ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

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 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 and also give preliminary experimental results which show the applicability of the technique.

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 we derive as special cases of the model, the profile [2] and mismatch kernels [3] which have been shown to achieve state of the art performance on the protein homology problem whilst retaining computational efficiency due to their efficient implementation using fast string matching algorithms such as suffix trees.

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 ($i = 1, 2, \ldots, |\alpha|$) and by $\alpha[i:j]$, where $j > i$, a section of the string beginning with the ith symbol and ending with the jth symbol. Let $\Phi(\alpha)$ be a feature vector with components

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

$\phi_\beta(\alpha)$. The corresponding kernel is then defined as a dot product in the feature space

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

Features in these kernels are based on counts of fixed length substrings $\beta \in \mathcal{A}^k$ of strings $\alpha \in \mathcal{A}^{|\alpha|}$ denoted as $\#(\beta|\alpha)$. Wherever $\lceil\rceil$ surround a boolean expression they will mean an indicator function: $\lceil A \rceil = 1$ if A is true and 0 otherwise.

We shall mostly be concerned with kernels where individual features correspond to contiguous substrings of a fixed length k. One such elementary feature mapping is given by

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

which is often referred to as the ngram or spectrum kernel.

A *mismatch kernel* [3] 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 m symbols. The mismatch kernel feature mapping of a string α is then defined in the following way:

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

Despite the large number of features this kernel can be efficiently computed, see [3] for details.

A *profile* $\mathbf{P}(\alpha)$ of a string α is given by $\mathbf{P}(\alpha) = \{p_i(a) : a \in \mathcal{A}\}_{i=1}^{|\alpha|}$, where $p_i(a)$ is the probability of observing symbol a at position i, and $\sum_{a\in\mathcal{A}} p_i(a) = 1$. In the application domain of protein sequences, profiles can be obtained for example by using tools such as PSI-BLAST. A k-length profile segment at position i is then simply $\mathbf{P}(\alpha[i:i+k-1])$. One can define a neighbourhood similar to a (k,m)-mismatch neighbourhood but in a 'profile sense': A *profile neighbourhood* $PN_{(k,\sigma)}(\mathbf{P}(\alpha[i:i+k-1]))$, $i = 1,\ldots,|\alpha|-k+1$, is a set of k-length strings which differ from $\alpha[i:i+k-1]$ with a log-probability not greater than σ:

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

The *profile kernel* [2] is then defined by the feature vector

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

Using the profile kernel in conjunction with PSI-BLAST has been shown to be experimentally superior to the mismatch kernel [2]. In order to aid future comparisons we shall rewrite (4) in a slightly more general way. 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:

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

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

2 A general model

The *Fisher kernel* [1] for a generative model $P(\alpha|\Theta)$, with parameters $\Theta = \{\theta_\beta\}$ is a kernel defined by $k(x,y) = \phi(x)' F^{-1} \phi(y)$ where F is the Fisher information matrix and $\phi(\cdot)$ is the Fisher score vector for the example, which in our case is defined as the following mapping:

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

Due to the difficulty of computing Fisher Information matrix, the naïve Fisher kernel is often used where F is replaced by the identity matrix. Details on Fisher kernels are omitted due to lack of space, please see [1] for details.

For our model we consider extending our original sequence by a series of mutations, motivated by the observation that the similarity of two sequences could be measured by the similarity of mutations they produce. 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 importance of each mutation can be visualized using the concept of a bag of mutated strings D as follows. Let $D = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ be a bag of mutations of the original string obtained by drawing randomly a sequence of mutually independent mutations $\mu_1, \mu_2, \ldots, \mu_N \in \mathcal{M}_\alpha$ according to the distribution $P_{\mu|\alpha}$ (where $\alpha_i = \mu_i(\alpha),\ i = 1, 2, \ldots, N$). We define the probability of D as the average

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

Let us assume that $P(\alpha_i)$ are given by a k-stage Markov model, then by letting B_α be the set of all k-length substrings β in the sequence α we can show that under this assumption $P(\alpha) = \prod_{\beta \in B_\alpha} p_\beta$, where p_β is the probability of observing a substring β (details omitted due to lack of space). Substituting this into (6) and taking the natural logarithm gives

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

In order to calculate derivatives correctly, we follow [1] and parameterize our model using arbitrary real numbers τ_β; that is we use parameters τ_β 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 strings are of equal length, then the second term in the equation is constant and therefore does not need to be considered. This condition can easily be satisfied by inserting 'dummy' symbols at the start of shorter sequences. 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 enough N, the set $\mathcal{D}$ contains strings resulted from almost all possible mutations $\mathcal{M}_d$, and therefore

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

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 and mismatch kernels.

