18. Kernel Methods and Applications in Bioinformatics

Yan Fu

The kernel technique is a powerful tool for constructing new pattern analysis methods. Kernel engineering provides a general approach to incorporating domain knowledge and dealing with discrete data structures. Kernel methods, especially the support vector machine (SVM), have been extensively applied in the bioinformatics field, achieving great successes. Meanwhile, the development of kernel methods has also been strongly driven by various challenging bioinformatic problems. This chapter aims to give a concise and intuitive introduction to the basic principles of the kernel technique, and demonstrate how it can be applied to solve problems with uncommon data types in bioinformatics. Section 18.1 begins with the product features to give an intuitive idea of kernel functions, then presents the definition and some properties of kernel functions, and then devotes a subsection to a brief review of kernel engineering and its applications to bioinformatics. Section 18.2 describes the standard SVM algorithm. Finally, Sect. 18.3 illustrates how kernel methods can be used to address the peptide identification and the protein homology prediction problems in bioinformatics, while Sect. 18.4 concludes.

Kernel methods are a large class of pattern analysis methods that use the kernel technique to implicitly map input patterns to a feature space [18.1]. As the dot product in the feature space, a kernel function can be incorporated into any computational processes that exclusively involve dot product operations on input patterns. Kernel functions have been most successful in statistical learning algorithms, exemplified by the support vector machine (SVM) [18.2]. They can also be viewed as similarity measures of input patterns and used in all distance-based algorithms.

When the input patterns or the variables taken by a kernel function are vectors, the mapping induced by the kernel function is usually nonlinear, and all dot-product-based linear algorithms can be directly extended to their nonlinear versions by simply replacing the dot product with the kernel function. When the input patterns are not vectors, such as strings, trees, and graphs, kernel functions provide a general approach to making vector-based algorithms applicable to these nonvector or discrete data structures.

Kernel methods, especially the SVM, have been extensively applied in bioinformatics, achieving great successes (see [18.3–5] for reviews). Meanwhile, the development of kernel methods has also been strongly driven by various challenging bioinformatic problems;

For example, bioinformatics is a field full of discrete data structures, such as nucleic acid and protein sequences, phylogenetic trees, and molecular interaction networks. Engineering proper kernel functions for these biological data has considerably expanded the application scope of kernel methods.

This chapter aims to give readers a concise and intuitive introduction to the basics of the kernel technique, and demonstrate how it can help construct a nonlinear algorithm and how it can be used to solve challenging problems in bioinformatics. Although the well-known SVM algorithm will be briefly described as an example of kernel methods, the emphasis of this chapter is on more extensive usages of kernel functions, such as kernel engineering and kernel functions for similarity search.

18.1 Kernel Functions

18.1.1 Product Features

To have an intuitive idea of kernel functions, let us start with the product features [18.6]. In pattern classification, the input patterns are usually from the real vector space, i. e., $x \in \mathbb{R}^N$, and linear classification algorithms use a hyperplane in the vector space to classify the patterns. Different ways to search for the hyperplane result in different algorithms. However, it is often the case in practice that the patterns cannot be classified by a hyperplane in the original space; For example, the essential features of patterns may be all the d-order products (called product features) of dimensions of the input space

$$x_{j_1} \cdot x_{j_2} \cdot \cdots \cdot x_{j_d} ,$$
$$j_1, j_2, \ldots, j_d \in \{1, \ldots, N\} . \tag{18.1}$$

All product features constitute a new space F (called the feature space), in which patterns are linearly separable; i. e., they can be separated by a hyperplane in F. For example, when $N = 2$ and $d = 2$, the dimensionality N_F of the feature space F is 3

$$(x_1, x_2) \rightarrow \left(x_1^2, x_2^2, x_1 x_2\right) . \tag{18.2}$$

In general, we have

$$N_F = \frac{(N+d-1)!}{d!(N-1)!} . \tag{18.3}$$

N_F will increase rapidly with N and d; For example, for images with 16×16 pixels, we have $N = 256$, and when $d = 2$, the order of magnitude of N_F is as large as 10^{10}. Such high dimensionality will lead to serious computational difficulties.

However, if an algorithm only involves dot product operations on input vectors, then we need not actually work in the feature space. We can construct a version of the algorithm for the feature space from the input space, as long as we can compute the dot product in the feature space from the original space. Consider the feature space with dimensions being the ordered product features. When $N = 2$ and $d = 2$, the input patterns are transformed from two dimensions to four dimensions

$$C_2 : (x_1, x_2) \rightarrow \left(x_1^2, x_2^2, x_1 x_2, x_2 x_1\right) . \tag{18.4}$$

The dot product of two vectors in this feature space is then

$$C_2(\boldsymbol{x}) \cdot C_2(\boldsymbol{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (\boldsymbol{x} \cdot \boldsymbol{y})^2 . \tag{18.5}$$

In general, if C_d maps $\boldsymbol{x} \in \mathbb{R}^N$ to vector $C_d(\boldsymbol{x})$ with elements consisting of all ordered d-order products, then we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = C_d(\boldsymbol{x}) \cdot C_d(\boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y})^d . \tag{18.6}$$

Further, if the feature space consists of all product features of up to d-orders, we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d , \tag{18.7}$$

which is the commonly used polynomial kernel function in kernel methods.

