Kernelizing the output of tree-based methods

Pierre Geurts, Louis Wehenkel, Florence d'Alché-Buc

Kernelizing the Output of Tree-Based Methods

Pierre Geurts[1,2] PGEURTS@IBISC.UNIV-EVRY.FR,P.GEURTS@ULG.AC.BE
Louis Wehenkel[2] L.WEHENKEL@ULG.AC.BE
Florence d'Alché-Buc[1] DALCHE@IBISC.UNIV-EVRY.FR
[1]IBISC FRE CNRS 2873 & Epigenomics Project, GENOPOLE, 523, Places des Terrasses, 91 Evry, France, [2]Department of EECS & CBIG/GIGA Centre of Genoproteomics, University of Liège, Sart Tilman, B28, B-4000, Liège, Belgium

Abstract

We extend tree-based methods to the prediction of structured outputs using a kernelization of the algorithm that allows one to grow trees as soon as a kernel can be defined on the output space. The resulting algorithm, called output kernel trees (OK3), generalizes classification and regression trees as well as tree-based ensemble methods in a principled way. It inherits several features of these methods such as interpretability, robustness to irrelevant variables, and input scalability. When only the Gram matrix over the outputs of the learning sample is given, it learns the output kernel as a function of inputs. We show that the proposed algorithm works well on an image reconstruction task and on a biological network inference problem.

1. Introduction

Extending statistical learning to structured output space is now surging as a new theoretical and practical challenge. In computational biology, natural language processing or more generally in pattern recognition, structured data such as strings, trees, graphs abound and require new tools to be mined relevantly. An elegant way to handle structured data is to make use of kernel methods that encapsulate the knowledge about the structure of a space of interest into the definition of a kernel. Already successful for structured inputs, original kernel-based methods have recently been proposed to address the structured output problem (Weston et al., 2002; Cortes et al., 2005; Tsochantaridis et al., 2005; Taskar et al., 2005; Weston et al., 2005).

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

In this paper, we start from a different family of models, namely tree-based models.Tree-based methods have proved to be useful when interpretability is required or when features need to be selected. If enhanced by ensemble methods, they present high performances and are considered as a general and powerful tool as soon as the attribute vector representation is appropriate. In this context, we propose a straightforward extension of multiple output regression trees using the kernel trick, noticing the fact that the variance-based score function used for evaluating splits requires only scalar product computations. Provided that we have defined a kernel on the output space, it thus becomes possible to build a tree that maps subregions of input space defined by hyperplanes parallel to axis into hyperspheres defined in the feature space corresponding to the output kernel. The algorithm can also be used from a Gram matrix only to learn an approximation of the underlying kernel as a function of the inputs. This new extension of trees which we called Output Kernel Trees (OK3) can also benefit from ensemble methods because of the linearity of the ensemble combination operator.

Section 2 presents the kernelization of tree-based methods and its properties. Section 3 describes and comments numerical experiments on two problems: a pattern completion task (Weston et al., 2002) and a graph inference problem (Vert & Yamanishi, 2004). Section 4 provides some perspectives.

2. Output kernel trees

The general problem of supervised learning may be formulated as follows: from a learning sample $LS = \{(x_i, y_i)|i = 1, \ldots, N_{LS}\}$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, find a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that minimizes the expectation of some loss function over the joint distribution of input/output pairs:

$$E_{x,y}\{\ell(f(x), y)\}. \quad (1)$$

2.1. Standard regression trees

Standard regression trees (Breiman et al., 1984) propose a solution to this problem when the output space is the real axis, $\mathcal{Y} = I\!R$, and the loss functions ℓ is the square error, $\ell(f(x), y) = (f(x) - y)^2$.

The general idea of regression trees is to recursively split the learning sample with binary tests based on the input variables, trying at each split to reduce as much as possible the variance of the output in the left and right subsamples of learning cases corresponding to that split. The splitting of a node is stopped when the output is constant in this node or some stopping criterion is met (e.g., the size of the local subsample goes below some threshold or the split is deemed non significant according to some statistical test).

More precisely, a score measure is defined to evaluate and select splits, which is written:

$$\text{Score}(T, S) = \text{var}\{y|S\} - \frac{N_l}{N}\text{var}\{y|S_l\} - \frac{N_r}{N}\text{var}\{y|S_r\},$$

where T is the split to evaluate, S is the local learning sample of size N at the node to split, S_l and S_r are its left and right successors of size N_l and N_r respectively, and $\text{var}\{y|S\}$ denotes the empirical variance of the output y in the subset S computed as:

$$\text{var}\{y|S\} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i)^2. \qquad (2)$$

Once the tree is grown, each leaf L is labeled with a prediction $\hat{y}_L$ computed as:

$$\hat{y}_L = \frac{1}{N_L}\sum_{i=1}^{N_L} y_i,$$

where N_L is the number of learning cases that reach this leaf. These predictions minimize the mean square error when the tree is used to predict the outputs of learning sample cases.

