To appear in *The Workshop on Algorithms in Bioinformatics (WABI)*, Philadelphia, PA, Sept 2007.

Fast kernel methods for SVM sequence classifiers

Pavel Kuksa[1] and Vladimir Pavlovic[1]

Department of Computer Science, Rutgers University, Piscataway, NJ 08854 pkuksa@cs.rutgers.edu, vladimir@cs.rutgers.edu

Abstract. In this work we study string kernel methods for sequence analysis and focus on the problem of species-level identification based on short DNA fragments known as barcodes. We introduce efficient sorting-based algorithms for exact string k-mer kernels and then describe a divide-and-conquer technique for kernels with mismatches. Our algorithm for the mismatch kernel matrix computation improves currently known time bounds for these computations. We then address the fast kernel vector evaluation problem during SVM testing. Finally, we introduce the mismatch kernel problem with feature selection, and present efficient algorithms for it. Our experimental results show that, for string kernels with mismatches, kernel matrices can be computed 100-200 times faster. Evaluations of kernel vectors on new sequences using new techniques also require far less time than the standard approaches. In experiments with several DNA barcode datasets, k-mer string kernels also considerably improve identification accuracy compared to prior results. String kernels with feature selection demonstrate enhanced or similar classification performance with substantially fewer computations than full feature kernels.

1 Introduction

Biological species identification through DNA barcodes has been proposed recently in [1]. In the DNA barcoding setting, DNA sequencing of the mitochondrial region is used to obtain a relatively short sequence or DNA barcode that is subsequently used as a marker for identification and classification of the species. This approach to species identification contrasts traditional identification methods that rely on markers from multiple genomic locations. DNA barcoding has shown great promise due to increased adaptability, robustness, and predictive value for rapid and accurate identification of species. For instance, barcoding analysis via cox1 gene of moth and fly specimens intercepted at New Zealand's borders resulted in improved correct placement of previously unknown species or increased resolution of specimens [2].

Reliance of DNA barcoding on a single fragment of DNA sequence necessitates new computational methods to deal efficiently with this single sequence-based assignment. Several methods, based on pairwise alignments [3] or statistical approaches using evolutionary distances [4], have been applied to the tasks of identification and analysis of the DNA barcode data. However, a number of challenges remain to be addressed, including the accuracy of identification [5, 6, 4, 3] as well as the efficiency and scalability of computational methods.

In this study we investigate kernel classification methods for the DNA barcoding. Kernel-based classification demonstrated strong performance in many related tasks of biological sequence analysis, such as protein classification and remote homology detection [7–10]. Kernel methods approach classification by mapping original data into a set of points in the feature space that potentially makes it easier to detect complex relationship in the data. This, in turn, leads to learning algorithms that can exhibit higher classification accuracy and robustness. In our approach, species identification is performed by first transforming sequences (potentially of varying length) into fixed-length representations (string spectra) and then classifying sequences into one of many established species classes. For classification, we use Support Vector Machine (SVM) classifiers [11, 12]. As a result, the string kernel-based identification of species sequences for the DNA barcoding in our study demonstrates high accuracy and improved classification performance compared to previously employed methods.

The improved accuracy of kernel-based classification methods in the sequence domain is typically challenged by their computational complexity. To address the computational aspects of the method, we propose novel and efficient algorithms for solving the string kernel-based learning problems and introduce string kernels with feature selection which perform as well as the methods based on the full feature sets while having significantly lower computation cost.

There are several types of kernels for the biological sequences, including kernels derived from probabilistic models [13], k-mer string kernels [7, 8], and weighted-decomposition kernels [14]. In this work we focus on recently proposed k-mer string kernels. The use of the string kernels can be expensive and requires efficient ways to compute kernel matrices that are used during training, as well as kernel vectors (dot-products) for the prediction (testing on new sequences). We address computation complexity of the string kernels and show how to efficiently compute kernel matrices and kernel vectors therefore improving training speed and testing (prediction) times.

Our experimental results show that, for the string kernel with mismatches, kernel matrices can be computed by factors of 100-200 times faster depending on the size of dataset. The evaluation of the kernel vector on new sequences using new techniques also requires far less time compared to the standard approaches. We also observe that the k-mer string kernels considerably improve identification accuracy compared to the previously reported results on several barcode datasets. The string kernels with feature selection demonstrate enhanced or similar classification performance, however require substantially less amount of the computations compared to the full feature kernels.

This paper is organized as follows. We start by introducing efficient sorting-based algorithms for the exact string k-mer kernels (Section 3) . In Section 4 we describe a divide-and-conquer technique for the exact k-spectrum kernels and k-mer kernels with m mismatches which combined with the sorting improves currently known time bounds for the mismatch kernel computations. We proceed by addressing the problem of the fast evaluation of the kernel vector during testing (or training). We also introduce the mismatch kernel problem with feature selection and present algorithms for the efficient computation of the string kernels with feature selection (Section 5). A comparison of our kernel method with a baseline computational approach is discussed in Section 6. In Section 7 we consider a way to introduce positional information into the traditionally position-free spectrum kernels. Finally, we evaluate several feature spaces and provide comparative analysis of the performance of our method on three publicly available barcode datasets in Section 9.

2 Species identification problem in the DNA barcoding setting

Species identification problem can be described in the following way: given an unlabeled sample (specimen) X and a set SP of known species represented by their reference barcode sequences or models, the task is to assign the given sample to one of the known species (or decide that this sample does not belong to any of the known categories). We solve this global multiclass problem by dividing it into a collection of binary membership problems. In a binary membership problem, the task is to decide whether an input sequence belongs to a particular class. To solve each membership problem we build a kernel-based binary classifier.

