Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings

Jean-Philippe VERT

Jean-Philippe.Vert@mines.org

September 18, 2001

To appear in the Proceedings of the Pacific Symposium on Biocomputing 2002

Abstract

A new class of kernels for strings is introduced. These kernels can be used by any kernel-based data analysis method, including support vector machines (SVM). They are derived from probabilistic models to integrate biologically relevant information. We show how to compute the kernels corresponding to several classical probabilistic models, and illustrate their use by building a SVM for the problem of predicting the cleavage site of signal peptides from the amino-acid sequence of a

protein. At a given rate of false positive this method retrieves up to 47% more true positives than the classical weight matrix method.

1 Introduction

Probabilistic models for strings play a central role in computational biology nowadays[1]: they include weight matrix for short signal sequences, Markov or hidden Markov models for DNA and protein sequences, stochastic context-free grammars for RNA or Bayesian graphical models for gene expression data[2]. Their strength comes from their ability to catch subtle regularities in the sequences and to quantify various biological features in a sound theoretical framework. They can integrate valuable biological knowledge (such as the possibility of mutations, insertions or deletions in hidden Markov models) difficult to handle otherwise.

A classical use of probabilistic models is to combine them with Bayes rule to classify sequences into one out of several competing categories. However experimental evidences in other areas as well as in computational biology[3] suggest that the resulting classifier can perform poorly compared to other discriminative methods, such as support vector machines (SVM).

There is therefore an incentive currently to adapt efficient discriminative methods to handle objects such as strings or graphs. For SVM this can be done by defining a kernel function $K(x, y)$ between any two objects x and y, which can be thought of as a dot product between objects. An question of interest is therefore: how to derive a kernel $K(x, y)$ from a probability density $p(x)$, in order to integrate in the kernel biological features caught by the probabilistic model?

Several methods have been proposed recently [4, 5, 6] with interesting experimental results[3]. This paper introduces a new way to derive a kernel from a probability distribution, motivated by the intuition that the "closeness" of two strings should increase when they share common substrings which appear with small probability under the probabilistic model.

For several widely-used probabilistic models on strings the kernel can be factorized and therefore efficiently computed. In this paper we explicitly show how to compute the kernel derived from either (i) independent probabilities or from (ii) Markov models. The resulting kernel can then be used by any kernel-based method, including SVM or non-linear principal component analysis[7].

As an application we derive an SVM algorithm to predict the cleavage site of signal peptide from the amino acid sequence of proteins. This algorithm is shown to significantly outperform the classical method based on a weight matrix.

2 SVM and kernels

SVM[8, 9, 10] are a family of algorithms remarkably efficient in many real-world applications. A growing interest for SVM in bioinformatics has emerged recently and resulted in powerful methods for various tasks[11, 3, 12].

A SVM basically learns how to classify objects $x \in \mathcal{X}$ into two classes $\{-1, +1\}$ from a set of labelled training examples $\{x_1, \ldots, x_m\}$. The resulting classifier is based on the decision function:

$$f(x) = \sum_{i=1}^{m} \lambda_i K(x_i, x), \tag{1}$$

where x is any new object to be classified, $K(.,.)$ is a so-called *kernel function* and the coefficients $\{\lambda_1, \ldots, \lambda_m\}$ are learned during training by solving a constrained optimization problem.

The kernel K can be considered as a dot product between objects, or more precisely between the images of the objects after a mapping to a high-dimensional Hilbert space. As a result it defines the metric properties of the space of objects, namely the "size" of each object and the "angle" between any two objects.

3 Kernels and probability distributions

Typical objects in computational biology are strings, for which many clever probabilistic models have been developed over the years to integrate biologically relevant information. In order to combine these clever models with efficient kernel-based methods it is important to develop general principles to derive kernels $K(x, y)$ from probability densities $p(x)$.

