Fast Kernel Methods for SVM Sequence Classifiers

Pavel Kuksa and Vladimir Pavlovic

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 algorithms for mismatch kernel matrix computations improve currently known time bounds for these computations. We then consider 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 than traditional approaches. Kernel vector evaluations on new sequences show similar computational improvements. On several DNA barcode datasets, k-mer string kernels considerably improve identification accuracy compared to prior results. String kernels with feature selection demonstrate competitive performance with substantially fewer computations.

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, DNA barcode, that is subsequently used as a marker for species identification and classification. This approach contrasts traditional identification methods that rely on markers from multiple genomic locations. DNA barcoding has shown great promise due to increased 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 short 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 [3–6], 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 [7–9]. There are several types of kernels for the biological sequences, including kernels derived from probabilistic models [10], k-mer string kernels [7, 8], and weighted-decomposition kernels [11]. In this work we focus on recently proposed k-mer string kernels.

In our approach, species identification is performed by first transforming sequences (potentially of varying length) into fixed-length representations (string spectra) and then classifying them into one of many established species classes using Support Vector Machine (SVM) classifiers [12, 13]. As a result, the string kernel-based species identification 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. We also 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.

Our experimental results show that, for string kernels with mismatches, kernel matrices can be computed by factors of 100-200 times faster. Identification of new sequences similarly requires significantly less time than 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. String kernels with feature selection demonstrate competitive classification performance with substantially fewer computations.

This paper is organized as follows. We start by introducing efficient sorting-based algorithms for exact string k-mer kernels (Section 3) . In Section 4 we describe a divide-and-conquer technique for 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 then introduce the mismatch kernel problem with feature selection and present algorithms for efficient computations 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. Finally, we evaluate several feature spaces and provide comparative analysis of the performance of our method on three publicly available barcode datasets in Section 8.

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.

We apply the kernel-based formalism [12, 13] 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 using standard SVM methods [13].

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.

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, the spectrum kernel matrix can be computed in linear $O(nN)$ space and $O(knN + \min(u, n) \cdot N^2)$ time, where the last term reflects the complexity of updating the kernel matrix [1] and u is the number of unique k-mers in the set bounded above by $\min(nN, |\Sigma|^k)$. The proposed algorithm improves the time bounds compared to the suffix tree algorithms with higher $O(N^2 kn)$ complexity, and is simpler and easy to implement. In summary, our counting sort algorithm for the 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 and store k-mers from the input sequences, $O(knN)$ time[2].
Step 2. Sort obtained list L using counting sort, $O(knN)$.
Step 3. Compute feature counts by scanning the sorted list and update the kernel matrix on each change in the feature value, $O(knN + \min(u, n) \cdot N^2)$.

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 $upd_f = \{i : f \in x_i\}$ is a set of input sequences that contain f and $c_f = [n_{x_i}(f)]_{i \in upd_f}$ is a vector of feature counts for each 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 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, 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 $L_1, \ldots, L_{|\Sigma|}$ using character-based clustering.
Step 2: The same procedure (Divide step) is applied to each of the subsets $L_1, \ldots, L_{|\Sigma|}$ 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).

[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

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

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

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 the cost of computing kernel matrix using N-length vectors of feature counts. 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 the 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 features that match the 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.

5 Kernels with Feature Selection

A common approach to feature selection relies on the filtering paradigm, when the subset F of the most informative features is extracted prior to learning. In our experiments, we use term-frequency $tf_c(f_i) = \frac{num_c(f_i)}{num(f_i)}$ for feature selection, where $num_c(f_i)$ and $num(f_i)$ are the number of times term f_i occurs in class c and in all classes, respectively. We characterize utility of each feature f_i using

[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.

the 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, and is also suitable for imbalanced data sets when number of sequences per class varies substantially.

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) the exact matching and mismatch kernels with feature selection can be computed in $O(knN+|F|N^2)$ and $O(|F|(k{\cdot}u+N^2)+knN)$ time, respectively, where u is the number of unique features and is bounded above by $\min(|\Sigma|^k, nN)$. In case of DNA barcode sequences ($|\Sigma| = 4$), for typical k, n and N, $u \ll nN$, which gives substantial performance improvement.

Mismatch kernel problem with feature selection can be solved in linear time using additional space. It takes $O(|F|vk)$ operations to build a suffix tree for the selected features and their neighbors, where v is the size of the mismatch neighborhood. The complexity of the mismatch kernel matrix computation using this tree is then $O(Nkn+|F|N^2)$. Alternative solution is to add selected features and their neighbors to the set of features extracted from the input sequences and then compute feature counts in linear $O(Nkn + |F|vk)$ time using sorting.

6 Comparison with Baseline Mismatch Kernel Algorithms

In this section we discuss and compare baseline methods for mismatch kernel computations, as well as kernels with feature selection and their complexity.

