Kernels Based on Weighted Levenshtein Distance

Jianhua XU
School of Mathematical and Computer Sciences
Nanjing Normal University
Nanjing 210097, China
E-mail: xujianhua@email.njnu.edu.cn

Xuegong ZHANG
Department of Automation
Tsinghua University
Beijing 10084, China
E-mail: zhangxg@mail.tsinghua.edu.cn

Abstract— **In some real world applications, the sample could be described as a string of symbols rather than a vector of real numbers. It is necessary to determine the similarity or dissimilarity of two strings in many training algorithms. The widely used notion of similarity of two strings with different lengths is the weighted Levenshtein distance (WLD), which implies the minimum total weights of single symbol insertions, deletions and substitutions required to transform one string into another. In order to incorporate prior knowledge of strings into kernels used in support vector machine and other kernel machines, we utilize variants of this distance to replace distance measure in the RBF and exponential kernels and inner product in polynomial and sigmoid kernels, and form a new class of string kernels: Levenshtein kernels in this paper. Combining our new kernels with support vector machine, the error rate and variance on UCI splice site recognition dataset over 20 run is 5.88$\pm$0.53, which is better than the best result 9.5$\pm$0.7 from other five training algorithms.**

I. INTRODUCTION

In the last ten years, support vector machine and other kernel machines have been paid more attention on in the fields of machine learning, pattern recognition and neural networks, due to their good generalization ability on many real world problems, e.g. handwritten digital recognition, text classification, bioinformatics, and etc [1-4]. These methods can utilize kernels satisfying Mercer condition to construct the nonlinear discriminant and regression functions in the original input space. Scholkopf and Smola [4] analyzed the kernels from different viewpoints and summarized that the choice of a kernel corresponds to choosing (a) a similarity measure for the data, (b) a linear representation of the data, (c) a function space for learning, (d) a regularization functional, (e) a covariance function for correlated observations, and (f) a prior over the set of functions. In short, the choice of kernel should reflect prior knowledge of problem being handled.

The polynomial, RBF, exponential and sigmoid kernels have been widely used in many applications where the training samples are described as the vectors of real numbers. Some useful mathematical tricks to construct more new kernels from old kernels are summarized in [3-5]. However, in order to improve the performances of kernel machines further, designing kernels based on prior knowledge about the problem at hand has become an attractive research field. Many studies have been done for the symbol data particularly, since it was found that converting symbols into integers might lose the intrinsic meaning of the problem. Haussler [6] and Watkins [7] introduced a general idea to construct kernels on discrete structures, such as strings, trees and graphs. Specially, Haussler presented the convolution kernel and its several special cases, e.g. string kernel etc. For text classification problem, Lodhi et al. [8] analyzed the string kernel in detail, proposed a dynamic programming to calculate it, and introduced its approximating kernel to reduce computation time further. For DNA start codon recognition, locality improved kernel was constructed through comparing the two sequences locally within a small window [9]. By incorporating prior knowledge into kernel through probabilistic model (HMM) of generative data, so-called Fisher kernel [10-12] and TOP kernel [13] were proposed. For DNA sequences, a comparison on IPData [14] was provided using locality improved kernel, Fisher kernel, and TOP kernel. It was demonstrated that the first one suits to the small sample sizes, and the others suit to the large sample sizes. For the two strings with the same lengths, a kernel based on Hamming distance was introduced and analyzed in detail [15], and was successfully applied to multimedia data analysis. For the two strings with different lengths, edit distance kernel was defined as the maximal length between two strings minus the minimum total number of insertions, deletions and substitutions required to convert one string into the other [15].

This work is supported by Natural Science Foundation of China, project No.60275007.

