Random Forest Based Feature Induction

Celine Vens
Department of Computer Science
Katholieke Universiteit Leuven
B-3001 Leuven, Belgium
Email: celine.vens@cs.kuleuven.be

Fabrizio Costa
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
D-79110 Freiburg, Germany
Email: costa@informatik.uni-freiburg.de

***Abstract*—We propose a simple yet effective strategy to induce a task dependent feature representation using ensembles of random decision trees. The new feature mapping is efficient in space and time, and provides a metric transformation that is non parametric and not implicit in nature (i.e. not expressed via a kernel matrix), nor limited to the transductive setup. The main advantage of the proposed mapping lies in its flexibility to adapt to several types of learning tasks ranging from regression to multi-label classification, and to deal in a natural way with missing values. Finally, we provide an extensive empirical study of the properties of the learned feature representation over real and artificial datasets.**

***Keywords*-feature induction; random forests;**

I. Introduction

The problem of determining a suitable metric space tailored for a given predictive task has been receiving increasing attention in the Machine Learning and Data Mining communities. It is well known in fact, that when features of low relevance are given the same importance as those of high relevance, the predictive performance of distance/similarity based learners, such as k-Nearest Neighbors, suffers. Additionally, it is often the case that the importance of each variable depends on the overall location of the instance in the feature space. Strategies to locally adapt the feature importance or to learn nonlinear transformations of the metric space can therefore significantly enhance the performance of various predictive methods.

In the last decade, due to the excellent generalization performance and theoretical guarantees offered by Support Vector Machines, kernelized methods have become mainstream. In this context one is interested in learning the *similarity* rather than the *distance* function, although the two tasks are intimately related as one can define one notion in terms of the other. The problem of learning the kernel function has therefore become of interest. Here we argue that learning directly a task dependent feature representation, rather than learning a kernel function, offers several advantages. We start with a short review of kernel learning methods and later argument for the benefits of the direct approach.

Although a variety of methods have been developed to learn kernel functions that agree on the partitioning of the instances according to the target under consideration, many of the proposed techniques are applicable only in transductive settings ([1], [2], [3]) where test instances are supposed to be available during the training phase. Among the inductive methods, there are approaches that attempt to learn a linear transformation of the original kernel, tackling the so called Mahalanobis learning task ([4], [5], [6]) (which fails to address the locality issue) or that learn a linear combination of kernels belonging to a specific family of parametrized functions (hyperkernels [7] or multiple kernel methods [2]). The main drawback of these approaches are high computational costs or the limited flexibility for the resulting kernel since they commit in advance to a specific functional form for the kernel family and only tune the relative kernel parameters or kernel combination parameters based on the empirical data.

Another fundamental limit of current methods is the type of information used to guide the metric induction procedure. While it is common to consider binary classification tasks, it is by no means the only scenario of interest; ranging from regression to multi-class, from multi-label classification to ranking, there are a variety of naturally occurring types of learning problems that could inform the metric induction in different ways. Even if the kernel learning could be adapted to the different tasks, we argue that the explicit feature mapping induction can offer several advantages over a metric space transformation, in terms of *flexibility* and *efficiency*.

Obtaining an explicit feature representation allows to directly use any learning algorithm without the need to *kernelize* the method. Once the new feature representation is available, it is however straightforward to employ kernel based algorithms for improved performance. Moreover, it is easier to perform additional transformations in the data processing pipeline (e.g. PCA or random projections for visualization) starting from the explicit feature representation, rather than having to manipulate the Gram matrix. Finally, there exist fast methods for very large datasets (millions of instances) that operate on explicit feature representations with high dimension (millions of features) ([8], [9]). Alternatively, one can resort to Locality Sensitive Hashing techniques [10], that need sparse vectors in input, to answer nearest neighbor queries in sub-linear times.

© 2011 IEEE

A computational advantage of the explicit representation is that the construction phase does not need pairwise distance information and that the resulting model does not need to store a quadratic amount of information (no need to store a Gram matrix).

Here we propose a simple yet effective strategy to address the aforementioned issues and achieve a sparse high dimensional encoding: the method is inductive, fully non-parametric and can be informed by a wide variety of supervised scenarios. The strategy we employ to induce features is to map the identity of the tests performed in each node of a decision tree to a feature indicator. We then aggregate all the features of each decision tree in a random forest via a hashing technique to obtain the final encoding in a vector space of arbitrary dimension. Due to the flexibility of the decision tree module in this framework we can deal in a natural way with missing values, multi-label classification and unsupervised settings. A further advantage of the proposed approach is that the implicit feature selection mechanism in decision trees induces features that are robust w.r.t. noisy or irrelevant attributes in the original data space.

II. Method

A. Random Forests

A random forest is an ensemble of random decision tree classifiers, that makes predictions by combining the predictions of the individual trees. Different random forests differ in how randomness is introduced in the tree building process. Here we consider the random forest version as used by Breiman [11], which combines bagging [12] and random feature selection [13]. More formally, a bootstrap procedure is used to create multiple randomized training sets, by sampling with replacement from the original training set, such that the bootstrap replicates contain an equal amount of instances as the original instance set. The random decision tree learner has a parameter $1 \leq v < d$ which is a positive integer. The instance space R^d is partitioned recursively to form a tree partition. The root of the random tree corresponds to R^d. At each step of the construction of the tree, v variables are selected uniformly at random from the d candidates $x(1), \ldots, x(d)$. A leaf node split is selected along one of these v variables to minimize the number of misclassified training points if a majority vote is used in each cell. The procedure is repeated until every cell is pure.

