Generalized Similarity Kernels for Efficient Sequence Classification

Pavel P. Kuksa
NEC Laboratories America, Inc.
Princeton, NJ 08540
pkuksa@nec-labs.com

Imdadullah Khan
Department of Computer Science
Gulf University for Science and Technology
Kuwait

Vladimir Pavlovic
Department of Computer Science
Rutgers University
Piscataway, NJ 08854

October 14, 2011

Abstract

String kernel-based machine learning methods have yielded great success in practical tasks of structured/sequential data analysis. They often exhibit state-of-the-art performance on tasks such as document topic elucidation, music genre classificatioin, protein superfamily and fold prediction. However, typical string kernel methods rely on *symbolic Hamming-distance* based matching which may not necessarily reflect the underlying (e.g., physical) similarity between sequence fragments. In this work we propose a novel computational framework that uses *general similarity metrics* $\mathcal{S}(\cdot,\cdot)$ and distance-preserving embeddings with string kernels to improve sequence classification. In particular, we consider two approaches that allow one either to incorporate non-Hamming similarity $\mathcal{S}(\cdot,\cdot)$ into similarity evaluation by matching only the features that are similar according to $\mathcal{S}(\cdot,\cdot)$ or to retain actual (approximate) similarity/distance scores in similarity evaluation. An embedding step, a distance-preserving bit-string mapping, is used to effectively capture similarity between otherwise symbolically different sequence elements. We show that it is possible to retain computational efficiency of string kernels while using this more "precise" measure of similarity. We then demonstrate that on a number of sequence classification tasks such as music, and biological sequence classification, the new method can substantially improve upon state-of-the-art string kernel baselines.

Keywords: string kernels, classification, sequence analysis

1 Introduction

Analysis of large scale sequential data has become an important task in machine learning and data mining, inspired by applications such as biological sequence analysis, text and audio mining. Classification of string data, sequences of discrete symbols, has attracted particular interest and has led to a number of new algorithms [3, 8, 14, 18]. These algorithms often exhibit state-of-the-art performance on tasks such as protein superfamily and fold prediction, music genre classification and document topic elucidation.

A family of state-of-the-art approaches to scoring similarity between pairs of sequences relies on fixed length, substring spectral representations and the notion of mismatch kernels, c.f. [8, 14]. There, a sequence is represented as the spectra (counts) of all short substrings (k-mers) contained within a sequence. The similarity score is established by exact or approximate matches of k-mers. Initial work, e.g., [14, 17], has demonstrated that this similarity can be computed using trie-based approaches in $O(k^{m+1}|\Sigma|^m(|X| + |Y|))$, for strings X and Y with symbols from alphabet Σ and up to m mismatches. More recently, [10] introduced linear time algorithms with alphabet-independent complexity applicable to computation of a large class of existing string kernels.

However, typical spectral models (e.g., mismatch/spectrum kernels, gapped and wildcard kernels) rely on *symbolic Hamming-distance* based matching of k-mers. For example, given a sequence X over alphabet Σ the *spectrum-k* kernel [13] and the *mismatch-*(k,m)

kernel [14] induce $|\Sigma|^k$-dimensional representation

$$\Phi_{k,m}(X) = \left(\sum_{\alpha \in X} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k} \tag{1.1}$$

where $I_m(\alpha, \gamma) = 1$ if $\alpha \in N_{k,m}(\gamma)$, and $N_{k,m}(\gamma)$ is the *mutational neighborhood* of γ, the set of all k-mers that differ from γ by at most m mismatches (note that the indicator function $I_{k,m}(\cdot, \cdot)$ only uses *symbolic Hamming distance* between k-mers.

This Hamming matching may not necessarily reflect the underlying (e.g., physical) similarity between sequence fragments. For example, in protein sequence analysis different pairs of symbols (amino acids) induce different similarity levels, a consequence of particular physical or chemical properties. Similarly, matching of n-grams of words should reflect semantic similarity and not only simple character-level differences. Unfortunately, traditional string kernel approaches do not readily extend to the case of general non-Hamming similarity metrics without introducing a high computational cost (e.g., requiring quadratic or exponential running time). Another difficulty with using $\mathcal{S}(\cdot, \cdot)$ instead of $I_m(\cdot, \cdot)$ is that the resulting similarity measure may not necessarily be a proper kernel.

In this work we propose novel *linear-time* algorithms for modeling sequences under inexact matching framework with general similarity metrics $\mathcal{S}(\cdot, \cdot)$ that exhibit improved performance on a variety of distinct classification tasks. In particular, we present a novel computational framework that (1) accepts a general (non-Hamming) similarity metric $\mathcal{S}(\cdot, \cdot)$, (2) approximately preserves $\mathcal{S}(\cdot, \cdot)$ through symbolic binary embedding of k-mers, and (3) uses Hamming-type matching on the embedded representation with string kernels (linear-time algorithms) to improve sequence classification. The adaptive embedding step, which learns the similarity/distance-preserving embeddings, is used to effectively capture similarity interrelationships between otherwise symbolically dissimilar sequence elements/features.

We demonstrate benefits of our algorithms on many challenging sequence classification problems, both *discrete*- and *continuous*-valued, such as detecting homology (evolutionary similarity) of remotely related proteins, recognizing protein fold, and performing classification of music samples. The algorithms display improved classification performance and run substantially faster than existing methods.