For determining the similarity of two strings with different lengths, the weighted Levenshtein (or edit) distance (WLD) is the widely used notion in the syntactic pattern recognition [16,17], which means the minimum weight summation of single symbol insertions, deletions and substitutions required to transform one string into the other. In WLD the different operations are evaluated with different weights, while in the edit (or Levenshtein) distance, the different operations are evaluated with the same weights (e.g. constant 1). Wagner and Fisher [18] designed a dynamic programming to calculate WLD, whose computational complexity is proportional to the product of two string lengths. In this paper, to fuse prior knowledge of strings into kernels effectively and deal with the string data directly, we use variants of WLD to replace distance term in the RBF and exponential kernels and inner product in the polynomial and sigmoid kernels, and design a new class of string kernels: Levenshtein kernels. In the experiment on UCI splice site recognition dataset for the binary classification problem, we combine our novel kernels with support vector machine, and obtain the error rate and variance 5.88 ± 0.53 over 20 run, which outperforms the best result 9.5 ± 0.7 from other five training algorithms [19].

II. Overview of Support Vector Machine and Kernels

Recently support vector machine (SVM) has been becoming a widely used supervised learning method. Let the given data set for the binary classification problem be $\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_i,y_i),\ldots,(\mathbf{x}_l,y_l)\}$, where $\mathbf{x}_i\in R^n$ and $y_i\in\{+1,-1\}$ $(i=1,\ldots,l)$ are the training samples and their corresponding labels, and l indicates the number of training samples. The construction of support vector machine corresponds to solving the following quadratic program:

$$\max_{\alpha_i} \sum_{i=1}^{l}\alpha_i-\tfrac{1}{2}\sum_{i,j=1}^{l}\alpha_i y_i\alpha_j y_j k(\mathbf{x}_i,\mathbf{x}_j), \qquad (1)$$

$$\text{s.t.} \sum_{i=1}^{l}\alpha_i y_i=0, \quad 0\le\alpha_i\le C,\ i=1,\ldots,l \qquad (2)$$

where $a_i(i=1,\ldots,l)$ are the coefficients to be solved, C is the regularization constant, and $k(\mathbf{x}_i,\mathbf{x}_j)$ is the kernel satisfying Mercer condition (i.e. positive semi-definite kernel) [1-4].

For some given input vector, SVM classifier assigns it to one of two classes according to the discriminant function:

$$f(\mathbf{x})=\operatorname{sign}(\sum_{i=1}^{l}\alpha_i y_i k(\mathbf{x},\mathbf{x}_i)+\beta), \qquad (3)$$

where β is the threshold or bias which could be calculated according to Kuhn-Tucker condition [1,2]. Due to the non-linearity of kernels, SVM implements the nonlinear classification in the original input space. The common kernels include:

$$k(\mathbf{x},\mathbf{y})=(\mathbf{x}\cdot\mathbf{y})^p \quad \text{(polynomial kernel)}, \qquad (4)$$

$$k(\mathbf{x},\mathbf{y})=e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2} \quad \text{(RBF kernel)}, \qquad (5)$$

$$k(\mathbf{x},\mathbf{y})=e^{-\gamma\|\mathbf{x}-\mathbf{y}\|} \quad \text{(exponential kernel)}, \qquad (6)$$

$$k(\mathbf{x},\mathbf{y})=\tanh(c(\mathbf{x}\cdot\mathbf{y})+d) \quad \text{(sigmoid kernel)}. \qquad (7)$$

In these kernels, p is the degree of polynomial kernel, γ is the width of RBF and exponential kernels, and c and d are two parameters for sigmoid kernel. Particularly $\mathbf{x}$ and $\mathbf{y}$ imply two training vectors with continuous valued components. When the training samples are described as strings of symbols, the usual strategy is to convert symbols into integers in a one-to-one mapping. However, this numeric representation may lose the intrinsic meaning of the problem. In this paper, we want to extend two vectors in these kernels to two strings of symbols with different lengths.

III. Weighted Levenshtein Distance

In order to cope with strings consisting of symbols in kernel machines directly, we consider the similarity or dissimilarity between two strings at first. Suppose the alphabet Σ is a non-empty set of symbols, which could include the letters and numbers. Let Σ^* be the set of all finite length strings of symbols from Σ. The notion $|a|$ denotes the length of some string a, i.e., its number of symbols. For the given two strings a and $\tilde{a}$, $\tilde{a}$ is a sub-string of a if and only if $\tilde{a}$ could be obtained by deleting some symbols from a.