B. Random Forest Feature Induction

Every tree in the random forest gives a hierarchical clustering of the instances: the root node corresponds to the set of all instances, which is recursively partitioned by moving down the tree, until the leaves contain a (usually small) set of instances from the same class. Intuitively, two instances that are sorted into two nearby leaves, share more similar features than two instances that are classified into leaves far apart. Moreover, a decision tree generates deeper subtrees when a region in the data space is "complex" with respect to the target function. As a result, the number of nodes an instance has to visit before it reaches a leaf is proportional to the complexity of the local region. For this reason, a similarity notion, such as the cosine angle, between two instances represented as binary vectors encoding their visited nodes, shows an adaptive behavior and becomes more conservative in more complex regions. The feature induction approach that we propose is exactly this encoding of visited nodes. A different way to look at the proposed encoding technique is as follows. Consider two binary instances x and $z \in \mathbb{R}^m$ and assume that their similarity $s(x, z) \in [0, 1]$ is measured as the fraction of features that they have in common. Assume for simplicity that in a decision tree the feature on which to base a split is chosen in a purely random fashion. The chance of selecting one feature on which x and z agree is therefore equal to $s(x, z)$ and the probability of proceeding k times in this fashion is proportional to $s(x, z)^k$ since the features are chosen independently. It is easy to see that as the similarity increases, the probability of having at least a prefix path of size k increases. This supports the intuition that similar instances are classified in nearby leaves in a random decision tree. Note that when the features are not chosen at random, but rather in a way that reflects their correlation to the target, then two instances can be still classified in nearby leaves even when they differ on many of the original features *if these features are irrelevant w.r.t. the task*.

Note that any hierarchical clustering method (supervised or unsupervised) could be used to induce features with the proposed scheme. However, the use of random forests has several advantages. First, in contrast to classical hierarchical clustering approaches, by taking the target information into account we induce a task-specific similarity notion. Second, being a bootstrapped ensemble method, it is robust to small perturbations in the data. Third, due to the implicit feature selection mechanism, the representation becomes robust to non-informative or noisy features. Fourth, it offers a natural way to deal with missing values by distributing instances with a missing split value over all branches or by selecting at random one branch to follow.

The base decision tree learners that we use, are set in the Predictive Clustering Tree (PCT) framework [14], which shares the hierarchical clustering view of decision trees with this work. PCTs are constructed such that each split maximally reduces intra-cluster variance. By appropriately defining the variance function, different learning tasks can be targeted. PCTs have been used a.o. for clustering [15], multi-objective classification and regression [16], and (hierarchical) multi-label classification [17]. PCTs, including random forests over them, are implemented in the Clus system[1].

Note finally, that the chance for different decision trees

[1] http://dtai.cs.kuleuven.be/clus/

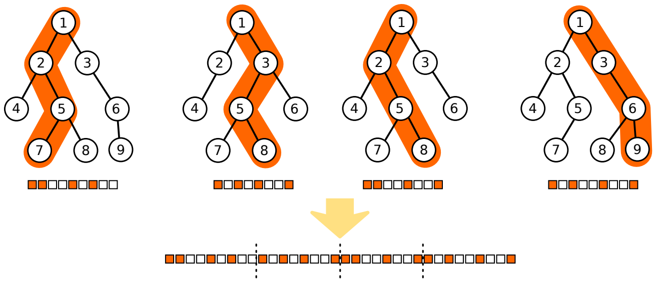

Figure 1. Example of feature induction. A single instance is classified in the leaf node by a random forest of 4 decision trees. Each decision node receive a unique identifier. If a test is satisfied in a node then the corresponding bit is asserted. Finally the encodings for each tree are combined by concatenation (or more generally by hashing the features ID onto a smaller dimensional space).

to induce identical or redundant tests (features) is low since decision trees are highly unstable classifiers [12].

C. Algorithm

We consider an instance domain X and a finite discrete class domain $\mathcal{C}$. We denote a sample of N instances with their associated class as $D = \{(x_i, c_i)|x_i \in X, c_i \in \mathcal{C}\}_{i=1}^{N}$. A decision tree model $h : X \rightarrow \mathcal{C}$ is a tree structured collection of decision nodes or tests V together with a vector containing class information associated to leaf nodes. For each instance $x \in X$ to be classified, a path $P_h(x) = \{v_1, \ldots, v_d\}$ is identified, where each $v_i \in V$ is one of the nodes that is encountered when x is sorted down the tree, and the class associated with the leaf node v_d is returned.

Here we are not interested in the prediction returned by the decision tree, but rather we use the list of the encountered tests P_h to induce a new feature space as follows. Each node in the tree represents a new binary feature. For a given instance x, the features corresponding to all nodes on the path $P_h(x)$ take value one, the value for the other features is zero. The final feature encoding induced by the random forest can be obtained by concatenating the binary vectors for all trees in the forest. For example, consider Figure 1. The four trees in the forest induce a 9+8+8+9 = 34-dimensional feature space. The sparse binary vector representation for an instance falling into the marked leaves, assuming a breadth-first numbering, is then 1:1 2:1 5:1 7:1 for the first tree, 1:1 3:1 5:1 8:1 for the second tree, 1:1 2:1 5:1 8:1 for the third tree and 1:1 3:1 6:1 9:1 for the last tree. In order to obtain a unique identifier for each node in the forest the identifiers in each decision tree receive an appropriate offset: 0 for the first tree, 9 for the second, etc. These vectors can finally be concatenated into a single sparse vector obtaining the vector: 1:1 2:1 5:1 7:1 10:1 12:1 14:1 17:1, 18:1 20:1 22:1 25:1 26:1 28:1 31:1 34:1.

Since the dimension of the induced feature space is proportional to the number of trees and the size of each tree, it is useful to allow a finer control on the feature space size for datasets with many instances. This can be obtained by applying a stopping criterion or pruning procedure to the decision tree algorithm, or by using a hash function to map the node identifier to a user definable integer interval. Figure 2 gives the pseudo code of the algorithm, where we used a hash function to control the feature space size.

```
for $k \Leftarrow 1$ to $M$ do
    $D_k \Leftarrow \mathrm{Bootstrap}(D)$
    $h_k \Leftarrow \mathrm{RandomTree}(D_k, \sqrt{f})$
$Forest \Leftarrow \bigcup h_k$
for each $h_k \in Forest$ do
    for each $x_i \in D$ do
        $P_{h_k}(x_i) \Leftarrow \mathrm{NodesOnPath}(x_i, h_k)$
        for each $n \in P_{h_k}(x_i)$ do
            $node\_id \Leftarrow \mathrm{GetNodeID}(n)$
            $feature\_id \Leftarrow \mathrm{Hash}(node\_id, F)$
            $x'_{i,\mathrm{feature\_id}} \Leftarrow x'_{i,\mathrm{feature\_id}} + 1$
return $D' \Leftarrow \bigcup x'_i$
```

Figure 2. Pseudo-code for random forest based feature induction. D denotes the instances, M the number of trees in the forest, f the number of features in the original space, and F the number of features desired in the induced space. *Bootstrap* is a function that builds a training set by sampling with replacement from the original instances. The function *NodesOnPath* returns the set of nodes on the path from the root to the leaf for a given instance and tree. *GetNodeID* returns a unique node identifier in the forest. *Hash* maps a node identifier onto a value in the range [1..F].

