Feature ranking for multi-label classification using predictive clustering trees

Dragi Kocev, Ivica Slavkov, and Sašo Džeroski

Department of Knowledge Technologies, Jožef Stefan Institute,
Jamova 39, 1000 Ljubljana, Slovenia
{Dragi.Kocev,Ivica.Slavkov,Saso.Dzeroski}@ijs.si

Abstract. In this work, we present a feature ranking method for multi-label data. The method is motivated by the the practically relevant multi-label applications, such as semantic annotation of images and videos, functional genomics, music and text categorization etc. We propose a feature ranking method based on random forests. Considering the success of the feature ranking using random forest in the tasks of classification and regression, we extend this method for multi-label classification. We use predictive clustering trees for multi-label classification as base predictive models for the random forest ensemble. We evaluate the proposed method on benchmark datasets for multi-label classification. The evaluation of the proposed method shows that it produces valid feature rankings and that can be successfully used for performing dimensionality reduction.

Key words: multi-label classification, feature ranking, random forest, predictive clustering trees

1 Introduction

The problem of single-label classification is concerned with learning from examples, where each example is associated with a single label λ_i from a finite set of disjoint labels $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_Q\}, i = 1..Q, Q > 1$. For $Q > 2$, the learning problem is referred to as *multi-class classification*. On the other hand, the task of learning a mapping from an example $\mathbf{x} \in \mathcal{X}$ ($\mathcal{X}$ denotes the domain of examples) to a set of labels $\mathcal{Y} \subseteq \mathcal{L}$ is referred to as a *multi-label classification* (MLC). In contrast to multi-class classification, alternatives in multi-label classification are not assumed to be mutually exclusive: multiple labels may be associated with a single example, i.e., each example can be a member of more than one class. Labels in the set $\mathcal{Y}$ are called relevant, while the labels in the set $\mathcal{L} \setminus \mathcal{Y}$ are irrelevant for a given example.

Many different methods have been developed for solving MLC problems. Tsoumakas et al. [16] summarize them into two main categories: a) algorithm adaptation methods, and b) problem transformation methods. Algorithm adaptation methods extend specific learning algorithms to handle multi-label data directly. Problem transformation methods, on the other hand, transform the MLC

problem into one or more single-label classification problems. The single-label classification problems are solved with a commonly used single-label classification method and the output is transformed back into a multi-label representation.

The issue of learning from multi-label data has recently attracted significant attention from many researchers, motivated by an increasing number of new applications. The latter include semantic annotation of images and videos (news clips, movies clips), functional genomics (gene and protein function), music categorization into emotions, text classification (news articles, web pages, patents, emails, bookmarks, ...), directed marketing and others.

Albeit the popularity of the task of MLC, the tasks of feature ranking and feature selection have not received much attention. The few available methods are based on the problem transformation paradigm [14], thus they do not fully exploit the possible label dependencies. More specifically, these methods use the label powerset (LP) approach for MLC [16, 6] from the group of problem transformation methods. The basis of the LP methods is to combine entire label sets into atomic (single) labels to form a single-label problem (i.e., single-class classification problem). For the single-label problem, the set of possible single labels represents all distinct label subsets from the original multi-label representation. In this way, LP based methods directly take into account the label correlations. However, the space of possible label subsets can be very large. To resolve this issue, Read [11] has developed a pruned problem transformation (PPT) method, that selects only the transformed labels that occur more than a predefined number of times. Tsoumakas et al. [16] use the LP transformed dataset to calculate simple χ^2 statistic thus producing a ranking of the features. Doquire and Verleysen [6] use the PPT transformed dataset to calculate mutual information (MI) for performing feature selection and they show that this method outperforms the χ^2-based feature ranking.

Feature ranking for MLC with problem transformation has two major shortcomings. First, the label dependencies and interconnections are not fully exploited. Second, these methods are not scalable to domains with large number of labels because of the exponential growth of the possible label powersets. Furthermore, the label powerset methods can yield a multi-class problem with extremely skewed class distribution. To address these issues, we propose an algorithm adaptation method for performing feature ranking. We extend the random forest feature ranking method [3] to the task of MLC. More specifically, we construct random forest that employs predictive clustering trees (PCTs) for MLC as base predictive models [1], [8]. PCTs can be considered a generalization of decision trees that are able to make predictions for structured outputs.

