# FAST AND ACCURATE MULTI-CLASS PROTEIN FOLD RECOGNITION WITH SPATIAL SAMPLE KERNELS

Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic*

*Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA*

*Email: {pkuksa;paihuang;vladimir}@cs.rutgers.edu*

Establishing structural or functional relationship between sequences, for instance to infer the structural class of an unannotated protein, is a key task in biological sequence analysis. Recent computational methods such as profile and neighborhood mismatch kernels have shown very promising results for protein sequence classification, at the cost of high computational complexity. In this study we address the multi-class sequence classification problems using a class of string-based kernels, the sparse spatial sample kernels (SSSK), that are both biologically motivated and efficient to compute. The proposed methods can work with very large databases of protein sequences and show substantial improvements in computing time over the existing methods. Application of the SSSK to the multi-class protein prediction problems (fold recognition and remote homology detection) yields significantly better performance than existing state-of-the-art algorithms.

## 1. INTRODUCTION

Protein homology detection and structure annotation are fundamental problems in computational biology. With the advent of large-scale sequencing techniques, experimental elucidation of an unknown protein sequence function and structure becomes an expensive and tedious task. Currently, there are more than $61$ million DNA sequences in GenBank [1], and approximately $349,480$ annotated and $5.3$ million unannotated sequences in UNIPROT [2]. The rapid growth of sequence databases makes development of computational aids for functional and structural annotation a critical and timely task.

The goals of remote homology detection and remote fold detection are to infer functional and structural information based only on the primary sequence of an unknown protein. In this study, we address these two problems in the context of Structural Classification of Proteins (SCOP) [3]. In SCOP, a manually curated protein data set derived from PDB [4], sequences are grouped into a tree hierarchy containing classes, folds, superfamilies, and families, from root to leaf. The difficulty of the remote homology and structural similarity detection tasks arises from low sequence identities among proteins on the superfamily and fold levels.

Early approaches to computationally-aided homology detection, such as BLAST [5] and FASTA [6], rely on pairwise alignment. Later methods, such as profiles [7] and profile hidden Markov models (profile HMM) [8], collect aggregate statistics from a group of sequences known to belong to the same family or superfamily. Studies have shown that these *generative* methods are accurate in detecting close homology (family detection) with moderate sequence identities. However, when sequence identities are low, which is typical for superfamilies (remote homology) and folds (structural similarity), generative method becomes insufficient and therefore *discriminative* methods are necessary. For protein remote homology detection, several types of discriminative kernel methods were proposed, for example, SVMFisher [9] by Jaakkola *et al*. and the class of string kernels [10, 11] by Leslie *et al*. Both classes of kernels demonstrated improved discriminative power over generative methods. Most of the studies formulated *binary-class* problems. In a different task, fold recognition [12–15], studies formulated *multi-class* learning problems. Ding *et al.* , in [12], proposed to extract features based on amino acid compositions and physico-chemical properties and in [13], Ie *et al.* extended the profile kernel [11] framework with adaptive codes for fold recognition. Both fold recognition methods showed promising results on detecting structural similarities based on primary sequences only.

Protein classification problems are typically characterized by few *positive training* sequences accompanied by a large number of negative training examples, which may result in weak classifiers. Enlarging the training sample size by experimentally labeling the sequences is

*Corresponding author.

costly, leading to the need to leverage *unlabeled data* to refine the decision boundary. The profile [16] and the mismatch neighborhood [17] kernels use large unlabeled datasets and show significant improvements over the sequence classifiers trained under the supervised setting. However, the promising results are offset by a significant increase in computational complexity, hindering wide application of such powerful tools.

In this study, we consider a new family of string-based kernel methods, the sparse spatial sample kernels (SSSK), for the multi-class sequence classification tasks. The proposed kernels are induced by the features that sample the sequences at different resolutions while taking mutations, insertions and deletions into account. These features are low-dimensional and their evaluation incurs low computational cost. Such characteristics open the possibility for analyzing very large unlabeled datasets under the semi-supervised setting with modest computational resources. The proposed methods perform significantly better and run substantially faster than existing state-of-the-art algorithms, including the profile [16, 11] and neighborhood mismatch [17] kernels, for both remote homology and fold detection problems on three well-known benchmark datasets. Moreover, in a multi-class setting, use of SSSK does not incur the need for formulating a complex optimization problem, as suggested in [13, 14]; we obtain our performance in a straightforward manner with no parameter adjusting.

## 2. BACKGROUND

In this section, we briefly review existing state-of-the-art methods, under supervised and semi-supervised learning paradigms. We also briefly discuss the multi-class learning problem.

**Supervised Methods**: The spectrum-like kernels, the state-of-the-art string kernels in the supervised setting, *implicitly* map a sequence $X$ to a $|\Sigma|^d$-dimensional vector, where $\Sigma$ is the alphabet set. The $mismatch(k, m)$ kernel [10] relaxes exact string matching by allowing up to $m$ mismatches, or substitutions, to accommodate the mutation process. 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.

**Semi-supervised Methods**: The performance of the supervised methods depends greatly on the availability and quality of the labeled training data. In the presence of limited number of labeled training sequences, the performance of the classifiers estimated under such setting, though promising [9, 10], is still sub-optimal. Enlarging the size of the training set will improve the accuracy of the classifiers; however, the cost of doing so by experimentally obtaining functional or other group labels for large numbers of protein sequences may be prohibitive, but the *unlabeled data* can still be leveraged to refine and potentially improve the decision boundary. Recent advances in computational methods for remote homology prediction have relied heavily on the use of such data sources [11, 17, 13]. The profile kernel [11] uses unlabeled data directly by constructing a profile and using local information in the profile to estimate the *mutation neighborhood* of all $k$-mers. Construction of profiles for each sequence may incur high computational cost since highly diverged regions in profiles may result in a mutational neighborhood size exponential in the number of $k$-mers.