18.1.2 Definition and Properties of Kernel Functions

A general definition of kernel functions is given as: A *kernel function*, or a *kernel* for short, is a binary function k such that, for all $\boldsymbol{x}, \boldsymbol{y} \in A$, we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y}) , \tag{18.8}$$

where ϕ is some mapping from the input space A to a feature space B. Usually, ϕ is a nonlinear mapping, and the feature space B has very high or even infi-

nite dimensionality (Hilbert space). According to this definition of kernel functions, any computations based on dot products in the feature space can be accomplished by the kernel function from the input space, thus avoiding the explicit mapping from the input space to the feature space. Typical examples of kernel functions are

$$\begin{aligned}\text{Linear kernel: } & \boldsymbol{x}\cdot\boldsymbol{y}\\ \text{Polynomial kernel: } & [a(\boldsymbol{x}\cdot\boldsymbol{y})+\gamma]^d\\ \text{RBF kernel: } & \exp(-\lambda||\boldsymbol{x}-\boldsymbol{y}||^2)\\ \text{Sigmoid kernel: } & \tanh[a(\boldsymbol{x}\cdot\boldsymbol{y})+\gamma]\,. \end{aligned} \tag{18.9}$$

RBF stands for radical basis function. Note that the sigmoid kernel satisfies the definition of kernel functions only for certain parameter values [18.2]. Some operations on kernel functions lead to still valid kernel functions; For example, if $k_1(\boldsymbol{x},\boldsymbol{y})$ and $k_2(\boldsymbol{x},\boldsymbol{y})$ are valid kernels, then the following functions are also valid kernels:

$$\begin{aligned} &k(\boldsymbol{x},\boldsymbol{y}) = a_1k_1(\boldsymbol{x},\boldsymbol{y})+a_2k_2(\boldsymbol{x},\boldsymbol{y})\,, \quad \text{for } a_1,a_2>0\,,\\ &k(\boldsymbol{x},\boldsymbol{y}) = k_1(\boldsymbol{x},\boldsymbol{y})k_2(\boldsymbol{x},\boldsymbol{y})\,,\\ &k(\boldsymbol{x},\boldsymbol{y}) = \mathrm{pol}^+[k_1(\boldsymbol{x},\boldsymbol{y})]\,,\\ &k(\boldsymbol{x},\boldsymbol{y}) = \exp[k_1(\boldsymbol{x},\boldsymbol{y})]\,, \end{aligned} \tag{18.10}$$

where pol^+ indicates polynomials with positive coefficients.

According to the definition of kernel functions, all operations that exclusively involve dot products in the feature space can be implicitly done in the input space by an appropriate kernel. A nonlinear version of an algorithm can be readily obtained by simply replacing the dot products with kernels. This is called the *kernel technique*. The underlying mathematical conclusions of it were derived almost one century ago [18.7], but it was very recently that the kernel technique was widely used. The most successful application of kernels is to extend various linear learning algorithms to their nonlinear versions. The first such attempt was made in 1964 [18.8]. However, the great success of kernel methods is due to the SVM algorithm, introduced in the early 1990s [18.2, 9]. Other famous kernel-based nonlinear learning algorithms include kernel principal analysis [18.10], kernel canonical correlation analysis [18.11], kernel discriminant analysis [18.12], kernel independence analysis [18.13], etc.

However, kernels are not limited to learning algorithms. As the dot product in the feature space, a kernel is a measurement of similarity, and defines the metrics of the feature space [18.6]; For example, we have the following kernel-based cosine and Euclidian distances in the feature space:

$$\cos[\phi(\boldsymbol{x}),\phi(\boldsymbol{y})] = \frac{k(\boldsymbol{x},\boldsymbol{y})}{\sqrt{k(\boldsymbol{x},\boldsymbol{x})\cdot k(\boldsymbol{y},\boldsymbol{y})}}\,, \tag{18.11}$$

$$d[\phi(\boldsymbol{x}),\phi(\boldsymbol{y})] = \sqrt{k(\boldsymbol{x},\boldsymbol{x})+k(\boldsymbol{y},\boldsymbol{y})-2k(\boldsymbol{x},\boldsymbol{y})}\,. \tag{18.12}$$

Therefore, in a more general sense, all distance-based algorithms can be kernelized [18.14]. Kernels can even be directly used for similarity search, e.g., nearest-neighbor search [18.15, 16] and information retrieval [18.17–19].

18.1.3 Kernel Engineering and Applications in Bioinformatics

In many real-world problems, the input patterns are not given in the form of vectors but are nonvector, discrete structures, such as strings, trees, graphs, or objects in any form. Usually, these discrete structures are very difficult, if not impossible, to represent as vectors. Thus, vector-based algorithms cannot be directly applied to these structures. Even if these structures can be vectorized in some way, the essential information may be irretrievably lost, and therefore the discriminative power of algorithms may be greatly reduced.

The common kernels given in (18.9) require the input patterns to be vectors. However, in fact, in the definition of kernel functions the input space can be the set of any types of patterns or objectives, as long as the kernel can be represented as the dot product in some space. If we can construct an appropriate kernel for the discrete structure of interest, all vector-based algorithms can be directly applied to the discrete structure by simply replacing the dot product with the kernel. In such cases, a kernel can be considered as a similarity measurement between input patterns [18.6].

A string kernel has been proposed for text classification, which implicitly maps strings into the feature space of all possible substrings [18.20]. *Watkins* showed that the conditionally symmetrically independent joint probability distributions can be written as dot products and therefore are valid kernels [18.21]. *Haussler* proposed a more general scheme for kernel engineering, named convolution kernels [18.22]. The Fisher kernel is a method for constructing kernels for classifiers using generative models [18.23]. *Kondor* and *Lafierty* proposed the diffusion kernels on graphs [18.24].

Kernel engineering has been successfully applied to bioinformatics. The mismatch kernels, a class of string kernels, showed good performance in the protein clas-

sification problem [18.25, 26]. A kernel on fixed-length strings was proposed and used for prediction of signal peptide cleavage sites [18.27]. It is based on the fact that strings are similar to each other if they possess many common rare substrings. *Zien* et al. [18.28] incorporated the local correlation in DNA sequences into a kernel function, and obtained better results than previous methods for translation initiation site prediction. The Fisher kernel based on the hidden Markov model (HMM) was used to detect protein homologies and performed better than other methods including HMM [18.29]. As pointed out by *Watkins* [18.21], the pair HMM used for scoring sequence alignment is in fact a valid kernel. Diffusion kernels were used for microarray data analysis [18.30]. A tree kernel on phylogenetic profiles was engineered and used together with SVM and principle component analysis (PCA) for biological function prediction [18.31]. Kernels on graphs were engineered and used to predict protein functions [18.32].

