced kernel regularization. Let $\Phi^{orig}(X)$ be the original representation of the sequence X and let $N(X)$ be the *neighborhood* of X[1]. Weston et al. proposed in [18] to re-represent X using:

$$\Phi^{new}(X) = \frac{1}{|N(X)|} \sum_{X' \in N(X)} \Phi^{orig}(X'), \tag{5}$$

Under the new representation, the kernel value between the two sequences X and Y becomes:

$$K^{nbhd}(X,Y) = \frac{\sum_{X' \in N(X), Y' \in N(Y)} K(X',Y')}{|N(X)||N(Y)|}. \tag{6}$$

[1] We will discuss how to define $N(X)$ in later sections.

This representation is a particular form of the regularization of the labeled set by unlabeled data. Note that in this setting, all *training* and *testing* sequences will assume a new representation, whereas in a traditional semi-supervised setting, unlabeled data are used during the *training phase only*. Using the mismatch representation for the sequences [18] showed that the discriminative power of the classifiers improves significantly once information regarding the neighborhood of each sequence becomes available.

3 The Sparse Spatial Sample Kernels

In this section, we present a new class of string kernels that avoids the costly explicit construction of sequence profiles to infer sequence similarity, and does not suffer from the exponential size of the feature space and, associated with it, a high computational cost. The proposed family of kernels are parameterized by three positive integers:

$$K^{(t,k,d)}(X,Y) = \sum_{a_i \in \Sigma^k, 0 \le d_i < d} C(a_1, d_1, \cdots, a_{t-1}, d_{t-1}, a_t|X) \cdot C(a_1, d_1, \cdots, a_{t-1}, d_{t-1}, a_t|Y) \quad , \tag{7}$$

where $C(a_1, d_1, \cdots, a_{t-1}, d_{t-1}, a_t|X)$ denotes the number of times we observe substring $a_1 \overset{d_1}{\leftrightarrow} a_2 \overset{d_2}{\leftrightarrow}, \cdots, \overset{d_{t-1}}{\longleftrightarrow} a_t$ (a_1 separated by d_1 characters from a_2, a_2 separated by d_2 characters from a_3, etc.) in Figure 1.

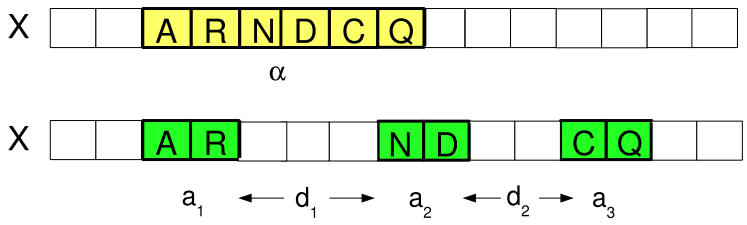


Figure 1: Contiguous k-mer feature α of a traditional spectrum/mismatch kernel (top) contrasted with the sparse spatial samples of the proposed kernel (bottom).

The new kernel implements the idea of sampling the sequences at different resolutions and comparing the resulting spectra; similar sequences will have similar spectrum at one or more resolutions. This takes into account possible mutations, as well as insertions/deletions. Each sample consists of t spatially-constrained probes of size k, each of which lies no more than d positions away from its neighboring probes. In the proposed kernels, the parameter k controls the individual probe size, d controls the locality of the sample, and t controls the cardinality of the sampling neighborhood. In this work, we use short samples of size 1 (i.e., $k = 1$), and set t to 2 (i.e. features are pairs of monomers) or 3 (i.e. features are triples). The proposed sample string kernels, unlike

the family of spectrum kernels [9, 10], not only take into account the feature counts, but also include spatial configuration information, *i.e.* how the features are positioned in the sequence with respect to each other. The spatial information can be critical in establishing similarity of sequences under complex transformations such as the evolutionary processes in biological sequences. The addition of the spatial information experimentally demonstrates very good performance, even with very short sequence features (*i.e.* *k=1*), as we will show in Section 4; the use of short features also leads to significantly lower computational complexity of the kernel evaluations as shown in Sections 3.2, and 4.3 .

3.1 Spatial Sample Kernel Algorithms

The proposed kernels can be efficiently computed using sorting and counting. The dimensionality of the features induced by the proposed kernel is $|\Sigma|^t d^{t-1}$ for our choice of $k = 1$. As a result, for triple-(1,3) and double-(1,5) feature sets, the dimensionalities are $72,000$ and $2,000$, respectively, compared to much higher dimensionality of $3,200,000$ for typical string kernels (e.g., the spectrum-(k) [9], mismatch(k,m) kernels [10], and profile(k,σ) kernels with the common choice of $k = 5$). To compute the kernel values, we first extract the features (t-tuples) from the sequences and sort the extracted features in linear time using counting sort. We then obtain counts of distinct features and for each distinct feature update the kernel matrix for strings containing this feature. For N sequences of the longest length n and u distinct features, computing the NxN kernel matrix takes time linear $O(dnN + min(u, dn)N^2)$ in the length of the sequences. Spatial sample kernel computations are summarized in Algorithm 1.

