1 Convolution and local alignment kernels

Jean-Philippe Vert
Hiroto Saigo
Tatsuya Akutsu

We present a family of kernels for strings based on the detection of local alignments, a widely used principle to compare biological sequences. These kernels belong to the family of convolution kernels introduced by Haussler (1999), and can be implemented with dynamic programming. We highlight the relations between these kernels and sequence alignment scores used in bioinformatics, and propose a solution to overcome the issue of diagonal dominance they suffer from. When tested on their ability to detect protein homology at the superfamily level on a benchmark data set, they outperform other state-of-the-art methods.

1.1 Introduction

With the advent of high-throughput sequencing technologies, biological sequences are accumulating at an unprecedented pace in databases. The need to analyze, compare, and annotate these sequences is more than ever a central problem in postgenomics. There has recently been a growing interest in the use of kernel methods to process sequences, particularly for classification of sequences with Support Vector Machines (SVMs). This trend has been accompanied by a growing interest in the development of kernel functions for strings, which form the basic ingredient that any kernel method is based on. Examples of string kernels include the Fisher kernel (Jaakkola et al., 2000), spectrum kernel (Leslie et al., 2002), mismatch kernel (Leslie et al., 2003), pairwise kernel (Liao and Noble, 2002), and the string kernel proposed by Lodhi et al. (2002).

A kernel function can often be thought of as a measure of similarity. Different kernels correspond to different notions of similarity. For example, each string kernel mentioned above corresponds to one notion of similarity between strings, such as the similarity of Fisher score vectors with respect to a hidden Markov model (HMM) for

the Fisher kernel, or the similarity of the frequency of small blocks in the sequences for the spectrum kernel.

Independent of kernel methods, the problem of assessing the similarity between biological sequences has been the object of much investigation in computational biology over the last three decades. The notion of local alignments (which we review below) has emerged as a powerful approach to detecting evolutionary relationships between sequences, and the measure of the best local alignment with the Smith-Waterman algorithm (Smith and Waterman, 1981) or its fast heuristics gapped BLAST, PSI-BLAST (Altschul et al., 1997), and FASTA (Pearson, 1990) is nowadays the method of choice to measure the similarity between sequences.

The main motivation behind this work is the observation that the notions of similarity between sequences defined by the string kernels developed so far differ considerably from the notion of similarity based on local alignment. This suggests that biologically more relevant kernels might be defined if the notion of similarity they induce are more similar to those found useful by the computational biology community, namely the similarity based on local alignment scoring.

The most direct way to reconcile kernel methods and classic measures of similarity for biological sequences would be to consider a classic measure of similarity such as the Smith-Waterman score as a kernel. However, it turns out that this choice is not a valid kernel because it violates the condition of positive definiteness for certain choices of parameters, as we show in the experimental part. Still, following the avenue paved by Haussler (1999) and Watkins (2000), we propose a family of string kernels based on the scoring of local alignments. Two strings are similar with these kernels when they have many high-scoring local alignments, which is the property that motivates this work.

We present promising results on a benchmark problem of remote homology detection, where the new kernels outperform all other string kernels tested. However, this performance is only obtained at the price of performing an operation on the kernel values, which otherwise can be exponentially small even for sequences with meaningful similarities.

The chapter is organized as follows. In sections 1.2 and 1.3 we recall the two main ingredients we need, namely the classic notions of local alignment and Smith-Waterman score on the one hand, and the idea of convolution kernels on the other hand. In section 1.4 we introduce a family of convolution kernels called local alignment kernels and present some of their properties. Their implementation is discussed in section 1.5 and the issue of diagonal dominance is raised in section 1.6. We conclude with experimental results on the problem of detecting protein remote homology in sections 1.7 and 1.8.

1.2 Sequence Local Alignment

In this section we introduce basic notations and recall the classic notion of sequence alignment and Smith-Waterman local alignment score.

1.2.1 Basic Notations

Let $\mathcal{A}$ be a finite set called the *alphabet*, typically the set of 20 amino acids or of 4 nucleotides {A,C,G,T}. Elements of the alphabet are called *letters*. A *string* of length $n \geq 0$ is a concatenation of n letters, $\mathbf{x} = x_1 \dots x_n$. The empty string (of length 0) is denoted by ϵ, and the set of all strings is denoted by $\mathcal{X} = \{\epsilon\} \cup \bigcup_{n=1}^{\infty} \mathcal{A}^n$. The length of a string $\mathbf{x}$ is denoted $|\mathbf{x}|$. The concatenation of two strings $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ is written $\mathbf{xy}$.

1.2.2 Alignments and Smith-Waterman Score

With these basic notations we can introduce the notion of alignment that plays an essential role below:

Sequence alignment

Definition 1.1 *An* alignment (with gaps) π *of* $p \geq 0$ *positions between two sequences* $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ *is a pair of* p*-tuples:*

$$\pi = ((\pi_1(1), \dots, \pi_1(p)), (\pi_2(1), \dots, \pi_2(p))) \in \mathbb{N}^{2p}$$

that satisfies

$$1 \leq \pi_1(1) < \pi_1(2) < \dots < \pi_1(p) \leq |\mathbf{x}|,$$
$$1 \leq \pi_2(1) < \pi_2(2) < \dots < \pi_2(p) \leq |\mathbf{y}|.$$

The simple idea behind this formal definition is that the alignment encodes p positions in each sequence that are aligned to each other, that is, the $\pi_1(i)$th letter of $\mathbf{x}$ is aligned to the $\pi_2(i)$th letter of $\mathbf{y}$ for $i = 1, \dots, p$. A convenient way to represent an alignment is to write the two sequences one above the other, by placing the p aligned positions on the same columns, and inserting the symbol '$-$' in both sequences whenever necessary to ensure that only aligned letters are on the same column (this representation is unique up to the choice of the position where the symbol '-' is inserted between two consecutively aligned positions).

As an example, if $\mathbf{x}$ = `GAATCCG` and $\mathbf{y}$ = `GATTGC`, then the 4-letter alignment $\pi = ((1, 2, 4, 6), (1, 3, 4, 5))$ is represented as follows:

```
G-AATCCG-
GAT-T-G-C
```

q The notion of alignment stems from the observation that the transformation of sequences during evolution can roughly be described by substitutions of single

letters, deletions, or insertions. As a result, a natural way to compare homologous proteins is to align them in order to detect the conserved regions.

Let us denote by $\Pi(\mathbf{x}, \mathbf{y})$ the set of all possible alignments between two sequences $\mathbf{x}$ and $\mathbf{y}$, and by $|\pi|$ the number of positions aligned by the alignment $\pi \in \Pi(\mathbf{x}, \mathbf{y})$. In order to evaluate how "good" an alignment $\pi \in \Pi(\mathbf{x}, \mathbf{y})$ is, and eventually find the best one, various scoring schemes have been developed. We now present one of the most widely used scoring schemes, often referred to as *local alignment score*. It is defined in terms of a substitution matrix $S \in \mathbb{R}^{\mathcal{A} \times \mathcal{A}}$ and a gap penalty function $g : \mathbb{N} \to \mathbb{R}$ such that $g(0) = 0$ as follows:

Definition 1.2 *The local alignment score of an alignment $\pi \in \Pi(\mathbf{x}, \mathbf{y})$ is equal to*

$$s_{S,g}(\pi) := \sum_{i=1}^{|\pi|} S\left(x_{\pi_1(i)}, y_{\pi_2(i)}\right) - \sum_{i=1}^{|\pi|-1} \left[g(\pi_1(i+1) - \pi_1(i)) + g(\pi_2(i+1) - \pi_2(i))\right]. \tag{1.1}$$

In other words, the local alignment score of an alignment π is the sum of the substitution scores between the letters at the aligned positions, minus the sum of the gap penalty when '-' symbols must be inserted between aligned positions. The name "local alignment" comes from the fact that no gap penalty is counted before the first and after the last aligned letters.

We can now define the local alignment score between sequences:

Smith-Waterman score

Definition 1.3 *The local alignment score or Smith-Waterman score (denoted SW score below) between two sequences $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ is the local alignment score of their best alignment, that is:*

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} s_{S,g}(\pi). \tag{1.2}$$

The SW score is a widely used measure of similarity between DNA or protein sequences. It can be computed with a complexity $O(|\mathbf{x}||\mathbf{y}|)$ by dynamic programming with the Smith-Waterman algorithm (Smith and Waterman, 1981), and lower complexity heuristics are implemented in the widely used softwares gapped BLAST (Altschul et al., 1997) and FASTA (Pearson, 1990).