D. Missing Values

A decision tree algorithm has to repeatedly select a specific test for each tree node and partition the instances over the children nodes, according to the test outcome. In order to find the best test, the algorithm considers all (or a random subset of) possible features and heuristically estimates a quality score for each of them. Instances with missing values are generally ignored and do not affect the computation of the heuristic preference. However, since those instances can be useful in the calculation of later splits, a common approach, that we follow, is to assign them to all children nodes. The same approach is taken when classifying an instance with a decision tree: instances with missing values end up generating multiple paths and arriving in multiple leaves. In order to keep the length of the resulting encoding vector comparable with those of the instances with no missing values, the value *mass* associated with multiple alternatives is equally distributed; the first n-ways test associated with a missing value generates therefore n features with weight $1/n$; a successive test over a missing value generates features with weight $1/n^2$, and so on.

An alternative procedure, that does not require explicit feature value rebalancing, is to perform a random choice and select with equal probability one of the resulting paths. This is the procedure that we use in the experiments. The effect, averaged over all decision trees in the forest, approximates the multiple paths strategy.

E. Complexity

In order to analyze the computational complexity of the random forest feature construction, we identify two phases: in the first phase a random forest is built, in the second phase the structure of the forest is used to generate new features.

The computational complexity of inducing a random feature selection based decision tree is $O(aN \log N)$ with a the number of tests considered to construct a node ($\sqrt{f}$ in our case, where f is the number of features)[2] and N the number of elements in the dataset, under the assumption that a reasonably symmetric tree is built (the depth of which is logarithmic in the number of leaves) and that the evaluation of a single test takes constant time in the size of the dataset. The complexity for the first phase, the induction of the whole random forest, is then $O(M\sqrt{f}N \log N)$, where M denotes the number of trees [18].

The complexity of the second phase is $O(MN \log N)$, which means that the overall complexity is dominated by the random forest construction, a process that can easily be parallelized.

F. Inducing the Random Forest Kernel

Inducing an explicit feature representation has several advantages over implicit methods as kernels, amongst which a higher computational efficiency (linear vs. quadratic complexity w.r.t. the number of data points). In some cases one might still want to employ kernelized algorithms; for example when an explicit notion of similarity or distance is required (e.g. in order to apply prototype based methods such as kNN) or when one desires to compose several kernels (e.g. a Gaussian kernel).

In these cases it is easy to derive a (linear) kernel function from the explicit feature representation $\phi_{RF} : \mathcal{X} \mapsto \mathbb{R}^d$ generated by the random forest: $\kappa_{RF}(x, z) = \langle \phi_{RF}(x), \phi_{RF}(z) \rangle$ and its normalized version: $K_{RF}(x, z) = \kappa_{RF}(x, z)/\sqrt{\kappa_{RF}(x, x)\kappa_{RF}(z, z)}$. Note that $\kappa_{RF}(x, z)$ corresponds to the cumulative size of the common path (i.e. the common prefix) between the two instances x and z, that is $\sum_k |P_{h_k}(x) \wedge P_{h_k}(z)|$, and that $\kappa_{RF}(x, x)$ corresponds to the cumulative depth of the node that classifies x, that is $\sum_k |P_{h_k}(x)|$.

G. Related Work

While there is quite a literature on the problem of metric learning [19], the topic of explicit feature induction has in general received less attention. Methods to explicitly manipulate feature representations can be categorized into two types: 1) methods for dimensionality reduction such as PCA, compact code construction, e.g. spectral hashing, or feature selection methods, e.g., based on random forest feature relevance [11], and 2) methods for dimensionality expansion, also known as feature induction methods.

Dimensionality reduction is invoked when the majority of features are believed to be irrelevant (w.r.t. the task at hand) or when human interpretability is desired or in order to increase time efficiency for downstream algorithms. Here we are not interested in these aspects, rather we are interested in how to induce features in order to achieve an overall improved learnability. Dimensionality expansion methods have recently received attention in the image processing community [20], [21] where sparse encoding in high dimensional spaces (called sparse-overcomplete representation) is used to increase the likelihood that image categories will be linearly separable (in a similar spirit to the underlying motivation for kernels).

Common approaches to increase or induce new dimensions operate by creating nonlinear combinations of existing features to improve prediction performance. Approaches vary from straightforward combinations of all input variables under a family of algebraic operators (products, logs, etc) to more refined boosting based selection methods [22]. Differently from those approaches, here we propose a more flexible method that is fully non-parametric, i.e. it is not restricted to generate novel features as a pre-defined combination of a (small) number of original features. Another major difference lies in the fact that our approach can be supervised i.e. informed by the learning task.

Another direction of comparison is considering the induction of a similarity notion specifically from the structure of decision trees. In [11] the author considers as similarity the fraction of trees where two instances appear together in the same leaf. Torkkola and Tuv [23] propose to compute a kernel between two instances by identifying the corresponding paths, count the number of common nodes and normalize by the length of the longest path. We note that the normalized dot product between two instances encoded under our scheme is in fact equivalent to such a kernel, with the difference that the normalization factor is not $\max(|P_h(x)|, |P_h(z)|)$, as in [23], but rather $\sqrt{|P_h(x)||P_h(z)|}$.

III. Empirical Results

In this section, we empirically evaluate the random forest based feature induction approach. We start by describing the experimental setup in Section III-A. In Section III-B we devise an artificial dataset to show a specific characteristic of the induced feature space, namely its capacity to adapt the similarity notion between two data instances locally to the appropriate scale.

