Modeling Position Specificity in Sequence Kernels by Fuzzy Equivalence Relations

Ulrich Bodenhofer, Karin Schwarzbauer, Mihaela Ionescu, and Sepp Hochreiter

Institute of Bioinformatics, Johannes Kepler University Linz, 4040 Linz, Austria
Email: {bodenhofer,schwarzbauer,ionescu,hochreit}@bioinf.jku.at

Abstract— *This paper demonstrates that several known sequence kernels can be expressed in a unified framework in which the position specificity is modeled by fuzzy equivalence relations. In addition to this interpretation, we address the practical issues of positive semi-definiteness, computational complexity, and the extraction of interpretable features from the final support vector machine classifier.*

Keywords— fuzzy equivalence relation, kernel, sequence classification, support vector machines.

1 Introduction

The classification of biological sequences — in particular, nucleotide sequences (DNA and RNA) and amino acid sequences (proteins) — is one of the fundamental tasks in computational biology [1]. In the early days of computational biology, sequence statistics were employed. These approaches were later improved by Hidden Markov Models (HMM) and Artificial Neural Networks (ANN). In the last decade, *Support Vector Machines* (SVM) have become increasingly popular for sequence classification [2]. In many tasks — in computational biology, but also in many other domains — SVMs have outperformed all competing methods. It is justified to state that SVMs are nowadays widely considered the most powerful class of classification methods.[1] SVMs have been applied successfully in a wide spectrum of sequence classification tasks, ranging from promoter and splice site detection (both on DNA data) [3, 4, 5] to protein fold and secondary structure prediction (both on amino acid sequences) [6, 7, 8].

SVM classifiers, in their simplest form, are nothing else but linear classifiers. What distinguishes SVMs from other linear classifiers like perceptrons or logistic regressors is the fact that SVMs determine the classification function by maximizing the margin between the classes — a principle that is known to be optimal in terms of bounds on the generalization error [9, 10]. Since linear SVMs use input vectors only to compute scalar products, SVMs facilitate the so-called *kernel trick*, i.e. the replacement of the scalar products by a *kernel*, i.e. a non-linear two-place function that is positive semi-definite. The use of kernels enables support vector machines to be applicable to almost any kind of data, including raw sequence data, signals, images, or graphs — only an appropriate kernel is needed. It is not surprising, therefore, that the design of appropriate sequence kernels is one of the central research topics in sequence classification with SVMs. Most of these sequence kernels are based on comparing sub-sequences and do not take the positions of these occurrences into account. In many applications, however, the occurrence of a specific sub-sequence is only indicative if it is at a specific position or region.

[1] although one has to admit that there are applications and circumstances/requirements under which other methods may be preferable

This paper repeats established sequence kernels that are based on occurrences of specific patterns along with some position-specific variants. We demonstrate that these kernels can be expressed in a unified framework in which the position specificity is modeled by fuzzy equivalence relations. This framework, on the one hand, allows for an interpretation of existing position-specific sequence kernels from the viewpoint of fuzzy equivalence relations. On the other hand, and more importantly from the practical viewpoint, this framework gives rise to new position-specific sequence kernels by allowing to use fuzzy equivalence relations that have not been considered for modeling position specificity previously.

2 Support vector classification

Suppose that we have to do binary classification of samples from a given arbitrary non-empty set $\mathcal{X}$. The two classes are denoted with $+1$ (positive class) and -1 (negative class). For a given training set

$$\{(x_i, y_i) \mid 1 \leq i \leq l\}$$

with $x_i \in \mathcal{X}$ and $y_i \in \{-1,+1\}$ for all $i = 1,\ldots,l$, the support vector machine classifier is represented as the following discriminant function:

$$f(x) = b + \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot k(x, x_i) \tag{1}$$

In this formula, b is a real value, α_i are non-negative factors, and $k(.,.)$ is the so-called *kernel*, that is, a symmetric $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ mapping fulfilling *positive semi-definitenes*, i.e.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} z_i \cdot z_j \cdot k(x_i, x_j) \geq 0 \tag{2}$$

for all $n \in \mathbb{N}$, all $(z_1,\ldots,z_n) \in \mathbb{R}^n$, and all $(x_1,\ldots x_n) \in \mathcal{X}^n$. With the notations $\mathbf{z} = (z_1,\ldots,z_n)^T$ and $\mathbf{K} = (k(x_i,x_j))_{i=1,\ldots,n}^{j=1,\ldots,n}$, Eq. (2) can be written as $\mathbf{z}^T\mathbf{K}\mathbf{z} \geq 0$. Hence, the positive semi-definiteness of k corresponds to the positive semi-definiteness of any kernel matrix $\mathbf{K}$.

