The Context-Tree Kernel for Strings

Marco Cuturi [a,b,*] Jean-Philippe Vert [a]

[a] *Computational Biology Group, Ecole des Mines de Paris, 35 rue Saint Honoré, 77300 Fontainebleau, France*

[b] *The Institute of Statistical Mathematics 4-6-7 Minami-azabu, Minato-ku, Tokyo 106-8569 Japan*

Abstract

We propose a new kernel for strings which borrows ideas and techniques from information theory and data compression. This kernel can be used in combination with any kernel method, in particular Support Vector Machines for string classification, with notable applications in proteomics. By using a Bayesian averaging framework with conjugate priors on a class of Markovian models known as probabilistic suffix trees or context-trees, we compute the value of this kernel in linear time and space while only using the information contained in the spectrum of the considered strings. This is ensured through an adaptation of a compression method known as the context-tree weighting algorithm. Encouraging classification results are reported on a standard protein homology detection experiment, showing that the context-tree kernel performs well with respect to other state-of-the-art methods while using no biological prior knowledge.

Key words: string kernel, mutual information kernel, universal coding, protein homology detection

1 Introduction

The need for efficient analysis and classification tools for strings remains a key issue in machine learning. This is notably the case in computational biology where the availability of an ever-increasing quantity of biological sequences calls for efficient and computationally feasible algorithms to detect, cluster, and annotate functional similarities between DNA or amino-acid sequences.

* Corresponding author.
Email addresses: `marco.cuturi@ensmp.fr` (Marco Cuturi), `jean-philippe.vert@ensmp.fr` (Jean-Philippe Vert).

Recent years have witnessed the rapid development of a class of algorithms called *kernel methods* [25] that may offer useful tools for these tasks. In particular, the Support Vector Machine (SVM) algorithms [4,29] provide state-of-the-art performance in many real-world problems of classifying objects into predefined classes. SVMs have already been applied with success to a number of issues in computational biology, including but not limited to protein homology detection [16,19,20,24,3,31], functional classification of genes [22,30], or prediction of gene localization [14]. A more complete survey of the application of kernel methods in computational biology is presented in [26].

The basic ingredient shared by all kernel methods is the kernel function, that measures similarities between pairs of objects to be analyzed or classified. To use kernel methods in the field of string classification requires a prior design of an efficient kernel function on strings. Indeed, while early-days SVM focused on the classification of vector-valued objects, for which kernels are well understood and easily represented, recent attempts to use SVM for the classification of more general objects have resulted in the development of several kernels for structured objects such as strings [32,13,16,19,20,24,3,31], graphs [17], or even phylogenetic profiles [30].

A useful kernel for sequences, as the one we wish to propose in this work, should have several properties. It should represent a *meaningful* measure of *similarity* between two sequences and be general enough to be efficient on different datasets without excessive tuning. This similarity measure needs to be further *positive definite* to be applied in the general framework of kernel methods and *rapid to compute* to sustain large-scale implementations (typically, have a linear complexity with respect to the lengths of the compared sequences). Such an ideal kernel probably does not exist, and different kernels might be useful in different situations. For large-scale studies which might involve comparing thousands of sequences, yielding to millions of kernel evaluations, or to answer simple queries which could be found in on-line applications, the computation cost becomes critical and only fast kernels, such as the spectrum [19] and mismatch [20] kernels can be accepted. In applications where accuracy is more important than speed, slower kernels that include more biological knowledge such as the Fisher [16], pairwise [22] or local alignment [31] kernels might be accepted if they improve the performance of a classifier.

Our contribution in this paper is to introduce a new class of string kernels which are both fast to compute and based on the spectrum of the considered strings. The spectrum of a string as defined by [19] is the weighted list of k-mers (or k-grams, that is a substring of k letters) contained in the string, where the weights stand for the occurrence (or relative frequency with respect to the string's length) of the considered k-mer in the string. While the work of [19] uses a linear dot-product on that representation, we propose in this work an alternative class of kernels on those counters.

The motivation behind these kernels is grounded on information theory, in a similar way to the work proposed recently in [21]. By applying an information theoretic viewpoint on the information carried out by strings, we present a way to compare strings through kernel methods using little prior knowledge on the structure of the alphabet, just as universal coding [7] aims at giving a sound compression of sequences with no prior assumptions on the nature of those sequences. This information theoretic viewpoint takes the form of a string compression algorithm, which is first applied on two strings X and Y to be compared taken separately, and then on their concatenation XY. Intuitively if the compression behaves in a similar way (in terms of gain for instance) for X, Y and XY, one can expect the strings to share similar properties. On the opposite, one might conclude that the strings are dissimilar if their concatenation cannot be efficiently compressed. This intuition can be translated mathematically in terms of differences in coding redundancy between X and Y with respect to XY, in the light of noiseless coding theory for instance [7].

The compression method we choose in this work is the popular context-tree weighting (CTW) algorithm [33], and we show how to derive a kernel out of it. The compression performed by the CTW algorithm involves a Bayesian averaging of the probability of a string under a large collection of weighted source distributions. These source distributions are chosen among variable-length Markov chains, which are also known as context-tree (CT) models. Using the CTW algorithm to derive a kernel brings a sound answer to the criterions expressed previously, since it guarantees positive definiteness, computational speed, and an additional interpretation (other than the one considered by compression) to our kernel.

Indeed, the integral representation of the CTW compression, not shared with ad-hoc heuristics such as the Lempel-Ziv algorithm, first enables us to cast easily the proposed kernels in the framework of *mutual information kernels* [28], which ensures their positive definiteness. Second, the Bayesian integration over Markovian (and hence exponential) models performed by such kernels provides us with an alternative probabilistic interpretation of their computation. Following that alternative perspective, the kernels project each sequence to be compared to the set of their probabilities under all distributions contained in the class of CT models, and compare different sequences in the light of their respective projections. These projections can be intuitively considered as feature extractions, where each considered context-tree distribution acts as a feature extractor, providing a feature which is the likelihood of the distribution for the considered sequence. Because we find that perspective to be clearer, we will favor this interpretation and present the family of context-tree kernels in a constructive manner and as a special case of mutual information kernels. However the reader should keep in mind that most choices in models and priors taken to devise such kernels are chosen to match the CTW algorithm's ones, so as to benefit from its properties including notably computational tricks presented by the authors of [33] to ensure linear (in time and space) computational costs.

The paper is organized as follows. In Section 2 we present the general strategy of devising mutual information kernels from families of probabilistic models. In Section 3 we define a kernel for sequences based on context-tree models. Its efficient implementation, derived from the CTW algorithm, is presented in Section 4. We present further interpretations of the context-tree kernel's computation as well as links with universal coding in Section 5. Experimental results on a benchmark problem of remote protein homology detection are then presented in Section 6.

2 Probabilistic Models and Mutual Information Kernels

A parametric probabilistic model on a measurable space $\mathcal{X}$ is a family of distributions $\{P_\theta, \theta \in \Theta\}$ on $\mathcal{X}$, where θ is the parameter of the distribution P_θ. Typically, the set of parameters Θ is a subset of $\mathbb{R}^n$, in which case n is called the dimension of the model. As an example, a hidden Markov model (HMM) for sequences is a parametric model, the parameters being the transition and emission probabilities [10]. A family of probabilistic models is a family $\{P_{f,\theta_f}, f \in \mathcal{F}, \theta_f \in \Theta_f\}$, where $\mathcal{F}$ is a finite or countable set, and $\Theta_f \subset \mathbb{R}^{\dim(f)}$ for each $f \in \mathcal{F}$, where $\dim(f)$ denotes the dimension of f. An example of such a family would be a set of HMMs with different architectures and numbers of states. Probabilistic models are typically used to model sets of elements $X_1, \ldots, X_n \in \mathcal{X}$, by selecting a model $\hat{f}$ and a choosing a parameter $\hat{\theta}_{\hat{f}}$ that best "fits" the dataset, using criteria such as penalized maximum likelihood or maximum a posteriori probability [10].

