Learning Multiple Tasks with Boosted Decision Trees

Jean Baptiste Faddoul, Boris Chidlovskii, Rémi Gilleron, and Fabien Torre

Xerox Research Center Europe
{boris.chidlovskii}@xrce.xerox.com
http://www.xrce.xerox.com
Lille University, LIFL and INRIA Lille Nord Europe
{jean-baptiste.faddoul,fabien.torre,remi.gilleron}@univ-lille3.fr
http://www.lifl.fr

Abstract. We address the problem of multi-task learning with no label correspondence among tasks. Learning multiple related tasks simultaneously, by exploiting their shared knowledge can improve the predictive performance on every task. We develop the multi-task Adaboost environment with Multi-Task Decision Trees as weak classifiers. We first adapt the well known decision tree learning to the multi-task setting. We revise the information gain rule for learning decision trees in the multi-task setting. We use this feature to develop a novel criterion for learning Multi-Task Decision Trees. The criterion guides the tree construction by learning the decision rules from data of different tasks, and representing different degrees of task relatedness. We then modify MT-Adaboost to combine Multi-task Decision Trees as weak learners. We experimentally validate the advantage of the new technique; we report results of experiments conducted on several multi-task datasets, including the Enron email set and Spam Filtering collection.

Keywords: Multi-Task Learning, Boosting, decision trees, information gain

1 Introduction

Multi-task learning [3] aims at improving the performance of related tasks by learning a model representing the common knowledge across the tasks. Traditionally, the existing techniques assume that tasks share the same instance and label space [14], in the case of classification [6, 18], regression [5, 1], ranking [4] and feature learning [2].

However, in many natural settings these assumptions are not satisfied. A known example is the automatic categorization of Web pages into hierarchical directories, like DMOZ or Yahoo! [12]. When building a categorizer for the Yahoo! directory, it is desirable to take into account DMOZ web directory, and vice versa. The two tasks are clearly related, but their label sets are not identical. Moreover, both ontologies can evolve with time when new categories are added to the directories and some old categories die naturally due to lack of interest.

Multi-task learning with no label correspondence was considered in Quadrianto et al. in [15], where the problem is formulated as learning the maximum entropy estimator $H(Y|X)$ for each task while maximizing the mutual information $-H(Y, Y')$

among the label sets Y and Y' of different tasks. Their approach relies on the hypothesis of the global correlation between tasks in the whole learning space. Tests on the real world datasets show however that this global relatedness assumption turns to be too strong. Indeed, task relatedness may show up different degrees or even different signs in different regions of the learning space. It is therefore important that the multi-task learner copes with the varying relatedness of tasks, learns its different degrees and accommodates the inductive bias accordingly.

We are interested in the multi-task learning where label sets are close but differ from one task to another and the number of classes might be different across tasks. Our motivating example is the automatic classification of e-mails in personal inboxes [13]. Similarly to the case of Yahoo! and DMOZ web directories, categories used in two e-mail inboxes may be related but not identical. For example, people may use *Family* or *Home* categories for personal e-mails and *Finance* or *Budget* for e-mails relevant to financial issues. The application becomes particularly critical when inboxes are owned by people from the same organization; they may share the same messages but classify them according to personal category names. We therefore expect that learning all tasks simultaneously can benefit to the classification model for each task.

In the previous work [7] we proposed a method for multi-task learning for tasks with different label sets which makes no assumption on global relatedness. For this purpose, we developed a multi-task learning algorithm *(MT-Adaboost)* which extends Adaptive boosting *(Adaboost)* [9] to the multi-task setting. The boosting technique is used to generate and combine multiple (weak) classifiers to improve the predictive accuracy. As weak classifiers, we introduced multi-task stumps which are trees having at each node a decision stump for one task. According to the boosting principle, a smart re-weighting of examples from different tasks without label correspondences can grasp the local relatedness of tasks. The method however suffers from some limitations. The greedy algorithm which learns a multi-task stump level-by-level, is based on a heuristic choosing at the root the best stump for the easiest task (where the training error is the lowest); then it forwards recursively to the next levels to learn the remaining tasks. In this kind of a cascade classification on tasks, it learns at each node a classifier for a task taking into account the other tasks' classifiers in the node's ancestors. Unfortunately, such a sequential design of multi-task stumps performs poorly when its greedy algorithm fails to capture task relatedness. In addition, multi-task stumps are binary classifiers, and their extension to multiple multi-class tasks requires additional efforts.

In this work, we propose a novel technique for the multi-task learning which addresses the limitations of previous approaches. First, we propose *Multi-Task Decision Tree (`MT-DT`)* as a multi-task weak classifier. We revisit the well known *C4.5* decision tree learning and adapt it to the multi-task setting. Decision trees

are naturally multi-class classifiers, thus `MT-DT` can learn multiple multi-class classification tasks. Our main contribution is in proving that `MT-DT` can benefit from an improved information gain criterion due to the multi-tasking. Unlike multi-task stumps, the criterion used to learn the nodes makes use of the data from several tasks at each node.