In summary, as the dot products in the feature space, kernels can be used in all dot-product-based algorithms. We can directly use a kernel without knowing the specific form of the feature space induced by the kernel. This advantage makes kernels a powerful tool to deal with problems on discrete structures. There are plenty of discrete structures in the biological domain, such as DNA or protein sequences, protein three-dimensional structures, phylogenetic trees, interaction networks, etc. These data are hard to convert into vectors for analysis by vector-based methods. The kernel technique is a promising solution to these problems. Research in this direction is still in its infancy. Engineering more powerful kernels on more discrete structures is an open problem.

18.2 Support Vector Machines

This section illustrates how the kernel function can be incorporated into SVM [18.2, 9, 33], the first and most successful kernel method.

18.2.1 Maximum-Margin Classifier

When two classes of patterns (represented as vectors) are linearly separable, multiple hyperplanes that can classify the patterns exist in the vector space. Different linear classifiers use different criteria to find the hyperplane. The so-called maximum-margin hyperplane is the one that not only separates the patterns but also has the largest margin to the patterns. According to the statistical learning theory [18.2], the maximum-margin hyperplane has low Vapnik–Chervonenkis (VC) dimension and better generalizability (ability to classify unseen patterns). The algorithm searching for the maximum-margin hyperplane is called a support vector machine (SVM). As we will see later, SVM only involves dot product operations on input vectors and therefore can be kernelized. In addition to the statistical foundation, the kernel technique is another factor contributing to the great success of SVM. For this reason, SVM is sometimes called a kernel machine. A third advantage of SVM is the global optimality of its solution. Multiple ways to formulate SVM have been presented, and there are SVMs for different purposes, e.g., classification and regression. Below, we introduce the standard SVM for classification.

Let H be a hyperplane that can separate two classes of samples, and H_1 and H_2 be two hyperplanes parallel to H and passing through the samples closest to H. The distance between H_1 and H_2 is called the classification margin. Suppose that the equation for hyperplane H is $\boldsymbol{x}\cdot\boldsymbol{w}+b=0$, where $\boldsymbol{x}\in\mathbb{R}^N$, $\boldsymbol{w}$ is the weight coefficient vector, and b is the displacement. We can normalize this equation so that a linearly separable two-class training sample set, $(\boldsymbol{x}_i, y_i)$, $i=1,2,\ldots,n$, $\boldsymbol{x}\in\mathbb{R}^N$, $y\in\{+1,-1\}$, satisfies

$$y_i(\boldsymbol{x}\cdot\boldsymbol{w}+b)-1\geq 0,\quad i=1,2,\ldots,n\,. \tag{18.13}$$

Under this condition, we have that the classification margin is $2/||\boldsymbol{w}||$. Maximizing the margin is equivalent to minimizing $||\boldsymbol{w}||$. The hyperplane that satisfies the condition in (18.13) and minimizes $||\boldsymbol{w}||$ is the maximum-margin hyperplane. Figure 18.1 shows an example in two-dimensional space. To find the

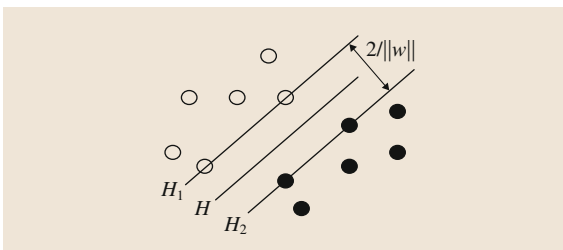

Fig. 18.1 Maximum margin

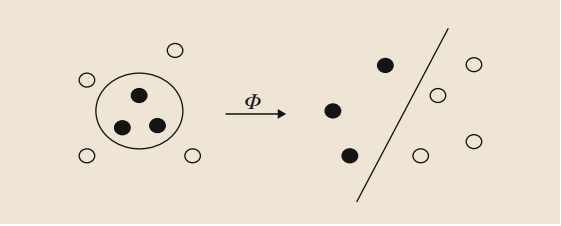

Fig. 18.2 Nonlinear mapping

maximum-margin hyperplane, we need to solve the following optimization problem

$$\min_{\boldsymbol{w},b} \; ||\boldsymbol{w}||^2$$
$$\text{with} \quad y_i(\boldsymbol{x}\cdot\boldsymbol{w}+b)-1\geq 0, \quad i=1,2,\ldots,n\,. \tag{18.14}$$

The Lagrangian dual problem of this problem is

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\boldsymbol{x}_i\cdot\boldsymbol{x}_j)$$
$$\text{with} \sum_{i=1}^{n} y_i\alpha_i = 0\,,$$
$$\forall i : \alpha_i \geq 0\,, \quad i=1,2,\ldots,n\,. \tag{18.15}$$

Solving this optimization problem leads to a classification function

$$f(x) = \operatorname{sgn}\left[\sum_{i=1}^{n}\alpha_i^* y_i(\boldsymbol{x}\cdot\boldsymbol{x}_i)+b^*\right]\,. \tag{18.16}$$

When the training samples are not linearly separable, we cannot find a solution to the optimization problem in (18.15). However, if the samples are nonlinearly separable, we can map the input samples to a higher-dimensional space with a properly chosen nonlinear mapping Φ so that the samples become linearly separable and the problem in (18.15) is solvable, as shown in Fig. 18.2. However, this may dramatically increase the dimensionality of the feature space, resulting in the so-called dimension disaster. The answer lies in the kernel technique introduced above. Notice that the problem in (18.15) and the optimal classification function in (18.16) exclusively involve the dot product operation on input vectors. We can avoid an explicit nonlinear mapping by using some appropriate kernel function $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ to replace the dot product $\boldsymbol{x}_i\cdot\boldsymbol{x}_j$, where $k(\boldsymbol{x}_i,\boldsymbol{x}_j)=\Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)$. In this way, we have the nonlinear maximum-margin classifier.