This work is motivated by several factors. First, the number of possible application domains for MLC and the size of the problems is increasing. For example, in image annotation the number of available images and possible labels is growing rapidly; in functional genomics the measurement techniques have improved significantly and there are high-dimensional genomic data available for analysis. Second, in Madjarov et al. [10] we have shown that the random forests of

PCTs for MLC are among the best predictive models for the task of MLC. Next, random forests as feature ranking algorithms are very successful on simple classification tasks [17, 19]. Finally, in Kocev et al. [8] we have shown that the random forests of PCTs are among the most efficient methods for predicting structured outputs. This is especially important, since many of the methods for MLC are computationally expensive and thus are not able to produce a predictive model for a given domain in a reasonable time (i.e., few weeks) [10].

We evaluate the proposed method on 4 benchmark datasets for MLC using 7 different evaluation measures. We compare the feature ranking produced by the proposed method to a random feature ranking. The random feature ranking is the worst feature ranking thus if the proposed method is able to capture the feature relevances then it should outperform the random ranking. We assess the performance of the obtained rankings and the random rankings by using error testing curves [12]. The goal of this study is to investigate whether random forests of PCTs for MLC can produce good feature rankings for the task of multi-label classification. Moreover, we want to check whether the produced rankings can be used to reduce the dimensionality of the considered multi-label domains.

The remainder of this paper is organized as follows. Section 2 presents the predictive clustering trees. The method for feature ranking using random forests is described in Section 3. Section 4 outlines the experimental design, while Section 5 presents the results from the experimental evaluation. Finally, the conclusions and a summary are given in Section 6.

2 Predictive clustering trees for multi-label classification

Predictive clustering trees (PCTs) [1] generalize decision trees [4] and can be used for a variety of learning tasks, including different types of prediction and clustering. The PCT framework views a decision tree as a hierarchy of clusters: the top-node of a PCT corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's prototype (prediction).

PCTs can be induced with a standard *top-down induction of decision trees* (TDIDT) algorithm [4]. The algorithm is presented in Table 1. It takes as input a set of examples (E) and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction in variance caused by partitioning ($\mathcal{P}$) the instances (see line 4 of BestTest procedure in Table 1). By maximizing the variance reduction the cluster homogeneity is maximized and it improves the predictive performance. If no acceptable test can be found (see line 6), that is, if the test does not significantly reduces the variance, then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf.

The main difference between the algorithm for learning PCTs and other algorithms for learning decision trees is that the former considers the variance function and the prototype function (that computes a label for each leaf) as parameters that can be instantiated for a given learning task. So far, PCTs have

Table 1. The top-down induction algorithm for PCTs.

procedure PCT(E) **returns** tree	**procedure** BestTest(E)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$	1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **if** $t^* \neq none$ **then**	2: **for each** possible test t **do**
3: **for each** $E_i \in \mathcal{P}^*$ **do**	3: $\mathcal{P} =$ partition induced by t on E
4: $tree_i = \text{PCT}(E_i)$	4: $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{\lvert E_i \rvert}{\lvert E \rvert} Var(E_i)$
5: **return** node$(t^*, \bigcup_i \{tree_i\})$	5: **if** $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ **then**
6: **else**	6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **return** leaf(Prototype(E))	7: **return** $(t^*, h^*, \mathcal{P}^*)$

been instantiated for the following tasks: prediction of multiple targets [8], [15], prediction of time-series [13] and hierarchical multi-label classification [18].

One of the most important steps in the induction algorithm is the test selection procedure. For each node, a test is selected by using a heuristic function computed on the training examples. The goal of the heuristic is to guide the algorithm towards small trees with good predictive performance. The heuristic used in this algorithm for selecting the attribute tests in the internal nodes is the reduction in variance caused by partitioning the instances. Maximizing the variance reduction maximizes cluster homogeneity and improves predictive performance.