Explicit map algorithm. Each input sequence is mapped explicitly to the vector of size $|\Sigma|^k$ indexed by all substrings of length k using $O(Nnvk)$ time and $O(|\Sigma|^k N)$ space, the kernel matrix can then be computed in $O(|\Sigma^k|N^2)$ time. Although this method is well suited for the exact spectrum kernel (for small k and $|\Sigma|$) with constant time mapping, mapping in the mismatch kernel is no longer a constant time since each k-mer is now mapped to its v neighbors.

Explicit map with presorting. Unique k-mers are first extracted using sorting and then mapped in $O(uvk)$ time. The overall speed improvement can be substantial since $u \ll nN$ and the mapping can be performed $O(nN/u)$ times faster.

In case of the feature selection, a small subset of k-mers is preselected and only the selected features contribute to the kernel value. The explicit mapping algorithm 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.

Table 1 summarizes complexity of the different algorithms for the mismatch kernel computations. It should be noted that EM approaches require larger $O(N|\Sigma|^k)$ storage than our divide-and-conquer approaches.

Table 1. Complexity of the mismatch kernel computations

	Mismatch kernel matrix	Mismatch kernel matrix with feature selection	Mismatch kernel vector
EM	$Nnvk + \lvert\Sigma\rvert^k N^2$	$Nnvk + \lvert F\rvert N^2$	$Nnvk + \lvert\Sigma\rvert^k N$
EM+ Sort	$Nnk + uvk +$ $uvN + \lvert\Sigma\rvert^k N^2$	$Nkn + uvk + uvN + \lvert F\rvert N^2$	$Nnk + uvk + uvN + \lvert\Sigma\rvert^k N$
DC	$Nnvk + u'N^2$	$Nn\lvert F\rvert k + \lvert F\rvert N^2$	$Nnvk + u'N$
DC+ Sort	$Nkn + uvk + u'N^2$	$Nkn + uk\lvert F\rvert + \lvert F\rvert N^2$	$Nkn + uvk + u'N$

EM=explicit map, EM+Sort=EM with presorting, DC=divide and conquer, v=neighborhood size, u=number of different k-mers in the input, u'=number of different k-mers including neighbors

7 Related Work

Traditional algorithms for computing string kernels [7, 8, 14] rely on suffix trees and arrays. Exact k-spectrum kernel for two sequences x and y of length n can be computed in $O(kn)$ time using suffix trees. All-substrings kernel problem [14] has also been solved in the linear time using suffix trees and matching statistics. However, it is not necessary to build suffix trees in order to obtain a kernel. Commonly used linear suffix tree algorithms (e.g., Ukkonen algorithm [15]) 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 time constants and storage overhead of the suffix-tree based algorithms [7].

Mismatch kernel over a set S of N sequences each of length n reported in [8, 16] to have complexity of $O(N^2 nk^{m+1}|\Sigma|^m)$. 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 size $O(k^m|\Sigma|^m)$ for each k-mer. It should be noted that in this case the total number of features considered by the algorithm can exceed the natural upper bound[4] of $|\Sigma|^k$. In our divide-and-conquer algorithm for mismatch kernel we take a different approach: our algorithm clusters unique 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.

[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 $|\Sigma|^k = 1024$, $N^2 nk^{m+1}|\Sigma|^m$ is 1.3e+10, $u(k^{m+1}|\Sigma|^m + N^2) =$ 2.2e+08 which is approximately 50 times less.

8 Experiments and Results

In our experiments we use three data sets of DNA barcodes. *Fish larvae*[5] dataset consists of 56 barcode sequences from 7 species. The *Astraptes.Fulgerator*[6] dataset contains 466 barcodes from 12 species. The number of sequences per species in the second dataset varies from as few as 3 barcodes to as many as 100 barcodes. Finally, a large *Hesperiidae* dataset has 2185 sequences and 355 classes.

8.1 Classification Performance

We evaluate performance of all methods using the ROC50 score [18] as well as the cross-validation error. In case of the Fisher kernel, profile HMM models are estimated from multiple sequence alignments for each sequence class. For classification, we use the existing implementation of SVM from a machine-learning package Spider [7]. In the experiments with feature selection, we preselect a small subset of k-mers before learning a classifier using filtering approach.

We compare the performance of full feature kernels and kernels based on the reduced feature set in Fig. 1 on the basis of the number of families with the score higher than corresponding ROC50 score. For the Fisher kernel, ROC50 plots clearly demonstrate that Fisher kernel with feature selection achieves higher performance compared with the full feature set kernel and performs as well as the mismatch kernel with feature selection. However, Fisher kernel offers position information and can therefore facilitate interpretation of the resulting model.

Table 2 displays performance results of all six methods on the *A.fulgerator* dataset. Similar validations are provided in Table 3 for the *fish larvae* dataset. The performance of the kernels with feature selection (SFK, SSK, SMK) on both datasets is, at worst, indistinguishable from that of full feature kernels (FK, SK, MK). 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).