Second, we proceed by plugging the `MT-DT` in the boosting framework; we modify MT-Adaboost to cope with the multi-class problems accordingly. We follow the work of Schapire et al. [17] which analyzed Adaboost with weak learners which abstain and proposed several variations of Adaboost to carry on multi-class problems. Our modification of MT-Adaboost adapts their *Adaboost.M1* algorithm.

As the experimental study will show, our method does not have a uniform margin of improvement over all the tasks. In other words, the tasks which have lower prediction accuracy in single task learning they have a higher improvement potential when learned by our multi-task algorithm.

In the following section we formalize the multi-task learning and introduce the multi-task decision trees. We revisit the information gain rule for the multi-task learning and derive a novel criterion for learning `MT-DT`. In Section 3 we present the boosting framework for the multi-task with `MT-DT` as weak learners. We report the evaluation results for synthetic and real world multi-task datasets in Section 4; Section 5 concludes the paper.

2 Multi-task learning

2.1 Notation and Setting

In the conventional setting, a supervised classification task T is defined over the instance space $\mathcal{X}$ and the space $\mathcal{Y}$ of labels. Let D denote a distribution over $(\mathcal{X}, \mathcal{Y})$, let $f : \mathcal{X} \to \mathcal{Y}$ be a target function. Given a set of training examples $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i = f(x_i),\ 1 \leq i \leq m\}$, the goal of learning is to find an hypothesis function h which minimizes an error function, defined over D as $\text{error}(h) = Pr_{<x,y>\sim D}[h(x) \neq y]$.

We now consider N classification tasks $T_1, \ldots, T_N$ over the instance space $\mathcal{X}$ and label sets $Y_1, \ldots, Y_N$, where labels in sets $\mathcal{Y}_i$ are correlated but not identical. Due to the label mismatch the label sets, we assume that $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$. We are interested in solving N classification tasks simultaneously, in order to improve classification accuracy. We suppose a distribution D^N over $\mathcal{X} \times \{1, \ldots, N\}$. We assume that, for every $j \in \{1, \ldots, N\}$, the projection on the distribution's j-th component will correspond to the original distribution for task T_j. A multi-task classification algorithm will take as input the training set $S = \{< x_i, y_i, j >| \ x_i \in \mathcal{X},\ y_i = f_j(x_i) \in \mathcal{Y}_j,\ j \in \{1, \ldots, N\},\ 1 \leq i \leq m\}$. It should be noted that the same instance x can appear in sample S in different tasks T_i and T_j

with corresponding labels y_i and y_j. The goal is to find an hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$ which minimizes error$(h) = Pr_{<x,y,j>\sim D^N}[h_j(x) \neq y]$, where $h_j(x)$ is the j-th component of $h(x)$ and $j \in \{1, \ldots, N\}$.

2.2 Multi-Task Decision Tree

Decision tree learning is a well known technique in machine learning; it uses a decision tree as a predictive model which maps observations from the instance space to the target values. In the case of classification, tree leaves represent class labels and branches represent conjunctions of item attributes that lead to those class labels [16].

In the C4.5 and C5.0 tree generation algorithms, the decision tree learning uses the concept of the *information gain* (IG) from the information theory. At the root of the tree, the algorithm chooses an attribute that yields the highest IG on the training set. Such an attribute splits the training set S into two subsets S_1 and S_2 whose sum of labels entropy is the lowest. The algorithm then recursively applies the information gain rule on the subsets. The recursion is stopped when all items of a subset have the same label, a decision leaf corresponding to this label [1].

The amount of information gain about a label variable $Y \in \mathcal{Y}$ obtained by observing that an attribute variable a takes value v can be measured by the Kullback-Leibler divergence $D_{KL}(p(Y|a)||p(Y))$ of the prior distribution $p(Y)$ from the posterior distribution $p(Y|a)$ for Y given a. The information gain rule estimates the average improvement. Thus the decision tree algorithm uses the rule to recursively split the instance space, by selecting an attribute with the high information gain.

In this paper we adapt the information gain based decision tree learning to the multi-task setting. One obvious difference between one- and multi-task setting is in the tree structure. One-task decision tree uses the internal test nodes to guide the decision process while the final decision on assigning a label to a sample is made in a tree leaf.

The structure of an multi-task decision tree (`MT-DT`) is different in the way it guides the decision process for multiple tasks. This process is not necessarily the same for all tasks. An `MT-DT` can make a final decision for some tasks in an internal test node, not a tree leaf. This happens when the internal test node has enough information to classify an instance of a certain task T, in such a case a decision leaf with the appropriate classification decision for T is added to the tree and the learning proceeds with the remaining tasks.

Figure 1.a gives an example of an `MT-DT` learned for two synthetic tasks generated from `2D` mixture of Gaussians (see Figure 1.b). T_1 has four labels ($\mathcal{Y}_1 = \{\square, \diamond, \triangle, \circ\}$) and T_2 has two labels ($\mathcal{Y}_2 = \{+, *\}$). Two labels of T_1 ($\square, \diamond$) are correlated with label + of T_2, while two other labels of T_1 ($\triangle, \circ$) are correlated with label

[1] Some pruning is often used to generalize the rules learned to unobserved items.