1.2.3 Is the SW Score a Valid String Kernel?

A natural question at this point is the following: is the SW score (1.2) a valid kernel for string? Remember that a function $k : \mathcal{X}^2 \to \mathbb{R}$ is a valid kernel if it is symmetric, that is, $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$ for any $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$, and positive definite, that is:

$$\forall n \in \mathbb{N}, \forall (\mathbf{x}_1, \ldots, \mathbf{x}_n, a_1, \ldots, a_n) \in \mathcal{X}^n \times \mathbb{R}^n, \quad \sum_{i,j=1}^{n} a_i a_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{1.3}$$

String kernels

We simply call such functions *string kernels* below. To the best of our knowledge, there is no general result that states for which parameters S and g the local

alignment score $SW_{S,g}$ is a string kernel. As the following toy example shows, there are cases where the SW score is a string kernel:

Proposition 1.4 *Let $g = 0$ (no gap penalty) and S be a substitution matrix null except for one letter $a \in \mathcal{A}$ on the diagonal, that is, $S(a,a) = 1$ and $S(u,v) = 0$ except if $u = v = a$. Then the score $SW_{S,g}$ is a string kernel.*

Proof Consider two sequences $\mathbf{x}$ and $\mathbf{y}$ with, respectively, m_x and m_y occurrences of the letter a. With no gap penalty and no contribution to the score of aligned letters except for pairs of a, the highest score is obtained when as many a as possible are aligned, that is, $\min(m_x, m_y)$. Each aligned pair of a adds 1 to the score, which shows that

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) = \min(m_x, m_y).$$

This is clearly a valid kernel, as it can be written as an inner product $\phi(\mathbf{x}).\phi(\mathbf{y})$ between two vector representations of $\mathbf{x}$ and $\mathbf{y}$, for instance, by the mapping $\phi : \mathcal{X} \to \mathbb{R}^{\mathbb{N}}$ defined by $\phi(\mathbf{x})_i = 1$ if $i \leq m_x$, 0 otherwise. ■

On the other hand, for more classic parameter S and g, we observe empirically (see section 1.7) that the SW score can lack positive definiteness. This motivates the study of convolution kernels in the next section, and the development of string kernels based on local alignment scores in section 1.4.

1.3 Convolution Kernels

In this section we recall the notion of convolution kernels for strings introduced in a more general framework by Haussler (1999). Roughly speaking, convolution is a powerful operation to combine two string kernels into a single one, particularly suited to the detection of local similarities between sequences. More formally, they are defined as follows:

Convolution kernel

Definition 1.5 *Let k_1 and k_2 be two functions $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The convolution of k_1 and k_2, denoted by $k_1 \star k_2$, is the following function:*

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \quad k_1 \star k_2(\mathbf{x}, \mathbf{y}) := \sum_{\mathbf{x}_1\mathbf{x}_2 = \mathbf{x}, \mathbf{y}_1\mathbf{y}_2 = \mathbf{y}} k_1(\mathbf{x}_1, \mathbf{y}_1) k_2(\mathbf{x}_2, \mathbf{y}_2). \tag{1.4}$$

Haussler (1999) proved the following important result:

Theorem 1.6 *The convolution of two string kernels is a string kernel.*

This theorem implies that we can build complex string kernels by convolving simple ones. Moreover, it can easily be checked that the convolution operation is

associative, and that the convolution of p kernels $k_1, \cdots, k_p$ is the following string kernel:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \quad k_1 \star \cdots \star k_p(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x}_1 \cdots \mathbf{x}_p = \mathbf{x}, \mathbf{y}_1 \cdots \mathbf{y}_p = \mathbf{y}} k_1(\mathbf{x}_1, \mathbf{y}_1) \cdots k_p(\mathbf{x}_p, \mathbf{y}_p).$$

Convolution kernels were introduced by Haussler (1999) in a more general context than string kernels, and were, for instance, successfully applied to define kernels for trees with applications in natural language processing (Collins and Duffy, 2002). In the context of biological sequences, Haussler (1999) showed that the probability

Pair HMM

$P(\mathbf{x}, \mathbf{y})$ of a pair of sequences under a pair HMM (Durbin et al., 1998) is a convolution of basic string kernel and is therefore a valid string kernel, under some assumptions on the model parameters. This result, which was proved independently by Watkins (2000), makes a first link between string kernels and string alignment scores. Indeed the probability $P(\mathbf{x}, \mathbf{y})$ studied by Haussler (1999) and Watkins (2000) is related to the score obtained by global alignment of two strings with the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), at least if the probability is replaced by a log-odds score which takes the form

$$s(\mathbf{x}, \mathbf{y}) = \log \frac{P(\mathbf{x}, \mathbf{y})}{q(\mathbf{x}) q(\mathbf{y})},$$

where q is a background model (Durbin et al., 1998).

The results, however, are of little practical interest for at least two reasons. First, it is commonly recognized in computational biology that local alignment scores like the SW score are more relevant than global alignment scores to assess the similarity between sequences. Second, the probabilities of two sequences under a pair HMM model are usually much too small (typically 10^{-100} to 10^{-1000}) to be of any interest as kernels. The objects of the next two sections are to overcome both these issues, first by proposing a convolution kernel based on local alignment score (section 1.4), and then on making it useful in real-world application (section 1.6).

1.4 Local Alignment Kernels

Following the approach pioneered by Haussler (1999), we define in this section a complex string kernel by convolution and sum of simple kernels. Let us therefore start by defining three basic string kernels, which will serve as "basic blocks" to derive a more complex kernel. To do so, we assume a substitution matrix S and a gap penalty function g are given, and we define the three kernels as functions of S and g.

The first basic kernel, denoted by k_0, is a trivial kernel constantly equal to 1, that is:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \quad k_0(\mathbf{x}, \mathbf{y}) := 1. \tag{1.5}$$

This kernel is useful to compare the parts of two sequences that don't contribute to the local alignment score, that is, the substrings before and after the alignment.

The second basic kernel, which we call k_a, is used to quantify the similarity of aligned letters. It is defined as follows:

$$\forall (\mathbf{x},\mathbf{y}) \in \mathcal{X}^2, \quad k_a^{(\beta)}(\mathbf{x},\mathbf{y}) := \begin{cases} 0 & \text{if } |\mathbf{x}| \neq 1 \text{ or } |\mathbf{y}| \neq 1, \\ \exp\left(\beta S(\mathbf{x},\mathbf{y})\right) & \text{otherwise,} \end{cases} \tag{1.6}$$

where $\beta \geq 0$ is a parameter. Observe that it is null as soon as one of the sequences is not exactly reduced to a single letter. Observe also that this is only a string kernel for the values of β which ensure that the matrix $\left(\exp\left(\beta S(a,b)\right)\right)_{a,b\in\mathcal{A}}$ is positive definite. This is the case whatever $\beta \geq 0$ iff the matrix $\left(S(a,b)\right)_{a,b\in\mathcal{A}}$ is conditionally positive definite (Berg et al., 1984), which can be checked case by case (this holds in particular if the matrix $\left(S(a,b)\right)_{a,b\in\mathcal{A}}$ is positive definite).

Finally, to translate the penalty gap model, we define the third basic string kernel, which we denote k_g, as follows:

$$\forall (\mathbf{x},\mathbf{y}) \in \mathcal{X}^2, \quad k_g^{(\beta)}(\mathbf{x},\mathbf{y}) := \exp\left[\beta\left(g(|\mathbf{x}|) + g(|\mathbf{y}|)\right)\right], \tag{1.7}$$

where $\beta \geq 0$ is the same parameter as in (1.6), and g is the gap penalty function. k_g is a valid string kernel, because it can be written as a scalar product $k_g^{(\beta)}(\mathbf{x},\mathbf{y}) = \phi_g^{(\beta)}(\mathbf{x}).\phi_g^{(\beta)}(\mathbf{y})$ between 1-dimensional vectors given by $\phi_g^{(\beta)}(\mathbf{x}) = \exp\left(\beta g(|\mathbf{x}|)\right)$.