Alternatively, probabilistic models can also be used to characterize each single element $X \in \mathcal{X}$ by the feature representation

$$\phi(X) = \Big(P_{f,\theta_f}(X)\Big)_{f\in\mathcal{F},\theta_f\in\Theta_f}, \tag{1}$$

spanning all possible probabilities of X within the considered families. If the probabilistic models are designed in such a way that each distribution is roughly characteristic of a class of objects of interest, then the representation $\phi(X)$ quantifies how X fits each class. In this representation, each distribution can be seen as a filter that extracts from X an information, namely the probability of X under this distribution, or equivalently how much X fits the class modelled by this distribution.

Kernels are real-valued function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that can be represented in the form of a dot product $\kappa(X,Y) = \langle \psi(X), \psi(Y) \rangle_F$ for some mapping ψ from $\mathcal{X}$ to a Hilbert space F[25]. Given the preceding mapping ϕ of Equation (1), a natural way to derive a kernel from a family of probabilistic models is to endow the set of representations $\phi(X)$ with a dot product, and set $\kappa(X,Y) = \langle \phi(X), \phi(Y) \rangle$. This can be done for example if a prior probability $\pi(f, d\theta_f)$ can be defined on the set of distributions in the models, by considering the following dot product:

$$\kappa(X,Y) = \langle \phi(X), \phi(Y) \rangle \overset{\text{def}}{=} \sum_{f\in\mathcal{F}} \pi(f) \int_{\Theta_f} P_{f,\theta_f}(X)\ P_{f,\theta_f}(Y)\ \pi(d\theta_f). \tag{2}$$

By construction, the kernel in Equation (2) is a valid kernel, that belongs to the class of mutual information kernels [28]. Observe that contrary to the Fisher kernel that also uses probabilistic models, no model or parameter estimation is required in Equation (2). Intuitively, for any two elements X and Y the kernel of Equation (2) automatically detects the models and parameters that explain both X and Y, with consequent weights if the models and parameters are likely to appear under the prior π. On the other hand, models and parameters for which X and Y present no simultaneous fit bring a marginal contribution to the value of the kernel and are thus ignored.

There is of course some arbitrariness in the previous definition, both in the definition of the models and in the choice of the prior distribution π. This arbitrary can be used to include prior knowledge in the kernel. For example, if one wants to detect similarity with respect to families of sequences known to be adequately modelled by HMMs, then using HMM models constrains the kernel to detect such similarities. However, these choices need to be decided having computational limitations in mind. The calculations involved in Equation (2), namely the computation of the likelihood of a distribution for two given sequences and the integration of those likelihoods over a set of parameters, should not only be tractable under a closed form but also fast to compute. This is not likely to be the case for most families of models and most choices of priors. We consider those limitations under the light of the solution proposed by the CTW algorithm in the framework of universal coding, to define below a suitable set of models and prior distributions.

Prior to this definition, we note that some biases might appear when attempting to compare sequences of different lengths, which is likely to be the case for most applications. Indeed, as the probability of a sequence under most models defined on strings (including Markovian models) decreases roughly exponentially with its length, the value of the kernel (2) can not only be strongly biased if we directly consider the probabilities of two strings of very different lengths, but will also quickly tend to negligible values when comparing long strings. This is a classical issue with many string kernels that leads to bad performance in classification with SVM [27,31]. This undesirable effect can easily be controlled in our case by normalizing the likelihoods as follows:

$$\kappa_\sigma(X,Y) = \sum_{f\in\mathcal{F}} \pi(f) \int_{\Theta_f} P_{f,\theta_f}(X)^{\frac{\sigma}{N_X}} \; P_{f,\theta_f}(Y)^{\frac{\sigma}{N_Y}} \; \pi(d\theta_f). \tag{3}$$

where σ is a width parameter and N_X and N_Y stand for the lengths of both sequences. Equation (3) is clearly a valid kernel (only the feature extractor ϕ is modified), and the parameter σ controls the range of values it takes independently of the lengths of the sequences used.

3 A Mutual Information Kernel Based on Context-Tree Models

In this section we derive explicitly a mutual information kernel for strings based on context-tree models with mixtures of Dirichlet priors. Context-tree models, also known as probabilistic suffix trees, are Markovian models which are actually equivalent to Markov chains up to a different parametrization as we will see below. They have been shown useful to model several families of sequences, including biological ones as illustrated by their use in [2,11] where different techniques to estimate such models on protein sequences where proposed. Note however that the use of context-trees in the present work should not be related excessively to their previous success in representing sequences, notably protein families. Arguably, we both believe and observe in our experiments that the overall performance of the kernels proposed in this paper does not rely so much on the individual ability of such distributions to model specific families of sequences, but rather on their overall efficiency to extract features out of strings.

3.1 Framework and notations

Starting with basic notations and definitions, let E be a finite set of size d called the alphabet. In our experiments E will be the 20 letters alphabet of amino-acids. For a given depth $D \in \mathbb{N}$ corresponding to the maximal memory of our Markovian models, we write E_D^* for the set of strings of E of length smaller or equal to D, i.e., $E_D^* = \cup_{i=0}^{D} E^i$, which includes $\varnothing$, the empty word. We introduce $\mathcal{X} = \cup_{n=0}^{\infty}(E^D \times E)^n$, the set on which we choose to define our kernel. Observe that we do not define directly the kernel on the set of finite-length sequences, but rather in a slightly different framework which stresses the fact that we are chiefly interested in the local behaviour of the sequence. Indeed, we see sequences as finite sets of (*context,letter*) couples, where the *context* is a D-letters long subsequence of the initial sequence and the *letter* is the element next to it. This transformation is justified by the fact that we consider Markovian models with a memory limited to D letters, and is equivalent to the information contained by the spectrum of order $D+1$ of a string. An element $X \in \mathcal{X}$ can therefore be written as $X = \{(x_c^i, x_l^i)\}_{i=1..N_X}$ where N_X is the cardinality of X, $x_c^i \in E^D$ and $x_l^i \in E$ for all $1 \le i \le N_X$. By considering strings as collections of transitions (or equivalently substrings of length $D+1$) we do not only follow previous approaches such as [20,19,3] but also refer to a recent framework in kernel design [18,9,8] which aims at computing kernels on compound objects (such as long strings) as kernels for collections of smaller components ($D+1$-mers in this case).

3.2 Context-Tree Models

Context-tree distributions require the definition of a complete suffix dictionary (c.s.d) $\mathcal{D}$. A c.s.d is a finite set of words of E_D^* such that any left-infinite sequence has a unique suffix in $\mathcal{D}$, but no word in $\mathcal{D}$ has a suffix in $\mathcal{D}$. We write $L(\mathcal{D})$ for the length of the longest word contained in $\mathcal{D}$ and $\mathcal{F}_D$ for the set of c.s.d $\mathcal{D}$ that satisfy $L(\mathcal{D}) \le D$. We note that c.s.d are in correspondance with suffix trees based on E as illustrated in Figure 1. Once this dictionary $\mathcal{D}$ or the equivalent suffix tree structure is set, a distribution on $\mathcal{X}$ can be defined by attaching a multinomial distribution[1] $\theta_s \in \Sigma_d$ to *each* word s of $\mathcal{D}$. Indeed, through the family of parameters $\theta = (\theta_s)_{s \in \mathcal{D}}$ we define a conditional distribution on $\mathcal{X}$ by the following equation:

$$P_{\mathcal{D},\theta}(X) = \prod_{i=1}^{N_X} \theta_{\mathcal{D}(x_c^i)}(x_l^i), \tag{4}$$

[1] writing Σ_d for the canonical simplex of dimension d, i.e., $\Sigma_d = \{\xi = (\xi_i)_{1 \le i \le d} : \xi_i \ge 0, \sum \xi_i = 1\}$.

