A probabilistic framework for mismatch and profile string kernels

Alexei Vinokourov[1], Andrei N. Soklakov[2] and Craig Saunders[1] *

1- University of Southampton - School of Electronics and Computer Science
Southampton, Hants, SO17 1BJ - UK

2- Royal Holloway, University of London - Department of Mathematics
Egham, Surrey, TW20 0EX - UK

Abstract. There has recently been numerous applications of kernel methods in the field of bioinformatics. In particular, the problem of protein homology has served as a benchmark for the performance of many new kernels which operate directly on strings (such as amino-acid sequences). Several new kernels have been developed and successfully applied to this type of data, including spectrum, string, mismatch, and profile kernels. In this paper we introduce a general probabilistic framework for string-type kernels which uses the fisher-kernel approach and includes spectrum, mismatch and profile kernels, among others, as special cases. The use of a probabilistic model however provides additional flexibility both in definition and for the re-weighting of features through feature selection methods, prior knowledge or semi-supervised approaches which use data repositories such as BLAST. We give details of the framework, place well-known kernels in the framework and give preliminary experimental results which show some effects of using the probabilistic approach.

1 Introduction

In this paper we focus on the protein homology problem, which has become a benchmark for the application of string-type kernels. The model we present, however, has a wider range of applications to other sequence data, as the probabilistic framework allows for many tailored kernels to be produced, each with a clear method for introducing prior knowledge. In particular, the profile-based kernel [4] and mismatch kernel [5] have been shown to achieve state of the art performance on the protein sequence homology detection problem, while retaining efficiency due to possibility of implementation through a modification of fast string kernel algorithm. In this work both of this kernels are obtained from our attempt to derive an efficient kernel from a generative probabilistic model under very general assumptions about input sequences.

We first give some general definitions that will be useful later on. Let α be a string of symbols from a fixed alphabet $\mathcal{A}$, i.e. $\alpha \in \mathcal{A}^*$. In what follows we will use the notation $\alpha[i]$ to mean the ith symbol in α ($i = 1, 2, \ldots, |\alpha|$). Furthermore,

*The first and third author thank the EPSRC for their support through grant no GR/S22301/01 (”Development and Application of String-Type Kernels”)

by $\alpha[i:j]$, where $j > i$, we will mean a section of the string beginning with the ith symbol and ending with the jth symbol: $\alpha[i:j] = \alpha[i]\alpha[i+1]\ldots\alpha[j]$. Let $\Phi(\alpha)$ be a feature vector with components $\phi_\beta(\alpha)$. The corresponding kernel is then defined as a dot product in the feature space

$$k(\alpha_1, \alpha_2) \triangleq \sum_\beta \phi_\beta(\alpha_1)\phi_\beta(\alpha_2)\,. \tag{1}$$

Our statistical methods will be based on counts of fixed length substrings $\beta \in \mathcal{A}^k$ of strings $\alpha \in \mathcal{A}^{|\alpha|}$ denoted as $\#(\beta|\alpha)$, when the fact that string β occurred in string α will be denoted as $\beta \sqsubseteq \alpha$. Wherever square brackets surround a boolean expression they will mean an indicator function: $[A] = 1$ if A is true and 0 otherwise.

We shall mostly be concerned with 'all contiguous substrings' type of kernels for which the feature mapping $\phi_\beta(\alpha)$ is simply

$$\phi_\beta(\alpha) = \#(\beta|\alpha) \triangleq \sum_{i=1}^{|\alpha|-k} [\alpha[i:i+k-1] = \beta]\,. \tag{2}$$

The *Fisher kernel* [3] for a generative model $P(\alpha|\Theta)$, $\alpha \in \mathcal{A}^*$, with parameters $\Theta = \{\theta_\beta\}_\beta$ is a kernel defined by the following mapping:

$$\phi_\beta^{Fisher}(\alpha) \triangleq \frac{\partial \log P(\alpha|\Theta)}{\partial \theta_\beta}\,,$$

where a common assumption of approximation of the Fisher information matrix by identity matrix is used, sometimes referred to as the naïve Fisher kernel.

A *mismatch kernel* [5] is a step further to account for possible mutations in input strings. A (k,m)-mismatch neighbourhood of a k-length string α is denoted $N_{(k,m)}(\alpha)$ and is a set of all such k-length strings that differ from α in no more than k symbols. The mismatch kernel feature mapping of a string α is then defined in the following way:

$$\phi_\beta^{mismatch}(\alpha) \triangleq \sum_{i=1}^{|\alpha|-k+1} [\beta \in N_{(k,m)}(\alpha[i:i+k-1])]. \tag{3}$$

This kernel can be efficiently computed, see [5] for details.