In Section III-C we compare the original and the induced feature spaces, both w.r.t. the notion of target alignment to measure the agreement of a Gram matrix on the transformed feature space to the actual target, as w.r.t. the performance of two classifier algorithms. We conclude that the induced

[2] In those datasets where the number of features is not comparable to the number of instances a can be absorbed in the Big O notation, but in several noteworthy cases (e.g. micro-array experiments in bioinformatics) one cannot make this assumption.

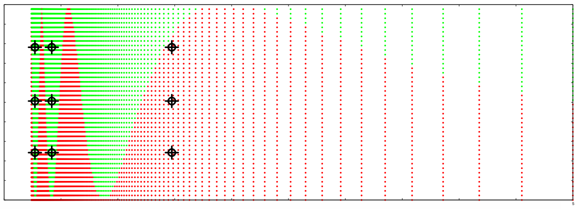

Figure 3. Artificial dataset for the $sin(1/x)$ function.

feature space is able to increase predictive performance. In Section III-D we show that this result carries over from the binary classification to multi-label classification case, thereby showing the flexibility of our approach. Another advantage of the proposed feature induction method, namely its suitability for data visualization purposes, is illustrated in Section III-E.

A. Experimental Setup

The random forest learning algorithm takes two parameters as input: the number of trees in the forest, and the number of features to be tested at each node of the trees. The latter is usually a function of the total number of features in the dataset. In our experiments we used the square root of the number of features, rounded up to the next integer [18]. The size of the forests is incrementally grown until the Gram matrix, induced on the new features, converges: we start with a forest of 50 trees, and repeatedly add 50 trees until a stopping criterion is met. In our experiments, we stop adding trees to the random forest when the average Gram matrix element-wise difference is below an arbitrary threshold of 0.01. We verified that adding more trees after this saturation point does not change the results. The optimal forest size is computed using a separate validation set.

Unless otherwise stated, we did not limit the number of features in the induced feature space, i.e., the dimension equals the total number of nodes in the random forest.

In the results, we consider the average accuracy or area under the ROC of method A (denoted a_A) to be significantly better than the average accuracy or area under the ROC of method B (denoted a_B) if $E[a_A] - std(a_A) > E[a_B] + std(a_B)$, where $std(\cdot)$ denotes the standard deviation.

B. Artificial Experiment

In this first experiment we show how the proposed strategy to induce features based on random forests is capable to enrich the Gram matrix with information from the specific task: we show that the kernel function obtained from the constructed features adapts to the complexity of the decision boundary in a local fashion. We consider the function $f(x) = sin(1/x)$ to build a target concept, with domain in $\mathbb{R}^2$, corresponding to: $c(x, y) = 1$ if $y \geq f(x)$ and 0 otherwise (see Fig. 3). The probability distribution underlying the empirical samples is $p(x, y) \propto 1/x$ which yields an increasing number of instances as we approach $x = 0$. Under these conditions we expect Support Vector Machine (SVM) models based on Gaussian kernels $K_G(x, z) = \exp(-\frac{1}{\gamma}(\langle x, x\rangle - 2\langle x, z\rangle + \langle z, z\rangle))$ to perform poorly as any fixed value for the γ parameter will lead to cases of over-generalization and cases of over-fitting in different parts of the data space. On the other hand, we expect the proposed procedure to induce a non isotropic kernel capable to adapt to the varying complexity in different regions. We have built a sample of 2000 instances and we have plotted the kernel function $K_{RF}(\hat{x}, \cdot)$ with $\hat{x}$ once in a low complexity area (Fig. 4 right column) and once in a high complexity area (Fig. 4 left column). As we can see, the kernel is wider in the direction of low complexity than in the direction of high complexity. In the high complexity region the kernel function is more peaked and generalizes only in a narrower region, preferring the vertical axis direction where the target concept is less variable.

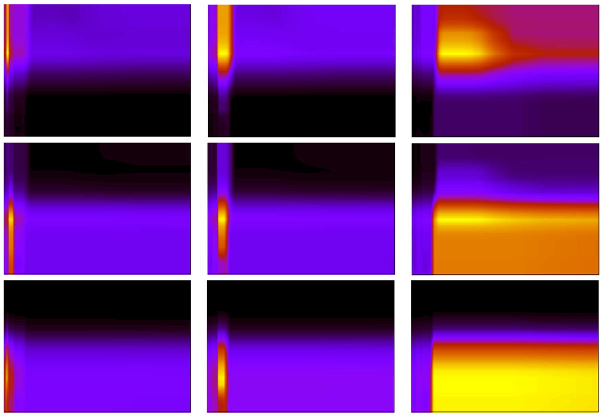

Figure 4. The kernel function $K_{RF}(\hat{x}, \cdot)$ with $\hat{x}$ in low complexity regions (right) and in high complexity regions (left). Different rows (columns) correspond to the vertical (horizontal) displacement of $\hat{x}$. In Fig. 3 the corresponding position of $\hat{x}$ is marked.

Finally, we compare this random forest based kernel to the Gaussian kernel for this dataset, by comparing the predictive performance of an SVM based on them. We have generated training sets of increasing size, using an equal number of instances from each class, and a fixed test set of 1000 instances. For each training set size, ten datasets were generated.

The SVM takes as input a parameter C that indicates the trade-off between training error and margin. The optimal value for C was chosen from the set {0.01, 0.1, 1, 10, 100, 1000}. This parameter was optimized, together with the γ parameter for the Gaussian kernel (using the set {1, 0.1, 0.01, 0.001}) and the random forest size, using one third of the training set as validation set. The complete training set was used to learn the final feature representation and SVM models with the optimized parameters. Figure 5 shows the learning curves. For completeness, we also include the

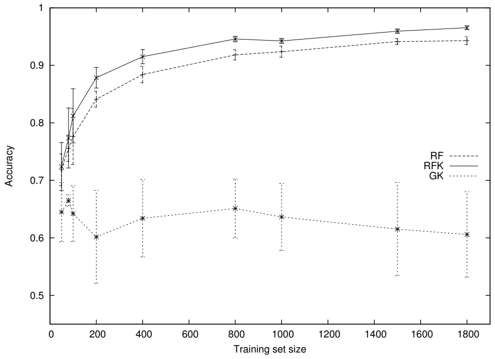

Figure 5. Learning curve for the $sin(1/x)$ function.

results of the random forest on which our kernel is induced. As we can see, for the random forest and the random forest kernel based SVM, the accuracy increases with increasing number of training instances. The predictive performance of the random forest kernel based SVM is overall the highest, obtaining 96.5% accuracy for the largest training set. As expected, the SVM based on the Gaussian kernel is not able to learn the target function.