18.2.2 Soft Margin

Above, we have shown how kernel functions can be used to implicitly map the samples into a feature space in which the samples become linearly separable. However, it may be problematic if we always rely on the mapping induced by a kernel function to achieve perfect separation of samples. This is because the samples in practice are usually imperfect and are subject to interferences of various noises, and the linear inseparability of samples in the input space may have resulted from the noises and not be an inherent characteristic of the patterns. Further, even if the patterns are not linearly separable in the input space, there are still noises in the samples. A strong nonlinear mapping seeking perfect separation may lead to potential overfitting to training samples and reduced generalizability of classifiers. Therefore, a good algorithm should be able to tolerate some training errors. In SVM, this is achieved by introducing the concept of soft margin. The idea of this is to augment the margin by allowing some misclassified samples in the training process. The optimization problem then becomes

$$\min_{\boldsymbol{w},b} \quad \tfrac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{with} \quad y_i(\boldsymbol{x}\cdot\boldsymbol{w}+b)-1+\xi_i\geq 0\,, \quad i=1,2,\ldots,n\,,$$
$$\xi_i\geq 0\,, \quad i=1,2,\ldots,n\,, \tag{18.17}$$

where ξ_i are slack variables, C is the parameter controlling the tradeoff between the classification margin and training errors, and the coefficient 1/2 is introduced for representation convenience. The Lagrangian dual problem of this problem is

$$\max_{\alpha} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\boldsymbol{x}_i\cdot\boldsymbol{x}_j)$$
$$\text{with} \quad \sum_{i=1}^{n} y_i\alpha_i = 0\,,$$
$$\forall i : 0\leq\alpha_i\leq C, \quad i=1,2,\ldots,n\,. \tag{18.18}$$

Again, the problem exclusively involves the dot product operation on input vectors. Therefore, the dot product can be replaced by a kernel function; That is,

$$\max_{\alpha} \quad \sum_{i=1}^{n}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)$$
$$\text{with} \quad \sum_{i=1}^{n} y_i\alpha_i = 0\,,$$
$$\forall i : 0\leq\alpha_i\leq C, \quad i=1,2,\ldots,n\,. \tag{18.19}$$

Solving this optimization problem leads to the nonlinear soft margin classifier

$$f(x) = \mathrm{sgn}\left[\sum_{i=1}^{n} \alpha_i^* y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b^*\right], \tag{18.20}$$

in which α_i^* is the solution to the problem in (18.19), and b^* can be calculated with

$$b^* = \frac{1}{y_i} - \sum_{j=1}^{n} \alpha_i^* y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\,, \tag{18.21}$$

where i is an arbitrary index satisfying $0 < \alpha_i < C$.

Only the samples with $\alpha_i^* > 0$ contribute to the final maximum-margin hyperplane, being called support vectors (SVs). SVs are those samples that are on or between the hyperplanes H_1 and H_2.

18.3 Applications of Kernel Methods to Bioinformatics

Kernel methods, especially the SVM, have been successfully used to address various biological problems, e.g., gene prediction [18.34], gene expression data analysis [18.35], RNA splicing site prediction [18.36], protein classification [18.26], protein structure prediction [18.37], protein function prediction [18.38], protein mass-spectrometric data analysis [18.39, 40], etc. This section does not list all applications of kernel methods in bioinformatics (some reviews already exist, e.g., [18.3–5]), but gives some detailed research results on two specific problems, namely peptide identification from tandem mass spectra and protein homology prediction.

18.3.1 Kernel Spectral Dot Product for Peptide Identification

Peptide identification from tandem mass spectra is the foundation of bottom-up proteomics [18.41]. In tandem mass spectrometry, peptides are ionized, isolated, and fragmented. Roughly speaking, the tandem mass spectrum of a peptide is a mass (or more accurately, mass-to-charge ratio m/z) histogram of fragment ions of this peptide [18.42]. To identify the peptide responsible for a tandem mass spectrum, the database search approach is commonly used, in which theoretical mass spectra are predicted from the peptide sequences in a database for comparison with the input experimental spectrum [18.43]. However, the mass spectrum of a peptide cannot in general be accurately predicted. Therefore, the scoring function that measures the similarity between the theoretical and the experimental spectra is the core of a peptide identification algorithm.

The spectral dot product (SDP) is the most widely used similarity measure for mass spectra [18.43, 44]. In SDP, mass spectra are represented as vectors and the SDP is simply the dot product of spectral vectors. A disadvantage of SDP is that it, as a linear function, totally ignores possible correlations among fragment ions. According to expert knowledge, when positively correlated fragment ions are matched together, the matches are more reliable as a whole than as individuals. Therefore, such matches should be scored higher than separate matches.

Kernel Spectral Dot Product

A straightforward idea is to extend the SDP with a kernel to incorporate the correlations between fragment ions. Since not all fragment ions are correlated, common kernel functions, e.g., the polynomial kernel, do not directly apply here. The central problem becomes how to find a kernel that only emphasizes co-occurring matches of truly correlated fragment ions while ignoring others. Inspired by the locally improved polynomial kernel [18.28, 45], we solved this problem by exerting kernels separately on all possible groups of correlated fragment ions, and then summing them up [18.46]. This is achieved by arranging the predicted fragment ions into a correlative matrix as shown in Fig. 18.3, and grouping the correlated fragment ions into local correlative windows, e.g., the dotted rectangles in Fig. 18.3.