A *string profile* $\mathbf{P}(\alpha)$ of a string α is a sequence of distributions $p_i(a)$ over the alphabet $\mathcal{A}$: $\mathbf{P}(\alpha) = \left\{p_i(a) : a \in \mathcal{A},\ i = 1\ldots|\alpha|,\ \sum_{a\in\mathcal{A}} p_i(a) = 1,\ p_i(a) \geqslant 0\right\}$. A k-length profile segment at position i is then simply $\mathbf{P}(\alpha[i:i+k-1])$. A profile can be obtained, for example, as a result of running a commonly known PSI-BLAST program [1]. One can define a neighbourhood similar to a (k,m)-mismatch neighbourhood but in a 'profile sense' as follows. Let $\mathbf{P}(\alpha)$ be a profile defined over a string α. A *profile neighbourhood* $PN_{(k,\sigma)}(\mathbf{P}(\alpha[i:i+k-1]))$, $i = 1,\ldots,|\alpha|-k$, is a set of k-length strings which differ from $\alpha[i+1:i+k]$

with a log-probability not greater than σ:

$$PN_{(k,\sigma)}(\mathbf{P}(\alpha[i:i+k-1])) = \Big\{\beta \in \mathcal{A}^k : -\sum_{j=1}^{k} \log p_{i+j-1}(\beta[j]) < \sigma\Big\},$$

A *profile kernel* [4] is then defined by the feature vector

$$\phi_\beta^{profile}(\alpha) \triangleq \sum_{i=1}^{|\alpha|-k} [\beta \in N^p_{(k,\sigma)}(\mathbf{P}(\alpha[i:i+k-1])]\,. \tag{4}$$

This can be implemented by a minor modification of the fast string kernel algorithm leading to the same computational complexity. At the same time it incorporates a widely tested PSI-BLAST model. It has been also reported experimentally superior to the mismatch kernel [4]. We shall rewrite (4) in probabilistic terms. Since we are given a profile $\mathbf{P}(\alpha[i:i+k-1])$ it would be natural to weight each component with the corresponding probability:

$$\phi_\beta^{weight-profile}(\alpha) \triangleq \sum_{i=1}^{|\alpha|-k} [\beta \in PN_{(k,\sigma)}(\mathbf{P}(\alpha[i:i+k-1]))] P_\alpha(\beta \,@\, i)\,, \tag{5}$$

where $P_\alpha(\beta \,@\, i) \triangleq \prod_{j=1}^{k} p_{i+j-1}(\beta[j])$.

2 The Model

Let $\mathcal{M}_\alpha$ be the set of all possible mutations of string α. For every mutation $\mu \in \mathcal{M}_\alpha$ we assume to know its probability $P_{\mu|\alpha}$ and the effect it has on the original string $\alpha \to \mu(\alpha)$. The effect and the importance of each mutation can be visualized using the concept of an extended string D as follows. Let $D = \alpha_1, \alpha_2, \ldots, \alpha_N$ be a sequence of strings obtained from the original string α by drawing randomly a sequence of mutually independent mutations $\mu_1, \mu_2, \ldots, \mu_N \in \mathcal{M}_\alpha$ according to the distribution $P_{\mu|\alpha}$ and defining $\alpha_i = \mu_i(\alpha)$, $i = 1, 2, \ldots, N$. By definition, D contains the entire ensemble of strings generated from the original α by mutations. We can estimate the probability of an element in D as the average

$$P_N(D) = \left(\prod_{i=1}^{N} P(\alpha_i)\right)^{1/N} , \tag{6}$$

where the values of $P(\alpha_i)$ are given by a k-stage Markov model. Let $B_\alpha = \beta_1, \beta_2, \ldots, \beta_{|\alpha|-k+1}$ be a sequence of all contiguous k-length substrings generated from α, i.e. $\beta_j = \alpha[j : j+k-1]$. Then we have according to the model $P(\alpha) = \prod_{\beta \in B_\alpha} p_\beta$. Substituting this into (6) and taking the logarithm gives

$$\ln P_N(D) = \frac{1}{N} \sum_{i=1}^{N} \sum_{\beta \in B_{\alpha_i}} \ln p_\beta. \tag{7}$$

In order to calculate derivatives correctly we now parameterize our model using arbitrary real numbers τ such that $p_\beta = \frac{\tau_\beta}{\sum_\beta \tau_\beta}$. This gives

$$\frac{\partial \ln P_N(D)}{\partial p_\beta} = \frac{1}{N} \sum_{i=1}^{N} \frac{\#(\beta|\alpha_i)}{\tau_\beta} - \frac{1}{N \sum_\beta \tau_\beta} \sum_{i=1}^{N} |\alpha_i| - k + 1. \tag{8}$$