2 Related Work

Over the past decade, various methods have been proposed to solve the string classification problem, including *generative*, such as HMMs, or *discriminative* approaches. Among the discriminative approaches, in many sequence analysis tasks, string kernel-based [19] methods provide some of the most accurate results [8, 14, 18, 12, 4].

The key idea of basic string kernel methods is to apply a mapping $\Phi(\cdot)$ to map sequences of variable length into a fixed-dimensional vector space. In this space a standard classifier such as a support vector machine (SVM) [19] can then be applied. As SVMs require only inner products between examples in the feature space, rather than the feature vectors themselves, one can define a *string kernel* which computes the inner product in the feature space without explicitly computing the feature vectors:

$$K(X, Y) = \langle \Phi(X), \Phi(Y) \rangle, \tag{2.2}$$

where $X, Y \in D$, D is the set of all sequences composed of elements which take on a finite set of possible values from the alphabet Σ.

Sequence matching is frequently based on co-occurrence of exact sub-patterns (k-mers, features), as in spectrum kernels [13] or substring kernels [20]. Inexact comparison in this framework is typically achieved using different families of mismatch [14] or profile [8] kernels. Both spectrum-k and mismatch(k,m) kernel directly extract string features from the observed sequence, X. On the other hand, the profile kernel, proposed by Kuang et al. in [8], builds a profile [6] P_X and uses a similar $|\Sigma|^k$-dimensional representation, now derived from P_X. Constructing the profile for each sequence may not be practical in some application domains, since the size of the profile is dependent on the size of the alphabet set. While for bio-sequences $|\Sigma| = 4$ or 20, for music or text classification $|\Sigma|$ can potentially be very large, on the order of tens of thousands of symbols.

Some of the most efficient available trie-based algorithms [14, 17] for mismatch kernels have a strong dependency on the size of alphabet set Σ and the number of allowed mismatches, both of which need to be restricted in practice to control the complexity of the algorithm. Under the trie-based framework, the list of k-mers extracted from given input strings is traversed in a depth-first search with branches corresponding to all possible $\sigma \in \Sigma$. Each leaf node at depth k corresponds to a particular k-mer feature (either exact or inexact instance of the observed exact string features) and contains a list of matching features from each string. The kernel matrix is updated at leaf nodes with corresponding matching feature counts. The complexity of the trie-based algorithm for mismatch kernel computation for two strings X and Y is $O(k^{m+1}|\Sigma|^m(|X|+|Y|))$ [14].

The algorithm complexity depends on the size of Σ since during a trie traversal, possible substitutions are drawn from Σ explicitly; consequently, to control the complexity of the algorithm we need to restrict the number of allowed mismatches m, as well as $|\Sigma|$.

We note that most of existing k-mer string kernels (e.g., mismatch/spectrum kernels, gapped and wildcard kernels, c.f. [12]) essentially use only *symbolic* Hamming-distance based matching, which may not necessarily reflect underlying similarity/dissimilarity between k-mers. For a large class of k-mer string kernels, which include mismatch/spectrum, gapped, wildcard kernels, the matching function, $I(\alpha, \beta)$, of two k-mers α and β is *independent* of the actual k-mers being matched and depends only on the Hamming distance [10]. As a result, related k-mers may not be matched because their symbolic dissimilarity exceeds the maximum allowed Hamming distance. This presents a limitation as in many cases similarity relationships are not entirely based on symbolic similarity, e.g., as in matching word n-grams or amino-acid sequences, where, for instance, words may be semantically related or amino-acids could share structural or physical properties not reflected on a symbolic level. Recent work in [10] have introduced linear time algorithms with *alphabet-independent* complexity for Hamming-distance based matching. This enables efficient computation of a wide class of existing string kernels for datasets with large $|\Sigma|$. However, above approaches do not readily extend to the case of a general (non-Hamming) similarity metrics (e.g., BLOSUM-based scoring functions in biological sequence analysis, or measures of semantic relatedness between words, etc.) without introducing a high computational cost (e.g., requiring quadratic or exponential running time as in, for instance, BLOSUM-based substitution kernels). In this work, we aim to extend the works presented in [10, 12] to the case of general (non-Hamming) similarity metrics and introduce efficient linear-time generalized string kernel algorithms (Sections 4, 5). We also show empirically that using these generalized similarity kernels provides effective improvements in practice for a number of challenging classification problems (Section 6).

3 Spectrum/Mismatch and General Similarity Kernels

In this section we will first discuss sequence matching with spectrum/mismatch kernels and then introduce general similarity string kernels as their generalization.

Given a sequence X over the alphabet Σ the *spectrum-k* kernel [13] and the *mismatch(k,m)* kernel [14] induce the following $|\Sigma|^k$-dimensional representation for the sequence X considered as a set of k-mers:

$$\Phi_{k,m}(\gamma|X) = \left(\sum_{\alpha \in X} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k} \tag{3.3}$$

where $I_m(\alpha, \gamma) = 1$ if $\alpha \in N_{k,m}(\gamma)$, and $N_{k,m}(\gamma)$ is the *mutational neighborhood*, the set of all k-mers that differ from γ by at most m mismatches. Note that, by definition, for spectrum-k kernels, $m = 0$. Effectively, these are the *bag-of-substrings* representations, with either exact (spectrum) or approximate/smoothed (mismatch) counts of substrings present in the sequence X.