$*$ of T_1. The generated MT-DT has three internal test nodes and each decision leaf carries one rule per task.

Another example of MT-DT is showed in Figure 2. Task T_1 is the same as Figure 1, while task T_2 is generated differently from a mixture of Gaussians (see Figure 2.b). This results in a different correlation pattern between the tasks. The learned MT-DT has an early decision leaf for T_2 since knowing that $x_1 > -2$ is enough to predict the label class $*$ for T_2.

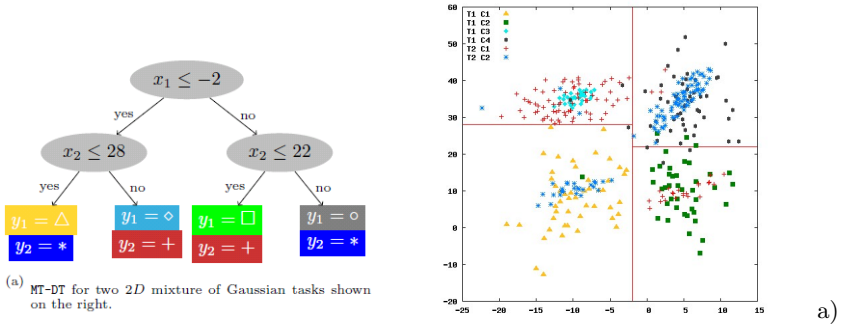


Fig. 1. a) MT-DT example for two tasks. b) Two $2D$ mixture of Gaussian tasks.

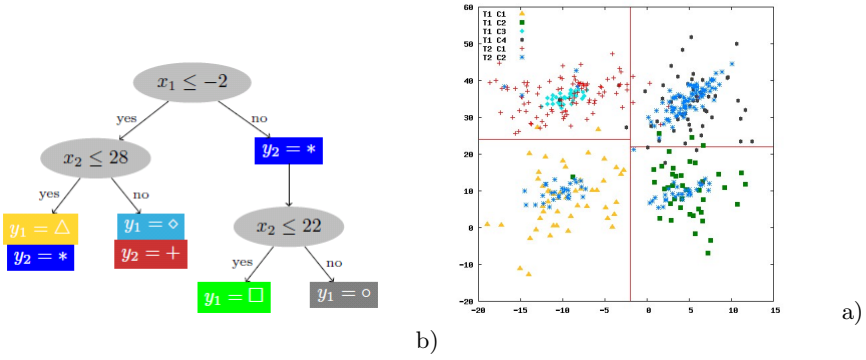


Fig. 2. a) MT-DT with early decision leaf. b) Two $2D$ mixture of Gaussian tasks.

When moving from one- to multi-task setting, the adaptation of the tree structure learning is straightforward. The main challenge is however in the optimal way of using the information gain criteria. In the next section we show how MT-DT can profit from the multi-task setting.

2.3 Multi-task information gain

As said before the decision tree learning is based on the entropy-based criteria, in particular, on the quantity of the mutual dependency between two random variables, the label variable $Y \in \mathcal{Y}$ and the observation attribute a which is one of the attributes of an input vector $x \in \mathcal{X}$. The information gain denoted $IG(Y;a)$ can be expressed as follows

$$IG(Y;a) = H(Y) - H(Y|a), \tag{1}$$

where $H(Y) = -\sum_{y \in \mathcal{Y}} p(y) \log p(y)$ is the marginal entropy of label set $\mathcal{Y}$ and $H(Y|a) = \sum_v p(v) H(Y|a=v)$ is the conditional entropy of Y knowing a.

Assume now we cope with N tasks with the corresponding label sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$, respectively. For learning the `MT-DT`, the baseline approach is to treat all the tasks together by concatenating the label sets, denoted as $\oplus_{j=1}^{N} \mathcal{Y}_j$. The concatenated task takes as input a sample $S = \{< x_i, y_i > |\ x_i \in \mathcal{X},\ y_i = f(x_i) \in \oplus_{j=1}^{N} \mathcal{Y}_j,\ 1 \leq i \leq m\}$. It can use the *joint information gain* for learning decision rules, defined as $IG_J = IG(\oplus_{j=1}^{N} Y_j; a)$. As an alternative to IG_J, we could use the unweighted sum of individual task information gains, $IG_U = \sum_{j=1}^{T} IG(Y_j; a)$. Evaluations however show that IG_U gives lower information gain values comparing to IG_J.

We will prove below that IG_J is equivalent to the weighted sum of individual task information gains and infer an IG criterion with higher values compared to IG_J. The novel IG criterion, denoted IG_M, takes the maximum value among the individual IGs, $IG_M = max\{IG(Y_j; a), j = 1, \ldots, N\}$.

We first recall the generalized grouping feature of the entropy [10] in the following lemma. It establishes a relationship between the entropy of an entire set of values and the entropies of its disjoint subsets.

