Spatial Representation for Efficient Sequence Classification

Pavel P. Kuksa, Vladimir Pavlovic
Department of Computer Science, Rutgers University
{pkuksa,vladimir}@cs.rutgers.edu

Abstract

We present a general, simple feature representation of sequences that allows efficient inexact matching, comparison and classification of sequential data. This approach, recently introduced for the problem of biological sequence classification, exploits a novel multi-scale representation of strings. The new representation leads to discovery of very efficient algorithms for string comparison, independent of the alphabet size. We show that these algorithms can be generalized to handle a wide gamut of sequence classification problems in diverse domains such as the music and text sequence classification. The presented algorithms offer low computational cost and highly scalable implementations across different application domains. The new method demonstrates order-of-magnitude running time improvements over existing state-of-the-art approaches while matching or exceeding their predictive accuracy.

1. Introduction

Analysis of large-scale sequential data has become an important task in machine learning and pattern recognition, inspired in part by numerous scientific and technological applications such as the document and text classification or the analysis of biological sequences. One particular way to analyze sequences is to consider them as *strings* of symbols (e.g., words, amino acid residues, etc.) In such framework, the goal may be to associate a class, or a label, with a string that has not been observed before, e.g., assign a topic such as sports or politics to a string of words (text document), or functional and structural annotations to a string of amino acids (protein sequences).

Classification of data in sequential domains is challenging because of the variability in the sequence lengths, absence of the readily available pattern/feature vectors (feature extraction problem), potential existence of important features on multiple scales, as well as the size of typical datasets. Discriminative learning methods that focus on capturing differences among classes of sequences have shown significant promise [1, 3, 6]. A typical *discriminative* setting requires that each sequence be represented using a fixed-length descriptor. Descriptors, such as the *bag-of-words* for text or the spectra of short string fragments for biological sequences have gained wide popularity and have led to increasingly accurate sequence classification algorithms.

Recently, [4] introduced a new set of biologically motivated features to model relationships between functionally similar but content distinct protein sequences. In this work, we generalize this approach and show that it leads to a general framework for accurate and scalable sequence representation. The features represent sequence data at multiple scales and capture local *fragment* dependencies within a string. This representation has two advantages: a) it models similarity of sequences under complex transformations and b) it allows efficient and scalable inexact sequence matching. We also extend the algorithms of [4] and show that they lead to general fast, alphabet-set independent families of algorithms for inexact matching of sequences, applicable to analysis of very large databases under both fully-supervised and semi-supervised settings.

We demonstrate that the developed approach is applicable to modeling of sequences in a wide range of domains, both discrete- and continuous-valued. Experiments using the new features and algorithms on text document categorization, protein classification, music and artist recognition show excellent predictive performance and significant improvements in running time over the existing state-of-the-art methods on these large alphabet, large sequence datasets.

2. Background

A number of state-of-the-art approaches to classification of sequences over finite alphabet Σ rely on fixed-length representations $\Phi(X)$ of sequences as the *spectra* ($|\Sigma|^k$-dimensional histogram) of counts of short substrings (k-mers), contained, possibly with up to m mismatches, in a sequence, c.f., spectrum/mismatch methods [5, 6, 3]. However, computing similarity scores, or kernels, $K(X,Y)$=$\Phi(X)^T\Phi(Y)$ using these representations can be challenging. E.g., efficient $O(k^{m+1}|\Sigma|^m(|X|+|Y|))$ trie-based mismatch kernel algorithm [6] strongly depends on the alphabet size and the number of mismatches m. On the other hand, the gapped [5] and subsequence [8] kernels have complexity independent of $|\Sigma|$, but quadratic in the sequence length (subsequence method) or show suboptimal performance compared to other methods (e.g., mismatch).

3. Spatial Representation of Sequences

In this section, we describe a general sequence classification framework that embeds the *observed* one-dimensional sequences into a richer, high-dimensional feature space where inexact string matching and comparison can be performed more efficiently.