**Multi-class classification**: One way to solve the multi-class learning problem is to directly formulate a multi-class optimization problem, as done in [18, 19]. An alternative is to combine *binary* predictors using one-vs-one or one-vs-rest schemes. For instance, in a one-vs-rest scheme, $|Y|$ classifiers are estimated, where $Y$ is the output space, and the predicted class, $\hat{y}$, corresponds to the highest scoring classifier output (Equation 1) where $f_y$ denotes the binary classifier for class $y$. In contrast to the simple decision rule, Ie *et al*. in [14] proposed to use the adaptive codes to tackle the multi-class fold recognition problem with decision rule in Equation 2 where $\times$ denotes component-wise multiplication, $\mathbf{f}(x)$ a 1-by-$(n_f + n_s)$ output vector from the binary classifiers and $C_y$ a binary code matrix and $W$ the parameters to estimate. Under such framework, one needs to train at least $n_f + n_s + n_{fa}$ independent binary classifiers, where $n_f$, $n_s$, and $n_{fa}$ denote the number of folds, superfamilies and families, respectively.

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} f_y(x), \tag{1}$$

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} (W \times \mathbf{f}(x)) C_y, \tag{2}$$

The practice of using one-vs-rest classifiers (Equation 1) has received both favorable and unfavorable comments. In [20], showed that formulating complex optimization problems, such as error-correcting codes [21], does not offer any advantage over the simple decision rules, for example, Equation 1. In contrast, Ie *et al*. in [14] argued

that the simple decision rule can only cope with problems with small number of classes. At present, no clear evidence indicates one decision rule dominates the other. In this study, we use one-vs-rest scheme (Equation 1) and only estimate $n_f$ or $n_s$ binary classifiers for fold and superfamily detection, respectively.

## 3. THE SPARSE SPATIAL SAMPLE KERNELS

Sparse spatial sample kernels (SSSK) present a new class of string kernels that effectively model complex biological transformations (such as highly diverse mutation, insertion and deletion processes) and can be efficiently computed. The SSSK family of kernels, parametrized by three positive integers, assumes the following form:

$$K^{(t,k,d)}(X,Y) = \sum_{\substack{(a_1,d_1,\ldots,d_{t-1},a_t) \\ a_i \in \Sigma^k, 0 \le d_i < d}} C(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t|X) \cdot C(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t|Y), \quad (3)$$

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 $a_2$, separated by $d_2$ characters from $a_3$, etc.) in $X$. This is illustrated in Figure 1.

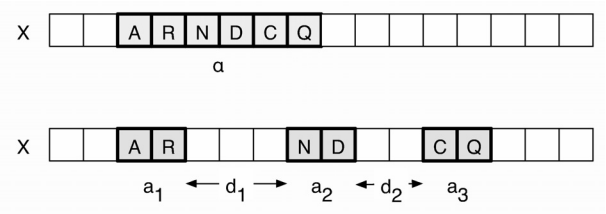



**Fig. 1.** Contiguous $k$-mer feature $\alpha$ of a traditional spectrum/mismatch kernel (top) contrasted with the sparse spatial samples of the proposed kernel (bottom).

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 lie 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). These sample string kernels, unlike the family of spectrum kernels [10, 11], not only take into account the feature counts, but also include spatial configuration information, *i.e.* how the features are positioned in the sequence. The spatial information can be critical in establishing similarity of sequences under complex transformations such as the evolutionary processes in protein 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 Section 5.1.

The use of short features can also lead to significantly lower computational complexity of the kernel evaluations. 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) ($t = 3$, $k = 1$, $d = 3$) and double-(1,5) ($t = 2$, $k = 1$, $d = 5$) feature sets, the dimensionalities are $72,000$ and $2,000$, respectively, compared to $3,200,000$ for the spectrum-($k$), mismatch($k$,$m$) [10], and profile($k$,$\sigma$) [11] kernels with the common choice of $k = 5$. The proposed kernels can be efficiently computed using sorting and counting. To compute the kernel values, we first extract the features from the sequences and sort the extracted features in linear time with counting sort. Then we count the number of distinct features and update the kernel matrix. For $N$ sequences of the longest length $n$ and $u$ distinct features, computing the kernel matrix takes linear $O(dnN + min(u, dn)N^2)$ time. We provide a comprehensive comparison of the computational complexity and running times in Section 5.

### 3.1. Spatial sample neighborhood kernels

The proposed kernel can be extended to accommodate the semi-supervised learning setting for sequence classification, for example, as in [17]. Denote $\Phi^{orig}(X)$ as the original representation of the sequence $X$ and $N(X)$ the *neighborhood* of $X$[a]; then, the smoothed

[a] We discuss how to define $N(X)$ in Section 4.2.

re-representation $\Phi^{new}(X)$ using the unlabeled data is defined as Equation 4 with the corresponding kernel, $K^{nbhd}(X,Y)$, in Equation 5:

$$\Phi^{new}(X) = \frac{1}{|N(X)|} \sum_{X' \in N(X)} \Phi^{orig}(X'), \quad (4)$$

$$K^{nbhd}(X,Y) = \sum_{X' \in N(X), Y' \in N(Y)} \frac{K(X',Y')}{|N(X)||N(Y)|}. \quad (5)$$

Note that in this setting, unlike in the traditional semi-supervised learning setting, both *training* and *test* sequences assume a new representation. Weston *et al*. in [17] showed that the discriminative power of the neighborhood mismatch kernel improves significantly; however such neighborhood kernel evaluation requires substantially longer computational time, as indicated by the Weston *et al*. in [16, 17].