Let us now combine these three basic kernels by convolution. For any fixed $n > 0$, consider first the following string kernel:

$$k_{(n)}^{(\beta)} := k_0 \star \left(k_a^{(\beta)} \star k_g^{(\beta)}\right)^{(n-1)} \star k_a^{(\beta)} \star k_0.$$

This kernel quantifies the similarity between two strings $\mathbf{x}$ and $\mathbf{y}$ based on local alignments of exactly n residues. Indeed the convolution operation sums up the contributions of all possible decompositions of $\mathbf{x}$ and $\mathbf{y}$ simultaneously into an initial part (whose similarity is measured by k_0), a succession of n aligned residues (whose similarity is measured by $k_a^{(\beta)}$) possibly separated by $n-1$ gaps (whose similarity is measured by $k_g^{(\beta)}$), and a terminal part (whose similarity is measured by k_0). For $n = 0$ (no residue aligned), we simply define the kernel $k_{(0)}^{(\beta)} = k_0$.

In order to compare two sequences through all possible local alignments, it is necessary to take into account alignments with different numbers n of aligned residues. A simple solution is to sum up the contributions of all kernels $k_{(n)}^{(\beta)}$ for $n \geq 0$, which results in the following *local alignment kernel* (which we call LA kernel below):

Local alignment kernel

$$k_{LA}^{(\beta)} := \sum_{i=0}^{\infty} k_{(i)}^{(\beta)}. \tag{1.8}$$

Note that for any pair of finite-length sequences $(\mathbf{x},\mathbf{y}) \in \mathcal{X}^2$ the right-hand term of (1.8) estimated at $(\mathbf{x},\mathbf{y})$ is a convergent series, because it has only a finite number of non-null terms. Therefore $k_{LA}^{(\beta)}$ is defined as a pointwise limit of string kernels,

and is therefore a string kernel by closure property of the class of positive definite kernels under pointwise limit (Berg et al., 1984).

The following theorem, whose proof is postponed to appendix A, highlights the relations between the LA string kernel, the local alignment scores of all possible alignments, and the SW score:

Theorem 1.7 *The LA kernel (1.8) is expressed in terms of local alignment scores as follows:*

$$\forall (\mathbf{x},\mathbf{y}) \in \mathcal{X}^2, \quad k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s_{S,g}(\mathbf{x},\mathbf{y},\pi)\right). \tag{1.9}$$

The SW score (1.2) is related to the LA kernel by the following equality:

$$\forall (\mathbf{x},\mathbf{y}) \in \mathcal{X}^2, \quad \lim_{\beta \to +\infty} \frac{1}{\beta} \ln k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = SW_{S,g}(\mathbf{x},\mathbf{y}). \tag{1.10}$$

These equations clarify the link between the LA kernel (1.8) and the SW score (1.2), and highlight why the SW score might fail to be a valid string kernel. First, the SW score only keeps the contribution of the best local alignment to quantify the similarity between two sequences, instead of summing up the contributions of all possible local alignments like the LA kernel does. Second, the SW score is the logarithm of this alignment score, and taking the logarithm is usually an operation which does not preserve the property of being positive definite (Berg et al., 1984).

1.5 Kernel Computation

A naive computation of the LA kernel using (1.9) would require a sum over $|\Pi(\mathbf{x},\mathbf{y})|$ alignments, resulting in a complexity exponential with respect to $|\mathbf{x}|$ and $|\mathbf{y}|$. In this section we show that this expression can be factorized and computed using dynamic programming by a slight modification of the Smith-Waterman algorithm, at least if the gap penalty function is affine. This efficient computation is summarized in the following theorem:

Theorem 1.8 *Let $(\mathbf{x},\mathbf{y}) \in \mathcal{X}^2$ be two strings, and g be an affine gap penalty function:*

$$\begin{cases} g(0) & = 0, \\ g(n) & = d + e(n-1) \text{ if } n \geq 1, \end{cases} \tag{1.11}$$

then the LA kernel $k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is equal to

$$k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = 1 + X_2(|\mathbf{x}|,|\mathbf{y}|) + Y_2(|\mathbf{x}|,|\mathbf{y}|) + M(|\mathbf{x}|,|\mathbf{y}|),$$

where $M(i,j), X(i,j), Y(i,j), X_2(i,j)$, *and* $Y_2(i,j)$ *for* $0 \leq i \leq |\mathbf{x}|$, *and* $0 \leq j \leq |\mathbf{y}|$ *are recursively defined by the following equations:*

$$\begin{cases} M(i,0) = M(0,j) = 0, \\ X(i,0) = X(0,j) = 0, \\ Y(i,0) = Y(0,j) = 0, \\ X_2(i,0) = X_2(0,j) = 0, \\ Y_2(i,0) = Y_2(0,j) = 0, \end{cases}$$

Dynamic programming

and for $i = 1, \ldots, |\mathbf{x}|$ *and* $j = 1, \ldots, |\mathbf{y}|$*:*

$$\begin{cases} M(i,j) & = \exp(\beta S(x_i, y_j)) \\ & \qquad [1 + X(i-1,j-1) + Y(i-1,j-1) + M(i-1,j-1)], \\ X(i,j) & = \exp(\beta d) M(i-1,j) + \exp(\beta e) X(i-1,j), \\ Y(i,j) & = \exp(\beta d) \left[M(i,j-1) + X(i,j-1)\right] + \exp(\beta e) Y(i,j-1), \\ X_2(i,j) & = M(i-1,j) + X_2(i-1,j), \\ Y_2(i,j) & = M(i,j-1) + X_2(i,j-1) + Y_2(i,j-1). \end{cases}$$

The proof of theorem 1.8 is postponed to appendix B. One can observe that the Smith-Waterman algorithm (Smith and Waterman, 1981) to compute the SW score is obtained from these equations by replacing each addition by a max operation, and taking the logarithm of the result divided by β.

Rational kernels

An equivalent way to describe the implementation of the LA kernel is through the weighted finite-state transducer (WFST) shown in figure 1.1 (see the legend for an explanation). The implementation as a WFST is possible because each basic kernel we defined can be implemented using such automata, and because the addition and convolution of WFST can be computed by WFST. Such kernels are called rational kernels (Cortes et al., 2003).

1.6 Diagonal Dominance Issue

In many cases of practical interest the LA kernel defined in (1.8) suffers from the diagonal dominance problem, namely the fact that $k_{LA}^{(\beta)}(\mathbf{x},\mathbf{x})$ is easily orders of magnitudes larger than $k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y})$ for two different sequences $\mathbf{x}$ and $\mathbf{y}$, even though $\mathbf{x}$ and $\mathbf{y}$ might share interesting similarities. This is particularly evident for increasing values of the parameter β, because from (1.10)

$$\frac{k_{LA}^{(\beta)}(\mathbf{x},\mathbf{x})}{k_{LA}^{(\beta)}(\mathbf{x},\mathbf{x})} \sim \exp \beta \left(SW_{S,g}(\mathbf{x}, xb) - SW_{S,g}(\mathbf{x},\mathbf{y})\right)$$

when $\beta \to \infty$. In practice, it has been observed that SVMs don't perform well in this situation. Indeed diagonal dominance implies that different objects are almost orthogonal in the feature space, and it can be shown that instead of learning a