The spectrum/mismatch kernel is then defined as

$$\begin{aligned} K(X, Y|k, m) &= \sum_{\gamma \in \Sigma^k} \Phi_{k,m}(\gamma|X) \Phi_{k,m}(\gamma|Y) \\ &= \sum_{\alpha \in X} \sum_{\beta \in Y} \sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma) \end{aligned} \tag{3.4}$$

One interpretation of this kernel is that of cumulative pairwise comparison of all k-long substrings α and β contained in sequences X and Y, respectively. In the case of mismatch kernels the level of similarity of each pair of substrings (α, β) is based on the number of identical substrings their mutational neighborhoods $N_{k,m}(\alpha)$ and $N_{k,m}(\beta)$ give rise to, $\sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma)$. For the spectrum kernel, this similarity is simply the exact matching of α and β.

One can generalize this and allow an arbitrary metric similarity function $\mathcal{S}(\alpha, \beta)$ to replace the Hamming similarity $\sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma) \rightarrow \mathcal{S}(\alpha, \beta)$ and obtain general similarity kernel:

$$K(X, Y|k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta). \tag{3.5}$$

Such similarity function can go significantly beyond the relatively simple substring mismatch (substitution) models mentioned above. However, a drawback of this general representation over the simple Hamming-similarity based mismatch model is, of course, that the complexity of comparing two sequences in general becomes *quadratic* in their lengths, $O(|X| \cdot |Y|)$. On the other hand, mismatch type of representations can be efficiently evaluated in $O(c_{k,m}(|X| + |Y|))$ time [10]. We also note that directly using Eq. 3.5 will not, in general, result in a proper kernel $K(X, Y|k, \mathcal{S})$ for arbitrary similarity metric $\mathcal{S}(\cdot, \cdot)$.

Our goal here is to approach the above process "in reverse": start with a general similarity metric $\mathcal{S}$ and replace it with an approximate, but computationally efficient, Hamming-type computation. In the following we consider two approaches for incorporating $S(\cdot, \cdot)$ into sequence matching that preserve *linear time* complexity

of kernel computation and result in a proper kernel function $K(X, Y|\mathcal{S})$. The first approach (Sec. 4) uses $S(\cdot,\cdot)$ indirectly by clustering the original feature set into groups of similar features. These groups are then used for matching, with two features matched only if they belong to same group (i.e. are similar according to $S(\cdot,\cdot)$). In the second approach (Section 5), based on recent work in similarity-based hashing in [21], an approximation of the actual values of $S(\cdot,\cdot)$ is used for the similarity score computation. The two approaches allow one either to incorporate non-Hamming similarity $S(\cdot,\cdot)$ into similarity evaluation by matching only the features that are similar according to $S(\cdot,\cdot)$ or to retain actual (approximate) similarity/distance scores in similarity evaluation.

4 Abstraction-based kernels

In this section we propose a generalization of the string kernels that extends the typically employed symbolic Hamming (0-1) distance and incorporates general similarity metrics to refine matching of otherwise symbolically different sequence fragments.

Standard string kernels typically use input sequences directly, i.e., they are defined over the input alphabet $|\Sigma|$. Here we assume that the alphabet Σ is supplemented by a set of features $\mathcal{F}$. For instance, in the case of proteins the 20 amino acids can be supplemented by ordinal features that describe their physical or chemical properties. Note that the feature set need not be specified explicitly, it is sufficient to define the similarity metric $\mathcal{S}(\alpha,\beta)$ that reflect symbol similarity in this feature space.

To incorporate a similarity metric $\mathcal{S}(\alpha,\beta)$ into string kernel framework, one possibility is to introduce an indicator (matching) function $I_{\mathcal{S}}(\alpha,\beta)$ that models the given scoring metric by defining a partition over the feature set (e.g., the set of all k-mers), which groups similar (according to $S(\cdot,\cdot)$) features together. The indicator function $I_{\mathcal{S}}(\cdot,\cdot)$ would only match two k-mers α and β if they belong to the same class, i.e. are similar according to $\mathcal{S}(\alpha,\beta)$. Partitioning of the feature set $\mathcal{F}$ essentially corresponds to *clustering* over the feature set that generates more *abstract* entities (features). Similar features that are grouped together can, for instance, indicate semantic closeness of words in text, or similarity in terms of physical or structural properties for amino-acids in biological sequences, etc. Grouping of features into similarity classes might also allow for better generalization over learned sequence patterns and improve sequence classification performance.