This algorithm can be extended in a straightfoward way to the case of multiple numerical outputs, i.e. $\mathcal{Y} = I\!R^n$, with the loss function $\ell(f(x), y) = ||f(x) - y||^2$, where $||\cdot||$ denotes the Euclidean norm in $I\!R^n$. In this case, the empirical variance computed by (2) and the predictions at leaf nodes are simply replaced respectively by:

$$\text{var}\{y|S\} = \frac{1}{N}\sum_{i=1}^{N}||y_i - \frac{1}{N}\sum_{i=1}^{N} y_i||^2 \qquad (3)$$

$$\hat{y}_L = \frac{1}{N_L}\sum_{i=1}^{N_L} y_i. \qquad (4)$$

The latter is the center of mass in the leaf and the former is the average in S of the squared Euclidean distance of output vectors to the center of mass.

2.2. Kernelizing the output

Noticing that variance (3) only requires computations involving scalar products between outputs, one can apply directly the kernel trick to the score function and thus derive a direct way to handle more complex output sets or spaces. More precisely, let us define a mapping $\phi : \mathcal{Y} \to \mathcal{H}$ that projects each output y into a vector of some Hilbert space $\mathcal{H}$ and then use the variance (3) in this feature space to grow the tree. This reduces to the supervised learning problem defined earlier with a loss function equal to:

$$\ell(f(x), y) = ||\phi(f(x)) - \phi(y)||^2. \qquad (5)$$

In this case, variance $\text{var}\{\phi(y)|S\}$ can be written:

$$\frac{1}{N}\sum_{i=1}^{N}\langle\phi(y_i), \phi(y_i)\rangle - \frac{1}{N^2}\sum_{i,j=1}^{N}\langle\phi(y_i), \phi(y_j)\rangle$$

where $||.||$ and $\langle\cdot,\cdot\rangle$ resp. denotes the norm and the inner product in $\mathcal{H}$.

Defining now $k : \mathcal{Y} \times \mathcal{Y} \to I\!R$ as a (positive semi-definite) kernel over the output space, which induces some feature map ϕ into some (possibly infinite-dimensional) Hilbert space $\mathcal{H}$ such that

$$k(y, y') = \langle\phi(y), \phi(y')\rangle,$$

the variance on the output vector in $\mathcal{H}$ becomes:

$$\text{var}\{\phi(y)|S\} = \frac{1}{N}\sum_{i=1}^{N} k(y_i, y_i) - \frac{1}{N^2}\sum_{i,j=1}^{N} k(y_i, y_j),$$

which plugs directly into the score computations, and thereby extends tree induction to a very large class of complex output spaces over which various kernels have already been defined. We will call this kernel formulation of the tree growing algorithm OK3, for Output Kernel Trees.

Notice that multiple output regression trees and classification trees are particular cases of OK3. Indeed, the first one consists of using a linear kernel between outputs while using the Dirac kernel, $k(y, y') = 1(y = y')$, with a qualitative output yields standard classification trees with the Gini entropy.

Notice also that in terms of the kernel, the loss function that the resulting trees try to minimize can be written:

$$\ell(f(x), y) = k(f(x), f(x)) + k(y, y) - 2k(f(x), y), \qquad (6)$$

which shows that choosing a kernel among outputs actually amounts at choosing a loss function.

2.3. Making a prediction

In the output feature space, the prediction (4) associated to a leaf node L becomes:

$$\hat{\phi}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(y_i). \quad (7)$$

Usually, however, we are interested in making predictions in $\mathcal{Y}$. One prediction of interest is thus the output in $\mathcal{Y}$ whose feature vector is $\hat{\phi}_L$, i.e. the pre-image $\hat{y}_L = \phi^{-1}(\hat{\phi}_L)$. However, we do not want to make the assumption that we have access to the feature space and furthermore, there may not be a point $\hat{y}_L$ in $\mathcal{Y}$ such that $\phi(\hat{y}_L) = \hat{\phi}_L$. In such cases, an approximate pre-image may be obtained from the kernel only by

$$\begin{aligned} \hat{y}_L &= \arg\min_{y' \in \mathcal{Y}} ||\phi(y') - \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(y_i)||^2 \\ &= \arg\min_{y' \in \mathcal{Y}} k(y', y') - \frac{2}{N_L} \sum_{i=1}^{N_L} k(y_i, y'), \quad (8) \end{aligned}$$

which is the output that when mapped in the feature space is the closest to the center of mass in the leaf. In practice, the computation of the arg min in (8) is tractable only for very specific choices of $\mathcal{Y}$ and kernel k. For example, with a linear kernel in $\mathbb{R}^n$, the computation of (8) leads to the prediction (4).

