Learning Many Tasks with Adaboost

Jean-Baptiste Faddoul, Boris Chidlovskii
Xerox Research Center Europe

Fabien Torre, Remi Gilleron
Lille university, INRIA LNE and LIFL

Abstract

Learning tasks simultaneously can improve predictive performance relative to learning these tasks independently. In this paper, we consider multi-task learning when the number of tasks is large. The weak classifiers we consider are 2-level decision stumps for different tasks. A weak classifier assigns a class to each instance on two tasks and abstain on other tasks. The weak classifiers allow to handle dependencies between tasks on the instance space. We introduce different efficient weak learners. We then consider Adaboost with weak classifiers which can abstain and adapt it to multi-task learning. In an empirical study, we compare the weak learners and we study the influence of the number of boosting rounds. Last, we give experimental evidence that Adaboost with 2-level decision stumps outperform existing multi-task learning algorithms on benchmark multi-task problems.

Keywords: Multi-Task Learning, Boosting

1 Introduction

Multi-task learning [Caruana, 1997] aims to improve the performance of related tasks by learning a model able to represent the common knowledge across the tasks. Previous work on multi-task learning were essentially focused on neural networks, k-nearest neighbors [Caruana, 1997], and support vector machines [Evgeniou and Pontil, 2004, Zheng et al., 2008]. Also, the multitask setting has been considered for semi-supervised learning [Liu et al., 2009]. Most of the existing techniques make some convenience assumptions, like one that the tasks share the same label and/or examples. They also try to model the relation between tasks. Recent works on Sparse Matrix Penalty [Jalali et al., 2010, Zhang and Schneider, 2010] try to learn a common feature representation shared by tasks and estimate a matrix of model parameters where the rows and columns correspond to tasks and features. In [Quadrianto et al., 2010], the authors consider multitask learning where label sets may be distinct. They formulate the problem as a maximization problem of mutual information among the label sets.

We follow this line of research because tasks with no label correspondence are much more frequent and critical to address. But, the work [Quadrianto et al., 2010] relies on the hypothesis that the relation between tasks is the same in the whole learning space. And, the global relatedness assumption turns often to be too strong; it might even hurt the performance, similarly to introducing noise in data. This 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 determines the relatedness of tasks, learns its different degrees and accommodates the inductive bias accordingly.

To relax the mentioned constraints on sharing labels/examples and global relatedness hypothesis, our approach is to extend Adaptive boosting (Adaboost) [Freund and Schapire, 1996, Schapire and Singer, 1999] to the multi-task setting. Indeed, we believe that a smart re-weighting of instances from different tasks without label correspondences can grasp well the local relatedness of tasks. Boosting algorithms have already been used in the context of multi-task learning. Dai et al. [Dai et al., 2007] developed a variation of AdaBoost that can incorporate training data from a different distribution than the test set. Instead of learning multiple models, their approach down-weighs data points from the other distribution that are not representative for the test set. The boosting principle was also used in multi-task learning for face verification in [Wang et al., 2009]. To learn a set of boosted classifiers, it uses a probabilistic model where a multinomial variable indicates how much each boosted classifier contributes to each task. The Expectation-Maximization (EM) is deployed to learn both the multinomial random variable and the classifiers. Chapelle et al. [Chapelle et al., 2010] introduced a multi-task learning algorithm for gradient boosting. They learned one model per task and one global model to capture relations between tasks. They considered applications to Web Search Ranking. Faddoul et al. [Faddoul et al., 2010] adapted Adaboost to the multi-task setting and considered as weak classifiers multi-task decision trees.

We follow this approach because there is no assumption on the label sets and the approach allows to learn the relatedness between tasks in different regions of the instance space. But, their approach do not scale up well to many tasks because the number of weak classifiers become too large. To avoid the complexity blow-up, we consider multi-task decision trees with only two levels (`2T-stumps`). When an instance is considered, the `2T-stump` assigns a label for at most two tasks and abstains for all other tasks. Thus, we consider the abstention not as an exception, but as the first class behavior for a weak classifier because abstaining on some tasks is a more natural choice that enforcing a weak classifier to make predictions for all tasks. We introduce and compare different weak learners for `2T-stumps`. We consider Adaboost with abstention as introduced in [Schapire and Singer, 1999]. We adapt it to the multi class setting and show convergence for training error. We consider different weighting schemes for Adaboost and compare the weighting schemes when the number of tasks increases. Last, we design experimental studies to compare the weak learners and to show the influence of the number of boosting rounds. Also, we give experimental evidence that our algorithm outperforms the algorithm of [Quadrianto et al., 2010] on the `MNIST` dataset. To summarize, the main contributions of this paper are the following:

- Design of efficient weak learners for `2T-stumps`,
- adapt Adaboost to the multi-task setting with abstention,
- study Adaboost when the number of task increases,
- show that our algorithm allows to capture local relations between tasks without priors,
- give experimental evidence that our algorithm outperforms some existing multi-class learning algorithms.

The rest of the paper is organized as follows. Our approach is presented and analyzed in Section 2. Then, Section 3 explains the experimental settings and reports the evaluation results. Section 4 concludes the paper and presents future work.

2 Adaboost with Abstention for Multi-task Learning

Let $\mathcal{X}$ be the instance space, a supervised classification task T is defined as follows. Let D be a distribution over $\mathcal{X}$, let $f : \mathcal{X} \to Y$ be a target function, given a sample $S = \{(x_i, f(x_i)) \mid x_i \in \mathcal{X},\ 1 \leq i \leq m\}$, find an hypothesis function h which minimizes $\mathrm{error}(h) = Pr_{x \sim D}[h(x) \neq f(x)]$.