For a given feature set $\mathcal{F}$ (e.g., set of k-mers over alphabet Σ or the alphabet Σ itself), with similarity relationships between features in $\mathcal{F}$ encoded as a partition over $\mathcal{F}$, i.e. in n disjoint cluster sets $C_1,\ldots,C_n$, $\bigcup C_i = \mathcal{F}$. Let $L : \mathcal{F} \rightarrow \{C_1,\ldots,C_n\}$ be the cluster (partition) indicator function. The abstraction/clustering-based kernel under $L()$ can then be defined as

$$K_L(X,Y) = \sum_{\gamma\in\mathcal{F}} \sum_{\alpha\in X} I_{\mathcal{S}}(\alpha,\gamma) \sum_{\beta\in Y} I_{\mathcal{S}}(\beta,\gamma)$$

$$= \sum_{\alpha\in X} \sum_{\beta\in Y} \sum_{\gamma\in\mathcal{F}} I_{\mathcal{S}}(\alpha,\gamma) I_{\mathcal{S}}(\beta,\gamma) \tag{4.6}$$

where the indicator $I_{\mathcal{S}}()$ reflects grouping over features induced by similarity measure $\mathcal{S}(\cdot,\cdot)$, $I_{\mathcal{S}}(\alpha,\beta) = 1$, if $L(\alpha) = L(\beta)$ and 0 otherwise. We note that the two k-mers α and β will be matched only if they belong to the same group, $L(\alpha) = L(\beta)$, i.e. α and β are similar according to $S(.,.)$. This allows one to incorporate non-Hamming similarity relationships between k-mers into the similarity/kernel evaluation. Similarity function $S(\alpha,\beta)$ between k-mers can be the Euclidean distance between the corresponding real-valued feature vectors, or may reflect similarity with respect to the physical/structural properties. Clustering of these feature vectors would then group the original k-mer features into groups of k-mers similar according to $S(\cdot,\cdot)$. We note that this essentially corresponds to a kernel defined over clustered representation $X_C = L(X)$ of the original sequence X, where X_C is obtained from X by mapping original sequence features into corresponding similarity clusters $C_1,...C_n$. This is illustrated in Figure 1 for the case of feature space $\mathcal{F}$ taken as an alphabet set. We note that the cluster indicator function $L(\cdot)$ can be obtained by clustering of the feature vectors corresponding to the original features in $\mathcal{F}$, e.g. rows of the BLOSUM substitution matrix can be used to cluster individual aminoacids or aminoacid k-mers according to the Euclidean distance/similarity $S(\cdot,\cdot)$ between feature vectors. Alternatively, the partition over $\mathcal{F}$ can be defined with respect to similarity $S(\cdot,\cdot)$ as given by the physical/chemical properties of aminoacids (e.g., $L()$ could be defined as a threshold function based on the hydropathy index). We show in experiments that using abstraction/clustering kernel improves over using original sequences alone.

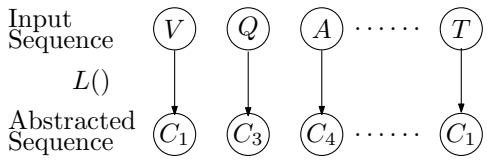

Figure 1: Abstraction-based kernel ($\mathcal{F}$ is the same as Σ). Both the input sequence X and the abstracted sequence $L(X)$ can be used simultaneously.

5 Similarity/distance-preserving Symbolic Embedding Kernels

While the abstraction/clustering-based kernel described in previous sections essentially uses a fixed partition over features, one may wish to retain the actual distance/score as given by $\mathcal{S}(\cdot,\cdot)$. For a given scoring function $\mathcal{S}(\alpha,\beta)$ that reflects the semantic similarity between sequence elements, a string kernel can then be defined as in Eq.3.5.

That definition, however, results in quadratic computation $O(|X|\cdot|Y|)$ complexity. We will now define a computationally more efficient string kernel based on symbolic sequence embeddings for which Hamming (0-1) similarity accurately approximates $\mathcal{S}(\cdot,\cdot)$. This will allow us to develop a *linear* time algorithm for computing string kernels with general similarity metrics.

The main idea of our approach is to (1) learn a symbolic embedding $E(\cdot)$ for which the Hamming similarity could approximate $\mathcal{S}(\cdot,\cdot)$, (2) use Hamming-type matching (linear time) algorithms over the symbolic embedding to efficiently evaluate the similarity kernel $K(X,Y|\mathcal{S})$.

To learn a symbolic embedding that approximates $\mathcal{S}(\cdot,\cdot)$, a *similarity hashing-based* approach, as in, for instance [21], could be applied to sequence features (e.g., k-mers or individual sequence elements) to obtain a *binary* Hamming-space embedded sequence

$$E(X) = E(x_1), ..., E(x_n)$$

for a given input sequence $X = x_1, ..., x_n$ of R-dimensional feature vectors, where $E(x_i) = e_1^i e_2^i \dots e_B^i$ is a symbolic Hamming embedding for item x_i in X, with $|E(x_i)| = B$, the number of bits in a resulting binary embedding of x_i. Under this embedding, the Hamming similarity, $h_{\alpha,\beta}$, between two feature embeddings $E(\alpha)$ and $E(\beta)$ is proportional to the original similarity score $\mathcal{S}(\alpha,\beta)$. The kernel evaluation $K(X,Y|\mathcal{S})$ (Eq. 3.5) then reduces to computing Hamming-type string kernels $K'(E(X),E(Y))$ (e.g., spectrum or mismatch) over embedded sequences $E(X)$, $E(Y)$. For a typical length of a Hamming embedding (e.g., B=64), computing the k-mer mismatch kernel may be difficult due to very large equivalent values of $k' = k\cdot B$, $m' = m\cdot B$ since the complexity of the best string kernel algorithms depends as $O(k^m)$ on the values of k, and m. To solve this problem, in what follows we develop a very effective approximation of the mismatch kernel that allows efficient inexact matching between embedded sequences.