Algorithm 1. Spatial Sample Kernel
Input: set of strings S, parameters t, d
repeat for every $d_1, ..., d_t \in \{1, ..., d\}$
1: Extract t-tuples of features from each string $s \in S$
2: Sort t-tuples using t rounds of counting sort
3: Scan the sorted list and for each distinct feature f
update the kernel matrix K for all strings containing f
Output: kernel matrix K

3.1.1 Spatial Sample Neighborhood Kernels

The proposed kernel can be extended to accommodate the semi-supervised learning setting for sequence classification, for example, as in [18]. In a semi-supervised learning, the training set is enlarged and includes unlabeled data. One approach to use the unlabeled data is to partition labeled and unlabeled data into groups of similar sequences (neighborhood sets), and use these sets in place of the single sequences. This enriches representation of each sequence and allows to generalize string representations, e.g. compute sequence profile [5], or single averaged or consensus representation for each string as in Equation 6.

However, direct use of the Equation 6 for computation of the refined kernel values between sequences X and Y requires $|N(X)| \times |N(Y)|$ kernel evaluations, i.e. a time quadratic in the

size of the neighborhood. On the other hand, use of Equation 5 requires explicit representation of the sequences which can be prohibitive when the dimensionality of the feature space is high. As a result, performing such *smoothing* operation over standard string kernels (e.g. *mismatch kernel representation*) is computationally intensive, as noted in [18, 6].

Equation 5 lends a useful insight into the complexity of the smoothing operation. For any explicit representation $\Phi(X)$, its smoothed version can be computed in time linear in the size of the neighborhood $|N(X)|$, therefore the smoothed kernel can also be evaluated in time linear in the neighborhood size. However, the smoothed representation in case of the standard string spectrum/mismatch kernels cannot be computed explicitly due to its exponential length.

Alternatively, it is also possible to perform implicit computations over long representations on a one-feature-at-a-time basis, for example by individually evaluating each distinct feature present in the data over the neighborhood sets. In that case, the complexity of the smoothing operation will still be linear in the sequence neighborhood size.

In our experiments, we use implicit computations over induced representations. For each neighborhood set $N(X)$, we first sort the features (e.g. pairs of characters) and then obtain counts for distinct features to evaluate the kernel (summarized in Algorithm 2). This leads to a low space and time complexity for the kernel computations. The same approach is not feasible in case of the mismatch kernels due to exponentially many features and the presence of mismatches. Both implicit and explicit computations depend on the representation length (size of the feature space), making it more advantageous to use low-dimensional representations.

We use Algorithm 1 to compute the kernels in supervised experiments on text, music, and protein sequence data, while Algorithm 2 is used to compute kernels in large-scale semi-supervised experiments. We show in the experiments that our kernels can be efficiently evaluated for very large sequence datasets as well as very large string alphabets.

Algorithm 2. Spatial Sample Neighborhood Kernel

Input: set of string sets $N(S) = \{N(s_1), ...N(s_N))$, where $N(s_i)$ is a set of neighbors for s_i, parameters t, d
repeat for every $d_1, ..., d_t \in \{1, ..., d\}$
1: Extract t-tuples of features from each input set $N(s) \in N(S)$
2: Sort t-tuples using t rounds of counting sort
3: Scan the sorted list and for each distinct feature f update the kernel matrix K for all string groups containing f
Output: kernel matrix K

3.2 Complexity Analysis

We first compare computational complexity of the methods in Table 1. Both mismatch and profile kernels have higher complexity compared to the sample kernels due to the exponential neighborhood size and high dimensionality of the feature space. The cardinalities of the mismatch and profile neighborhoods are $O(k^m|\Sigma|^m)$ and $O(M_\sigma)$, with $k^m|\Sigma|^m \leq M_\sigma \leq |\Sigma|^k$, where $k \geq 5$, and

$|\Sigma| = 20$, compared to a much smaller feature space size of $d^{t-1}|\Sigma|^t$ for the sample kernels, where t is 2 or 3, and d is 3 or 5, respectively. This complexity difference leads to order-of-magnitude improvements in the running times (Section 4.3) of the sample kernels over the standard string kernels (mismatch and profile kernels). The difference is even more pronounced under a semi-supervised setting when kernel smoothing is used. The neighborhood mismatch kernel becomes substantially more expensive to compute for large datasets as indicated in [6, 18] by the authors.

Table 1: Complexity of computations.

Method	Time complexity	Running time (s)
Triple kernel	$O(d^2nN + d^2\|\Sigma\|^3N^2)$	112
Double kernel	$O(dnN + d\|\Sigma\|^2N^2)$	54
Mismatch	$O(k^{m+1}\|\Sigma\|^m nN + \|\Sigma\|^k N^2)$	948
Gapped kernel	$O(\binom{g}{k}gnN + \|\Sigma\|^k N^2)$	-
Profile kernel	$O(kM_\sigma nN + \|\Sigma\|^k N^2)$	10 hours†
Neighborhood kernels		
Triple kernel	$O(d^2HnN + d^2\|\Sigma\|^3N^2)$	327
Double kernel	$O(dHnN + d\|\Sigma\|^2N^2)$	67
Mismatch	$O(k^{m+1}\|\Sigma\|^m HnN + \|\Sigma^k\|N^2)$	-

† the running time is as in [6]

Notations used in the table: N-number of sequences, n-sequence length,
H is the sequence neighborhood size, $|\Sigma|$ is the alphabet size
k, m are mismatch kernel parameters ($k = 5, 6$ and $m = 1, 2$ in most cases)
M_σ is the profile neighborhood size, $k^m|\Sigma|^m \leq M_\sigma \leq |\Sigma^k|$

4 Experimental Results

We evaluate the proposed methods over three distinct sequence classification settings: categorization of the text documents using Reuters[2] data, protein homology detection using SCOP[3] dataset, and music genre classification[4]. We use standard benchmark datasets to allow comparison with previously published results. In all experiments, we use an existing implementation of SVM from a standard machine learning package SPIDER[5] with the linear kernel and default SVM parameters.