We consider N binary classification tasks $T_1, \ldots, T_N$ over the instance space $\mathcal{X}$ and label sets $Y_1, \ldots, Y_N$. For sake of clarity, we consider binary classification tasks and we assume without loss of generality that the binary labels for all tasks are encoded as -1 and $+1$. The objective is to solve the N classification tasks simultaneously. We suppose a distribution D over $\mathcal{X} \times \{1, \ldots, N\}$. We will assume that, for every j in $\{1, \dots, N\}$, 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 a sample $S = \{< x_i, y_i, j_i >| \ x_i \in \mathcal{X}, y_i = f_{j_i}(x_i) \in \{-1, +1\}, j_i \in \{1, \dots, N\}, \ 1 \leq i \leq m\}$. It should be noted that a same instance x can appear in a sample S with its label for different tasks. The goal is to find an hypothesis $h : \mathcal{X} \to Y_1 \times \ldots \times Y_N$ which minimizes $\text{error}(h) = Pr_{<x,j>\sim D}[h_j(x) \neq f_j(x)]$, where $h_j(x)$ is the j-th component of $h(x)$ and $j \in \{1, \dots, N\}$.

In this section, we present the weak classifiers, we consider, they allow both multi-task predictions and abstention. Then we show that our formalization of multi-task problems will allow to adapt Adaboost and to benefit from theoretical results of boosting. Finally, we propose several weak learners in order to provide weak multi-task hypotheses to Adaboost.

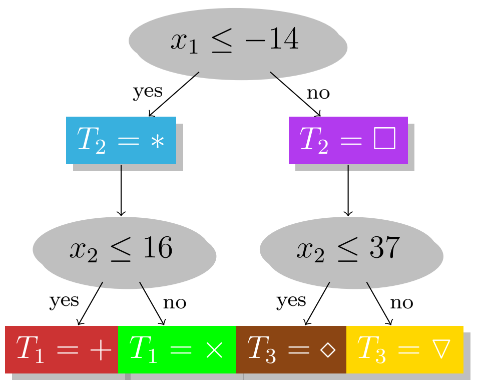
2.1 Multi-task Weak Classifiers

We generalize stumps for multi-task problems. Recall that stumps are one-level decision trees, i.e. a stump is defined by a test node and prediction nodes. For sake of clarity and since n-ary tests are a straightforward generalization of binary tests, we will consider only binary tests. Stumps can be used as weak classifiers and they allow to learn accurate strong classifiers when used in boosting algorithms [Freund and Schapire, 1996].

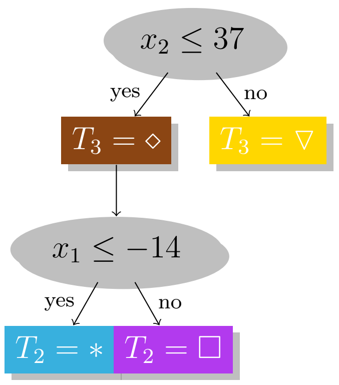
For the multi-task setting, the weak classifiers we consider are 2-level decision stumps called two-task stumps (`2T-stump`). The first level of a `2T-stump` is a decision stump for one of the N tasks. At the second level, there are two decision stumps, one attached at each of the two prediction nodes. Each of these two decision stumps corresponds to one of the $N - 1$ remaining tasks. An example is given in Figure 1.

For classifying a given instance x, the root test is considered and the instance x will descend to one of the two prediction nodes, which assigns a label for the first-level task. Then the second-level test is considered, the instance x will descend to one of the two prediction nodes and a label is assigned for the corresponding second-level task. It should be noted that the `2T-stump` will abstain on x for all other tasks. Examples are given in Figure 1. Formally, let us consider that the value 0 stands for abstention, a `2T-stump` defines an hypothesis $h : \mathcal{X} \to \{-1, 0, +1\}^N$, which gives for every instance x and every task j an output $h_j(x)$ in $\{-1, 0, +1\}$, where $h_j(x)$ is non zero for at most two tasks.

A `2T-stump` defines a partition of the instance space dependent on the tasks it contains. For instance, let us consider the `2T-stump` defined in Figure 1(a), it defines a partition of the instance space shown in Figure 2(a). The line with equation $x_1 = 14$ defines the first-level separator for task T_2 which allows to define two different separators for the tasks T_1 and T_3 in each of the two sub-spaces.



(a) This 2T-stump has a stump for task T_2 at the first level, and stumps for tasks T_1 and T_3 at the second level. Consider an instance x such that $x_1 = -7$ and $x_2 = 4$ then the 2T-stump assigns to x the class $\square$ for task T_2 and the class $\diamond$ for task T_3 while abstaining for other tasks



(b) This 2T-stump has no right child. Consider an instance x such that $x_2 = 40$ then the 2T-stump assigns to x the class ∇ for task T_3 while abstaining for other tasks

Figure 1: Two 2T-stumps.

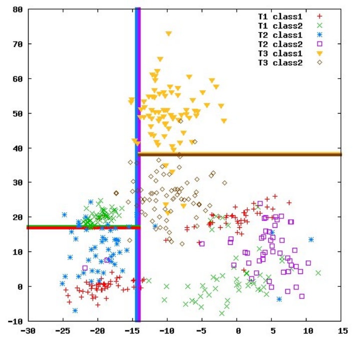
Thus 2T-stumps will allow to capture relations between tasks. One can consider more than two levels but the number of such weak classifiers grows exponentially with the number of levels. This is why we consider 2T-stumps with only two levels and we will show that combining 2T-stumps in a boosting algorithm will allow to capture local dependencies between N tasks.

Now, let us consider a sample S in the multi-task setting, it can also be written as $S = \cup_{j=1}^{j=N} S_j$ where, for every j, $S_j = \{e_i =< x_i, y_i, j >| \ y_i = f_j(x_i)\}$, i.e. we decompose S in samples corresponding to the different tasks. Let us consider a distribution over S. It is worth noting that an instance x can appear in different S_j and will have different probabilities (because the probability is in fact defined on the pair $< x_i, j >$); but, we do not require that such instances exist.