The pre-image problem is common to all methods learning from kernelized outputs (see Section 2.8). Several techniques have been proposed to approximate (8) (see, e.g., Scholkopf & Smola, 2002; Ralaivola & d'Alché-Buc, 2003). The approximation we propose is inspired from the simple approximation used in (Weston et al., 2002) which replaces the $\arg\min_{y' \in \mathcal{Y}}$ over $\mathcal{Y}$ by an $\arg\min_{y' \in LS}$ over outputs that appear in the learning sample only. Taking into account the tree structure, we further restrict this search to those outputs that appear in the leaf reached by the test example, which further reduces the computational burden.

It is important, however, to notice that during tree growing there is no need to compute pre-images. Also, the loss of a prediction on a test case (x, y) may be obtained without computing pre-images, by using (6) and exploiting (7) and the kernel formulation of scalar products to express $k(f(x), f(x))$ from the learning sample values $k(y_i, y_j)$ only. Thus cross-validation (e.g. in the context of pruning, or meta-parameter tuning) can also be done without computing pre-images.

2.4. Ensembles of output kernelized trees

While useful for their interpretability, single trees are usually not competitive with other methods in terms of accuracy, essentially because of their high variance. Thus, in the context of classification and regression problems, ensemble methods have been proposed to reduce variance and improve accuracy. In general, these methods grow an ensemble of diverse trees instead of a single one and then combine in some fashion the predictions of these trees to yield a final prediction.

Among these methods, those which only rely on score computations to grow the trees and which (in the context of regression) combine predictions by simply averaging them, can be directly extended to OK3.

As a matter of fact, the average prediction of an ensemble of trees $\mathcal{T}$, which is an average of sums like (7), may be written as a weighted sum of output feature space vectors from the learning sample, i.e.:

$$\hat{\phi}_{\mathcal{T}}(x) = \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) \phi(y_i). \quad (9)$$

Indeed, for a single tree t (7) may be rewritten as

$$\hat{\phi}_t(x) = \sum_{i=1}^{N_{LS}} k_t(x_i, x) \phi(y_i),$$

where $k_t(x, x') = N_L^{-1}$ if x and x' reach the same leaf L in t and 0 otherwise; hence the average prediction of an ensemble of trees $\mathcal{T} = \{t_1, \ldots, t_M\}$ is given by (9) with

$$k_{\mathcal{T}}(x, x') = M^{-1} \sum_{i=1}^{M} k_{t_i}(x, x').$$

Notice that $k_{\mathcal{T}}$ is non-negative and normalized over the learning sample ($\sum_i k_{\mathcal{T}}(x_i, x) = 1$). It is positive only for pairs of inputs which reach at least in one of the trees the same leaf, and it is actually a (positive semi-definite) kernel over the input space inferred in a supervised way by tree growing (Geurts et al., 2006).

An ensemble prediction can thus be obtained by an approximate pre-image in the following way:

$$\begin{aligned} \hat{y}_{\mathcal{T}}(x) &= \arg\min_{y' \in \mathcal{Y}} ||\phi(y') - \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) \phi(y_i)||^2 \\ &= \arg\min_{y' \in \mathcal{Y}} k(y', y') - 2 \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) k(y_i, y'). \quad (10) \end{aligned}$$

Like for single trees, we will further simplify (10) by an arg min over the y_i in the learning sample such that $k_{\mathcal{T}}(x_i, x)$ is non zero.

In our experiments, we will compare OK3 with single trees and OK3 ensembles grown with the randomization of bagging and extra-trees. Bagging grows each

tree from a bootstrap sample of the original learning sample (Breiman, 1996) while locally maximizing scores to choose splits. The extra-trees, on the other hand, are grown from the complete learning sample while randomizing the choice of the split at each node. We refer the interested reader to (Geurts et al., 2006) for the exact description of this algorithm and its comparison with other ensemble methods.