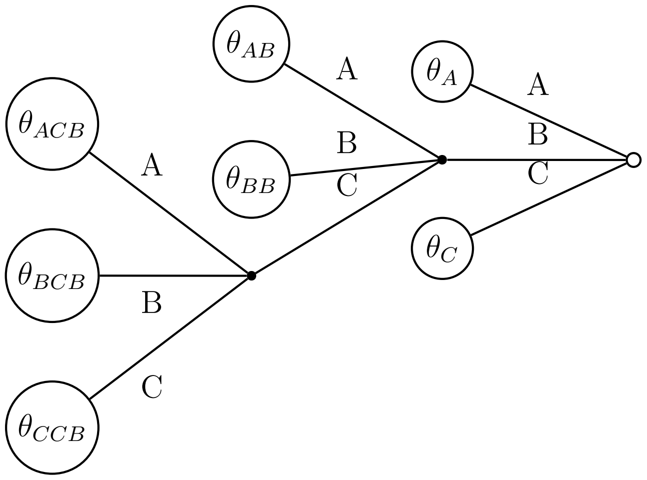

Fig. 1. Tree representation of a context-tree distribution. The alphabet E is set to $\{A, B, C\}$, the maximal depth D to 3 and the complete suffix dictionary $\mathcal{D}$ is the set of strings $\{A, AB, BB, ACB, BCB, CCB, C\}$. Each parameter θ_s for $s \in \mathcal{D}$ is in that case a vector of the 3-dimensional simplex Σ_3.

where for any word m in E^D, $\mathcal{D}(m)$ is the unique suffix of m in $\mathcal{D}$. Note that Markov chains are a simple case of context-tree distributions when the c.s.d. is set to E^D. Conversely a context-tree distribution $\mathcal{D}$ can be easily expressed as a Markov chain by assigning the transition parameter θ_s to all the contexts in E^D which admit s as their unique suffix in $\mathcal{D}$. Context-trees can thus be seen as an alternative parametrization and a handier representation of Markov chains, where the importance of some suffixes is highlighted by developing further or stopping the tree expansion in branches which have more or less significance in the generation of our string. We present in Figure 1 an example where the alphabet has been set to $E = \{A, B, C\}$ and the maximal depth D to 3. We write $\mathcal{P}_D$ for $\{P_{\mathcal{D},\theta} : \mathcal{D} \in \mathcal{F}_D, \theta \in \Theta_{\mathcal{D}}\}$, the set of context-tree distributions of depth D.

3.3 Prior Distributions on Context-Tree Models

We define in this section priors on the family of distributions $\mathcal{P}_D$ introduced in the previous section, following the framework set in Equation (3). Namely, we propose a prior probability $\pi(\mathcal{D}, d\theta)$ on $\mathcal{P}_D$ to finalize the definition of the family of kernels presented in this paper, which we name *context-tree kernels*. Note that we use and adapt the priors proposed by [33] to our computation to ensure the computation feasibility of the proposed kernels. The prior probability $\pi(\mathcal{D}, d\theta)$ on $\mathcal{P}_D$ factorizes as $\pi(\mathcal{D}, d\theta) = \pi(\mathcal{D})\,\pi(d\theta|\,\mathcal{D})$, two terms which are defined as follows.

3.3.1 Prior on the Tree Structure

The set $\mathcal{F}_D$ of complete suffix dictionaries is equivalent to the set of complete d-ary trees of depth smaller than D, namely the set of trees where each node has either d sons or none, up to nodes of depth D which can only be leaves. Following [33] we define a simple probability

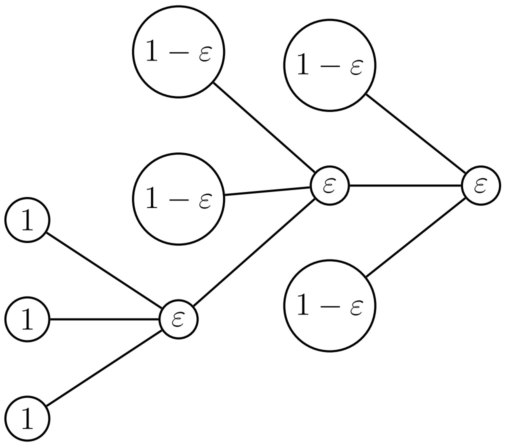

Fig. 2. Branching-process generation of the example shown in Figure 1 with a depth $D = 3$. The prior value for that tree is $\varepsilon^3(1-\varepsilon)^4$.

π_D on the set $\mathcal{F}_D$ of trees that is the direct translation of an intuitive random generation of trees stopped at depth D. Starting from the root, the tree generation process follows recursively the following rule: up to depth $D-1$, each node has probability ε of giving birth to d children, and probability $1-\varepsilon$ of having no children, that is probability $1-\varepsilon$ of becoming a leaf; if the node is however located at depth D of the tree, it becomes automatically a leaf with no successors. In mathematical terms, this defines a branching process on d-ary trees, truncated at depth D. The typical outcome of this generation is completely parameterized by ε, since a low value will favour short-depth trees while values closer to 1 will yield fully grown trees of depth D up to the case where $\varepsilon = 1$ and only the full tree of depth D is considered. If we denote by $\overset{\circ}{\mathcal{D}}$ the set of all strict suffixes (corresponding to inner nodes of the tree) of elements of $\mathcal{D}$, the probability of a tree is given by:

$$\pi_D(\mathcal{D}) = \prod_{s \in \overset{\circ}{\mathcal{D}}} \varepsilon \prod_{\substack{s \in \mathcal{D} \\ l(s) < D}} (1-\varepsilon) = \varepsilon^{\frac{|\mathcal{D}|-1}{d-1}} (1-\varepsilon)^{\operatorname{card}\{s \in \mathcal{D} \,|\, l(s) < D\}} . \tag{5}$$

This probability is illustrated with the case of the tree shown in Figure 2, with a prior value for that example of $\varepsilon^3(1-\varepsilon)^4$.

3.3.2 Priors on Multinomial Parameters

For a given tree $\mathcal{D}$ we now define a prior on the family of multinomial parameters $\Theta_{\mathcal{D}} = (\Sigma_d)^{\mathcal{D}}$ which fully characterizes a context-tree distribution based on a dictionary of suffixes $\mathcal{D}$. We assume an independent prior among multinomials attached to each of those suffixes as

$$\pi(d\theta|\mathcal{D}) = \prod_{s \in \mathcal{D}} \omega(d\theta_s),$$

where ω is a prior distribution on the simplex Σ_d. Following [33] a simple choice is to make use of Dirichlet priors:

$$\omega_\beta(d\theta) = \frac{1}{\sqrt{d}} \frac{\Gamma(\sum_{i=1}^d \beta_i)}{\prod_{i=1}^d \Gamma(\beta_i)} \prod_{i=1}^d \theta_i^{\beta_i - 1} \lambda(d\theta),$$

where λ is Lebesgue's measure and $\beta = (\beta_i)_{i=1..d}$ is the parameter of the Dirichlet distribution. The parameter β incorporates all the prior belief we have on the distribution of the alphabet. It can be either tuned based on empirical data or chosen having theoretical considerations in mind. A natural choice in the latter case is to use Jeffrey's prior [1, p.44] also known as the Krichevski-Trofimov prior [33] and set $\beta_i = \frac{1}{2}$ for $1 \le i \le d$. Alternative choices, such as Laplace's successor rule ($\beta_i = 1$) or the Schurmann-Grassberger estimate ($\beta_i = \frac{1}{d}$) have been advocated in the literature and will also be explored in the experimental section of this work, taking into account discussions presented in [23] for instance. Furthermore, the use of a simple Dirichlet prior can be extended to additive mixtures of Dirichlet priors since the latter have been shown to incorporate more efficiently information on the distributions of amino-acids [5]. We propose to include such priors in the construction of our kernel and extend the computational framework of the CTW by doing so. An additive mixture of n Dirichlet distributions is defined by a family of n Dirichlet parameters $\beta^{(1)}, \dots, \beta^{(n)}$ and n weights $\gamma^{(1)}, \dots, \gamma^{(n)}$ (with $\sum_{k=1}^n \gamma^{(k)} = 1$) to yield the prior:

$$\omega_{\gamma,\beta}(d\theta_s) = \sum_{k=1}^n \gamma^{(k)} \, \omega_{\beta^{(k)}}(d\theta_s). \tag{6}$$

3.4 Triple Mixture Context-Tree Kernel

Combining the definition of the kernel of Equation (3) with the definition of the context-tree model distributions in Equation (4) and of the priors on the set of distributions of Equations (5), (6), we obtain the following expression for the context-tree kernel:

$$\kappa_\sigma(X,Y) = \sum_{\mathcal{D} \in \mathcal{F}_D} \pi_D(\mathcal{D}) \int_{\Theta_\mathcal{D}} P_{\mathcal{D},\theta}(X)^{\frac{\sigma}{N_X}} \; P_{\mathcal{D},\theta}(Y)^{\frac{\sigma}{N_Y}} \prod_{s \in \mathcal{D}} \left(\sum_{k=1}^n \gamma^{(k)} \omega_{\beta^{(k)}}(d\theta_s) \right). \tag{7}$$

We observe that Equation (7) involves three summations respectively over the trees, the Dirichlet components used in our additive mixtures, and the multinomial parameters over which a Bayesian averaging is performed. This generalizes the double mixture performed in [33] in the context of sequence compression by adding a mixture of Dirichlet priors.