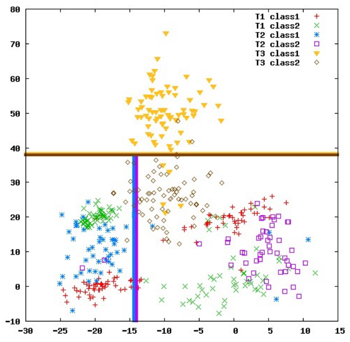
2.2 MTAA: Adaboost for Multi-task Learning with Abstention

We consider the multi-task setting as defined above, with 2T-stumps used as weak classifiers. Given a 2T-stump $h : \mathcal{X} \to \{-1, 0, +1\}^N$, we define W_0, W_{-1} and W_{+1} by

$$W_b = \sum_{i=1}^{i=N} D(\{e_i =< x_i, y_i, j >| \ e_i \in S, \ y_i \times h_j(x_i) = b\})$$



(a) Visualization of the partition defined by the `2T-stump` in Figure 1(a). It is learned on the synthetic dataset with algorithms `WL-Best-K` with $K = 5$ or `WL-Best-K` with $K = 25$. The score $W_- + \frac{1}{2}W_0$ is equal to 0.134



(b) Visualization of the partition defined by the `2T-stump` in Figure 1(b). It is learned on the synthetic dataset with a naive greedy algorithm. The score $W_- + \frac{1}{2}W_0$ is equal to 0.197

Figure 2: Visualization of `2T-stumps` learned on a synthetic dataset.

for $b \in \{-1, 0, +1\}$. We abbreviate W_{-1} and W_{+1} by W_- and W_+ respectively. W_- is the sum of weights of misclassified instances, W_+ is the sum of of weights of well-classified instances, and W_0 is the sum of weights of instances on which h abstains. Since D is a distribution, $W_+ + W_- + W_0 = 1$.

2.2.1 MTAA.

Adapted from [Schapire and Singer, 1999], the generic Adaboost algorithm with abstention for multi-task learning (`MTAA`) is presented in Algorithm 1 where 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 a `2T-stump` given as input a sample S and a distribution D over S. The final output is a classifier H from $\mathcal{X}$ into $\{-1, +1\}^N$. We should note that we suppose that the choice of the number of boosting iterations, the choice of the updating coefficients α and the weak learner `WL` For `MTAA`, the value of K is set to 30, and T to 500. For Adaboost we set T to 100 for each task.ensure that H does not abstain for every instance and for every task.

The training error of the final classifier H is defined by

$$\text{error}(H) = Pr_{<x_i, y_i, j> \sim D_1}[H_j(x_i) \neq y_i].$$

We now prove that the training error decreases to zero exponentially fast.

Theorem 1. *Let us consider `MTAA` with the update rule in line 5 of Algorithm 1 and let us suppose that there exists $\gamma > 0$ such that, at each boosting iteration, $W_+ - W_- \geq \gamma$, then*

```
Require: $S = \cup_{j=1}^{j=N}\{e_i =< x_i, y_i, j >| \; x_i \in \mathcal{X};\; y_i \in \{-1,+1\}\}$
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 2T-stump}
4:     Choose $\alpha_t$
5:     $D_{t+1}(e_i) = \frac{D_t(e_i)\exp(-\alpha_t y_i h_j^t(x_i))}{Z_t}$ {update distribution}
6: end for
7: return Classifier $H$ defined by $H_j(x) = \texttt{sign}(\sum_{i=1}^{i=T} \alpha_t h_j^t(x))$, $1 \le j \le N$
```

Algorithm 1: A generic version of `MTAA`

(i) error$(H) \le \prod_{t=1}^{T} Z_t$,

(ii) Z_t is minimized by choosing

$$\alpha_t = \frac{1}{2}\ln\left(\frac{W_+}{W_-}\right), \qquad (1)$$

(iii) $\prod_{t=1}^{T} Z_t \le e^{-T\frac{\gamma^2}{2}}$.

Proof. Following [Schapire and Singer, 1999], it can be shown that error$(H) \le \prod_{t=1}^{T} Z_t$. Also, for every t, $Z_t = W_0 + e^{\alpha}W_- + e^{-\alpha}W_+$ where W_0, W_- and W_+ are computed as above with $D = D_t$. And, it can be verified that Z_t is minimized when $\alpha_t = \frac{1}{2}\ln(\frac{W_+}{W_-})$. With this setting of α_t, we have $Z_t = W_0 + 2\sqrt{W_- W_+} = 1 - (\sqrt{W_+} - \sqrt{W_-})^2$.

It remains to show property (iii), i.e. to show that the training error decreases exponentially with the number of boosting iterations. Let $W_+ - W_- = \gamma_t \ge \gamma > 0$, then $\sqrt{W_+} - \sqrt{W_-} = \frac{\gamma_t}{\sqrt{W_+}+\sqrt{W_-}} \ge \frac{\gamma_t}{\sqrt{2}}$ because $\sqrt{W_+} + \sqrt{W_-} = \sqrt{W_+} + \sqrt{1 - W_0 - W_+} \le \sqrt{W_+} + \sqrt{1 - W_+} \le \sqrt{2}$. Thus we get $Z_t \le 1 - \frac{\gamma_t^2}{2}$, and then we have $\prod_{t=1}^{T} Z_t \le \prod_{t=1}^{T}(1 - \frac{\gamma_t^2}{2}) = e^{\sum_{t=1}^{T}\ln(1-\gamma_t^2)}$. Last using $\gamma_t \ge \gamma > 0$ and $\ln(1-x) \ge -x$, we obtain: error$(H) \le \prod_{t=1}^{T} Z_t \le e^{-T\frac{\gamma^2}{2}}$. □ □

Thus, the output of the weak learner is a weak classifier that must satisfy $W_+ - W_- \ge \gamma > 0$. Or, equivalently, it must satisfy $W_- + \frac{1}{2}W_0 \le \frac{1}{2} - \frac{\gamma}{2} < \frac{1}{2}$ because $W_+ + W_- + W_0 = 1$. Then, the goal of a weak learner will be to maximize $W_+ - W_-$ which is equivalent to minimize $W_- + \frac{1}{2}W_0$. It should be noted that these equivalent criteria avoid hypothesis such that $W_- \le W_+$ with small values for W_- and W_+ (and W_0 is high). This is consistent with the intuition that weak classifiers are not allowed to abstain on a sample of high weight.