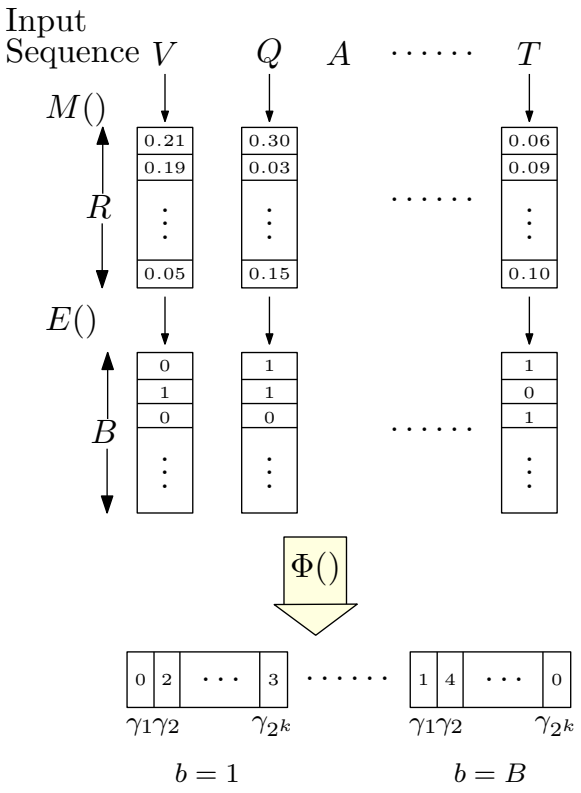

Figure 2: Symbolic embedding for similarity evaluation. For discrete inputs, discrete-to-real mapping $M(\cdot)$ is used to obtain sequence X of R-dim. feature vectors that are then embedded into B-bit vectors to obtain $E(X)$. Feature map $\Phi(E(X))$ yields $B\times 2^k$ representation of the original sequence that is used for similarity evaluation. For real-valued inputs, the M-step is not used.

5.1 Approximate Mismatch Kernel for Symbolic Hamming Embeddings

Instead of solving a (kB,mB)-mismatch problem, we will show that a collection of B simpler (k,m)-mismatch problems could be solved. This will allow a very efficient inexact matching for even high-dimensional embeddings, E.

For a given sequence set $\mathcal{D}$, where each $X\in\mathcal{D}$ is a sequence of R-dimensional feature vector, let $\mathcal{E} = \{E(X) : X\in\mathcal{D}\}$. In the following, we describe a kernel measure, $K_k(E(X),E(Y))$, for computing similarity between two Hamming embeddings $E(X)$, $E(Y)$ of the original sequences X and Y. To do this we note that our binary embedding of each sequence X results in this sequence being represented as a $B\times|X|$ binary matrix, with each column l being the binary embedding of the l-th element x_l. The rationale for the choice of K_k below will be discussed in detail in Sec. 5.3.

To compare two embeddings $E(X)$ and $E(Y)$ (i.e., two binary matrices of size $B\times|X|$, $B\times|Y|$), we define a feature map Φ_k, from $\mathcal{E}$ into a $B\times 2^k$ dimensional

feature space $\Phi_k : \mathcal{E} \rightarrow \mathbb{R}^{B \times 2^k}$:

$$\Phi_k(E(X)) = \left(\phi^b_\gamma\left(E(X)\right)\right)_{\gamma \in \{0,1\}^k, 1 \leq b \leq B} \tag{5.7}$$

where $\phi^b_\gamma(E(X)) = \sum_{\alpha \in E(X)} I(\alpha[b], \gamma)$ is the number of times γ occurs in the b^{th} row of $E(X)$ matrix. The kernel is then the dot-product between corresponding feature maps:

$$K_k(E(X), E(Y)) = \langle \Phi_k(E(X)), \Phi_k(E(Y)) \rangle. \tag{5.8}$$

In Figure 2 we illustrate similarity evaluation using similarity-preserving symbolic embedding. Discrete inputs are first mapped into a sequence of R-dimensional feature vectors ($M()$-step, e.g. by replacing discrete amino acid characters with rows of BLOSUM substitution matrix). Those feature vectors are then mapped using $E()$ into binary vectors to obtain a $B \times |X|$ matrix representation of the original sequence. These binary matrices are then compared using $K_k(\cdot, \cdot)$.

5.2 Efficient Kernel Computation We first reduce the kernel computation (Eq.5.8) to the following form.

$$K_k(E(X), E(Y)) = \langle \Phi_k(E(X)), \Phi_k(E(Y)) \rangle \tag{5.9}$$

$$= \sum_{b=1}^{B} \sum_{\gamma \in \{0,1\}^k} \left(\sum_{\alpha \in E(X)} I(\alpha[b], \gamma) \sum_{\beta \in E(Y)} I(\beta[b], \gamma) \right)$$

$$= \sum_{b=1}^{B} \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} I(\alpha[b], \beta[b]) \tag{5.10}$$

$$= \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} \sum_{b=1}^{B} I(\alpha[b], \beta[b]) \tag{5.11}$$