The factors $\alpha_1,\ldots,\alpha_n$ are the Lagrange multipliers of a convex quadratic optimization problem arising from margin maximization. For details, the reader is referred to introductory tutorials [11, 12] and standard SVM literature [9, 10, 13, 14]. We only note shortly that samples contributing to the final classifier, obviously those x_i for which $\alpha_i > 0$, are called *support vectors*.

The positive semi-definiteness of the kernel k serves for two purposes. First, it ensures that the SVM optimization problem is a convex quadratic one, which ensures the existence

ISBN: 978-989-95079-6-8

of a global solution that can be determined efficiently. More importantly, by the famous *Mercer theorem* [15], it guarantees that there exists a Hilbert space $(\mathcal{H}, \langle .,. \rangle)$ and a mapping $\varphi : \mathcal{X} \to \mathcal{H}$ such that the representation

$$k(x,y) = \langle \varphi(x), \varphi(y) \rangle \tag{3}$$

holds for all $x, y \in \mathcal{X}$. This means that the kernel k can be understood as a scalar product in a feature space $\mathcal{H}$.

3 An overview of sequence kernels

In this paper, we restrict ourselves to sequence kernels that are based on the occurrence of specific patterns. Assume in the following that we have a certain finite alphabet $\mathcal{A}$ and that $\mathcal{X}$ is a set of finite strings over the alphabet $\mathcal{A}$. Further assume that $\mathcal{M}$ is a finite set of patterns. A pattern can either be a string over $\mathcal{A}$ itself (i.e. an *exact pattern*) or contain *wildcards*, i.e. positions that match to an arbitrary symbol or a subset of symbols from the alphabet $\mathcal{A}$. We consider sequence kernels that can be expressed as

$$k(x,y) = \sum_{m \in \mathcal{M}} N(m,x) \cdot N(m,y), \tag{4}$$

where $N(m,x)$ denotes the number of occurrences/matches of pattern m in string x. It is obvious that, according to the representation (3), such a kernel is nothing else but a scalar product in an $|\mathcal{M}|$-dimensional real feature space, where the feature mapping φ is explicitly given as

$$\varphi(x) = (N(m,x))_{m \in \mathcal{M}}. \tag{5}$$

In order to accommodate different matching criteria used in existing sequence kernels, let us express $N(m,x)$ as

$$N(m,x) = \sum_{p=1}^{\text{length}(x)} \mathbf{1}(m,x,p), \tag{6}$$

where $\mathbf{1}(m,x,p)$ is a kernel-specific indicator function whose value is 1 if pattern m matches string x at position p and 0 if not. So, in the framework presented here, a sequence kernels is basically given by its pattern set $\mathcal{M}$ and its match indicator function $\mathbf{1}$. This representation accommodates the following well-known sequence kernels:

Spectrum kernel [6]: $\mathcal{M} = \mathcal{A}^K$, i.e. the set of patterns is the set of all K-length strings; the indicator function is given as[2]

$$\mathbf{1}(m,x,p) = \begin{cases} 1 & \text{if } p+K-1 \leq \text{length}(x) \text{ and} \\ & m = x[p \ldots p+K-1], \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the spectrum kernel maps each of the two sequences to the numbers of occurrences of all strings of length K and computes the scalar product of these two feature vectors.

Mismatch kernel [7]: this kernel is analogous to the spectrum kernel with the exception that matches are considered up to a maximal number of M mismatches, i.e. $\mathcal{M} = \mathcal{A}^K$ and

$$\mathbf{1}(m,x,p) = \begin{cases} 1 & \text{if } p+K-1 \leq \text{length}(x) \text{ and} \\ & |\{1 \leq r \leq K \mid m[r] \neq x[p+r-1]\}| \leq M, \\ 0 & \text{otherwise.} \end{cases}$$

Motif kernel [16]: $\mathcal{M}$ is a set of predefined patterns known or assumed to be related to the given classification task; the indicator mapping $\mathbf{1}$ is given as follows:[3]

$$\mathbf{1}(m,x,p) = \begin{cases} 1 & \text{if } p+\text{length}(m)-1 \leq \text{length}(x) \text{ and} \\ & m[r] = x[p+r-1] \text{ for all } 1 \leq r \leq \text{length}(m) \\ & \text{such that } m[r] \neq \text{“}*\text{”} \\ 0 & \text{otherwise} \end{cases}$$

Beside these three prominent kernels, also some others comply with the representation (4), e.g. spatial sample kernels [8].