The optimal rule given in Equation 1 for updating coefficients α_t may lead to very large or infinite values, then it can be necessary to smooth the update rule for coefficients, then leading to the update rule:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{W_+ + \epsilon}{W_- + \epsilon} \right) \tag{2}$$

where ϵ is a small real number.

2.2.2 MTAA with conservative weighting strategy.

We now consider a more conservative version of Adaboost for hypotheses that abstain proposed in [Schapire and Singer, 1999] with a modified update weighting rule as defined in Theorem 2. We have the following result:

Theorem 2. *Let us consider* `MTAA` *with the new update rule*

$$D_{t+1}(e_i) = \frac{D_t(e_i) \exp(-\alpha_t y_i h_j^t(x_i))}{Z_t} \qquad \textit{if } h_j^t(x_i) \neq 0 \tag{3}$$

$$= \frac{D_t(e_i)(\frac{exp(\alpha)+exp(-\alpha)}{2})}{Z_t} \qquad \textit{otherwise,} \tag{4}$$

and let us suppose that there exists $\gamma > 0$ such that, at each boosting iteration, $W_- + \frac{1}{2}W_0 \leq \frac{1}{2} - \gamma$, then

(i) $error(H) \leq \prod_{t=1}^{T} Z_t$,

(ii) Z_t is minimized by choosing

$$\alpha_t = \frac{1}{2} \ln \left(\frac{W_+ + \frac{1}{2}W_0}{W_- + \frac{1}{2}W_0} \right), \tag{5}$$

(iii) $\prod_{t=1}^{T} Z_t \leq e^{-2T\gamma^2}$.

The proof is not given because it is very similar to the proof of Theorem 1.

2.2.3 Discussion on the weighting strategies.

Let us compare the two weighting strategies implied by Theorems 1 and 2 when the number of tasks N increases. Every `2T-stump` must abstain for every instance on $N - 2$ tasks, which leads to larger values of W_0. Accordingly, the quantities Z_t and $\prod_{t=1}^{T} Z_t$ become close to 1. Moreover, with the weighting strategy given by Theorem 2, large values of W_0 may lead to values of α_t close to 0 and the normalization factor Z_t converges more quickly to 1. Finally, let us denote by Z_1

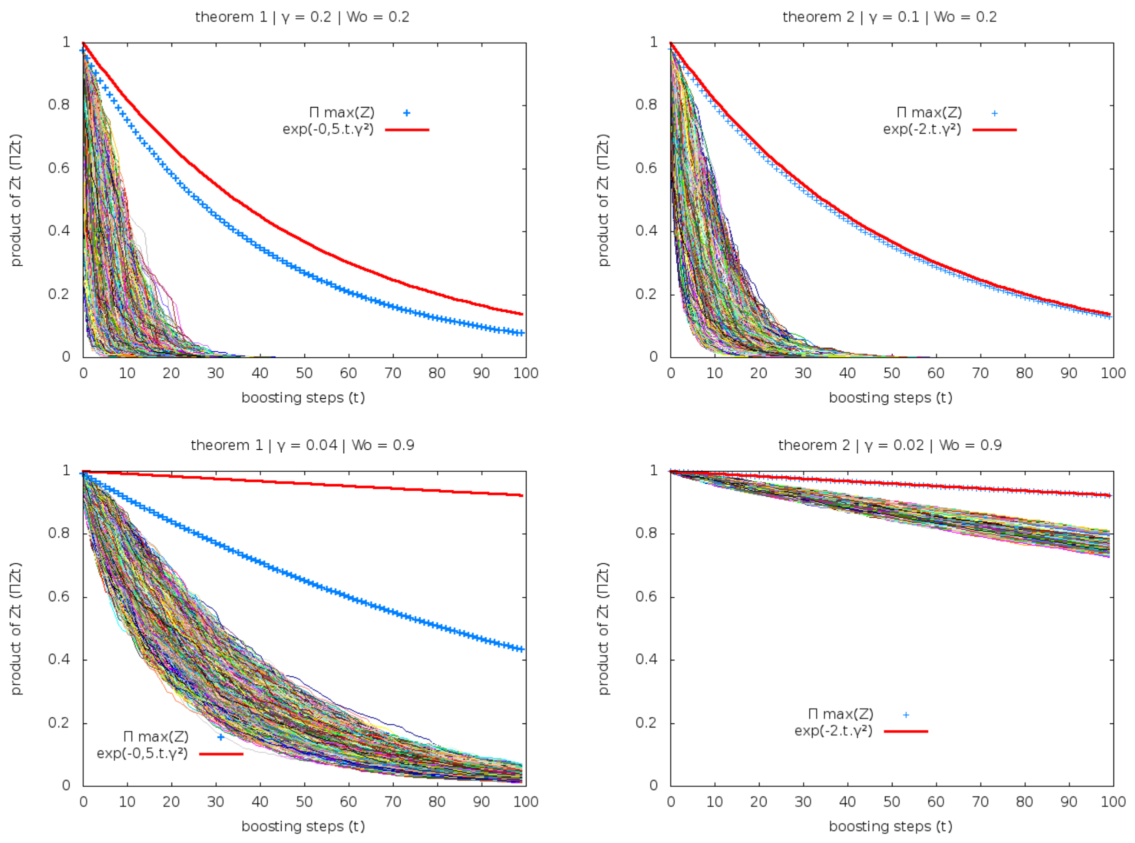


Figure 3: Evolution of $\prod_{t=1}^{T} Z_t$ following the two possible strategies for different values of W_0: Theorem 1 (left) and Theorem 2 (right); $W_0 = 0.2$ (up) and $W_0 = 0.9$ (down).

and Z_2 the normalization factors according to Theorems 1 and 2, we can note that Z_1 is always smaller than Z_2 because $Z_2^2 - Z_1^2 = 2W_0(\sqrt{W_+} - \sqrt{W_-})^2$.