Kernel-based classification methods [12, 11] provide a general paradigm for the design of classifiers with well studied theoretical properties and computational formalisms. The paradigm relies on the existence of positive-definite kernels, defined as inner products in feature spaces, that naturally induce similarity metrics among data points.

Kernel-based methods have been successfully applied to the domain of biological sequences. In particular, string kernels have been shown to achieve good performance on protein classification and homology detection tasks. The class of string kernels introduced recently includes the exact spectrum kernel [7], mismatch kernel [8], and profile kernel [9]. All the kernels have general form of

$$k(x, y) = \sum_i \phi_i(x)\phi_i(y)$$

where $\phi_i(x), i = 1...|\Phi(x)|$ are components of the feature vector $\Phi(x)$ that defines mapping of the input string x to the feature space. Both spectrum and mismatch kernels, as well as profiles kernel, project sequences into a feature space indexed by all k-length substrings (k-mers, k-tuples) for some fixed and a-priori chosen k. Projection in case of spectrum kernel is based on the set of explicit features that appear in the sequences. In the case of (k, m)-mismatch kernel, projection of a sequence x will include not only substrings that appear explicitly in the sequence, but also substrings that are in the (k, m)-neighborhoods of substrings that are present in x, where (k, m)-neighborhood of k-length substring s is defined as all substrings of length k

that differ from s in no more than m positions. In the case of the profile kernel, sequence is modeled by a profile [15] where each position is described by a symbol emission probability distribution. A neighborhood is then defined probabilistically and feature is included in the projection if its probability of being generated from profile is greater than some fixed threshold. An alternative approach based on Fisher kernels [13] defines a similar feature mapping based on the profile models.

We apply the kernel-based formalism to design of classifiers for binary species identification. Given a training set of species barcodes SP and their corresponding species labels S, a sequence kernel is used in a SVM setting to learn a species classifier for new sequences:

$$species(x = s) = \begin{cases} \text{yes}, \sum_{i \in M} \alpha_{i,s} k_s(x, x_i) > 0 \\ \text{no}, \ \text{otherwise} \end{cases}$$

where $\alpha_{i,s}$ and $M \subseteq SP$ are estimated in the learning process using standard SVM methods [12].

A critical point in this formalism, when applied to the domain of sequences, is the complexity of kernel computations. ([7, 8]) proposed an efficient algorithm using suffix-trees to address this problem. We next describe an alternative algorithm for sequence kernels that exhibits improved performance both in time and in space, compared to the traditional approach. The proposed algorithm can further incorporate feature selection to reduce dimensionality of the problems. Selection of a small subset of features not only implies computationally more efficient procedures, but also is biologically interesting since selected features can facilitate understanding of the species identity.

2.1 Barcode data representation

For the purpose of this study, a DNA barcode sequence X_j is a sequence $(x_1, \ldots, x_{n_j})$ where each x_i stands for one of the four nucleotides A, T, C, or G. Since sequences have variable lengths, to perform classification, we represent each sequence as a fixed-length vector of features, hence embedding the sequences into a space of fixed dimensionality. In our studies, we use two feature spaces: feature space indexed by k-length strings over alphabet Σ and feature space indexed by (state, symbol) tuples where state corresponds to the match state of a profile model.

3 Counting-sort formalism for string kernel algorithms

In this section we introduce sorting-based algorithms to more efficiently compute string k-mer kernels. Counting-sort based framework leads to fast and scalable practical algorithms for string kernels suitable for large k and m. In a counting-sort framework each substring of length k (k-mer) is considered as a k-digit integer number base $|\Sigma|$. A list of n integer k-digit numbers where each digit is from integer alphabet $1 \ldots |\Sigma|$ can be sorted in $\Theta(kn)$ time using k passes of counting sort. Space complexity of this approach is $\Theta(n)$ since we can reuse space and store only the current column to be sorted. Given n integers in a sorted order, one pass over the list is sufficient to output for each distinct element in the list frequency of its occurrence. Then the exact k-spectrum kernel for the two sequences can be computed in time $\Theta(kn)$ linear in the length n of sequences.

Given a set of N sequences, exact spectrum kernel matrix $K_{N \times N} = [k(x, y)]$ can be computed in linear $O(nN)$ space and $O(knN + \min(u, n) \cdot N^2)$ time where last term reflects complexity of updating the kernel matrix [1] and u is a number of unique features (k-mers) in the set (u is bounded above by $\min(nN, |\Sigma|^k)$). The proposed algorithm improves the time bounds compared to the suffix tree algorithms that have higher $O(N^2 kn)$ complexity, and is simpler and easy to implement. In summary, our counting sort based algorithm for the exact spectrum kernel performs the following steps:

[1] It is easy to see that the complexity of updating the matrix is $\min(u, n)N^2$. Consider the u-by-N matrix $C = [c_{i,j}]$ of k-mer counts, where $c_{i,j}$ is the number of times k-mer i occurs in the jth sequence. Since there are no more than $\min(u, n)N$ non-zero elements in C, the update complexity kernel matrix is $\min(u, n)N^2$.

Step 1. Extract k-length substrings from each input sequence x and store them in an array L, $O(knN)$[2].
Step 2. Sort list L using counting sort, $O(knN)$.
Step 3. Read off feature counts from sorted list by scanning and simultaneously update kernel matrix on each change in the feature value, $O(knN + u \cdot N^2)$.

For each unique feature f (there are u of them), in step 3 kernel matrix is updated as follows:

$$K(upd_f, upd_f) = K(upd_f, upd_f) + c_f c_f^T \tag{1}$$

where $upd_f = \{i : f \text{ is in } x_i\}$ is a set of input sequences that contain f and $c_f = [n_{x_i}]_{i \in upd_f}$ is a $|upd_f| \times 1$ vector of feature counts for each input sequence from upd_f.