C. Comparing Original and Induced Feature Spaces on Binary Classification Tasks

1) Datasets: In order to compare the generalization performance of classifiers built on top of the induced and original feature spaces, we have selected all datasets from the UCI repository[3] [24] that satisfy a predefined and explicit set of properties, hoping, in this way, to avoid biased conclusions induced by dataset selection phenomena. In particular, we have selected all datasets that satisfy the following conditions: 1) the type of learning task is classification, 2) the attributes type is numerical, 3) the type of data is multivariate, 4) the dataset should have between 100 and 1000 instances, and 5) should be in matrix format type. This resulted in 20 datasets of which 16 were retained for our experiments and 4 were discarded for the following reasons: the Iris dataset since we can achieve (near) perfect accuracy for all classifiers considered; the Japanese Vowels dataset since it is a time series; the Musk dataset since it is in fact a multi-instance task; and the Parkinsons dataset since it is of a relational type. Moreover, we added 4 other datasets with mixed attribute types. Table I lists the datasets and their properties. Non-binary classification tasks were transformed into binary tasks[4]. This was achieved either by considering the largest class v.s. the remaining ones, or by partitioning the classes to obtain a balanced binary classification task. Although the random forest does not require any pre-processing of the data, we transformed nominal features into a set of binary features using a one-hot encoding[5]. Finally we have *whitened* the data by normalizing all features to have zero mean and unit standard deviation. In all results involving kernel construction or k-Nearest Neighbor classifiers, missing values in the original feature representation are replaced by the feature's average value for numeric features, and the feature's mode for nominal features.

Table I
THE UCI DATASETS USED IN THE EVALUATION. WE LIST THE NUMBER OF INSTANCES, THE NUMBER OF FEATURES (INCL. THE NUMBER OF FEATURES AFTER TRANSFORMATION OF THE NOMINAL FEATURES), THE NUMBER OF INSTANCES WITH MISSING VALUES, AND THE BINARY CLASSIFICATION TASK.

Dataset	Inst.	Feat.	Missing
Arcene	200	10000	
Blood transfusion	748	4	
Breast cancer Winsc. (prog.)	198	32	4 (2%)
Breast cancer Winsc. (diag.)	569	30	
Breast cancer Winsc. (orig.)	699	9	16 (2.3%)
Connectionist bench (Sonar)	208	60	
Credit approval	690	15 (47)	37 (5.4%)
Cylinder bands	512	35 (79)	302 (59%)
Ecoli (in plasm / attached to membrane)	336	7	
Glass identification (float / non-float glass type)	163	9	
Haberman's survival	306	3	
Heart disease (Cleveland) (presence / absence of disease)	303	13 (25)	6 (2%)
Hepatitis	155	19	75 (48.4%)
Ionosphere	351	34	
Libras movement (horiz. / vertical movements)	192	90	
Mammographic mass	961	4 (14)	130 (13.5%)
Pima Indians diabetes	768	8	
Spectf heart	267	44	
Statlog (Vehicle silhouettes) (opel or saab / bus or van)	846	18	
Wine (class2 / rest)	178	13	

2) Target Alignment: We investigate the quality of the target alignment [3] for Gram matrices constructed on the induced features with respect to the matrices obtained directly from the (normalized) original data representation. We repeat here the definition of this quality measure and intuitions about its meaning as introduced in [3].

We define as inner product between two Gram matrices K_1, K_2 the quantity: $\langle K_1, K_2 \rangle_{\hat{F}} = \sum_{i,j=1,i\neq j}^{n} K_1(x_i, x_j) K_2(x_i, x_j)$, that is, as the Frobenius inner product without the contribution of the diagonal elements (since for finite samples

[3] http://archive.ics.uci.edu/ml/datasets.html, queried on April 1, 2010.

[4] For the *Libras movement* and *Glass identification* datasets, the transformation into a binary task reduced the number of instances. This reduced number is reported in the table.

[5] If the nominal feature domain has size d, we mapped the class indicator into a d dimensional vector of unit ℓ_1 norm $v = \{0, 1\}^d : ||v||_{\ell_1} = 1$.

this would result in a bias of the estimate). The empirical target alignment of two kernels k_1, k_2 that produce the two Gram matrices K_1, K_2 is defined as: $A(k_1, k_2) = \langle K_1, K_2\rangle_{\hat{F}} / \sqrt{\langle K_1, K_1\rangle_{\hat{F}} \langle K_2, K_2\rangle_{\hat{F}}}$. The alignment is a measure of similarity between Gram matrices and can be used to assess the quality of a kernel w.r.t. a specific target concept. In fact, given the specific target vector $c \in \{-1, 1\}^n$ for the sample $\{(x_i, c_i)\}_i = 1^n$, the optimal Gram matrix is cc', that is a $n \times n$ matrix T, with $T(x, z) = 1$ if two instances x and z have the same target, and $T(x, z) = -1$ otherwise.

The quality of a Gram matrix can then be assessed by measuring its similarity w.r.t. the target matrix. An intuition for this quality measure is that in order to receive a high positive score, a good Gram matrix should have high similarity values for instances that belong to the same class (this contribution will be weighted positively by the corresponding target matrix entry) and low similarity for instances belonging to different classes (this contribution will be weighted negatively).

We compare the target alignment score of a linear kernel over the random forest based features to the target alignment of a linear and Gaussian kernel over the original features. As our feature induction approach considers the target information, we applied a ten fold cross validation to compute the target alignment, in order to conduct a fair comparison. For each partition the features have been induced using 90% of the data for training, and have been evaluated over the remaining 10%. Moreover, we report the average results over ten cross validation runs with different random seeds. For each seed and each partition into folds, the number of trees in the forest is optimized by setting aside one fold out of the 9 fold training set as validation set. Once the forest size is chosen, the 9 folds are used to build the final feature set.