Obviously there is not a single way to do that[4, 5, 6]. In this paper we investigate particular kinds of kernels, which are probability densities on the product space $\mathcal{X} \times \mathcal{X}$, i.e., which satisfy:

$$\begin{cases} \forall (x, y) \in \mathcal{X} \times \mathcal{X}, \quad 0 \leq K(x, y) \leq 1, \\ \sum_{(x,y) \in \mathcal{X} \times \mathcal{X}} K(x, y) = 1. \end{cases} \tag{2}$$

Such kernels are called *P-kernels*[5]. An example of a P-kernel is the *product kernel*

$$K_{\text{prod}}(x, y) = p(x)p(y). \tag{3}$$

The decision function corresponding to the product kernel is simply $f(x) = a.p(x) + b$ with $a = \sum_{i=1}^{m} \lambda_i p(x_i)$, which shows that the resulting classifier simply classifies a new object x depending on whether $p(x)$ is above or below the threshold $-b/a$. The corresponding feature space is the 1-dimensional line, and each point is mapped to the number $p(x)$; two objects are "close" when their probabilities are close.

A second P-kernel example is the *diagonal kernel*:

$$K_{\text{diag}}(x, y) = p(x)\delta(x, y), \tag{4}$$

where $\delta(x, y)$ is 1 if $x = y$, 0 otherwise. The corresponding decision function tests whether the object has been seen in the training set, in which case it assigns it to the most probable class. In the corresponding feature space the set of objects $\mathcal{X}$ forms an orthogonal basis.

These two kernels are extreme and show how the choice of the kernel influences the metric properties of the space of objects. To derive an interesting kernel from a probability model, the product kernel is a good starting point because the resulting SVM classifier maps any object to a class based on the probability of that object, just like the classical Bayes classifier. In order to enhance the ability of the SVM to discriminate between two objects, it is natural to allow the "angle" between two objects to vary in order to reflect some notion of "closeness", which can not be handled by the diagonal nor the product kernel.

4 A P-kernel based on rare common subsets

Let us now introduce the main contribution of this paper, namely the definition of a new general kernel for discrete objects. We propose to consider two strings as "close" when they share rare common substrings. Here "rare" refers to the probability of the substring under the model p. As an example, if a particular sequence of amino acids is very rare in a training database, then it is natural to think that two proteins which share it could be particularly related to each other.

Let us introduce some notations to formalize this intuition. Let S be a finite set (usually $\{0, 1, \ldots, N\}$ for sequences), $\mathcal{A}$ a finite set called alphabet, and $X = (X_s)_{s\in S}$ a family of random variables defined on a probability space $(\Omega, \mathcal{F}, P)$ and indexed by the elements of S with values in $\mathcal{A}^S$. For any subset $T \subset S$ we note $X_T = (X_s)_{s\in T}$. For any subset $T \subset S$ and realization $x_T \in A^T$ we note $p_T(x_T) = P(X_T = x_T)$. If there is no ambiguity we simply write $p(x_T)$ for $p_T(x_T)$. We define similarly $p(x_T, y_U) = P(X_T = x_t, X_U = y_U)$ and $p(x_T \mid y_U) = P(X_T = x_T \mid X_U = y_U)$ for any two subsets $T \subset S$ and $U \subset S$ and realizations $x_T \in A^T$ and $y_U \in A^U$. Finally let $\mathcal{P}(S)$ be the power set of S (i.e., the set of subsets of S) and $\mathcal{V} \subset \mathcal{P}(S)$ be a particular set of subsets.

Using these notations we can define a kernel as follows:

Definition 1 *For any probability density p on $\mathcal{X}$ and any set of subsets $\mathcal{V} \subset \mathcal{P}(S)$ we define the (p,$\mathcal{V}$)-common subset kernel $K_{p,\mathcal{V}}$ by the formula:*

$$K_{p,\mathcal{V}}(x, y) = \frac{p(x)p(y)}{|\mathcal{V}|} \sum_{T\in\mathcal{V}} \frac{\delta(x_T, y_T)}{p(x_T)}, \tag{5}$$

for any two realizations $(x, y) \in A^{2S}$, where $\delta(x_T, y_T)$ is 1 if $x_T = y_T$, 0 otherwise.