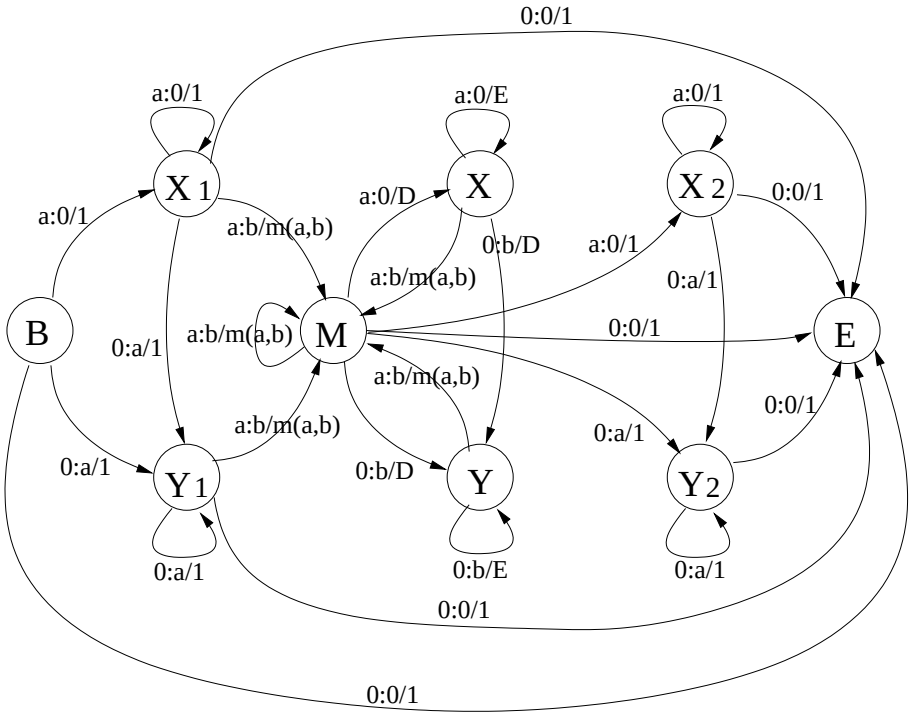

Figure 1.1 A weighted finite-state transducer is a directed graph with an initial state B and a terminal state E. To each edge is attached a label of the form $u : v/w$. The label corresponds to an action of processing an input sequence $\mathbf{x}$ and an output sequence $\mathbf{y}$, and a weight associated with the action. The action of a label $u : v/w$ is "read nothing (if $u = 0$) or one letter $a \in \mathcal{A}$ (if $u = a$) in $\mathbf{x}$, and output nothing (if $v = 0$) or one letter $b \in \mathcal{A}$ (if $v = b$) in $\mathbf{y}$". The weight associated with the action is w, with the conventions that $m(a, b) = \exp(\beta S(a, b))$, $D = \exp(\beta d)$, and $E = \exp(\beta e)$. A path from B to E is *compatible* with two sequences $\mathbf{x}$ and $\mathbf{y}$ if the successive actions performed on the edges along the path correspond to reading $\mathbf{x}$ and outputting $\mathbf{y}$. The score of a compatible path is the product of its edge weights. The kernel $k_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y})$ is computed by the automaton shown on this picture as the sum of the scores of all compatible paths with $\mathbf{x}$ and $\mathbf{y}$.

classifier with good generalization performances, an SVM tends only to memorize the data in that case.

The exponential dependency of the LA kernel with respect to the β parameters suggests the following normalization, in order to remove the diagonal dominance when β increases:

$$\tilde{k}_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}) = \frac{1}{\beta} \ln k_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}). \tag{1.12}$$

An obvious problem with this operation is that the logarithm of a positive definite kernel is not a positive definite kernel in general (Berg et al., 1984). Still $\tilde{k}_{LA}^{(\beta)}$ has several interesting properties. It increases monotonically with $k_{LA}^{(\beta)}$, and by (1.9) it is always non-negative, because at least one local alignment between any two sequences has a non-negative score (at worst the score of aligning no residue is null). Besides, by (1.10), $\tilde{k}_{LA}^{(\beta)}$ behaves like the SW score for large β. Intuitively, $\tilde{k}_{LA}^{(\beta)}$ is therefore of the order of magnitude of the SW score but includes contributions from all possible local alignments.

Because $\tilde{k}_{LA}^{(\beta)}$ might not be a positive definite kernel, some care must be taken to ensure that the SVM converges to a large margin discrimination rule during learning. We tested two approaches to make the symmetric function $\tilde{k}_{LA}^{(\beta)}$ positive definite on a given training set of sequences, which we now describe.

Spectral translation

The first approach we propose is to add to the diagonal of the training Gram matrix a non-negative constant large enough to make it positive definite. In all experiments presented below we perform this operation by subtracting from the diagonal the smallest negative eigenvalue of the training Gram matrix, if there are negative eigenvalues. The resulting kernel, which we call LA-eig, is equal to $\tilde{k}_{LA}^{(\beta)}$ except eventually on the diagonal.

Empirical kernel map

We compare this approach to the method proposed in Schölkopf et al. (2002) which consists in working with the *empirical kernel map*. In this case, for a given training set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of sequences, each possible sequence $\mathbf{x}$ is mapped to the n-dimensional vector $\left(\tilde{k}_{LA}^{(\beta)}(\mathbf{x}, \mathbf{x}_1), \ldots, \tilde{k}_{LA}^{(\beta)}(\mathbf{x}, \mathbf{x}_n)\right)^\top$. These vector representations are then used to train the SVM and predict the class of new sequences. The corresponding kernel between two sequences $\mathbf{x}$ and $\mathbf{y}$, which we call the LA-ekm kernel, is therefore equal to $\sum_{i=1}^n \tilde{k}_{LA}^{(\beta)}(\mathbf{x}, \mathbf{x}_i) \tilde{k}_{LA}^{(\beta)}(\mathbf{y}, \mathbf{x}_i)$. On the training set, this amounts to taking $\left(\tilde{k}_{LA}^{(\beta)}\right)^\top . \tilde{k}_{LA}^{(\beta)}$ as a new symmetric positive definite Gram matrix.

1.7 Experiments

The accuracy of an SVM combined with the LA-eig and LA-ekm kernels is evaluated with the benchmark procedure used in Liao and Noble (2002). Three other state-of-the-art methods were also run on this benchmark, in order to compare them with the method proposed in this chapter: SVM with a Fisher kernel (Jaakkola et al.,

2000), SVM-pairwise (Liao and Noble, 2002), and SVM with a mismatch kernel (Leslie et al., 2003).

Remote homology detection

The problem is to classify protein domains into superfamilies in the structural classification of proteins (SCOP) (Murzin et al., 1995) version 1.53. The data, already used in Liao and Noble (2002),[1] consists of 4352 sequences extracted from the Astral database (similar sequences were removed with an E-value threshold of 10^{-25}), grouped into families and superfamilies. For each family, the protein domains within the family are considered positive test examples, and protein domains within the superfamily but outside the family are considered positive training examples. This yields 54 families with at least 10 positive training examples and 5 positive test examples. Negative examples for the family are chosen from outside of the positive sequences' fold, and were randomly split into training and test sets in the same ratio as the positive examples. For more details on this data set and the experiments, the reader is referred to Liao and Noble (2002) and Jaakkola et al. (2000).

To measure the quality of the methods, the receiver operating characteristic (ROC) scores, the ROC_{50} scores, and the median rate of false positives (RFP) are used. The ROC score is the normalized area under a curve that plots true positives against false positives for different possible thresholds for classification (Gribskov and Robinson, 1996). The ROC_{50} is the area under the ROC curve up to 50 false positives, and is considered a useful measure of performance for real-world application. The median RFP is the number of false positives scoring as high or better than the median-scoring true positives. These measures were used for evaluation in Jaakkola et al. (2000) and Liao and Noble (2002).

1.7.1 SVM

All methods based on SVM were tested with a common procedure, which we now describe. We used the Gist publicly available SVM software[2] which implements the soft margin optimization algorithm described in Jaakkola et al. (2000). For each kernel k to be tested, the following systematic preprocessing was performed. First, all points were normalized to unit norm in the feature space and separated from the origin by adding a constant to the kernel, resulting in the following kernel:

$$k_{norm}(\mathbf{x}, \mathbf{y}) = \frac{k(\mathbf{x}, \mathbf{y})}{\sqrt{k(\mathbf{x}, \mathbf{x})k(\mathbf{y}, \mathbf{y})}} + 1. \tag{1.13}$$

The necessity to separate the points from the origin stems from the fact that in the implementation we use, all separating hyperplanes are required to pass through the origin. Second, because most classification problems below are very unbalanced in the sense that the numbers of positive and negative examples are very different,

1. Available from `www.cs.columbia.edu/compbio/svm-pairwise` .
2. Available from `http://microarray.cpmc.columbia.edu/gist` .

we used a class-dependent regularization parameter. This amounts to adding to the diagonal a constant $0.02\alpha_+/0.02\alpha_-$ to all positive/negative examples, where α_+/α_- is the fraction of positive/negative examples in the training set (see Liao and Noble, 2002; Jaakkola et al., 2000, for details and justifications).

1.7.2 SVM-Pairwise Kernel

We reproduced the implementation of the best method tested in Liao and Noble (2002). For a given training set of size n, each sequence $\mathbf{x} \in \mathcal{X}$ is represented by a feature vector $U^{\text{pw}}(\mathbf{x})$ of dimension n, where each coordinate is the E-value of the SW scores $SW(\mathbf{x}, \mathbf{y}_i)$ $(1 \leq i \leq n)$ between $\mathbf{x}$ and a sequence $\mathbf{y}_i$ in the training set, as computed by the SSearch software (Lipman and Pearson, 1988). The Smith-Waterman algorithm here uses the BLOSUM 62 substitution matrix, gap opening penalty of 11, and gap extension penalty of 1. Following Liao and Noble (2002) the kernel is defined as a radial basis function kernel on the vectors $U^{pw}(\mathbf{x})$ normalized to unit length, that is:

$$k^{\text{pw}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\left(\frac{U^{pw}(\mathbf{x})}{||U^{pw}(\mathbf{x})||} - \frac{U^{pw}(\mathbf{y})}{||U^{pw}(\mathbf{y})||}\right)^2\right),$$

where the width σ is the median Euclidean distance (in the feature space) from any positive training example to the nearest negative example.