Lemma 1. *For $q_{kj} \geq 0$, such that $\sum_{k=1}^{n} \sum_{j=1}^{m} q_{kj} = 1, p_k = \sum_{j=1}^{m} q_{kj}, \forall k = 1, \ldots, n$, the following holds*

$$H(q_{11}, \ldots, q_{1m}, q_{21}, \ldots, q_{2m}, \ldots, q_{n1}, \ldots, q_{nm}) = \tag{2}$$

$$H(p_1, \ldots, p_n) + \sum p_k H\left(\frac{q_{k1}}{p_k}, \ldots, \frac{q_{km}}{p_k}\right), p_k > 0, \forall k. \tag{3}$$

Using Lemma 1, we can prove the following theorem on the relationship between the joint information gain $IG(\oplus_{j=1}^{N} Y_j; a)$ of the full task set and of the individual tasks $IG(Y_j; a), j = 1, \ldots, N$.

Theorem 1. *For N tasks with the class sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$, let p_j denote the fraction of task j in the full dataset, $p_j = \frac{|S_j|}{\sum_{j=1}^{N} |S_j|}, j = 1, \ldots, N$, $\sum_{j=1}^{N} p_j = 1$. Then we*

have

$$IG(\oplus_{j=1}^{N} Y_j; a) = \sum_{j=1}^{N} p_j IG(Y_j; a) \leq max(IG(Y_1; a), \ldots, IG(Y_N; a)). \quad (4)$$

Proof. First, we use Lemma 1 to develop the entropy term $H(\oplus_{j=1}^{N} Y_j)$ of the information gain (1). We have

$$H(\oplus_{j=1}^{N} Y_j) = H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j), \quad (5)$$

where $\sum_{j=1}^{N} p_j = 1$.

Second, we develop the conditional entropy term in (1), as follows

$$H(\oplus_{j=1}^{N} Y_j | X) = \sum_{x} p(x) H(\oplus_{j=1}^{N} Y_j | a = v) \quad (6)$$

$$= \sum_{v} p(v) \left(H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j | a = v) \right) \quad (7)$$

$$= H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j | a). \quad (8)$$

Now we combine the entropy (5) and the conditional entropy (8) terms to evaluate the joint information gain $IG(\oplus_{j=1}^{N} Y_j; a)$. We obtain

$$IG(\oplus_{j=1}^{N} Y_j; a) = H(\oplus_{j=1}^{N} Y_j) - H(\oplus_{j=1}^{N} Y_j | a) \quad (9)$$

$$= \sum_{j=1}^{N} p_j IG(Y_j; a) \quad (10)$$

$$\leq max(IG(Y_1; a), \ldots, IG(Y_N; a)). \quad (11)$$

This completes the proof of the theorem.

Theorem 1 says that criterion IG_M for the decision tree learning in the multi-task case gives larger information gain values comparing to the joint one IG_J.

Figure 2.3 compares three criteria IG_U, IG_J and IG_M for some randomly generated two-task datasets. Two label sets are generated by sampling from the Uniform, Normal (with $\mu = 0$, $\sigma = 1$) and Poisson ($\lambda = 1$) distributions; the number of labels in the two sets vary from 2 to 20. Attributes values are sampled from uniform distributions in all cases. We measure the relative values of IG_M

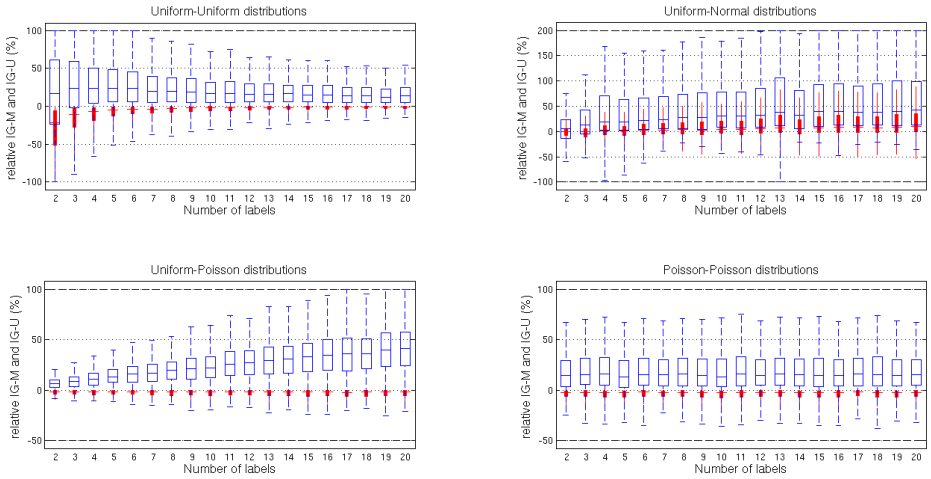


Fig. 3. Information gain for synthetic two-task datasets. The relative values of IG_M (in blue) and IG_U (in red).

and IG_U with respect to IG_J. In all cases, we report the median, the upper and lower percentiles, and the whiskers over 100 runs. As the figure shows, IG_M yields on average up to 42% larger information gain values than IG_J, with the minimal gain in the case of two Uniform distributions.