The main properties of this function are summarized here:

Proposition 1

1. *For any probability density p on $\mathcal{X}$ and set of subsets $\mathcal{V} \subset \mathcal{P}(S)$ the function $K_{p,\mathcal{V}}$ is a valid P-kernel on $\mathcal{X}\times\mathcal{X}$.*
2. *When $\mathcal{V}$ only contains the empty set $\emptyset$, we have $K_{p,\{\emptyset\}} = K_{prod}$.*

3. *When $\mathcal{V}$ only contains the full set S, we have $K_{p,\{S\}} = K_{diag}$.*

4. *For a general set of subsets $\mathcal{V} \subset \mathcal{P}(S)$ we have for any $(x, y) \in A^{2S}$:*

$$K_{p,\mathcal{V}}(x, y) = \frac{1}{|\mathcal{V}|} \sum_{T \in \mathcal{V}} \sum_{z_T \in A^T} p(z_T) p(x|z_T) p(y|z_T). \tag{6}$$

This proposition, whose proof can be found in the Appendix, shows that the kernel $K_{p,\mathcal{V}}$ interpolates between the diagonal kernel and the product kernel. Equation (5) shows that correlations are introduced between strings through their common substrings indexed by $\mathcal{V}$. The contribution of a common substring is inversely proportional to its probability, so the rarer a common substring the more it increases the similarity between the strings.

For a general density p and a general set $\mathcal{V}$ there is usually no way of computing $K(p.\mathcal{V})$ without computing the $|\mathcal{V}|$ terms in the sum defining the kernel. This might be prohibitive as soon as the set $\mathcal{V}$ becomes large, which happens when one considers for instance the set $\mathcal{V} = \mathcal{P}(S)$ with size $2^{|S|}$. In the next two sections we provide two examples where the kernel can be factorized and computed in linear time with respect to $|S|$. All proofs can be found in the Appendix.

5 Independent variables

In this section we compute the kernel derived from a product probability density, corresponding to modeling the variables as independent.

Examples of such models include many probabilistic profiles for signal sequences or transcription factors binding sites in DNA. The corresponding kernels can be computed as follows:

Proposition 2 *Let $\{p_i, i \in S\}$ be a family of probability densities on A, and let p be the product distribution on A^S, i.e.,*

$$\forall x \in A^S, \quad p(x) = \prod_{i \in S} p_i(x_i). \tag{7}$$

Then the kernel $K_{p,\mathcal{V}}$ derived from p when $\mathcal{V} = \mathcal{P}(S)$ is the set of all subsets of S can be computed in linear time with respect to $|S|$ by:

$$K_{p,\mathcal{V}}(x, y) = \frac{1}{2^{|S|}} \prod_{i \in S} \phi_i(x_i, s_i), \tag{8}$$

with:

$$\phi_i(x_i, y_i) = \begin{cases} p_i(x_i) + p_i(x_i)^2 & \text{if } x_i = y_i, \\ p_i(x_i)p_i(y_i) & \text{if } x_i \neq y_i. \end{cases} \tag{9}$$

6 Markov chain and common blocks

In this section we suppose that $S = \{0, \ldots, N\}$ and that the density p is first-order Markovian, i.e.:

$$\forall x \in A^S, \quad p(x) = p_0(x_0) \prod_{i=1}^{N} p_i(x_i \,|\, x_{i-1}), \tag{10}$$

for a density p_0 on A and a set of conditional densities $p_i(x_i \,|\, x_{i-1})$ for $i = 1, \ldots, N$. The kernel resulting from such a Markov distribution is not easily computed for a general set of subsets $\mathcal{V}$ because the computation of $p(x_T)$ can be tricky for a general subset $T \subset S$. However it is possible to get a factorized expression of the kernel if we restrain the set $\mathcal{V}$ to be the set of integer intervals:

$$\mathcal{V} = \{[k,l] : 0 \leq k \leq l \leq N\} \cup \{\emptyset\}. \tag{11}$$

Observe that two realizations x and y have the same value on an interval $T = [k,l]$ (i.e., $x_T = y_T$) if and only if they share the common block $x_k \ldots x_l = y_k \ldots y_l$.