## 4. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed methods on multi-class remote fold recognition and multi-class remote homology detection in Sections 4.3 and 4.4, and multi-class fold as well as superfamily prediction in Section 4.5. We use three standard benchmark datasets to allow comparison with previously published results. The datasets used in our experiments and supplementary data are made available at `http://seqam.rutgers.edu/projects/bioinfo/spatial-kernels/csb08.html`.

### 4.1. Datasets

**Ding and Dubchak dataset**[12]: Ding *et al*. designed a challenging fold recognition data set [b], which has been used as a benchmark in many studies, for example [13]. The data set contains sequences from 27 folds divided into two *independent* sets, such that the training and test sequences share less than 35% sequence identities and within the training set, no sequences share more than 40% sequence identities.

**Remote fold detection data set** [14]: Melvin *et al*. derived this data set from SCOP 1.65 [3] for the tasks of multi-class remote fold detection. The data set contains 26 folds, 303 superfamilies and 652 families for training with 46 superfamilies held out for testing to model remote fold recognition setting.

**Remote homology detection data set** [14]: Ie *et al*. prepared a different data set for remote homology detection in a similar fashion. The derived data set contains 74 superfamilies and 544 families for training with from 110 families held out for testing.

The three datasets lead to a 27-, a 26- and a 74-way multi-class classification problems.

### 4.2. Settings, parameters used and performance measures

For all kernel methods, we perform kernel normalization to remove the dependencies between the kernel values and sequence lengths:

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

In all experiments, we use an existing implementation of SVM from a standard machine learning package SPIDER[c] with the linear kernel and default SVM parameters. For kernel smoothing (Equation 5) on each sequence $X$, we query the unlabeled database with 2 PSI-BLAST iterations and recruit the sequences with low e-values ($\leq 0.05$) to form the neighborhood $N(X)$. We use Swiss-Prot [22] (moderate size) and the *non-redundant* (large size) sets as unlabeled databases.

To evaluate our classifiers, we use zero-one and balanced error rates, as well as top-5 error rates, as in [12–14]. In addition, we also use standard performance measures from the information retrieval literature: sensitivity (recall) (r), precision (p) and $F1 = 2pr/(p+r)$ scores.

### 4.3. Comparison on Ding and Dubchak benchmark

We compare the performance of our methods under supervised and semi-supervised settings with previously published methods on Ding and Dubchak benchmark data set in Table 1. As can be seen from the table, our spatial kernels achieve higher performance compared to the state-of-the-art classifiers (we highlight the best performance in each category). Under the supervised setting, the triple spatial features consistently demonstrate better overall performance. The observed higher precision of the mismatch and profile kernels is achieved at

[b] `http://ranger.uta.edu/~chqding/bioinfo.html`
[c] `http://www.kyb.tuebingen.mpg.de/bs/people/spider`

the cost of lower recall rates; the F1 score, a function of both recall and precision measures, suggests that the triple spatial features achieve better performance. Under semi-supervised learning, we again observe better overall performance of spatial kernels. We also note that the spatial kernels strongly outperform the profile kernel evaluated in the same setting with 2 PSI-BLAST iterations as used by all our methods.

To demonstrate the benefit of leveraging unlabeled data, in Figure 2 we contrast the confusion matrices under the supervised (Figure 2(a)) and semi-supervised (Figure 2(b)) settings using the triple(1,3) feature set. Similar to Damoulas *et al*. [23], we observe that in the supervised setting, testing examples in classes with fewer training examples tend to be incorrectly assigned to two overly represented folds (7 and 16). However, such problem is alleviated in the semi-supervised setting when we enlarge the training sets with neighboring sequences which, in turn, reinforces class assignments of the target proteins. Under the supervised setting, we observe 3 folds with accuracy higher than $90\%$, one of which achieves $100\%$ accuracy, while under the semi-supervised setting, we observe 9 folds with accuracy higher than $90\%$, 7 of which achieves $100\%$ accuracy.

### 4.4. Remote fold and remote homology detection

We report the performance on remote fold and homology detection problems on the SCOP 1.65 benchmark data set in this section.

Table 2 corresponds to the remote fold prediction problem and we highlight the best performance in each category. The spatial kernels consistently outperform the one-vs-rest mismatch and profile kernels, as well as the best profile NR with adaptive codes, under both supervised and semi-supervised settings.

We also note that the performance of profile NR with adaptive codes was obtained by solving a complex optimization problem of minimizing the balanced error, whereas all other methods use the simple one-vs-rest decision rule (Equation 1). The use of the adaptive and error-correcting codes offers no clear evidence of advantage over the simple decision rule.

Table 3 summarizes results of the remote homology (superfamily) detection problem. Under the supervised learning, the spatial kernels consistently outperform the state-of-the-art mismatch(5,1) kernel. We make the same observation under the semi-supervised setting with the Swiss-Prot data set. With the non-redundant database, the spatial kernels consistently outperform the profile(5,7.5) kernel and, in particular, the triple (1,3) kernel shows the best top-5 error rate of $8.6\%$. We similarly outperform the highly optimized one-vs-rest profile NR by all measures. Further, compared to the profile NR, estimated using adaptive codes with complex optimization, our triple kernel with the simple one-vs-rest decision rule achieves better top-5 error and comparable error rates.

Substantial improvement in performance comes from the use of unlabeled data. We observe that use of a large unlabeled data set (non-redundant) results in significantly better performance over a Swiss-Prot data set, which is only moderate in size. The non-redundant data set provides a richer neighborhood for the data sequences, improving sensitivity of the methods. We take a further look at this improvement in Section 5.3.