4 Kernel Implementation

As pointed out in the introduction, the models and priors selected to define the mutual information kernel of Equation (7) may not fit in the best way the natural process which generates the considered sequences. Some distributions favoured by these priors may not even correspond to the ones that are frequently observed in sequences generated by the natural phenomenon. While this may already seem arguably not so important in the context of this paper (which highlights feature extraction as opposed to parameter estimation), we also advocate such choices having in mind they yield an efficient computation of the value of Equation (7).

For $r \in \mathbb{N}$, and $\beta = (\beta_i)_{1 \le i \le r} \in (\mathbb{R}^{+*})^r$ and $\alpha = (\alpha_i)_{1 \le i \le r} \in (\mathbb{R}^{+})^r$ we write $\mathbf{G}_\beta(\alpha)$ for

$$\mathbf{G}_\beta(\alpha) \overset{\text{def}}{=} \int_{\Sigma_r} \prod_{i=1}^{r} \theta_i^{\alpha_i} \omega_\beta(d\theta) = \frac{\Gamma(\beta_{\bullet})}{\prod_{i=1}^{r} \Gamma(\beta_i)} \frac{\prod_{i=1}^{r} \Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_{\bullet} + \beta_{\bullet})},$$

where Γ is the Gamma function, Σ_r the r-dimensional simplex, $\beta_{\bullet} = \sum_{i=1}^{r} \beta_i$, and $\alpha_{\bullet} = \sum_{i=1}^{r} \alpha_i$. The quantity $\mathbf{G}_\beta(\alpha)$ corresponds to the averaging of the multinomial likelihood $P_\theta(\alpha)$ under a Dirichlet prior of parameter β when θ spans Σ_r.

The computation of the context-tree kernel on two strings can be divided into two phases for more clarity, which can be implemented alongside each other. A look at Figure 3 may give a better intuition on the computations actually performed by the CTW algorithm.

4.1 Defining Counters

The first step of the algorithm is to compute for $m \in E^D$ the counter

$$\rho_m(X) \overset{\text{def}}{=} \sum_{i=1}^{N_X} \quad (x_c^i = m),$$

which simply counts the occurrences of m within contexts enumerated in X. For contexts present in the string X, that is words m such that $\rho_m(X) > 0$, the empirical behaviour of transitions can be estimated as

$$\hat{\theta}_{m,e}(X) \overset{\text{def}}{=} \frac{\sum_{i=1}^{N_X} \quad (x_c^i = m, x_l^i = e)}{\rho_m(X)}.$$

$\hat{\theta}_{m,e}$ summarizes the empirical probability of the appearance of letter e after m has been observed. We finally define a last counter:

$$a_{m,e}(X,Y) \overset{\text{def}}{=} \frac{\rho_m(X)}{N_X} \hat{\theta}_{m,e}(X) + \frac{\rho_m(Y)}{N_Y} \hat{\theta}_{m,e}(Y).$$

$a_{m,e}(X,Y)$ is a weighted average of the transitions encountered in X and Y. Once those counters are computed on visited contexts, which are up to $N_X + N_Y$, the following downward recursion on the length of the string m (when m spans all suffixes of visited contexts)

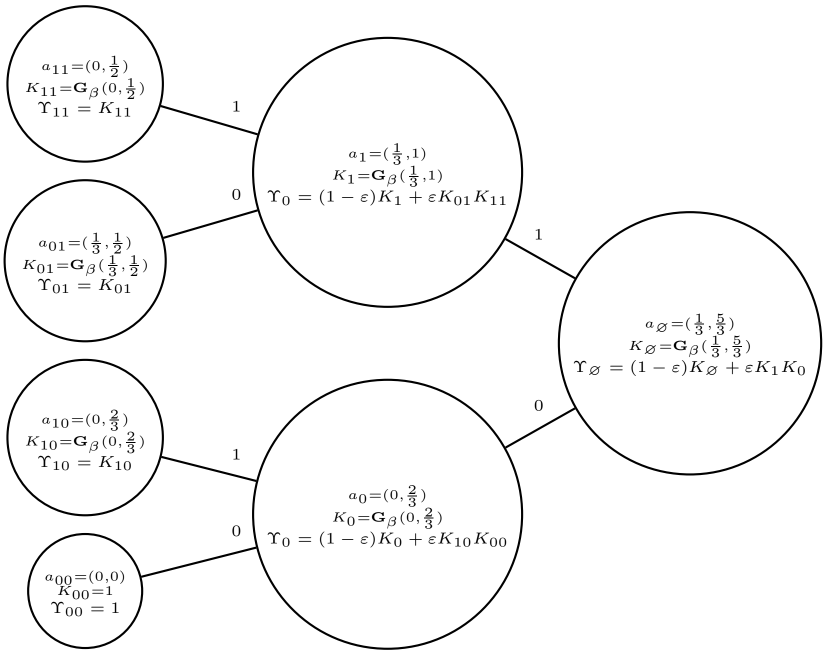
$a_{11}=(0,\frac{1}{2})$
$K_{11}=\mathbf{G}_\beta(0,\frac{1}{2})$
$\Upsilon_{11}=K_{11}$
1
$a_{01}=(\frac{1}{3},\frac{1}{2})$
$K_{01}=\mathbf{G}_\beta(\frac{1}{3},\frac{1}{2})$
$\Upsilon_{01}=K_{01}$
0
$a_1=(\frac{1}{3},1)$
$K_1=\mathbf{G}_\beta(\frac{1}{3},1)$
$\Upsilon_0=(1-\varepsilon)K_1+\varepsilon K_{01}K_{11}$
1
$a_\varnothing=(\frac{1}{3},\frac{5}{3})$
$K_\varnothing=\mathbf{G}_\beta(\frac{1}{3},\frac{5}{3})$
$\Upsilon_\varnothing=(1-\varepsilon)K_\varnothing+\varepsilon K_1K_0$
0
$a_{10}=(0,\frac{2}{3})$
$K_{10}=\mathbf{G}_\beta(0,\frac{2}{3})$
$\Upsilon_{10}=K_{10}$
1
$a_0=(0,\frac{2}{3})$
$K_0=\mathbf{G}_\beta(0,\frac{2}{3})$
$\Upsilon_0=(1-\varepsilon)K_0+\varepsilon K_{10}K_{00}$
0
$a_{00}=(0,0)$
$K_{00}=1$
$\Upsilon_{00}=1$

Fig. 3. CTW calculation tree for two binary strings $X = 0111$ and $Y = 10101$, with a depth $D = 2$, $\sigma = 1$ and an arbitrary Dirichlet parameter β. The two string are considered as sets of weighted transitions $X = \{(01, 1), (11, 1)\}$ and $Y = \{(10, 1), (10, 1), (01, 0)\}$, and the resulting kernel value $K(X, Y)$ is $\Upsilon_\varnothing$.

computes equivalent counters for shorter suffixes:

$$\begin{aligned}
\rho_m(X) &= \sum_{f\in E} \rho_{f.m}(X),\\
\hat{\theta}_{m,e}(X) &= \frac{\sum_{f\in E}\ \rho_{f.m}(X)\ \hat{\theta}_{f.m,e}(X)}{\rho_m(X)},\\
a_{m,e}(X,Y) &= \sum_{f\in E} a_{f.m,e}(X,Y).
\end{aligned}$$

So far, the memory needed to store the information on which the kernel will be computed (essentially counters a which can be stored in the leaves of a suffix tree generated while scanning only visited contexts) is linear with respect to the size of our strings and is loosely upper-bounded by $D(N_X + N_Y)$.