The corresponding kernel can be computed as follows:

Proposition 3 *For a Markov probability density as defined by Eq. (10) and for the set of integer intervals $\mathcal{V} = \{[k,l] : 0 \leq k \leq l \leq N\} \cup \{\emptyset\}$, the kernel between two strings $(x,y) \in A^{2S}$ can be computed in linear time with respect to N as:*

$$K_{p,\mathcal{V}}(x,y) = \phi_0(N) + \phi_1(N) + \phi_2(N), \tag{12}$$

where ϕ_0, ϕ_1 and ϕ_2 are defined recursively by:

$$\begin{cases} \phi_0(0) & = p_0(x_0)p_0(y_0), \\ \phi_1(0) & = p_0(x_0)\delta(x_0,y_0), \\ \phi_2(0) & = 0 \end{cases} \tag{13}$$

and for $i = 1, \ldots, N$:

$$\begin{cases} \phi_0(i) = p_i(x_i \,|\, x_{i-1})p_i(y_i \,|\, y_{i-1}) \times \phi_0(i-1), \\ \phi_1(i) = p_i(x_i \,|\, x_{i-1})\delta(x_i,y_i) \times \left[\phi_1(i-1) + \frac{p_i(y_i \,|\, y_{i-1})}{p_i(x_i)}\phi_0(i-1)\right], \\ \phi_2(i) = p_i(x_i \,|\, x_{i-1})p_i(y_i \,|\, y_{i-1}) \times \left[\phi_1(i-1) + \phi_2(i-1)\right]. \end{cases} \tag{14}$$

Remark 1 *The term $p_i(x_i)$ which appears in the recursive definition of $\phi_1(i)$ can itself be computed using the classical recursive algorithm:*

$$p_i(x_i) = \sum_{x_{i-1} \in A} p_{i-1}(x_{i-1}) p_i(x_i \mid x_{i-1}). \tag{15}$$

7 Experiment : cleavage site prediction of signal peptides

As an application to test the performance of the kernels introduced in this paper we consider the problem of predicting the cleavage site of protein signal sequences. These sequences, also called signal peptides, play a central role in the process of directing each newly created polypeptide to a particular destination in the organism[13]. They comprise the amino terminus of the amino-acid chain and are cleaved off while the protein is translocated through the membrane.

The identification of signal peptides and their cleavage site is of interest to the development of new effective drugs. The rapid increase in the number of available protein sequences in databases requires the use of effective prediction tools to reduce the time and cost of experimental verifications.

A simple weight matrix method[14] is known to be quite efficient to recognize cleavage sites. Indeed many cleavage sites are strongly characterized by a set of simple rules which are quantified by the weight matrix methods, e.g., the residues at positions -3 and -1 relative to the cleavage site are usually small and neutral. Computing scores from a weight matrix method is equivalent to computing the probability of a sequence under an independent model[1], so it is possible to use the kernel presented in Sec. 5 instead of the classical scoring function in order to recognize cleavage site.

In order to evaluate the performance of the kernels defined in this paper we focus on the following problem: given a window of amino acids, predict whether cleavage will occur at a given position of the window. In our experiments we chose to predict a cleavage site from the observation of 8 amino acids before the site (i.e., which should belong to the signal part) and 2 amino acids after the site (i.e., which should belong to the mature part of the protein). Hence a basic window is a sequence $x = x_{-8}x_{-7}\dots x_{-1}x_1x_2$ of length 10.

We experimented on the database of proteins used by Nielsen et al.[15][1]. We used a total number of 1418 non-redundant sequences (1011 from eukaryotes, 266 from Gram-negative prokaryotes and 141 from Gram-positive prokaryotes) made of the signal peptide and the first 30 amino acids of the mature protein. We extracted all possible amino acid windows of size 10, resulting in 66,634 windows, divided into 1418 "positive" windows (i.e., with a cleavage site between the amino acids x_{-1} and x_{+1}) and 65,216 "negative" windows.

We randomly split this database into a training set (80 % of the windows) and a test set (20%). From the training set we built:

[1]Available at `ftp://virus.cbs.dtu.dk/pub/signalp`

- a weight matrix as $w_i(x_i) = \log p_i^+(x_i) - \log p_i^{\text{total}}(x_i)$, where $p_i^+(x_i)$ is the probability that amino acid x_i occurs at position i estimated from the positive training set (using pseudocounts[1]), and p_i^{total} is the probability that amino acid x_i occurs at position i estimated from the total training set (usually referred to as the background model);
- a SVM classifier based on the product probability $p^+ = \prod_i p_i^+$ and trained on the training set. We used the public domain implementation of `mySVM`[16][2] where we implemented a user-defined kernel as presented in Sec. 5. All parameters and files necessary to reproduce this experiment can be downloaded from the author's web page [3].