### 4.5. Multi-class protein fold and superfamily recognition

For the remote homology and remote structural similarity detection, the goal is to detect a possibly new subclass within the class of interest. Another, simpler goal may be to match an unknown sequence to one of the known classes. We consider the direct multi-class protein fold and superfamily prediction on the SCOP $1.65$ datasets and evaluate our classifiers using a $10$-fold cross validation. We show the classification performance on the multi-class fold recognition and superfamily prediction tasks in Tables 4 and 5, respectively. The high error rates of 45% and 37% of the methods under the supervised setting highlight the difficulty of the problem. Tables 4 and 5, in both supervised and semi-supervised settings, indicate that the spatial kernels consistently outperform the mismatch and profile kernels on the multi-class fold recognition task.

## 5. DISCUSSION

The low computational complexity and running times are distinct characteristics of the spatial kernels, which we discuss in the following section. We also contrast the induced features of our kernel and those of traditional string kernels. Finally, we illustrate potential benefits of SSSK in view of the kernel-induced data manifolds on different string kernels.

**Table 1.** Comparison on Ding and Dubchak benchmark data set

| Method | Error | Top 5 Error | Balanced Error | Top 5 Balanced Error | Recall | Top 5 Recall | Precision | Top 5 Precision | F1 | Top5 F1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Supervised | | | | | | | | | | |
| SVM(D&D)† | - | - | 56.5 | - | - | - | - | - | - | - |
| Mismatch(5,1) | 51.17 | 22.72 | 53.22 | 28.86 | 46.78 | 71.14 | **90.52** | **95.25** | 61.68 | 81.45 |
| Double(1,5) | 44.13 | 23.50 | 46.19 | 23.92 | 53.81 | 76.18 | 61.90 | 79.85 | 57.57 | 77.97 |
| Triple (1,3) | **41.51** | **18.54** | **44.99** | **21.09** | **55.01** | **78.91** | 80.42 | 89.19 | **65.33** | **83.74** |
| Semi-supervised (Swiss-Prot) | | | | | | | | | | |
| Profile(5,7.5) | 36.03 | 16.19 | 37.78 | 18.46 | 62.22 | 81.54 | **88.39** | **94.53** | 73.03 | 87.56 |
| Double(1,5) | 27.42 | 16.45 | **21.81** | **13.26** | **76.57** | 86.74 | 77.73 | 86.07 | 77.15 | 86.4 |
| Triple(1,3) | **25.33** | **13.05** | 22.72 | 13.27 | 76.27 | **86.74** | 84.48 | 92.05 | **80.17** | **89.31** |
| Semi-supervised (Non-redundant data set) | | | | | | | | | | |
| Profile(5,7.5) | 31.85 | 15.14 | 32.17 | 16.73 | 67.83 | 83.27 | **89.49** | **94.9** | 77.16 | 88.71 |
| Double(1,5) | 28.72 | 14.99 | 24.74 | **11.6** | 75.26 | **88.4** | 76.02 | 86.86 | 75.63 | 87.62 |
| Triple(1,3) | **24.28** | **12.79** | **22.38** | 11.79 | **77.62** | 88.21 | 84.02 | 91.45 | **80.69** | **89.8** |
| Profile NR(Perceptron)‡ | - | - | 26.5 | - | - | - | - | - | - | - |

All measures are presented as percentages.
†: quoted from [12]
‡: quoted from [14]

**Table 2.** Multi-class **remote** fold prediction

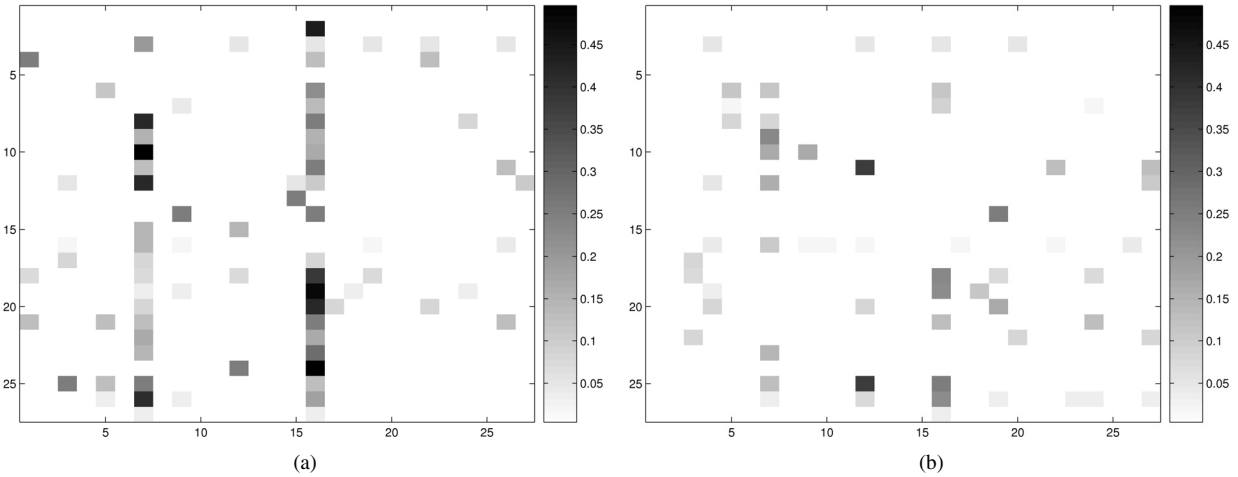
| Method | Error | Top 5 Error | Balanced Error | Top 5 Balanced Error | Recall | Top 5 Recall | Precision | Top 5 Precision | F1 | Top5 F1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Supervised | | | | | | | | | | |
| Mismatch(5,1) | 53.75 | 29.15 | 82.75 | 52.4 | 17.25 | 47.6 | 16.61 | 70 | 16.92 | 56.67 |
| Double (1,5) | 50.98 | 25.73 | **70.77** | **37.6** | 29.23 | **62.49** | 33.27 | 67.04 | **31.12** | **63.26** |
| Triple (1,3) | **48.7** | **25.08** | 73.04 | 44.05 | **26.96** | 55.95 | **35.28** | **70.46** | 30.57 | 62.37 |
| Semi-supervised (Swiss-Prot) | | | | | | | | | | |
| Profile(5,7.5) | 49.35 | 20.36 | 76.67 | 35.28 | 23.33 | 64.72 | 29.47 | 71.82 | 26.05 | 68.09 |
| Double (1,5) | **43** | 19.22 | **60.94** | 27.76 | **39.06** | 72.24 | **45.8** | 70.48 | **42.16** | 71.35 |
| Triple(1,3) | 43.97 | **15.64** | 62.29 | **26.77** | 37.71 | **73.23** | 40.74 | **77.04** | 39.17 | **75.08** |
| Semi-supervised (Non-redundant data set) | | | | | | | | | | |
| Profile(5,7.5) | 45.11 | 15.8 | 66.88 | 31.55 | 28.73 | 68.45 | 36.98 | **84.63** | 32.34 | 75.68 |
| Double (1,5) | **36.65** | 13.36 | **47.87** | 23.61 | **49.59** | 76.39 | 55.86 | 78.66 | **52.54** | 77.51 |
| Triple (1,3) | 37.13 | **10.91** | 49.34 | 20.07 | 47.19 | **79.93** | **55.91** | 82.78 | 51.18 | **81.33** |
| Profile NR (one-vs-rest)‡ | 46.3 | 14.5 | 62.8 | 23.5 | - | - | - | - | - | - |
| Profile NR (Adaptive codes)‡ | 37.0 | 11.4 | 49.9 | **15.5** | - | - | - | - | - | - |