4.2 Recursive Computation of the Triple Mixture

We can now attach to each m for which we have calculated the previous counters the value:

$$K_m(X,Y) = \sum_{k=1}^{n} \gamma^{(k)} \mathbf{G}_{\beta^{(k)}} \left(\sigma \cdot a_{m,e}\left(X,Y\right)_{e\in E}\right),$$

which computes two mixtures, the first being a continuous Bayesian averaging on the possible values of θ weighted by a given Dirichlet prior and the second being a discrete weighted summation using the weighted Dirichlet distributions provided by the mixture $(\gamma^{(k)}, \beta^{(k)})_{k=1..n}$.

A numerical approximation of $\mathbf{G}_{\beta(k)}$ can be used in practice, through Lanczos' approximation of the $\ln\Gamma$ function for instance. By defining now the quantity $\Upsilon_m(X,Y)$, which is also attached to each visited word m and computed recursively through

$$\Upsilon_m(X,Y) = \begin{cases} K_m(X,Y) & \text{if } l(m) = D, \\ (1-\varepsilon)K_m(X,Y) + \varepsilon \prod_{e\in E} \Upsilon_{e.m}(X,Y) & \text{if } l(m) < D. \end{cases}$$

we actually perform the third mixture over all possible tree structures by taking into account the branching probability ε. Indeed, we finally have, recalling $\varnothing$ is the empty word, that:

$$\kappa_\sigma(X,Y) = \Upsilon_\varnothing(X,Y). \tag{8}$$

Proof. For a c.s.d model $(\mathcal{D},\theta)$ and two sets of transitions $X = (x_c^i, x_l^i)_{i=1\le N_X}$ and $Y = (y_c^i, y_l^i)_{1\le i\le N_Y}$ we have that

$$P_{\mathcal{D},\theta}(X)^{\frac{\sigma}{N_X}} P_{\mathcal{D},\theta}(Y)^{\frac{\sigma}{N_Y}} = \prod_{s\in\mathcal{D}}\prod_{e\in E}\theta_s(e)^{\sigma a_{s,e}(X,Y)}.$$

The latter product of likelihoods can thus be calculated using only counter a, and we further have that

$$\begin{aligned} &\int_{\Theta_\mathcal{D}} P_{\mathcal{D},\theta}(X)^{\frac{\sigma}{N_X}} P_{\mathcal{D},\theta}(Y)^{\frac{\sigma}{N_Y}} \prod_{s\in\mathcal{D}}\left(\sum_{k=1}^n \gamma^{(k)}\omega_{\beta^{(k)}}(d\theta_s)\right) \\ &\quad = \int_{\Theta_\mathcal{D}} \prod_{s\in\mathcal{D}}\left[\prod_{e\in E}\theta_s(e)^{\sigma a_{s,e}(X,Y)}\left(\sum_{k=1}^n \gamma^{(k)}\omega_{\beta^{(k)}}(d\theta_s)\right)\right] \\ &\quad = \prod_{s\in\mathcal{D}}\sum_{k=1}^n \gamma^{(k)}\int_{\Sigma_d}\prod_{e\in E}\theta_s(e)^{\sigma a_{s,e}(X,Y)}\omega_{\beta^{(k)}}(d\theta_s) \\ &\quad = \prod_{s\in\mathcal{D}}\sum_{k=1}^n \gamma^{(k)}\mathbf{G}_{\beta^k}\left(\sigma\left(a_{s,e}\left(X,Y\right)\right)_{e\in E}\right) = \prod_{s\in\mathcal{D}} K_s(X,Y), \end{aligned}$$

where we have used Fubini's theorem to factorize the integral in the second line. Having in mind Equation (7), we have thus proved that $\kappa_\sigma(X,Y) = \sum_{\mathcal{D}\in\mathcal{F}_D}\pi_D(\mathcal{D})\prod_{s\in\mathcal{D}}K_s(X,Y)$. The second part of the proof is identical to the one given in [33], and developed in [6] whose recursive treatment we adopt. Let us prove by induction, with respect to successively decreasing lengths of m (i.e., over words m such that $l(m) = D, ..., 0$), that

$$\Upsilon_m(X,Y) = \sum_{\mathcal{D}\in\mathcal{F}_{D-l(m)}} \pi_{D-l(m)}(\mathcal{D})\prod_{s\in\mathcal{D}} K_{s.m}(X,Y), \tag{9}$$

where $\pi_{D-l(m)}$ is the distribution of a tree according to the branching process prior previously presented stopped at level $D-l(m)$. We notice that the set $\mathcal{F}_{D-l(m)}$ of c.s.d's of depth $D-l(m)$ can be further divided into:

$$\mathcal{F}_{D-l(m)} = \Big\{\{(s,y) : y \in E, s\in\mathcal{D}_y\}, \mathcal{D}_y \in \mathcal{F}_{D-l(m)-1}\Big\} \cup \{\{\varnothing\}\},$$

where we have that:

$$\pi_{D-l(m)}(\{(s,y) : y \in E, s \in \mathcal{D}_y\}) = \varepsilon \prod_{y\in E} \pi_{D-l(m)-1}(\mathcal{D}_y),$$
$$\pi_{D-l(m)}(\{\varnothing\}) = 1 - \varepsilon.$$

Starting our recursion with words of length $d = D$, where Equation (9) is valid by the recursive definition of Υ, we assume Equation (9) to be valid with words of length d and prove that it holds for words of length $d-1$. Given m such that $l(m) = d-1$, we can write:

$$\begin{aligned}
\Upsilon_m(X,Y) &= (1-\varepsilon)K_m(X,Y) + \varepsilon \prod_{y\in E} \Upsilon_{y.m}(X,Y) \\
&= (1-\varepsilon)K_m(X,Y) + \varepsilon \prod_{y\in E} \sum_{\mathcal{D}\in\mathcal{F}_{D-d}} \pi_{D-d}(\mathcal{D}) \prod_{s\in\mathcal{D}} K_{s.y.m}(X,Y) \\
&= (1-\varepsilon)K_m(X,Y) + \varepsilon \sum_{(\mathcal{D}_y)\in(\mathcal{F}_{D-d})^E} \prod_{y\in E} \pi_{D-d}(\mathcal{D}_y) \prod_{s\in\mathcal{D}_y} K_{s.y.m}(X,Y) \\
&= \pi_{D-l(m)}(\{\varnothing\})K_m(X,Y) \\
&\quad + \sum_{(\mathcal{D}_y)\in(\mathcal{F}_{D-d})^E} \pi_{D-l(m)}(\{(s,y) : y \in E, s \in \mathcal{D}_y\}) \prod_{(s,y)\in\mathcal{D}_y\times E} K_{s.y.m}(X,Y) \\
&= \sum_{\mathcal{D}\in\mathcal{F}_{D-d}} \pi_{D-d}(\mathcal{D}) \prod_{s\in\mathcal{D}} K_{s.m}(X,Y)
\end{aligned}.$$

Applying Equation (9) to the case where $m = \varnothing$ we finally prove Equation (8).■

As previously recalled, the computation of the counters has a linear cost in time and memory with respect to $D(N_X + N_Y)$. As only counters that correspond to visited suffixes of X and Y are created, recursive computation of Υ_m is also linear in time and space (the values Υ_m for suffixes m not encountered, such that $\rho_m(X) = \rho_m(Y) = 0$, being equal to 1). As a final result, the computation of the kernel is linear in time and space with respect to $D(N_X+N_Y)$.

5 Source Coding and Compression Interpretation

There is a very classical duality between source distributions (a random model to generate infinite sequences) and sequence compression [7]. Roughly speaking, if a finite sequence X has a probability $P(X)$ of being generated by a source distribution P, then one can design a binary code to represent X by $r(X) = -\log_2 P(X)$ bits, up to 2 bits, using for example arithmetic coding. In this section, we provide an interpretation of the context-tree kernel in terms of information theory and compression, and highlight its differences with the spectrum kernel.