The class of kernels (4) facilitates easy feature extraction, which is not surprising given the fact that the feature mapping φ is known explicitly. Suppose we are given a support vector machine with Lagrange multipliers $\alpha_1, \ldots, \alpha_l$ and an offset b. Then we can make the following rearrangements:

$$\begin{aligned} f(x) &= b + \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot k(x, x_i) \\ &= b + \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot \sum_{m \in \mathcal{M}} N(m,x) \cdot N(m,x_i) \\ &= b + \sum_{m \in \mathcal{M}} N(m,x) \cdot \underbrace{\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot N(m,x_i)}_{=w(m)} \end{aligned} \tag{7}$$

So we obtain $w(m)$ as the linear scaling factor with which every occurance/match of m in the sequence x is scored. Obviously, the higher the absolute value $w(m)$, the more important the pattern m is for the final classification. If $w(m)$ is positive, the pattern m is indicative for the positive class and, if $w(m)$ is negative, m is indicative for the negative class.

Note that the representation (7) is not only beneficial for feature extraction, i.e. for determining which patterns are indicative for the classification tasks. If the total number of patterns actually occurring in the training set is not excessively large, the explicit representation (7) also has strong advantages in terms of computational complexity: the weights $w(m)$ can be stored in a simple hash table indexed by the patterns. For a new sample x, it is then only necessary to sum up the weights for all occurrences of all patterns in x. For the spectrum kernel, for instance, this means adding up $\text{length}(x) - K + 1$ numbers, whereas the direct implementation of the general SVM (1) requires the computation of all $k(x, x_i)$ for which $\alpha_i > 0$ and, therefore, much more computational effort.

[2]For a given string x, we define $x[r]$ to be the r-th character in x and $x[r \ldots q]$ to be the substring that starts with the r-th and ends with the q-th position.

[3]We restrict to fixed-length motifs with only regular characters and the wildcard “$*$” that matches all single characters, although the generalization to more sophisticated patterns, e.g. by regular expressions, is straightforward.

4 Fuzzy position preference

The position-independent kernels presented in Section 3 have been used successfully in protein classification. Many protein classification tasks are concerned with inferring the structure and/or function of proteins from their amino acid sequences. Structure and function of proteins are largely determined by shorter sub-sequences (patterns/motifs), mostly irrespective of their exact position in the sequence, so position specificity is not an important issue.

When analyzing DNA or RNA nucleotide sequences, the situation is quite different. There is a rather small alphabet (four nucleic acids instead of twenty amino acids), so some patterns (in particular, shorter ones) are less indicative as they may occur by chance. The position where a pattern occurs, for example, relative to the start or end of a gene, plays a much bigger role. Therefore, there is a strong need for position-specific sequence classification.

We now want to illustrate how position specificity can be integrated into the class of kernels introduced in the previous section. So let us assume that all sequences we consider have the same length L, i.e. $\mathcal{X} = \mathcal{A}^L$, and are *aligned in a way meaningful for the given classification task*.

The simplest way to include position dependence is to consider each pattern at each position completely independent of the other occurrences at other positions (similar to the philosophy behind the *weighted degree kernel* [3]). This means that, instead of mapping a sequence to the total numbers of occurrences of all patterns (cf. (5)), each pattern induces L features, one for each position. The explicit feature mapping can be formulated as follows:

$$\varphi(x) = \big(\underbrace{(\mathbf{1}(m,x,1),\ldots,\mathbf{1}(m,x,L))}_{\text{binary list of occurrences of pattern } m}\big)_{m\in\mathcal{M}} \tag{8}$$

Then the corresponding position-specific kernel is given as follows:

$$\bar{k}(x,y) = \sum_{m\in\mathcal{M}} \sum_{p=1}^{L} \mathbf{1}(m,x,p)\cdot\mathbf{1}(m,y,p) \tag{9}$$

So this kernel basically counts the number of positions where the same pattern matches both sequences x and y. It is clear that the number of features (i.e. the dimension of the feature space $\mathcal{H}$) grows by a factor of L compared to the position-independent kernel (4) — potentially increasing the risk of overfitting. Moreover, this approach is not ideally suited for tasks in which patterns are indicative if they appear in certain regions (but not necessarily at exact positions).