For the correction operation of two strings a and b in Σ^*, there could exist the following three string-to-string correction rules:

(a). Single symbol deletion: $a=cxd$, $b=cd$, which means that the symbol x between c and d is deleted;

(b). Single symbol insertion: $a=cd$, $b=cxd$, which means that the symbol x is inserted between c and d;

(c). Single symbol substitution: $a=cxd$, $b=cyd$, which means that the symbol x is replaced by the symbol y.

Here $c,d\in\Sigma^*$ denote two sub-strings, $x,y\in\Sigma^*$ denote two different symbols (i.e., $x\neq y$).

If one of rules above holds true, the string a can be transformed into the string b in one step correction. The Levenshtein distance (or edit distance) is defined as the smallest number of correction operations converting the string a into the string b.

Since in many applications three correction operations indicate different meanings, it is necessary to determine different weights for the different rules. In this case, the weighted Levenshtein distances (or WLD) was introduced

[16,17], which is defined as the minimum total weight of single symbol insertions, deletion and substitutions required to convert one string into the other. If the weight of insertion is equal to that of deletion, WLD satisfies all three conditions for the distance definition [16].

Wagner and Fisher [18] proposed a dynamic programming algorithm for WLD, whose computational complexity is $O(|a||b|)$. Let a_i be the sub-string consisting of the first i symbols of string a where $0 \le i \le |a|$, and b_j be the sub-string consisting of the first j symbols of string b where $0 \le j \le |b|$. Assume that $D_{i,j}$ denotes WLD between two sub-strings a_i and b_j, where $D_{0,0} = 0$. Then WLD between a_i and b_j becomes

$$D_{i,j} = \min\ (D_{i-1,j} + w^{\mathrm{I}}, D_{i-1,j-1} + w^{\mathrm{S}}, D_{i,j-1} + w^{\mathrm{D}}), \qquad (8)$$

where $w^{\mathrm{I}}, w^{\mathrm{D}}$ and w^{S} are weights of insertion , deletion and substitution operations respectively. Obviously $\mathrm{WLD}(a,b) = D_{|a|,|b|}$. Fig.1 illustrates the computational procedure about WLD, where each path from (0,0) to $(|a|,|b|)$ (e.g., the solid line in Fig.1) corresponds to a sequence of corrections converting a = ACGC into b = ACGT . WLD is the optimal path minimizing the total weight summation (the dotted line in Fig.1).

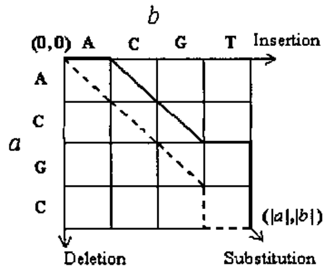

Fig.1. The Computational Procedure of WLD

In order to handle the application problems effectively in numerical computation, it is necessary to convert WLD into the interval between 0 and 1. In our experiment below, we define weights of both insertion and deletion operations as 1. For the substitution operation, if two symbols are identical, the weight is equal to zero; otherwise the weight equals 3. Thus WLD varies from 0 to $|a|+|b|$ (i.e. the length summation of two strings). The original WLD can be simply divided by $(|a|+|b|)$ to convert its value into the range from 0 to 1.

IV. Kernels Based on the Weighted Levenshtein Distance

The kernels play an important role in support vector machine and other kernel machines. The choice of kernel should reflect prior knowledge of problem being dealt with. In this section, we consider how to incorporate prior knowledge about strings into kernels.

The RBF and exponential kernels (i.e. (5) and (6)) are positive semi-definite for any width $\gamma \in (0,\infty)$ and for any dimension of input space, where $\|\mathbf{x}-\mathbf{y}\|$ indicates the distance between two vectors $\mathbf{x}$ and $\mathbf{y}$, e.g., Euclidean distance. If $\mathbf{x}$ and $\mathbf{y}$ become two strings with different lengths, such a distance measure between $\mathbf{x}$ and $\mathbf{y}$ can be directly replaced by weighted Levenshtein distance. The corresponding kernels still are positive semi-definite since WLD satisfies the definition of distance.