We show in Figure 3 random trajectories of $\prod_{t=1}^{T} Z_t$ with fixed values of W_0 and γ. For small values of W_0, the two weighting strategies are similar. But with a greater value of W_0, we obtain very different results for the second strategy: the trajectories are close to $y = 1$. Let us also note that the theoretical bound for the first strategy on the training error is not precise for small values of the edge γ induced by great values of W_0 and we have to directly minimize $\prod_{t=1}^{T} Z_t$ in order to control the training error.

We conclude the section with a note on the number of boosting iterations. When the number N of tasks increases, the edge γ is small, thus we have to increase the number of boosting iterations: if T is the number of boosting iterations for a single task boosting algorithm, we suggest to use MTAA with a number of boosting iterations equal to $\frac{N.T}{2}$. And, we recall that the goal of a weak learner is to output hypotheses maximizing $W_+ - W_-$ (or equivalently minimizing $W_- + \frac{1}{2}W_0$).

In the sequel of the paper, we will only consider the weighting strategy implied by Theorem 1.

2.3 Multi-task Weak Learners

A weak learner WL takes as input a sample S and a probability distribution D over S and returns a 2T-stump h. The objective of WL is that the score $W_- + \frac{1}{2}W_0$ of the output h w.r.t. S and D is as small as possible. We suppose that a set of tests has been previously computed according to attributes and values observed in the sample S. For sake of clarity, we will suppose that the tests are binary. This defines a set of stumps where every stump is defined by a test, a task and two decision nodes.

A first naive weak learner compute the score of all 2T-stumps and then select the best 2T-stump. But the complexity is cubic in the number of tests. A second naive algorithm is to select at the first level the stump with the best score and then greedily select, at each of the two decision nodes, the stump with the best score. It can be easily shown that such an algorithm does not output the best, at least a good, 2T-stump because the algorithm does not take into account the correlation between tasks. For instance, a greedy algorithm will learn the 2T-stump shown in Figure 1(b) on a synthetic dataset visualized in Figure 2(b) because the best first-level decision stump is obtained for task T_3 while a less naive algorithm, as defined below, will find the 2T-stump shown in Figure 1(a) and visualized in Figure 2(a). Therefore we now introduce weak learners with the aim of finding a good 2T-stump while avoiding the cubic complexity.

Let S be a (multi-task) sample and W be (a vector of) weights for examples in S. We define a procedure Score which assigns to every stump *st* a score w.r.t. S and W. The score will be chosen equal to $W_- + \frac{1}{2}W_0$. We define functions Best and Best-per-Task as follows. Given a sample S and weights W, Best takes as input a positive integer K and outputs the set of K stumps with highest score; Best-per-Task takes as input an integer K and output, for every task *task*, the set of K stumps for *task* with highest score. Also, we can define a distribution over stumps where the probability of a stump is inversely proportional to its score. Then, the function Sto-Best takes as input an integer K and output a list of K stumps drawn randomly according to this distribution.

First, we define the weak learner WL-Best-K in Algorithm 2. Given a score function, the K stumps with the best scores are chosen as candidates for the root. Let us suppose that the chosen stump *st* corresponds to task j. Then, we consider the set $S \setminus S_j$ and descend elements according to the test defined by *st*. This defines the sets S_1 and S_2. We define the weights of examples to be equal to the probabilities given by D. For the second level, we choose the best stump for each branch. Note that if we consider the score defined by $W_- + \frac{1}{2}W_0$, we can show that this implies to choose the best `2T-stump` with root stump st. Then `WL-Best-K` output the `2T-stump` with the best score among the K candidate `2T-stumps`.

```
Require: $S = \cup_{j=1}^{j=N}\{e_i =< x_i, y_i, j >\}$ and a distribution $D$ on $S$; parameter $K$
1: Compute $BeST =$ Best($K$) w.r.t. $S$, $D$ {choose $K$ root stumps}
2: for every stump $st$ in $BeST$ do
3:     Compute $S_1$ and $S_2$ {descend examples according to the root test}
4:     Let $W_1 = D|_{S_1}$; $st_1 =$ Best(1) w.r.t. $S_1$, $W_1$ {the best stump for the left child}
5:     Let $W_2 = D|_{S_2}$; $st_2 =$ Best(1) w.r.t. $S_2$, $W_2$ {the best stump for the right child}
6:     Let $h_{st}$ be the 2T-stump with root $st$, left child $st_1$ and right child $st_2$
7: end for
8: return 2T-stump $h$ with the best score among all $h_{st}$ for $st$ in $BeST$
```

Algorithm 2: Weak learner `WL-Best-K`

The weak learner `WL-Best-K` chooses the K best stumps as candidates for the root of the output `2T-stump`. It may be the case that these K best stumps concern only some of the tasks because slight variations of a test can lead to slight variations of the score. Therefore, we also consider the weak learner `WL-Best-per-Task-K` obtained by replacing the instruction $BeST =$ `Best`(K) in line 1 of `WL-Best-K` by $BeST = \cup_{j=1}^{N}$`Best-per-Task`(k). Let K be an integer and let us consider $K = k \times N$, where N is the number of tasks. The weak learner `WL-Best-per-Task-K` with parameter value K will ensure that among the K stumps chosen at the root, for every task, K stumps are chosen. Thus, all tasks are considered when computing a hypothesis `2T-stump` with `WL-Best-per-Task-K` which was not the case for `WL-Best-K`.