The empirical results, shown in Table II, confirm the competitive quality of the newly generated features compared to the original features, as estimated by the target alignment notion of a linear kernel constructed over them. We observe significant improvements in 6/20 cases for the induced features, versus in 1/20 case for the original features. A Wilcoxon Matched-Pairs Signed-Ranks Test finds the results significant with a p-value=0.12. The results for a Gaussian kernel over the original features are not included in the table due to space restrictions. Comparing to the best target alignment obtained when varying the Gaussian kernel's γ parameter using the values $\{100, 10, 1, 0.1, 0.01\}$, our approach obtains significant improvements in 5/20 cases, versus 0/20 cases for the Gaussian kernel.

Table II

Comparison of the target alignment of a linear kernel based on the induced and original features. Best result in bold and significantly best result indicated with *. The last column shows the number of trees in the forest.

Dataset	Induced	Original	$\|RF\|$
Arcene	0.061 (0.072)	**0.106** (0.139)	200
Blood trans.	***0.255** (0.036)	0.073 (0.047)	150
Br. cancer (prog.)	***0.233** (0.050)	0.019 (0.074)	200
Br. cancer (diag.)	0.565 (0.046)	**0.654** (0.071)	200
Br. cancer (orig.)	0.615 (0.029)	***0.869** (0.048)	150
Connectionist	0.083 (0.058)	**0.120** (0.093)	250
Credit approval	**0.301** (0.053)	0.259 (0.058)	200
Cylinder bands	0.036 (0.014)	**0.045** (0.039)	200
Ecoli	0.469 (0.075)	**0.546** (0.095)	200
Glass identif.	**0.095** (0.092)	0.035 (0.101)	150
Haberman	***0.235** (0.057)	0.062 (0.073)	150
Heart disease	0.210 (0.069)	**0.335** (0.098)	150
Hepatitis	***0.434** (0.063)	0.199 (0.138)	150
Ionosphere	***0.370** (0.054)	0.195 (0.089)	150
Libras movement	**0.098** (0.078)	0.058 (0.091)	200
Mammogr. mass	0.208 (0.048)	**0.291** (0.074)	200
Pima diabetes	**0.191** (0.035)	0.166 (0.051)	150
Spectf heart	***0.363** (0.059)	0.021 (0.058)	200
Statlog (Vehicle)	**0.263** (0.050)	0.190 (0.070)	150
Wine	**0.426** (0.079)	0.341 (0.114)	200

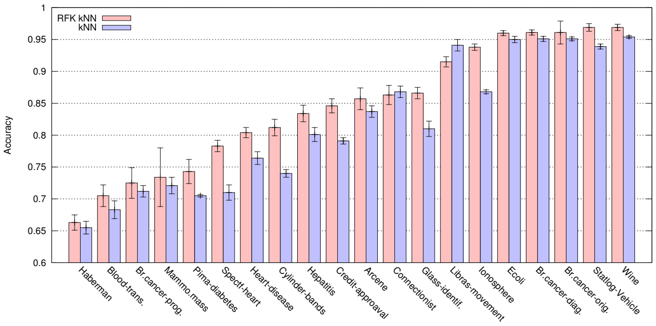

Figure 6. 1-NN accuracy for distance based on the induced or the original data space, over 20 UCI datasets.

3) Nearest Neighbor Classification: To assess the metric space induced by the generated features, we evaluate the performance of the k-Nearest Neighbor (kNN) classifier. We again compare both feature spaces using ten repetitions of tenfold cross validation, with the forest size optimized as before. Note that for each train/test partition, only the training set is used to induce the new feature space. We report results (see Figure 6) for k=1 (although k=3 and k=5 exhibit an analogous trend). We observe that the proposed method only in 1/20 case yields metric information that leads to significantly worse kNN generalization performance (Libras movement) while it leads to significantly better performance in 12/20 cases. A Wilcoxon Matched-Pairs Signed-Ranks Test finds the results significant with a p-value=0.0004.

A noteworthy characteristic of the proposed approach is that it exploits the implicit random forest feature selection mechanism: noisy and non-informative features are seldom selected by decision trees so that on average they will not be represented in the induced feature space, and not contribute to the definition of the similarity notion. A metric inducing strategy that does not perform any kind of feature selection, on the other hand, will be negatively affected by those noisy features. To show the size of this effect we run a simple experiment on the Connectionist dataset. We add an increasing number of noisy features (20%, 100% and 500% of the original 60 features) with uniform random values with zero mean and unit standard deviation. Under these conditions we observe, as expected, a moderate decrease in the predictive performance of 1-NN based on the RF induced features: 0.858 ± 0.017, 0.850 ± 0.020, 0.820 ± 0.010, for 20%, 100% and 500% added features respectively; the standard 1-NN performance however is severely affected: 0.834 ± 0.009, 0.775 ± 0.009, 0.667 ± 0.009.

4) Support Vector Machine Classification: We also compare the predictive performance of a linear Support Vector Machine (SVM) classifier when based on the induced features or the original features. Again all experiments were run using ten repetitions of ten fold cross validation, with different random seeds to create the folds.

For each seed and each fold, an internal parameter optimization is performed as follows. One fold out of the 9 fold training set is set aside as validation set, to determine the optimal forest size for the random forest - by calculating the resulting kernel matrix difference as explained above. The 9 folds are then used to construct the induced feature set with the optimal forest size. For both the original and induced feature representations, the same procedure (i.e., setting one fold aside as validation set) is used to determine the optimal value of the C parameter from the following set $\{0.01, 0.1, 1, 10, 100, 1000\}$. Once the parameter is chosen, the 9 folds are used to build the final SVM models.

The average number of trees in the forests ranges from 131.5 to 225 for all datasets. The accuracy comparison between the SVM constructed over the induced features and the original features is shown in Fig 7. The numerical results for the area under the ROC are reported in Table III. We observe that there is a clear performance improvement in building a linear SVM on top of the induced features. The AUROC results reported in Table III show that in 7/20 cases there is a significant performance increase, versus 1/20 case for the original features. A Wilcoxon Matched-Pairs Signed-Ranks Test finds the results significant with a p-value=0.09. Moreover we observe that, as expected, the proposed method improves on the Cylinder bands dataset which exhibits a large number of missing values. On the other dataset with many missing values (Hepatitis) though, there is no significant performance difference, an effect that we ascribe to the small size of the dataset.