In this work, we focus on the task of multi-label classification, which can be considered as a special case of multi-target prediction. Namely, in the task of multi-target prediction, the goal is to make predictions for multiple target variables. The multiple labels in MLC can be viewed as multiple binary variables, each one specifying whether a given example is labelled with the corresponding label. Therefore, we compute the variance function same as for the task of predicting multiple discrete variables [8], i.e., the variance function is computed as the sum of the *Gini indices* [4] of the variables from the target tuple, i.e., $Var(E) = \sum_{i=1}^{T} Gini(E, Y_i)$, $Gini(E, Y_i) = 1 - \sum_{j=1}^{C_i} p_{c_{ij}}$, where T is the number of target attributes, c_{ij} is the j-th class of target attribute Y_i and C_i is the number of classes of target attribute Y_i. The prototype function returns a vector of probabilities for the set of labels that indicate whether an example is labelled with a given label. For a detailed description of PCTs for multi-target prediction the reader is referred to [1, 8]. The PCT framework is implemented in the CLUS system[1].

3 Feature ranking via random forests

We construct the random forest using predictive clustering trees as base classifiers. We exploit the random forests mechanism [3] to calculate the variable importance, i.e., the feature ranking. In the following subsections, first we present the random forest algorithm and then we describe how it can be used to estimate the importance of the descriptive variables.

[1] CLUS is available for download at `http://clus.sourceforge.net`

3.1 Random Forests

An ensemble is a set of classifiers constructed with a given algorithm. Each new example is classified by combining the predictions of every classifier from the ensemble. These predictions can be combined by taking the average (for regression tasks) and the majority or probability distribution vote (for classification tasks)[2], or by taking more complex combinations [9].

A necessary condition for an ensemble to be more accurate than any of its individual members, is that the classifiers are accurate and diverse [7]. An accurate classifier does better than random guessing on new examples. Two classifiers are diverse if they make different errors on new examples. There are several ways to introduce diversity: by manipulating the training set (by changing the weight of the examples [2] or by changing the attribute values of the examples [3]), or by manipulating the learning algorithm itself [5].

A random forest [3] is an ensemble of trees, where diversity among the predictive models is obtained by using bootstrap replicates, and additionally by changing the feature set during learning. More precisely, at each node in the decision trees, a random subset of the input attributes is taken, and the best feature is selected from this subset. The number of attributes that are retained is given by a function f of the total number of input attributes x (e.g., $f(x) = 1$, $f(x) = \lfloor \sqrt{x} + 1 \rfloor$, $f(x) = \lfloor log_2(x) + 1 \rfloor \ldots$). By setting $f(x) = x$, we obtain the bagging procedure.

3.2 Feature ranking using random forests

Feature ranking of the descriptive variables can be obtained by exploiting the mechanism of random forests. This method uses the internal out-of-bag estimates of the error and noising the descriptive variables. To create each tree from the forest, the algorithm first creates a bootstrap replicate (line 4, from the $Induce\_RF$ procedure, Table 2). The samples that are not selected for the bootstrap are called out-of-bag (OOB) samples (line 7, procedure $Induce\_RF$). These samples are used to evaluate the performance of each tree from the forest. The complete algorithm is presented in Table 2.

Suppose that there are T target variables and D descriptive variables. The trees in the forest are constructed using a randomized variant of the PCT construction algorithm (PCT_{rand}), i.e., in each node the split is selected from a subset of the descriptive variables. After each tree from the forest is built, the values of the descriptive attributes for the OOB samples are randomly permuted one-by-one thus obtaining D OOB samples (line 3, procedure $Update\_Imp$). The predictive performance of each tree is evaluated on the original OOB data ($Err(OOB_i)$) and the permuted versions of the OOB data ($Err_i(f_d)$). The performance is averaged across the T target variables. Then the importance of a given variable (I_j) is calculated as the relative increase of the mis-classification error that is obtained when its values are randomly permuted. The importance is at the end averaged over all trees in the forest. The variable importance is calculated using the following formula:

Table 2. The algorithm for feature ranking via random forests. E is the set of the training examples, k is the number of trees in the forest, and $f(x)$ is the size of the feature subset that is considered at each node during tree construction.

procedure Induce_RF($E, k, f(x)$)
returns Forest, Importances
1: $F = \emptyset$
2: $I = \emptyset$
3: **for** $i = 1$ **to** k **do**
4: E_i = Bootstrap_sample(E)
5: $Tree_i = PCT_{rand}(E_i, f(x))$
6: $F = F \bigcup Tree_i$
7: $E_{OOB} = E \setminus E_i$
8: Update_Imp($E_{OOB}, Tree, I$)
9: I = Average(I, k)
10: **return** F, I

procedure Update_Imp($E_{OOB}, Tree, I$)
1: Err_{OOB} = Evaluate($Tree, E_{OOB}$)
2: **for** $j = 1$ **to** D **do**
3: E_j = Randomize(E_{OOB}, j)
4: Err_j = Evaluate($Tree, E_j$)
5: $I_j = I_j + (Err_j - Err_{OOB})/Err_{OOB}$
6: **return**

procedure Average(I, k)
1: $I^T = \emptyset$
2: **for** $l = 1$ **to** $size(I)$ **do**
3: $I_l^T = I_l/k$
4: **return** I^T

$$Importance(f_d) = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{Err_i(f_d) - Err(OOB_k)}{Err(OOB_k)} \tag{1}$$

where k is the number of bootstrap replicates and $0 < d \leq D$.

4 Experimental design

In this section, we give the specific experimental design used to evaluate the performance of the proposed method. We begin by briefly summarizing the multi-label datasets used in this study. Next, we present the evaluation measures and discuss the construction of the error curves. Finally, we give the specific parameter instantiation of the methods.

4.1 Data description

We use four multi-label classification benchmark problems. Parts of the selected problems were used in various studies and evaluations of methods for multi-label learning. Table 3 presents the basic statistics of the datasets. We can note that the datasets vary in size: from 391 up to 4880 training examples, from 202 up to 2515 testing examples, from 72 up to 1836 features, from 6 to 159 labels, and from 1.25 to 3.38 average number of labels per example (i.e., label cardinality [16]). From the literature, these datasets come pre-divided into training and testing parts: Thus, in the experiments, we use them in their original format. The training part usually comprises around 2/3 of the complete dataset, while the testing part the remaining 1/3 of the dataset.

The datasets come from the domains of multimedia and text categorization. *Emotions* is a dataset from the multimedia domain where each instance is a piece

Table 3. Description of the benchmark problems in terms of number of training ($\#tr.e.$) and test ($\#t.e.$) examples, the number of features (D), the total number of labels (Q) and label cardinality (l_c). The problems are ordered by their overall complexity roughly calculated as $\#tr.e. \times D \times Q$.

	$\#tr.e.$	$\#t.e.$	D	Q	l_c
emotions	391	202	72	6	1.87
medical	645	333	1449	45	1.25
enron	1123	579	1001	53	3.38
bibtex	4880	2515	1836	159	2.40

of music. Each piece of music can be labelled with six emotions: sad-lonely, angry-aggressive, amazed-surprised, relaxing-calm, quiet-still, and happy-pleased. The domain of text categorization is represented with 3 datasets: *medical*, *enron* and *bibtex*. *Medical* is a dataset used in the Medical Natural Language Processing Challenge[2] in 2007. Each instance is a document that contains brief free-text summary of a patient symptom history. The goal is to annotate each document with the probable diseases from the International Classification of Diseases. *Enron* is a dataset that contains the e-mails from 150 senior Enron officials categorized into several categories. The labels can be further grouped into four categories: coarse genre, included/forwarded information, primary topics, and messages with emotional tone. *Bibtex* contains metadata for bibtex items, such as the title of the paper, the authors, book title, journal volume, publisher, etc. These datasets are available for download at the web page of the Mulan project[3].