We normalize all kernel values $K(X, Y)$ using

$$K'(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X)K(Y, Y)}} \tag{8}$$

to remove the dependency between the kernel value and the sequence length.

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[3] http://scop.mrc-lmb.cam.ac.uk/scop/
[4] http://opihi.cs.uvic.ca/sound/genres
[5] http://www.kyb.tuebingen.mpg.de/bs/people/spider/

Results on the Reuters set and the music data are described in Sections 4.1.1 and 4.1.2, respectively. We present experimental results for the protein remote homology prediction problem under the supervised and semi-supervised settings on the SCOP dataset in Section 4.1.3 and the results for large-scale semi-supervised homology detection in Section 4.2. For repeatability purposes, datasets used in our experiments and supplementary data are made available at http://paul.rutgers.edu/~pkuksa/large-scale/spatial-kernels.htm

4.1 Supervised Setting

4.1.1 Reuters Dataset

We use Reuters-21578 data set for the experiments on text categorization. We limit our attention to the most frequent topics and documents labeled with a single topic class. The resulting dataset consists of 8 classes with 8983 documents. We use a standard ModApte split for the training and test sets. The documents are preprocessed by removing the stop words [6] , punctuation marks, and figures and word stemming. The resulting word alphabet contains $29,224$ distinct words. We evaluate performance of our classifiers using the F1 measure, accuracy, precision and recall (we use F1 measure with $\beta = 1$, i.e. $F1 = 2pr/(p + r)$, where p is the precision, and r is the recall). We also compute macro- and micro- averaged estimates of these measures to evaluate overall performance of the methods.

Table 2 depicts performance of the double(1,5) kernel, contrasted with other state-of-the art methods. The double kernel performs better on five classes (acq, earn, interest, ship, and trade) and produces state-of-the-art results on par with the other methods tuned specifically for text categorization. Accuracy, precision and recall are reported in Table 3. We note that our classifiers use a set of direct features (words and their combinations) with no additional feature weighting or filtering, unlike other approaches that rely on tf-idf embedding of the features to achieve the same level of performance. Furthermore, it is important to note that the double kernel can be computed very efficiently even for the large $29,224$ alphabet size. In particular, it takes only 76 seconds to evaluate an 8983-by-8983 kernel matrix on a PC with a single 2.8GHz CPU.

4.1.2 Music Genre Dataset

In the experiments on classification of music data by genre, we use a dataset [11] that contains 10 genres, with each genre represented by 100 audio sequences. The ten genres are Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock. Audio sequences of duration $\sim$ 30s are represented as strings over $\{1, ..., 1024\}$ alphabet of discretized 12-MFCC coefficients. We train our classifiers for each genre individually. To evaluate performance we use 10-fold cross-validation. Table 4 summarizes the performance of the double(1,5) kernel and the DWHCs method [11]. The double(1,5) kernel compares well with DWCH, a state-of-the-art method specifically developed for music classification. We also achieve better performance on five genres (disco,

[6] We use a standard list of 571 stop words available at http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

Table 2: Comparison on top Reuters categories (F1 scores).

Class	TF-IDF	KSG	Double
Acq	97.11	96.8	**97.72**
Crude	87.71	89.4	**90.11**
Earn	98.70	98.3	**99.02**
Grain	93.29	92.5	90.18
Interest	71.80	81.5	**81.63**
Money	77.61	84.0	82.66
Ship	72.97	81.9	**85.56**
Trade	84.26	90.2	**93.47**
Mean (macro-average)	85.43	89.33	90.05
Mean (micro-average)	93.18	93.9	94.51

KSG=key-substring-group features [19]

Table 3: Precision, recall, specificity, and accuracy for Reuters dataset

Class	Accuracy	Precision	Recall	Specificity
Acq	98.62	98.72	96.38	99.50
Crude	98.58	92.14	88.17	99.40
Earn	98.97	98.53	99.07	98.90
Grain	99.33	97.64	83.78	99.87
Interest	97.91	91.49	65.65	99.87
Money	97.63	85.63	79.89	98.98
Ship	98.97	82.80	88.51	99.35
Trade	99.09	89.08	91.38	99.46
Mean (macro-average)	98.54	92.00	86.60	99.39
Mean (micro-average)	98.59	95.98	93.08	99.42

Table 4: Performance on music genre classification.

#	Genre	DWCH†	Double (MFCC)
1	Blues	95.49 (1.27)	93.6 (4.77)
2	Classical	98.89 (1.1)	95.6 (1.35)
3	Country	94.29 (2.49)	94.3 (2.21)
4	Disco	92.69 (2.54)	**94.3 (1.41)**
5	Jazz	97.9 (0.99)	95.5 (2.27)
6	Metal	95.29 (2.18)	94.7 (1.42)
7	Pop	95.8 (1.69)	**96.2 (1.75)**
8	Hiphop	96.49 (1.28)	**97.1 (0.99)**
9	Reggae	92.3 (2.49)	**95.5 (1.58)**
10	Rock	91.29 (2.96)	**95.1 (1.66)**

†: DWCH=Daubechies Wavelet Coefficient Histograms [11]

hiphop, reggae, and rock). Note that the performance is achieved using the MFCC features that are calculated for short-time frames of sound (i.e. capture the local information in the music signals), unlike the DWHC method that uses features that capture both local and global information. By incorporating longer-term dependencies using our double kernel with $k = 1, d = 5$, the performance improves significantly compared to the use of MFCC features alone as shown in Table 5, where we compare classification accuracy in the multiclass setting.

