IEICE Transactions on Information and Systems,
vol.E97-D, no.6, pp.1677–1681, 2014.

Tree-Based Ensemble Multi-Task Learning Method for Classification and Regression

Jaak Simm*
Tallinn University of Technology, Estonia

Ildefons Magrans de Abril*
Vrije Universiteit Brussel, Belgium

Masashi Sugiyama
Tokyo Institute of Technology, Japan.
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/˜sugi

Abstract

Multi-task learning is an important area of machine learning that tries to learn multiple tasks simultaneously to improve the accuracy of each individual task. We propose a new tree-based ensemble multi-task learning method for classification and regression (MT-ExtraTrees), based on Extremely Randomized Trees. MT-ExtraTrees is able to share data between tasks minimizing negative transfer while keeping the ability to learn non-linear solutions and to scale well to large datasets.

Keywords

machine learing, multi-task learning, tree-based methods, ensemble methods

1 Introduction

Often, related machine learning tasks are treated separately. The methods of *multi-task learning* (MTL) try to overcome this limitation by sharing data between these related tasks where possible. We are considering the same MTL setup as a previous work [1], where T supervised learning tasks share the same input-output space $X \times Y$ and task distributions $P_t(X, Y)$ are different but overlap among each other.

One of the first approaches for supervised MTL was proposed by Evgeniou and Pontil [1], where sharing between SVM tasks is achieved by pushing together SVM solutions through introducing a multi-task penalty. This multi-task penalty approach has been extended to logistic regression [2] and to least-squares probabilistic classifier [3] but they do

*Jaak Simm and Ildefons Magrans de Abril contributed equally as main authors.

not fit well for *large scale* use as their time complexity with dense kernels is $O(N^3)$ where N is the number of training samples. Another significant issue is that such multi-task penalty creates a uniform sharing between all tasks that can lead to reduced accuracy if $P_t(Y|X)$ of different tasks are too dissimilar. Such a situation is called *negative transfer*.

An approach to solve the *negative transfer* problem is to perform task clustering. For example, Xue et al. [4] use Dirichlet processes to cluster logistic regression tasks. The idea behind the clustering approach is to only share between tasks that are in the same cluster, thus avoiding sharing between too dissimilar tasks. However, this comes at the cost of computational speed as EM-based approaches and Bayesian sampling methods are even more computationally extensive than the ones mentioned earlier. Compared to task clustering approaches like the work by Xue et al. [4], tree-based approaches are more flexible as they allow us to share between tasks in certain areas of input space X but not in others.

To propose an MTL version of decision trees there are two approaches. First one is to modify the splitting criterion, to include the effects of multiple tasks. This approach has been taken by Multi-task Adaboost [5] and by multi-label boosting implemented in Multiboost [6]. This approach implies that all tasks share the same tree structure, which may not be optimal from the point of view of each individual task. A second approach that we propose consists of growing multi-task decision trees with specific branches dedicated to subsets of tasks. A suitable implementation of this idea should minimize the *negative transfer* because the decision in each node has been learned only from samples of a task subset with similar $P_t(Y|X)$, instead of finding a decision which minimizes the penalty over all tasks.

In this paper we propose an MTL extension to a binary decision tree based ensemble method called *Extremely Randomized Trees* (ExtraTrees) [7]. The initial motivation for using ExtraTrees as a base method is its ability to learn non-linear solutions and to scale well to large datasets (up to 100,000 training samples or even millions of samples if parallelized over many computation servers). Our implementation is freely available. To the best of our knowledge, it is the first multi-task binary classification and regression package in R[1].

2 ExtraTrees

ExtraTrees [7] is an ensemble learning method that builds upon a large number of binary decision trees (e.g., $N_{tree} = 500$). All trees are built independently of each other, thus, it is straightforward to parallelize to multiple computing CPUs or servers. We assume a D-dimensional input space, $\mathbf{x} \in \mathbb{R}^D$, and either binary, $y \in \{-1, 1\}$, or real valued outputs, $y \in \mathbb{R}$. Let $\{(\mathbf{x}_n, y_n)\}_{n \in I}$ be the training data in a node of one of the trees (I is the set of sample indexes of that node), then the tree building algorithm proceeds as:

1. Randomly chooses K input dimensions. For each selected dimension d, it chooses a random binary splitting value c, denoting data points $n \in I$ that have $x_{n,d} < c$ by

[1]URL to R package: `http://cran.r-project.org/web/packages/extraTrees/`

L (left) and those with $x_{n,d} \geq c$ by R (right). This step is the only difference with Random Forest [8] which uses an optimal splitting criteria for value c. Therefore, our proposed multi-task method could be also implemented in Random Forest.