We note that the kernel value above corresponds to the cumulative count of the number of common rows in α and β. I.e., for each pair of $B \times k$ submatrices (α,β) the kernel value is incremented by the number of rows that are identical. Direct use of the Eq. 5.11 would result in *quadratic* time kernel evaluation. A more efficient approach is to note that the number of pairs of $B \times k$ submatrices (α,β) from $E(X)$ and $E(Y)$ that have the same bth row can be computed by running the exact spectrum-k kernel B times, i.e. for each row $b = 1...B$ of $E(X)$ and $E(Y)$. This gives a *linear time* $O(Bnk)$ algorithm for comparing the embedded representations $E(X)$ and $E(Y)$. This is an improvement over directly using $O(c_{kB,mB}nk)$ mismatch kernel algorithm for inexact matching as the above linear-time row-based comparison between two $B \times k$ submatrices in effect corresponds to *inexact* matching of these matrices. We also note that the computational complexity $O(Bkn)$ of evaluating $K(X, Y|\mathcal{S})$ compares well with computational complexity of typical Hamming-based string kernels [12] (e.g., $O(C(g,k)kn)$ of the gapped kernel, or $O(k^{m+1}|\Sigma|^m n)$ of the mismatch kernel).

The described algorithm has linear complexity compared to quadratic complexity of evaluating general kernel of the form as in Equation 3.5. Furthermore, the approach we propose is substantially different from *trie-based* approaches described, for example, in [12] for computing e.g. substitution kernels that use biologically meaningful similarity measure based on probabilistic substitution models.

In summary, the proposed algorithms exhibit the following advantages:

- Enhances original *discrete/symbolic* or *real-valued* representation by re-representing sequence data using different sequence alphabet that captures inter-relationships between individual sequence elements or features (e.g., k-mers).
- Allow efficient (linear time) matching (Section 5.1)
- Incorporate similarity/distance $\mathcal{S}(\cdot, \cdot)$ between features or sequence elements into matching which otherwise is typically limited to symbolic matching (symbolic hamming distance)
- Refine matching of otherwise symbolically different sequence elements/features (e.g., different amino-acids in proteins, music samples, as shown in the experiments in Section 6)

5.3 Relationship with the Hamming distance and similarity scores $\mathcal{S}$ In the following we discuss the relationship between the kernel $K_k(E(X), E(Y))$ (Eq. 5.11) computed over similarity-preserving embedding and the original similarity kernel $K(X, Y|\mathcal{S})$ (Eq. 3.5) that uses $\mathcal{S}(\cdot, \cdot)$.

The resulting kernel value $K_k(E(X), E(Y))$ is an approximation of the total Hamming similarity between all pairs (α, β) of $B \times k$ submatrices

$$H(E(X)E(Y)) = \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} h_{\alpha\beta} \tag{5.12}$$

where $h_{\alpha\beta}$ is the Hamming similarity between the two $B \times k$ submatrices α and β (the number of corresponding identical bits). The kernel in Eq. 5.11 can be written as

$$K_k(E(X), E(Y)) = \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} \sum_{b=1}^{B} I(\alpha[b], \beta[b]) \tag{5.13}$$

$$= \sum_{\alpha \in X} \sum_{\beta \in Y} s_{\alpha\beta} \tag{5.14}$$

where $s_{\alpha\beta}$ is the number of identical rows in α and β, and $\alpha[b]$ denotes bth row of α.

It is easy to see that we have the following relationship between $s_{\alpha\beta}$, the number of of identical rows between α and β, and $h_{\alpha\beta}$, the hamming similarity between α and β

$$\max\{h_{\alpha\beta} - (k-1)B, 0\} \leq s_{\alpha\beta} \leq \frac{h_{\alpha\beta}}{k} \tag{5.15}$$

Using the above relationship between $s_{\alpha\beta}$ and $h_{\alpha\beta}$, we observe that the kernel value in Eq. 5.11 is related to the total Hamming similarity $H(E(X), E(Y))$, between embeddings $E(X)$ and $E(Y)$ as

$$\max\{H(E(X),E(Y))-\hat{B},0\} \leq K(E(X),E(Y)) \leq \frac{H(E(X),E(Y))}{k} \tag{5.16}$$

where $\hat{B} = \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} (k-1)B$ (note that $\hat{B}$ is bounded above by $|X||Y|kB$). By the similarity preserving property of E, $H(E(Y), E(Y)) \sim \mathcal{S}(X, Y)$. This together with the above bounds implies that the kernel value in Eq. 5.11 can be used to efficiently evaluate similarity kernel $K(X, Y|\mathcal{S})$ in Eq. 3.5.

6 Evaluation

We study the performance of our algorithms, in terms of predictive accuracy, on standard benchmark datasets for protein sequence analysis and music genre classification.

Datasets and experimental setup. We use four standard benchmark datasets to compare with previously published results: the SCOP dataset (7329 sequences with 2862 labeled) [22] for remote protein homology detection, the Ding-Dubchak dataset [2] (27 folds, 694 sequences) [5, 16] for protein fold recognition, multi-class remote fold recognition dataset [16], and music genre data [1] (10 classes, 1000 sequences consisting of 13-dim. MFCC feature vectors) [15] for multi-class genre prediction. For protein sequence classification under the semi-supervised setting, we also use the Swiss-Prot dataset, a collection of about 100K protein sequences, as an unlabeled dataset, following the setup of [11]. For remote protein homology experiments, we follow standard experimental setup used in previous studies [22] and evaluate average classification performance on 54 remote homology problems. For music genre classification we use 5-fold cross-validation error to evaluate classification performance. For multi-class fold prediction, we use standard data splits as described in [5, 16]. All experiments are performed on a *single* $2.8GHz$ CPU. The datasets used in our experiments and the supplementary data/code are available at `http://paul.rutgers.edu/~pkuksa/generalkernels/`.