‡: quoted from [14]

**Table 3.** Multi-class **remote** superfamily prediction

| Method | Error | Top 5 Error | Balanced Error | Top 5 Balanced Error | Recall | Top 5 Recall | Precision | Top 5 Precision | F1 | Top5 F1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Supervised | | | | | | | | | | |
| Mismatch(5,1) | 65.34 | 35.91 | 85.32 | 63.31 | 14.68 | 36.69 | 17.47 | 46.93 | 15.96 | 41.18 |
| Double (1,5) | 64.71 | 36.91 | **79.16** | **49.34** | **20.84** | **50.66** | **19.98** | 52.83 | **20.4** | 51.72 |
| Triple (1,3) | **58.85** | **29.55** | 80.34 | 52.05 | 19.66 | 47.95 | 18.93 | **61.14** | 19.29 | **53.75** |
| Semi-supervised (Swiss-Prot) | | | | | | | | | | |
| Profile(5,7.5) | 41.4 | 19.58 | 66.07 | 39.51 | 33.93 | 60.49 | 34.63 | 64.34 | 34.28 | 62.36 |
| Double (1,5) | 32.04 | 15.09 | 46.99 | **25.75** | 53.01 | **74.25** | 53.37 | 76.21 | 53.37 | **76.21** |
| Triple(1,3) | **28.8** | **14.34** | **46.98** | 27.14 | **53.07** | 72.86 | **55.84** | **76.28** | **54.39** | 74.53 |
| Semi-supervised (Non-redundant data set) | | | | | | | | | | |
| Profile(5,7.5) | 37.16 | 14.22 | 58.96 | 28.86 | 41.05 | 71.15 | 46.95 | 72.19 | 43.80 | 72.19 |
| Double(1,5) | 22.44 | 9.85 | 39.21 | 18.69 | **60.79** | 81.31 | 56.85 | 80.58 | 58.76 | 80.94 |
| Triple(1,3) | 22.94 | **8.6** | 39.86 | 17.29 | 60.14 | **82.72** | **61.62** | **83.76** | **60.87** | **83.24** |
| Profile NR (one-vs-rest)‡ | 27.1 | 10.5 | 44.5 | 19.7 | - | - | - | - | - | - |
| Profile NR (Adaptive codes)‡ | **21.7** | 10.3 | **32.0** | **15.3** | - | - | - | - | - | - |

‡: quoted from [14]



(a) (b)

**Fig. 2.** (a) Confusion matrix using the triple(1,3) feature set under the supervised setting. (b) Confusion matrix using the triple(1,3) feature set under the semi-supervised setting using the non-redundant (NR) unlabeled sequence database. In both figures, we remove the main diagonal terms to emphasize the differences in off-diagonal (error) terms.