In the case of (k, m)-mismatch kernel when up to m mismatches are allowed, a set of unique features u that are present in the dataset can be extracted first using the above algorithm and then used in computations instead of the original redundant set. This preprocessing step takes $O(knN)$ time, however, since nN (number of features collected from all the input sequences) is much larger than u in the case of DNA sequences over 4-symbol alphabet, this preprocessing step results in the improved performance of the algorithm.

4 Divide-and-conquer algorithms for exact and mismatch string kernels

While the sorting formalism results in an improved computational method for kernel evaluation it is also possible to gain further reduction in computational complexity. The efficient computation is achieved by using linear time character-based clustering to divide the problem into subproblems and the merging procedure that updates the kernel matrix.

Using a divide-and-conquer technique, the exact and mismatch kernel problems can be solved recursively as follows:

Step 1: Original set L of features composed of all k-mers extracted from N input sequences is divided into subsets $L_1, \dots, L_{|\Sigma|}$ using character-based clustering.
Step 2: The same procedure (Divide step) is applied to each of the subsets $L_1, \dots, L_{|\Sigma|}$ recursively. The depth of recursion is bounded by k (since clustering continues until there is no substrings left or depth k reached). Each node at depth k corresponds to some feature f and stores counts $n_{x_i}(f)$ (number of times f appears in x_i) for all the sequences that contain f, these counts are used to update kernel matrix as in (1).

The procedure above builds one recursion tree for all input sequences. At each level $l = 1 \dots k$ of the recursion tree there are $|\Sigma|^l$ clusters, each of the l-level cluster corresponds to a distinct substring of length l. Each cluster C consists of a number of subclusters SC where each subcluster is formed by k-mers from one particular sequence. At level k of the recursion tree, each node contains a collection of subclusters where each subcluster corresponds to a set of substrings from one particular sequence that are in the neighborhood of node feature, i.e., each node points to all the substrings that are in the neighborhood of the base string (node feature). The recursion procedure above results in an incremental algorithm for the mismatch kernel computation.

4.1 Analysis of incremental mismatch kernel algorithm

The time complexity of the incremental mismatch kernel algorithm can be expressed as

$$u \cdot \sum_{l=1}^{k} \sum_{i=0}^{\min(m,l)} \binom{l}{i} (\Sigma - 1)^i + u \cdot N^2$$

[2] Complexity of the feature extraction step can be reduced if k-mers are stored as integers (e.g. 32-bit word can store k-mers with value of k up to 16 when $|\Sigma| = 4$) Features then can be extracted from all the input sequences in $O(N(n + k))$ time since feature i can be computed from feature $i - 1$ in $O(1)$ time and it takes $O(k)$ time to compute the first feature. Extracted features then can be sorted in $\Theta(knN)$ time and $\Theta(nN)$ space

At each level l algorithm gives solution for the $(l, \min(m, l))$-mismatch problem. Last term in the expression for the time complexity reflects complexity of computing kernel matrix using N-length vectors of counts where each vector component $n_{x_i}(f)$ is the number of times k-mer f appears in the string x_i. For small values of m ($m \ll k$) complexity of processing each k-mer can be approximated as $k^m|\Sigma|^m$. As m grows, time complexity approaches limit of $|\Sigma|\frac{|\Sigma|^k-1|}{|\Sigma|-1}$, i.e. $O(u \cdot |\Sigma|^k)$[3]. Incremental mismatch kernel algorithm and recursive exact spectrum kernel algorithm are very similar with the only difference being that at each step of clustering some of the features that do not match base character may remain in the subset provided that number of mismatches is no more than m. Number of mismatches can be tracked using an indicator array that is initialized with k. At each step indicator value is reduced by 1 for those features that match base character, then at step l all the features for which indicator value is greater than $k - l + m$ are removed. The filtering step takes linear time. From implementation point of view, exact spectrum and incremental mismatch algorithms utilizing character-based clustering are the same except the filtering step.

4.2 Evaluation of the kernel vector

In the sections above we considered ways to compute efficiently the full kernel matrix. Similar approach can be followed in addressing the problem of computing a kernel vector for sequence s and N sequences $s_1, \ldots, s_N$ of length n. This type of the kernel computation is required for evaluation of a new sequence s during testing (prediction). Computation of the kernel vector is also a part of the SVM training if kernel caching is not used or cannot be used due to the size of the problem.

Evaluation of the new sequence s requires computation of the N kernel values $k(s, s_i)$, $i = 1, \ldots, N$, where N is the number of the support vectors. This evaluation can be done faster using sorting as an initial preprocessing step. Sorting and extracting unique k-mers from the support vectors takes linear $O(Nkn)$ time. Then the kernel vector $k(s, s_i)$ can be computed in $O(uvk + u'N)$ time (compared to $O(Nnvk + u'N)$ time when no preprocessing is done), where v is the size of the (k, m)-neighborhood. Since $u \ll Nn$ kernel vector can be evaluated considerably faster.

5 Kernels with feature selection

A common approach to feature selection relies on the filtering paradigm. The set A of all possible features is reduced to the subset F of the most informative features prior to applying any training methods. In our experiments we used term-frequency tf as our feature selection criteria:

$$tf_c(f_i) = \frac{num_c(f_i)}{num(f_i)}$$

where c is the class number, $num_c(f_i)$ is the number of times term f_i occurs in class c, $num(f_i)$ is the total number of times term f_i appears in all classes. We characterize utility of each feature f_i using maximum value $tf_{max}(f_i) = \max_c tf_c(f_i)$. Features are then selected globally according to the maximum of $\log tf_{max}(f_i)$. The criteria above is similar to the mutual information when all classes are equally likely. The criteria above is also suitable for imbalanced data sets when number of sequences per class varies substantially. Given a set $F = \{f_i\}_{i=1}^{|F|}$ of features and a set $S = \{x_i\}_{i=1}^{N}$ of sequences with typical length n, mismatch kernel can be computed in $O(|F|(k \cdot u + N^2) + knN)$ time (number of features $|F|$ in a typical setting is much less than total number of features u). Similarly, exact k-spectrum kernel with feature selection can be computed in $O(knN + |F|N^2)$ time.