Evaluation measures. The methods are evaluated using 0-1 and top-q balanced error rates as well as F1 scores and precision and recall rates. Under the top-q error cost function, a classification is considered correct if the rank of the correct label, obtained by sorting all prediction confidences in non-increasing order, is at most q. On the other hand, under the balanced error cost function, the penalty of mis-classifying one sequence is inversely proportional to the number of sequences in the target class (i.e. mis-classifying a sequence from a class with a small number of examples results in a higher penalty compared to that of mis-classifying a sequence from a large, well represented class). We evaluate reemote protein homology performance using standard Receiver Operat- ing Characteristic (ROC) and ROC50 scores. The ROC50 score is the (normalized) area under the ROC curve computed for up to 50 false positives. With a small number of positive test sequences and a large number of negative test sequences, the ROC50 score is typically more indicative of the prediction accuracy of a homology detection method than the ROC score.

6.1 Empirical performance analysis In this section we show predictive performance for several sequence analysis tasks using our similarity/distance-preserving symbolic embedding string kernels (Sec. 5) and abstraction-based string kernels (Sec. 4).

For protein sequences we use as feature vectors rows of the BLOSUM62 substitution matrices to represent 20 aminoacids (M step in Fig. 2, R=20). In case of the music data, input sequences are sequences of 13-dim. MFCC features (R=13). For symbolic Hamming embedding we use $B = 16, 32, 64$ and k-mers with k=5 (we use method described in [21] to obtain binary Hamming embeddings). For abstraction/clustering based kernel, we cluster a set of MFCC into $|\Sigma| = 2048$ clusters; for protein sequence data, 20-dim. BLOSUM score vectors are clustered into $|\Sigma| = 4$ groups (this corresponds to the number of typical groups of amino acids according to their hydropathy). For the experiments on the protein sequences, we report results for incorporating BLOSUM similarity between aminoacids using abstraction/cluster kernel (results using binary embedding are similar and not shown). For the music experiments, we compare clustering approach to incorporating $\mathcal{S}$ and similarity/distance-preserving Hamming approach that preserves (approximate) distance according to $\mathcal{S}$.

We consider the tasks of multi-class music genre classification [15], with results in Table 1, and the protein remote homology (superfamily) prediction [13, 8, 7] in Table 3. We also include results for multi-class fold prediction [5, 16] in Table 4 and Table 5.

On the music classification task (Table 1), we observe significant improvements in accuracy using

Table 1: Classification performance on music genre prediction (multi-class)

Method	Error
Spectrum (Abstraction, $\|\Sigma\| = 2048$)	34.5
Mismatch-(k=5,m=2, $\|\Sigma\| = 2048$) (Abstraction)	32.6
Double-(k=1,d=5) (Abstraction)	31.1
Spectrum (Abstraction + Embedding, B=32)	**26.1**
Double(k=1,d=5) (Abstraction + Embedding)	**25.9**

Table 2: Music genre classification performance and running time of the kernel as a function of the embedding size.

Embedding size	Error %	Running time (s), 1000×1000 kernel matrix computation	Running time (s), pairwise kernel computation $K(X, Y)$
B=16	28.3	24	0.05
B=32	26.1	61	0.12
B=64	28.2	202	0.28
(5,2) VQ $\|\Sigma\|$=2048	32.6	28	0.015
(5,2) VQ $\|\Sigma\|$=1024	32.5	26	0.018
Explicit Euclidean $\mathcal{S}$	-	-	220

the distance-preserving kernels. For instance, using distance-preserving embedding improves error from 34.5% using clustering (which does not retain actual distance) to 26.1%. Similarly, in the case of spatial kernel (double-(1,5)), we observe reduction in error from 31.1% to 25.9% when using the distance-preserving embedding. The obtained error rate (25.9%) on this dataset compares well with the state-of-the-art results based on the same signal representation in [15]. Table 2 summarizes the classification performance and running time for the similarity-preserving and abstraction kernels with varying values of the dimensionality of the embedding space B and the number of clusters $|C|$. We note that for mismatch-(k,m) kernel computation with use linear time sufficient-statistic based algorithm from [10].

The remote protein homology detection, as evident from Table 3, which incorporates biological distances (BLOSUM scoring function) into kernel computation, effectively improves over traditional string kernels. For instance, we observe improvement in the average ROC-50 score from 41.92 to 46.32 in the case of the mismatch kernel. We also observe that using the semi-supervised neighborhood kernel approach [22] with distance-based kernel improves over standard mismatch kernel-based cluster kernel. For example, for the neighborhood kernel computed on the unlabeled subset (4000 sequences) of the SCOP dataset, using abstraction/ clustering-based kernel (BLOSUM) achieves the mean ROC50 70.14 compared to ROC50 67.91 using the standard mismatch string kernel.