When sequences are generated by an unknown source P, it is classical to form a coding source distribution by averaging several a priori sources. Under reasonable assumptions, one can design this way universal codes, in the sense that the average length of the codes be almost as short as if P was known and the best source was used. As an example, the context-tree weighting (CTW) algorithm [33] defines a coding probability P_π for sequences by averaging source distributions defined by context-trees as follows:

$$P_\pi(X) \overset{\text{def}}{=} \sum_{\mathcal{D}\in\mathcal{F}_D} \pi(\mathcal{D}) \int_{\Theta_\mathcal{D}} P_{\mathcal{D},\theta}(X) \prod_{s\in\mathcal{D}} \omega_\beta(d\theta_s), \tag{10}$$

where ω_β is the Krichevski-Trofimov prior. Up to the mixture of Dirichlet and the exponents (used to renormalize the probabilities with respect to the sequences' lengths), we therefore see, by comparing (10) with (7), that the context-tree kernel between two sequences can be roughly interpreted as the probability under P_π of the concatenation of the two sequences. Our kernel actually considers a sequence as a list of weighted empirical distributions $\{(\rho_m, \hat{\theta}_m)\}_{m\in E^D} \in (\mathbb{R}^+ \times \Sigma_d)^{E^D}$ which summarizes the local behaviour of its letter transitions. These coordinates, whose information is equivalent to the one contained in the spectrum of the sequence, can be used to compute the likelihood of a specific context-tree distribution $(\mathcal{D}, \theta)$ on such a set by deriving $\{(\rho_s, \hat{\theta}_s), s \in \mathcal{D}\}$ recursively, as in the previous computation.

We write $\text{kl}(\theta||\theta')$ for the Kullback-Leibler divergence between θ and θ', two multinomial parameters of size d, i.e $\text{kl}(\theta||\theta') = \sum_{i=1}^d \theta_i \ln\frac{\theta_i}{\theta'_i}$. We also note $h(\theta)$ the entropy of θ, i.e., $h(\theta) = -\sum_{i=1}^d \theta_i \ln\theta_i$. We use the following identity on θ and θ',

$$\begin{aligned}\prod_{i=1}^d \theta_i^{\theta'_i} &= e^{\sum_{i=1}^d \theta'_i \ln\theta_i}\\ &= e^{\sum_{i=1}^d \theta'_i \ln\frac{\theta_i}{\theta'_i} + \sum_{i=1}^d \theta'_i \ln\theta'_i}\\ &= e^{-h(\theta')-kl(\theta'||\theta)},\end{aligned}$$

to reformulate the mixture coding probability P_π on $\mathcal{X}$ in the context of the context-tree kernel computation. Indeed, following the priors previously defined on $\mathcal{P}_D$, the following formula expresses the value of the coding probability of a given string through its counters ρ and $\hat{\theta}$:

$$P_\pi(\rho,\hat{\theta}) = \sum_{\mathcal{D}\in\mathcal{F}_D} \pi(\mathcal{D}) \prod_{s\in\mathcal{D}} e^{-\sigma\rho_s h(\hat{\theta}_s)} \int_{\Sigma_d} e^{-\sigma\rho_s kl(\hat{\theta}_s||\theta)}\ \omega_{\gamma,\beta}(d\theta).$$

We write r_π for $-\ln P_\pi$, $\hat{\rho}(X)$ for the normalized counters $\frac{1}{N_X}\rho(X)$ and introduce the following function t_π of two strings,

$$\begin{aligned}t_\pi(X,Y) =& \frac{1}{2}\left[r_\pi\left(\hat{\rho}(X),\hat{\theta}(X)\right) + r_\pi\left(\hat{\rho}(Y),\hat{\theta}(Y)\right)\right]\\ &-\ r_\pi\left(\frac{\hat{\rho}(X)+\hat{\rho}(Y)}{2}, \frac{\hat{\theta}(X)+\hat{\theta}(Y)}{2}\right).\end{aligned} \tag{11}$$

Finally we have, by defining the renormalized kernel $\tilde{\kappa}_\sigma$ as

$$\tilde{\kappa}_\sigma(X,Y) = \kappa_\sigma(X,Y)/\sqrt{\kappa_\sigma(X,X)\kappa_\sigma(Y,Y)},$$

that

$$\tilde{\kappa}_\sigma(X,Y) = e^{-t_\pi(X,Y)}.$$

We note here that the function t_π can be interpreted in the light of semigroup kernels on sets of components or measures, as proposed in [9,8]. A semigroup is roughly a set with an associative composition law, which in our case is just the addition of counters and estimated transitions as in Equation (11). What the structure of t_π highlights is that the similarity computed by context-tree kernels between two strings, and more precisely the sequences of counters indexed on E^D that describe them, is just a function of their sum. This is opposed to the computations led by the spectrum kernel, which considers products on those counters (namely a linear-dot product on those vectors of counters). The whole family of context-tree kernels are hence defined through a prior belief on the behaviour of sequences of counters (tuned through a selection of specific priors), which is first applied to the sequences individually, $\left(\hat\rho(X), \hat\theta(X)\right)$ and $\left(\hat\rho(Y), \hat\theta(Y)\right)$, before evaluating it on their mean $\left(\frac{\hat\rho(X)+\hat\rho(Y)}{2}, \frac{\hat\theta(X)+\hat\theta(Y)}{2}\right)$. This formulation makes the link with compression more precise, where instead of concatenating strings we rather perform counter averaging. This viewpoint can also bring forward a geometrical perspective on the actual computation which is performed. The choice of a compression algorithm (namely a selection of priors) defines the shape of the function r_π on the whole space of counters, and the similarity between two sequences is measured through the difference between three evaluations of r_π, first taken on the two points taken apart and then on their average, which is directly related to the convexity of r_π.

6 Experiments

6.1 Protein Domain Homology Detection Benchmark

We report results concerning the performance of the context-tree family of kernels on a benchmark experiment that tests the capacity of SVMs to detect remote homologies between protein domains. This is simulated by recognizing domains that are in the same SCOP (Structural Classification of Proteins [15], ver. 1.53) superfamily, but not in the same family, using the procedure described in [16]. We used the files compiled by the authors of [24], which consist in 4352 sequences extracted from the Astral database of protein domains. For each of the 54 tested families, the protein domains within the family where considered positive *test* examples while protein domains within the superfamily but outside the family were considered as positive *training* examples. This results in 54 classification experiments with at least 10 positive training examples and 5 positive test examples. Negative examples were selected outside of the positive sequences' fold with a similar ratio. Following previous studies of this benchmark, we computed the ROC (Receiving Operator Characteristic, [12]), ROC50 and RFP (Rate of False Positives) of each of the classification performed by a SVM based on various parameter settings of the context-tree kernel. The ROC score (or AUC, Area Under the ROC Curve) is the normalized area under the curve which plots the number of true positives as a function of false positives; the ROC50 is the area under the ROC curve up to 50 false positives while the median RFP is the number of false positives scoring as high or better than the median scoring true positives. We average those criterions on the 54 experiments to provide an overall measure of the performance of the considered kernels on this task.