5.1 Mismatch algorithm with feature selection

Brute force algorithms for exact spectrum and mismatch kernels with feature selection have the same time complexity of $O(|F|(knN + N^2))$. Using counting-sort formalism (section 3) exact matching kernel with

[3] It should be noted, however, that $m = k$ represents a special case that essentially takes into account only sequence lengths and kernel matrix can be computed in $O(N^2)$ time as $|\Sigma|^k len \cdot len'$, where len is $N \times 1$ vector of sequence lengths.

feature selection can be computed in $O(knN + |F|N^2)$ time and linear $\Theta(nN)$ space. Counting-sort based mismatch kernel with feature selection has time complexity of $O(|F|(k \cdot u + N^2) + knN)$, where u is the number of unique features and is bounded above by $\min(|\Sigma|^k, nN)$. In case of DNA sequences, when $|\Sigma| = 4$, for typical values of k, n and N, maximum number of features $|\Sigma|^k \ll nN$, i.e. $u \ll nN$, which gives substantial performance improvement.

Mismatch kernel problem with feature selection can be solved in linear time if we allow to use additional space. The idea is to build a suffix tree for the expanded feature set (selected features augmented by their neighbors) which needs $O(|F|vk)$ operations, where v is the size of the mismatch neighborhood. After the tree is built, we can traverse a tree in linear $O(Nkn)$ time. The complexity of the kernel matrix computation for the mismatch kernel with feature selection is then $O(Nkn + |F|N^2)$ (after construction of the tree). Alternative solution is to add selected features and their neighbors to the set of features extracted from the input sequences and sort the resulting set in $O(Nkn + |F|vk)$ time. After sorting feature counts can be computed in the traverse of the sorted array in linear time. The overall complexity of this approach is $O(Nkn + |F|vk + |F|N^2)$.

6 Comparison with baseline mismatch kernel algorithms

In this section we discuss and compare baseline methods for computing mismatch kernel matrices and kernel vectors, as well as ways to compute kernels with feature selection and their complexity.

6.1 Mismatch kernel

Explicit map algorithm. Each input sequence is mapped explicitly to the vector of size $|\Sigma|^k$ indexed by all substrings of length k. It takes $O(Nnvk)$ time to map all N sequences and $O(|\Sigma|^k N)$ space. The kernel matrix is then computed in $O(|\Sigma^k|N^2)$ time by computing pairwise dot-products between the spectrum representations. Although this method is well suited for the exact spectrum kernel (for small k and $|\Sigma|$) where mapping can be done in constant time, in the case of the mismatch kernel mapping operation for each k-mer can no longer be done in the constant time since each k-mer is now mapped to its v neighbors, where v is the size of the (k, m)-neighborhood, i.e. one needs to generate a set of v indices for each k-mer which is an expensive operation.

Explicit map with presorting. The explicit map algorithm can be accelerated by first sorting k-mers and extracting a set of u unique k-mers (this step takes linear time). Then only these unique k-mers are mapped in $O(uvk)$ time. The overall speed improvement can be substantial since $u \ll nN$ and mapping step can be performed $O(nN/u)$ times faster.

6.2 Mismatch kernel with feature selection

In case of the feature selection, a small subset of k-mers is preselected and only the selected features will contribute to the kernel value. Since we allow up to m mismatches, the target feature f is said to be present in the input sequences if there is a k-mer in the input which neighborhood contains f. We now briefly discuss two simple algorithms for the mismatch kernel with feature selection.

A straightforward algorithm. The straightforward algorithm for the mismatch kernel with feature selection is to count separately for each of the selected features number of times it appears in the input in a simple scan of the input. The complexity of this approach is $O(|F|Nkn + |F|N^2)$, where $|F|$ is the number of the features selected.

Explicit mapping algorithm. Explicit mapping algorithm for the mismatch kernel computation can be extended to incorporate feature selection by using only the selected positions in the sequence spectrum representations to compute kernel instead of all the positions (i.e. use reduced spectrum representations instead of full representations).

Table 1 summarizes complexity of the different algorithms for the kernel matrix computation. Complexity of the kernel vector computations is summarized in Table 2. Table 3 compares complexity of the different algorithms for the mismatch kernel with feature selection. It should be noted that the explicit map approaches require larger storage $O(N|\Sigma|^k)$ compared to the linear space required by our divide-and-conquer approaches.

Table 1. Complexity of the mismatch kernel matrix computation

EM	$Nnvk + \|\Sigma\|^k N^2$
EM+Sort	$Nnk + uvk + uvN + \|\Sigma\|^k N^2$
DC	$Nnvk + u'N^2$
DC+Sort	$Nkn + uvk + u'N^2$

EM=Explicit map, EM+Sort=EM with presorting, DC=divide and conquer, v=size of the (k, m)-neighborhood, u=number of different k-mers in the input sequences, u' is the total number of different k-mers in the input sequences including neighbors

Table 2. Complexity of the mismatch kernel vector computation

EM	$Nnvk + \|\Sigma\|^k N$
EM+Sort	$Nnk + uvk + uvN + \|\Sigma\|^k N$
DC	$Nnvk + u'N$
DC+Sort	$Nkn + uvk + u'N$

Table 3. Complexity of the mismatch kernel with feature selection. Kernel matrix computation

brute force	$\|F\|Nkn + \|F\|N^2$
EM	$Nnvk + \|F\|N^2$
EM+Sort	$Nkn + uvk + uvN + \|F\|N^2$
DC	$Nn\|F\|k + \|F\|N^2$
DC+Sort	$Nkn + uk\|F\| + \|F\|N^2$