2.5. Attribute selection and ranking

An interesting feature of tree-based methods is that they can be exploited to rank attributes according to their importance for predicting the output value. This feature is especially interesting with ensemble methods which are not interpretable by themselves.

In the context of OK3, we propose to compute the importance of an attribute by computing for each split (in a tree, or in an ensemble of trees) where the attribute is used the total reduction of variance brought by the split, which is actually $N.Score(S,T)$, and by summing these reductions. Thus, attributes that do not appear in a tree have an importance of zero, and those that are selected close to the root nodes of the trees typically receive high scores. Since the variance reduction can be computed in terms of kernel values only, this computation again does not require to refer to the output spaces $\mathcal{Y}$ or $\mathcal{H}$.

2.6. Supervised kernel learning

Let us assume that the outputs corresponding to learning sample cases are not available, and that only the values of the kernel between these (hypothetical) outputs are given, in the form of a $N_{LS}\times N_{LS}$ gram matrix K with $K_{i,j} = k(y_i, y_j)$. Since the construction of a tree only requires the output kernel matrix, we can still grow a tree or an ensemble from such data. Furthermore, from these latter we can make predictions about kernel values between the unknown outputs that would correspond to any two points of the input space.

Indeed, let us first consider a single tree and two inputs x_1 and x_2 reaching leaves L_1 and L_2 respectively. If we knew the outputs over the learning sample, these leaves would point respectively to the subsamples of outputs $\{y_1^1, \ldots, y_{N_{L_1}}^1\}$ and $\{y_1^2, \ldots, y_{N_{L_2}}^2\}$ and if, moreover, we knew the feature map ϕ, we could compute from this information the predictions in $\mathcal{H}$ for x_1 and x_2, and their inner product by:

$$\hat{k}(x_1, x_2) = \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} \langle \phi(y_i^1), \phi(y_j^2) \rangle.$$

Since the latter expression refers only to inner products it can be computed directly from the Gram matrix by

$$\hat{k}(x_1, x_2) \quad = \quad \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} k(y_i^1, y_j^2).$$

The general formulation of this, for an ensemble of trees $\mathcal{T}$, writes nicely as a convolution of the Gram matrix K by the tree-kernel $k_{\mathcal{T}}$:

$$\hat{k}_{\mathcal{T}}(x_1, x_2) = \sum_{i=1}^{N_{LS}} \sum_{j=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x_1) k_{\mathcal{T}}(x_j, x_2) K_{i,j}. \quad (11)$$

Obviously this approximate kernel will be positive semi-definite as soon as the given Gram matrix is so. Hence, OK3 also provides a means to generalize arbitrary kernels as a function of attributes defined over some input space. Our second experiment below will show that the problem of supervised graph inference may be formulated like this. This provides also a way to solve the kernel completion task defined in (Tsuda et al., 2003). With respect to the transductive approach of (Tsuda et al., 2003) however, we build a model in the form of a function $\hat{k}_{\mathcal{T}}$ and do not exploit inputs of the test samples. Although this will not be illustrated here, we mention that this feature could also be useful to approximate a kernel whose practical evaluation is computationally intensive.

2.7. Implementation issues

In the case of the formulation (3), it is possible to compute incrementally the score for all possible splits based on a numerical input variable and hence the determination of the best split among N was in the order of $O(N)$ for a subset of size N. With OK3, this incremental update is still possible but requires now $O(N)$ operations, leading to a quadratic complexity of the determination of the best split at a tree node. This is unavoidable in the general case. However, if there exists an output feature space of lower dimension d in correspondence with the kernel, it may be more advantageous to use multiple output regression trees in this space for computational efficiency reasons, the latter method being of order $O(N.d)$. Both approaches will nevertheless output exactly the same tree.

In the general case, there is also an additional burden for the computation of a prediction with respect to standard trees. The computation of the pre-image, as well as the computation of a kernel prediction, is in the order of $O(N_L^2)$ for a single tree and quadratic in the support of $k_{\mathcal{T}}(x, .)$ for an ensemble. In the case of single trees, however, it is possible to pre-compute predictions at all tree leaves so as to avoid this quadratic effect at test time. Since ensemble methods are usually

used with unpruned trees, we can also expect in practice that the size of the support $k_{\mathcal{T}}(x, .)$ will be almost independent of the learning sample size and thus that the test complexity will remain far below its worst case value of $O(N_{LS}^2)$.