Table 5: Multi-class classification performance on music genre data (accuracy/standard deviation)

Method	Accuracy
MFCC	53.7 (4.11)
Double-(1,5) (MFCC)	67.6 (4.17)

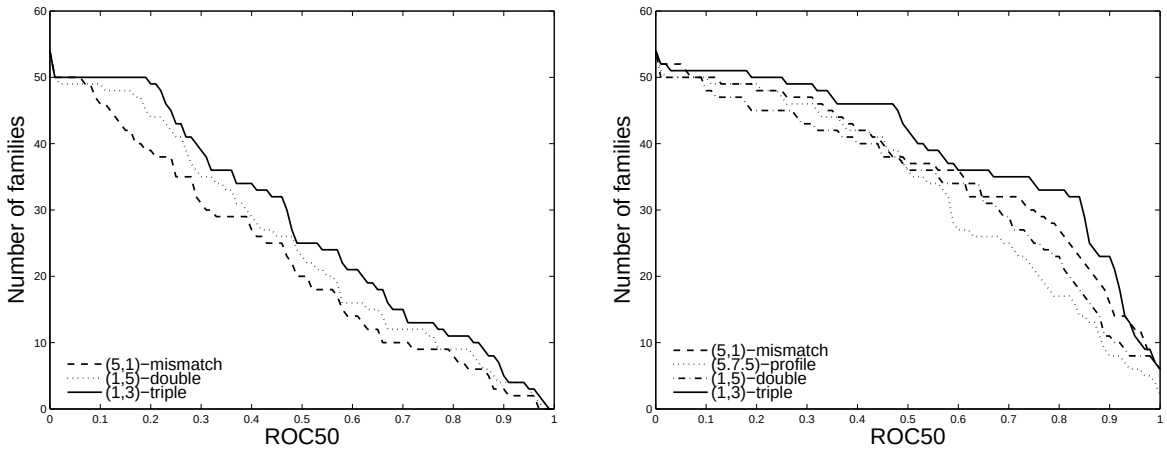
4.1.3 SCOP Dataset

We use the benchmark dataset [18] to perform our experiments on *remote* homology detection. The dataset contains 54 target families from SCOP 1.59 [13] with $7,329$ isolated domains. Only $2,862$ domains out of $7,329$ are labeled, which allows to perform experiments in both supervised (labeled sequences only) and semi-supervised (labeled and unlabeled sequences) settings. Our experimental setup is the same as that of Jaakkola [3, 7]. In each of the 54 experiments, to simulate the remote homology problem one of the families is held out for testing (i.e. the classifies are tested on the the sequences from unseen families). Different instances of this dataset have been used as a gold standard for protein remote homology detection in various studies [5, 15, 18, 12, 10].

In the supervised setting, only the labeled sequences participate in the experiments. In the semi-supervised experiments, we also use the unlabeled dataset consisting of the remaining unlabeled sequences (4467 sequences).

Performance of all methods is evaluated using the *Receiver Operating Characteristic* (ROC) and ROC-50 [2] scores. The ROC-50 score is the (normalized) area under the ROC curve computed up to 50 false positives. With a small number of positive testing sequences and a large number of negative testing sequences, the ROC-50 score is typically more indicative of the prediction accuracy of a homology detection method than the ROC score. This is also a standard measure used in protein homology literature.

We compare the performance of our proposed methods with previously published state-of-the-art methods [12, 10] under the supervised learning setting in Table 6. The table also displays the dimensionality of the induced features and the observed experimental running times, measured on a single 2.8GHz CPU, for constructing the 7329x7329 kernel matrix. The results clearly indicate that the proposed kernels (doubles and triples) not only show significantly better performance than the existing methods, but also require substantially less computational time. Also, as can be seen from the comparison with the gapped kernels, the addition of the spatial information substantially improves the classification performance. A more thorough insight on the performance can be gleaned from the ROC-50 plot in Figure 2(a). In the plot, the horizontal axis corresponds to the ROC-50 scores and the vertical axis denotes the number of classes, out of 54, with an equivalent or higher ROC-50 score. For clarity, we do not display the plot for every method. Our results indicate that both double and triple kernels dominate the mismatch(5,1) kernel, as well as other supervised methods.



(a) SCOP dataset. Supervised setting.

(b) SCOP dataset. Semi-supervised setting

Figure 2: Left panel: Comparison of the performance (ROC50) in a supervised setting. Spatial kernels (triples and doubles) outperform other supervised methods. Right panel: Comparison of the performance (ROC50) in a semi-supervised setting using SCOP 1.59 as the unlabeled dataset. Spatial triple kernel outperforms both profile and mismatch neighborhood kernels.

Table 6: Comparison of the performance on the SCOP 1.59 dataset under the supervised setting.

Method	ROC	ROC50	# dim.	Time (s)
(5, 1)-mismatch	0.8749	0.4167	3200000	938
SVM-pairwise†	0.8930	0.4340	-	-
gapped(6,2)[8]	0.8296	0.3316	400	-
gapped(7,3)	0.8540	0.3953	8000	-
(1,5) double	0.8901	0.4629	2000	54
(1,3) triple	0.9148	0.5118	72000	112

†: directly quoted from [12]

4.2 Large-Scale Semisupervised experiments

In many cases large sets of unlabeled data are available and, if used for training classifiers, can potentially increase classification accuracy. We demonstrate applications of our methods to large unlabeled databases of protein sequences to learn even more accurate classifiers. We use three unlabeled databases for these experiments, with the sizes varying from several thousand sequences to more than 500,000 sequences.