If all documents are of equal length, then the second term in the equation is constant. We can easily extend a document by a set of extra symbols that do not appear in any corpus so that all documents are of equal length. We can therefore ignore the second term and just use the first one directly. In the following we use p_β rather than τ_β as the two models can be made equivalent. Let $\#(\gamma|D)$ be the number of times the string γ appears in D, and let $\mathcal{D}$ be the set of all (different) strings that constitute D. Then one can find that

$$\frac{\partial \ln P_N(D)}{\partial p_\beta} = \frac{1}{N} \sum_{\gamma \in \mathcal{D}} \frac{\#(\beta|\gamma)\ \#(\gamma|D)}{p_\beta} \ . \tag{9}$$

For large values of N one can replace the ratio $\#(\gamma|D)/N$ by the probability $P_{\mu|\alpha}$ of the mutation that corresponds to γ, i.e. $\gamma = \mu(\alpha)$. Similarly, for large N, the set $\mathcal{D}$ contains strings resulted from all possible mutations $\mathcal{M}_d$, and therefore

$$\frac{\partial \ln P_N(D)}{\partial p_\beta} \overset{N\to\infty}{\longrightarrow} \phi_\beta^{profker-fisher}(\alpha) = \frac{1}{p_\beta} \sum_{\mu \in \mathcal{M}_d} \#(\beta|\mu(\alpha))\ P_{\mu|\alpha} \ . \tag{10}$$

Equation (10) defines a kernel which, within our model, accounts for all possible mutations $\{\mathcal{M}_\alpha\}$ of the original strings $\{\alpha\}$. However, the direct use of (10) demands large computational resources: in the most general case one has no alternative apart from using Monte-Carlo sampling over all possible mutations. Let us now develop approximations that result in a much more efficient algorithm. As a byproduct of our analysis we derive the profile kernel.

Let $P_\alpha(\beta\ @\ i)$ be the probability of finding β as a substring of $\mu(\alpha)$ at the ith position. Then, the probability, $P_\alpha(\beta)$, that β was found in $\mu(\alpha)$ regardless of the position can be calculated as

$$\begin{aligned} P_\alpha(\beta) = P_\alpha(\beta\ @\ 1) + P_\alpha(\bar\beta\ @\ 1)P_\alpha(\beta\ @\ 2) + P_\alpha(\bar\beta\ @\ 1,2)P_\alpha(\beta\ @\ 3) \\ + \ldots + P_\alpha(\bar\beta\ @\ 1,2,\ldots,(|\alpha|-k))P_\alpha(\beta\ @(\ |\alpha|-k+1))\ , \end{aligned} \tag{11}$$

where $P_\alpha(\bar\beta\ @\ r,s,t,\ldots)$ denotes the probability that β was not found as a substring of $\mu(\alpha)$ at any of the positions $r,s,t,\ldots$. In (11) we have written out a decomposition of $P_\alpha(\beta)$ starting with the position 1. It is clear that one can write similar expressions for $P_\alpha(\beta)$ starting with any position i, i.e., $P_\alpha(\beta) = P_\alpha(\beta\ @\ i) + P_\alpha(\bar\beta\ @\ i)P_\alpha(\beta\ @(\ i+1)) + \ldots$, where we assume that after the position $|\alpha| - k$ we proceed with the position $1, 2, \ldots$ to go through all the positions as in (11). For $P_\alpha(\beta)$ we have $|\alpha| - k$ possible decompositions of this type, and since all such decompositions are equivalent we can write

$$P_\alpha(\beta) = \frac{1}{|\alpha| - k} \sum_{i=1}^{|\alpha|-k} \Big(P_\alpha(\beta\ @\ i) + P_\alpha(\bar\beta\ @\ i)P_\alpha(\beta\ @\ (i+1)) + \ldots \Big)\ . \tag{12}$$

It is easy to see that the component sums in (12) are decreasing.