It can be shown that neither of these algorithms is ensured to output an optimal hypothesis because the choice of root stumps is made independently of the choice of the second-level stumps. And it can be the case that the best `2T-stump` has a root stump which is not in the K-best stumps. Thus, we also consider the idea to introduce diversity in the choice of the candidate root stumps. For this, we define the weak learner `WL-Sto-Best-K` by replacing the instruction $BeST =$ `Best`(K) in line 1 of `WL-Best-K` by $BEsT =$ `Sto-Best`(K).

The three weak learners are not optimal. But, the complexity of `WL-Best-K` and `WL-Best-per-Task-K` is in $O((N \times Tt \times k)$ where N is the number of tasks, Tt the number of tests and K is the parameter value. The complexity of `WL-Sto-Best-K` must include also a logarithmic factor. We compare empirically the weak learners in the next section.

3 Empirical Studies

First, we describe in detail the datasets used, the preprocessing applied on them. Second, we compare the three weak learners `WL-Best-K`, `WL-Best-per-Task-K` and `WL-Sto-Best-K`. Third, we study the influence of the number of boosting rounds on our algorithms `MTAA`. Last, we report our experimental comparisons between `MTAA` with the closest prior art multi-task algorithm `MTL` [Quadrianto et al., 2010]. Last, we give the performance of `MTAA` on the ENRON dataset.

3.1 Datasets

3.1.1 MNIST.

We use this dataset adapted to the multi-task setting because it was used in [Quadrianto et al., 2010] and we follow their protocol. For the experiments, we consider multi-task learning problems with a number of tasks equal to 5, 7 or 10. We consider digits $\{6, 7, 8, 9, 0\}$, $\{4, 5, 6, 7, 8, 9, 0\}$ and $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$ for the 5-task, 7-task and 10-task problems, respectively. Every task is a binary classification task. For instance, in the 5-task problem, the first task has binary labels $\{+1, -1\}$, where label $+1$ means digit 6 and label -1 means digit 7, 8, 9 or 0; for the second task, label $+1$ means digit 9 and label -1 means other digits; and so on for other tasks. Similar one-against-all setting is also used for 7-task and 10-task problems. We use a small subset of the whole sample as training set to simulate the situation when we only have limited number of labeled examples. In the experiments, we draw 5 learning sets whose size is recalled in the tables according to the choices done in [Quadrianto et al., 2010]. We present the average accuracy results over the 5-random runs where the accuracy is estimated on the fixed test set defined by the dataset creators.

3.1.2 Enron.

Lawsuits involving companies and/or individuals have huge collections of documents varying from hard copy official documents to emails. A group of lawyers are engaged to mine those collections of millions of documents in order to decide which ones are *responsive* for a certain lawsuit. Case mining is costly, time consuming and critical since a single document might have an impact on the lawsuit. This kind of legal document collections is not easily available and even if they were, they would require considerable annotation efforts due to their huge size. To the best of our knowledge the Enron dataset[1] is the most known dataset of this kind and it is widely used by the machine learning community [McCallum et al., 2007, Shetty, 2005] and [Bekkerman et al.,]. It contains all e-mails sent and received by some 150 accounts of the top management of Enron and spanning a period of several years

Annotations of the Enron dataset come from two different sources. The first is from the Department Of Justice of the United States DOJ[2], 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*. Since the two sets are small, and they share a common knowledge (ex. a personal email is not likely to be a responsive email), so learning them simultaneously would be advantageous. It should be noted, that those two sets are disjoint, i.e., there are no examples provided with both annotations.

We used the textual features of Enron dataset along with the social features generated from the underlying social network (for more details, see [Hovelynck and Chidlovskii, 2010]. The main task is to discover responsive documents. We will try to improve performance on this task by considering the multi-task setting by also considering tasks from the topic annotated set: *Legal* Vs. *Personal* and *Business* Vs. *Arrangement*.

3.2 Weak Learners Comparison

First, we have done experiments to compare the weak learners independently of the boosting algorithms. For this, we have considered the datasets MNIST-5, MNIST-7 and MNIST-10 and a fixed distribution over instances. Here, we only report our conclusions:

`WL-Best-K` is the simplest weak learner. When the parameter value K is not large enough, `WL-Best-K` may not find an optimal `2T-stump`. When the number of tasks increases, one may need a large value of K. For instance, $K = 30$ is enough for MNIST-5 while $K = 100$ is required for MNIST-10. This is because a greater value of K is necessary for every task to appear in a candidate root stump. Nevertheless, the output `2T-stump` is always not far from optimal;

[1]http://www.cs.cmu.edu/~enron/

[2]http://www.usdoj.gov/enron/

Tasks	Train (Test)	`MTAA`-30Best	`MTAA`-3Best-per-task	`MTAA`-30StoBest
1/-1	100 (10000)	94.15 ± 0.98	94.73 ± 0.88	93.95 ± 1.64
2/-2	100 (10000)	86.45 ± 1.55	86.01 ± 1.85	86.52 ± 1.08
3/-3	100 (10000)	84.60 ± 1.47	86.71 ± 1.62	86.94 ± 0.97
4/-4	100 (10000)	88.02 ± 1.25	86.19 ± 1.11	85.79 ± 1.32
5/-5	100 (10000)	83.36 ± 1.05	81.84 ± 1.47	84.01 ± 1.89
6/-6	100 (10000)	92.86 ± 0.86	92.92 ± 1.22	93.21 ± 1.96
7/-7	100 (10000)	91.98 ± 0.91	91.62 ± 1.69	90.11 ± 0.98
8/-8	100 (10000)	82.66 ± 1.74	80.73 ± 1.78	83.98 ± 1.67
9/-9	100 (10000)	84.27 ± 1.49	84.80 ± 1.58	84.56 ± 0.73
0/-0	300 (10000)	96.44 ± 0.46	96.55 ± 0.39	95.78 ± 0.37
Avg		88.48	88.21	88.49

Table 1: Comparison on the dataset MNIST-10 of `MTAA` with the weak learners `WL-Best-K` with $K = 30$, `WL-Best-per-Task-K` with $K = 3$, and `WL-Sto-Best-K` with $K = 30$.

