BIOINFORMATICS
Vol. 1 no. 1 2002
Pages 1–9

A tree kernel to analyze phylogenetic profiles

Jean-Philippe Vert

Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto, 611-0011, Japan

ABSTRACT

Motivation: The phylogenetic profile of a protein is a string that encodes the presence or absence of the protein in every fully sequenced genome. Because proteins that participate in a common structural complex or metabolic pathway are likely to evolve in a correlated fashion, the phylogenetic profiles of such proteins are often "similar" or at least "related" to each other. The question we address in this paper is the following: how to measure the "similarity" between two profiles, in an evolutionarily relevant way, in order to develop efficient function prediction methods?

Results: We show how the profiles can be mapped to a high-dimensional vector space which incorporates evolutionarily relevant information, and we provide an algorithm to compute efficiently the inner product in that space, which we call the tree kernel. The tree kernel can be used by any kernel-based analysis method for classification or data mining of phylogenetic profiles.

As an application a Support Vector Machine (SVM) trained to predict the functional class of a gene from its phylogenetic profile is shown to perform better with the tree kernel than with a naive kernel that does not include any information about the phylogenetic relationships among species. Moreover a kernel principal component analysis (KPCA) of the phylogenetic profiles illustrates the sensitivity of the tree kernel to evolutionarily relevant variations.

Availability: All data and softwares used are freely and publicly available upon request.

Contact: Jean-Philippe.Vert@mines.org

INTRODUCTION

The availability of an increasing number of fully sequenced genomes has promoted the development of large-scale comparative genomic approaches to better understand the machinery of the cell. One such approach is the use of phylogenetic profiles (Pellegrini *et al.*, 1999) to assign functions to genes based on their patterns of inheritance across species. The hypothesis behind this method is that proteins that participate in a common structural complex or metabolic pathway evolve in a correlated fashion, and should therefore be present in the same organisms. As a matter of fact the phylogenetic profile of a gene has been shown to provide information about its function (Pellegrini *et al.*, 1999) and is an interesting alternative to other approaches to gene function prediction based on direct sequence comparisons or gene co-expression (Marcotte *et al.*, 1999).

In its simplest form the phylogenetic profile of a gene is a string of bits, each bit indicating the presence or absence of a homologue of the gene in a different organism. Once the phylogenetic profiles for all the genes of a given organism have been computed, the function of a gene can be inferred to some extent by examining the functions of other genes with "similar" or at least "related" profiles. A fundamental question addressed in this paper is therefore the following: how to measure the "similarity" between phylogenetic profiles?

The simplest approach to quantify the difference between two profiles is to count the number of organisms where they differ. Intuitively, the smaller this number, the more similar the profiles. This method was used in (Pellegrini *et al.*, 1999) where two profiles are defined as "neighbors" when they differ by less than 3 bits. A more appealing measure is the differential parsimony introduced in (Liberles *et al.*, 2002), based on the phylogenetic reconstruction of the ancestors and the comparison of the reconstructed trees. Indeed, it is intuitively more meaningful to compare the whole historical evolutions of two genes, rather than just their presence or absence in current organisms.

In this paper we generalize this idea and embed it into a powerful mathematical framework that opens new analysis opportunities. Instead of just defining a measure of similarity between profiles, such as the number of bits where they differ or the differential parsimony of the reconstructed phylogenetic trees, we propose to map each profile to a high-dimensional vector space (which we call *feature space*), defined in such a way that each coordinate in the feature space corresponds to one particular pattern of inheritance during evolution. As an example, one pattern of inheritance could be "the gene has been transmitted to proteobacteria and eukaryotes, but not to Gram-positive bacteria". As it is impossible to know for sure the ancestor genomes the mapping of a phylogenetic profile to the feature space is defined thanks to a probabilistic model of evolution (a classical Bayesian tree model) to give weights to features that correspond to

plausible patterns of inheritance for the given profile.

If one considers all possible patterns of evolution the dimension of the feature space is very large, and the explicit computation of the image of a profile could be prohibitive in terms of computation time and memory storage. However, as a main contribution, this paper provides an algorithm to compute very simply and quickly the inner product between the images of any two profiles in the feature space, which does not involve the explicit computation of each image.

The resulting function, which maps any two profiles to the inner product of their images in the high-dimensional feature space, is called a *tree kernel*. It belongs to a larger class of functions, called *kernels*, defined as the inner product of two objects mapped to any vector space. Once a kernel and a corresponding feature space are chosen, it is possible to define the distance between two profiles as the Euclidean distance between their images in the feature space. In the case of the tree kernel presented in this paper, two profiles are near each other in the feature space if they are likely to have shared common patterns of inheritance during evolution, which is an appealing property.

However, more than just computing a Euclidean distance can be done once a kernel is defined. In particular a whole set of algorithms, known as *kernel methods*, have been developed recently and shown to be very helpful in many real-world applications (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2001). Kernel methods work implicitly in the feature space using only the kernel function, and include such diverse algorithms as Support Vector Machines for classification or regression (Vapnik, 1998), kernel principal component analysis (Schölkopf *et al.*, 1999), kernel clustering (Ben-Hur *et al.*, 2001) or kernel Fisher discriminants (Mika *et al.*, 1999).

To illustrate the use of kernel methods to analyze phylogenetic profiles, we test the ability of a Support Vector Machine to predict the functional class of a gene from its phylogenetic profile, and show that incorporating knowledge about the phylogenetic relationships between species into the kernel consistently improves its accuracy. In order to get a better insight of the relative positions of the phylogenetic profiles in the feature space, we also perform a kernel principal component analysis that shows that the tree kernel is more sensitive to evolutionary patterns than a kernel that does not incorporate any information about the phylogeny.