**Table 4.** Multi-class protein fold recognition (10-fold cross-validation)

| Method | Error | Top 5 Error | Balanced Error | Top 5 Balanced Error | Recall | Top 5 Recall | Precision | Top 5 Precision | F1 | Top5 F1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Supervised | | | | | | | | | | |
| Mismatch(5,1) | 23.81 (1.25) | 6.86 (1.32) | 44.54 (1.94) | 16.29 (2.47) | 55.46 | 83.71 | 84.87 | 97.43 | 67.09 | 90.05 |
| Double (1,5) | 23.52 (1.38) | 9.09 (0.63) | 38.76 (3.18) | 14.46 (1.95) | 61.25 | 85.54 | 69.03 | 89.01 | 64.91 | 87.24 |
| Triple (1,3) | **19.79** (1.04) | **5.78** (0.61) | **37.85** (2.94) | **12.41** (2.75) | **62.15** | **87.59** | **86.67** | **97.46** | **72.39** | **92.26** |
| Semi-supervised (Swiss-Prot) | | | | | | | | | | |
| Profile(5,7.5) | 12.20 (1.49) | 4.09 (1.01) | 22.05 (3.92) | 9.38 (2.48) | 77.95 | 90.62 | **95.12** | **98.42** | 85.68 | 94.36 |
| Double (1,5) | 9.64 (1.61) | 4.14 (0.86) | 15.36 (2.72) | 6.53 (1.59) | 84.64 | **93.48** | 89.22 | 95.10 | 86.87 | 94.28 |
| Triple(1,3) | **8.60** (1.39) | **3.57** (0.56) | **15.21** (3.09) | **8.48** (1.69) | **84.79** | 91.52 | 94.52 | 97.71 | **89.39** | **94.51** |
| Semi-supervised (Non-redundant data set) | | | | | | | | | | |
| Profile (5,7.5) | 9.74 (1.32) | 2.72 (0.65) | 16.72 (2.11) | 6.52 (1.98) | 82.36 | 93.49 | 95.79 | **98.74** | 88.57 | 96.04 |
| Double(1,7) | 6.61 (0.86) | 2.87 (0.80) | 10.20 (1.66) | 4.07 (1.46) | 89.43 | 95.93 | 92.81 | 96.69 | 91.09 | 96.31 |
| Triple (1,3) | **5.78** (0.98) | **1.76** (0.45) | **10.06** (1.84) | **3.35** (1.19) | **89.61** | **96.65** | **96.29** | 98.73 | **92.83** | **97.68** |

*standard deviation for cross-validation error rates is indicated in parenthesis

**Table 5.** Multi-class superfamily prediction (10-fold cross-validation)

| Method | Error | Top 5 Error | Balanced Error | Top 5 Balanced Error | Recall | Top 5 Recall | Precision | Top 5 Precision | F1 | Top5 F1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Supervised | | | | | | | | | | |
| Mismatch(5,1) | 21.86 (1.28) | 9.42 (1.49) | 37.12 (1.70) | 18.77 (2.61) | 62.88 | 81.23 | 80.24 | 91.62 | 70.51 | 86.11 |
| Double (1,5) | 23.04 (1.67) | 9.98 (1.81) | 35.16 (2.09) | 15.07 (2.55) | 64.84 | 84.93 | 70.43 | 88.13 | 67.52 | 86.5 |
| Triple (1,3) | **18.09** (1.48) | **7.17** (1.12) | **31.31** (2.45) | **14.01** (2.15) | **68.69** | **85.99** | **82.21** | **93.37** | **74.84** | **89.53** |
| Semi-supervised (Swiss-Prot) | | | | | | | | | | |
| Profile(5,7.5) | 8.69 (1.86) | 4.12 (1.02) | 14.67 (3.68) | 8.33 (2.51) | 85.33 | 91.69 | 93.15 | 96.07 | 89.07 | 93.82 |
| Double (1,5) | 6.03 (1.13) | 3.05 (0.60) | **8.02** (2.18) | 4.04 (1.39) | **91.98** | 95.95 | 93.59 | 96.48 | 92.77 | 96.22 |
| Triple (1,3) | **5.43** (0.69) | **2.25** (0.39) | 8.31(1.39) | **4.03** (1.09) | 91.69 | **95.97** | **95.76** | **97.97** | **93.68** | **96.96** |
| Semi-supervised (Non-redundant dataset) | | | | | | | | | | |
| Profile(5,7.5) | 6.20 (1.36) | 2.59 (0.51) | 10.91 (2.60) | 5.27 (1.52) | 89.09 | 94.73 | 94.79 | 97.63 | 91.85 | 96.16 |
| Double (1,5) | 3.9 (0.95) | 1.85 (0.37) | **5.14** (1.45) | 2.38 (0.82) | **94.86** | 97.62 | 95.42 | 97.93 | 95.14 | 97.77 |
| Triple(1,3) | **3.39** (0.83) | **1.39** (0.59) | 5.24 (1.55) | **2.10** (1.19) | 94.76 | **97.89** | **97.00** | **98.89** | **95.86** | **98.39** |

*standard deviation for cross-validation error rates is indicated in parenthesis

## 5.1. Complexity and running time analysis

Table 6 shows the computational complexity of various string kernels. Both mismatch and profile kernels have higher complexity compared to the spatial 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.

**Table 6.** Complexity of computations

| Method | Time complexity |
|---|---|
| Supervised methods | |
| Triple kernel | $O(d^2nN + d^2\|\Sigma\|^3N^2)$ |
| Double kernel | $O(dnN + d\|\Sigma\|^2N^2)$ |
| Mismatch | $O(k^{m+1}\|\Sigma\|^m nN + \|\Sigma\|^k N^2)$ |
| Semi-supervised methods | |
| Triple kernel | $O(d^2HnN + d^2\|\Sigma\|^3N^2)$ |
| Double kernel | $O(dHnN + d\|\Sigma\|^2N^2)$ |
| Mismatch | $O(k^{m+1}\|\Sigma\|^m HnN + \|\Sigma^k\|N^2)$ |
| Profile kernel | $O(kM_\sigma nN + \|\Sigma\|^k N^2)$ |

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 the mismatch kernel parameters
($k = 5, 6$ and $m = 1, 2$ in most cases)
$M_\sigma$ is the profile neighborhood size,
$k^m|\Sigma|^m \leq M_\sigma \leq |\Sigma^k|$
$d$ is the distance parameter for the spatial kernel.

This complexity difference leads to order-of-magnitude improvements in the running times, as shown in Table 7, of the spatial kernels over the standard string kernels (mismatch and profile).

The running time measurements are obtained on a single 2.8GHz CPU machine. The code used for evaluation of the competing methods has been highly optimized to perform on par or better than the published spectrum/mismatch kernel code. We also used an existing implementation of the profile kernel provided by Kuang *et al*. in [16].