1.7.3 SVM-Fisher Kernel

The SVM-Fisher method (Jaakkola et al., 2000) represents any sequence $\mathbf{x} \in \mathcal{X}$ by a fixed-length vector $U(\mathbf{x})$, namely the gradient of the log-likelihood of $\mathbf{x}$ under a profile HMM probabilistic model, with respect to the emission parameters of the model. We followed exactly the procedure described in Jaakkola et al. (2000).

1.7.4 Mismatch Kernel

The mismatch kernel (Leslie et al., 2003) consists in representing a sequence by the set of fixed-length blocks it contains (typically three to six amino acids long), and augmenting this set by the blocks obtained by mutating a small number of amino acids (typically one or two) in the blocks observed. The mismatch kernel between two sequences is the inner products between these bag-of-blocks representations. We tested the kernel based on blocks of length 5, with up to 1 mutation, as it was reported to have the best performance in Leslie et al. (2003).

1.7.5 Local Alignment Kernels

As explained in section 1.5 we compute the LA kernels using dynamic programming. These kernels have several parameters: the gap penalty parameters e and d, the

Table 1.1 ROC, ROC_{50} and median RFP averaged over 54 families for different kernels. The LA-eig and LA-ekm kernels with $\beta = +\infty$ correspond to the SW score (modified to become positive definite on the set of proteins used to train the SVM).

Kernel	Mean ROC	Mean ROC_{50}	Mean mRFP
LA-eig ($\beta = +\infty$)	0.908	0.591	0.0654
LA-eig ($\beta = 1$)	0.912	0.612	0.0626
LA-eig ($\beta = 0.8$)	0.908	0.597	0.0679
LA-eig ($\beta = 0.5$)	**0.925**	**0.649**	0.0541
LA-eig ($\beta = 0.2$)	0.923	**0.661**	0.0637
LA-eig ($\beta = 0.1$)	0.868	0.429	0.111
LA-ekm ($\beta = +\infty$)	0.916	0.585	0.0580
LA-ekm ($\beta = 1$)	0.920	0.587	**0.0539**
LA-ekm ($\beta = 0.8$)	0.916	0.585	0.0592
LA-ekm ($\beta = 0.5$)	**0.929**	0.600	**0.0515**
LA-ekm ($\beta = 0.2$)	0.877	0.453	0.125
LA-ekm ($\beta = 0.1$)	0.596	0.052	0.500
Pairwise	0.896	0.464	0.0837
Mismatch	0.872	0.400	0.0837
Fisher	0.773	0.250	0.204

amino acid mutation matrix s, and the factor β which controls the influence of suboptimal alignments in the kernel value. A precise analysis of the effects of these parameters would be beyond the scope of this chapter so we limit ourselves to the analysis of the effect of the β parameter. In order to be consistent with the SVM-pairwise method the substitution matrix is always the BLOSUM 62 matrix and the gap parameters are always ($e = 11$,$d = 1$) below.

1.8 Results

Table 1.1 summarizes the performance of the different methods in terms of ROC, ROC_{50}, and RFP scores averaged over all 54 families tested. We tested the local alignment kernels LA-eig and LA-ekm for several values of β ranging from $+\infty$, in which case they are derived from the SW score (1.12), to $\beta = 0.1$.

These results show that both LA-eig and LA-ekm perform best for β in the range of 0.2 to 0.5, and have almost similar performances in that case. This suggests that the normalization of $\tilde{k}_{LA}^{(\beta)}$ into a positive definite kernel through the empirical kernel map of Schölkopf et al. (2002) or by subtracting the smallest negative eigenvalue from the diagonal has little influence on the final performance.

Second, the fact that the performance of the LA-eig and LA-ekm kernels is better for β in the range of 0.2 to 0.5 than for $\beta = \infty$ shows that the SW score as a

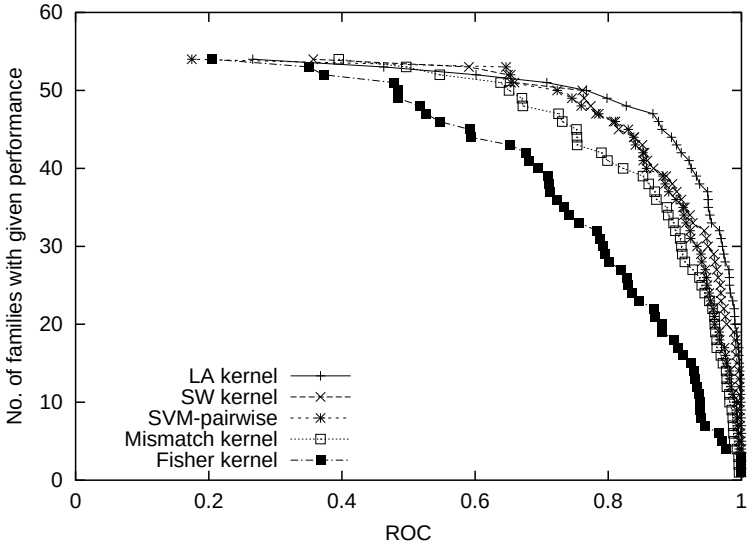

Figure 1.2 ROC score distribution for different kernels. The curve denoted LA kernel corresponds to the LA-eig kernel with $\beta = 0.5$. The curve denoted SW kernel corresponds to the LA-eig kernel with $\beta = \infty$, which is equal to the SW score up to a constant on the diagonal.

kernel is outperformed by variants which take into account suboptimal alignments to quantify the similarity between protein sequences.

The results obtained with the Fisher, pairwise, and mismatch kernels are consistent with Jaakkola et al. (2000), Liao and Noble (2002), and Leslie et al. (2003). The pairwise and mismatch kernels perform almost at the same level, with some advantage for the pairwise kernel in terms of ROC and ROC_{50} scores. They both outperform the Fisher kernel on this benchmark, but this is likely due to the facts that only a single HMM is used to build the Fisher score vector for each family, on the one hand, and that no protein outside the training data set was used to train the HMM, on the other hand. In more realistic conditions, the Fisher kernel performs roughly at the same level as the mismatch kernel (C. Leslie, private communication).

More important, most of the LA kernels tested slightly outperform all three other methods in this benchmark. As an illustration, the distribution of ROC, ROC_{50}, and median RFP scores for all three methods and the LA-eig kernel with $\beta = 0.5$ and $\beta = \infty$ are shown in figures 1.2, 1.3 and 1.4. In each case a higher curve corresponds to a more accurate remote homology detection method. The LA-eig kernel with $\beta = 0.5$ retrieves more than twice as many families as the best other method tested (the pairwise method) at a ROC_{50} score of 0.8 or higher. This remains true for a wide range of values for β, including $\beta = +\infty$. This means that the SW score as a kernel also outperforms the Fisher, pairwise, and mismatch kernels.

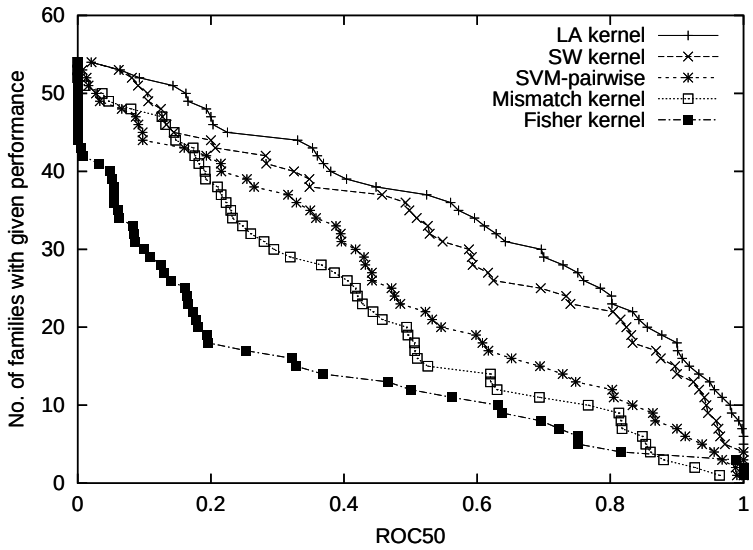

Figure 1.3 ROC_{50} score distribution for different kernels.