Let N^k_α be the total number of different strings of length k that can be derived as substrings of all possible versions $\{\mu(\alpha)\}_{\mu\in\mathcal{M}_\alpha}$ of the original string α. Then the expected number of times that a string β appears as a substring in $\mu(\alpha)$ is $\sum_{\mu\in\mathcal{M}_\alpha}\#(\beta|\mu(\alpha))\,P_{\mu|\alpha} = P_\alpha(\beta)N^k_\alpha$. Substituting this into (10) we obtain $\phi^{profker-fisher}_\beta(\alpha) = \frac{N^k_\alpha}{p_\beta}P_\alpha(\beta)$. Ignoring the higher order terms in (12) we thus obtain from (10)

$$\phi^{profker-fisher}_\beta(\alpha) = \frac{N^k_\alpha}{p_\beta}P_\alpha(\beta) \approx \frac{N^k_\alpha}{(|\alpha|-k)\,p_\beta}\sum_{i=1}^{|\alpha|-k} P_\alpha(\beta\,@\,i)\,. \qquad (13)$$

One can ignore small terms in the sum above by introducing a threshold σ:

$$\phi^{profker-fisher}_\beta(\alpha) = \frac{N^k_\alpha}{(|\alpha|-k)\,p_\beta}\sum_{i=1}^{|\alpha|-k}[\beta\in N_{(k,\sigma)}(P(\alpha\,@\,i))]P_\alpha(\beta\,@\,i)\,. \qquad (14)$$

Apart from the prefactor this coincides with the weighted profile kernel (5). It can be observed also that the mismatch kernel (3) can be recovered from (14) by setting all profiles $P_\alpha(\beta[j]\,@\,i)$ to a constant value, in which case profile neighbourhoods $PN_{(k,\sigma)}$ turn into mismatch neighbourhoods $N_{(k,m)}$.

3 Experiments

The experimental setting was similar to one described in [6] [5] and [4] for SCOP dataset. For the mismatch kernel (`plain-mism`) the parameter setting was ($k = 5, m = 1$) (for the spectrum kernel [6] $k = 5$ too) and for the profile kernel (`profkernel`) - ($k = 4$ and $\sigma = 6$). To test Monte Carlo approach to (10) we have generated $N = 20$ mutations of each training example. The FSA model was trained as described in [7] with the Markov process memory $k = 3$ to get parameter estimates p_β that were also used in kernel (14) (`profker-fisher`). The extended examples were subsequently used in both spectrum (`ext-spectr`) and mismatch (`ext-mism`) kernels. We noticed that results are much improved if one adds a 0.5 prior to the probability of 'non-mutation'. Higher values did not affect much the output.

For our experiments we chose a subset collected by Liao et al. [2] which has been described as lacking positive training examples and therefore more challenging. We plotted the number of protein families with performance above given ROC score vs. ROC score each method in Figure 1. One can observe that even the first approximation of (12) achieves the state-of-the-art performance of the profile kernel.

4 Conclusions

We started with an N-th rank Monte Carlo approximation of a Markov model with possible mutations defined by a BLAST profile and obtained (10) as a limit

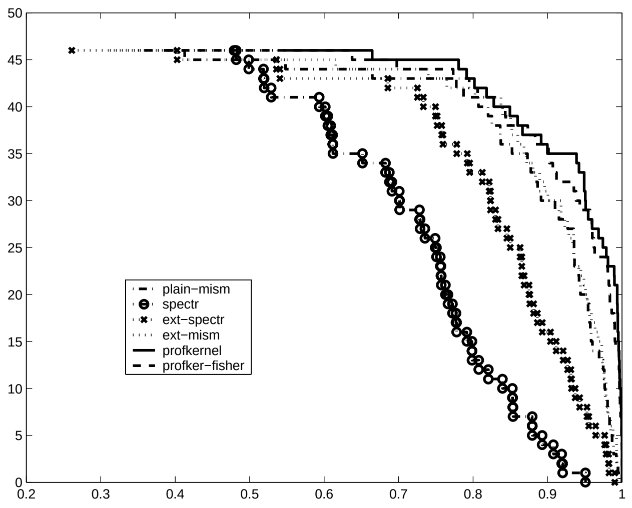

Fig. 1: The number of protein families with performance above given ROC score vs. ROC score for each method.

case $N \to \infty$. Having made rather weak assumptions we obtained a computable version of (10) given by formula (14) which includes state of the art performing profile kernel as well as mismatch and spectrum kernels as special cases. This result opens wide prospectives for further theoretical analysis of all-contiguous-substrings type of kernels along with possible modifications and extensions.

References

[1] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

[2] C.Liao and W.C. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, 2002.

[3] T. Jaakkola, M. Diekhaus, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. *Journal of Computational Biology*, 7(1,2):95–114, 2000.

[4] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina S. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *3rd International IEEE Computer Society Computational Systems Bioinformatics Conference (CSB 2004)*, pages 152–160. IEEE Computer Society, 2004.

[5] Christina Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–76, 2004.

[6] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[7] Craig Saunders, John Shawe-Taylor, and Alexei Vinokourov. String kernels, fisher kernels and finite state automata. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances of Neural Information Processing Systems 15*. MIT Press, 2003.