All fragment ions in a theoretical spectrum are assumed to possess unique m/z values. Under this assumption, all nonzero dimensions in the theoretical spectral vector $\boldsymbol{t}$ can be arranged into a matrix $\boldsymbol{T} = (t_{pq})_{m\times n}$ according to their fragmentation positions and fragment ion types, where m is the number of fragment ion types for theoretical fragmentation and $n+1$ is the peptide sequence length; For example, $t_{2,3}$ corresponds to the fragment ion b_3^* in Fig. 18.3. For the experimental spectral vector $\boldsymbol{c}$, the dimensions at the m/z value corresponding to t_{pq} are also extracted and arranged into a matrix $\boldsymbol{C} = (c_{pq})_{m\times n}$. It follows that

$$\mathrm{SDP} = \boldsymbol{c}\cdot\boldsymbol{t} = \sum_{p=1}^{m}\sum_{q=1}^{n} c_{pq} t_{pq}\,. \tag{18.22}$$

Given the definition of correlative windows, the general form of the kernel spectral dot product (KSDP) is defined as

$$\mathrm{KSDP} = \sum_j k(\boldsymbol{c}_j, \boldsymbol{t}_j) \,, \tag{18.23}$$

where $k(\boldsymbol{c}_j, \boldsymbol{t}_j)$ is a kernel function, $\boldsymbol{c}_j$ is the vector with elements c_{pq}, and $\boldsymbol{t}_j$ is the vector with elements t_{pq}, in which the subscript (p, q) traverses all the elements in the j-th correlative window in the matrices $\boldsymbol{C}$ and $\boldsymbol{T}$.

The KSDP given in (18.23) is also a kernel function. If $k(\boldsymbol{c}_j, \boldsymbol{t}_j)$ is the dot product kernel, the KSDP reduces to the SDP. For a properly chosen kernel function $k(\boldsymbol{c}_j, \boldsymbol{t}_j)$, the KSDP implicitly maps the spectral space to a high-dimensional space where the new dimensions correspond to the combinations of correlated fragment ions.

KSDP for Consecutive Fragment Ions

In our earlier work [18.46], we developed the computation of the KSDP for consecutive fragment ions using the locally improved polynomial kernel, given by

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\sum_{k=j-l_1}^{j+l_2}(c_{ik}t_{ik})^{\frac{1}{d}}\right]^d \,, \tag{18.24}$$

where the positive integers l_1 and l_2 are equal to $\lfloor (l-1)/2 \rfloor$ and $\lceil (l-1)/2 \rceil$, respectively, the integer l is the size of the correlative window, and c_{ik} and t_{ik} are set to zero for $k \le 0$ and $k > n$.

Here, we build on our previous work by developing a radial basis function (RBF) version of the KSDP for consecutive fragment ions as

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\exp\left[-\gamma\sum_{k=j-l_1}^{j+l_2}(c_{ik}-t_{ik})^2\right] \,, \tag{18.25}$$

where γ is the parameter in the RBF kernel. The RBF-KSDP can be computed in $O(mn)$ time, similarly to the polynomial KSDP.

Performance of KSDP

To show the effectiveness of the KSDP and to explore its parameters, the polynomial KSDP given in (18.24) and the RBF-KSDP given in (18.25) were directly used as the scoring function of the pFind search engine [18.46, 47]. Spectra from a standard protein dataset were searched [18.46, 48]. The error rates of the two KSDP implementations are given in Fig. 18.4. Both implementations significantly outperformed SDP

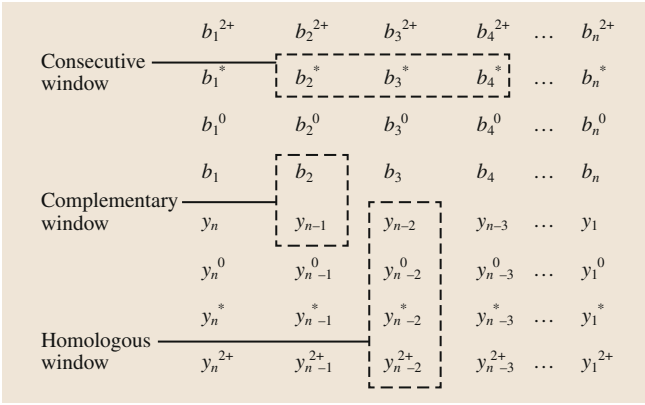

Fig. 18.3 Correlative matrix and examples of correlative windows. *Horizontal direction* is the fragmentation position in peptides, and *vertical direction* is the ion type

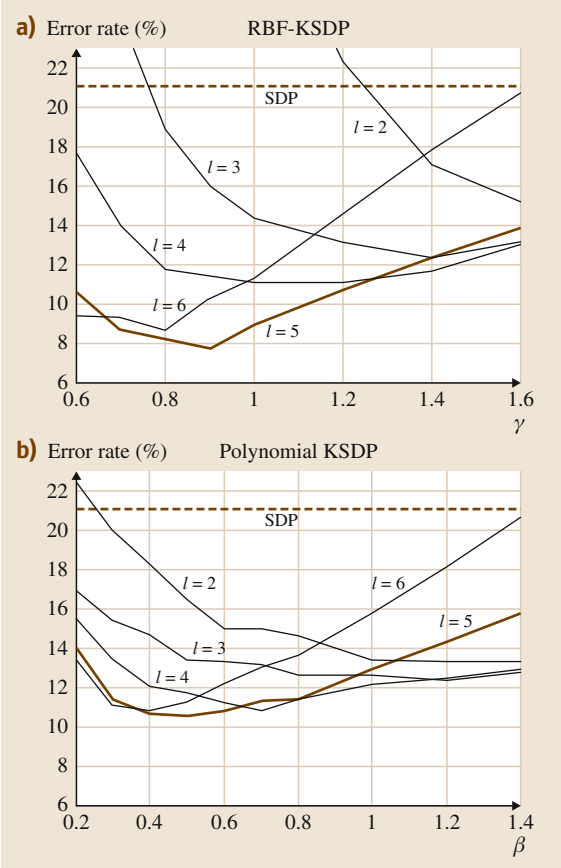

Fig. 18.4a,b Error rates of KSDP for consecutive fragment ions **(a)**. In the polynomial KSDP, the parameter d is equal to $1 + \beta(l-1)$ **(b)**

for a large range of parameter values. Compared with SDP, RBF-KSDP reduced the error rate by 13% at best in this experiment.