METHODS AND ALGORITHMS

Bayesian tree models for phylogenetic profiles

The transmission of genes during evolution is modeled by a simple Bayesian tree model, i.e., a graphical model (Lauritzen, 1996) based on a rooted tree. A rooted tree is a connected acyclic directed graph in which each node is connected by a unique path from a designated node called the root and denoted λ. We denote by $\mathcal{T}$ the set of nodes of the tree, $\mathcal{L} \subset \mathcal{T}$ the set of leaves (i.e., the nodes with no child), and $\mathcal{T}^* = \mathcal{T} \backslash \{\lambda\}$ the set of nodes without the root. For any node $u \in \mathcal{T}^*$ we write $p(u) \in \mathcal{T}$ to represent the parent node of u, and for any node $u \in \mathcal{T}$, $c(u) \subset \mathcal{T}$ and $l(u) \subset \mathcal{L}$ to represent respectively the set of children of the node u, and the set of leaves descendants of u (Figure 1).

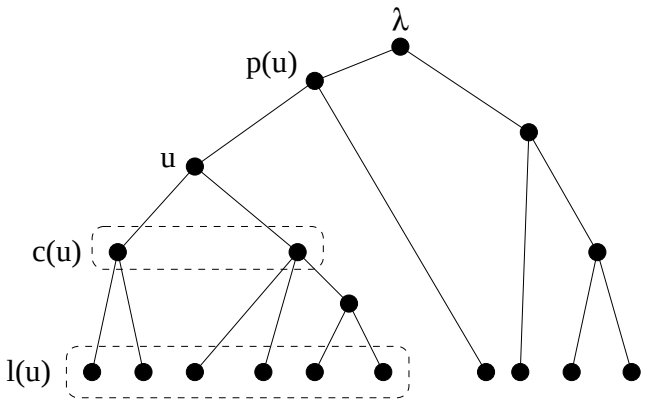

Fig. 1. A tree and the notations used in this article

In this paper the tree is meant to be a phylogenetic tree: each leaf of the tree represents one organism currently living, and each internal node represents an ancestor of the current organisms (see for example Figure 3).

The tree is used to define a joint probability distribution P for a set of random variables $\{X_u, u \in \mathcal{T}\}$ indexed by the nodes of the tree and with values in a finite alphabet $\mathcal{A}$. We will focus on binary variables, i.e., $\mathcal{A} = \{0, 1\}$, but this work could be generalized to any finite alphabet. The variable X_u attached to a node $u \in \mathcal{T}$ is supposed to be 1 whenever the corresponding organism has a homologue of the gene under study, 0 otherwise.

For convenience we will use the following notations in the sequel. For any set of nodes $\mathcal{S} \subset \mathcal{T}$, $X_\mathcal{S} = \{X_u, u \in \mathcal{S}\}$ is the subset of variables indexed by $\mathcal{S}$, and $x_\mathcal{S} \in \mathcal{A}^\mathcal{S}$ a particular set of values that $X_\mathcal{S}$ can take. For any two sets of nodes $\mathcal{S} \subset \mathcal{T}$ and $\mathcal{S}' \subset \mathcal{T}$, and any values $x_\mathcal{S} \in \mathcal{A}^\mathcal{S}$ and $y_{\mathcal{S}'} \in \mathcal{A}^{\mathcal{S}'}$, the notation $p(x_\mathcal{S} \,|\, y_{\mathcal{S}'})$ represents the following probability:

$$p(x_\mathcal{S} \,|\, y_{\mathcal{S}'}) \stackrel{def}{=} P\left\{\forall u \in \mathcal{S}, X_u = x_u \,|\, \forall u' \in \mathcal{S}', X_{u'} = y_{u'}\right\}.$$

With these notations a phylogenetic profile can be defined as follows:

DEFINITION 1. *A phylogenetic profile is a set of bits assigned to the leaves of the tree, i.e., $x_\mathcal{L} \in \mathcal{A}^\mathcal{L}$.*

To model the evolution of genomes along the branches of the evolutionary tree an initial distribution p_λ on $\mathcal{A}$

is assigned to the root of the tree, and a conditional probability distribution $p_u(i \mid j)$ for $(i,j) \in \mathcal{A}^2$ is assigned to each node $u \in \mathcal{T}^*$. The tree together with the set of distributions $\{p_u, u \in \mathcal{T}\}$ defines a joint probability distribution for $X_\mathcal{T}$ as follows:

$$\forall x_\mathcal{T} \in \mathcal{A}^\mathcal{T}, \quad p(x_\mathcal{T}) = p_\lambda(x_\lambda) \prod_{u \in \mathcal{T}^*} p_u(x_u \mid x_{p(u)}).$$

Kernels

Our approach to analyze phylogenetic profiles is to map them to a high-dimensional space whose dimensions correspond to evolutionarily relevant features. Let us denote by $\Phi : \mathcal{A}^\mathcal{L} \to \mathbb{R}^D$ such a mapping, where D is the dimension of the feature space. Then the image of a profile $x_\mathcal{L} \in \mathcal{A}^\mathcal{L}$ can be written as $\Phi(x_\mathcal{L}) = (\Phi_1(x_\mathcal{L}), \ldots, \Phi_D(x_\mathcal{L}))$, and each coordinate $\Phi_i(x_\mathcal{L})$ for $i = 1, \ldots, D$ is called a *feature* of the profile $x_\mathcal{L}$.

In the sequel we consider a large set of evolutionarily relevant features, leading to a high-dimensional feature space. In such a case, rather than computing explicitly the image $\Phi(x_\mathcal{L})$ of each profile $x_\mathcal{L} \in \mathcal{A}^\mathcal{L}$, it is more interesting to be able to compute efficiently the inner product between the images of any two profiles $(x_\mathcal{L}, y_\mathcal{L}) \in \mathcal{A}^\mathcal{L} \times \mathcal{A}^\mathcal{L}$ in the feature space, i.e.,

$$K(x_\mathcal{L}, y_\mathcal{L}) \stackrel{def}{=} \sum_{i=1}^{D} \Phi_i(x_\mathcal{L})\Phi_i(y_\mathcal{L}). \tag{1}$$

The resulting function $K(.,.)$ is called a *kernel* (Schölkopf & Smola, 2001) and can be used to perform implicitly various data analysis algorithms in the feature space.