Evaluation under spatial representation amounts to sampling the sequence features at different resolutions and comparing the resulting spectra; similar sequences will have similar spectra at one or more resolutions. Each sampled spatial feature consists of t substrings of length k, with each substring no more than d positions away from its neighbors. We illustrate the spatial features in Figure 1(a). The upper panel shows a typical contiguous spectrum 6-mer and the lower panel shows how a spatial sample method with k=2,t=3 would extract features from the string. Much like the spectrum features, the spatial feature "AR___ND__CQ" shown has the value proportional to the number of times it occurs in string X.

Spectrum/mismatch representations rely on *contiguous* string fragments of length k. On the other hand, the spatial representation is multi-dimensional, made of variably distanced string fragment combinations (Figure 1(b)). In the figure, we show a spatial embedding ($t = 2$) with string fragments as single symbols (k=1) displaced by d (e.g., row "AA" and column $d = 2$ shows number of occurrences of "A__A").

This multi-resolution representation allows to more efficiently compute sequence similarity under substitution, insertion, and deletion *without* explicitly inducing a mutational neighborhood as in mismatch/profile methods [6, 3]. It also captures *short-term dependencies* among the individual local k-mers by including their spatial configuration.

In Sec. 4, and 4.1, extending the methods in [4], we will develop two very efficient algorithms for computing spatial feature similarity (kernels) of the form

$$K^{(t,k,d)}(X,Y) = \sum_{a_i\in\Sigma^k, 0\le d_i<d} \begin{array}{l} C_X(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t)\cdot \\ \quad C_Y(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t) \end{array}, \quad (1)$$

where $C_X(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t)$ is the number of times the spatial feature $a_1 \overset{d_1}{\leftrightarrow} a_2 \overset{d_2}{\leftrightarrow},\cdots,\overset{d_{t-1}}{\longleftrightarrow} a_t$ (a_1 separated from a_2 by d_1 sequence elements, a_2 separated by d_2 elements from a_3, etc.) is observed in X. We refer to specific cases ($t = 2, k, d$) and ($t = 3, k, d$) as double(k,d) and triple(k,d).

4. Algorithms for Spatial Feature Similarity Evaluation

High-dimensionality of feature spaces for large $|\Sigma|$ necessitates development of efficient algorithms for evaluating feature-based similarity between pairs of strings. Sequence matching under spatial representation (Eq. 1) can be efficiently computed using sorting and counting (Algorithm 1) in linear time, and independent of the alphabet size, despite potentially very large feature space $O(d^{t-1}|\Sigma|^k)$. In the algorithm, spatial features (k, t, d) extracted from the input (step 3) are sorted in linear time using t rounds of counting sort (step 5) to group same features together so that feature counts for the input strings can be obtained in one pass over the sequence index associated with the features (step 6). Similar to the gapped [5] and subsequence [8] kernels, Algorithm 1 is independent of the alphabet set size $|\Sigma|$, however it is *linear* in the sequence length $O(c_{k,t,d}n)$, while the subsequence kernels are quadratic in the length of the sequence.

Algorithm 1: String Matching under Spatial Embedding

```
Input: set of strings $S = \{s_1,\dots,s_N\}$, parameters $k, t, d$ (in our
    applications, $k = 1$, $t \le 3$, $d \le 5$)
1: $\mathbf{K} = \mathbf{0}$
2: for all $(d_1,\dots,d_{t-1}), 0 \le d_i < d$ do
3:     Construct the complete set of features $L = \bigcup_{i=1}^{N} L_{s_i}$, $L_{s_i}$ is
       the set of spatial features $(k,t,d)$ from $s_i$, with each feature
       containing the index of its corresponding sequence
4:     Obtain a permutation $\pi_L$ that lexicographically orders $L$
       using $t$ rounds of counting sort
5:     Obtain counts $c(f, s_i), i = 1,\dots,N$ for each distinct
       feature $f$ in $L_{\pi_L}$ in a single pass over the list and update
       kernel $\mathbf{K} = \mathbf{K} + \mathbf{c}(f)\mathbf{c}(f)^T$, $\mathbf{c}(f) = \{c(f,s_i)\}_{i=1}^N$
6: end for
```