The neighborhood mismatch kernel becomes substantially more expensive to compute for large datasets, as indicated in [16, 17]. Table 8 summarizes the size of the unlabeled datasets and the mean, median, and maximum number of neighbors used for kernel smoothing.

**Table 7.** Comparison of the running time (kernel matrix computations)

| Method | Running time (s) |
|---|---|
| Supervised (fold data set) | |
| Mismatch | 396 |
| Double | 22 |
| Triple | 52 |
| Semi-supervised (fold data set) | |
| Profile | 1633 |
| Double | 165 |
| Triple | 701 |
| Mismatch | - |

**Table 8.** Data set characteristics

| Data set | # Seq. | # Neighbors (mean/median/max) | |
|---|---|---|---|
| | | Superfamily | Fold |
| Swiss-Prot | 101602 | 42/30/244 | 30/17/174 |
| NR | 534936 | 79/58/356 | 52/28/360 |

## 5.2. Comparison of the features induced by different string kernels

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.

To further illustrate the differences and the trade-off between different features, we consider an example of modeling a slightly diverged region using the mismatch and spatial kernel similarity measures. We first compare the spectrum-induced features with our proposed spatial features, extracted from a string S = 'HKYNQLIM', in Figure 3(a). The symbol 'x' in the mismatch-$(k,m)$ fea-

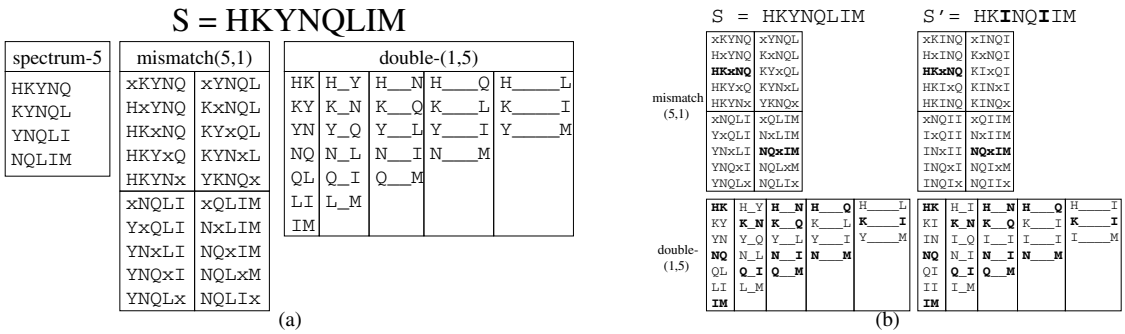



**Fig. 3.** (a) Comparison of features extracted by the spectrum-like and spatial kernels. In the mismatch features, each symbol 'x' represent an arbitrary symbol in the alphabet set. As a result, each feature basis corresponds to $|\Sigma|$ features. (b) Differences in handling substitutions by the mismatch and spatial features. We represent all common features between the original and the mutated strings, $S$ and $S'$, with bold fonts.

tures corresponds to an arbitrary symbol in $\Sigma$. As a result, each mismatch feature basis in the figure 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. In contrast, the spatial features scans the strings at different resolutions and the number of features induced is small. Furthermore, as shown in Figure 3(b), for two slightly diverged strings, $S$ and $S'$, very few common features (bold font) are observed in the case of the mismatch kernel, leading to low similarity scores. On the other hand, the larger subset of common features indicates that the spatial kernels are still able to capture the similarity between the two sequences. For the mismatch features to handle such a slightly diverged region, one needs to increase the number of allowed mismatches $m$, at the expense of an increase in computational effort.

### 5.3. Kernel-induced data manifolds

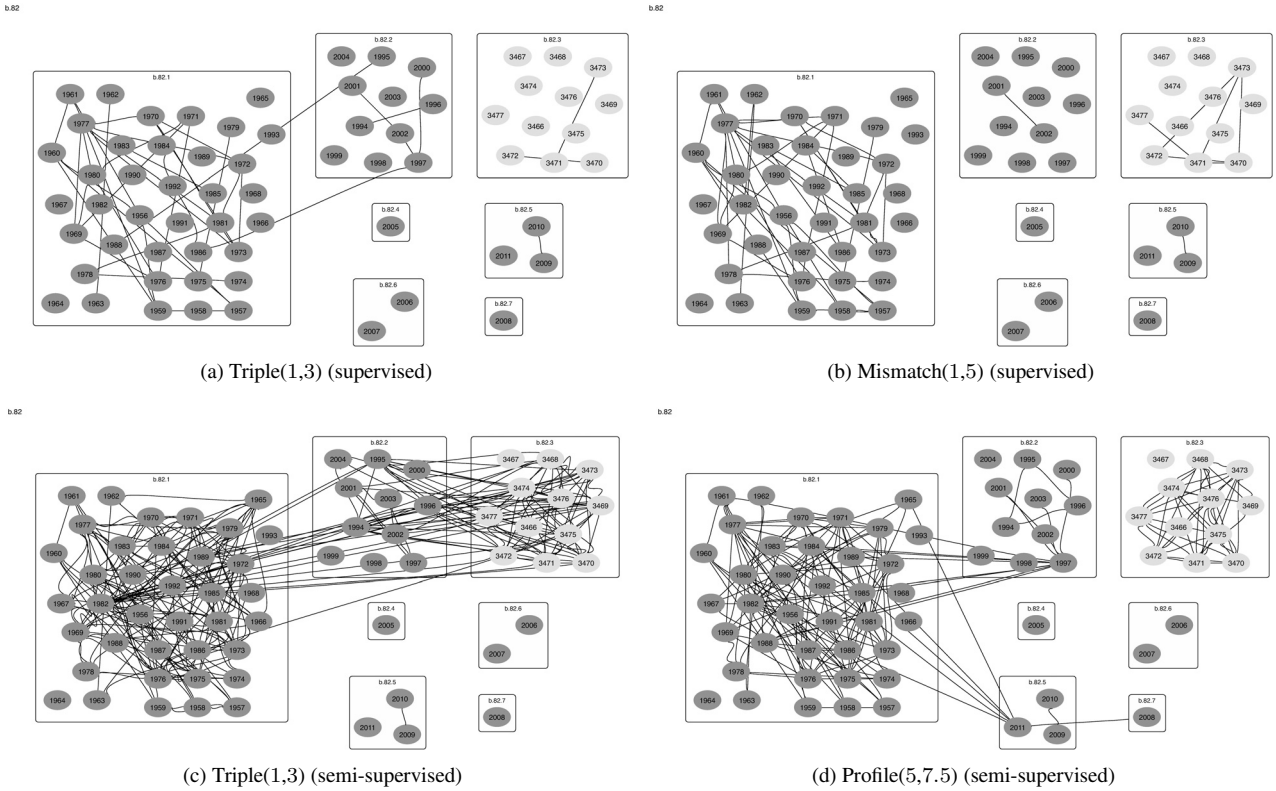
To shed more light on the causes of improved performance of SSSK, we compare the data manifolds induced by different kernels in both supervised and semi-supervised settings[d]. We show the kernel-induced manifolds for the *double-stranded beta-helix* (b.82) fold in Figures 4(a) and 4(b) for the supervised setting and in Figures 4(c) and 4(d) for the semi-supervised setting. The fold contains proteins carrying out a diverse range of functions and participating in many biological processes. Each node in the graph represents a sequence, with darker nodes corresponding to the training sequences and lighter nodes corresponding to the test sequences (superfamily b.82.3). Each cluster (box) represents a superfamily in the fold. We normalize the kernel as discussed in Section 4.2 to remove the dependencies between kernel values and sequence length. We draw an edge between two sequences, $X$ and $Y$, if $K(X,Y) > \delta$, ($\delta$ is chosen so that the total number of nodes outside the fold having similarity values above the threshold with nodes inside the fold is small).