To incorporate the unlabeled data we use kernel smoothing (Equation 6) of [18]. The neighborhood graph is established by querying, for each sequence X, the unlabeled dataset using 2 iterations of PSI-BLAST. We recruit the sequences with low e-values (≤ 0.05) to be the neighbors of X. To adhere to the true semi-supervised setting, *all sequences in the unlabeled datasets that are identical to any test sequence are removed*.

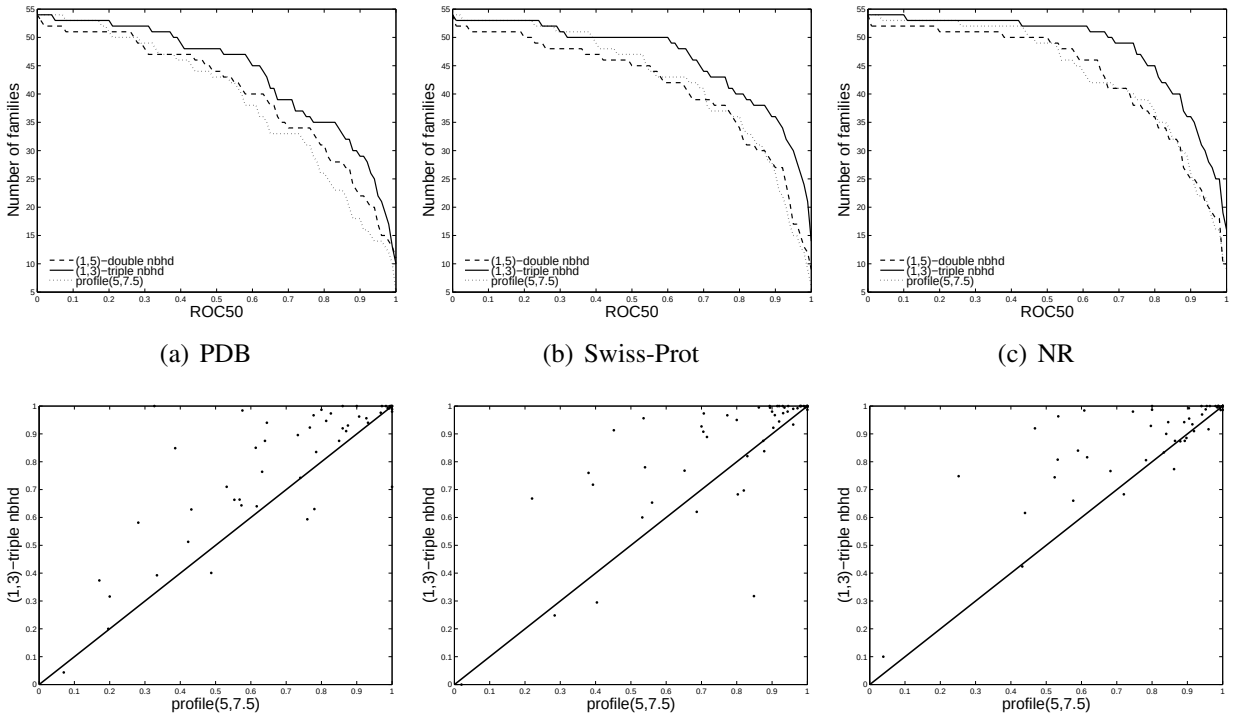
The semi-supervised experiments are performed on the unlabeled set extracted from SCOP ($4,467$ sequences), the *non-redundant* (NR) dataset, and Swiss-Prot[7]. The latter two sets contain $101,602$ and $534,936$ sequences, respectively. Table 7 summarizes the characteristics of the unlabeled datasets used in this study. The second column shows the size of the unlabeled datasets while the third column shows the mean, median and maximum number of neighbors per sequence recruited using the corresponding unlabeled dataset.

Table 7: Dataset characteristics. The number of neighbors indicates number of sequences recruited in various unlabeled datasets (mean/median/max).

Dataset	# Seq	# Neighbors
Scop	4467	11/2/465
Swiss-Prot	101602	56/28.5/385
NR	534936	114/86/490

We perform all semi-supervised experiments on a 2.8GHz processor with 2GB of memory. Evaluation of the mismatch neighborhood kernels is computationally demanding and typically cannot be accomplished on a single machine for anything but relatively small unlabeled datasets.

[7]We use the same version as the one employed in [18] for comparative analysis of performance.



(a) PDB (b) Swiss-Prot (c) NR

(d) PDB (e) Swiss-Prot (f) NR

Figure 3: In the upper panel, we show the ROC-50 plots of three different features using PDB, Swiss-Prot and NR databases as unlabeled datasets, respectively. In the lower panel, we show the scatter-plot of ROC-50 scores of the triple-(1,3) kernel (vertical) and the profile(5,7.5) kernel (horizontal). Any point above the diagonal line in the figures (d),(e),(f) indicates better performance for the triple-(1,3) kernel.