2.8. Related algorithms

OK3 is related to a couple of works in the tree world. To the best of our knowledge, multiple output regression trees date back to (Segal, 1992) which has proposed to replace variance (2) by an average Mahalanobis distance to the center of mass. This is strictly equivalent in using OK3 with a kernel $k(y_i, y_j) = y_i V^{-1} y_j^T$ where V is a covariance matrix and reduces to multiple output regression trees when V is the identity matrix. Our work is also closely related to the predictive clustering trees (PCT) proposed by Blockeel et al. (1998) and applied, e.g., for hierarchical classification (Todorovski et al., 2002) and ranking (Blockeel et al., 2002). PCT generalize classification and regression trees by replacing the notion of variance by the general form: $\frac{1}{N}\sum_{i=1}^{N} d(y_i, p)^2$ where d is some arbitrary distance metric and p, called the prototype, is the point that minimizes the latter expression according to p. When d is the euclidian distance in the output feature space, PCT will produce the same trees as OK3 would. The main difference however is that PCT requires to compute explicitly a prototype p.

Besides trees, several researchers have focused recently on the problem of complex output prediction with kernel methods. Weston et al. (2002) define a kernel on the output and try to find a prediction that minimizes (1) with the loss function (5) associated with the kernel. Their method, called kernel dependency estimation (KDE), first applies kernel PCA on the output kernel to reduce the dimensionality of the output feature space and then fit a kernel ridge regression model for each output direction. In (Cortes et al., 2005), a new formulation of KDE is proposed that does not require anymore the prior dimensionality reduction and, like OK3, works (implicitly) in the full output feature space. Compared to our method, the use of kernel ridge regression allows to take advantage of a kernel on the input space which may improve accuracy, at the expense however of interpretability and computational complexity.

The approaches adopted in (Tsochantaridis et al., 2005; Weston et al., 2005; Taskar et al., 2005) exploit a kernel on input-output pairs with large-margin methods. Unlike OK3 and KDE, this leads to a combined feature space representation of inputs and outputs, especially appropriate when these two are interdependent. These methods are thus able to solve more complex structured input/output problems. The price to pay however is an increase of computing time, partially due to the requirement of pre-image computations during the training stage.

Our method is also related to kernel k-means (Dhillon et al., 2004). Both methods try to minimize the same loss function and are based on the same kernel trick, i.e., the computation of the average distance to the center of mass from kernel evaluations only. The main difference is obviously that kernel k-means minimizes this criterion in an unsupervised way while our algorithm does it in a supervised way from input variables.

3. Experiments

To highlight the generality of the approach, we first consider a problem of structured ouput prediction and then a supervised kernel learning task.

3.1. Image reconstruction

To give a first illustration of the method, we reproduce the image reconstruction experiment of (Weston et al., 2002). The problem is to predict the bottom half of an image representing a handwritten digit from its top half. The dataset we use is a subset of 1000 images from the USPS handwritten 16×16 pixel digit database[1]. We took the first 100 examples of each digit in the order of the original dataset. Each variable is scaled between -1 and +1. The first 8 lines (128 input variables) were used as input and the last 8 lines as output. As output kernel, we used a radial basis-function (RBF) kernel $k(y, y') = exp(-||y - y'||^2/2\sigma^2)$ with $\sigma = 7.0711$[2]. As in (Weston et al., 2002), we compare algorithms with 5-fold cross-validation[3].

We compare three families of algorithms: k-NN, KDE (Weston et al., 2002), and OK3[4]. For the k-NN, we found the nearest neighbors using a RBF kernel on the input and computed the prediction as the closest output in the learning sample to the center of mass among the neighbors. For KDE, we tried both linear and RBF kernels on the inputs. The parameters of these methods were selected by another run of 5-

[1] the ZIP code dataset from `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html`

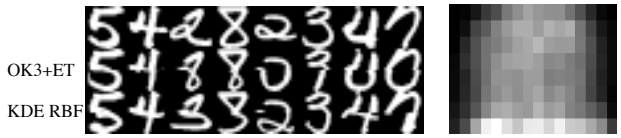
[2] Since we could not find the exact setting of (Weston et al., 2002), this value was chosen so that the range of our errors were comparable to theirs.

[3] To ease the reproduction of our results, the folds were class-stratified and selected in the order of the 1000 images.

[4] For k-NN and KDE, we use the Matlab implementation of these methods in the spider library (`http://www.kyb.tuebingen.mpg.de/bs/people/spider/`). For OK3, we used our own implementation in C.