7 Kernels with position information

Position of features in the original sequences can play a key role in interpretability of the resulting model when feature selection is used. However, string kernels inherently contain no positional information. One way to induce position information is via context-specific kernels. Context-specific kernels take into consideration information about similarity of the contexts where matches occur in the form of different weights depending on the degree of context similarity. Matches in highly dissimilar environments induce low weights, whereas in the case of highly similar contexts there is no penalty and the weight value is high. Percentage of matches w_m (2) between the two contexts c_1 and c_2 and context composition-based measures w_c (3) were used in our experiments as measures of context similarity.

$$w_m(c_1, c_2) = \frac{|\{i : c_1(i) = c_2(i)\}|}{\text{context length}} \tag{2}$$

$$w_c(c_1, c_2) = exp(-\frac{(cc_1 - cc_2)'(cc_1 - cc_2)}{2}) \tag{3}$$

where cc_1 (cc_2) is $|\Sigma| \times 1$ vector of character frequencies.

8 Related work

Traditional algorithms for computing string kernels [7, 8, 16] rely on suffix trees and arrays. Exact matching k-spectrum kernel $K_k(x, y)$ for two sequences x and y of length n can be computed in $O(kn)$ time using suffix trees. All-substrings kernel problem [16] has also been solved in the time linear in the length of sequences using suffix trees along with matching statistics. However, it is not necessary to build suffix trees in order to obtain a kernel. Commonly used linear time suffix tree construction algorithms (e.g., Ukkonen algorithm [17]) have large running time constants and large memory requirements. Moreover, in many applications algorithms make no use of suffix links after construction of a tree.

In our framework, the exact k-spectrum kernel $K_k(x, y)$ can be computed in time $\Theta(kn)$ linear in the length of sequences x and y, however proposed counting sort formalism provides a much simpler and more efficient implementation, as well as minimal memory requirements. Also, in our framework, the k-mer spectrum kernel can be efficiently computed for N sequences of length n in $O(Nnk)$ time and linear space using sorting, resulting in the elimination of the large storage overhead and time constants associated with the suffix-tree based algorithms [7].

Mismatch kernel over a set S of N sequences each of length n reported in [8, 18] to have complexity of $O(N^2nk^{m+1}|\Sigma|^m)$. Our divide-and-conquer algorithm for mismatch kernel computation improves this bound and has running time of $O(u \cdot k^{m+1}|\Sigma|^m + u \cdot N^2)$ where u is the number of different k-mers in the input sequences. In [8] mismatch algorithm considers (k, m)-neighborhood of each k-mer. The size of the neighborhood is shown to be $O(k^m|\Sigma|^m)$. It should be noted that in this case the total number of features that is considered by algorithm can exceed the natural upper bound [4] of $|\Sigma|^k$. In our divide-and-conquer algorithm for mismatch kernel we take different approach: our algorithm clusters features of S and naturally finds groups of features that are (k, m)-neighbors, size of the resulting clusters (subclusters that correspond to different input strings) gives desired counts of number of times features occur in the input strings.

9 Experiments and results

In our experiments we use three data sets of DNA barcodes. *Fish larvae*[5] dataset contains barcodes for 7 species and consists of 56 sequences. The second data set *Astraptes.Fulgerator*[6] contains 466 barcodes from 12 species. The number of sequences per species class in the second dataset varies substantially from as few as 3 barcodes to as many as 100 barcodes. Substantial variations in the lengths of the barcodes within classes as well as across classes are also present. Finally, a large *Hesperiidae* dataset contains more than 2185 sequences and 355 classes.

9.1 Classification performance

We evaluate performance of all methods using the ROC50 score [20] as well as the cross-validation error. In case of Fisher kernel, profile HMM models were estimated from multiple alignment of sequences from one species class and then used as feature extractors to map barcode sequences to the space of fixed dimensionality. The classifier that we used is the SVM with RBF kernel normalized by the distance to the nearest neighbor. For SVM classifiers, we use existing implementation of SVM from a machine-learning package Spider [7]. In the experiments with feature selection, a small subset of k-mers is preselected before learning a classifier using filtering approach.

We compare the performance of the full feature kernels and kernels based on the reduced feature set in Fig. 1. Dimensionality of the full feature space is 500×4 in our experiments. Dimensionality of the reduced feature set varies from 2 to 50 dimensions. Fig. 1 compares performance of the kernels on the basis of the number of families with the score higher than corresponding ROC50 score (an ideal method should result in a plot where for any ROC50 score the number of families that achieve that score is equal to the total number of families). ROC50 plots for the Fisher kernel clearly demonstrate that the kernel with feature selection achieves higher performance compared with the full feature set kernel. Fisher kernel with feature selection has performed as well as the mismatch kernel with feature selection, however Fisher kernel offers position information and can therefore facilitates interpretation of the resulting model.

Table 4 displays performance results (crossvalidation error rates) of all eight methods on the *A.fulgerator* dataset for each of the classes. Similar validations are provided in Table 5 for the *fish larvae* dataset. The results indicate that the performance of the kernels with feature selection (SFK, SSK, SMK, SCON) on both datasets is, at worst, indistinguishable from that of kernels based on full feature sets (FK, SK, MK, CON). In some cases, as with the Fisher kernel, the feature reduction has resulted in significant improvement (on *A.fulgerator* dataset, SFK is better than FK with a p-value of 0.0063 as measured by a two-tailed signed rank test ([21])).