`WL-Best-per-Task-K` find the optimal `2T-stump` in a large number of cases even with small values of the parameter K (the number of optimal root stumps per task). For instance $K = 5$ gives optimal results for all MNIST datasets.

`WL-Sto-Best-K` allows to introduce diversity in the choice of the root stumps, but with small values of K (the number of root stumps drawn randomly), it fails to output an optimal `MTAA`. Moreover, the output `2T-stump` has often a lower score than the `MTAA` output by `WL-Best-K` for the same parameter value K.

`WL-Best-per-Task-K` is the more robust w.r.t. the choice of the parameter value. But `WL-Best-K` is the simplest and it often output good hypotheses. Thus, we also compare the weak learners when used in the `MTAA` algorithm. We give experimental results on the MNIST-10 dataset in Table 1. They show no significant differences between the weak learners when used in `MTAA`. For instance `WL-Best-K` with $K = 30$ gives good results when used in `MTAA` although non optimal as an independent weak learner.

3.3 Varying the Number of Boosting Iterations

We consider our algorithm `MTAA` with `WL-Best-K` chosen as weak learner with a parameter value chosen to be 30. We consider the MNIST-10 dataset and we let vary the number of boosting iterations. The experimental results are given in Table 2. They show that the accuracy increases with the number of boosting iterations as announced in Section 2.2.

Tasks	Train (Test)	$T = 100$	$T = 200$	$T = 500$	$T = 1000$
1/-1	100 (10000)	93.75 ± 1.31	94.15 ± 1.42	96.43 ± 0.78	96.24 ± 0.88
2/-2	100 (10000)	86.21 ± 1.65	86.45 ± 1.48	85.33 ± 0.54	84.52 ± 1.01
3/-3	100 (10000)	81.50 ± 2.31	84.60 ± 1.03	85.09 ± 0.89	85.39 ± 0.95
4/-4	100 (10000)	87.45 ± 2.61	88.02 ± 1.15	88.24 ± 1.25	88.75 ± 1.21
5/-5	100 (10000)	81.33 ± 1.11	83.36 ± 1.15	81.04 ± 2.46	82.2 ± 1.76
6/-6	100 (10000)	92.29 ± 2.86	92.86 ± 1.16	94.06 ± 1.25	94.14 ± 1.32
7/-7	100 (10000)	88.67 ± 1.32	91.98 ± 0.93	90.27 ± 0.64	90.29 ± 1.03
8/-8	100 (10000)	81.88 ± 2.2	82.66 ± 1.81	85.1 ± 0.97	85.13 ± 1.61
9/-9	100 (10000)	83.1 ± 1.56	84.27 ± 1.29	86.49 ± 1.82	86.47 ± 0.68
0/-0	300 (10000)	94.98 ± 1.01	96.44 ± 0.66	95.58 ± 0.41	95.57 ± 0.38
Avg		87.12	88.48	88.76	88.87

Table 2: Experimental results on the dataset MNIST-10 with `MTAA` when varying the number T of boosting iterations. The weak learner used is `WL-Best-K` with $K = 30$

3.4 Comparison between MTAA and MTL

We compare `MTAA` with `MTL` defined in [Quadrianto et al., 2010] on the MNIST datasets. We also compare `MTAA` with (single-task) Adaboost. For `MTL`, we take the results from the paper. For Adaboost, we fix the number of boosting iterations to be $T = 100$. For `MTAA`, we fix the number of boosting iterations to be $T = 500$ and, as before we choose the weak learner `WL-Best-K` with K set to 30. For the initial distribution of Adaboost, we balance the probability mass between classes, which means that the probability of sampling an instance from a certain class is the same across classes. For `MTAA`, since we learn many tasks simultaneously, the probability mass is balanced between each pair (task, class).

The experimental results are presented in Table 3. Statistical significant improvements (according to `t-test` with $\alpha = 0.05$) are shown in bold face and they show that `MTAA` outperforms both Adaboost and `MTL`. It is also worth noting that the standard deviation of accuracy results is lower for `MTAA` than for `MTL`, which shows the stability of boosting methods across the different runs.

3.5 MTAA on the ENRON Dataset

We consider in this paper the Enron dataset because it is a real world large scale dataset and it is associated with difficult learning tasks because the number of annotated examples is low. The different learning tasks have been defined independently by different communities. We consider three tasks: the case mining task, i.e. responsive Vs. non-responsive; a topic task legal Vs. personal; and another topic task business Vs. arrangement). The number of tasks is small but the tasks are difficult enough to study the performance of `MTAA` on this 3-task learning problem.

Since there are no available multi-task results on the Enron dataset, we compare `MTAA` with (single-task) Adaboost. No test set is available thus accuracy is