4.1. Efficient Semi-supervised Kernel Algorithm

Our spatial representation can also exploit the unlabeled data using the *sequence neighborhood* approach, which was shown to be effective for mismatch kernels in [11]. In that approach, sets $N(X)$ of unlabeled sequences neighboring X are used to obtain neighborhood kernel measure from smoothed features,

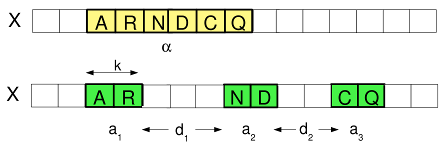
$$\Phi^{nbhd}(X)=\tfrac{1}{|N(X)|}\textstyle\sum_{X'\in N(X)}\Phi^{orig}(X'), \quad (2)$$

$$K^{nbhd}(X,Y)=\frac{\sum_{X'\in N(X),Y'\in N(Y)} K(X',Y')}{|N(X)||N(Y)|}. \quad (3)$$

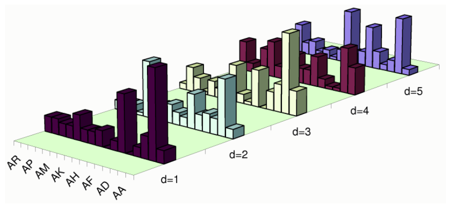
While Eq. 3 and Eq. 2 require either *quadratic* number of kernel evaluations or *explicit* feature representations $\Phi(X)$, we observe that Algorithm 1 can evaluate the neighborhood kernel more efficiently without requiring explicit representations $\Phi(X)$ since the kernel (Eq. 3) can be re-written similar to Eq. 1 as

$$K^{nbhd}(X,Y)= \sum C_{N(X)}(a_1,d_1,\ldots,d_{t-1},a_t)\cdot C_{N(Y)}(a_1,d1,\ldots,d_{t-1},a_t), \quad (4)$$

thus reducing the neighborhood kernel computation to that of computing kernel as in Algorithm 1 (with X replaced by a set of strings $N(X)$). Using the above algorithms allows to very efficiently compute kernels on large databases and very high-dimensional spaces (Sec. 5).



(a)



(b)

Figure 1. Left: Contiguous k-mer feature α of a traditional spectrum feature (top) contrasted with the spatial samples (bottom). Right: The multi-dimensional spatial sample embedding.

5 Experimental Results

We test proposed methods on four distinct multi-class sequence classification tasks[1]: (1) text document categorization, (2) music genre classification[2], (3) artist identification, and (4) multi-class protein fold prediction. We display results for best choice of d among $double(1,d)$ and $triple(1,d)$ (typically double(1,5) and triple(1,3)) and SVM classifier[3] with default parameters. Gapped/subsequence kernels use similar settings (e.g., k=3, at most g=4 gaps). In the results, top-n error or F1 evaluate whether the correct class is in the top n class predictions. We give running time comparisons in Section 5.5.

5.1. Text Classification

We use a benchmark Reuters-21578 data set[4] for the experiments on text categorization. The word alphabet after standard preprocessing contains $|\Sigma|$=29, 224 distinct words. The double(1,5) kernel in word-feature space (doubles of words displaced by d other words) improves over other state-of-the-art methods (n-gram, subsequence, TF-IDF word kernels, KSG [12]), tuned and designed specifically for text categorization (Table 1). KSG [12] often displays performance similar to our approach, but uses significantly more intricate grouping of substrings into key groups (based on, e.g., the maximum parent-child conditional probability in suffix tree decomposition, etc.)

Furthermore, it is important to note that even for the large alphabet set ($|\Sigma|$=29, 224) a 8986-by-8986 document similarity matrix for the double(1,5) kernel takes only 16 seconds to compute on a 2.8GHz CPU.

5.2. Music Genre Classification

Music genre recognition is a particularly interesting problem in our setting because music data is originally continuous-valued and string representations (in the absence of musical notation) may require a large alphabet.