2. It computes the score for each dimension d and its splitting value as follows:

$$s_d = |L| f_{score}(y_L) + |R| f_{score}(y_R), \tag{1}$$

where $|L|$ ($|R|$) is the number of data points assigned to left (right) branch, y_L (y_R) denotes the y values in the left (right) branch. For binary classification function f_{score} is Gini index calculated as

$$f_{score}(\mathbf{y}) = 1 - (p_{-1}^2 + p_1^2), \tag{2}$$

where p_{-1} is the proportion of samples with $y = -1$ and respectively, p_1 is the proportion of $y = 1$. For regression, f_{score} is negative of variance:

$$f_{score}(\mathbf{y}) = -\frac{1}{n} \sum_{i=1}^{n} (y_i - \text{mean}(\mathbf{y}))^2, \tag{3}$$

where n is the size of $\mathbf{y}$ and $\text{mean}(\mathbf{y}) = \dfrac{1}{n} \sum y_i$.

3. It chooses the dimension with the highest score s_d, stores its split in the node and recursively calls these 3 steps on the two subtrees made out of L and R, respectively. ExtraTrees recursively builds every tree until a minimum node size (N_{leaf}) is reached. Then, a leaf node is built and it stores the most frequent value of outputs in the case of classification and the average value in the case of regression.

Training of a tree takes $O(N \log N)$ time and space where N is the number of samples. For prediction, all the trees have to be traversed until the leaf nodes, requiring $O(\log N)$ computation per tree.

3 MT-ExtraTrees

As explained in the introduction, we propose growing multi-task decision trees with specific branches dedicated to subsets of tasks. One way of achieving that goal is adding one new splitting criterion able to divide samples according to tasks. This new split will create two new branches and each branch will contain samples corresponding to separated task subsets.

A simple and straightforward way to incorporate task id information to ExtraTrees would be to add T binary variables to all training and testing samples such that the tth new variables is equal to 1 iff a sample belongs to task t. By doing so the ExtraTrees method can grow trees with branches dedicated to single tasks. This would certainly limit

negative transfer between the specific task and the rest. However, the task-specific branch would also lose the possibility to capture relevant information from other tasks.

To cope with this limitation, we propose growing multi-task decision trees with specific branches dedicated to subsets of tasks. One way of achieving that goal is adding one new splitting criterion able to divide samples according to tasks. This new split will create two new branches and each branch will contain samples corresponding to separated task subsets.

To accomplish the splitting by tasks, we modified the step 1 so that in addition to ordinary feature splits, the method can also split samples according to their task. A new parameter, $\lambda \in [0, 1]$, controls the probability of evaluating a task-wise split at the current decision node. The higher the value of λ is, the more likely that the method generates a larger number of sub-branches dedicated to particular subsets of tasks. More precisely, $\lambda = 0$ means that we will never attempt to create a task-wise split in our decision trees, and $\lambda = 1$ means that we will always consider whether the task-wise split is better option than K ordinary feature-wise splits. It's interesting to mention that when $\lambda = 0$, the new MT-ExtraTrees is equivalent to the original ExtraTrees with all data from all tasks pooled together. A nice capability of the method is that it simplifies the selection of a good λ because even when λ is very high, the method limits the excessive task- wise splitting as it always compares against other feature-wise splits. This will be shown in the evaluation section.

When a task wise split is evaluated, for each task t, the method computes a task feature ϕ_t. In the case of binary classification, ϕ_t measures the local sample probability $P_t(Y = 1|X)$. In the case of regression, ϕ_t measures the local sample expected value $\mathrm{E}_t[Y|X]$. In detail, let I denote the set of sample indexes at the node and I_t denote the set of sample indexes of the task t at the node. Task features for classification are calculated by

$$\phi_t = \frac{N_{t,y=1} + \alpha\gamma^{class}}{|I_t| + \alpha}, \tag{4}$$

and for regression by

$$\phi_t = \frac{\sum_{n \in I_t} y_n + \alpha\gamma^{regr}}{|I_t| + \alpha}, \tag{5}$$

where α is the regularization parameter (with default value 1), $N_{t,y=1}$ is the number of samples of class 1 in task t at the current node and γ^{regr} and γ^{class} are multi-task priors that average data of all tasks in the current node:

$$\gamma^{class} = \frac{1}{|I|} \sum_t N_{t,y=1}, \tag{6}$$

$$\gamma^{regr} = \frac{1}{|I|} \sum_{n \in I} y_n. \tag{7}$$

After calculating all ϕ_t the method chooses a random value $c \in (\min_t \phi_t, \max_t \phi_t)$ and splits the samples according to the task split defined by c. After that, MT-ExtraTrees follows steps 2 and 3 as in ExtraTrees (see Section 2).