The simplest kernel for phylogenetic profiles is the inner product when strings of bits are considered as real-valued vectors. In other words the naive kernel is defined as follows:

$$K_{naive}(x_\mathcal{L}, y_\mathcal{L}) = \sum_{u \in \mathcal{L}} x_u y_u. \tag{2}$$

The resulting squared Euclidean distance between two profiles is simply the number of organism where they differ, hence the naive kernel is the natural way to represent profiles geometrically when their similarity is defined in such a way.

The tree kernel

The naive kernel (2) does not incorporate any knowledge about the nature of phylogenetic profiles, in particular the phylogenetic relationships among species. In order to develop a more biologically relevant measure of similarity it would be natural not to compare just the profiles themselves, but also their global patterns of inheritance if the genomes of all ancestors were available. In that case we could check, for any two genes, when in evolution they were transmitted together or not, and get a more pertinent view of how similar their phylogenetic histories are than by just comparing their phylogenetic profiles organism per organism.

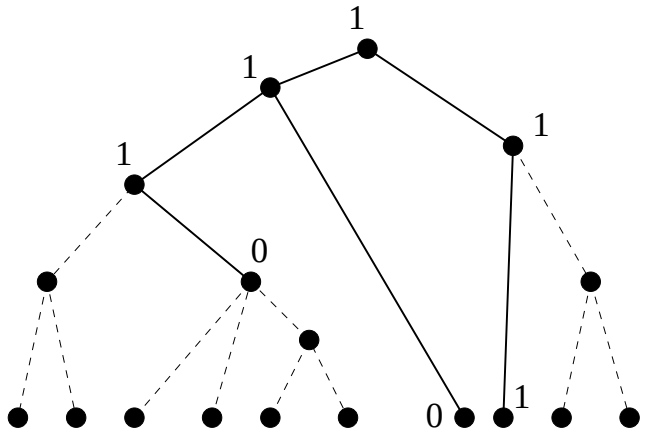

Fig. 2. An evolution pattern is a subtree (represented with thick lines) whose nodes are assigned values in $\{0, 1\}$.

In order to create a distance for phylogenetic profiles that reflects the similarity between evolutions we propose to map any profile to a feature space where each feature corresponds to a particular pattern of evolution defined as follows:

DEFINITION 2. *An* evolution pattern *is a pair* $(\mathcal{S}, z_\mathcal{S})$ *where* $\mathcal{S} \subset \mathcal{T}$ *is a* subtree *of the complete Bayesian tree (i.e., a connected subgraph which contains the root), and* $z_\mathcal{S} \in \mathcal{A}^\mathcal{S}$ *is a set of values attached to the nodes of* $\mathcal{S}$ *(see an example on Figure 2)*

For each evolution pattern $(\mathcal{S}, z_\mathcal{S})$ we define a feature $\Phi_{\mathcal{S},z_\mathcal{S}} : \mathcal{A}^\mathcal{L} \to \mathbb{R}$ which ideally would be 1 for profiles whose histories match the pattern, and 0 otherwise. The history of the each profile (i.e., the assignment of a 0 or a 1 to each ancestor organism in the phylogenetic tree) is of course not known, but the probabilistic model of evolution described previously can help approximate these ideal features. One way to proceed is to replace the ideal $0/1$ assignment by the probability that the profile $x_\mathcal{L}$ be observed conditionally to the pattern $z_\mathcal{S}$. We therefore define the feature $\Phi_{\mathcal{S},z_\mathcal{S}}$ as follows:

$$\forall x_\mathcal{L} \in \mathcal{A}^\mathcal{L}, \quad \Phi_{\mathcal{S},z_\mathcal{S}}(x_\mathcal{L}) \stackrel{def}{=} p(x_\mathcal{L} \mid z_\mathcal{S}). \tag{3}$$

Of course there is a priori no reason to focus on only one particular subtree $\mathcal{S}$ and one particular pattern $z_\mathcal{S}$. To the contrary it is more interesting to look at as many

features as possible in order to characterize each profile. If we denote by $\mathcal{C}(\mathcal{T})$ the set of all subtrees of $\mathcal{T}$ we therefore define the set of features as the features $\Phi_{\mathcal{S},z_\mathcal{S}}$ corresponding to all possible subtrees $\mathcal{S} \in \mathcal{C}(\mathcal{T})$ and patterns $z_\mathcal{S} \in \mathcal{A}^\mathcal{S}$.

Combining (1) and (3) we see that the resulting kernel is the following, for any two profiles $(x_\mathcal{L}, y_\mathcal{L}) \in \mathcal{A}^\mathcal{L} \times \mathcal{A}^\mathcal{L}$:

$$K(x_\mathcal{L}, y_\mathcal{L}) = \sum_{\mathcal{S} \in \mathcal{C}(\mathcal{T})} \sum_{z_\mathcal{S} \in \mathcal{A}^\mathcal{S}} p(x_\mathcal{L} \mid z_\mathcal{S}) p(y_\mathcal{L} \mid z_\mathcal{S}).$$

In this expression all features are given the same importance in the sense that the kernel is obtained by adding the contributions of all features. However some features might be considered more "important" than others, because some evolution patterns are more likely to have effectively occurred than others. In other words it is likely that most features corresponding to patterns $(\mathcal{S}, z_\mathcal{S})$ which have a very low probability of appearance are irrelevant and only add noise to the feature space. One solution to reduce this effect is to weight the contribution of each feature in the kernel by the probability of the corresponding pattern, which results in the following kernel:

DEFINITION 3. *The* tree kernel *is the kernel defined for any phylogenetic profiles $(x_\mathcal{L}, y_\mathcal{L}) \in \mathcal{A}^\mathcal{S} \times \mathcal{A}^\mathcal{S}$ by:*

$$K_{tree}(x_\mathcal{L}, y_\mathcal{L}) \stackrel{def}{=} \sum_{\mathcal{S} \in \mathcal{C}(\mathcal{T})} \sum_{z_\mathcal{S} \in \mathcal{A}^\mathcal{S}} p(z_\mathcal{S}) p(x_\mathcal{L} \mid z_\mathcal{S}) p(y_\mathcal{L} \mid z_\mathcal{S}). \quad (4)$$

In this form the tree kernel appears to be related to convolution kernels introduced by (Watkins, 1999) and (Haussler, 1999), and to generalize the class of kernels developed in (Vert, 2002) to graphical models with latent variables.

Computation of the tree kernel

Whereas the naive kernel between two profiles can easily be computed from (2), Equation (4) should not be directly implemented to compute the tree kernel because of the large number of terms to add (the number of subtrees of a tree increases exponentially with the number of nodes, and for each subtree $\mathcal{S}$ there are $2^{|\mathcal{S}|}$ features to compute). In this section we present an algorithm to compute the tree kernel with a complexity linear with respect to the size of the phylogenetic profiles.

Given a Bayesian tree model, the algorithm first computes the following functions η, indexed by the nodes of the tree, for each phylogenetic profile $x_\mathcal{L}$ using a post-order traversal of the tree:

- If u is a leaf:
$$\forall i \in \mathcal{A}, \quad \eta_i(x_\mathcal{L}, u) = p_u(x_u \mid i).$$
- If u is an internal node different from the root:
$$\forall i \in \mathcal{A}, \quad \eta_i(x_\mathcal{L}, u) = \sum_{j \in \mathcal{A}} p_u(j \mid i) \prod_{u' \in c(u)} \eta_j(x_\mathcal{L}, u').$$
- If $u = \lambda$:
$$\eta(x_\mathcal{L}, \lambda) = \sum_{j \in \mathcal{A}} p_\lambda(j) \prod_{u' \in c(\lambda)} \eta_j(x_\mathcal{L}, u').$$

In a second step, in order to compute the kernel between any two profiles $x_\mathcal{L}$ and $y_\mathcal{L}$, a second functional ζ indexed by the nodes of the tree is computed using a post-order traversal of the tree:

- If u is a leaf:
$$\zeta_i(s) = \begin{cases} 1 & \text{if } x_u = y_u = i, \\ 0 & \text{otherwise .} \end{cases}$$
- If u is not a leaf:
$$\zeta_i(u) = \prod_{u' \in c(u)} \left\{ \sum_{j \in \mathcal{A}} p_{u'}(j \mid i) \zeta_j(u') + \eta_i(x_\mathcal{L}, u') \eta_i(y_\mathcal{L}, u') \right\}.$$

Finally the kernel value is obtained as follows:

$$K_{tree}(x_\mathcal{L}, x_\mathcal{L}) = \sum_{j \in \mathcal{A}} p_\lambda(j) \zeta_j(\lambda) + \eta(x_\mathcal{L}, \lambda) \eta(y_\mathcal{L}, \lambda).$$

This algorithm is based on post-order traversals of the tree and has therefore a complexity linear with respect to the size of the tree (and of the phylogenetic profiles). The reason why it returns the correct kernel value is a consequence of the following proposition which can be proved by induction by working up the tree from the leaves (due to space limitation the complete proof is omitted in this paper):

PROPOSITION 1. *The functions η and ζ computed by the previous algorithm satisfy the following equations, for any two phylogenetic profiles $(x_\mathcal{L}, y_\mathcal{L}) \in \mathcal{A}^\mathcal{L} \times \mathcal{A}^\mathcal{L}$:*

- *For any node $u \in \mathcal{T}^*$ and $i \in \mathcal{A}$:*
$$\eta_i(x_\mathcal{L}, u) = P(X_{l(u)} = x_{l(u)} \mid X_{p(u)} = i),$$
and for the root:
$$\eta(x_\mathcal{L}, \lambda) = P(X_\mathcal{L} = x_\mathcal{L}).$$
- *For any node $u \in \mathcal{T}$ and $i \in \mathcal{A}$:*
$$\zeta_i(u) = \sum_{\mathcal{S} \in \Delta(u)} \sum_{z_\mathcal{S} \in \mathcal{A}^\mathcal{S}} P(X_\mathcal{S} = z_\mathcal{S} \mid X_u = i) \times p(x_{l(u)} \mid z_\mathcal{S}).p(y_{l(u)} \mid z_\mathcal{S}),$$
where $\Delta(u)$ is the set of non-empty subtrees rooted in u.

Support Vector Machines

Support Vector Machines (SVMs) are a class of supervised learning algorithms first introduced by Vapnik and coworkers (Boser *et al.*, 1992; Vapnik, 1998). Given a kernel $K(.,.)$ and a set of training examples (phylogenetic profiles in our case) labeled as positive or negative, SVMs learn a linear decision boundary in the feature space defined by the kernel in order to discriminate between positive and negative examples. Any new unlabeled example is then predicted to be positive or negative depending on the position of its image in the feature space relatively to the linear boundary.

Apart from having strong mathematical foundations in statistical learning theory SVMs have been shown to perform very well in many real-world applications. They have found recently several applications in computational biology, including but not limited to protein function prediction (Jaakkola *et al.*, 2000), genes or tissues classification from microarray data (Brown *et al.*, 2000; Furey *et al.*, 2000) or translation initiation site recognition in DNA (Zien *et al.*, 2000). More details about the theory and practice of SVMs can be found in several books (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2001).