Therefore, the results for the mismatch neighborhood kernel can only be shown using the previously published summary statistics. In the upper panel of Figure 3, we show the ROC-50 plots of the double-(1,5) neighborhood, triple-(1,3) neighborhood and profile(5,7.5) kernels using unlabeled SCOP (first column), Swiss-Prot (second column) and NR (third column) sequence databases as the unlabeled datasets. The ROC-50 curves of the triple-(1,3) neighborhood kernel on all unlabeled datasets consistently show strong dominance over those of the other two kernels. Furthermore, the performance of the double-(1,5) neighborhood kernel is on par with that of the profile(5,7.5) kernel. In the lower panel, we show the scatterplots of the ROC-50 scores of the triple-(1,3) kernel and the profile(5,7.5) kernel. Any point falling above the diagonal line in the figures indicates better performance of the triple-(1,5) kernel over the profile(5,7.5) kernel. As can be seen from these plots, the triple kernel outperforms the profile kernel on all three datasets (43/37/34 wins and 3/5/10 ties on SCOP, Swiss-Prot, and NR datasets, respectively). We also show the statistical significance of the observed differences between pairs of methods on various unlabeled datasets in Table 8. All the entries in the table are the p-values of the Wilcoxon signed rank test using the ROC-50 scores. For each unlabeled dataset, we highlight the method that has the

Table 8: Statistical significance (p-values of the Wilcoxon signed rank test) of the observed differences between pairs of methods (ROC-50 scores) on unlabeled datasets. Triple denotes the triple-(1,3) neighborhood kernel, double denotes the double-(1,5) neighborhood kernel, mismatch denotes the mismatch(5,1) neighborhood kernel, and profile denotes the profile(5,7.5) kernel.

SCOP 1.59				
	mismatch	profile	double	triple
mismatch	-	2.245e-03	1.804e-02	3.570e-06
profile	2.245e-03	-	2.874e-01	9.615e-09
double	1.804e-02	2.874e-01	-	6.712e-06
triple	3.570e-06	9.615e-09	6.712e-06	-
PDB				
	double	triple	profile	
double	-	1.017e-01	4.762e-02	
triple	1.017e-01	-	7.666e-06	
profile	4.762e-02	7.666e-06	-	
Swiss-Prot				
	double	triple	profile	
double	-	9.242e-05	4.992e-01	
triple	9.242e-05	-	2.419e-04	
profile	4.992e-01	2.419e-04	-	
NR				
	double	triple	profile	
double	-	8.782e-06	9.762e-01	
triple	3.570e-06	9.615e-09	6.712e-06	-

best overall performance. The triple-(1,3) kernel consistently outperforms all other kernels, with high statistical significance.

The overall prediction performance of our methods in comparison with the state-of-the-art methods over various unlabeled datasets is shown in Table 9. For each unlabeled dataset, we highlight the best ROC and ROC-50 scores; on all datasets, the triple-(1,3) neighborhood kernel achieves the best performance. Furthermore, we achieve such performance by only 2 PSI-BLAST iterations in contrast to the five iterations necessary for the profile(5,7.5) kernel.

4.3 Running time analysis

Running time analysis is particularly revealing, in conjunction with the performance scores described previously. We show the observed running time of the methods in Table 10. Running time measurements for our methods are done on a 2.8GHz CPU. All competing methods were highly optimized.

For supervised experiments on SCOP dataset, we compute the full 7329x7329 kernel matrix

Table 9: The overall prediction performance of all compared methods over various unlabeled datasets.

SCOP Unlabeled	ROC	ROC50
(5, 1)-mismatch neighborhood	0.9093	0.6745
(5,7.5)-profile	0.9190	0.6069
(1,5)-double neighborhood	0.9282	0.6383
(1,3)-triple neighborhood	**0.9382**	**0.7262**
Swiss-Prot		
double-(1,5) neighborhood	.9582	.7701
triple-(1,3) neighborhood	**.9732**	**.8605**
profile(5,7.5)	.9709	.7914
mismatch nbhd†	.955	.810
NR		
double-(1,5) neighborhood	.9720	.8076
triple-(1,3) neighborhood	**.9861**	**.8944**
profile(5,7.5)-2 iterations	.9734	.8151
profile(5,7.5)-5 iterations‡	.984	.874

†: directly quoted from [18]
‡: directly quoted from [6]

for all methods. For the semi-supervised setting (neighborhood kernels), we report running time on the datasets used (i.e. Swiss-Prot, and non-redundant (NR) databases). We compare the running time of the methods in large-scale semisupervised setting in Table 11. For neighborhood kernels, we report the average running time on the unlabeled datasets used.

The difference in complexity between profile and mismatch kernels and the proposed kernels (as discussed in Section 3.2) leads to order-of-magnitude improvements in the running times of the sample kernels over mismatch and profiles, with a difference more pronounced in a large-scale settings. This provides a clear indication of the potential applicability of the proposed method to very large datasets.

Table 10: Running time, (s)

Dataset	Alphabet size	# Seq.	Seq. length	Double, (s)	Triple, (s)
Reuters	29,224	8983	70	76	97
Music genre	1024	1000	6895	99	141
SCOP	20	7329	172	54	112
Swiss-Prot	20	101,602	332	64	276
NR	20	534,936	367	81	587

Table 11: Running time comparison

Method	Running time (s)
Triple kernel	112
Double kernel	54
Mismatch	948
Profile kernel	2 hours
Neighborhood kernels (semi-supervised)	
Triple kernel	327
Double kernel	67
Mismatch neighborhood	-

5 Discussion

In spite of the original biological inspiration for our features, improvement over existing methods, sometimes substantial, is constantly observed in all applications in the experimental section. Compared to the spectrum-like string features, the feature sets induced by our kernels cover segments of variable length (for example, $2-6$ residues in the case of the double-$(1,5)$ kernel), whereas the spectrum-like features remain restricted to fixed-length segments (e.g., 5 or 6 residues long.) The spectrum-like features have a close relationship with the *n-gram* representation and fixed-order Markov chains. On the other hand, the strength of our features is derived from the fact that variable orders of Markov chains are considered simultaneously, as depicted in Figure 4. In each chain, the first circle corresponds to the observation at position t (X_t) and the second circle corresponds to X_{t+1}. A direct consequence of using the spatial feature is that local, and short-term dependencies among symbols in the sequences are taken into consideration.