5) Run Times: Finally, we report run times for inducing the feature encoding with a single random forest of fixed size (200 trees) on the full dataset. On a Xeon E5420 (8 cores @2.5GHz) architecture, the average run time over all considered UCI datasets is 61.5s, ranging from 11.0s for Hepatitis to 431.7s for Arcene. Note that the constant hidden in the asymptotic notation (see Section II-E) is very small (10^{-5}).

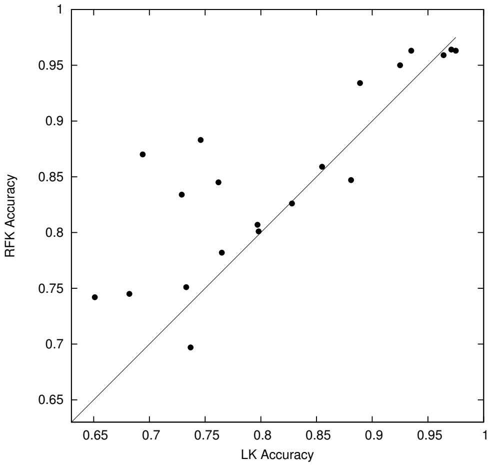

Figure 7. Accuracy scatter-plot. Points above the diagonal show a win for the SVM learned on the induced w.r.t. the original feature space.

Table III
AUROC FOR 10 REPETITIONS OF A 10-FOLD CROSS VALIDATION FOR THE SVM CLASSIFIER. BEST RESULT IN BOLD AND SIGNIFICANTLY BEST RESULT INDICATED WITH *.

Dataset	Induced features	Original features
Arcene	0.922 (0.006)	**0.929** (0.018)
Blood trans.	0.693 (0.011)	***0.734** (0.007)
Br. cancer (prog.)	0.571 (0.026)	**0.580** (0.029)
Br. cancer (diag.)	0.989 (0.002)	**0.992** (0.003)
Br. cancer (orig.)	0.986 (0.002)	**0.989** (0.002)
Connectionist	***0.933** (0.010)	0.815 (0.015)
Credit approval	**0.919** (0.005)	0.913 (0.004)
Cylinder bands	***0.900** (0.006)	0.797 (0.006)
Ecoli	***0.978** (0.004)	0.969 (0.004)
Glass identif.	***0.927** (0.008)	0.733 (0.016)
Haberman	0.654 (0.017)	**0.656** (0.014)
Heart disease	**0.904** (0.007)	0.898 (0.007)
Hepatitis	0.826 (0.032)	**0.843** (0.033)
Ionosphere	***0.971** (0.003)	0.925 (0.016)
Libras movement	***0.944** (0.015)	0.803 (0.026)
Mammo. mass	0.841 (0.004)	**0.846** (0.002)
Pima diabetes	0.817 (0.007)	**0.818** (0.006)
Spectf heart	**0.814** (0.024)	0.794 (0.019)
Statlog (Vehicle)	***0.995** (0.001)	0.986 (0.001)
Wine	0.996 (0.001)	**0.997** (0.001)

D. Comparing Original and Induced Feature Spaces on Multi-label Classification Tasks

To illustrate the flexibility of the proposed feature induction method, we also report results for multi-label datasets. In multi-label classification, each instance is associated with a set of labels. We have chosen 5 datasets (available with a predefined train/test split) from the Mulan system webpage[6] [25] with various characteristics, see Table IV.

[6] http://mulan.sourceforge.net/datasets.html

Table IV
THE MULTI-LABEL DATASETS USED IN THE EVALUATION. FOR EACH DATASET WE LIST THE NUMBER OF (TRAINING) INSTANCES, DESCRIPTIVE FEATURES, AND LABELS. RIGHT: MICRO-AVERAGED AUC FOR THE ML-KNN CLASSIFIER. BEST RESULT IN BOLD.

Dataset	Inst. (Train)	Feat.	Labels	Induced	Orig.
yeast	2417 (1500)	103	14	0.811	**0.812**
scene	2407 (1211)	294	6	**0.875**	0.822
emotions	593 (391)	72	6	**0.769**	0.733
medical	978 (333)	1449	45	**0.925**	0.904
rcv1v2 (subset1)	6000 (3000)	47236	101	**0.532**	0.524

We used ML-kNN [26], a k-Nearest Neighbor classifier for multi-label tasks included in the Mulan software package, to compare the predictive performance obtained on the original and induced features. To optimize the number of trees in the random forest, we set aside 30% of the training set as validation set. After determining the forest size, the complete training set was used to compute the final forest. The evaluation measure we report is the micro-average of the AUROC for the different labels. The results are shown in Table IV (right part). We conclude that the induced feature space leads to better classification performance in 4/5 cases. A Wilcoxon Matched-Pairs Signed-Ranks Test finds the results significant with a p-value=0.05.

E. Data Visualization

The explicit feature encoding eases subsequent data elaboration, for example data visualization using PCA or, more conveniently [27], random projections. Given the supervised nature of the feature induction we expect the low dimensional plots to show a better class separation than what is naturally observable in the original data space.

To illustrate this effect, we use a protein-protein interaction (PPI) dataset for the yeast organism, as prepared by [28]. The dataset consists of a training and a test set of 8917 protein pairs each, of which 4917 are positive (i.e., interacting) and 4000 are negative (i.e. non-interacting). The negative pairs are randomly generated, as non-interacting protein data are not available. The features consist of 4293 Pfam domains. For each domain, the value is 0, 1, or 2 depending on how many of the two interacting proteins exhibit the specific domain. The original dataset is very sparse with an average of 4.26 non-zero features per instance.

Using a random forest of 150 trees and a simple *modulo* hash function, we generated 4293 new features, i.e., the dimension of the induced feature space is the same as the dimension of the original one. Fig. 8 (bottom) shows the overlapping class distributions in a 2D plot obtained via random projections. When data is represented in the induced feature space instead (Fig. 8 top), a clearer multimodal distribution structure is visible with peaks associated to the different classes.