In this paper we use SVMs to infer the function of a gene from its phylogenetic profile. We test the ability of the tree kernel to incorporate biologically relevant information by comparing the performances of SVMs using either the tree kernel (4) or the naive kernel (2).

We used the software SVMLight v.3.50 (Joachims, 1999), a public implementation[†] of SVMs that offers the possibility to implement user-defined kernels.

Once a kernel is fixed only two parameters have to be specified before running the algorithm, namely the constant C which penalizes more or less misclassification errors, and the relative weights of positive and negative examples. To ensure a comparison as fair as possible between the naive and the tree kernels we used in all experiments the default value proposed by the software for the C parameter (namely $C = m/\sum_{i=1}^{m} K(x^{(i)}, x^{(i)})$ for a training set $(x^{(1)}, \ldots, x^{(m)})$), and the relative weight parameter was adjusted to ensure that the total weight of positive examples be equal to the total weight of negative examples.

Kernel PCA

In order to get a glimpse of the relative positions of the phylogenetic profiles in the feature space defined by a kernel, it is possible to perform a principal component analysis (PCA) of their images in the feature space, and to look at their projections on the first few principal components. The PCA in the feature space, which is called a *kernel PCA*, can be performed entirely without computing the images of the profiles, but instead by computing the kernel between all pairs of profiles and then performing an eigenvalue decomposition of the resulting Gram matrix centered in the feature space (see details in (Schölkopf *et al.*, 1999)). We computed the Gram matrix with the C implementation of the kernel used by SVMLight, and performed the kernel-PCA using the free and public mathematical software Octave[‡].

DATA

We performed experiments with the genes of the budding yeast *Saccharomyces cerevisiae*. Using the same data as (Pavlidis *et al.*, 2001) the phylogenetic profiles of 2465 yeast genes selected for their accurate functional classifications were generated by computing the E-value reported by BLAST version 2.0 (Altschul *et al.*, 1997) in a search against each of the 24 genomes listed in Figure 3 and collected from The Institute for Genomic Research website[§] or from the Sanger Center website[¶]. Each bit in the phylogenetic tree was set to 0 or 1 if the E-value for the corresponding organism was larger or smaller than 1 respectively.

The phylogenetic tree used to compute the tree kernel, shown in Figure 3, is similar to the one used in (Liberles *et al.*, 2002). The parameters of the Bayesian model were arbitrarily set as follows: $p_\lambda(1) = 0.9$ and for every node u, $p_u(1 \mid 1) = p_u(0 \mid 0) = 0.9$. In other words this probabilistic model favors the transmission of genes along evolution but enables the appearance or disappearance of a gene along any branch with probability 0.1. In spite of its crudeness this simplistic model of gene transmission is sufficient to capture information about the phylogenetic relationships among different organisms.

The function prediction experiments were based on the functional categories in the Munich Information Center for Protein Sequences Comprehensive Yeast Genome Databases (CYGD)[‖], whose functional catalog contains several hundreds of functional classes. We extracted all classes with at least 10 genes, resulting in 133 classes.

RESULTS

Function prediction

For each functional category we trained a SVM to predict whether a gene should be assigned to it or not based on its phylogenetic profile only. Using 3-fold cross validation repeated 50 times for each class, we compared the performances of a SVM using the naive kernel with a SVM using the tree kernel through there receiver operating characteristic (ROC) curves, i.e., the plot of true positives

[†] http://svmlight.joachims.org

[‡] http://www.octave.org

[§] http://www.tigr.org/tdb

[¶] http://www.sanger.au.uk

[‖] http://www.mips.biochem.mpg.de/proj/yeast

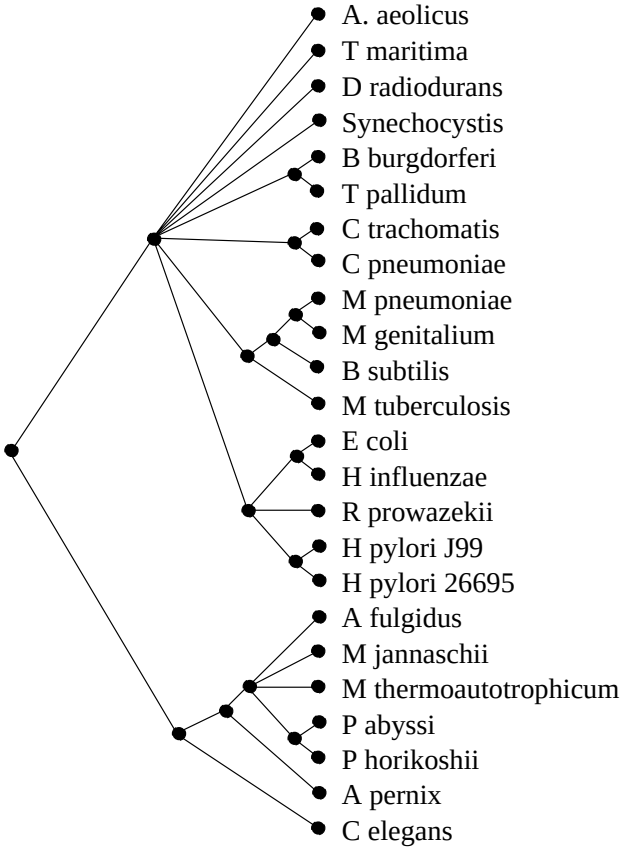

Fig. 3. The phylogenetic tree for 24 completely sequenced organisms (the lengths of the branches are arbitrary).

as a function of false positives. When predicting the category of a gene, a SVM outputs a score (related to the distance between the example and the linear boundary in the feature space) so we could compute the ROC curves by varying a threshold and classifying the profiles depending on their scores compared to the threshold.