4.2 Experimental setup

We evaluate the proposed method using seven evaluation measures: *accuracy*, *micro precision*, *micro recall*, *micro* F_1, *macro precision*, *macro recall*, and *macro* F_1. These measures are typically used for evaluation of the performance of multi-label classification methods. The micro averaging implicitly includes information about the label frequency, while macro averaging treats all the labels equally. Due to the space limitations, we only show the results for *micro* F_1 because the F_1 score unites the values for *precision* and *recall*. Moreover, the results and the discussion are similar if the other measures were used. These measures are discussed in detail in [10] and [16].

We asses the performance of the proposed method using *error curves* [12]. The error curves are based on the idea that the 'correctness' of the feature rank is related to predictive accuracy. A good ranking algorithm would put on top of a list a feature that is most important, and at the bottom a feature that is least important w.r.t. some target concept. All the other features would be in-between, ordered by decreasing importance. By following this intuition, we

[2] http://www.computationalmedicine.org/challenge/
[3] http://mulan.sourceforge.net/datasets.html

evaluate the ranking by performing a stepwise feature subset evaluation, which is used for obtaining an error curve.

We generate two types of curves: *forward feature addition* (FFA) and *reverse feature addition* (RFA) curve. Examples of these curves are shown in Figures 1, 2, 3, and 4. The FFA curve is constructed from the top-k ranked features, i.e., from the beginning of the ranking. In contrast, the RFA curve is constructed from the bottom-k ranked features. For the FFA curves, we can expect that as the number of features used to construct the predictive model increases, the accuracy of the predictive models also increases. This can be interpreted as follows: By adding more and more of the top-ranked features, the feature subsets constructed contain more relevant features, reflected in the improvement of the error measure.

On the other hand, for the RFA curves, we can expect a slight difference at the beginning of the curve, which considering the previous discussion, is located at the end of the x-axis. Namely, the accuracy of the models constructed with the bottom ranked features is minimal, which means the ranking is correct in the sense that it puts irrelevant features at the bottom of the ranking. As the number of bottom-k ranked features used to construct the predictive model increases, some relevant features get included and the accuracy of the models increases. In summary, at each point k, the FFA curve gives us the error of the predictive models constructed with the top-k ranked features, while the RFA curve gives us the error of the bottom-k ranked features. The algorithm for constructing the curves is given in Table 4.

Table 4. The algorithm for generating forward feature addition (FFA) and reverse feature addition (RFA) curves.$\mathcal{R} = \{F_{r1}, \ldots, F_{rn}\}$ is the feature ranking and F_t is the target feature.

procedure ConstructErrorCurve($\mathcal{R}, F_t$)
returns Error Curve Err

$\mathcal{R}_S \Leftarrow \emptyset$
for $i = 1$ to n **do**
  $\mathcal{R}_S \Leftarrow \mathcal{R}_S \cup feature(\mathcal{R}, i)$
  $Err[i] = Err(\mathcal{M}(\mathcal{R}_S, F_t))$
return Err

for FFA curve:
$feature(\mathcal{R}, i) = \{F_{ri}\}$
for RFA curve:
$feature(\mathcal{R}, i) = \{F_{r(n-i+1)}\}$

We compare the performance of the proposed method to the performance of a random ranking. We base this comparison on the idea that the random ranking is the worst ranking possible [12]. This is similar to the notion of random predictive model in predictive modelling. If our algorithm indeed is able to capture the variable importance correctly, then its error curves should be better than the curves of a random ranking. In this work, we generate 100 random feature rankings for each dataset and we show the averaged error curves. We opted for the comparison with the random rankings instead with the methods presented in [6]

and [16] because of the un-stable results produced by these rankings (especially for the Emotions dataset). Moreover, the reported accuracies for the emotions dataset is in the range of 0.3 to 0.5, and in our experiments the accuracy for the emotions dataset is in the range of 0.6 to 0.8.

4.3 Parameter instantiation

The feature ranking algorithm with random forests of PCTs for MLC take as input two parameters: the number of base predictive models in the forest and the feature subset size. In these experiments, we constructed the feature rankings using a random forest with 500 PCTs for MLC. Each node in a PCT was constructed by randomly selecting 10% of the features (as suggested in [8]).