6.2 Parameter Tuning and Comparison with Alternative String Kernels

Let us now recall, along with the formula of the context-tree kernel, the different parameters which need to be set to control the output of the family of context-tree kernels;

$$\kappa_\sigma(X,Y) = \sum_{\mathcal{D}\in\mathcal{F}_D} \pi_D(\mathcal{D}) \int_{\Theta_\mathcal{D}} P_{\mathcal{D},\theta}(X)^{\frac{\sigma}{N_X}} P_{\mathcal{D},\theta}(Y)^{\frac{\sigma}{N_Y}} \prod_{s\in\mathcal{D}} \left(\sum_{k=1}^{n} \gamma^{(k)} \omega_{\beta^{(k)}}(d\theta_s)\right).$$

- σ represents the width taken by the probabilities used to compute the kernel, allowing us to control the range of values appearing in Gram matrices. Large values of σ will favor diagonal-dominant matrices while lower values will tend to create Gram matrices of similar elements. We thus tuned these values empirically, so that none of the two previous problematic cases appears. Using a σ value between 1 and 5 typically ensures this and we usually set $\sigma = 2$.
- The branching-process probability π_D is parameterized by ε, which controls the typical amount of suffixes numbered in dictionaries in relation with D, their maximal depth. A sound choice for ε, as well as being validated by experiments and used in the original paper [33] is to set $\varepsilon = 1/d$, as this keeps a good balance between small trees which might capture simple interactions and larger trees which might detect longer range interactions but which may also be accused of overfitting the data by mapping them on too complex models.
- The depth parameter D controls the maximal memory of our Markovian models. This parameter influences the complexity of our features extractors and adds computational time to most calculations. The submitted sequences have typical lengths of roughly two hundred amino-acids. Hence lengths set between 2 and 4 for our substrings (that is contexts of length 1 to 3) should suffice to capture most of the available information, following the empirical observation of [21] that the logarithm of the average length of the sequences suffices as a context length to capture most of the letter-to-letter transition information. Those lengths were also shown to give the best performance on our datasets.
- Finally, different Dirichlet priors but also families of Dirichlet mixtures $(\gamma^{(k)}, \beta^{(k)})_{1\leq k\leq n}$ can be considered to compute mixtures at the level of each node. We tested three popular uniform priors, including the Jeffrey's prior ($\beta_i = 1/2$) used originally in [33] and usually favoured in the context of universal coding, but also Laplace's successor rule ($\beta_i = 1$) and the Schurmann-Grassberger estimate ($\beta_i = 1/d$). Both first choices appeared effective and similar in result while the last led to poor results. We also tested Dirichlet mixtures, hoping they would increase the accuracy compared to the use of uniform priors. We considered 3, 9 and 20 components additive mixtures (respectively `hydro-cons.3comp`; `byst-4.5-0-3.9comp`; `recode3.20comp`, `fournier20.comp` and `dist20.comp`) which were

all publicly available and downloaded from a Dirichlet mixture repository [2]. These mixtures gave disappointing results when averaged over the 54 families (considering ROC average this means a performance of roughly 87% to 88%) but produced somehow different results for some families which seemed hard to classify through other methods. However, we interpret the fact that those families of Dirichlet mixtures did not improve overall accuracy as a form of overfitting. Again, while this biological knowledge might improve the selection of a specific model to fit sequences (notably Hidden Markov Models), it does not seem to work in our framework where we only use statistical models as feature extraction tools.

Up to the poor performances of context-tree kernels defined with Dirichlet mixtures, the few experiments we led on different parameters yielded no surprises and favoured ranges of parameters which were theoretically motivated, namely short depths, a branching process prior of roughly $1/d$ and uniform Dirichlet priors (either the Laplace of the Krichevski-Trofimov rule). Note further that the variety of all 54 protein families used in the experiment prevents overfitting since an increase in performance over certain families usually implies a decrease in other ones. We compare the performance of context-tree kernels with other string kernels, where the performances we report were computed according to the parameters known to perform in a good way on that dataset and proposed by the respective authors of those kernels. We present here the best mismatch kernel (5,1) reported in [20], which can also be computed in linear time and space, but also more greedy algorithms such as the pairwise kernel [22] and the two local alignment kernels (LA-Eig, LA-Ekm) presented in [31], which, as opposed to the context-tree Kernel, take into account relevant information known to be of capital importance for biological sequences (such as gaps, deletions or mutations of amino-acids). We also report the results of the spectrum kernel [19] with depth 3 and 4 and show that based on the same information (D-grams) the context-tree kernel clearly outperforms the latter. The classification was led using the Gist (version 2.1.1) implementation of SVM [3], where all parameters specific to SVM optimization were set to default values (elementary attempts to tune the latter parameters did not yield to significative improvements in accuracy).

6.3 Mean Performances and Curves

We present in Figure 4 the performance of all previously quoted kernels, along with an implementation of the context-tree kernel where $\sigma = 2$, $D = 4$, $\varepsilon = 1/20$ and where a uniform Jeffrey prior was used. The results show that the CTK performs roughly better than the mismatch kernel and overall similarly to the pairwise kernel, notably in regions where classification becomes more difficult and ROC scores become lower for all techniques. Except in those regions, it is outperformed by both versions of the local-alignment kernels. The CTK is computed in linear time and without any biological knowledge, a property exclusively shared with the spectrum kernel which performs much worse (only results obtained for the spectrum with a depth 3 have been represented in the plot).

[2] `http://www.cse.ucsc.edu/research/compbio/dirichlets/`

[3] `http://microarray.cpmc.columbia.edu/gist/download.html`

Method	ROC	ROC50	RFP
CTK	0.894	0.371	0.0869
Spectrum 3	0.781	0.277	-
Spectrum 4	0.716	0.208	-
Mismatch 5,1	0.872	0.400	0.0837
Pairwise	0.894	0.461	0.0846
LA-ekm	0.934	0.663	0.0525
LA-eig	0.923	0.646	0.0552

Table 1
Mean results for ROC, ROC50 and RFP as produced over the 54 families by all compared kernels, where CTK denotes the context-tree kernel set with $\sigma = 2$, $\varepsilon = 1/20$, Jeffrey's prior and depth $D = 4$.

Table 1 summarizes the three main statistics used to compare performances over the studied benchmark between context-tree kernels and all other kernels. In this table, context-tree kernels perform (relatively to other kernels) better in terms of ROC score than in terms of ROC50 and RFP, and we have no explanation for this. As can be easily deduced from the previous figure, the context-tree kernel clearly outperforms the spectrum kernel while using exactly the same information. In the general case where only the spectrum information of a string is available, the context-tree kernel may hence prove more useful than the simple spectrum kernel.

Additionally, we report that using the 20-components Dirichlet mixture `fournier20` with usual parameters ($D = 4$ and $\varepsilon = 0.05$) produced the triplet of means $(0.887, 0.366, 0.096)$. Simpler mixtures, that is with less components, did not yield a substantial increase in performance either and we hence did not find them useful in the context of this experiment since they add computational cost. However, we observed important variations on the performance for each family with respect to other context-tree kernels which only use uniform priors, while their overall performance was similar or slightly worse. This might be interpreted as some complementary between the two kinds of kernels (using mixtures of Dirichlet priors and just a single component) and may be a subject of future research, through a linear combination of kernels for instance. Finally we present in Table 2 a few results for meaningful settings of the context-tree kernels using Jeffrey's prior. These results show clearly that an increase in the complexity of the models used to perform the Bayesian mixture does not yield better results in practice. Surprisingly, a context-tree kernel of depth 1 suffices to provide good results, while more complex models which require far more computational cost give relatively poor results. These observations show once more that in the context of mutual information kernels, the relevance of distributions to model the data does not seem to be an important criterion.