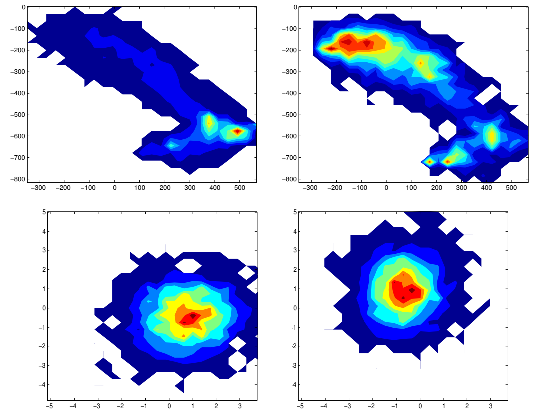

Figure 8. Data density plots for the PPI dataset. In the first (second) row, the transformed (original) feature space. Left (right) column instances for the positive (negative) class.

Finally we compare the predictive performance of a Gaussian SVM over the induced representation against the best results reported by Chen et al. [28]. They use a variant of random forests specifically targeted to the nature of PPI data to predict interactions. Both the feature subsampling and the voting process are altered to reflect the sparsity of the data. Using a forest of 150 trees, they report specificity at a sensitivity level of 0.7978, and compare the result to a maximum likelihood estimation method [29]. We employed a standard random forest (with the same number of trees as in [28]) to induce the features, without special tuning or optimizations to deal with sparsity. We used 30% of the training set as a validation set to optimize the C parameter of the SVM (see Sect. III-C4), and obtained slightly higher specificity at the same sensitivity level: 0.6580 against 0.6438 for Chen et al. and 0.3753 for Deng et al. [29].

IV. CONCLUSIONS

We have introduced a simple non-parametric technique that uses the structure of an ensemble of random decision trees to derive a task dependent feature encoding. We have empirically investigated the encoding characteristics; quality measures, such as the empirical target alignment and classification performance of kNN and SVM classifiers built on top of such representations, suggest that the proposed procedure is effective in integrating information from a given specific task. We showed that in this way learners based on the induced features can successfully exploit more information and achieve better predictive performance. We believe that the flexibility and efficiency of the proposed approach represents a distinguishing positive quality. Directions for future work include the extension to relational data using relational random forest learners and the study of different types of learning settings (e.g. ranking, transfer learning) to inform the feature construction process.

Acknowledgment

Celine Vens is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO). Fabrizio Costa is funded by the DFG Priority Program SPP 1395 InKoMBio (Grant BA 2168/4-1 to Rolf Backofen).

References

[1] K. Tsuda, G. Ratsch, and M. Warmuth, "Matrix exponentiated gradient updates for on-line learning and bregman projection," *Jrnl of Machine Learning Research*, vol. 5, pp. 1–48, 2004.

[2] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. Jordan, "Learning the kernel matrix with semidefinite programming," *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[3] N. Cristianini, J. Kandola, and A. Elisseeff, "On optimizing kernel alignment," *Journal of machine learning research*, pp. 1532–4435, Jan 2003.

[4] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, Jan 2009.

[5] J. Davis, B. Kulis, P. Jain, and S. Sra, "Information-theoretic metric learning," *Intl conference on Machine learning*, 2007.

[6] S. Shalev-Shwartz, Y. Singer, and A. Ng, "Online and batch learning of pseudo-metrics," *International conference on Machine learning*, Jan 2004.

[7] C. Ong, A. Smola, and R. Williamson, "Hyperkernels," *Advances in Neural Information Processing Systems*, pp. 495–504, 2003.

[8] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, Y. Lechevallier and G. Saporta, Eds. Paris, France: Springer, August 2010, pp. 177–187.

[9] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," in *International Conference on Machine Learning*, 2007.

[10] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Foundations of Computer Science*, pp. 459–468, 2006.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] ——, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, pp. 1545–1588, 1997.

[14] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *International Conference on Machine Learning*, 1998, pp. 55–63.

[15] J. Struyf and S. Džeroski, "Clustering trees with instance level constraints," in *Proceedings of the 18th European Conference on Machine Learning*, ser. Lecture Notes in Computer Science, vol. 4701. Springer, 2007, pp. 359–370.

[16] D. Demšar, S. Džeroski, T. Larsen, J. Struyf, J. Axelsen, M. Bruus Pedersen, and P. Henning Krogh, "Using multi-objective classification to model communities of soil microarthropods," *Ecological Modelling*, vol. 191, no. 1, pp. 131–143, 2006.

[17] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, pp. 185–214, 2008.

[18] L. Breiman, "RFtools–two-eyed algorithms, Invited talk at SIAM International Conference on Data Mining," 2003. [Online]. Available: http://oz.berkeley.edu/users/breiman/siamtalk2003.pdf

[19] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Technical Report*, Jan 2006.

[20] C. M. Ranzato, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," *Advances in neural information processing systems*, vol. 19, 2006.

[21] Y. Teh, M. Welling, S. Osindero, and G. Hinton, "Energy-based models for sparse overcomplete representations," *The Journal of Machine Learning Research*, vol. 4, pp. 1235–1260, 2003.

[22] R. Jin and H. Liu, "Robust feature induction for support vector machines," in *International Conference on Machine Learning (ICML)*, May 2004, pp. 1–8.

[23] K. Torkkola and E. Tuv, "Ensemble learning with supervised kernels," in *Proceedings of the 16th European Conference on Machine Learning (ECML)*, 2005, pp. 400–411.

[24] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[25] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, L. R. O. Maimon, Ed. Springer, 2010, pp. 222–243.

[26] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recogn.*, vol. 40, no. 7, pp. 2038–2048, 2007.

[27] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," *ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, 2001.

[28] X. Chen and M. Liu, "Prediction of protein–protein interactions using random decision forest framework," *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, 2005.

[29] M. Deng, S. Mehta, F. Sun, and T. Chen, "Inferring domain-domain interactions from protein-protein interactions," *Genome Research*, vol. 12, pp. 1540–1548, 2002.