In the supervised setting (Figures 4(a) and 4(b)), we observe a slightly more connected graph induced by the triple kernel compared to the mismatch kernel. Similarly, in the semi-supervised setting (Figures 4(c) and 4(d)) with the non-redundant set, compared to the profile kernel the triple kernel induces a data manifold with stronger connectivity, suggesting better sensitivity of the spatial kernels (on this fold, the triple and profile kernels achieve $91.67\%$ and $83.33\%$ recall rate, both with $100\%$ precision). This, in turn, leads to lower error rates of classifiers with the SSSK.

## 6. CONCLUSIONS

We present a new family of sparse spatial sample kernels that demonstrate state-of-the-art performance for multi-class protein fold and remote homology prediction problems. The key component of the method is the spatially-constrained sample kernel for efficient sequence comparison which, combined with kernel smoothing using unlabeled data, leads to efficient and accurate semi-supervised protein remote homology detection and remote fold recognition. We show that our methods can work with large, unlabeled databases of protein sequences, taking full advantage of all available data and

[d] We use the *fdp* package in *Graphviz* `http://graphviz.org` for visualization.



(a) Triple(1,3) (supervised)

(b) Mismatch(1,5) (supervised)

(c) Triple(1,3) (semi-supervised)

(d) Profile(5,7.5) (semi-supervised)

**Fig. 4.** Kernel-induced data manifold for fold $b.82$, with 7 superfamilies, under the supervised and semi-supervised settings. The darker and lighter nodes are the training and testing sequences, respectively. The numbers in the nodes index the sequences in the database.

substantially improving the classification accuracy. This opens the possibility for the proposed methodology to be readily applied to other challenging problems in biological sequence analysis.

## References


1. Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucl. Acids Res.*, 33(suppl-1):D34–38, 2005.
2. Amos Bairoch, Rolf Apweiler, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Darren A. Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L. Yeh. The Universal Protein Resource (UniProt). *Nucl. Acids Res.*, 33(suppl-1):D154–159, 2005.
3. L. Lo Conte, B. Ailey, T.J. Hubbard, S.E. Brenner, A.G. Murzin, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Res.*, 28:257–259, 2000.
4. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.
5. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, pages 403–410, 1990.
6. W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.
7. M. Gribskov, A.D. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84:4355–4358, 1987.
8. SR Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.
9. Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. In *Journal of Computational Biology*, volume 7, pages 95–114, 2000.
10. Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In *NIPS*, pages 1417–1424, 2002.
11. Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina 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. http://www.cs.columbia.edu/compbio/profile-kernel.

12. Chris H.Q. Ding and Inna Dubchak. Multi-class protein fold recognition using support vector machines an d neural networks . *Bioinformatics*, 17(4):349–358, 2001.
13. Eugene Ie, Jason Weston, William Stafford Noble, and Christina Leslie. Multi-class protein fold recognition using adaptive codes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 329–336, New York, NY, USA, 2005. ACM.
14. Iain Melvin, Eugene Ie, Jason Weston, William Stafford Noble, and Christina Leslie. Multi-class protein classification using adaptive codes. *J. Mach. Learn. Res.*, 8:1557–1581, 2007.
15. Jianlin Cheng and Pierre Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, June 2006.
16. Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *J Bioinform Comput Biol*, 3(3):527–550, June 2005.
17. Jason Weston, Christina Leslie, Eugene Ie, Dengyong Zhou, Andre Elisseeff, and William Stafford Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.
18. J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 4 1999.
19. Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.
20. Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.
21. Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
22. B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res*, 31:365–370, 2003.
23. Theodoros Damoulas and Mark A. Girolami. Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.