The *oligo kernel* [5] solves this challenge by replacing the binary indicators in the feature mapping (8) by the sum of proximity functions around the occurrences of the pattern, with Gaussian bell functions being the standard choice. We leave this variant aside for a moment and concentrate on a different approach in line with the *shifted weighted degree (SWD) kernel* [4]. As in the kernel $\bar{k}$, a pattern m occurring at the same position in both sequences contributes 1 to the sum, but also an occurrence of a pattern m at position p in the sequence x and an occurrence of the same pattern m at position q in the sequence y may contribute a non-zero value to the sum. This is solved by a two-place weighting function $E : \{1,\ldots,L\}^2 \to [0,1]$ which corresponds to the closeness of positions p and q. The farer p and q are apart, the lower the value $E(p,q)$ should be. If $p = q$, i.e. if the positions coincide exactly, $E(p,q) = 1$ holds. So we may generalize the kernel $\bar{k}$ in the following way:

$$\tilde{k}(x,y) = \sum_{m\in\mathcal{M}} \sum_{p=1}^{L} \sum_{q=1}^{L} \mathbf{1}(m,x,p)\cdot E(p,q)\cdot\mathbf{1}(m,y,q) \tag{10}$$

If we leave some constant factors and the SWD kernel's option to define importance factors for each position aside, the position preference weights used by the SWD kernel essentially come down to

$$E_{\mathrm{SWD}}(p,q) = 2^{-|p-q|}.$$

It is easy to see that the kernel $\bar{k}$ given in (9) is a special case of $\tilde{k}$ with the position weighting function

$$E_{=}(p,q) = \begin{cases} 1 & \text{if } p = q, \\ 0 & \text{otherwise,} \end{cases}$$

i.e. the ordinary equality relation, and the position-independent variant defined in (4) is also a special case of (10) with the trivial position-independent weighting function $E_1(p,q) = 1$ (for all $p,q \in \{1,\ldots,L\}$).

Since the range of the weighting function E is the unit interval and since it is intended to be a model of the closeness of the two positions, it is immediate to ask the question whether fuzzy equivalence relations [17, 18] would be reasonable choices for E. As usual, a mapping $E : \mathcal{X}^2 \to [0,1]$ is called *fuzzy equivalence relation* with respect to a given triangular norm T if it has the following properties (for all $x,y,z \in \mathcal{X}$):

(i) Reflexivity: $E(x,x) = 1$
(ii) Symmetry: $E(x,y) = E(y,x)$
(iii) T-transitivity: $T(E(x,y),E(y,z)) \le E(x,z)$

It is trivial that the two special cases $E_{=}$ and E_1 highlighted above are fuzzy equivalence relations, no matter which triangular norm (t-norm) we consider.

It is also straightforward to prove that E_{SWD} is a fuzzy equivalence relation with respect to the product t-norm $T_{\mathbf{P}}(x,y) = x\cdot y$. This can be proved directly, but it is essentially a consequence of the well-known correspondence between pseudo-metrics and fuzzy equivalence relations with respect to continuous Archimedean t-norms [19]. As further consequences of this result, we can deduce some more examples (generally for $x,y \in \mathbb{R}$, containing the special case of natural numbers $1,\ldots,L$):

- $E_{\mathrm{lin},\sigma}(x,y) = \max(1 - \frac{1}{\sigma}|x-y|, 0)$ is a fuzzy equivalence relation with respect to the Łukasiewicz t-norm $T_{\mathbf{L}}(x,y) = \max(x+y-1,0)$.

- $E_{\mathrm{exp},\sigma}(x,y) = \exp(-\frac{|x-y|}{\sigma})$ is a fuzzy equivalence relation with respect to the product t-norm $T_{\mathbf{P}}$.

- $E_{\mathrm{Gauss},\sigma}(x,y) = \exp(-\frac{(x-y)^2}{\sigma^2})$ is a fuzzy equivalence relation with respect to the t-norm
$$T(x,y) = \exp\big(-(\sqrt{-\ln x} + \sqrt{-\ln y})^2\big),$$
which is nothing else but the Aczél-Alsina t-norm with parameter $\lambda = \frac{1}{2}$ [20].