1.8.1 Complexity

Another important factor for practical use of these kernels is their computation cost and speed. The mismatch kernel has a linear complexity with respect to the sum of the sequence lengths, and is by far the fastest to operate. The complexity of the LA kernel is proportional to the product of the sequence lengths. Moreover, the LA kernel is faster to compute for $\beta = +\infty$ (SW score) than for other values of β, because in that case all computations can be performed with sum and max operations on integer instead of logarithms on floating point real numbers. In our experiments the computation of the SW kernel was 4 times slower than the computation of the mismatch kernel (using the SSEARCH software for the SW score, and an implementation of the mismatch kernel described in Leslie et al. (2003), likely to be optimized in the future). The computation of the LA kernel for other values of β was 2 orders of magnitude slower than the case $\beta = \infty$, using a very naive implementation of the dynamic programming algorithm.

The computation of the kernel Gram matrix on the training set for SVM-pairwise and the LA-ekm kernels requires $\mathrm{O}(n^3)$ further operations to multiply the empirical kernel map matrix by its transpose. Only $\mathrm{O}(n^2)$ are approximately required by the LA-eig kernels to compute the smallest eigenvalue using the power method (Golub and Van Loan, 1996) and subtract it from the diagonal. However, in both cases this operation is very fast compared to the time required to compute the LA kernel values or E-values.

Finally, classification of a new sequence with SVM-pairwise or the LA-ekm kernels requires computing the explicit empirical kernel map representation of the sequence, that is, computing n E-value or LA kernels. In the case of the spectrum and LA-eig

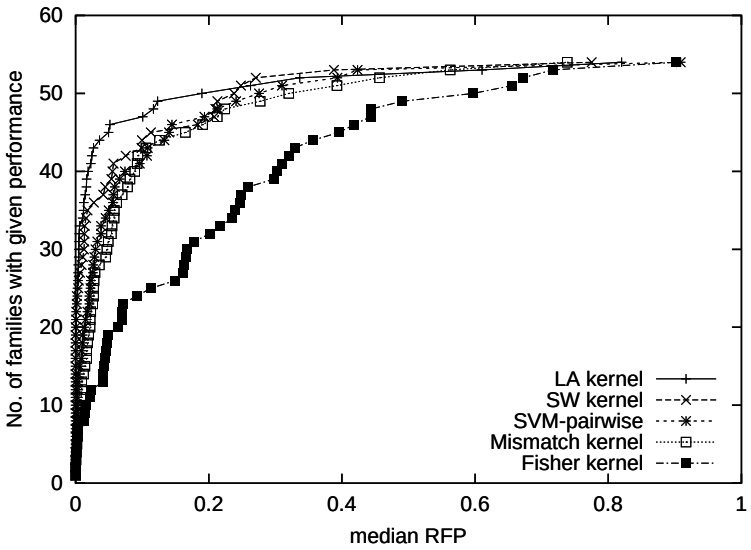

Figure 1.4 Median RFP distribution for different kernels.

kernels, the kernels are only computed between the new sequence and the support vector sequences, which usually form only a subset of the training set. Because the performances of LA-eig and LA-ekm are very similar, this suggests a preference for the former.

1.9 Discussion and Conclusion

In this chapter we introduced a family of kernels specifically adapted to protein sequences, based on the detection of high-scoring local alignments. These kernels are biologically motivated, and extend classic work on local alignment scoring to the framework of kernel functions.

The theoretically valid local alignment kernels we come up with suffer in practice from diagonal dominance. Hence we employed tricks to turn them into useful kernels, by taking a logarithm and adding a constant on the diagonal, or by using the empirical kernel map. The resulting kernels significantly outperform all other state-of-the-art methods on a benchmark experiment of SCOP superfamily recognition, which was designed to simulate the problem of remote homology detection.

The remarkable accuracy of our method certainly comes from the combination of two widely used algorithms. On the one hand, the SVM algorithm is based on a sound mathematical framework and has been shown to perform very well on many real-world applications. One of its particularities is that it can perform classification of any kind of data, such as strings in our case, as soon as a kernel function is provided. On the other hand, local alignment scores, in particular the SW score, have been developed to quantify the similarity of biological sequences.

Their parameters have been optimized over the years to provide relevant measures of similarity for homologous sequences, and they now represent core tools in computational biology.

However, direct pairwise comparisons of sequences through local alignment scores such as the SW scores are often considered naive and weak methods to detect remote homology. They are usually outperformed by methods such as PSI-BLAST which extend pairwise comparisons to pools of sequences extracted iteratively. The main contribution of this chapter is to show that pairwise sequence comparison can be extremely powerful when used as a kernel function combined with an SVM, and that the SW score itself provides a state-of-the-art method for remote homology detection when used as a kernel. An interesting conclusion of our experiments is that the SW score, however, is outperformed by local alignment scores which sum up the contributions of all possible local alignments. Summing up over local alignments has an important cost in terms of computation time due to the operations required with floating point numbers, but can be worth the cost when one is interested in precision more than speed. On the other hand the SW score itself is computed by dynamic programming and is therefore slower to compute than the mismatch kernel which it outperforms. Here again a tradeoff must be found between speed and accuracy, depending on the application. However, due to its wide use in computational biology the SW score has been precomputed and stored in databases such as KEGG's SSDB (Kanehisa et al., 2002) for virtually all known or predicted proteins of sequenced genomes, which suggests that practical applications for the Smith-Waterman kernel could be implemented in relation to such databases.

Suboptimal alignments

The only parameter whose influence was tested is the parameter β of the LA kernel, which controls the importance of the contribution of nonoptimal local alignments in the final score. It should be pointed out here that the optimal values for β we observed (in the range of 0.2 to 0.5) are only optimal for an average performance on the 54 families tested, and that the optimal value for each family might fluctuate. Moreover, a number of other parameters could be modified, in particular the gap penalty parameters and the similarity matrix between amino acids, and the optimal values for β might also depend on these parameters. Further theoretical and practical studies, which are beyond the scope of this chapter, should be performed to evaluate the influence of these parameters. In particular, it would be interesting to know for which values of these parameters the SW score itself is a valid kernel, and which parameter tuning results in the most accurate remote protein homology detection.

Parameter setting

An important open problem with the LA kernels as well as with most other string kernels is the following: how to make the kernel independent of the lengths of the sequences compared. Indeed long sequences typically result in small kernel values when the kernel is normalized with (1.13). While much work has been done to estimate the significance of alignment scores for varying sequence length, these approaches remain difficult to adapt to the kernel framework. The importance of this issue might be underestimated in the benchmark experiment presented in this chapter, because protein sequences in a SCOP family tend to have similar

Length normalization

sequence lengths. However applying kernel-based homology detection in a more realistic setting might reveal important effects on this issue.

While we focused in this paper on the application of the new kernels to the problem of remote homology detection, it should finally be pointed out that possible use of these kernels, and more generally of all other string kernels, go beyond this single goal. In combination with SVM, string kernels can be applied to various classification and regression tasks such as gene function, localization, or structure prediction. Moreover recent studies suggest that kernels provide a useful framework for integrating and performing inference from heterogeneous data (Vert and Kanehisa, 2003; Yamanishi et al., 2003), which we plan to investigate in the future.

1.10 Acknowledgments

We thank Li Liao and William Stafford Noble for making their data sets and software available, Mark Diekhans for providing information about the Fisher kernel, and Christina Leslie, Eleazar Eskin, and Jason Weston for information and software concerning the mismatch kernel. The computational resource was provided by the Bioinformatics Center, Institute for Chemical Research, Kyoto University and the Supercomputer Laboratory, Kyoto University. Part of this work was supported by a Grant-in-Aid for Scientific Research on Priority Areas (C) "Genome Information Science" from MEXT of Japan.