This results in two competing classification functions for amino acid windows. The first one is the score function:

$$s(x) = \sum_{i=-8}^{2} w_i(x_i), \tag{16}$$

and the second one is the classification function used by the SVM:

$$f(x) = \sum_{x^{(j)} \text{ in the training set}} \lambda^{(j)} K_{p^+,\mathcal{V}}(x^{(j)}, x). \tag{17}$$

At a given threshold δ, each of these functions classifies a new example as positive or negative depending on whether the function is above or below the threshold. By varying the threshold and classifying the

[2] Available from `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`
[3] `http://web.kuicr.kyoto-u.ac.jp/~vert/`

examples in the test set, we can build a curve of true positive versus false positive for each function, and compare them.

The curves (averaged over a number of random training/test set splits) are shown on Fig. 1 and Fig. 2. The curve of the weight matrix method shows that about 44% of the sequences can be very "easily" recognized by that method, because they exhibit strong characteristics of typical cleavages sites; for the remaining sequences, the curve increases smoothly. The curve of the SVM method, on the other hand, is above the first curve, and increases smoothly from "easy" examples to "hard" examples. The difference between the two curves is particularly important for small false positive ratios, which is the most important part of the curve for concrete application. As an example, if one is ready to have 3% of false positive, then the weight matrix method would retrieve on average 46% of true positives, while the SVM method would retrieve 68% of true positive. This corresponds to an increase of 47% in terms of true positive retrieval, and illustrates the discriminative power of SVM compared to simple score functions

8 Conclusion and future work

We introduced a new class of kernels for strings which we think can be of interest for many applications in computational biology, and showed on the example of signal sequence cleavage site prediction how a SVM using this kernel significantly outperforms a classical weight matrix

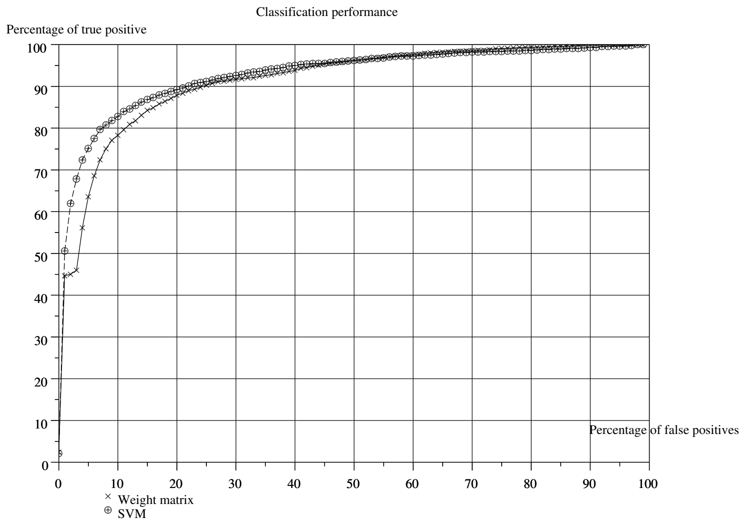

Figure 1: Classification performance of the weight matrix method and the SVM method

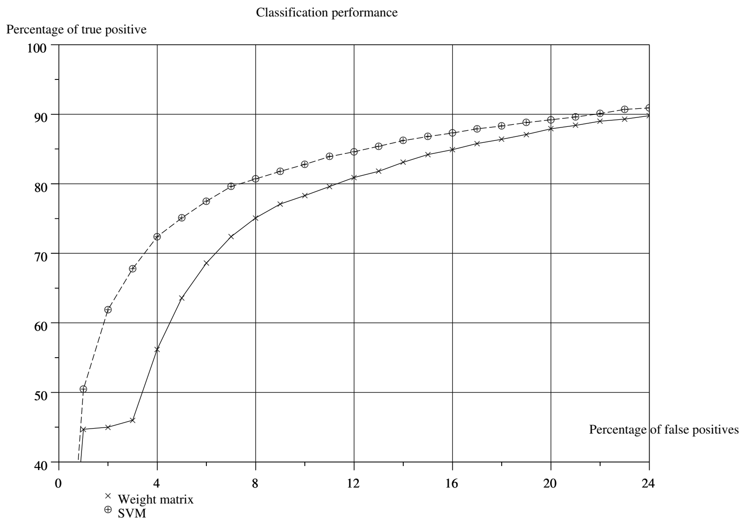

Figure 2: Classification performance for small false positive rates

method, providing a new evidence that SVM will play a more and more important role in the coming years in computational biology.