Table 1. Comparison of different methods on the image reconstruction task (RBF loss)

Method	$N_{LS} = 200$	$N_{LS} = 800$
Baseline	1.0945 ± 0.0125	1.0853 ± 0.0227
Best achievable	0.4701 ± 0.0064	0.3584 ± 0.0118
k-NN	0.8587 ± 0.0172	0.7501 ± 0.0316
KDE linear	0.8630 ± 0.0030	0.7990 ± 0.0215
KDE RBF	0.7892 ± 0.0066	0.6778 ± 0.0264
OK3+Single trees	1.0399 ± 0.0150	0.9013 ± 0.0232
OK3+Bagging	0.8643 ± 0.0096	0.7337 ± 0.0263
OK3+Extra-trees	0.8169 ± 0.0109	0.6949 ± 0.0261



(a) (b)

Figure 1. (a) Examples of predictions and (b) attribute importance measurement on the image reconstruction task

fold cross-validation internal to the previous run[5]. For OK3, we compared single (unpruned[6]) trees, bagging, and extra-trees. We used ensembles of 100 trees and the default setting of extra-trees (Geurts et al., 2006).

For each method, we computed the average loss corresponding to the RBF kernel (i.e., $\ell(y, y') = 2(1 - exp(-||y-y'||^2/2\sigma^2))$, which is the loss that KDE and OK3 explicitly seek to minimize. Results are gathered in Table 1. The two columns correspond respectively to the case where we used learning folds of size 200 (and testing folds of size 800) or learning folds of size 800 (and testing folds of size 200). The first line corresponds to the baseline error, obtained by a model which outputs the lower part of the learning image which is closest to the average output feature vector in the learning sample. All methods return outputs from the learning sample for their predictions, which imposes a lower bound on the loss over a test sample. This lower bound of error is achieved when the output prediction is selected as the lower part of all learning images which is the closest to the lower part of the test image according to the *output* RBF kernel, and is shown on the second line of the table.

We observe from the results given in Table 1 that, for both learning sample sizes, all methods are better than the baseline, but remain quite sub-optimal with respect to the best achievable. Among tree-based methods, single trees are only slightly better than the mean prediction, while bagging and extra-trees both significantly improve with a clear advantage to the latter. The best results are obtained by KDE with the RBF kernel, i.e. while using the same kernel on the input space as the one targeted over the output space. It is interesting to note that OK3 with extra-trees, while it has to learn its kernel from the sample, is able to reach almost as good results. Figure 1(a) gives some examples of predictions returned by KDE+RBF and OK3+Extra-trees with $N_{LS} = 800$. Note that OK3 results were obtained by limiting the search of pre-images to the outputs in the leaves of interest. This approximation had no effect on accuracy but significantly decreased computing times for testing (up to a factor 15 with Extra-trees and $N_{LS} = 800$).

As a further illustration of OK3 features, Figure 1(b) plots the importance of each input pixel in a model derived with extra-trees from the learning sample of all 1000 images. Useless (zero importance) pixel values are drawn in black and the most important ones are drawn in white. This shows that most information is concentrated in the lower part of the input images, i.e. where they should align with the output image.

3.2. Supervised graph inference

Definition of the task. Let $G = (V, E)$ be a graph with vertices V and edges $E \subset V \times V$. We suppose that each vertex is described by some features in some input space $\mathcal{X}$, and denote by $x(v)$ this information. Let $V' \subset V$ and let $E' = \{(v, v') \in E | v, v' \in V'\}$; thus $G' = (V', E')$ is a subgraph of G. We suppose that the graph is undirected, i.e. $(v, v') \in E \Rightarrow (v', v) \in E$. The goal of supervised graph inference is then to determine from the knowledge of G' a function $e(x(v), x(v')) : V \times V \to \{0, 1\}$, ideally such that $e(x(v), x(v')) = 1 \Leftrightarrow (v, v') \in E$. To our knowledge this task was first introduced by Vert and Yamanishi (2004) in the context of biological networks but can be instantiated in various application domains.

To solve this problem with OK3, we first have to define an output kernel $k(v, v')$ between vertices such that adjacent vertices lead to high values of k and non-adjacent ones lead to smaller ones. Then, OK3 can be applied on the known subgraph, by using the set of vertices V' described by their inputs and the gram matrix $k(v, v')$. It will build a tree-based model allowing to predict, according to Eqn. (11), the kernel of two arbitrary vertices as a function of their inputs. Thresholding the value of the learned kernel allows then to infer edges over unseen vertices.

[5] k was selected in {1,5,10,15,20}, σ of the RBF kernel in {22.3607, 7.0711, 2.2361, 0.7071, 0.2236, 0.0707, 0.0224}, and the regularization parameter of KDE in {10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}}

[6] Pruning did not improve in this case.