On a standard benchmark dataset [7] (10 genres, each 100 audio sequences, quantized into strings with $|\Sigma| = 1024$) our method achieves better overall performance (Table 2, 10-fold cross-validation) compared to the gapped(4,2) kernel, and the DWCH method [7], an approach specifically developed for music classification. This is achieved using very simple MFCC features that capture only *local* information in the music signals. In contrast, the DWCH method uses more sophisticated features with both *local and global* information. Compared to the gapped kernel with no spatial information, our method achieves better performance in eight out of ten genres. Both of these facts point to importance of considering longer-term spatial relationships in music signals for genre prediction. Similar conclusion carries over to a multi-class setting, Table 3, The raw MFCC features achieve 41.6 ± 3.31 error rate [7]. Our double kernel that incorporates *longer*-term dependency (using d=5) improves the error significantly to 29.2 ± 1.61.

Table 1. Test F1 scores on top Reuters categories. Spatial features improve over baseline and state-of-the-art methods.

Class	TF-IDF	KSG	Double	SS-4[†]	NG-4[‡]
Earn	98.70	98.3	**98.76**	97.0	98.40
Acq	97.11	96.8	**97.68**	88.0	93.20
Money	77.61	**84.0**	83.71	76.0	75.70
Grain	93.29	92.5	**93.38**	84.0	84.0
Crude	87.71	89.4	**90.51**	84.0	84.80
Trade	84.26	90.2	**91.23**	73.0	77.90
Interest	71.80	**81.5**	81.15	66.0	71.90
Wheat	80.00	**81.8**	80.92	79.7	79.0
Ship	72.97	81.9	**84.39**	65.0	62.60
Corn	81.63	**87.1**	83.33	63.0	61.0
Macro-average	84.51	88.3	**88.51**	77.57	78.80
Micro-average	93.18	93.9	**94.39**	-	-

KSG=key-substring-group features [12]
[†]*approximate* subsequence kernel [8], [‡]N-gram character kernel [8]

Table 2. Music genre classification.

#	Genre	DWCH[†]	Double(1,5)	gapped(4,2)
1	Blues	95.49 (1.27)	93.6 (4.77)	93.8 (2.33)
2	Classical	98.89 (1.1)	95.6 (1.35)	97.2 (1.04)
3	Country	94.29 (2.49)	**94.3 (2.21)**	91.7 (2.02)
4	Disco	92.69 (2.54)	**94.3 (1.41)**	91.9 (1.14)
5	Jazz	97.9 (0.99)	95.5 (2.27)	93.4 (1.29)
6	Metal	95.29 (2.18)	94.7 (1.42)	94.0 (2.37)
7	Pop	95.8 (1.69)	**96.2 (1.75)**	95.5 (1.32)
8	Hiphop	96.49 (1.28)	**97.1 (0.99)**	94.8 (1.25)
9	Reggae	92.3 (2.49)	**95.5 (1.58)**	92.3 (2.02)
10	Rock	91.29 (2.96)	**95.1 (1.66)**	91.7 (1.52)
	Mean	95.04	**95.19**	93.63

[†]: DWCH=Daubechies Wavelet Coefficient Histograms [7]

Table 3. Multi-class music genre classification

method	Error	Top-2 Error	F1	Top-2 F1
gapped	34.5 ± 2.6	19.9 ± 2.27	65.35	80.31
double	29.2 ± 1.61	17.5 ± 1.77	70.82	82.62
triple	29.3 ± 1.86	17.3 ± 1.89	70.61	82.78

[1]Datasets and code used in our experiments are made available at `http://paul.rutgers.edu/~pkuksa/large-scale/large-scale.htm`

[2]http://opihi.cs.uvic.ca/sound/genres

[3]http://www.kyb.tuebingen.mpg.de/bs/people/spider/

[4]http://www.daviddlewis.com/resources/testcollections/reuters21578/

5.3. Artist recognition

We also illustrate the utility of our generic string features on multi-class artist identification on a standard *artist20* dataset[5] with 20 artists, 6 albums each. Table 4 lists results for 6-fold album-wise cross-validation with one album per artist held out for testing. Using spatial information with quantized MFCC features ($|\Sigma| = 1024$) yields 32.5% error compared to 44% using MFCC features alone [2], indicating that our spatial features may be well suited for this task, especially when coupled with more domain-specific information.