18.3.2 Pair Kernel for Protein Homology Prediction

The three-dimensional structures of proteins are crucial for the biological functions of proteins. The experimental approach to protein structure determination is both slow and expensive, and therefore, theoretical prediction of protein structure becomes an important research topic in bioinformatics. Since homologous proteins (evolved from the same ancestor) usually share similar structures, predicting protein structures based on protein homologies has been one of the most important bioinformatic problems [18.49]. Protein homology prediction is a key step in template-based protein structure prediction methods.

In terms of information retrieval and machine learning, protein homology prediction is a typical ranking problem [18.50]. In this problem, the database objects are protein sequences with known three-dimensional structures, and the query is a protein sequence with unknown structure. The objective is to find those proteins in the database that are homologous to the query protein so that the homologous proteins can be used as structural templates. The homology or match of two proteins can be captured by multiple features, which, as in other retrieval problems, need to be integrated into a single score (called a ranking function) in an intelligent manner in order to accurately rank the proteins in the database. Currently, this is often done by learning a ranking function from a training dataset.

A major characteristic of the ranking-function learning problem is that each feature vector is computed based on a query. Therefore, all feature vectors are partitioned into groups by queries. Here, we call each group of data associated with a query a block. Unlike traditional learning tasks, e.g., classification and regression, in which data are assumed to be independently and identically distributed, the ranking data belonging to the same block are correlated via the same query. This block structure of the data is a unique feature of the ranking-function learning problem. Although ignoring the existence of the block structure would reduce the complexity of the ranking-function learning problem, a potential source of information for improving the ranking performance would also be ignored. In the past, the block structure was not fully explored by most ranking-function learning algorithms.

Pair Kernel

Here, we explore a kernel engineering approach to making use of the block structure information and give some preliminary results. The approach is quite general. We propose the following kernel, named a *pair kernel*, for learning to rank

$$k_{\mathrm{p}}(\langle d_i, q_u\rangle, \langle d_j, q_v\rangle) = k_{\mathrm{s}}(s_{iu}, s_{jv}) + k_{\mathrm{q}}(q_u, q_v)\ , \tag{18.26}$$

where d_i and d_j are two database items, q_u and q_v are two queries, $\langle d_i, q_u\rangle$ and $\langle d_j, q_v\rangle$ are item–query pairs whose relevances are of interest, s_{iu} and s_{jv} are feature vectors that measure the similarities between database item i and query u and between item j and query v, respectively, k_{s} is a kernel on the similarity feature vectors, and k_{q} is a kernel on queries.

The pair kernel defined above says that, when we are predicting the relevance or similarity of an item–query pair according to another (training) pair, we should not only consider the pairs as a whole but also consider the similarity between queries alone. When $k_{\mathrm{q}} \equiv 0$, the pair kernel reduces to the common kernel used for ranking problems in information retrieval. Different implementations of k_{q} lead to different query kernels. Here, we give an implementation of k_{q} as

$$k_{\mathrm{q}}(q_u, q_v) = k_{\mathrm{b}}(B_u, B_v) = k[\Phi(B_u), \Phi(B_v)]\ , \tag{18.27}$$

where B_u and B_v are the data blocks associated with query q_u and query q_v, respectively, $\Phi(B_u)$ and $\Phi(B_v)$ are feature vectors describing block B_u and block B_v, respectively, k_{b} is a kernel defined on blocks, and $k[\Phi(B_u), \Phi(B_v)]$ is a kernel defined on vectors. Further, we define

$$\Phi(B_u) = \langle \mu_{u1}, \sigma_{u1}, \mu_{u2}, \sigma_{u2}, \ldots, \mu_{ud}, \sigma_{ud}\rangle\ , \tag{18.28}$$

where μ_{uk} and σ_{uk} are the mean and the standard deviation, respectively, of the k-th feature in block B_u.

Performance of Pair Kernel

The dataset used to validate the pair kernel is from the ACM KDD Cup 2004 competition [18.51]. ACM KDD Cup is the annual data mining and knowledge discovery competition organized by the ACM special interest group on data mining and knowledge discovery, the leading professional organization of data miners. One of the tasks in KDD Cup 2004 was to predict protein

Table 18.1 Performances of SVM with the common kernel and the pair kernel for protein homology prediction

	TOP1 (maximize)	RKL (minimize)	APR (maximize)	RMS (minimize)
Common kernel	0.8497	54.78	0.8033	0.0373
Pair kernel	0.8497	47.26	0.8253	0.0363

homologies. The best results on this task were mostly obtained with SVMs [18.52–56]. The success of SVMs in the competition demonstrated that kernel methods are among the best methods for solving complicated bioinformatic problems.

In this dataset, homology between a database protein and the query protein is described by 74 features (constituting the input vector s of the k_s kernel). These features were generated by a protein fold recognition program named LOOPP [18.57] and include various scores of sequence alignments, scores of threading features, measures of secondary structure fitness, etc. [18.58]. There are a total of 153 training queries (test queries are without published labels and are not used here). For each query, a data block consisting of about 1000 samples is given. A sample is a 74-dimensional vector measuring the homology between the query and a protein in the database.

Four metrics were used to evaluate the performance of ranking algorithms, including the frequency of a relevant item being ranked highest (TOP1), the average rank of the last relevant item (RKL), the average overall ranking precision (APR), and the average root-mean-square error (RMS). For model selection, the leave-one-block-out strategy was used [18.52]. SVM was used as the learning algorithm. Table 18.1 presents the results obtained with the pair kernel and the common kernel. In both cases, the dot product kernel was used. It is demonstrated that the proposed pair kernel can significantly improve the ranking performance, in comparison with the commonly used kernel. This improvement is owing to the query kernel k_q added to the common kernel. Finally, it should be noted that the pair kernel proposed here is a general scheme, and the constituent query kernel implemented here is only a simple demonstration of this methodology.