As we can notice from the plots, the variance of IG_M values is very high compared to this of IG_U which have almost zero variance. An information gain criterion with small variance is not a good indicator to help choosing a good node because all node will have very close IG values. The explanation of such small variance is that when we take the sum of IGs for all tasks, it is difficult to come out with a node that satisfies all of them.

2.4 Learning Algorithm for `MT-DT`

The learning algorithm for `MT-DT` applies one of proposed information gain criteria to the available training set S:

$$MTIG(S) \equiv (\mathbf{a}^*, v^*) = max_{a \in \mathcal{X}, v \in V_a} IG_*(S),$$

where a is an attribute in feature space $\mathcal{X}$, S is the training set, a takes one of the possible values $v \in V_a$ and a pair (a^*,v^*) yields the optimal split on S using as criterion IG_* which can refer to IG_J, IG_U or IG_M.

The pseudo code of the MT-DT algorithm is presented in Algorithm 1. The algorithm makes a call to a function $MTIG$ which returns the node with rule $a \leq v$ that maximizes a given information gain on a multi-task training set S, Then it

gets subsets S_1, S_2 resulting from splitting S on the chosen node. At each node the algorithm adds decision leaves for the tasks having no items in the subset or having items with the same label. Then, it calls recursively the procedure on each of subsets. The learning algorithm returns at the end a tree that gives for each example (x) one label for each task.

In the evaluation section, we test three versions of the IG criterion introduced before, IG_J, IG_U and IG_M. It is worth noting that we can limit the depth of the trees by modifying the stopping criterion, instead of stopping the growth of a certain branch when we have homogenous labels for all tasks in the subspace corresponding to that branch, we can stop when we exceed a threshold. For instance, when 80% of the examples are from the same labels. This should not be an issue as long as we are using an ensemble of trees learned by a boosting algorithm.

Require: $S = \cup_{j=1}^{N} \{e_i =< x_i, y_i, j > |\ x_i \in \mathcal{X};\ y_i \in \mathcal{Y}_j\}$
Require: $MTIG$: multi-task information gain criterion
`res = []` {Will contain the chosen node and early decision leaves, if any}
for $j = 1$ to N **do**
  if task j's examples (S_j) has all the same label **or** $S_j = \emptyset$ **then**
    Add to `res` a leaf for task j and label y. {y is either the unique label of S_j in case it is homogeneous or it is the majority label of its parent subset in case $S_j = \emptyset$}
    $S = S \setminus S_j$
  end if
end for
Get the `bestnode` rule $(a, v) = MTIG(S)$ which maximizes the information gain
Call `split`(S, a, v))
Get back $[S_1, S_2]$, two subsets resulted from splitting S based on `bestnode`
Add `bestnode` to `res`
Call recursively the algorithm on S_1 and S_2 to get the children of `res`
return `res`

Algorithm 1: `MT-DT` algorithm.

3 Multi-Task Adaboost

In the previous section we developed a novel technique for learning `MT-DT`'s with an improved information gain criterion. To avoid all disadvantages of the decision trees such as overfitting, in this section we proceed by plugging the `MT-DT`'s in the boosting framework.

We adapt Adaboost.M1 which was introduced in [9]. We preferred M1 to MH or other multi-class boosting algorithm because it requires a weak classifier which is naturally multi-class. It does not need a weak learner which transforms a problem to several binary problems, and since, we propose a multi-task learner based on decision trees that are naturally multi-class classifiers, Adaboost.M1 is a good

candidate. In addition to that, it is the most straightforward multi-class extention of Adaboost. Nevertheless, it puts strong requirement on the weak learner; actually, it requires the classification error of the weak classifier to be less than 0.5 w.r.t. to the current weight distribution, regardless the number of class labels. Some weak learners, such as stumps, are unable to satisfy such a strong boosting condition. But, normally, decision trees can satisfy this condition.

The proposed Multi-Task Adaboost algorithm (`MT-Adaboost`) is presented in Algorithm 2. T is the number of boosting iterations; `init` is a procedure to initialize the distribution D_1 over S; and `WL` is a weak learner that returns an `MT-DT` given as input a sample S and a distribution D over S. The final output is a multi-task classifier H from $\mathcal{X}$ into $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$. As in single task boosting algorithms, `MT-Adaboost` calls `WL` repeatedly in a series of rounds. On each round t, the algorithm provides `WL` with the current distribution D_t and the training sample S, in return `WL` learns a classifier $h_t : \mathcal{X} \rightarrow \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$ which minimizes the training error on S with respect to D_t. The distribution D_{t+1} is then calculated from D_t and h_t as follows. Correctly classified examples by h_t will have their weights multiplied by $0 \leq \beta_t \leq 1$ (i.e., decreased), and the weights of misclassified examples will be left unchanged. Finally, the weights are renormalized by using the normalization constant Z_t.

The final classifier H for a given task j is a weighted vote of the weak classifiers' predictions for this task. The weight given to hypothesis h_t is defined to be $ln(1/\beta_t)$ so that greater weight is given to hypotheses with lower error. `MT-Adaboost` has the same theoretical properties of Adaboost.M1, that is, if the weak hypotheses have error only slightly better than 1/2, then the (training) error of the final hypothesis H drops to zero exponentially fast in function to the number of boosting iterations T.