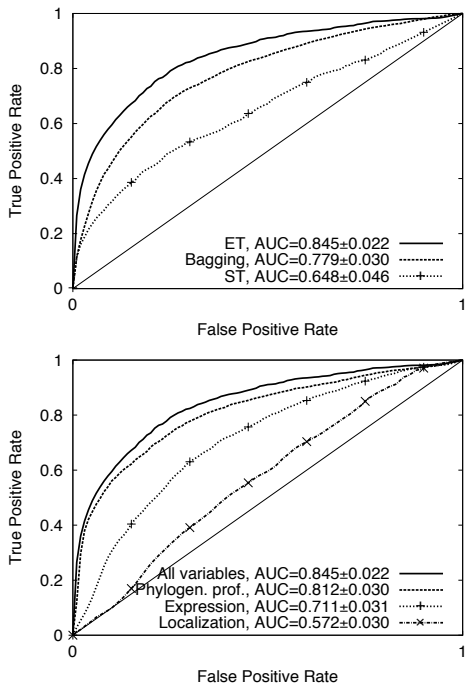


Figure 2. Top, ROC curves of OK3 with single (ST), bagged (TB) and extra-trees (ET), bottom, ROC curves using different sets of features with ET

Enzyme network inference. We reproduce the experiments from (Yamanishi et al., 2005) which aim at reconstructing the yeast enzyme network, where each vertex corresponds to an enzyme and edges connect enzymes which catalyze two successive reactions in a biochemical pathway. The "golden standard" network used in (Yamanishi et al., 2005)[7] was retrieved from the KEGG/PATHWAY database. It is composed of 668 enzymes connected by 2782 edges. Each enzyme is described by three sets of attributes: (1) *Gene expression data*: 157 numerical attributes measuring expressions in different microarray experiments of the gene coding for the enzyme; (2)*Localization data*: 23 boolean attributes coding the presence/absence of the enzyme in various intracellular locations; (3) *Phylogenetic profile*: 145 boolean variables coding the presence/absence of an orthologuous protein across 145 organisms. To represent the graph structure we use a diffusion kernel (Kondor & Lafferty, 2002), yielding a Gram matrix $K = \exp(-\beta L)$, (where $L = D - A$ is the Laplacian, D the diagonal matrix of node connectivities, and A the adjacency matrix). We evaluate our algorithm by 10-fold cross-validation: on each run, we compute the Gram matrix on 9 folds, apply OK3 and then compute from the resulting model all kernel predictions that involve at least one enzyme from the test fold.[8] A network can then be reconstructed by connecting enzyme pairs with a kernel value above a threshold. To evaluate the accuracy of such predictions, we analyze ROC curves obtained by varying the threshold, the true positive rate being the proportion of correctly predicted existing edges and the false positive rate the proportion of non existing edges erroneously predicted. We (vertically) average the ROC curves obtained on the different folds and also compute (average) areas under the ROC curves (AUC values).

The top graph of Figure 2 compares OK3 with different tree-based methods, while using all 325 attributes. We observe that single trees are clearly outperformed by bagging, itself clearly outperformed by extra-trees. Comparing the AUC of 0.845 we get with extra-trees to the best value of 0.804 obtained in (Yamanishi et al., 2005) with the same inputs[9], we conclude that our results are quite competitive.

Reproducing an experiment from (Yamanishi et al., 2005), the bottom graph of Figure 2 shows the effect of using different subsets of candidate attributes. The phylogenetic profiles appear as the most informative ones, while localization data alone does not bring much information at all about enzyme relations. This is also confirmed by attribute ranking, where we found that the first five attributes correspond to phylogenetic features, while the last 20 ones are almost all localization variables. Comparing our ROC curve for phylogenetic data only with the one in (Yamanishi et al., 2005), we observe here a much higher AUC, which suggests that OK3 is well suited for handling discrete attributes.

Discussion. The algorithm proposed in (Yamanishi et al., 2004) and (Vert & Yamanishi, 2004) determines a mapping of inputs $x(v)$ into a vector $f(x(v))$ of $\mathbb{R}^L$, such that vertices v, v' of G' known to be adjacent are mapped to nearby vectors $f(x(v)), f(x(v'))$, and to use this mapping to predict unknown edges. In (Yamanishi et al., 2004), the mapping f is learned as the first few components obtained by kernel canonical correlation analysis between a kernel on the inputs and a diffusion kernel on the graph. In (Vert & Yamanishi, 2004), the mapping f is learned directly by minimizing a functional expressing the fact that known connected pairs should be close in the feature space.