Figure 2 shows performance on the *Hesperiidae* dataset. The results indicate that the string kernels with feature selection performed extremely well on this dataset obtaining perfect results in more than 94% test cases. Feature selection results in improved performance for the exact spectrum kernel. In the case of the

[4] For example, one of our datasets contains N=466 DNA sequences of length $n = 600$, for $|\Sigma| = 4, k = 5, m = 1$, u is bounded by $|\Sigma|^k = 1024$, $N^2nk^{m+1}|\Sigma|^m$ is 1.3e+10, $u(k^{m+1}|\Sigma|^m + N^2)$ = 2.2e+08 which is approximately 50 times less.

[5] See [3] for the details on dataset.

[6] From Barcode of Life Data Systems (BOLD) collection *www.barcodinglife.org.* see [19] for detailed description of the dataset.

[7] Available from *http://www.kyb.mpg.de/bs/people/spider/.*

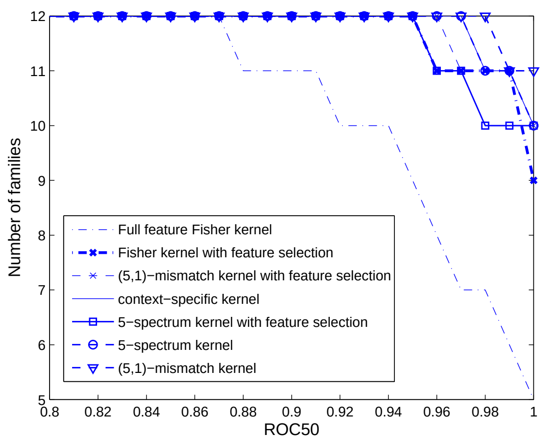

Fig. 1. Comparison of performance of the Fisher kernel, (5,1) mismatch with feature selection and context-specific kernel. Number of families with the ROC-50 score greater than threshold vs. ROC-50 score (*Astraptes fulgerator* dataset). Note the positive role of the feature selection, especially in the case of Fisher kernels.

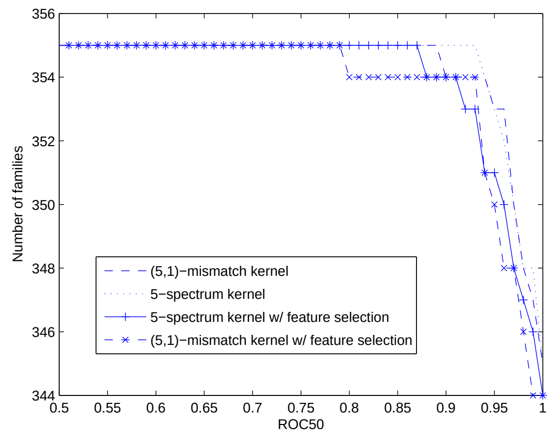

Fig. 2. Comparison of performance of string kernels with and without feature selection. Number of families with the ROC-50 score greater than threshold vs. ROC-50 score (*Hesperiidae* dataset). Reducing the number of features correlates mostly positively with the identification performance.

Table 4. Cross-validation error rates (%) for *A.fulgerator* species.

class	FK	SK	MK	CON	SFK	SSK	SMK	SCON
BYTTNER	0.43	0	0	0	0	0	0	0.21
CELT	1.07	0	0	0	0	0	0	0
FABOV	1.29	0	0	0	0	0.21	0.21	0.21
HIHAMP	0.86	0	0	0	0	0	0	0
INGCUP	1.30	0	0.64	0	0.22	0.64	0.64	0.21
LOHAMP	1.30	0	0	0	0	0	0	0
LONCHO	1.06	0	0	0	0	0.21	0	0.43
MYST	0.64	0.64	0.85	0.64	1.07	1.28	0.64	0.85
NUMT	0.22	0	0	0	0	0	0	0
SENNOV	2.57	1.07	0.86	1.29	1.71	1.07	1.29	1.5
TRIGO	0.86	0	0	0	0	0	0	0
YESENN	3.86	0.43	0.43	0.43	1.29	0.86	0.43	0.43

number of features for SSK, SMK and SCON is 100,
for SFK number of features is class-specific:
5, 2, 5, 30, 10, 10, 5, 5, 50, 5, 5, and 50, respectively

Table 5. Cross-validation error rates (%) for *Fish larvae* species.

class	SK	MK	CON	SSK	SMK	SCON
Perca	1.66	1.66	1.66	0	1.66	1.66
Rutilus	3.33	0.05	3.33	5.0	3.33	0.09
Gasterosteus	0	0	0	1.66	0	0
Barbatula	0	1.66	0	0	0	0
Lota	3.33	3.33	3.33	3.66	3.33	3.33
Anguilla	0	0	0	0	0	0
Phoxinus	3.09	3.09	3.09	3.66	5.33	3.66

FK = SVM-Fisher, SK = 5-spectrum kernel,
MK = (5,1)-mismatch kernel,
CON = (5, 10)-context-specific kernel,
S = with feature selection

Table 6. Comparison of the identification accuracy (avg.error/s.d.)

method	Astraptes species	Fish species	Hesperiidae species
svm w/ linear kernel (one-vs-rest)	0.0074/0.0092	0.1342/0.1358	0.0132/0.0022
svm + pca	0.0067/0.0074	0.1692/0.0707	0.0168/0.0038
nearest neighbor	0.0251/0.0304	0.1552/0.0893	0.1038/0.0130
ridge regression (one-vs-rest)	0.0215/0.0207	0.3077/0.1203	0.1121/0.0165
nearest neighbor + pca	0.0300/0.0271	0.1521/0.1387	0.0895/0.0153
PSI-BLAST	0.0963/0.0658	0.1615/0.0765	0.0160/0.0042
Fisher kernel	0.0415/0.0182	0.1245/0.0903	-

mismatch kernel the reduced set of features provides performance very similar to the full feature mismatch kernel. However the cost of computation for the mismatch kernel with reduced feature set is significantly lower than that of the full feature kernel, as shown in Sec. 5. Average cross-validation error rates on the *Hesperiidae* dataset are $3.7 \cdot 10^{-4}$ (maximum $1.5 \cdot 10^{-2}$) and $3.5 \cdot 10^{-4}$ (maximum $1.5 \cdot 10^{-2}$) for the spectrum and mismatch kernels, respectively.