As each functional class contains a small number of genes each learning problem is very unbalanced (there are few positive examples but many negative ones). We handled this issue by giving more weight to the positive examples in the SVM learning (such that the total weight of positive examples be equal to the total weight of negative examples). Moreover this implies that only a small percentage of false positives can be tolerated in real-world applications (such as function predictions), so we measured the ROC_{50} score for each SVM, i.e., the area under the ROC curve up to the first 50 false positives (Gribskov & Robinson, 1996).

Table 1 shows the functional categories with the highest ROC_{50} scores obtained by a SVM using the naive kernel, together with the ROC_{50} scores of the SVM using the tree kernel. It turns out that the tree kernel performs better than the naive kernel on most of the classes, and sometimes improves considerably the ROC_{50} score (for such classes as Glycolysis and gluconeogenesis, or tRNA modification). Classes that are better predicted by the naive kernel (amino-acid metabolism, transport facilitation, organization of plasma membrane and metabolism) tend to be larger and more general classes than the others.

We plotted on Figures 4 the ROC curves up to 50 false positives corresponding to the four classes with the highest naive ROC_{50} score, in order to further study the differences in performance. These plots show that in several cases the tree kernel significantly outperforms the naive kernel when a very small number of false positives is allowed (for example, if 3 false positives are accepted, the tree kernel retrieves on average more than 65% of all amino-acid transporters, while the naive kernel retrieves less than 30%).

Kernel PCA analysis of four functional families

In order to better understand the difference between the tree kernel and the naive kernel we performed a kernel PCA analysis of the phylogenetic profiles belonging to the first four categories listed in Table 1. Figure 5 shows the projections of these profiles on the first four principal components (PC) in the feature space defined by the tree and the naive kernel respectively.

These plots show a clear difference between the geometries generated by both kernels. In the feature space defined by the naive kernel, the profiles are more scattered in all dimensions that in the feature space defined by the tree kernel. With the naive kernel, the profiles seem to form a convex set, but with the tree kernel, each PC seems to characterize particularly well a few profiles, which happen to belong to the same functional category (amino-acid transporters for the first component, or ABC transporters for the third one). The resulting shape looks like a small cluster at the center with many branches starting from the center toward various orthogonal directions, each direction being particularly relevant to one functional class.

These observations show that the tree kernel is particularly sensitive to biologically relevant variations in the profiles. As an example, the 6 ABC transporters (∇) characterized by a small value on the third PC with the tree kernel are also clustered together by the naive kernel (they have for instance a small value on the 4-th PC). However they are aligned along one particular PC with the tree kernel, which indicates that the variations among them are very typical of a neat combination of evolution patterns detected by the tree kernel. To the contrary their variations are spread among several PCs by the naive kernel.

Table 1. ROC_{50} scores for the prediction of 16 functional categories by a SVM using either a naive kernel or a tree kernel (the last column expresses the relative increase or decrease when the naive kernel is replaced by the tree kernel).

Functional class	Naive kernel	Tree kernel	Difference
Amino-acid transporters	0.74	0.81	**+ 9%**
Fermentation	0.68	0.73	**+ 7%**
ABC transporters	0.64	0.87	**+ 36%**
C-compound, carbohydrate transport	0.59	0.68	**+ 15%**
Amino-acid biosynthesis	0.37	0.46	**+ 24%**
Amino-acid metabolism	0.35	0.32	*- 9%*
Tricarboxylic-acid pathway	0.33	0.48	**+ 45%**
Transport Facilitation	0.33	0.28	*- 15%*
Organization of plasma membrane	0.31	0.30	*- 3%*
Amino-acid degradation (catabolism)	0.30	0.52	**+ 73%**
Lipid and fatty-acid transport	0.29	0.52	**+ 79%**
Homeostasis of other cations	0.26	0.33	**+ 27%**
Glycolysis and gluconeogenesis	0.25	0.66	**+ 164%**
Metabolism	0.24	0.20	*- 17%*
Cellular import	0.20	0.27	**+ 35%**
tRNA modification	0.15	0.32	**+ 113%**

DISCUSSION AND CONCLUSION

This paper presents an attempt to define a measure of similarity between phylogenetic profiles that incorporates knowledge about the phylogenetic relationships between species. More than just a measure of similarity, we were able to build a whole Euclidean geometry for phylogenetic profiles which is tractable in spite of its large dimension thanks to the availability of an efficient algorithm to compute the inner product in that space. Defining such a Euclidean geometry is more appealing than just defining a measure of similarity, because many data mining methods can be readily applied in such spaces.

We demonstrated through a series of function prediction experiments and kernel PCA analysis that the geometry defined by the tree kernel is more sensitive to biologically relevant patterns in the profiles than the geometry generated when no evolutionary information is used. As a result a SVM using the tree kernel performed better on average than a SVM using a naive kernel.

Because of the efficiency of kernel methods such as SVMs for classification, developing kernels for particular objects has received much attention recently. As an example, a number of kernels for strings have been developed and applied to the classification of biological sequences or text written in natural language, including the Fisher kernel (Jaakkola *et al.*, 2000), convolution kernels (Haussler, 1999; Watkins, 1999), string kernels (Lodhi *et al.*, 2000), interpolated kernels (Vert, 2002) or spectrum kernels (Leslie *et al.*, 2002).

More than a string kernel, the tree kernel introduced in this paper should be regarded as a *graph kernel*, as it is computed for sets of strings indexed by nodes of a tree.The algorithm can easily be generalized to handle any tree graphical model with or without latent variables, in order to define a kernel on the set of realizations of these graphical models. As such models are widely used not only in computational biology but also in many other fields, the potential applications of this kernel in combination with the increasing library of kernel methods for data mining are numerous.

ACKNOWLEDGEMENTS

I thank Minoru Kanehisa for helpful discussions and for supporting this research through a Research for the Future Program of the Japan Society for the Promotion of Science.

REFERENCES

Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. (1997). Gapped blast and psi-blast: A new generation of protein database search programs. *Nucleic Acids Research*, **25**, 3389–3402.