Sampling at different resolutions also allows one to capture similarity in the presence of more complex substitution, insertion, and deletion processes, whereas sampling at a fixed resolution, the approach used in spectrum-like features, limits the sensitivity in the case of multiple insertions/deletions or substitutions. In Figure 5 we compare the spectrum-like features with our proposed spatial features, extracted from a string: S='HKYNQLIM'. The symbol 'X' in the mismatch-(k,m) representation corresponds to an arbitrary symbol in the alphabet set. As a result, each mismatch feature basis in the figure actually corresponds to $|\Sigma|$ features. With such representation, the number of neighboring k-mers induced by an observation grows exponentially with the choice of m, the number of mismatches allowed. On the other hand, the spatial features scans the strings at different resolutions and the number of features induced is small. We further provide an example of how different features handle substitutions in Figure 6. In the example, the two strings, S and S' are similar but only moderately conserved. We show the set of features induced by the mismatch and double kernels with the common features, for each feature set, in bold font and red color. In a moderately conserved region like this, the spatial features handle substitutions by attempting to find a match in the next position. On the other hand, for the mismatch features to handle such a region, the number of allowed mismatches m, needs to be increased. This, in turn, results in high computational cost.

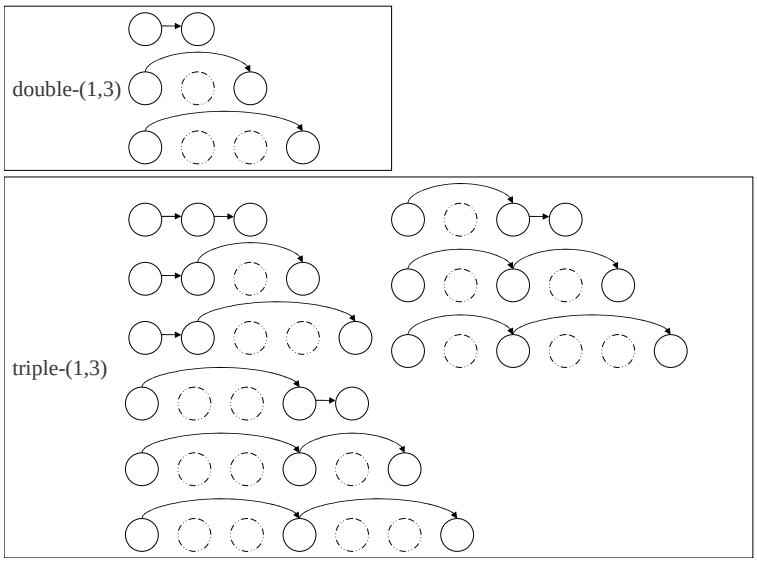


Figure 4: The spatial features incorporate different orders of Markov chains simultaneously. In each chain, the first circle corresponds to the observation at position t (X_t), and the second $X_{t+1} \cdots$ etc.

S = HKYNQLIM

spectrum-5
HKYNQ
KYNQL
YNQLI
NQLIM

mismatch(5,1)	
XKYNQ	XYNQL
HXYNQ	KXNQL
HKXNQ	KYXQL
HKYXQ	KYNXL
HKYNX	YKNQX
XNQLI	XQLIM
YXQLI	NXLIM
YNXLI	NQXIM
YNQXI	NQLXM
YNQLX	NQLIX

double-(1,5)				
HK	H_Y	H__N	H___Q	H____L
KY	K_N	K__Q	K___L	K____I
YN	Y_Q	Y__L	Y___I	Y____M
NQ	N_L	N__I	N___M	
QL	Q_I	Q__M		
LI	L_M			
IM				

Figure 5: Comparison of features extracted by the spectrum-like and spatial kernels. In the mismatch representation, each symbol 'X' represent an arbitrary symbol in the alphabet set. As a result, each feature basis actually corresponds to $|\Sigma|$ features.

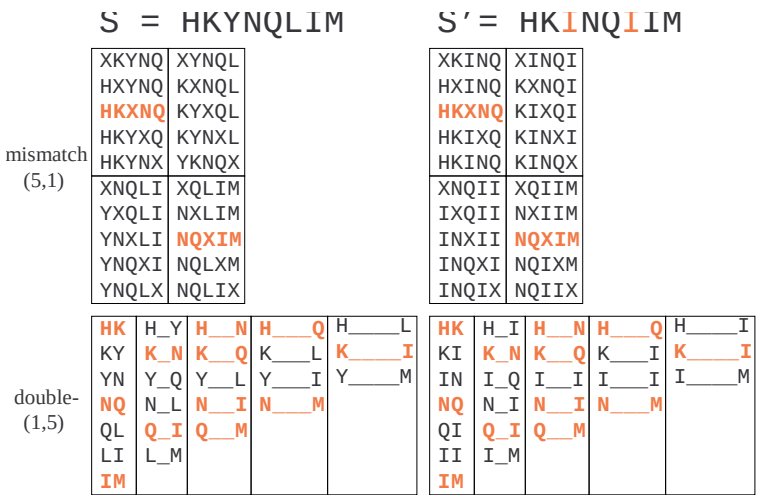


Figure 6: Differences in handling substitutions by the spectrum-like and spatial features. The spatial features handle substitutions by attempting to find a match in the next position. For the spectrum-like kernel to handle such a moderately conserved region, the number of allowed mismatches needs to be increased.