Tasks	Train (Test)	Adaboost	`MTL`	`MTAA`
MNIST-3				
6/-6	25 (4949)	89.84 ± 0.37	83.86 ± 9.51	91.56 ± 3.21
7/-7	25 (4949)	85.25 ± 2.35	72.84 ± 15.77	83.35 ± 1.22
8/-8	25 (4949)	81.73 ± 3.21	66.77 ± 9.43	84.11 ± 2.02
9/-9	25 (4949)	73.215 ± 6.51	67.26 ± 12.65	76.85 ± 2.11
0/-0	150 (4949)	96.43 ± 0.28	96.60 ± 1.64	97.29 ± 0.62
Avg	-	85.29	77.74	**86.63**
MNIST-5				
4/-4	70 (6823)	86.11 ± 1.071	73.49 ± 6.77	87.52 ± 1.46
5/-5	70 (6823)	83.99 ± 2.92	70.10 ± 4.61	86.26 ± 1.03
6/-6	70 (6823)	92.23 ± 1.01	87.21 ± 2.77	93.02 ± 1.41
7/-7	70 (6823)	$87.97 \pm 0.0.51$	84.02 ± 3.69	90.05 ± 2.01
8/-8	70 (6823)	88.32 ± 0.13	76.97 ± 5.12	87.63 ± 1.38
9/-9	70 (6823)	78.09 ± 1.33	65.74 ± 10.15	80.31 ± 1.38
0/-0	210 (6823)	96.00 ± 0.81	96.56 ± 1.67	96.12 ± 0.61
Avg	-	87.53	79.16	**88.70**
MNIST-10				
1/-1	100 (10000)	94.62 ± 1.23	96.80 ± 1.91	96.43 ± 0.78
2/-2	100 (10000)	85.72 ± 0.58	69.95 ± 2.68	85.33 ± 0.54
3/-3	100 (10000)	85.71 ± 0.99	74.18 ± 5.54	85.09 ± 0.89
4/-4	100 (10000)	88.31 ± 0.64	71.76 ± 5.47	88.24 ± 1.25
5/-5	100 (10000)	82.34 ± 2.11	57.26 ± 2.72	81.04 ± 2.46
6/-6	100 (10000)	91.28 ± 0.4	80.54 ± 4.53	94.06 ± 1.25
7/-7	100 (10000)	90.20 ± 0.50	77.18 ± 9.43	90.27 ± 0.64
8/-8	100 (10000)	81.66 ± 2.13	65.85 ± 2.50	85.1 ± 0.97
9/-9	100 (10000)	81.42 ± 0.38	65.38 ± 6.09	86.49 ± 1.82
0/-0	300 (10000)	96.85 ± 0.35	97.81 ± 1.01	95.58 ± 0.41
Avg	-	87.77	75.67	**88.76**

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

Tasks	Train (Test)	Adaboost	`MTAA`
Responsive Vs. NonResponsive	299 (74)	90.49 ± 0.90	90.99 ± 2.74
Legal Vs. Personal	265 (66)	83.90 ± 0.75	84.44 ± 4.90
Business Vs. Arrangement	615 (153)	71.69 ± 1.5	74.32 ± 3.54
Avg		82.03	**83.25**

Table 4: Comparison on the Enron dataset of (single-task) Adaboost and `MTAA`

estimated over 3-runs of 5-fold cross validation. For `MTAA`, the number of boosting iterations is set to 300 while the number of boosting iterations for Adaboost is set to 100. The weak learner used in `MTAA` is again `WL-Best-K` with K set to 9. The experimental results are shown in Table 4 in which statistical significant improvements are shown in bold face. The results on Enron emphasizes the claimed advantage behind learning multiple related tasks together.

4 Conclusion

For the multi-task setting, we introduced `2T-stumps` as weak classifiers that abstain and we defined weak learners. We adapted Adaboost and defined `MTAA` as a multi-task learning algorithm. We gave empirical evidence that `MTAA` achieves good results and allows to capture relations between tasks without explicit priors. This is when the number of tasks remains between 2 and 20. We think that more empirical validation and more theoretical work is needed in the case of larger number of tasks. Also, we should relate our work on Adaboost for multi-task learning with the recent work of Mukherjee and Schapire [Mukherjee and Schapire, 2010] on multi class boosting.

References

[Bekkerman et al.,] Bekkerman, R., Mccallum, A., and Huang, G. Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. In *Technical Report, Computer Science department, IR-418*, pages 4–6.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. In *Machine Learning*, volume 28, pages 41–75.

[Chapelle et al., 2010] Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., and Tseng, B. (2010). Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD inter-*

national Conference on Knowledge Discovery and Data Mining (KDD), pages 1189–1198.

[Dai et al., 2007] Dai, W., Yang, Q., Xue, G. R., and Yu, Y. (2007). Boosting for transfer learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM.

[Evgeniou and Pontil, 2004] Evgeniou, T. and Pontil, M. (2004). 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.

[Faddoul et al., 2010] Faddoul, J. B., Chidlovskii, B., Torre, F., and Gilleron, R. (2010). 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.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 148–156.

[Hovelynck and Chidlovskii, 2010] Hovelynck, M. and Chidlovskii, B. (2010). Multi-modality in one-class classification. In *Proceedings of the 19th international conference on World wide web (WWW)*, pages 441–450.

[Jalali et al., 2010] Jalali, A., Ravikumar, P., Sanghavi, S., and Ruan, C. (2010). A Dirty Model for Multi-task Learning. In *NIPS*.

[Liu et al., 2009] Liu, Q., Liao, X., Carin, H. L., Stack, J. R., and Carin, L. (2009). Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1074–1086.

[McCallum et al., 2007] McCallum, A., Wang, X., and Emmanuel, A. C. (2007). Topic and role discovery in social networks with experiments on enron and academic email. *J. Artif. Int. Res.*, 30(1):249–272.

[Mukherjee and Schapire, 2010] Mukherjee, I. and Schapire, R. (2010). A theory of multiclass boosting. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS)*.

[Quadrianto et al., 2010] Quadrianto, N., Smola, A., Caetano, T., Vishwanathan, S., and Petterson, J. (2010). Multitask learning without label correspondences. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1957–1965.

[Schapire and Singer, 1999] Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37.

[Shetty, 2005] Shetty, J. (2005). Discovering important nodes through graph entropy: The case of enron email database. In *Proceedings of the 3rd international workshop on Link discovery*, pages 74–81.

[Wang et al., 2009] Wang, X., Zhang, C., and Zhang, Z. (2009). Boosted multi-task learning for face verification with applications to web image and video search. In *Proceedings of the 22nd IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 142–149.

[Zhang and Schneider, 2010] Zhang, Y. and Schneider, J. (2010). Learning Multiple Tasks with a Sparse Matrix-Normal Penalty. In *NIPS*.

[Zheng et al., 2008] Zheng, V. W., Pan, S. J., Yang, Q., and Pan, J. J. (2008). Transferring multi-device localization models using latent multi-task learning. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 1427–1432. AAAI Press.