Appendix A: Proof of Theorem 1.7

In this proof we assume the similarity matrix S and gap penalty function g are fixed, and remove them as subscripts in $s_{S,g}$ and $SW_{S,g}$ for convenience. For any two sequences $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$, let

$$\Pi_n^0(\mathbf{x}, \mathbf{y}) = \{\pi \in \Pi_n(\mathbf{x}, \mathbf{y}) : \pi_1(n) = |\mathbf{x}|, \pi_2(n) = |\mathbf{y}|\}.$$

We can prove a first useful result:

Lemma 1.9 *With the notations of theorem 1.7, the following holds for any two sequences* $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ *and* $n \geq 1$*:*

$$\sum_{\pi \in \Pi_n^0(\mathbf{x}, \mathbf{y})} \exp\left(\beta s(\mathbf{x}, \mathbf{y}, \pi)\right) = k_0 \star k_a \star (k_g \star k_a)^{n-1}(\mathbf{x}, \mathbf{y}), \tag{1.14}$$

and for $n = 0$*:*

$$\sum_{\pi \in \Pi_0^0(\mathbf{x}, \mathbf{y})} \exp\left(\beta s(\mathbf{x}, \mathbf{y}, \pi)\right) = k_0(\mathbf{x}, \mathbf{y}) \tag{1.15}$$

Proof Let us proceed by induction on n. For $n = 0$, $\Pi_n^0(\mathbf{x}, \mathbf{y})$ is reduced to the singleton $\{\emptyset\}$ (the alignment where no position is aligned), which has a null score. As a result:

$$\sum_{\pi \in \Pi_0^0(\mathbf{x},\mathbf{y})} \exp\left(\beta s(\mathbf{x}, \mathbf{y}, \pi)\right) = \exp\left(\beta s(\mathbf{x}, \mathbf{y}, \emptyset)\right) = 1 = k_0(\mathbf{x}, \mathbf{y}),$$

which proves (1.15) for any $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$.

For $n = 1$, $\Pi_n^0(\mathbf{x}, \mathbf{y})$ is reduced to the singleton $\{(|\mathbf{x}|, |\mathbf{y}|)\}$ (only the last letters are aligned). The score of this alignment is $S(x_{|\mathbf{x}|}, y_{|\mathbf{y}|})$. On the other hand, by (1.4), (1.5), and (1.6), the following holds:

$$k_0 \star k_a(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x}_1\mathbf{x}_2=\mathbf{x}, \mathbf{y}_1\mathbf{y}_2=\mathbf{y}} k_0(\mathbf{x}_1, \mathbf{y}_1) k_a(\mathbf{x}_2, \mathbf{y}_2) = \exp\left[\beta S(x_{|\mathbf{x}|}, y_{|\mathbf{y}|})\right].$$

This proves (1.14) in the case $n = 1$, for any $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$.

Let us now suppose that (1.14) is true at the level $n - 1$, and prove it is then true at the level n. Let $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ be two sequences. Then any alignment $\pi \in \Pi_n^0(\mathbf{x}, \mathbf{y})$ induces a decomposition $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$, with $\mathbf{x}_1 = x_1 \cdots x_{\pi_1(n-1)}$, $\mathbf{x}_2 = x_{\pi_1(n-1)+1} \cdots x_{|\mathbf{x}|-1}$, and $\mathbf{x}_3 = x_{|\mathbf{x}|}$ (and similarly for $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\mathbf{y}_3$). By construction, the alignment $f(\pi)$ obtained from π by removing the last aligned positions satisfies $f(\pi) \in \Pi_n^0(\mathbf{x}_1, \mathbf{y}_1)$. Conversely, it is easy to see that for any decompositions $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ and $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\mathbf{y}_3$ with $|\mathbf{x}_3| = |\mathbf{y}_3| = 1$, and for any $\pi \in \Pi_{n-1}^0(\mathbf{x}_1, \mathbf{y}_1)$, we can find a unique $\pi' \in \Pi_n^0(\mathbf{x}, \mathbf{y})$ such that $f(\pi') = \pi$, namely the alignment obtained by adding to the alignment π' the pair $(|\mathbf{x}|, |\mathbf{y}|)$. This bijection enables us to write

$$\sum_{\pi \in \Pi_n^0(\mathbf{x},\mathbf{y})} \exp\left(\beta s(\mathbf{x}, \mathbf{y}, \pi)\right) = \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2\mathbf{y}_3} \sum_{\pi' \in \Pi_{n-1}^0(\mathbf{x}_1,\mathbf{y}_1)} \exp\left(\beta s(\mathbf{x}_1, \mathbf{y}_1, f^{-1}(\pi'))\right), \quad (1.16)$$

where the first sum in the right-hand side is over all decompositions of $\mathbf{x}$ and $\mathbf{y}$ with $|\mathbf{x}_3| = |\mathbf{y}_3| = 1$.

By definition of the local alignment score (1.1), we have for any $\pi \in \Pi_n^0(\mathbf{x}, \mathbf{y})$:

$$s(\mathbf{x}, \mathbf{y}, \pi) = s(\mathbf{x}_1, \mathbf{y}_1, f(\pi)) + g(|\mathbf{x}_2|) + g(|\mathbf{y}_2|) + S(x_{|\mathbf{x}|}, y_{|\mathbf{y}|}),$$

and therefore, by (1.7) and (1.6):

$$\exp\left(\beta s(\mathbf{x}, \mathbf{y}, \pi)\right) = \exp\left(\beta s(\mathbf{x}_1, \mathbf{y}_1, f(\pi))\right) . k_g(\mathbf{x}_2, \mathbf{y}_2) . k_a(\mathbf{x}_3, \mathbf{y}_3). \quad (1.17)$$

Observing that $k_a(\mathbf{x}_3, \mathbf{y}_3)$ is null when $\mathbf{x}_3$ or $\mathbf{y}_3$ is not reduced to a single letter, we can plug (1.17) into (1.16) to get

$$\begin{aligned}&\sum_{\pi \in \Pi_n^0(\mathbf{x},\mathbf{y})} \exp\left(\beta s(\mathbf{x},\mathbf{y},\pi)\right) \\ &= \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2\mathbf{y}_3} \sum_{\pi \in \Pi_{n-1}^0(\mathbf{x}_1,\mathbf{y}_1)} \exp\left(\beta s(\mathbf{x}_1,\mathbf{y}_1,f(\pi))\right) . k_g(\mathbf{x}_2,\mathbf{y}_2).k_a(\mathbf{x}_3,\mathbf{y}_3),\end{aligned} \tag{1.18}$$

where the first sum in the right-hand side is now over all possible decompositions of $\mathbf{x}$ and $\mathbf{y}$ with $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \in \mathcal{X}^6$.

Using the induction hypothesis and the definition of convolution we can now conclude as follows:

$$\begin{aligned}&\sum_{\pi \in \Pi_n^0(\mathbf{x},\mathbf{y})} \exp\left(\beta s(\mathbf{x},\mathbf{y},\pi)\right) \\ &\quad= \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2\mathbf{y}_3} \sum_{\pi \in \Pi_{n-1}^0(\mathbf{x}_1,\mathbf{y}_1)} \exp\left(\beta s(\mathbf{x}_1,\mathbf{y}_1,\pi)\right) . k_g(\mathbf{x}_2,\mathbf{y}_2).k_a(\mathbf{x}_3,\mathbf{y}_3) \\ &\quad= \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2\mathbf{y}_3} k_0 \star k_a \star (k_g \star k_a)^{n-2} (\mathbf{x}_1,\mathbf{y}_1).k_g(\mathbf{x}_2,\mathbf{y}_2).k_a(\mathbf{x}_3,\mathbf{y}_3) \\ &\quad= k_0 \star k_a \star (k_g \star k_a)^{n-1} (\mathbf{x},\mathbf{y}),\end{aligned} \tag{1.19}$$

■

Going back to the proof of theorem 1.7, let $n > 0$ and $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$. Observing that any alignment $\pi \in \Pi_n(\mathbf{x}, \mathbf{y})$ is in fact an element of $\Pi_n^0(\mathbf{x}_1, \mathbf{y}_1)$ where $\mathbf{x}_1 = x_1 \cdots x_{\pi_1(n)}$ and $\mathbf{y}_1 = y_1 \cdots y_{\pi_2(n)}$, we can write

$$\begin{aligned}\sum_{\pi \in \Pi_n(\mathbf{x},\mathbf{y})} \exp\left(\beta s(\mathbf{x},\mathbf{y},\pi)\right) &= \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2} \sum_{\pi \in \Pi_n^0(\mathbf{x}_1,\mathbf{y}_1)} \exp\left(\beta s(\mathbf{x}_1,\mathbf{y}_1,\pi)\right) \\ &= \sum_{\mathbf{x}=\mathbf{x}_1\mathbf{x}_2, \mathbf{y}=\mathbf{y}_1\mathbf{y}_2} k_0 \star k_a \star (k_g \star k_a)^{n-1} (\mathbf{x}_1,\mathbf{y}_1) \\ &= k_0 \star k_a \star (k_g \star k_a)^{n-1} \star k_0(\mathbf{x},\mathbf{y}),\end{aligned} \tag{1.20}$$

where the first equality is obtained by organizing the alignments in terms of $\pi_1(n)$ and $\pi_2(n)$, the second is the consequence of lemma 1.9, and the last is the definition of convolution. (1.9) now follows by summing this equality over n.