Compared to mismatch/profile kernels, the feature sets induced by our kernels cover segments of variable length (e.g., $2-6$ residues in the case of the double-$(1,5)$ kernel), whereas the mismatch and profile kernels cover segments of the fixed length (e.g., 5 or 6 residues long) as illustrated in Figure 1. Sampling at different resolutions also allows to capture similarity in the presence of more complex substitution, insertion, and deletion processes, whereas sampling at a fixed resolution, the approach used in mismatch and spectrum kernels, limits the sensitivity in the case of multiple insertions/deletions or substitutions. Increasing the parameter m (number of mismatches allowed) to accommodate the multiple substitutions, in the case of mismatch/spectrum kernels, leads to an exponential growth in the neighborhood size, and results in high computational complexity. The proposed features also capture short-term dependencies and interactions between local sequence features by explicitly encoding the spatial information; in contrast, such information is not present in the gapped/subsequence kernels [8]. Also, in a weighted version of the subsequence kernel, where each instance (subsequence) of a particular k-mer is weighted inversely proportional to the length of the subsequence, the count for a particular k-mer is the sum of such weights; when sequences are matched under the weighted subsequence kernel, the final counts (i.e. sum of weights) are compared and no distinction is made as to how the features were positioned in the sequences, i.e. the information on the spatial configuration of the features within the sequence is not retained. However, as suggested by previous biological studies (e.g. [4]), the spatial information such as distances between key positions on some specific secondary structures can play a key role in capturing inherent characteristics of superfamilies. We further illustrate differences between the proposed kernels and gapped/subsequence kernels for the case when the basic features (individual samples) of the spatial sample kernels are single characters in the Equations 9 (spatial kernels)

and 10 (gapped/subsequence kernels) below:

$$K(X,Y) = \sum_{\substack{(a_1,d_1,\ldots,d_{t-1},a_t) \\ a_i \in \Sigma, 0 \leq d_i < d}} \begin{array}{c} c((a_1,d_1,\ldots,d_{t-1},a_t)|X) \cdot \\ c((a_1,d_1,\ldots,d_{t-1},a_t)|Y) \end{array} \tag{9}$$

$$K_g(X,Y) = \sum_{(a_1,a_2,\ldots,a_t)} \left(\sum_{d_1,d_2,\ldots,d_{t-1}} c((a_1,d_1,\ldots,d_{t-1},a_t)|X) \right) \cdot \left(\sum_{d_1,d_2,\ldots,d_{t-1}} c((a_1,d_1,\ldots,d_{t-1},a_t)|Y) \right) \tag{10}$$

where $c(\bullet|X)$ is the (weighted) count and $\sum_{i=1}^{t-1} d_i = g - t$ for the gapped (g,t) kernels. Note how the spatial configuration information is integrated out in the gapped/subsequence kernels.

6 Conclusion

We present a new family of sparse spatial string kernels that demonstrate state-of-the-art performance in the sequence classification settings of the supervised and large-scale remote homology detection on the SCOP benchmarks and text categorization tasks. The proposed methods yield significantly improved homology detection performance compared to a number of existing state-of-the-art approaches while require modest computational resources. We also achieve the state-of-the-art performance on the text classification and substantial improvement in accuracy for music genre classification. We show that our methods can work with large databases in supervised and semi-supervised settings. This opens the possibility for the proposed methodology to be readily applied to other challenging problems in sequence analysis and text mining.

References

[1] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, June 2006. 1

[2] M. Gribskov and N. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching, 1996. 11

[3] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. In *Journal of Computational Biology*, volume 7, pages 95–114, 2000. 10

[4] A. E. Kister, A. V. Finkelstein, and I. M. Gelfand. Common features in structures and sequences of sandwich-like proteins. *PNAS*, 99(22):14137–14141, 2002. 18

[5] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *CSB '04: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB'04)*, pages 152–160, August 2004. 3, 5, 10

[6] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *J Bioinform Comput Biol*, 3(3):527–550, June 2005. 3, 6, 7, 15

[7] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. S. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *CSB*, pages 152–160, 2004. 1, 2, 10

[8] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004. 12, 18

[9] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002. 2, 4, 5

[10] C. S. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for svm protein classification. In *NIPS*, pages 1417–1424, 2002. 1, 2, 3, 4, 5, 10, 11

[11] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, New York, NY, USA, 2003. ACM. 8, 10

[12] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *RECOMB*, pages 225–232, 2002. 1, 2, 10, 11, 12

[13] L. Lo Conte, B. Ailey, T. Hubbard, S. Brenner, A. Murzin, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Res.*, 28:257–259, 2000. 10

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected apllications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Kaufmann, San Mateo, CA, 1990. 2

[15] H. Rangwala and G. Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005. 10

[16] S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence svm classifiers. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 848–855, New York, NY, USA, 2005. 1, 2

[17] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. 2

[18] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W. S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005. 3, 4, 5, 6, 7, 10, 12, 15

[19] D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 474–483, New York, NY, USA, 2006. ACM. 9