In the polynomial and sigmoid kernels (i.e. (4) and (7)), there exists inner product between two vectors, i.e. $(\mathbf{x}\cdot\mathbf{y})$, which reflects cosine of two vectors and implies similarity measure of two vectors. Inversely WLD denotes the dissimilarity measure of two strings. Here we define $1-\mathrm{WLD}(.,.)$ as the similarity measure of two strings and consider this quantity as inner product of two strings. Thus such a quantity is utilized to replace inner product in the polynomial and sigmoid kernels. For the polynomial kernel, when the degree p trends to infinite, its corresponding kernel matrix becomes identity matrix which is positive definite. Although we do not offer a theoretical proof of the Mercer validity for these two kernels, in our experiment the Mercer condition is satisfied in many cases since the corresponding kernel matrix is positive definite in the numerical computation.

Since the distance and inner product in the original kernels (4)-(7) are replaced by variants of weighted Levenshtein distance, we simply refer to these kernels based on WLD as Levenshtein kernels.

V. Experiment Results and Analysis

The dataset of splice site recognition originally comes from the UCI repository of machine learning databases (http:// www.ics.uci.edu/~mlearn/MLRepository.html). The task is to recognize two types of splice junctions in DNA sequences: exon/intron (EI) or intron/exon (IE) sites. A splice junction is a site in a DNA sequence at which uperflous DNA is removed during protein creation. Intron indicates the portion of the sequence spliced out while exon is the part of the sequence retained. The original data set includes 3175 samples, which have been classified into three classes (EI, IE and Neither). Each sample is described as a sequence of symbols (A, C, G and T), whose length is 60. In order to adapt some machine learning algorithms, Raetsch et al utilized 1,2, 3 and 4 to replace four letters (http://www.first.gmd.de/~raetsch). Furthermore he divided this set 20 times randomly, and constructed 20 training sets (1000 samples) and 20 test sets (2175 samples) of binary classifications. The performances from RBF Nets, AdaBoost, Regularized Adaboost, support vector machine and kernel Fisher Discriminant were provided in [19], as shown in Table I.

TABLE I
THE PERFORMANCES FROM DIFFERENT ALGORITHMS

Algorithms	Error rate &variance
SVM with WLD-exponential	**5.88±0.53**
SVM with WLD-polynomial	9.77±0.57
Regularized AdaBoost	9.5±0.7
RBF nets	10.0±1.0
AdaBoost	10.1±0.5
Kernel Fisher discriminant	10.5±0.6
Support vector machine	10.9±0.7
7NN	26.27±1.90
15NN with WLD	26.64±1.00

In our experiment, we do not convert the numbers (1,2,3 and 4) into symbols (A, C, G and T). Such four numbers are directly considered as four symbols and every sample as a string of 60 symbols. The quadratic programming software loqo of C language (http://www.kernel-machines.org/ softaware) is used to construct the support vector machine for classification. For the exponential kernel, when C=200 and $\gamma = 2.6$, the performance over 20 run is 5.88 ± 0.53 (SVM with WLD-exponential in Table I), which outperforms the best result from Regularized AdaBoost (9.5±0.7). Our kernel obviously improves the performance of classification in splice site recognition. For the polynomial kernel, when C=100 and $p = 3$, the error rate and variance become 9.77 ±0.57 (SVM with WLD-polynomial in Table I) which is very close to the best published result. We also provide the results from the 7 nearest neighbors with Euclidean distance and the 15 nearest neighbors with WLD in Table I. WLD can reduce the variance effectively

VI. Conclusions

In many applications, the sample could be described as a string of symbols rather than a vector with continuous value components. It is not always proper way to convert the symbolic data into integers or real numbers and to measure the similarity or dissimilarity using distances notion in the real space. In this paper, we directly consider the symbolic data of strings and use the weighted Levenshtein distance (WLD) as the similarity or dissimilarity of two strings. The variants of this distance are utilized to replace the distance measure and inner product in the four common kernels. Thus a new class of kernels based on WLD is constructed, which can be referred to as Levenshtein kernels. In the splice site recognition, the state-of-the-art performance has been achieved by combining SVM with our kernels. It is obviously demonstrated that incorporating prior knowledge about problems at hand into kernels can improve the performance of kernel machines effectively.