4 Experiments

In this section we present a series of experiments on three datasets. We describe the datasets and evaluation framework, then we compare the predictive performance of single task decision trees to different MT-DTs learned using IG_J, IG_U and IG_M criteria. Then we report experimental results on boosted trees using MT-Adaboost.

4.1 Datasets

Synthetic We generate synthetically tasks with local relatedness patterns, by following the data generation technique described in [8]. Each pattern is generated a random Bayesian network (BN) from which one can derive different but related probabilistic distributions. The BN is created by generating (a) a random (directed acyclic) graph, (b) a set of functions (with random parameters) characterizing the

Require: $S = \cup_{j=1}^{N}\{e_i =< x_i, y_i, j >|\ x_i \in \mathcal{X};\ y_i \in \mathcal{Y}_j\}$
1: $D_1 = \texttt{init}(S)$ *initialize distribution*
2: **for** $t = 1$ to T **do**
3: $h^t = \texttt{WL}(S, D_t)$ {*train the weak learner and get an hypothesis* `MT-DT`}
4: Calculate the error of h^t: $\epsilon_t = \sum_{j=1}^{N}\sum_{i:h_j^t(x_i)\neq y_i} D_j(x_i)$.
5: **if** $\epsilon_t > 1/2$ **then**
6: Set $T = t - 1$ and abort loop.
7: **end if**
8: $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
{*Update distribution:*}
9: **if** $h_j^t(x_i) == y_i$ **then**
10: $D_{t+1}(e_i) = \frac{D_t(e_i)\times\beta_t}{Z_t}$
11: **else**
12: $D_{t+1}(e_i) = \frac{D_t(e_i)}{Z_t}$
13: **end if**
14: **end for**
{*Where* Z_t *is a normalization constant chosen so that* D_{t+1} *is a distribution*}
15: **return** Classifier H defined by:

$$H_j(x) = \arg\max_{y\in\mathcal{Y}_j}(\sum_{i=1}^{i=T}(\ln 1/\beta_t)),\ 1 \leq j \leq N$$

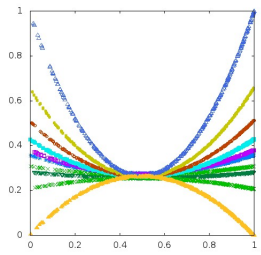
Algorithm 2: MT-Adaboost.

dependence of every node on each one of its parents in the graph, and (c) a set of functions (with randomly assigned parameters) defining the probability density of each node [2].

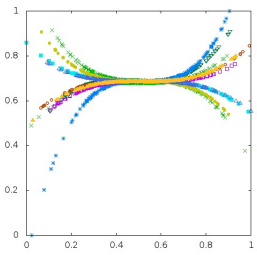
Figure 4 shows some examples of the local tasks relatedness generated using such method. In the plotted examples, the distributions feature cubic, exponential and linear correlation functions, with Beta, Gaussian and Laplacian densities. Using the random relatedness generator we generate three multi-task learning datasets. DS_1 consists of two tasks T_1 and T_2, having three and two labels, respectively. They are plotted in Figure 5.a. We can see that the red class of T_1 is locally correlated with the light blue class of T_2; similarly, the green class is locally correlated with the violet. However the dark blue class of T_1 which is locally correlated with the violet in the upper part of its density and with the light blue in the lower part. The second dataset DS_2 is shown in Figure reffig:mtsynthetic.b with tasks being also locally correlated. Finally, random noise is added to the labels of all tasks as follows. For a certain example with label y we place a discrete probability distribution over the label set with 90% of mass concentrated over y and the rest distributed equally over the other labels. Then we sample the noisy label from this distribution. It should be noted that we generate tasks with different number of

[2] The code is provided by Antonino Freno `http://researchers.lille.inria.fr/~freno/software.html`

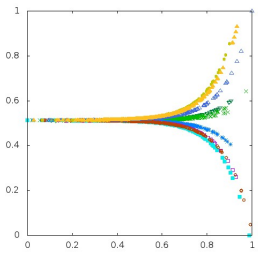
class labels on purpose, in order to test the proposed methods on configurations not addressed by prior-art methods.



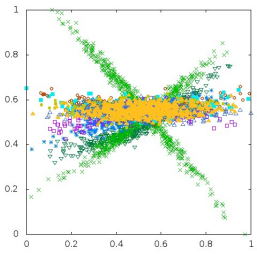
(a) A correlation pattern from beta-quadratic distributions