18.4 Conclusions

The kernel technique is a powerful tool for constructing new algorithms, especially nonlinear ones. Kernel engineering provides a promising methodology for addressing nonvector data structures. Over the past two decades, kernel methods have become very popular for pattern analysis, especially in the bioinformatics field. It is neither possible nor necessary to enumerate all applications of kernel methods in bioinformatics in this short chapter, because they have actually been applied to almost all bioinformatic problems that one can imagine. This chapter, instead of giving a comprehensive review or focusing on a specific research, presents basic principles of kernel methods and focuses on kernel engineering, in the hope that readers, after reading this chapter, can explore their own kernel methods to address various problems with uncommon data types in bioinformatics.

References

18.1 J. Shawe-Taylor, N. Cristianini: *Kernel Methods for Pattern Analysis* (Cambridge University Press, Cambridge 2004)

18.2 V.N. Vapnik: *The Nature of Statistical Learning Theory* (Springer, Berlin Heidelberg 1995)

18.3 B. Schölkopf, K. Tsuda, J.-P. Vert (Eds.): *Kernel Methods in Computational Biology* (MIT Press, Cambridge 2004)

18.4 A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, G. Rätsch: Support vector machines and kernels for computational biology, PLoS Comput. Biol. **4**(10), e1000173 (2008)

18.5 K.M. Borgwardt: Kernel methods in bioinformatics,. In: *Handbook of Statistical Bioinformatics*, ed. by H.H. Lu, B. Schölkopf, H. Zhao (Springer, Berlin Heidelberg 2011) pp. 317–334

18.6 B. Schölkopf: Support Vector Learning. Dr. Thesis (Technische Universität Berlin, Berlin 1997)

18.7 J. Mercer: Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. R. Soc. **A209**, 415–446 (1909)

18.8 M.A. Aizerman, E.M. Braverman, L.I. Rozonofier: Theoretical foundations of the potential function

method in pattern recognition learning, Autom. Remote Control **25**, 821–837 (1964)
18.9 B.E. Boser, I.M. Guyon, V.N. Vapnik: A training algorithm for optimal margin classifiers, Proc. 5th Annu. ACM Workshop Comput. Learn. Theory (1992) pp. 144–152
18.10 B. Schölkopf, A.J. Smola, K.R. Müller: Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. **10**, 1299–1319 (1998)
18.11 P.L. Lai, C. Fyfe: Kernel and nonlinear canonical correlation analysis, Int. J. Neural Syst. **10**, 365–377 (2000)
18.12 S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller: Fisher discriminant analysis with kernels, Neural Networks for Signal Processing, Vol. 9, ed. by Y.-H. Hu, J. Larsen, E. Wilson, S. Douglas (IEEE, 1999) pp. 41–48
18.13 F.R. Bach, M.I. Jordan: Kernel independent component analysis, J. Mach. Learn. Res. **3**, 1–48 (2003)
18.14 B. Schölkopf: The kernel trick for distances. In: *Advances in Neural Information Processing Systems*, Vol. 13, ed. by T.K. Leen, T.G. Dietterich, V. Tresp (MIT Press, Cambridge, MA 2001) pp. 301–307
18.15 K. Yu, L. Ji, X. Zhang: Kernel nearest-neighbor algorithm, Neural Process. Lett. **15**, 147–156 (2002)
18.16 J. Peng, D.R. Heisterkamp, H.K. Dai: Adaptive quasiconformal kernel nearest neighbor classification, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 656–661 (2004)
18.17 Y. Fu: Machine Learning Based Bioinformation Retrieval, Dissertation (Chinese Academy of Sciences 2007)
18.18 J. Xu, H. Li, C. Zhong: Relevance ranking using kernels, Proc. 6th Asian Inf. Retr. Soc. Symp. (2010) pp. 1–12
18.19 W. Wu, J. Xu, H. Li, S. Oyama: Learning a robust relevance model for search using kernel methods, J. Mach. Learn. Res. **12**, 1429–1458 (2011)
18.20 H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins: Text classification using string kernels, J. Mach. Learn. Res. **2**, 419–444 (2002)
18.21 C. Watkins: Dynamic alignment kernels,. In: *Advances in Large Margin Classifiers*, ed. by A.J. Smola, P.L. Bartlett, B. Schölkopf, D. Schuurmans (MIT, Cambridge 1999) pp. 39–50
18.22 D. Haussler: Convolution kernels on discrete structures, Technical Report UCSC-CRL-99-10 (1999)
18.23 T.S. Jaakkola, D. Haussler: Exploiting generative models in discriminative classifiers,. In: *Advances in Neural Information Processing Systems*, Vol. 11, ed. by M.S. Kearns, S.A. Solla, D.A. Cohn (MIT Press, Cambridge, MA 1999) pp. 487–493
18.24 R.I. Kondor, J. Lafierty: Diffusion kernels on graphs and other discrete input spaces, Proc. 9th Int. Conf. Mach. Learn. (2002) pp. 315–322
18.25 C. Leslie, E. Eskin, W.S. Noble: The spectrum kernel: A string kernel for SVM protein classification, Proc. 7th Pac. Sympos. Biocomput. (2002) pp. 564–575
18.26 C. Leslie, E. Eskin, J. Weston, W.S. Noble: Mismatch string kernels for discriminative protein classification, Bioinformatics **20**(4), 467–476 (2004)
18.27 J.P. Vert: Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings, (2002) pp. 649–660
18.28 A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, K.-R. Muller: Engineering support vector machine kernels that recognize translation initiation sites, Bioinformatics **16**, 799–807 (2000)
18.29 T. Jaakkola, M. Diekhans, D. Haussler: A discriminative framework for detecting remote protein homologies, J. Comput. Biol. **7**, 95–114 (2000)
18.30 J.P. Vert, M. Kanehisa: Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In: *Advances in Neural Information Processing Systems*, Vol. 15, ed. by S. Becker, S. Thrun, K. Obermayer (MIT Press, Cambridge, MA 2003) pp. 1425–1432
18.31 J.P. Vert: A tree kernel to analyze phylogenetic profiles, Bioinformatics **18**, S276–S284 (2002)
18.32 K.M. Borgwardt, C.S. Ong, S. Schönauer, S.V.N. Vishwanathan, A.J. Smola, H. Kriegel: Protein function prediction via graph kernels, Bioinformatics **1**(Suppl.), 47–56 (2005)
18.33 N. Cristianini, J. Shawe-Taylor: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (University Press, Cambridge 2000)
18.34 G. Schweikert, A. Zien, G. Zeller, J. Behr, C. Dieterich, C.S. Ong, P. Philips, F. De Bona, L. Hartmann, A. Bohlen, N. Krüger, S. Sonnenburg, G. Rätsch: mGene: Accurate SVM-based gene finding with an application to nematode genomes, Genome Res. **19**(11), 2133–2143 (2009)
18.35 I. Guyon, J. Weston, S. Barnhill, V. Vapnik: Gene selection for cancer classification using support vector machines, Mach. Learn. **46**, 389–422 (2002)
18.36 Y. Sun, X. Fan, Y. Li: Identifying splicing sites in eukaryotic RNA: Support vector machine approach, Comput. Biol. Med. **33**(1), 17–29 (2003)
18.37 J. Gubbi, A. Shilton, M. Palaniswami: Kernel methods in protein structure prediction. In: *Machine Learning in Bioinformatics*, ed. by Y.-Q. Zhang, J.C. Rajapakse (Wiley, Hoboken 2008)
18.38 G.R.G. Lanckriet, M. Deng, N. Cristianini, M.I. Jordan, W.S. Noble: Kernel-based data fusion and its application to protein function prediction in yeast, Proc. 9th Pac. Symp. Biocomput. (2004) pp. 300–311
18.39 H. Wang, Y. Fu, R. Sun, S. He, R. Zeng, W. Gao: An SVM scorer for more sensitive and reliable peptide identification via tandem mass spectrometry, Proc. 11th Pac. Symp. Biocomput. (2006) pp. 303–314
18.40 Y. Li, P. Hao, S. Zhang, Y. Li: MoI cell proteomics, feature-matching pattern-based support vector