OK3 differs with their approaches in that it computes the kernel approximation directly in the original at-

[7] `http://web.kuicr.kyoto-u.ac.jp/~yoshi/ismb05/`

[8] These conditions, including the value of $\beta = 1$ chosen for the kernel, are those of (Yamanishi et al., 2005), except for the specific folds, which we could not find out.

[9] The best AUC obtained by Yamanishi et al. (2005) is actually 0.836 but this value was obtained with some additional chemical information that was not exploited here.

tribute space, i.e. without explicitly inferring the mapping f into a Euclidean space of prespecified dimensionality, and without having to define *a priori* a kernel over the input space. In general, the advantage of our approach is that it is more straightforward, since totally automatic, and that the flexibility of extra-tree based kernel learning may improve accuracy.

4. Conclusion and perspectives

The kernelization of tree-based methods that we propose in this paper opens these methods to structured output prediction and supervised kernel learning. It allowed us to tackle two problems of different nature: a pattern completion task and a graph inference problem. Its combination with ensemble methods can reach high level performances while keeping the possibility to rank and select features. The interpretability, non parametric nature, and reasonable complexity of these methods make them interesting alternatives to full kernel-based methods, enriching the panel of methods able to handle structured outputs.

Immediate future works will focus on the application of the method on various tasks which will also raise the question of the choice of an appropriate output kernel for the problem at hand. While we restrained this work on ensemble methods based on randomization, the same kernel trick applies to boosting type ensemble methods and deserves to be studied. On a broader point of view, the structured output prediction problem addressed here is still at its infancy and suggest new open questions such as the impact of the output kernel together with the choice of input space on generalization ability. These questions are certainly worth further investigations in general and in particular, in the context of output kernelized trees.

Acknowledgments

The authors thank Y.Yamanishi for providing the enzyme network data. P.G. is a postdoctoral researcher at the CNRS (France) and a scientific research worker at the FNRS (Belgium). F. d'A.-B. thanks Genopole for funding her research activities (ATIGE funding).

References

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi-classification. *KDD-2002 Workshop Notes: MRDM 2002, Workshop on Multi-Relational Data Mining* (pp. 21–35).

Blockeel, H., De Raedt, L., & Ramon, J. (1998). Top-down induction of clustering trees. *Proc. of ICML 1998* (pp. 55–63).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L., Friedman, J., Olsen, R., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International (California).

Cortes, C., Mehryar, M., & Weston, J. (2005). A general regression technique for learning transductions. *Proceedings of ICML 2005* (pp. 153–160).

Dhillon, I., Guan, Y., & Kullis, B. (2004). *Kernel k-means, spectral clustering and normalized cuts* (Technical Report). University of Austin, Texas.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning Journal (advance access: DOI 10.1007/s10994-006-6226-1)*.

Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input. *Proc. of ICML 2002* (pp. 315–322).

Ralaivola, L., & d'Alché-Buc, F. (2003). Dynamical modeling with kernels for nonlinear time series prediction. *Advances in Neural Information Processing Systems*.

Scholkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Segal, M. (1992). Tree structured methods for longitudinal data. *Journal of the American Statistical Association*, *87*, 407–418.

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. (2005). Learning structured prediction models: A large margin approach. *Proc. of the 22nd International Conference on Machine Learning* (pp. 897–904).

Todorovski, L., Blockeel, H., , & Dzeroski, S. (2002). Ranking with predictive clustering trees. *Proc. of the 13th European Conference on Machine Learning* (pp. 444–456).

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Tsuda, K., Akaho, S., & Asai, K. (2003). The em algorithm for kernel matrix completion with auxiliary data. *Journal of machine learning research*, *4*, 67–81.

Vert, J.-P., & Yamanishi, Y. (2004). Supervised graph inference. *Advances in Neural Information Processing Systems*, *17*, 1433–1440.

Weston, J., Chapelle, O., Elisseeff, A., Schoelkopf, B., & Vapnik, V. (2002). Kernel dependency estimation. *Advances in Neural Information Processing Systems*, *15*.

Weston, J., Schoelkopf, B., & Bousquet, O. (2005). Joint kernel maps. *Proceedings of the 8th International Work-Conference on Artificial Neural Networks* (pp. 176–191).

Yamanishi, Y., Vert, J.-P., & Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, *20*, i363–i370.

Yamanishi, Y., Vert, J.-P., & Kanehisa, M. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, *21*, i468–i477.