For construction of the error curves, we selected random forests of PCTs for MLC as predictive models. The random forests model in this case consists of 100 PCTs for MLC and each node was constructed using 10% of the features. Both the predictive models and the feature rankings were constructed on the training set, while the performance for the error curves is the one obtained on the testing set.

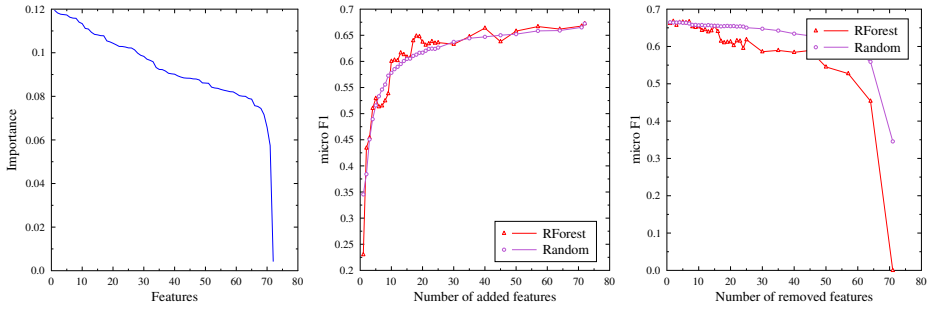
5 Results and discussion

In this section, we present the results from the experimental evaluation of the proposed method. We explain the results with respect to the variable importance scores for the features, the FFA curves and the RFA curves. The FFA and RFA curves are constructed using the *micro* F_1, however, the conclusions are still valid if we consider the other evaluation measures. In the remainder, we discuss the results for each of the datasets considered in this study.

The results for the *Emotions* dataset are given in Figure 1. They show that the obtained ranking performs slightly better than the random ranking. Both FFA curves increase with a similar rate and have a similar shape. However, on a larger part the FFA curve of the ranking is above the curve of the random ranking. The RFA curve shows that the obtained ranking places more non-relevant features at the bottom of the ranking.

This finding can be confirmed and explained with the variable scores (Figure 1(a)). Namely, the curve with the variable scores is somewhat parallel to the x-axis. This means that the majority of features in this dataset are approximately equally relevant for the target concept (i.e., the multiple labels). Moreover, this could indicate that there are redundant features that are present in the dataset. All in all, selecting randomly a feature subset with a reasonable size (e.g., 25-30 features) is good enough to produce a predictive model with satisfactory predictive performance (i.e., the dimensionality can be easily reduced without a significant loss of information).

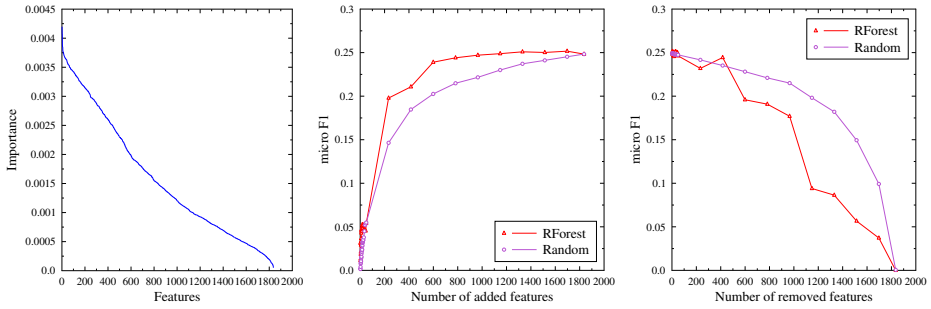
The results for the *Bibtex* (Figure 2) and *Enron* (Figure 3) datasets are somewhat similar to each other, thus we discuss them together. We can see from the figures that the obtained ranking is clearly better than the random



(a) (b) (c)

Fig. 1. The performance of random forests of PCTs for MLC feature ranking algorithm on the *Emotions* dataset. (a) feature importances reported by the ranking algorithm, (b) FFA curve and (c) RFA curve.