For multi-class protein fold recognition (Table 4), we similarly observe improvements in performance for clustering-based kernel over standard string kernels. The top-5 balanced error of 28.92% for the clustering-based mismatch neighborhood kernel using Swiss-Prot compares well with the best error rate of 35.28% for the state-of-the-art profile kernel [8, 16].

7 Conclusions

We presented new linear-time algorithms for inexact matching of the discrete- and continuous- valued string representations under general similarity metrics. The proposed approach uses similarity/distance-preserving embedding with string kernels to improve sequence classification. On four benchmark datasets, including music classification and biological sequence analysis problems, our algorithms effectively improve over string kernel baselines, while retaining efficient string kernel running times.

References

[1] Music genre prediction data set. http://opihi.cs.uvic.ca/sound/genres.

[2] Protein fold prediction data set. http://ranger.uta.edu/chqding/bioinfo.html.

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

[4] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

Table 3: Classification performance (mean ROC50) on protein remote homology

Method	Baseline	+Abstraction (BLOSUM)
Spectrum (n-gram) [13]	27.91	**35.14**
Mismatch [14]	41.92	**46.35**
Spatial sample [9]	50.12	**52.00**
Semi-supervised Cluster kernel [22]	67.91	**70.14**

Table 4: Classification performance on fold prediction (multi-class)

Method	Error	Top 5 Error	Balanced Error	Top 5 Balanced Error	F1	Top 5 F1
Baseline 1: PSI-BLAST [16]	64.80	51.80	70.30	54.30	-	-
Baseline 2: Substitution kernel [12] (BLOSUM62)	51.95	27.04	66.17	36.72	34.49	66.27
Baseline 3: Profile (5,7.5) (Swiss-prot)	49.35	20.36	76.67	35.28	26.05	68.09
Mismatch(k=5,m=1)	53.75	29.15	82.75	52.40	16.92	56.67
Mismatch (k=5,m=1) +Abstraction	**52.12**	**24.10**	**81.76**	**43.30**	**22.74**	**64.30**
Spatial sample kernel	48.7	25.08	73.04	44.05	30.57	62.37
Spatial sample kernel (hydropathy) +Abstraction	**47.88**	**19.38**	**70.81**	**30.99**	**32.86**	**74.57**
Semi-supervised Cluster kernel (Swiss-Prot)	48.86	19.54	72.88	34.06	26.59	70.07
Semi-supervised Cluster kernel (Swiss-Prot) +Abstraction	48.86	**18.40**	74.87	**28.92**	**27.06**	**74.24**

Table 5: Comparison on Ding and Dubchak benchmark data set

Method	Error	Top 5 Error	Balanced Error	Top 5 Balanced Error	F1	Top 5 F1
Baseline 1: PSI-BLAST	44.13	-	35.16	-	-	-
Baseline 2: Substitution kernel [12] (BLOSUM62)	45.43	22.72	48.02	25.05	53.54	75.89
Baseline 3: SVM(D&D) [5]	-	-	56.5	-	-	-
Baseline 4: Profile (5,7.5) (Swiss-prot)	36.03	16.19	37.78	18.46	73.03	87.56
Mismatch(5,1)	51.17	22.19	53.22	28.65	61.68	81.68
Mismatch (5,1) + Abstraction	**45.95**	**19.84**	**50.90**	**22.35**	**62.41**	**85.09**
Spectrum (k=3)	47.52	27.94	50.48	30.03	55.25	74.00
Spectrum (k=3) + Abstraction	**45.17**	**24.80**	**48.25**	26.92	**57.25**	76.12
Semi-supervised Cluster Kernel	29.77	13.32	29.13	13.36	78.17	89.76
Semi-supervised Cluster Kernel + Abstraction	29.77	**11.75**	**28.58**	**12.22**	**78.6**	**90.98**

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

[6] M. Gribskov, A. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84:4355–4358, 1987.

[7] 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.

[8] 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.

[9] P. Kuksa, P.-H. Huang, and V. Pavlovic. Fast protein homology and fold detection with sparse spatial sample kernels. In *ICPR 2008*, 2008.

[10] P. Kuksa, P.-H. Huang, and V. Pavlovic. Scalable algorithms for string kernels with inexact matching. In *NIPS*, 2008.

[11] P. Kuksa, P.-H. Huang, and V. Pavlovic. Efficient use of unlabeled data for protein sequence classification: a comparative study. *BMC Bioinformatics*, 10(Suppl 4):S2, 2009.

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

[13] 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.

[14] C. S. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In *NIPS*, pages 1417–1424, 2002.

[15] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *SIGIR '03*, pages 282–289, New York, NY, USA, 2003. ACM.

[16] I. Melvin, E. Ie, J. Weston, W. S. Noble, and C. Leslie. Multi-class protein classification using adaptive codes. *J. Mach. Learn. Res.*, 8:1557–1581, 2007.

[17] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[18] S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence SVM classifiers. In *ICML '05*, pages 848–855, New York, NY, USA, 2005.

[19] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[20] S. V. N. Vishwanathan and A. Smola. Fast kernels for string and tree matching. *Advances in Neural Information Processing Systems*, 15, 2002.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1753–1760. 2009.

[22] 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.