The question arises which choices of E are feasible. In principle, every choice for which $E(p,q)$ is a non-increasing transformation of $|p-q|$ seems reasonable, and all examples mentioned above are of that kind.[4] However, there is also a technical requirement that should not be forgotten: the resulting function $\tilde{k}$ must be positive semi-definite. In order to deduce meaningful requirements on the fuzzy equivalence relation E, let us make a simple reformulation of $\tilde{k}$. It is trivial that the two inner sums are just a vector-matrix-vector product

$$\sum_{p=1}^{L}\sum_{q=1}^{L}\mathbf{1}(m,x,p)\cdot E(p,q)\cdot\mathbf{1}(m,y,q)=\varphi_m(x)^T\cdot\mathbf{E}\cdot\varphi_m(y),$$

with

$$\begin{aligned}\varphi_m(x)&=\big(\mathbf{1}(m,x,1),\ldots,\mathbf{1}(m,x,L)\big)^T,\\ \varphi_m(y)&=\big(\mathbf{1}(m,y,1),\ldots,\mathbf{1}(m,y,L)\big)^T,\\ \mathbf{E}&=\begin{pmatrix}E(1,1)&\cdots&E(1,L)\\ \vdots&\ddots&\vdots\\ E(L,1)&\cdots&E(L,L)\end{pmatrix}.\end{aligned}$$

Hence, we can rewrite $\tilde{k}$ as

$$\tilde{k}(x,y)=\sum_{m\in\mathcal{M}}\underbrace{\varphi_m(x)^T\cdot\mathbf{E}\cdot\varphi_m(y)}_{=\tilde{k}_m(x,y)}.$$

The sum of kernels is again a kernel [21]. So if we can guarantee that each $\tilde{k}_m$ is a kernel, $\tilde{k}$ is guaranteed to be a kernel. It is clear that, if the matrix $\mathbf{E}$ is positive semi-definite, $\tilde{k}_m$ is a scalar product, hence a kernel. Then the positive semi-definiteness of E, i.e. that E can be understood as a kernel itself, is a sufficient criterion for $\mathbf{E}$ to be positive semi-definite — and finally for $\tilde{k}$ to be a kernel.

For the first two examples above, the situation is clear: $E_=$ induces the identity matrices, E_1 induces matrices containing only 1's; both classes of matrices are trivially positive semi-definite. For non-trivial situations, fortunately, several results are known as well:

- Every fuzzy equivalence relation with respect to the minimum t-norm $T_{\mathbf{M}}(x,y)=\min(x,y)$ is positive semi-definite [22].
- $E_{\exp,\sigma}$ and $E_{\text{Gauss},\sigma}$ are long known to be positive semi-definite [23], where the latter is one of the most used kernels — the RBF kernel.
- $E_{\text{lin},\sigma}$ has recently been proved to be positive semi-definite [24].
- E_{SWD} is also positive semi-definite, as obviously every matrix $\mathbf{E}$ induced by E_{SWD} is diagonally dominant.

So we can conclude that the three choices $E_{\exp,\sigma}$, $E_{\text{Gauss},\sigma}$, $E_{\text{lin},\sigma}$ and E_{SWD} are reasonable and technically feasible. Further note that the former three choices have in common that the resulting kernels tend to the position-independent variant (4) as σ goes to infinity, and the resulting kernels tend to the most position-specific variant (9) as $\sigma\to 0$. So the parameter σ allows for adjusting the degree of position specificity in a continuous manner.

[4]More generally, we could also apply a transformation to the positions first.

We further note that also the oligo kernel [5] can be accommodated in the framework introduced above. This kernel uses an explicit feature representation for each pattern that convolves the occurrences with Gaussian neighborhood (for some a priori chosen σ). So we can integrate the oligo kernel into the above framework with

$$\mathbf{E}_{\text{oligo},\sigma}=\mathbf{E}_{\text{Gauss},\sigma}\cdot\mathbf{E}_{\text{Gauss},\sigma}.$$

So, from the computational point of view, everything said here is also valid for the oligo kernel. However, since entries of the matrix $\mathbf{E}_{\text{oligo},\sigma}$ can exceed the unit interval, this distance weighting matrix cannot be interpreted as a fuzzy equivalence relation.