Ben-Hur, A., Horn, D., Siegelmann, H. T. & Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, **2**, 125–137.

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

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Manuel Ares, J. & Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA*, **97**, 262–267.

Cristianini, N. & Shawe-Taylor, J. (2000). An introduction

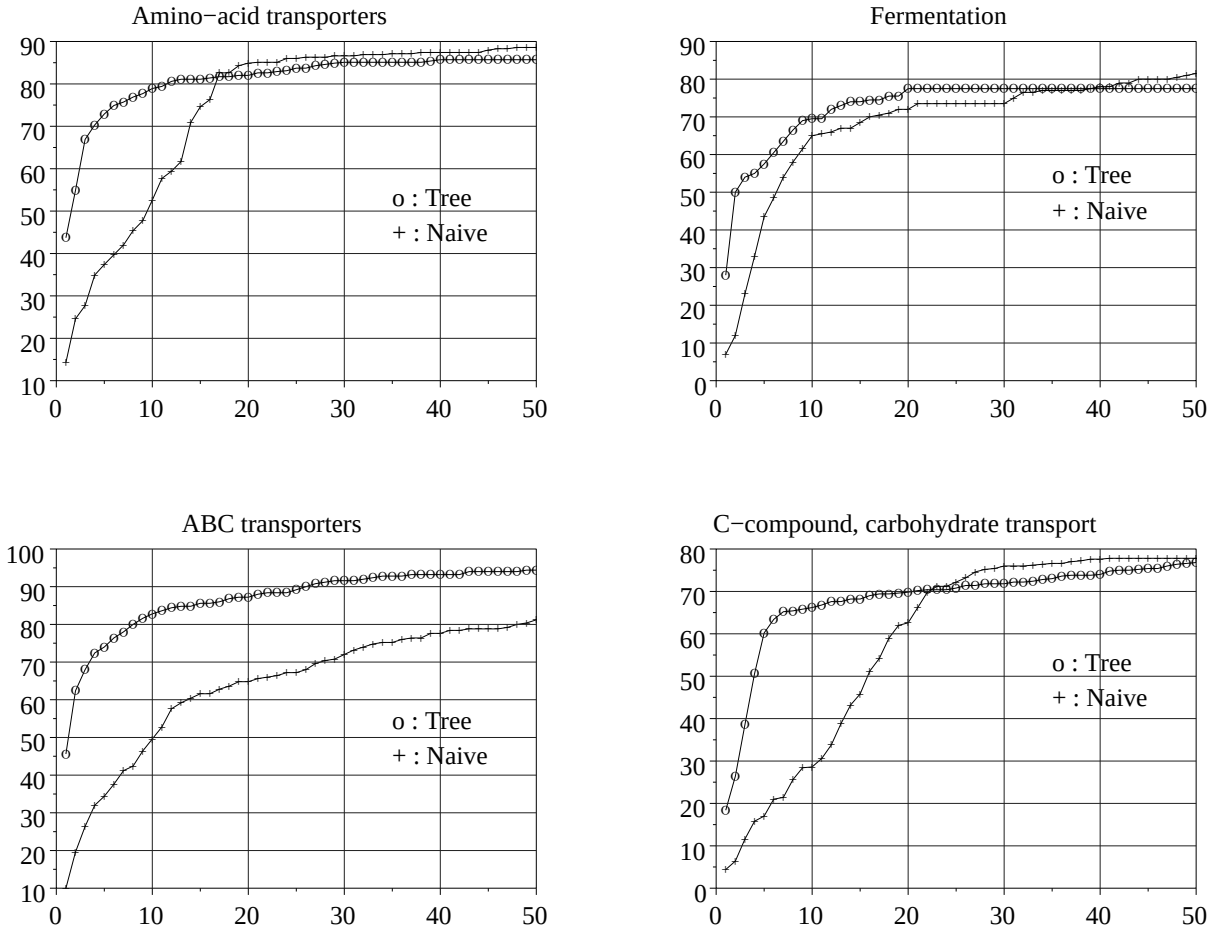

Fig. 4. ROC curves for the prediction of four functional classes from the phylogenetic profiles of the yeast genes with a Support Vector Machine using the tree kernel (labeled "tree") or the naive kernel (labelled "naive"). The number of false positives is on the X-axis, the percentage of true positives retrieved (i.e., the recall) on the Y-axis. Curves are averaged over a number of 3-fold cross-validation experiments.

to Support Vector Machines and other kernel-based learning methods. Cambridge University Press.

Furey, T. S., Duffy, N., Cristianini, N., Bednarski, D., Schummer, M. & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**, 906–914.

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

Haussler, D. (1999). Convolution kernels on discrete structures. Technical report, UC Santa Crux.

Jaakkola, T., Diekhans, M. & Haussler, D. (2000). A discrimitive framework for detecting remote protein homologies. *Journal of Computational Biology*, **7**, 95–114.

Joachims, T. (1999). Making large-scale svm learning practical. In Schölkopf, B., Burges, C. & Smola, A., (eds.) *Advances in Kernel Methods - Support Vector Learning*. MIT Press, pp. 169–184.

Lauritzen, S. (1996). Graphical Models. Oxford.

Leslie, C., Eskin, E. & Noble, W. S. (2002). The spectrum kernel: a string kernel for svm protein classification. In Altman, R. B., Dunker, A. K., Hunter, L., Lauerdale, K. & Klein, T. E., (eds.) *Proceedings of the Pacific Symposium on Biocomputing 2002.* World Scientific, pp. 564–575.

Liberles, D. A., Thoren, A., von Heijne, G. & Elofsson, A. (2002). The use of phylogenetic profiles for gene predictions. *Current Genomics*. In press.

Lodhi, H., Shawe-Taylor, J., Cristianini, N. & Watkins, C. J. C. H. (2000). Text classification using string kernels. In *NIPS*. pp. 563–569.