4 Experimental results

We run two experiments on well-known multi-task learning benchmark datasets to show the performance and behavior of our method in both binary classification and regression settings.

Landmine detection problem: The main objective of this subsection is to evaluate the performance of the proposed MT-ExtraTrees in the setting of multi-task binary classification. MT-ExtraTrees is compared to basic ExtraTrees, to ExtraTrees with added T binary variables representing the task ids and to Multi-task Adaboost using the Multiboost implementation [6]. The latter is a multi-label classification method by Schapire and Singer [9] encoded as a multi-task classification following the guidelines of the Multiboost reference paper [6].

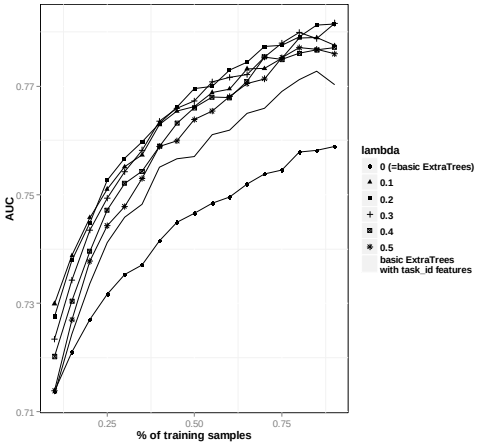
The Landmine detection problem dataset was collected from various landmine fields. It consists of 29 tasks and each task contains between 450 and 700 samples represented by 9 features. The feature vector was extracted from radar images and it is built concatenating four moment based features, three correlation-based features, one energy ratio feature and one spatial variance.[2]

The same percentage of samples from each task is used to train the classification models. The performance is measured by average AUC on 100 runs. Figure 1(a) shows the performance as a function of the training percentage for different models. The lower curve corresponds to the basic ExtraTrees model (=MT-ExtraTrees with $\lambda = 0$) where we pool together all samples from all tasks. The following curve drawn as a plain line is the performance curve corresponding to the basic ExtraTrees model again pooling together all samples from all tasks with added T binary variables representing the task ids. Finally, we have 5 additional curves always in top of the previous ones that show the performance of the MT-ExtraTrees model with different λs bigger than 0. We can observe how, for all λs, our multi-task model always outperforms the basic ExtraTrees models. It is interesting also to comment the experimental results obtained with the library multi-boost [6]. The performance was measured as the average AUC of the method on 10 runs using 75% of samples from all tasks for training the models. In this case, the average AUC is 0.743, which clearly underperforms our method and even the basic ExtraTrees models.

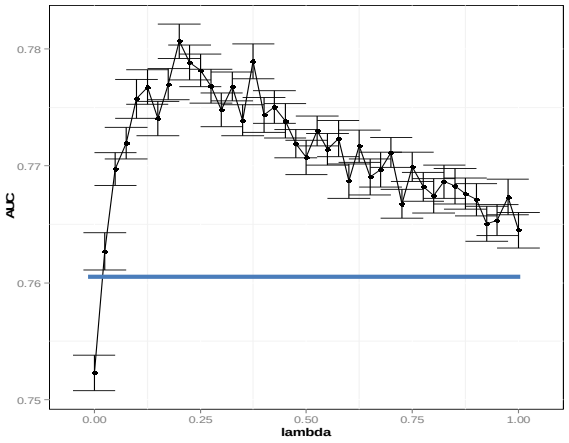
Figure 1(b) is a more detailed plot of the model behavior as a function λ. Each point of the curve corresponds to the average AUC on 100 runs using 75% of samples from all tasks for training the models. We can observe that for any λ between 0.125 and 0.6, the average AUC is bigger than 0.77 while the performance of the ExtraTrees with added T binary variables representing the task ids has an average performance just above 0.76. A large continuous range of suitable λs is an indication that our method can provide an enhanced performance with little additional model selection complexity.

School data from the Inner London Education: The main objective of this subsection is to evaluate the performance of the proposed MT-ExtraTrees in the setting of multi-task regression. MT-ExtraTrees is compared to basic ExtraTrees, to ExtraTrees with added T binary variables representing the task ids and to Regularized Multi-task

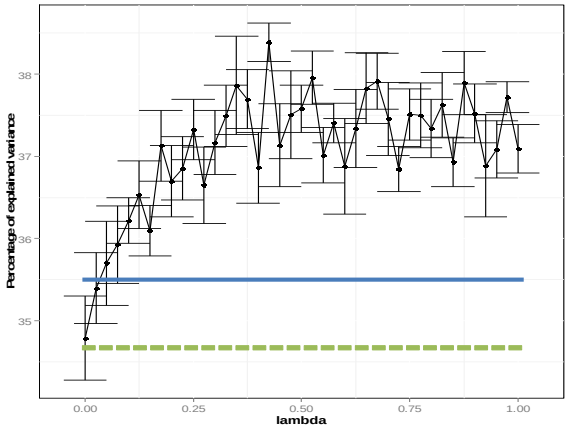
[2] `http://www.ee.duke.edu/~lcarin/LandmineData.zip`



(a) AUC as a function of training percentage for Landmine dataset.