(b) A correlation pattern from gaussian-cubic distributions



(c) A correlation pattern from gaussian-exponential distributions

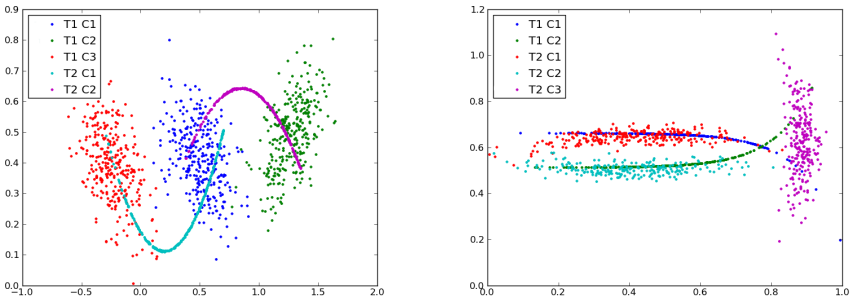


(d) A correlation pattern from gaussian-quadratic distributions

Fig. 4. Tasks Relatedness Patterns for synthetic 2D data

Enron Enron dataset[3] contains all e-mails sent and received by some 150 accounts of the top management of Enron company and spans a period of several years. Annotations of the Enron dataset come from two different sources, thus, naturally constituting two tasks. The first is from the Department Of Justice of the United States DOJ[4], which has published a list of responsive emails used in the trials against the two CEO's of Enron. This set along with a manually annotated non-responsive emails constitute a binary classification task, *Responsive* Vs. *Non-responsive*, with total of 372 emails. The second annotated set comes from students of Berkeley University. Emails in this set are annotated by topic, for an average of 250 emails per topic. Five topics are used in our experiments: *Business*, *Legal*, *Influence*, *Arrangement* and *Personal*. We used the textual features of Enron dataset along with the social features (see [11] for more details).

[3] `http://www.cs.cmu.edu/~enron/`.
[4] `http://www.usdoj.gov/enron/`.



(a) Two related multi-class tasks (b) Two related multi-class tasks

Fig. 5. Two classification problems, each with two multi-class tasks.

Spam Filtering This dataset was used for the ECML/PKDD 2006 discovery challenge. It contains email inboxes of 15 users. Each inbox has 400 spam/ham emails. They are encoded by standard bag-of-word vector representation. We consider each user as a task.

MNIST Character Recognition We use this dataset adapted to the multi-task setting because it was used by a state-of-the-art method [15], so we can and we can compare with their results. For the experiments, we consider multi-task learning problems with 10 tasks representing the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$. We follow the same protocol given in [15] so we can be able to have a good comparison.

4.2 Results on Trees

In this section we report experimental results of MT-DTs learned using IG_J, IG_U and IG_M criteria introduced in Section 2.3. We also compare MT-DT to single task decision trees learned with C4.5 algorithm. In all experiments we use the 5-fold cross validation, where each run consists of training on four folds and testing on the remaining one. We run all methods fifty times on random shuffles of the data and report the average values. Results in bold are statistically significant by a t-test with $\alpha = 0.05$.

Table 1 reports the size of training/test sets and the evaluation results for three synthetic datasets. We note that MT-DT with IG_M brings a significant improvement over C4.5. While IG_J and IG_U behave comparably to C4.5, they are slightly better on Task-1, but suffer an accuracy drop on Task-2.

The same behavior is observed on ECML'06 data (see Table 2). It shows a superiority of IG_M over other MT-DT criteria in accuracy values. However, learning tasks simultaneously does not bring the same improvement to all tasks, some tasks

Tasks	Train (Test)	C4.5	IG_J	IG_U	IG_M
Data Set 1					
Task-1	300 (1200)	86.432 ± 0.003	86.116 ± 0.063	86.070 ± 0.029	87.180 ± 0.037
Task-2	200 (1300)	89.532 ± 0.167	88.980 ± 0.391	89.237 ± 0.445	89.246 ± 0.341
Avg		87.982	87.548	87.653	**88.213**
Data Set 2					
Task-1	200 (1300)	90.738 ± 0.092	88.008 ± 0.606	89.848 ± 0.063	90.751 ± 0.085
Task-2	300 (1200)	83.525 ± 0.600	88.056 ± 1.363	88.221 ± 1.316	88.366 ± 0.366
Avg		87.132	88.032	89.035	**89.559**

Table 1. Average classification accuracy on the three synthetic datasets

tend to benefit more from multi-task learning than others. Results show that more difficult tasks (tasks with a lower accuracy) got a higher improvement on their prediction accuracy when learned by other tasks.

Tasks	Train (Test)	C4.5	IG_J	IG_U	IG_M
User-1	320 (80)	86.45 ± 1.23	86.19 ± 1.14	86.00 ± 1.88	**87.65±3.42**
User-2	320 (80)	85.13 ± 2.16	85.53 ± 2.22	85.07 ± 3.16	**88.93±3.44**
User-3	320 (80)	88.03 ± 2.11	88.22 ± 2.56	**88.52±1.33**	88.19 ± 2.51
Avg		86.54	86.65	86.53	**88.26**

Table 2. Average classification accuracy on three ECML'06 user inboxes.

4.3 Results on Boosted Trees

In the previous section we experimentally validated the advantage of learning related tasks simultaneously, by using multi-task information gain criteria, in particular IG_M. In this section we compare boosted `MT-DT`'s to the boosted C4.5 trees. We use Adaboost.M1 [17] and MT-Adaboost (see algorithm 2) as boosters for C4.5 and for `MT-DT` respectively. Both algorithms have only one parameter, the number of boosting iterations which we set on a separated validation set.