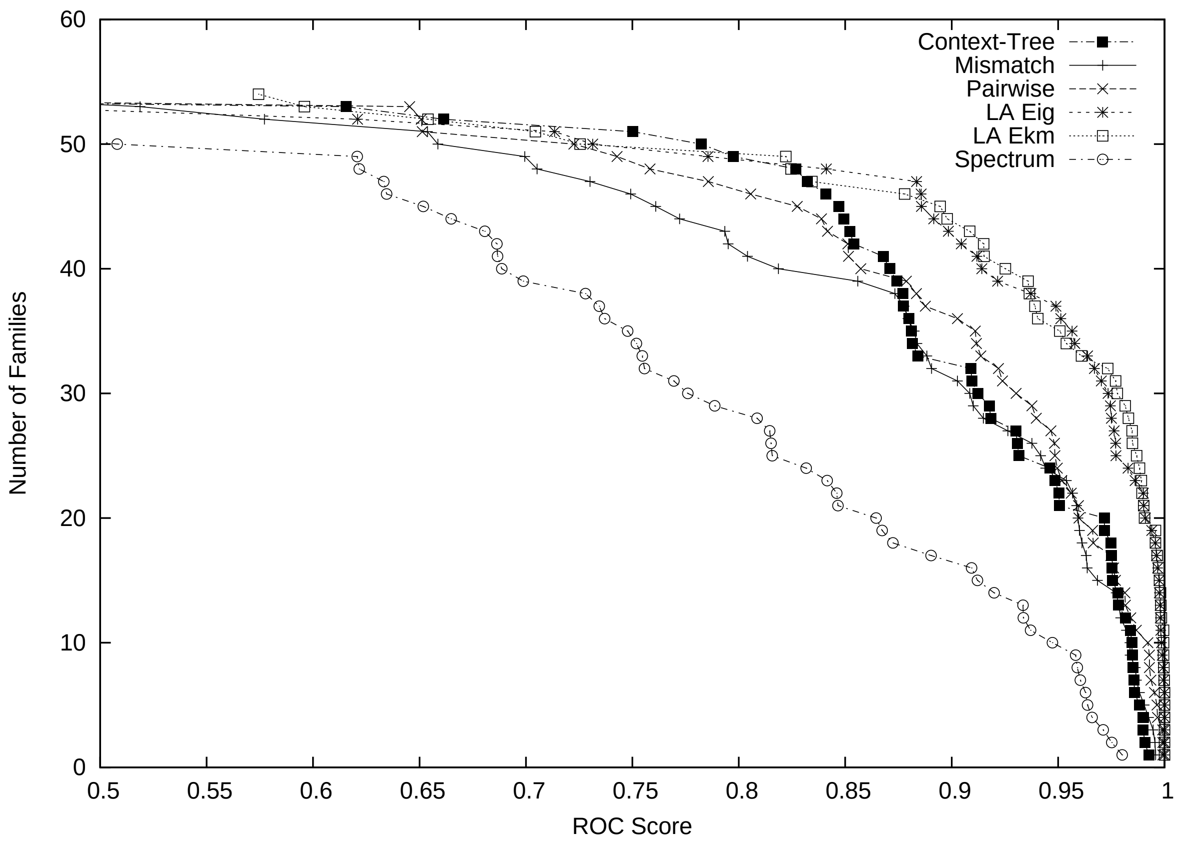

Fig. 4. Performance of all considered kernels on the problem of recognizing domain's superfamily. The curve shows the total number of families for which a given methods exceeds a ROC score threshold. CTK denotes the context-tree kernel set with $\sigma = 2$, $\varepsilon = 1/20$, Jeffrey's prior and depth $D = 4$.

7 Conclusion

We introduced a novel class of kernels for sequences that are fast to compute while only using the spectrum of the submitted strings. The kernel is a mutual information kernel based on a family of context-tree models, and makes a link between the comparison of two string and the ability of universal coding algorithms to compress them when taken together. On a benchmark experiment of remote homology detection it performs at a level close to state-of-the-art levels reached by kernels which involve heavier computational cost and make use of biological knowledge. The context-tree kernels clearly outperform the spectrum kernel on the same benchmark while using exactly the same information. The context-tree kernel, whose computation is inspired by universal coding theory, may thus share one of the qualities of the latter algorithms, which is to appear as a sound prior choice to explore similarities between sequences for whom little knowledge is available and at a reasonable computational cost.

Parameters (with Jeffrey's prior and $\sigma = 2$)	ROC	ROC50	RFP
$D = 1, \ \ \varepsilon = 1/20$	0.886	0.373	0.0796
$D = 2, \ \ \varepsilon = 1/20$	0.892	0.391	0.0857
$D = 3, \ \ \varepsilon = 1/20$	0.895	0.385	0.0865
$D = 4, \ \ \varepsilon = 1/20$	0.894	0.371	0.0869
$D = 4, \ \ \varepsilon = 1/4$	0.893	0.378	0.0857
$D = 4, \ \ \varepsilon = 1/2$	0.889	0.367	0.0877
$D = 4, \ \ \varepsilon = 1$	0.872	0.326	0.101
$D = 6, \ \ \varepsilon = 1/20$	0.889	0.362	0.0923
$D = 8, \ \ \varepsilon = 1/20$	0.885	0.355	0.0986

Table 2
From short trees to long and dense trees: mean results of ROC, ROC50 and RFP scores for different settings of the branching process prior and of the length of the models selected. Note that when only the complete tree is selected ($\varepsilon = 1$) the performance decreases significantly. In that case, namely when no mixture is performed on the class of models, the context-tree computation resembles the simpler computation performed by the spectrum kernel. Note also that a good performance is reached when the context-tree only uses contexts of length 1 (namely Markov chains of depth 1), which shows that models should be selected to extract features and not to model sequences, a hint which is further confirmed by the fact that long trees do not perform very well despite their better ability to absorb more knowledge about the strings' transitions.

8 Acknowledgments

The authors would like to specially thank Hiroto Saigo for providing us with pieces of code and results on the studied benchmark, and Eugene T. Ie for additional data and results on the class of kernels proposed by his group at Columbia University. We would also like to thank Shun-Ichi Amari and Tatsuya Akutsu for fruitful discussions, Kenji Fukumizu and the ISM staff for their help and advice on computational issues, and anonymous reviewers for their accurate comments and insights on this work. Finally, MC acknowledges a JSPS doctoral short-term grant which made his stay in Japan possible. JPV is supported by NHGRI NIH award R33 HG003070 and by the ACI "Nouvelles Interfaces des Mathmatiques" of the French Ministry of Research. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] Shun-Ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry.* AMS vol. 191, 2001.

[2] G. Bejerano and G. Yona. Modeling protein families using probabilistic suffix trees. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the3rd Annual International Conference*

on Computational Molecular Biology (RECOMB), pages 15–24, Lyon, France, 1999. ACM Press.

[3] Asa Ben-hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 1:1, 2003.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[5] M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, pages 47–55, Menlo Park, CA, 1993. AAAI/MIT Press.

[6] Olivier Catoni. *Statistical learning theory and stochastic optimization, Saint-Flour lecture notes.* Springer Verlag, 2001.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991.

[8] Marco Cuturi, Kenji Fukumizu, and Jean philippe Vert. Semigroup kernels on measures. 2005. ISM Research Memorandum 933.

[9] Marco Cuturi and Jean-Philippe Vert. Semigroup kernels on finite sets. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17.* MIT Press, Cambridge, MA, 2005.

[10] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis - Probabilistic models of proteins and nucleic acids.* Cambridge University Press, Cambridge, UK, 1998.

[11] Eleazar Eskin, William Noble, and Yoram Singer. Protein family classification using sparse markov transducers. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, August 2000.

[12] M. Gribskov and N. Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching, 1996.

[13] David Haussler. Convolution kernels on discrete structures. Technical report, UC Santa Cruz, 1999. USCS-CRL-99-10.

[14] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728, 2001.

[15] T. Hubbard, A. Murzin, S. Brenner, and C. Chothia. Scop: a structural classification of proteins database, 1997.

[16] Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1,2):95–114, 2000.

[17] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In T. Faucett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press, 2003.

[18] Risi Kondor and Tony Jebara. A kernel between sets of vectors. In *ICML Proceedings*, 2003.

[19] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for svm protein classific ation. In *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 564–575. World Scientific, 2002.

[20] Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[21] Ming Li, Xin Chen, Xin Li, Bin Ma, and Vitanyi Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[22] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology*, pages 225–232, 2002.

[23] I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[24] William Stafford Noble and Li Liao. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, pages 225–232, 2002.

[25] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization , Optimization, and Beyond.* MIT Press, Cambridge, MA, 2002.

[26] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology.* MIT Press, Cambridge, MA, 2004.

[27] Bernhard Schölkopf, Jason Weston, Eleazar Eskin, Christina Leslie, and William Stafford Noble. A kernel approach for learning from almost orthogonal patterns. In Tapio Elomaa, Heikki Mannila, and Hannu 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. Springer, 2002.

[28] Matthias Seeger. Covariance kernels from bayesian generative models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 905–912, Cambridge, MA, 2002. MIT Press.

[29] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley, New-York, 1998.

[30] Jean-Philippe Vert. A tree kernel to analyze phylogenetic profiles. *Bioinformatics*, 18:S276–S284, 2002.

[31] Jean-Philippe Vert, Hiroto Saigo, and Tatsuya Akutsu. Local alignment kernels for protein sequences. In Bernhard Schoelkopf, Koji Tsuda, and Jean-Philippe Vert, editors, *Kernel Methods in Computational Biology.* MIT Press, 2004.

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

[33] F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, pages 653–664, 1995.