(b) AUC as a function of λ for Landmine dataset. Solid bold line shows the performance of Extra-Trees with added T binary variables representing the task ids.



(c) Explained variance as function of λ for the School dataset. Solid bold line shows the performance of ExtraTrees with added T binary variables representing the task ids. Dash line shows the state-of-the art reference performance reported in [1].

Figure 1: Experimental results of MT-ExtraTrees.

method [1], which is state-of-the-art multi-task learning method for regression.

The School dataset consists of examination records of 15362 students from 139 secondary schools. Each record consists of 8 features and the corresponding examination score. The features are: year of the exam (1985, 1986 or 1987), school code (139), percentage of students eligible for free school meals, percentage of students in Verbal Reasoning (VR) band one, gender of the student, VR band of student, ethnic group of student, school gender and school denomination (Maintained, Church of England, Roman Catholic).[3] To compare our results with Evgeniou and Pontil [1], we used the same setup described in the paper [10]: we converted categorical features into binary vectors, we used 10 random splits of the data, we trained our models with 75% of all the samples from each school/task and we used the percentage of explained variance[4] as performance measure.

As a reference we computed the average performance using a basic ExtraTrees model pooling together all samples from all tasks/schools with added T binary variables representing the task ids, showing a final performance of 35.5%. Figure 1(c) shows the performance as a function λ. We can observe that for any λ bigger than 0.125, the multi-task model always outperforms the basic ExtraTrees model with a top performance over 38%. It is also a performance improvement with respect to the performance reported by Evgeniou and Pontil [1] of 34.37%.

As we discussed in Section 3 we can observe both in Figure 1(b) and in Figure 1(c) that even when the values of λ are high, MT-ExtraTrees outperforms basic ExtraTrees and state-of-the-art methods. This is an indication that our method can provide an enhanced performance with little additional model selection complexity.

5 Conclusions

We have introduced a new multi-task learning method MT-ExtraTrees based on a binary decision tree ensemble method. Our implementation is able to grow trees that contain branches uniquely dedicated to particular subsets of tasks with similar behavior. Consequently, this facilitates sharing among similar tasks while minimizing negative transfer. We have implemented a binary classification and regression open source library in R and evaluated it with well-known benchmark datasets for multi-task binary classification and regression. Results show an enhanced performance with respect to the base method and state-of-the-art methods.

Acknowledgments

Ildefons Magrans de Abril was supported by EU FP7 framework's Marie Curie Industry-Academia Partnerships and Pathways (IAPP) project SCANERGY, under grant agreement number 324321. Masashi Sugiyama was supported by KAKENHI 23300069 and AOARD.

[3] `http://www.bristol.ac.uk/cmm/learning/support/datasets/ilea567.zip`

[4] Total variance minus the sum-squared error on the test set as a percentage of the total data variance

References

[1] T. Evgeniou and M. Pontil, "Regularized multi–task learning," Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp.109–117, ACM, 2004.

[2] A. Lapedriza, D. Masip, and J. Vitrià, "A hierarchical approach for multi-task logistic regression," Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part II, Berlin, Germany, pp.258–265, Springer-Verlag, 2007.

[3] J. Simm, M. Sugiyama, and T. Kato, "Computationally efficient multi-task learning with least-squares probabilistic classifiers," Information and Media Technologies, vol.6, no.2, pp.508–515, 2011.

[4] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with Dirichlet process priors," Journal of Machine Learning Research, vol.8, pp.35–63, 2007.

[5] J.B. Faddoul, B. Chidlovskii, R. Gilleron, and F. Torre, "Learning multiple tasks with boosted decision trees," in Machine Learning and Knowledge Discovery in Databases, pp.681–696, Springer, 2012.

[6] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.D. Collin, and B. Kégl, "Multiboost: a multi-purpose boosting package," The Journal of Machine Learning Research, vol.13, pp.549–553, 2012.

[7] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," Machine learning, vol.63, no.1, pp.3–42, 2006.

[8] L. Breiman, "Random forests," Machine learning, vol.45, no.1, pp.5–32, 2001.

[9] R.E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Machine learning, vol.37, no.3, pp.297–336, 1999.

[10] B. Bakker and T. Heskes, "Task clustering and gating for Bayesian multitask learning," Journal of Machine Learning Research, vol.4, pp.83–99, 2003.