5 Feature extraction and computational issues

There is no doubt that the position-specific variant introduced in the previous section is more complicated than the position-independent variant presented in Section 3. The question arises whether the position-specific framework also allows for simple feature extraction and computational efficiency like the position-independent variant. Fortunately, this is the case. The basis of this result is the following rearrangement of the SVM discriminant function (analogous to Section 3):

$$\begin{aligned}f(x)&=b+\sum_{i=1}^{l}\alpha_i\cdot y_i\cdot\tilde{k}(x,x_i)\\ &=b+\sum_{i=1}^{l}\alpha_i\cdot y_i\cdot\sum_{m\in\mathcal{M}}\sum_{p=1}^{L}\sum_{q=1}^{L}\mathbf{1}(m,x,p)\cdot E(p,q)\cdot\mathbf{1}(m,x_i,q)\\ &=b+\sum_{m\in\mathcal{M}}\sum_{p=1}^{L}\mathbf{1}(m,x,p)\cdot\underbrace{\sum_{i=1}^{l}\sum_{q=1}^{L}\mathbf{1}(m,x_i,q)\cdot\alpha_i\cdot y_i\cdot E(p,q)}_{=\tilde{w}(m,p)}\end{aligned}$$

A value $\tilde{w}(m,p)$ can be interpreted as the weight to which an occurrence of pattern m at position p contributes to the final result. Analogously to Section 3, $\tilde{w}(m,p)>0$ means that an occurrence of pattern m at position p is indicative for the positive class, whereas $\tilde{w}(m,p)<0$ tells us that an occurrence of pattern m at position p is indicative for the negative class. The absolute value of $\tilde{w}(m,p)$ corresponds to the strength of this influence.

It is worth to point out that the way the weights $\tilde{w}(m,p)$ are computed has a very intuitive interpretation too. Suppose that pattern m occurs at position q in a support vector x_i. Obviously, if we consider p a free variable, $E(p,q)$ is the proximity (equivalence class) of position q. In case $E=E_{\text{lin},\sigma}$ is used, it is nothing else but a triangle with width σ; in the case $E=E_{\exp,\sigma}$, it is a Gaussian bell. So the term $\alpha_i\cdot y_i\cdot E(p,q)$ is nothing else but this proximity scaled by the Lagrange multiplier α_i and the sign y_i. Thus, the list $(\tilde{w}(m,p))_{p=1,\ldots,L}$ is the superimposition of proximities of all occurrences of pattern m in the training set scaled by corresponding Lagrange multipliers and signs/classes.

The list $(\tilde{w}(m,p))_{p=1,\ldots,L}$ can be plotted as a graph over the sequence showing the influence of pattern m at each position, indicating in which regions of the sequence the pattern is indicative for either class or not indicative at all. Thus, the framework presented here allows for the same simple feature extraction as the position-independent variant. For other methods to extract features from SVM-based sequence classifiers, see [25, 26].

Moreover, it is obvious from the reformulation above that the discriminant function $f(x)$ can be evaluated in the same way as the position-independent variant — by adding up weights over all patterns occurring in the sequence x. So there is no difference in computational complexity between the position-specific and the position-independent variant. However, we have to store L times as many weights. This may be an obstacle if the number of patterns occurring in the training set and the sequence length L are both high.

For the position-specific spectrum kernel, for instance, the situation is very convenient. A sequence contains exactly $L-K+1$ sub-strings. Thus, the computation of $f(x)$ can be done by summing up $L-K+2$ values (again assuming that the lists of weights are stored in a convenient hash table). Obviously, the choice of K is of little influence on this complexity, which is is not true if we consider memory requirements. For low K's, the memory needed to store the weights grows exponentially (as $|\mathcal{M}| = \mathcal{A}^K$), but the total number of patterns that need to be considered is bounded above by the total number of patterns occurring in all support vectors.

All said above is valid for the spectrum kernel only. The mismatch kernel, for instance, is a much more difficult matter, as every sub-string of length K can match several patterns (where this number grows exponentially with M). Thus, the total number of patterns to be considered can be much higher and, more importantly, the number of patterns that need to be taken into account for computing $f(x)$ according to the above principle is much higher. Moreover, the extraction of features in the above manner is possible in principle, but impeded by the fact that the sets of sub-strings matching two different patterns may have large overlaps.

6 A DNA classification example

In this section, we demonstrate the concept introduced above by means of a case study. We consider the task of characterizing long nucleosome-free DNA segments.