Much research remains to be done to fully exploit the capacities of kernel-based methods in bioinformatics. On the theoretical point of view, the development of new kernels for specific objects (strings, graphs...) and specific applications is an important research topic today. The particular class of P-kernels should be more deeply investigated and might give birth to interesting theoretical links between probability theory and machine learning. On the practical point of view, kernel-based methods could be tested on new tasks; moreover the class of kernel-based methods is not limited to SVM, but also includes algorithms such as non-linear principal component analysis[7], which might be of great use for data mining of biological databases.

Appendix : proofs

Proof of Proposition 1

The points 2 and 3 follow directly from the definition of the kernel in Eq. (5). To prove point 4 observe that for any subset $T \subset S$ the following holds for any $x \in A^S$ and $z_T \in A^T$:

$$p(x|z_T) = \frac{p(x)}{p(z_T)}\delta(x_T, z_T). \tag{18}$$

As a result we can use the definition of the kernel in Eq. (5) to compute, for any $(x, y) \in A^{2S}$ and $\mathcal{V} \subset \mathcal{P}(S)$:

$$\begin{aligned} K_{p,\mathcal{V}}(x,y) &= \frac{p(x)p(y)}{|\mathcal{V}|}\sum_{T\in\mathcal{V}}\frac{\delta(x_T,y_T)}{p(x_T)} \\ &= \frac{p(x)p(y)}{|\mathcal{V}|}\sum_{T\in\mathcal{V}}\sum_{z_T\in A^T}\frac{\delta(x_T,z_T)\delta(y_T,z_T)}{p(z_T)} \\ &= \frac{1}{|\mathcal{V}|}\sum_{T\in\mathcal{V}}\sum_{z_T\in A^T}p(z_T)p(x|z_T)p(y|z_T), \end{aligned} \tag{19}$$

which concludes the proof of point 4. Summing this expression for all possible x and y easily shows that $K_{p,\mathcal{V}}$ is a probability density, i.e., $\sum_{(x,y)} K_{p,\mathcal{V}}(x,y) = 1$. Hence it is a conditionally independent probability density (as defined in[6]), and it is therefore a valid P-kernel (from the main result of [6]).

Proof of Proposition 2

For a product density $p(x) = \prod_{i \in S} p_i(x_i)$, the following holds for any subset $T \subset S$:

$$\forall x_T \in A^T, \quad p(x_T) = \prod_{i \in T} p_i(x_i). \tag{20}$$

Therefore, we can compute for any $(x, y) \in A^{2S}$:

$$\frac{p(x)p(y)\delta(x_T, y_T)}{p(x_T)} = \prod_{i \in T} p(x_i)\delta(x_i, y_i) \times \prod_{i \notin T} p(x_i)p(y_i). \tag{21}$$

Using Eq. (5) and the fact that $|\mathcal{V}| = 2^{|S|}$ we can therefore compute:

$$\begin{aligned} K(x, y) &= \frac{1}{2^{|S|}} \sum_{T \subset S} \frac{p(x)p(y)\delta(x_T, y_Y)}{p(x_T)} \\ &= \frac{1}{2^{|S|}} \sum_{T \subset S} \left\{ \prod_{i \in T} p(x_i)\delta(x_i, y_i) \times \prod_{i \notin T} p(x_i)p(y_i) \right\} \\ &= \frac{1}{2^{|S|}} \prod_{i \in S} \Big\{ p(x_i)\delta(x_i, y_i) + p(x_i)p(y_i) \Big\}. \end{aligned} \tag{22}$$