The use of feature selection for the string kernels has resulted in a substantial reduction in the number of features (only 10% of features were selected) for all our test data sets, whereas performance has remained the same or improved. Feature selection in case of the Fisher kernel resulted not only in the improved performance, but also in even more dramatic decrease in the number of features (see Table 4).

9.2 Comparison to other methods

We compared performance of the string kernel-based method using SVM as a base learner with a number of other classification methods. In particular, we evaluated Fisher kernel method [13], PSI-BLAST, ridge regression and nearest neighbor methods. In the Fisher kernel approach we have built profile HMM models for each sequence class. All sequences are mapped to the space of fixed dimensionality $M \cdot |\Sigma|$, where M denotes the profile length. We used recursive feature elimination technique [22] where f_j is the output of jth binary classifier.

Table 6 displays classification performance results (accuracy of identification) for the methods tested on the three barcode datasets (note that the complexity of estimating Fisher kernel is very high and the results are not included for the large *Hesperiidae* set). We observe that k-mer string kernels considerably improve identification accuracy compared to previously reported results of [5, 4] (for example, on *Astraptes* dataset [19], the test error rate of multi-class SVM is only 0.67% compared to 9% in [5] or 20% in [4]).

9.3 Running time analysis

We performed running time analysis of our algorithms to demonstrate the behavior of the proposed algorithms under variety of circumstances, including various feature filtering levels, mismatch factors, and sequence feature lengths. We implemented and tested our algorithms in MATLAB. On a 2.8Ghz machine with 1GB RAM (MATLAB v.7.0.4.352), running time of our mismatch kernel algorithm on *Astraptes.fulgerator* data set ($N = 466$, $n = 600$) is 16.92 seconds and 240.36 seconds on a larger *Hesperiidae* dataset (N=2185, n=600) (to compute full $(N \times N)$-kernel matrix, $k = 5, m = 1$). It takes about 2820 seconds and about 20 hours, respectively, to compute the same matrices when we used publicly available string kernel package that implements the state-of-art method for spectrum/mismatch kernel[8]. Our experiments show order of magnitude running time improvement (Table 7) for the k-mer kernel with m mismatches (by factors of 100-200 times depending on the dataset size)[9]. We also observe that our extension of the explicit map (EM) algorithm that uses sorting as an initial preprocessing time results in significant speed improvements, however the explicit map algorithm requires much larger storage (exponential in $|\Sigma|$ and k) compared to the divide-and-conquer algorithm. Table 8 shows running times for the mismatch kernel matrix computation in the case when feature selection was used (filtering level in the table indicates percentage of the features filtered out). As we can see from the results, explicit map algorithms do not not scale with the number of the selected features, whereas divide-and-conquer approach scales almost linearly with the number of the selected features. The running times for the mismatch kernel vector computation are listed in Table 9. We observe that in case of the kernel vector evaluation, divide-and-conquer approach outperforms in many cases explicit mapping with sorting, especially for larger k and m (note that the for large k, explicit map algorithm have exceeded memory capacity), which makes our algorithms also particularly suited for the fast evaluation of new sequences during testing.

[8] From *http://www1.cs.columbia.edu/compbio/string-kernels/*

[9] This improvement is especially significant in light of the differences in the two implementations: MATLAB is an interpretive language while the competing package is implemented in C.

Table 7. Running time comparison. Mismatch kernel matrix computation

data set	K	EM	EM+Sort	D&C	String kernel package
Astraptes	5	202.11	3.14	16.92	2820
Hesperiidae	5	938.79	14.73	240.36	75219

EM=Explicit map, EM+Sort = EM with presorting, D&C=divide and conquer

Table 8. Mismatch kernel with feature selection. Computation of the kernel matrix (running time, s)

	Astraptes data			Hesperiidae data		
filtering level	EM	EM+ Sort	D&C	EM	EM+ Sort	D&C
0.1	212.23	4.03	12.91	961.94	17.75	163.80
0.2	212.01	3.62	11.43	964.55	19.83	144.61
0.3	185.62	3.88	10.41	982.72	17.20	122.34
0.4	193.08	3.76	9.51	974.43	18.73	113.83
0.5	193.74	2.31	8.54	989.4	18.56	89.25
0.6	196.87	2.30	7.47	969.48	17.37	78.12
0.7	192.54	2.51	6.31	977.07	14.85	52.32
0.8	200.59	2.51	5.18	964.67	10.73	39.18
0.9	184.75	3.60	3.74	966.68	10.57	23.92

EM=Explicit map, EM+Sort = EM with presorting, DC=divide and conquer

Table 9. Mismatch kernel. Computation of the kernel vector $k(s, s_i), i = 1, \ldots, N$(running time, s)

k m	Astraptes data			Hesperiidae data		
	D&C	DC+Sort	EM+ Sort	D&C	D&C+ Sort	EM+ Sort
5 1	9.25	3.25	3.06	53.36	14.44	13.70
6 1	12.64	4.95	3.92	70.70	12.31	18.74
7 1	18.01	9.15	5.84	78.39	16.35	48.62
8 1	26.58	20.37	9.99	113.46	29.57	-
9 1	39.88	42.70	-	200.62	99.37	-
5 2	33.28	3.9	5.30	197.34	26.23	20.03
6 2	53.09	5.71	8.28	354.11	34.92	35.60
7 2	80.65	10.14	12.50	537.61	49.31	75.63
8 2	121.02	23.79	20.00	797.65	82.53	-
9 2	192.04	70.88	-	1166.6	173.9	-

EM=Explicit map, EM+Sort = EM with presorting, DC=divide and conquer

10 Conclusions

In this paper we present a kernel classification based approach to the DNA barcoding problem that resulted in substantially improved identification accuracy compared to traditional approaches. We also present a framework for efficient computation of the string kernels that results in substantial speed improvements. We introduce string kernels with feature selection which have lower computational cost and, at the same time, comparable or better classification performance. The presented algorithmic approaches to implementing string kernels are general and can be applied to many other problems of biological sequence analysis.