Nucleosomes are a DNA packing mechanism in eukaryotic genomes. A nucleosome basically consists of a protein complex (a histone octamer) around which approximately 147 DNA base pairs (bp) are wrapped in $1\frac{2}{3}$ turns. Nonchalantly speaking, the histone complex acts as a reel around which DNA is wrapped. The DNA segment around the histone is mostly inaccessible for interactions with other molecules, most importantly, RNA polymerase. That is why the positions of nucleosomes and possible change of those positions play an essential role in transcription regulation. Therefore, it is of essential interest for the systems biology of eukaryotic cells which mechanisms determine the positioning and repositioning of nucleosomes. An interesting sub-topic is whether there are specific feature of the DNA that favor or hamper the positioning of nucleosomes.

For the yeast sub-species *Saccharomyces cerevisiae*, a commonly used model organism, data are available about where nucleosomes are located [27]. Instead of trying to find out which DNA features favor nucleosomes [28, 29], we tried to elicit DNA features that hamper the positioning of nucle-

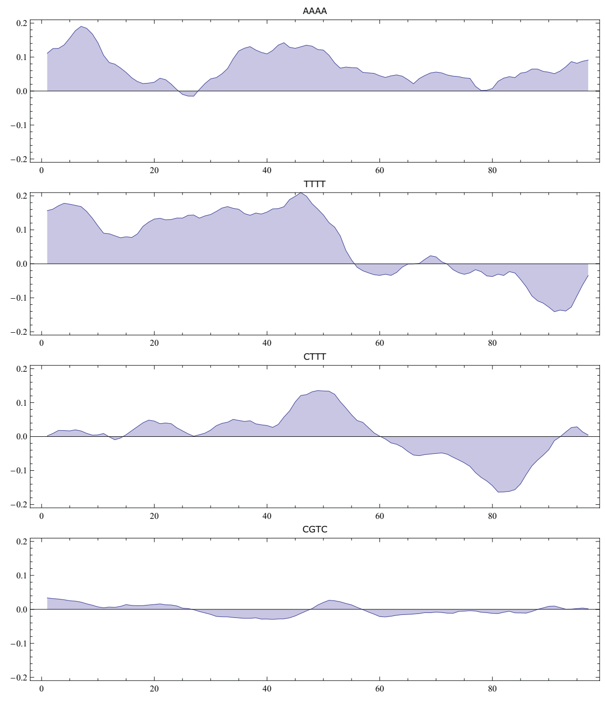

Figure 1: Weight profiles for the four patterns “AAAA”, “TTTT”, “CTTT” and “CGTC” (from top to bottom).

osomes. To this end, we selected a total number of 10226 nucleosome-free sub-sequences of 100 bp length which occur before (on either strand) a well-positioned nucleosome (with the end of the sequence aligned to the start of the nucleosome coming thereafter). This was done using the data published along with [27]. These sequences were labeled as $+1$. The data set was complemented by the same number of random 100 bp long DNA fragments occurring in nucleosomes or in between two nucleosomes that are less than 100 bp apart. These sequences were labeled as -1.

Our investigations were carried out as follows. The choices $\mathcal{A} = \{\text{“A”}, \text{“G”}, \text{“C”}, \text{“T”}\}$ and $L = 100$ were given by the data set under consideration. We used the spectrum kernel, both position-independent and position-specific, with different values for K ranging from 2 to 6. For the position-specific variant, we used $E = E_{\text{lin},\sigma}$ with σ's of 5, 10, and 20. We employed libSVM [30] to train standard soft margin SVMs. The best cross validation accuracy of 79.2% was obtained for the position-specific variant with $K = 4$ and $\sigma = 10$.

Let us have a closer look at the best setting $K = 4$. Obviously, we have $|\mathcal{M}| = 4^4 = 256$ different patterns. Hence, in order to realize the principle described in the previous section we have to store at most $256 \cdot 100 = 25,600$ numbers, which is easily manageable. Computing the discriminant function $f(x)$ for a new sequence x requires only to slide a window of length $K = 4$ over the sequence and to sum up the corresponding weights, hence only 98 additions are necessary, which is negligible compared to directly applying Eq. (1).