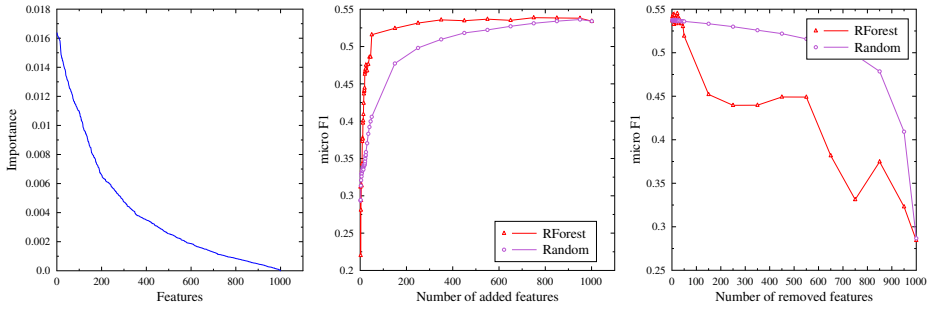
ranking. The FFA curve of the obtained ranking is always above the FFA curve of the random ranking, and, conversely, the RFA curve of the obtained ranking is always bellow the RFA curve of the random ranking. Hence, more relevant features are placed at the top and more non-relevant features are placed at the bottom.



(a) (b) (c)

Fig. 2. The performance of random forests of PCTs for MLC feature ranking algorithm on the *Bibtex* dataset. (a) feature importances reported by the ranking algorithm, (b) FFA curve and (c) RFA curve.

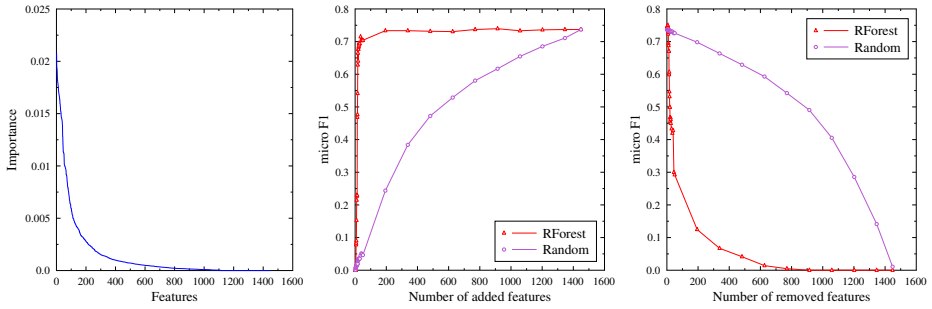
This is also confirmed with the variable importance scores. We can note that the curve of the variable importances drops linearly, which means that there are multiple features in the dataset that are more relevant for the target concept than the remaining features. The dimensionality in these cases can be significantly reduced. We will still obtain very good predictive performance if we select the 500 top-ranked features (out of 1836) for the *Bibtex* dataset and the 50 top-ranked features (out of 1001) for the *Enron* dataset.



(a) (b) (c)

Fig. 3. The performance of random forests of PCTs for MLC feature ranking algorithm on the *Enron* dataset. (a) feature importances reported by the ranking algorithm, (b) FFA curve and (c) RFA curve.

Finally, we discuss the results for the *Medical* dataset (given in Figure 4). We can note that the obtained ranking is significantly better than the random ranking. The FFA and RFA curves of the obtained ranking exhibit very steep increase and decrease, respectively. On the other hand, the FFA and RFA curves of the random ranking have linear increase and decrease. This means that there are a few features that are very relevant for the target concept and that these features carry the majority of the information for the target concept. This is further confirmed with the curve of the variable importances: this curve descends exponentially. Considering all of this, we can drastically reduce the dimensionality of this dataset. The good predictive performance will be preserved even if we select the 35 top-ranked features (out of 1449).



(a) (b) (c)

Fig. 4. The performance of random forests of PCTs for MLC feature ranking algorithm on the *Medical* dataset. (a) feature importances reported by the ranking algorithm, (b) FFA curve and (c) RFA curve.