Figure 2 shows performance on the *Hesperiidae* dataset. The string kernels with feature selection performed extremely well on this dataset obtaining perfect results in more than 94% test cases. Feature selection improved performance for the exact spectrum kernel, and demonstrated performance very similar to that of the full feature mismatch kernel, however the computation cost is significantly lower, as shown in Sec. 5. Average cross-validation error rates on the *Hesperiidae* dataset are $3.7 \cdot 10^{-4}$ for spectrum and $3.5 \cdot 10^{-4}$ for mismatch kernels.

The use of feature selection for 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 for the Fisher kernel not only improved performance, but also even more dramatically decreased the number of features (see Table 2).

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

[6] From Barcode of Life Data Systems (BOLD) collection *www.barcodinglife.org.* see [17] for detailed description of the dataset.

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

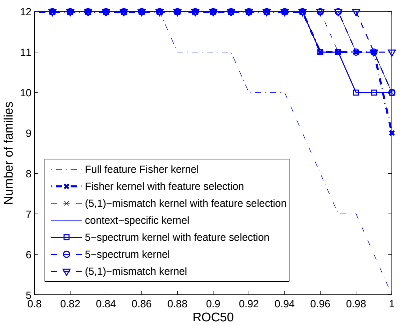

Fig. 1. Comparison of performance of the Fisher and string kernels with and without feature selection (*Astraptes* dataset). Note the positive role of the feature selection, especially in the case of Fisher kernels.

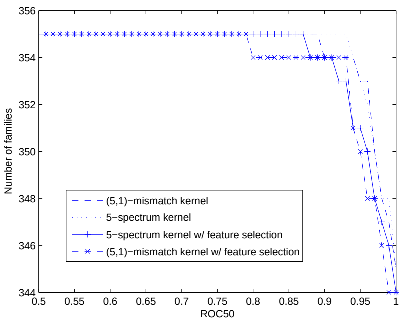

Fig. 2. Comparison of performance of string kernels with and without feature selection (*Hesperiidae* dataset). Reducing the number of features correlates mostly positively with the identification performance.

Table 2. Cross-validation error rates (%) for *A.fulgerator* species.

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

number of features for SSK and SMK 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 3. Cross-validation error rates (%) for *Fish larvae* species.

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

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

Table 4. 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	-

8.2 Comparison to Other Methods

We compared performance of the string kernel-based method using SVM with a number of other classification methods. In particular, we evaluated Fisher kernel method [10], PSI-BLAST, ridge regression and nearest neighbor methods. Table 4 displays classification performance results 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 [4, 5] (for example, on *Astraptes* dataset [17], the test error rate of multi-class SVM is only 0.67% compared to 9% in [5] or 20% in [4]).

8.3 Running Time Analysis

We performed running time analysis of the proposed algorithms to demonstrate their behavior 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), the 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 5) 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 a preprocessing step results in significant speed improvements, however the explicit map algorithm requires much larger storage (exponential in $|\Sigma|$ and k) than the divide-and-conquer algorithm.

Table 6 shows running times for mismatch kernel matrix computations with feature selection (filtering level in the table is a fraction of features filtered out). As we can see from the results, EM algorithms do not scale with the number of selected features, while divide-and-conquer approach scales almost linearly. Similarly to kernel matrix computations, extracting unique k-mers from support vectors using sorting ($O(Nkn)$ time) accelerates kernel vector computations for new sequences during testing. In mismatch kernel vector computations (Table 7), divide-and-conquer approach outperforms in many cases explicit mapping with sorting, especially for larger k and m (note that for the large k, EM algorithm exceeded memory capacity), which makes our algorithms also particularly suited for the fast identification of new sequences.

[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 5. 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

Table 6. Computation of mismatch kernel matrices with feature selection (time, s)

	Astraptes data			Hesperiidae data		
filt. 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

Table 7. Computation of the mismatch kernel vector (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	-

9 Conclusions

In this paper, we present a kernel classification based approach to the DNA barcoding problem that substantially improved identification accuracy compared to traditional approaches. We also present a framework for efficient computations of 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 improved classification performance. 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 and a divide-and-conquer technique for the string kernels 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 $O(Nkn + |F|N^2)$ time mismatch algorithm with feature selection for the kernel matrix computation.

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 not only implies more effective computations, but may 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. Jaakkola, T., Diekhans, M., Haussler, D.: A discriminative framework for detecting remote protein homologies. Journal of Computational Biology **7**(1-2) (2000) 95–114
11. 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
12. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press (2002)
13. Vapnik, V.: Statistical learning theory. Wiley (1998)
14. Vishwanathan, S.V.N., Smola, A.J.: Fast kernels for string and tree matching. In: NIPS. (2002) 569–576
15. 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
16. Leslie, C., Kuang, R.: Fast string kernels using inexact matching for protein sequences. J. Mach. Learn. Res. **5** (2004) 1435–1455
17. 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
18. Gribskov, M., Robinson, N.L.: Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. Computers & Chemistry **20**(1) (1996) 25–33