Figure 1 shows four examples of such weight profiles obtained for the given classification task. It is obvious that “AAAA” is indicative for the positive class (longer nucleosome-free positions), more or less regardless of the position. This is in line with the result of Peckham et al. [29]

who have found out that poly-A occurrences are indicative for non-nucleosome sequence regions. One would expect the same situation for the complementing pattern "TTTT", but the profile in Figure 1 shows that this pattern is indicative for the positive class only in the first half of the sequence, whereas it is not indicative for the last 45 bases before a nucleosome. The pattern "CTTT" is not indicative for either class in the first half of the sequence, it seems to occur frequently around 50 bases before a nucleosome, and it occurs with less-than-random probability in the last 40 bases before a nucleosome (note that the negative class is randomly sampled from nucleosomes and shorter nucleosome-free fragments). The last graph shows that pattern "CGTC" seems to be of no significance at all. Although the biological interpretation of these results is a more advanced topic, the examples demonstrate how easily claims about the significance of specific patterns can be deduced from the weight profiles.

7 Concluding remarks

In this paper, we have formulated a generalization of position-specific sequence kernels by using fuzzy equivalence relations for modeling position specificity. This is not only an interesting link, but also gives rise to new kernels, e.g. the one based on $E_{\text{lin},\sigma}$. We have obtained that these kernels facilitate an explicit representation which allows for computational efficiency and easy feature extraction.

References

[1] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[2] B. Schölkopf, K. Tsuda, and J.-P. Vert, editors. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

[3] G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, pages 277–298. MIT Press, Cambridge, MA, 2004.

[4] G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: recognition of alternatively spliced exons in *C.elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, 2005.

[5] P. Meinicke, M. Tech, B. Morgenstern, and R. Merkl. Oligo kernels for datamining on biological sequences: a case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5:169, 2004.

[6] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, K. Lauderdale, and T. E. D. Klein, editors, *Pacific Symposium on Biocomputing 2002*, pages 566–575. World Scientific, 2002.

[7] C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 1(1):1–10, 2003.

[8] P. Kuksa, P.-H. Huang, and V. Pavlovic. A fast, large-scale learning method for protein sequence classification. In *8th Int. Workshop on Data Mining in Bioinformatics*, pages 29–37, Las Vegas, NV, 2008.

[9] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1986.

[10] V. N. Vapnik. *Statistical Learning Theory*. Adaptive and Learning Systems. Wiley Interscience, 1998.

[11] C. J. M. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2:121–167, 1998.

[12] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks*, 12(2):181–201, 2001.

[13] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[14] B. Schölkopf and A. J. Smola. *Learning with Kernels*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2002.

[15] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[16] A. Ben-Hur and D. Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19:i26–i33, 2003.

[17] L. A. Zadeh. Similarity relations and fuzzy orderings. *Inform. Sci.*, 3:177–200, 1971.

[18] L. Valverde. On the structure of F-indistinguishability operators. *Fuzzy Sets and Systems*, 17(3):313–328, 1985.

[19] B. De Baets and R. Mesiar. Pseudo-metrics and T-equivalences. *J. Fuzzy Math.*, 5(2):471–481, 1997.

[20] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Kluwer Academic Publishers, Dordrecht, 2000.

[21] C. H. FitzGerald, C. A. Micchelli, and A. Pinkus. Functions that preserve families of positive semidefinite matrices. *Linear Alg. Appl.*, 221:83–102, 1995.

[22] B. Moser. On the T-transitivity of kernels. *Fuzzy Sets and Systems*, 157(1):1787–1796, 2006.

[23] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2:11–22, 1986.

[24] L. Belanche, J. L. Vázquez, and M. Vázquez. Distance-based kernels for real-valued data. In C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, editors, *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 3–10. Springer, Berlin, 2008.

[25] T. Lingner and P. Meinicke. Remote homology detection based on oligomer distances. *Bioinformatics*, 22(18):2224–2231, 2006.

[26] S. Sonnenburg, A. Zien, P. Philips, and G. Rätsch. POIMs: positional oligomer importance matrices—understanding support vector machine-based signal detectors. *Bioinformatics*, 24:i6–i14, 2008.

[27] W. Lee, D. Tillo, N. Bray, R. H. Morse, R. W. Davis, T. R. Hughes, and C. Nislow. A high-resolution atlas of nucleosome occupancy in yeast. *Nature Genetics*, 39(10):1235–1244, September 2007.

[28] E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. C. Thåström, Y. Field, I. K. Moore, J.-P. Z. Wang, and J. Widom. A genomic code for nucleosome positioning. *Nature*, 442(17):1235–1244, August 2006.

[29] H. E. Peckham, R. E. Thurman, Y. Fu, J. A. Stamatoyannopoulos, W. S. Noble, K. Struhl, and Z. Weng. Nucleosome positioning signals in genomic DNA. *Genome Res.*, 17:1170–1177, 2007.

[30] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.