Table 4. Artist recognition performance

method	Error	Top-2 Error	F1	Top-2 F1
gapped	44.66	32.24	55.33	67.99
double	32.50	21.51	67.56	78.63
triple	32.69	21.15	67.43	78.67

5.4. Multi-class protein fold prediction

Protein fold prediction is a difficult task in biological sequence classification where one aims to predict a complex, high-level protein category, using only the primary sequence information. The fold recognition benefits from large unlabeled datasets that "link" otherwise only remotely similar sequences in the same fold c.f., [11]. We use the non-redundant (NR) sequence database with 534,936 unlabeled sequences to compute neighborhood kernel (see Sec. 4.1).

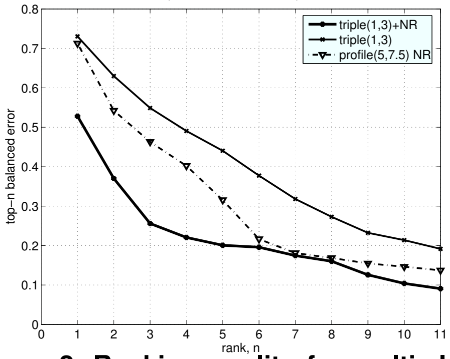


Figure 2. Ranking quality for multi-class protein remote fold recognition (top-n balanced error rates). Spatial kernels improve over the profile kernel.

On a benchmark multi-class fold recognition dataset [9] (26 folds), we observe significantly lower error rates (Figure 2) than the state-of-art profile kernel [9], e.g., top-3 error is almost 50% smaller. One reason for this may be the use of the multi-scale spatial representation, which matches what some biologists believe to be "critical position" fragments in proteins [10].

5.5. Running time

One important benefit of our approach lies in the computational efficiency of evaluating similarity of sequences (kernel values). As shown in Table 5, both double and triple kernels demonstrate order of magnitude running time improvements over other algorithms.

[5]http://labrosa.ee.columbia.edu/projects/artistid/

Table 5. Running time (s) for kernel computation between two strings on *real* data

	protein (semi-sup)	protein	text	music
n, $|\Sigma|$	36672, 20	116, 20	242, 29224	6892, 1024
(5,1)-mismatch	1.6268	2.82e-02	20398	526.8
subseq. (p=3)	1222.4	3.27e-02	0.4846	2.4321
double	0.1155	9e-04	3.6e-03	1.37e-02
triple	0.1967	2.5e-03	7.5e-03	3.45e-02

*n-sequence length, $|\Sigma|$-alphabet size

6. Conclusion

We present a general approach for sequence classification based on a multi-dimensional embedding that permits rapid and accurate sequence matching, comparison, and classification. The algorithms we propose are scalable (for large datasets and long sequences) with complexity independent of the alphabet size. We study our method in three different domains. It achieves state-of-the-art performance in text classification, offers significant improvements for music genre classification and artist identification, and as well as protein fold prediction. We also show that our method can readily leverage unlabeled data and work with very large databases. As a consequence, the same approach may be applicable to other challenging problems on large corpora, e.g., web document classification or literature mining.

References

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

[2] D. Ellis. Classifying music audio with timbral and chroma features. In *Proc. Int. Conf. on Music Information Retrieval ISMIR-07*, 2007.

[3] 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.

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

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

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

[7] 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.

[8] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002.

[9] 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.

[10] B. Reva, A. Kister, S. Topiol, and I. Gelfand. Determining the roles of different chain fragments in recognition of immunoglobulin fold. *Protein Eng.*, 15(1):13–19, 2002.

[11] 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.

[12] D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *KDD '06*, pages 474–483, New York, NY, USA, 2006. ACM.