Table 3 reports our comparison with the work in [15], we can see that our multi-task algorithm significantly outperform Adaboost with trees for single task learning and it proves a large margin of improvement over the method in [15], despite that we exactly follow their protocol.

Table 4 reports the average values of classification accuracy over three random runs for Enron dataset. With boosted trees we observe an accuracy improvement similar to simple trees. Namely, MT-Adaboost+MT-DT is significantly better than Adaboost+C4.5; also the most difficult tasks enjoy a larger margin of improvement.

Tasks	Train (Test)	Adaboost	`MTL [15]`	`MT-Adaboost`
1/-1	100 (10000)	91.770 ± 1.188	96.80 ± 1.91	96.802 ± 0.562
2/-2	100 (10000)	83.138 ± 2.347	69.95 ± 2.68	86.875 ± 0.676
3/-3	100 (10000)	82.959 ± 1.245	74.18 ± 5.54	87.679 ± 1.038
4/-4	100 (10000)	83.975 ± 1.408	71.76 ± 5.47	90.382 ± 0.713
5/-5	100 (10000)	78.423 ± 0.691	57.26 ± 2.72	84.253 ± 0.731
6/-6	100 (10000)	88.954 ± 1.601	80.54 ± 4.53	92.880 ± 0.896
7/-7	100 (10000)	87.105 ± 0.904	77.18 ± 9.43	92.811 ± 0.575
8/-8	100 (10000)	77.513 ± 1.905	65.85 ± 2.50	85.279 ± 1.727
9/-9	100 (10000)	81.842 ± 1.850	65.38 ± 6.09	86.904 ± 1.258
0/-0	300 (10000)	93.660 ± 1.287	97.81 ± 1.01	97.137 ± 0.418
Average	-	84.934	75.67	**90.100**

Table 3. Comparison on the MNIST datasets of (single-task) Adaboost, `MTL` and `MT-Adaboost`.

Tasks	Train (Test)	Adaboost C4.5	MT-Adaboost IG_J	MT-Adaboost IG_U	MT-Adaboost IG_M
Responsive Vs. NonResponsive	299 (74)	85.10 ± 1.21	84.66 ± 2.15	84.52 ± 1.2	**86.01±1.53**
5 Topics	265 (66)	51.34 ± 0.43	52.89 ± 0.87	52.17 ± 0.74	**57.11±0.02**
Avg		68.22	68.78	68.35	**71.65**

Table 4. Average classification accuracy of boosted trees on Enron tasks.

5 Conclusion

We proposed an adaptation of decision tree learning to the multi-task setting, with the following important contributions. First, we developed multi-task decision trees to deal with multi-class tasks with no label correspondence. The criterion to learn the decision rules makes use of the data from several tasks at each step of the decision tree learning, thus enabling to capture any degree of relatedness between the tasks. We then feature an important property of information gain rule when working with multiple tasks. This enabled us derive the new information gain criterion for learning decision trees in the multi-task setting. We also modified MT-Adaboost to cope with multi-class problems. We finally validated the proposed methods by series of experiments on three cases of multi-task learning.

6 Acknowledgment

This work was partially supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council, FEDER through the *CPER 2007-2013* and the LAMPADA project co-funded by the *French Association on Research - ANR.*

References

1. Cedric Archembeau, Shengbo Guo, and Onno Zoeter. Sparse Bayesian Multi-Task Learning. In *NIPS*, 2011.
2. Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*, 2007.
3. Rich Caruana. Multitask learning. In *Machine Learning*, volume 28, pages 41–75, 1997.
4. Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proc. 16th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 1189–1198, 2010.
5. Wenyuan Dai, Qiang Yang, Gui R. Xue, and Yong Yu. Boosting for transfer learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.
6. Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD '04: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117. ACM, 2004.
7. Jean Baptiste Faddoul, Boris Chidlovskii, Fabien Torre, and Rémi Gilleron. Boosting multi-task weak learners with applications to textual and social data. In *Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA)*, pages 367–372, 2010.
8. Antonino Freno, Edmondo Trentin, and Marco Gori. Kernel-based hybrid random fields for non-parametric density estimation. In *ECAI*, pages 427–432, 2010.
9. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
10. Robert M. Gray. *Entropy and Information Theory, Second Edition.* Springer-Verlag, 2011.
11. Matthijs Hovelynck and Boris Chidlovskii. Multi-modality in one-class classification. In *Proceedings of the 19th international conference on World wide web (WWW)*, pages 441–450, 2010.
12. Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43, June 2005.
13. Amin Mantrach and Jean-Michel Renders. Search engine for mailboxes based on cross-modal, topics and communities query expansion. In *Proceedings of European Conference on Information Retrieval (ECIR'2012)*, 2012.
14. Sinno J. Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345 – 1359, 2010.
15. N. Quadrianto, A. Smola, T. Caetano, S.V.N. Vishwanathan, and J. Petterson. Multitask learning without label correspondences. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1957–1965, 2010.
16. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
17. Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37, 1999.
18. Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, 8:35–63, May 2007.