In order to prove (1.10) we derive from (1.9) the following:

$$\begin{aligned}\frac{1}{\beta}\log k_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) &= \frac{1}{\beta}\log\left(\sum_{\pi\in\Pi(\mathbf{x},\mathbf{y})}\exp\left(\beta s(\mathbf{x},\mathbf{y},\pi)\right)\right)\\ &= SW(\mathbf{x},\mathbf{y}) + \frac{1}{\beta}\log\left(\sum_{\pi\in\Pi(\mathbf{x},\mathbf{y})}\exp\left[\beta\left(s(\mathbf{x},\mathbf{y},\pi) - SW(\mathbf{x},\mathbf{y})\right)\right]\right).\end{aligned} \tag{1.21}$$

By definition of the SW score (1.2) it follows that:

$$\lim_{\beta\to+\infty}\log\left(\sum_{\pi\in\Pi(\mathbf{x},\mathbf{y})}\exp\left[\beta\left(s(\mathbf{x},\mathbf{y},\pi) - SW(\mathbf{x},\mathbf{y})\right)\right]\right) = \log l, \tag{1.22}$$

where l is the number of alignments in $\Pi(\mathbf{x},\mathbf{y})$ with maximal alignment. (1.10) is now a consequence of (1.21) and (1.22), which concludes the proof of theorem 1.7. ■

Appendix B: Proof of Theorem 1.8

For any sequences $(\mathbf{x},\mathbf{y}) \in \mathcal{X}^2$ let us first introduce some notations. For $0 \le i \le |\mathbf{x}|$ and $0 \le j \le |\mathbf{y}|$, we define the following sets of alignments:

$$\begin{aligned}\Pi_{i,j} &:= \{\pi \in \Pi(\mathbf{x},\mathbf{y})\backslash\{\emptyset\} : \pi_1(|\pi|) \le i, \pi_2(|\pi|) \le j\},\\ \Pi^0_{i,j} &:= \{\pi \in \Pi_{i,j} : \pi_1(|\pi|) = i, \pi_2(|\pi|) = j\},\\ \Pi^1_{i,j} &:= \{\pi \in \Pi_{i,j} : \pi_1(|\pi|) < i, \pi_2(|\pi|) = j\},\\ \Pi^2_{i,j} &:= \{\pi \in \Pi_{i,j} : \pi_1(|\pi|) < i, \pi_2(|\pi|) < j\}.\end{aligned} \tag{1.23}$$

Hence $\Pi_{i,j}$ is the set of alignments that only involve up to the first i and j letters of $\mathbf{x}$ and $\mathbf{y}$, and the three subsets $\Pi^0_{i,j}$, $\Pi^1_{i,j}$, and $\Pi^2_{i,j}$ form a partition of $\Pi_{i,j}$. Let us also introduce a score $s_{i,j}$ for the alignments of $\Pi_{i,j}$ derived from the local alignment score (1.1) by

$$\forall \pi \in \Pi_{i,j}, \quad s_{i,j}(\pi) := s(\pi) - g\left(i - \pi_1(|\pi|)\right) - g\left(j - \pi_2(|\pi|)\right). \tag{1.24}$$

In other words, $s_{i,j}$ is an alignment score that penalizes the unaligned part after the last aligned letters, contrary to s. With these definitions we can state a key lemma that explains the meaning of the functions computed in theorem 1.8

Lemma 1.10 *With the notations of theorem 1.8, the following equalities hold for* $0 \leq i \leq |\mathbf{x}|$ *and* $0 \leq j \leq |\mathbf{y}|$*:*

$$
\begin{aligned}
M(i,j) &= \sum_{\pi \in \Pi^0_{i,j}} \exp\left(\beta s_{i,j}(\pi)\right) = \sum_{\pi \in \Pi^0_{i,j}} \exp\left(\beta s(\pi)\right), \\
X(i,j) &= \sum_{\pi \in \Pi^1_{i,j}} \exp\left(\beta s_{i,j}(\pi)\right), \\
Y(i,j) &= \sum_{\pi \in \Pi^2_{i,j}} \exp\left(\beta s_{i,j}(\pi)\right), \\
X_2(i,j) &= \sum_{\pi \in \Pi^1_{i,j}} \exp\left(\beta s(\pi)\right), \\
Y_2(i,j) &= \sum_{\pi \in \Pi^2_{i,j}} \exp\left(\beta s(\pi)\right).
\end{aligned}
$$

Proof This lemma is proved by induction in (i,j). If $i = 0$ or $j = 0$, then $\Pi_{i,j} = \emptyset$ and the statements are true. To prove the statements for $i > 0$ and $j > 0$, consider first the statement about $M(i,j)$. Let $f : \Pi(\mathbf{x},\mathbf{y}) \to \Pi(\mathbf{x},\mathbf{y})$ be the application that removes the last pair of aligned position in an alignment. Then f is clearly a bijection between $\Pi^{(0)}_{i,j}$ and $\Pi_{i-1,j-1} \cup \{\emptyset\}$, and $s(\pi) = s_{i,j}(\pi) = s_{i-1,j-1}(f(\pi)) + S(x_i, y_j)$ for any $\pi \in \Pi^0_{i,j}$. We therefore can write

$$
\begin{aligned}
&\sum_{\pi \in \Pi^0_{i,j}} \exp\left(\beta s(\pi)\right) \\
&\quad = \sum_{\pi \in \Pi^0_{i,j}} \exp\left(\beta s_{i,j}(\pi)\right) \\
&\quad = \sum_{\pi \in \Pi_{i-1,j-1} \cup \{\emptyset\}} \exp\left(\beta s_{i-1,j-1}(\pi) + \beta S(x_i, y_j)\right) \\
&\quad = e^{\beta S(x_i,y_j)} \left(M(i-1,j-1) + X(i-1,j-1) + Y(i-1,j-1) + 1\right) \\
&\quad = M(i,j),
\end{aligned}
$$

where the third equality uses the induction hypothesis and the fourth one the definition of $M(i,j)$. The same approach can be followed to prove the other statements of lemma 1.10, so we don't explicitly write them down for lack of space. ■

Let $m = |\mathbf{x}|$ and $n = |\mathbf{y}|$. theorem 1.8 is now a direct consequence of lemma 1.10 by using (1.9) and observing that $\Pi(\mathbf{x},\mathbf{y}) = \Pi_{m,n}$ is the disjoint union of $\{\emptyset\}$, $\Pi^0_{m,n}$, $\Pi^1_{m,n}$ and $\Pi^2_{m,n}$. ■

References

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

C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. New York, Springer Verlag, 1984.

M. Collins and N. Duffy. Convolution kernels for natural language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 625–632, Cambridge, MA, MIT Press, 2002.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 601–608. Cambridge, MA, MIT Press, 2003.

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

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore, Johns Hopkins University Press, 1996.

M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, 1999.

T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2): 95–114, 2000.

M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

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. Lauerdale, and T.E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575. River Edge, NJ, World Scientific, 2002.

C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1417–1424. Cambridge, MA, MIT Press, 2003.

L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In G. Myers, S. Hannenhalli, D. Sankoff, S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 225–232, New York, ACM Press, 2002.

D. Lipman and W. Pearson. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

W. R. Pearson. Rapid and sensitive sequence comparisions with FASTP and FASTA. *Methods in Enzymology*, 183:63–98, 1990.

B. Schölkopf, J. Weston, E. Eskin, C. Leslie, and W. S. Noble. A kernel approach for learning from almost orthogonal patterns. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of ECML 2002, 13th European Conference on Machine Learning, Helsinki, Finland, August 19-23, 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 511–528. Heidelberg, Springer Verlag, 2002.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1425–1432. Cambridge, MA, MIT Press, 2003.

C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

Y. Yamanishi, J.-P. Vert, A. Nakaya, and M. Kanehisa. Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. *Bioinformatics*, 19:i323–i330, 2003.