6 Conclusions

In this work, we presented and evaluated a feature ranking method for the task of multi-label classification (MLC). The proposed method is based on the random forest feature ranking mechanism. The random forests are already proven as a good method for feature ranking on the simpler tasks of classification and regression. Here, we propose and extension of the method to the task of MLC. To this end, we use predictive clustering trees (PCTs) for MLC as base predictive models. The random forests of PCTs have state-of-the-art predictive performance for the task of MLC. Here, we investigate whether this method can be also successful for the task of feature ranking for MLC.

We evaluated the method on 4 benchmark multi-label datasets using 7 evaluation measures. The quality of the feature ranking was assessed by using *forward feature addition* and *reverse feature addition* curves. To investigate whether the obtained feature ranking is valid, i.e., that it places the more relevant features closer to the top of the ranking and the non-relevant features closer to the bottom of the ranking, we compare it to the performance of a random feature ranking.

We summarize the results as follows. First, we show that in a datasets where many of the features are relevant for the target concept the produced ranking can slightly outperform the random ranking. This is due to the fact that if several features are (randomly) selected then the predictive model will have satisfactory predictive performance. Next, in the datasets where there are several relevant features for the target concept the produced ranking clearly outperforms the random ranking. This means that the ranking algorithm is able to detect these features and place them at the top of the ranking. Furthermore, in the datasets where there are only few features of high relevance for the target concept, the obtained ranking drastically outperforms the random ranking and satisfactory predictive performance can be obtained by using only $2-3\%$ of the features. All in all, the experimental evaluation demonstrates that the random forests feature ranking method can be successfully applied to the task of MLC.

We plan to extend this work in future along three major dimensions. First, we plan to include other measures for predictive performance in the ranking algorithm. In the current version, we use mis-classification rate. However, we will consider MLC specific evaluation measures. Next, we will extend the proposed method for other structured output prediction tasks, such as multi-target regression and hierarchical multi-label classification. Finally, we could estimate the relevance of a feature by considering the reduction of the variance the feature causes when selected for a test in a given node.

References

1. Blockeel, H.: Top-down induction of first order logical decision trees. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1998)
2. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
3. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)

4. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
5. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Proc. of the 1st International Workshop on Multiple Classifier Systems - LNCS 1857. pp. 1–15. Springer (2000)
6. Doquire, G., Verleysen, M.: Feature Selection for Multi-label Classification Problems. In: Advances in Computational Intelligence. pp. 9–16 (2011)
7. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(10), 993–1001 (1990)
8. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition 46(3), 817–833 (2013)
9. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
10. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9), 3084–3104 (2012)
11. Read, J., Pfahringer, B., Holmes, G.: Multi-label Classification Using Ensembles of Pruned Sets. In: Proc. of the 8th IEEE International Conference on Data Mining. pp. 995–1000 (2008)
12. Slavkov, I.: An Evaluation Method for Feature Rankings. Ph.D. thesis, IPS Jožef Stefan, Ljubljana, Slovenia (2012)
13. Slavkov, I., Gjorgjioski, V., Struyf, J., Džeroski, S.: Finding explained groups of time-course gene expression profiles with predictive clustering trees. Molecular BioSystems 6(4), 729–740 (2010)
14. Spolaôr, N., Cherman, E.A., Monard, M.C., Lee, H.D.: A comparison of multi-label feature selection methods using the problem transformation approach. Electronic Notes in Theoretical Computer Science 292, 135 – 151 (2013)
15. Struyf, J., Džeroski, S.: Constraint Based Induction of Multi-Objective Regression Trees. In: Proc. of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933. pp. 222–233. Springer (2006)
16. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer Berlin / Heidelberg (2010)
17. Čehovin, L., Bosnić, Z.: Empirical evaluation of feature selection methods in classification. Intelligent Data Analysis 14(3), 265–281 (2010)
18. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
19. Verikas, A., Gelzinis, A., Bacauskiene, M.: Mining data with random forests: A survey and results of new tests. Pattern Recognition 44(2), 330–349 (2011)