machines for robust peptide mass fingerprinting, Mol. Cell. Proteomic. **10**(12), M110.0057852011 (2011)

18.41 R. Aebersold, M. Mann: Mass spectrometry-based proteomics, Nature **422**, 198–207 (2003)

18.42 H. Steen, M. Mann: The ABC's (and XYZ's) of peptide sequencing, Nat. Rev. Mol. Cell **5**, 699–711 (2004)

18.43 J.K. Eng, A.L. McCormack, J.R. Yates: An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database, J. Am. Soc. Mass Spectrom. **5**, 976–989 (1994)

18.44 K.X. Wan, I. Vidavsky, M.L. Gross: Comparing similar spectra: From similarity index to spectral contrast angle, J. Am. Soc. Mass Spectrom. **13**, 85–88 (2002)

18.45 B. Schölkopf, P. Simard, A. Smola, V. Vapnik: Prior knowledge in support vector kernels, Adv. Neur. Inf. Proces. Syst., Vol. 10, ed. by M. Jordan, M. Kearns, S. Solla (MIT, Cambridge 1998) pp. 640–646

18.46 Y. Fu, Q. Yang, R. Sun, D. Li, R. Zeng, C.X. Ling, W. Gao: Exploiting the kernel trick to correlate fragment ions for peptide identification via tandem mass spectrometry, Bioinformatics **20**, 1948–1954 (2004)

18.47 D. Li, Y. Fu, R. Sun, C. Ling, Y. Wei, H. Zhou, R. Zeng, Q. Yang, S. He, W. Gao: pFind: A novel database-searching software system for automated peptide and protein identification via tandem mass spectrometry, Bioinformatics **21**, 3049–3050 (2005)

18.48 A. Keller, S. Purvine, A.I. Nesvizhskii, S. Stolyar, D.R. Goodlett, E. Kolker: Experimental protein mixture for validating tandem mass spectral analysis, Omics **6**, 207–212 (2002)

18.49 Y. Zhang: Progress and challenges in protein structure prediction, Curr. Opin. Struct. Biol. **18**, 342–348 (2008)

18.50 T. Liu: *Learning to Rank for Information Retrieval* (Springer, New York 2011)

18.51 R. Caruana, T. Joachims, L. Backstrom: KDD Cup 2004: Results and analysis, SIGKDD Explorations **6**, 95–108 (2004)

18.52 Y. Fu, R. Sun, Q. Yang, S. He, C. Wang, H. Wang, S. Shan, J. Liu, W. Gao: A block-based support vector machine approach to the protein homology prediction task in KDD Cup 2004, SIGKDD Explorations **6**, 120–124 (2004)

18.53 C. Foussette, D. Hakenjos, M. Scholz: KDD-Cup 2004 – Protein homology task, SIGKDD Explorations **6**, 128–131 (2004)

18.54 B. Pfahringer: The Weka solution to the 2004 KDD cup, SIGKDD Explorations **6**, 117–119 (2004)

18.55 Y. Tang, B. Jin, Y. Zhang: Granular support vector machines with association rules mining for protein homology prediction, Artif. Intell. Med. **35**, 121–134 (2005)

18.56 Y. Fu, R. Pan, Q. Yang, W. Gao: Query-adaptive ranking with support vector machines for protein homology prediction. In: *ISBRA 2011*, Lecture Notes in Bioinformatics, Vol. 6674, ed. by J. Chen, J. Wang, A. Zelikovsky (Springer, Berlin Heidelberg 2011) pp. 320–331

18.57 D. Tobi, R. Elber: Distance dependent, pair potential for protein folding: Results from linear optimization, Proteins Struct. Funct. Genet. **41**, 40–46 (2000)

18.58 O. Teodorescu, T. Galor, J. Pillardy, R. Elber: Enriching the sequence substitution matrix by structural information, Proteins Struct. Funct. Bioinform. **54**, 41–48 (2004)