Let $P_\alpha(\beta \,@\, r, s, t, \ldots | u, v, w, \ldots)$ be the probability of finding β as a substring of $\mu(\alpha)$ at any of the positions $r, s, t, \ldots$ given that it was not found at $u, v, w \ldots$. Then, the probability, $P_\alpha(\beta)$, that β was found in $\mu(\alpha)$ regardless of the position can be calculated as

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

where $\bar{\beta}$ denotes the set of strings that are different from β. 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., $P_\alpha(\beta) = P_\alpha(\beta \,@\, i) + P_\alpha(\bar{\beta} \,@\, i\,|\,\beta \,@\, (i+1)) P_\alpha(\beta \,@\, (i+1)) + \ldots$, where we assume that after the position $|\alpha| - k + 1$ we proceed with the position $1, 2, \ldots$ to go through all the positions as in (11). We have $|\alpha| - k + 1$ of such decompositions, and since all of them are equivalent we can write $P_\alpha(\beta)$ as the average

$$P_\alpha(\beta) = \frac{1}{|\alpha| - k + 1} \sum_{r=1}^{|\alpha|-k+1} \left(\sum_{i=1}^{|\alpha|-k+1} P_\alpha(\beta \,@\, i)\ \pi_\alpha^{i,r} \right), \tag{12}$$

where $\pi_\alpha^{i,r} \leq \pi_\alpha^{i,r-1} \leq \cdots \leq \pi_\alpha^{i,1} = 1$.

Let N_α^k 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 $\sum_{\mu\in\mathcal{M}_\alpha} \#(\beta|\mu(\alpha))\, P_{\mu|\alpha} = P_\alpha(\beta) N_\alpha^k$. Substituting this into (10) we obtain $\phi_\beta^{profker-fisher}(\alpha) = \frac{N_\alpha^k}{p_\beta} P_\alpha(\beta)$. Ignoring the higher order terms in (12) we thus obtain from (10)

$$\phi_\beta^{profker-fisher}(\alpha) = \frac{N_\alpha^k}{p_\beta} P_\alpha(\beta) \approx \frac{N_\alpha^k}{(|\alpha| - k + 1)\, p_\beta} \sum_{i=1}^{|\alpha|-k+1} P_\alpha(\beta \,@\, i)\,. \qquad (13)$$

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

$$\phi_\beta^{profker-fisher}(\alpha) \approx \frac{N_\alpha^k}{(|\alpha| - k + 1)\, p_\beta} \sum_{i=1}^{|\alpha|-k+1} \lceil \beta \in N_{(k,\sigma)}(P(\alpha \,@\, i)) \rceil P_\alpha(\beta \,@\, i)\,. \qquad (14)$$

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

In order to show the applicability of the model we performed preliminary experiments using a similar setting to that used in [4, 3, 2] for the SCOP dataset. For our experiments we chose a subset collected by Liao et al. [5] which has been described as lacking positive training examples and therefore more challenging. As is standard we plotted the number of protein families with performance above given ROC score vs. ROC score for each method.

For the spectrum and mismatch kernels we used the parameters $k = 5$ and $m = 1$ which have been shown to yield good performance. For the profile kernel (profker) we used $(k = 4, \sigma = 6)$. We then compared these standard kernels to the Monte Carlo approach to (10), where we have generated $N = 20$ mutations of each training example and then applied the spectrum and mistmatch kernels on the extended mutated sequences (denoted ext-spectr and ext-mism respectively). In order to obtain the parameter estimates p_β needed for (10) we simply counted the frequency of k-length subsequences in the training examples in order to get a estimate. The probabilites $P_\alpha(\beta \,@\, i)$ were obtained using PSI-BLAST. In all cases, we added a prior of 0.5 for the probability of 'non-mutation'; higher values than this did not affect results, but setting this prior too low causes too many random mutations and loses the information in the original sequence. Finally we also give results for the first-term approximation to the profker-fisher kernel, with p_β generated as above. The results can be seen in Figure 1. One can observe that even the first approximation of (12) achieves the state-of-the-art performance of the profile kernel.

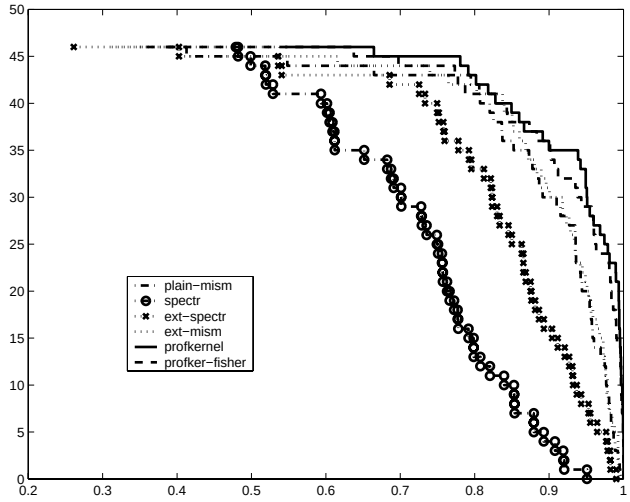

Fig. 1: The number of protein families with performance above given ROC score vs. ROC score for each method.

4 Conclusions

We have presented a general probabilistic model for sequences, based on the idea of extending the original sequence by series of mutations obtained from a general probabilistic model. By analysing the limit case $N \to \infty$ of Nth-rank Monte-Carlo approximations to the model, we obtain a very general feature mapping; although one that is difficult to compute. However we show that by making rather weak assumptions one can obtain a computable version of the mapping (14) which includes state of the art performing profile kernel as well as mismatch and spectrum kernels as special cases. This framework provides a basis further theoretical analysis of string kernels along with possible modifications and extensions.

References

[1] 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.

[2] R. Kuang, E. Ie, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *3rd International IEEE Computer Society Computational Systems Bioinformatics Conference*, pages 152–160, 2004.

[3] C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–76, 2004.

[4] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[5] 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.