Marcotte, E. M., Pellegrini, M., Thompson, M. J., Yeates, T. O. & Eisenberg, D. (1999). A combined algorithm for genome-wide prediction of protein function. *Nature*, **402**, 83–86.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B. & Müller, K. (1999). Fisher discriminant analysis with kernels. In Hu, Y.-H., Larsen,

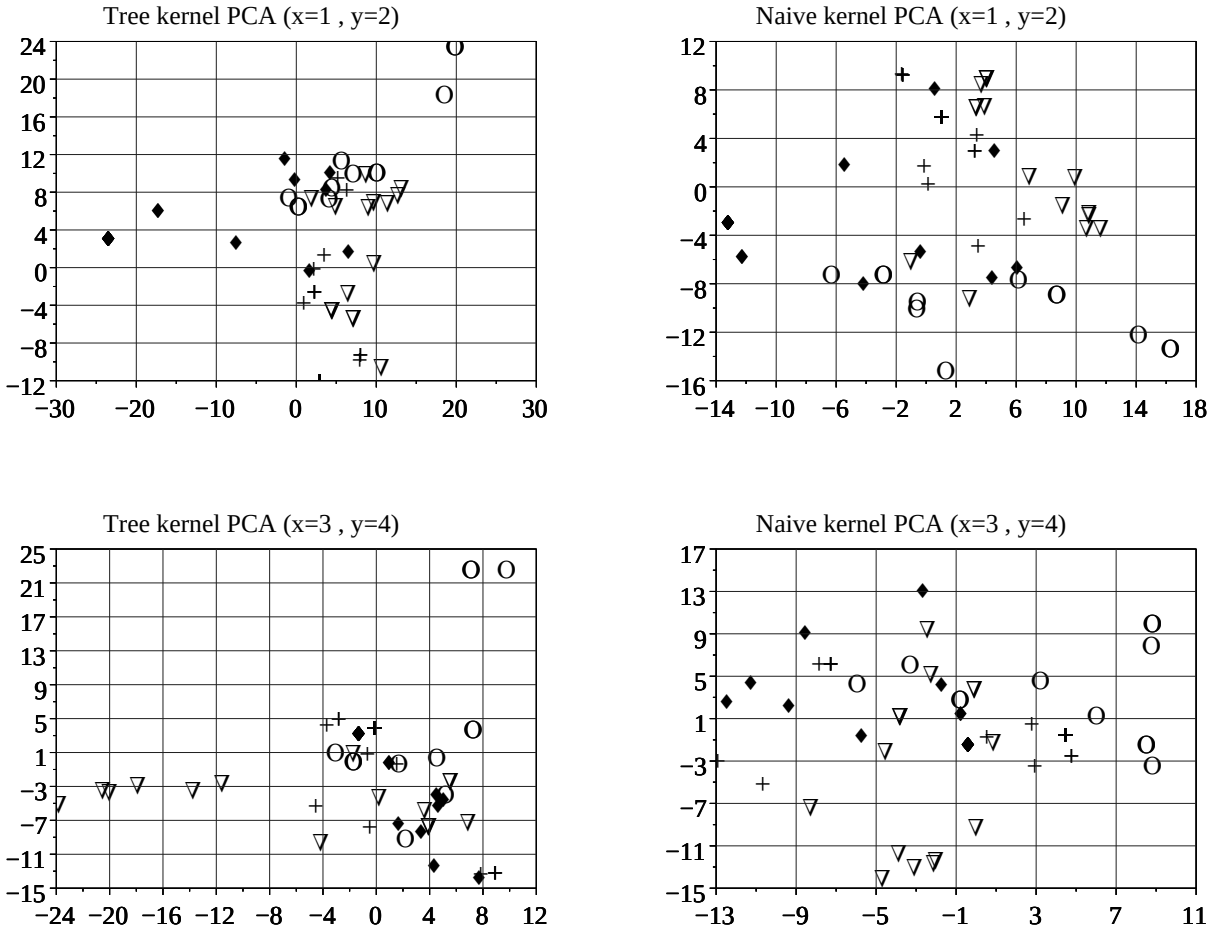

Fig. 5. Kernel PCA analysis of the phylogenetic profiles of the genes belonging to four functional classes, with either the tree kernel (on the left) or the naive kernel (on the right). In each case the upper plot represents the first PC versus the second, and the lower plot represents the third against the fourth. Each symbol represents one functional class: amino-acid transporters (⋄), fermentation (○), ABC transporters (▽) and C-compound, carbohydrate transport (+)

J., Wilson, E. & Douglas, S., (eds.) *Neural Networks for Signal Processing IX*. IEEE, pp. 41–48.

Pavlidis, P., Weston, J., Cai, J. & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. In *Proceedings of the Fifth Annual International Conference on Computational Biology*. pp. 249–255.

Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D. & Yeates, T. O. (1999). Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA*, **96**, 4285–4288.

Schölkopf, B. & Smola, A. J. (2001). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press.

Schölkopf, B., Smola, A. J. & Müller, K.-R. (1999). Kernel principal component analysis. In Schölkopf, B., Burges, C. & Smola, A., (eds.) *Advances in Kernel Methods - Support Vector Learning*. MIT Press, pp. 327–352.

Vapnik, V. N. (1998). Statistical Learning Theory. Wiley, New-York.

Vert, J.-P. (2002). Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In Altman, R. B., Dunker, A. K., Hunter, L., Lauerdale, K. & Klein, T. E., (eds.) *Proceedings of the Pacific Symposium on Biocomputing 2002*. World Scientific, pp. 649–660.

Watkins, C. (1999). Dynamic alignment kernels. Technical report, UL Royal Holloway.

Zien, A., Ratsch, G., Mika, S., Scholkopf, B., Lengauer, T. & Muller, K.-R. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, **16**, 799–807.