Proof of Proposition 3

First observe that by mapping any interval $[k, l]$ (with $0 \leq k \leq l \leq N$) into the sequence $(s_1, \ldots, s_N)$ defined by $s_i = 0$ if $i < k$, $s_i = 1$ if $k \leq i \leq l$ and $s_i = 2$ if $i > l$, and by mapping the empty set $\emptyset$ to the constant sequence $(s_1, \ldots, s_N) = (0, \ldots, 0)$, we define a bijection between $\mathcal{V}$ and the set of sequences $\mathcal{S}$ with values in $0, 1, 2$, starting value $s_0 \in \{0, 1\}$ and increments $s_{i+1} \in \{s_i, s_i + 1\}$ for $i = 1, \ldots, N$.

Now, for any interval $T = [k, l]$ and corresponding sequence $(s_0 \ldots s_N)$, the Markov property of p yields the following equality for any realization $x_T \in A^T$:

$$p(x_T) = p_k(x_k) \prod_{i=k+1}^{l} p_i(x_i \mid x_{i-1}). \tag{23}$$

As a result it is easy to check the following relation:

$$\forall (x,y,T) \in A \times A \times \mathcal{V}, \quad \frac{p(x)p(y)\delta(x_T, y_T)}{p(x_T)} = g_0(s_0) \prod_{i=1}^{N} g_i(s_{i-1}, s_i), \tag{24}$$

where g_i is defined by $g_0(0) = p_0(x_0)p_0(y_0)$, $g_0(1) = p_0(x_0)\delta(x_0, y_0)$ and for $i = 0, \ldots, N$:

$$\begin{cases} g_i(0,0) = g_i(1,2) = g_i(2,2) = p_i(x_i|x_{i-1})p_i(y_i|y_{i-1}), \\ g_i(0,1) = \frac{p_i(x_i|x_{i-1})p_i(y_i|y_{i-1})\delta(x_i,y_i)}{p_i(x_i)}, \\ g_i(1,1) = p_i(x_i \mid x_{i-1})\delta(x_i, y_i), \end{cases} \tag{25}$$

By definition of the kernel in Eq.(5) and using Eq. (24) we therefore get:

$$K_{p,\mathcal{V}}(x_S, y_S) = \sum_{(s_1 \ldots s_N) \in \mathcal{S}} \left\{ g_0(s_0) \prod_{i=1}^{N} g_i(s_{i-1}, s_i) \right\}. \tag{26}$$

The set of equations in Proposition 3 is now the classical forward algorithm corresponding to the dynamic programming computation of this sum.

References

[1] R. Durbin *et al*, *Biological sequence analysis : Probabilistic models of proteins and nucleic acids* (Cambridge University Press, 1998).

[2] N. Friedman *et al., Journal of Computational Biology*, **7**, 601 (2000).

[3] T. Jaakkola *et al., Journal of Computational Biology*, **7**, 95 (2000).

[4] T. Jaakkola and D. Haussler in *Advances in Neural Information Processing Systems* **11**, 1998.

[5] D. Haussler, *Technical report UCSC-CRL-99-10* (1999).

[6] C. Watkins, *Technical report CSD-TR-98-11* (1999).

[7] B. Schölkopf *et al.*, in *Advances in kernel methods: support vector learning*, 327 (The MIT Press, 1999).

[8] V. Vapnik, *Statistical learning theory* (Wiley, 1998)

[9] N. Christianini and J. Shawe-Taylor, *An introduction to Support Vector Machines and other kernel-based learning methods* (Cambridge University Press, 2000).

[10] C. Burges,*Data Mining and Knowledge Discovery*, **2**, 121 (1998).

[11] M.P.S. Brown *et al., Proc. Natl. Acad. Sci. USA*, **97**, 262 (2000).

[12] P. Pavlidis *et al, Proceedings of the Pacific Symposium on Biocomputing 2001*, 151 (2001).

[13] L.M. Gierarch,*Biochemistry*, **28**, 923 (1989).

[14] G. von Heijne, *Nucleic Acids Res.*, **14**, 4683 (1986).

[15] H. Nielsen *et al., Protein Eng.* **10**, 1 (1997).

[16] S. Rüping, *mySVM - Manual* (2000).