In this paper we only consider the strings with different single symbols. If a single symbol is replaced by a single word or a proper subsequence, our work still holds true. In addition, the new kernels can still be applied in other kernel machines for classification and clustering. Our further work will deal with more data sets of strings, save the computational time consuming for the long strings and etc.

References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer-Verlag, 1995.

[2] V. N. Vapnik, *Statistical Learning Theory*, New York: Wiley, 1998.

[3] N. Cristianini, and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge UK: Cambridge University Press, 2000.

[4] B. Scholkopf and A. J. Smola, *Learning with Kernels - Support Vector machines, Regularization, Optimization and Beyond*, Cambridge MA: MIT Press, 2002.

[5] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *Journal of Machine Learning Research*, vol. 2, pp. 299-312, Dec. 2001.

[6] D. Haussler, "Convolution kernels on discrete structure," *Technical Report UCSC-CRL-99-10*, Computer Science Department, University of California at Santa Cruz, 1999.

[7] C. Watkins, "Dynamic alignment kernels," In A. J. Smola, P. L. Bartlett, B. Scholkopf and D. Schuurmans (editors), *Advances in Large Margin Classifications*, pp. 39-50, Cambridge MA: MIT Press, 2000.

[8] H. Lodhi, C. Saunders, J. Shawe-Tayor, N. Cristianini and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, pp. 419-444, Feb. 2002.

[9] A. Zein, G. Raetsch, S. Mika, B. Scholkopf, T. Lengauer, and K-R Muller, "Engineering svm kernels that recognize translation initiation sites," *Bioinformatics*, vol. 16, pp. 799-807, Sep. 2000.

[10] T. Jaakkola, M. Diekhaus and D. Haussler, "Using the Fisher kernel method to detect remote protein homologies," *Proceedings of the 7th intelligent Systems for Molecular Biology*, Heidelberg, Germany, 1999, pp. 149-158.

[11] T. Jaakkola, M. Diekhaus and D. Haussler, "A discriminative framework for detecting remote protein homologies," *Journal of Computational Biology*, vol. 7, pp. 95-114, Feb/Apr. 2000.

[12] K. Tsuda, S. Akaho, M. Kawanabe, and K-R. Muller, "Asymptotic properties of the fisher kernel," *Neural Computation*, vol. 16, pp. 115-137, Jan. 2004.

[13] K. Tsuda, M. Kawanabe, Rätsch, G., S. Sonnenburg and K-R. Müller, "A new discriminative kernel from probabilistic models," *Neural Computation*, vol. 14, pp. 2397- 2414, Oct. 2002.

[14] S. Sonnenburg, G. Ratsch, A. Jagota and K-R. Muller, "New methods for splice site recognition," *Proceedings of the 2002 International Conference on Artificial Neural Networks*, Madrid, Spain, 2002, pp. 329-336.

[15] H. Aradhye and C. Dorai, "New kernels for analyzing multimodal data in multimedia using kernel machines," *Proceedings of 2002 IEEE International Conference on Multimedia and Exposition*, Lausanne, Switzerland, 2002, pp 37-40.

[16] K. S. Fu, *Syntactic Pattern Recognition and Application*, New Jersey: Prentice-Hall, Englewood Cliffs, 1982.

[17] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics-Doklady*, vol. 10, pp. 707-710, Aug. 1966.

[18] R. Wagner and M. Fisher. "The string-to-string correction problem," *Journal of the ACM*, vol. 21, pp. 168-178, Jan. 1974.

[19] S. Mika, G. Ratsch, J. Weston, B. Scholkopf and K-R. Muller, "Fisher discriminant analysis with kernels". *Proceedings of Neural Networks for Signal Processing IX*, Piscataway, NJ, USA, 1999, pp. 41-48.