We have presented a counting-sort formalism for the exact spectrum kernel and divide-and-conquer technique for the string kernels on DNA barcode data that provides a fast and scalable solution for many string kernel computation problems. In particular, for an input set S of N sequences over alphabet Σ with each sequence of typical length n we developed:

An $\Theta(knN + \min(u, n) \cdot N^2)$ time and $O(nN)$ space algorithm based on counting sort for the exact k-spectrum kernel
An $O(uk^{m+1}|\Sigma|^m + u \cdot N^2)$ time divide-and-conquer algorithm for the (k, m)-mismatch kernel
An improved $O(uvk + u'N)$ time algorithm for the mismatch kernel vector evaluation during testing
An $O(Nkn + |F|N^2)$ mismatch algorithm with feature selection for the kernel matrix computation

The main idea behind our first result is to consider k-mers as numbers base $|\Sigma|$; in contrast, previous algorithms for spectrum kernels consider k-mers as strings and build suffix trees to compute kernel. Our first result leads to a simple and easy to implement algorithm for exact k-spectrum kernel that matches bound for suffix-tree based algorithms. The main idea underlying our result for the mismatch kernel with improved running time is that of character-based clustering of k-mers and the use of divide-and-conquer technique. We present a divide-and-conquer algorithm for the (k, m)-mismatch kernel that improves previously known bound on running time and is more memory efficient and easy to implement. The algorithms were successfully applied for computation of string kernels for sequence classification in the DNA barcoding task.

We have demonstrated that the use of feature selection applied to the high dimensional space of string sequence features can often result in dramatic reduction in the number of features and may be of particular interest in the DNA barcoding setting. Reduced set of features implies that not only more effective computations become possible (as we have demonstrated), but can also facilitate biological interpretability of the resulting models, a task that is being addressed in our ongoing work.

References

1. Hebert, P.D.N., Cywinska, A., Ball, S., deWaard, J.: Biological identifications through dna barcodes. In: Proceedings of the Royal Society of London. (2003) 313–322
2. Armstrong, K., Bal, S.: Dna barcodes for biosecurity: invasive species identification. Philos. R. Soc. Lond. B. Biol. Sci. **360**(1462) (October 2005) 1813–1823
3. Steinke, D., Vences, M., Salzburger, W., Meyer, A.: Taxi: a software tool for dna barcoding using distance methods. Philosophical Transactions of the Royal Society B: Biological Sciences **360**(1462) (2005) 1975–1980
4. Nielsen, R., Matz, M.: Statistical approaches for dna barcoding. Systematic Biology **55**(1) (2006) 162–169
5. Matz, M.V., Nielsen, R.: A likelihood ratio test for species membership based on dna sequence data. Philosophical Transactions of the Royal Society B: Biological Sciences **360**(1462) (2005) 1969–1974
6. Meyer, C.P., Paulay, G.: Dna barcoding: error rates based on comprehensive sampling. PLoS Biol **3**(12) (December 2005)
7. Leslie, C.S., Eskin, E., Noble, W.S.: The spectrum kernel: A string kernel for svm protein classification. In: Pacific Symposium on Biocomputing. (2002) 566–575
8. Leslie, C.S., Eskin, E., Weston, J., Noble, W.S.: Mismatch string kernels for svm protein classification. In Becker, S., Thrun, S., Obermayer, K., eds.: NIPS, MIT Press (2002) 1417–1424
9. Kuang, R., Ie, E., Wang, K., Wang, K., Siddiqi, M., Freund, Y., Leslie, C.: 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), Washington, DC, USA, IEEE Computer Society (2004) 152–160
10. Cheng, J., Baldi, P.: A machine learning information retrieval approach to protein fold recognition. Bioinformatics **22**(12) (June 2006) 1456–1463
11. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press (2002)
12. Vapnik, V.: Statistical learning theory. Wiley (1998)
13. Jaakkola, T., Diekhans, M., Haussler, D.: A discriminative framework for detecting remote protein homologies. Journal of Computational Biology **7**(1-2) (2000) 95–114
14. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: ICML '05: Proceedings of the 22nd international conference on Machine learning, New York, NY, USA, ACM Press (2005) 585–592
15. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1999)
16. Vishwanathan, S.V.N., Smola, A.J.: Fast kernels for string and tree matching. In: NIPS. (2002) 569–576
17. Ukkonen, E.: Constructing suffix trees on-line in linear time. In: Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1, North-Holland (1992) 484–492
18. Leslie, C., Kuang, R.: Fast string kernels using inexact matching for protein sequences. J. Mach. Learn. Res. **5** (2004) 1435–1455
19. Hebert, P.D.N., Penton, E.H., Burns, J.M., Janzen, D.H., Hallwachs, W.: Ten species in one: Dna barcoding reveals cryptic species in the neotropical skipper butterfly astraptes fulgerator. In: PNAS. Volume 101. (2004) 14812–14817
20. Gribskov, M., Robinson, N.L.: Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. Computers & Chemistry **20**(1) (1996) 25–33
21. Salzberg, S.L.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov. **1**(3) (1997) 317–328
22. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1-3) (2002) 389–422