Regularizing AdaBoost with validation sets of increasing size

Dirk W.J. Meijer and **David M.J. Tax**
Delft University of Technology
Delft, The Netherlands
d.w.j.meijer@student.tudelft.nl, d.m.j.tax@tudelft.nl

***Abstract*—AdaBoost is an iterative algorithm to construct classifier ensembles. It quickly achieves high accuracy by focusing on objects that are difficult to classify. Because of this, AdaBoost tends to overfit when subjected to noisy datasets. We observe that this can be partially prevented with the use of validation sets, taken from the same noisy training set. But using less than the full dataset for training hurts the performance of the final classifier ensemble. We introduce ValidBoost, a regularization of AdaBoost that takes validation sets from the dataset, increasing in size with each iteration. ValidBoost achieves performance similar to AdaBoost on noise-free datasets and improved performance on noisy datasets, as it performs similar at first, but does not start to overfit when AdaBoost does.**

Keywords: Supervised learning, Ensemble learning, Regularization, AdaBoost

I. Introduction

In 1997 Freund and Schapire introduced AdaBoost ([1], algorithm 1), a highly successful, iterative ensemble method that adapts to difficult objects by re-weighting based on correct or incorrect classification in earlier iterations. A problem with this method is that it might overfit when run for too many iterations, i.e. the performance on an unseen test set may deteriorate compared to earlier iterations ([2], [3]). Although this point is heavily debated (see [4] for example), the presence of mislabeled training objects seems to increase the likelihood of overfitting drastically ([5], [6]).

A common method to avoid overfitting a classifier is to use a holdout sample as validation set to estimate to actual error a method is making, instead of relying only on the training error. An extension of this is cross-validation, we rotate the set in such a way that every training object is in a holdout sample exactly once, and we average the obtained validation errors. This allows you to still use the entire training set, while still relying on a holdout sample for validation.

Bylander and Tate ([7]) incorporated validation sets in AdaBoost for the same reasons. In their method they perform a stratified split of the training data once, train AdaBoost twice, using both sets as training set and validation set in turn and combining the results of both into a single ensemble classifier. They show very promising results, especially in the presence of label noise. It should be noted however, that an unlucky initial split can cause problems.

Torres-Sospedra et. al ([8]) go a little further and perform k-fold cross-validation within AdaBoost. Their method of implementing it is somewhat curious however, the number of folds and the number of iterations is equal by definition and each iteration uses a different validation fold. This limits its application, as we have to keep this in mind when choosing a number of iterations to run AdaBoost for, and particularly small datasets will limit this method to a low number of iterations. This method was also made to work specifically with neural networks, which may explain why they have no need for a large number of base classifiers, as neural networks are already a relatively strong classifier.

Algorithm 1: AdaBoost

```
   Input: Dataset x, consisting of $N$ objects
          $\langle x_1, \dots, x_N \rangle \in X$ with labels
          $\langle y_1, \dots, y_N \rangle \in Y = \{1, \dots, c\}$
   Input: Weak learning algorithm WeakLearn
   Input: Number of iterations $T$
   Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1, \dots, N$
 1 for $t = 1$ to $T$ do
 2     Call WeakLearn, providing it with weights $w_i^t$
 3     Get back hypothesis $h_t : X \to Y$
 4     Compute $\epsilon = \sum_{i=1}^{N} w_i^t [h_t(x_i) \neq y_i]$
 5     if $\epsilon > \frac{1}{2}$ then
 6         Set $\alpha_t = 0$
 7         Set $w_i^{t+1} = w_i^t$
 8     else
 9         Compute $\alpha_t = \log((1-\epsilon)/\epsilon)$
10         Set $w_i^{t+1} = w_i^t \exp(\alpha_t [h_t(x_i) \neq y_i])$ for all $i$
11         Normalize $w_i^{t+1}$
12     end
13 end
   Output: Hypothesis $H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \alpha_t [h_t(x) = y]$
```

The exact reason for overfitting is not entirely clear. It has been attributed to the fact that outliers get extremely large weights and the choice of the exponential loss function, which has spawned methods such as LogitBoost [2] and DOOM II [9], that regularize the way weights are assigned in one way or another. While it may certainly be true that the weights play a role, we believe that a large part of the problem is that fact that the error estimation for the classifier weights (line 4 in algorithm 1) is biased.

In this paper we investigate the overfitting behavior of

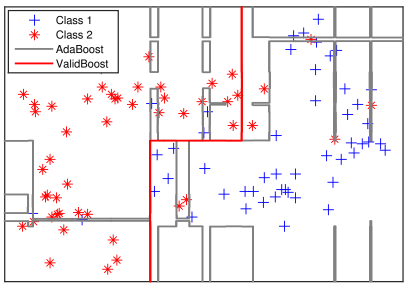

Fig. 1: Two-dimensional, two-class, banana-shaped dataset with 10% noise

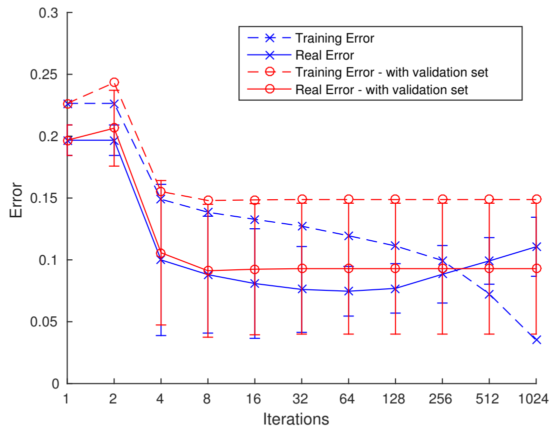

Fig. 2: Behavior of AdaBoost and the altered algorithm with an unbiased error estimator. Averaged over 20 trials, with the standard deviation in error bars

AdaBoost. To this end we start in section II by showing that overfitting behavior can be suppressed by good error estimates ε. In section III a practical algorithm ValidBoost is proposed to avoid this overfitting. In section IV ValidBoost is compared to AdaBoost and a more robust versions of boosting, and the paper is concluded in section V with some discussion.

II. AdaBoost overfitting

To illustrate the overfitting behavior of AdaBoost, we look at an artificial example. A two-dimensional two-class banana-shaped dataset with 10% added label noise is used, as shown in Figure 1. The 10% label noise is added to this dataset by picking 10% of the objects at random and changing the label to that of the opposite class. We run AdaBoost for a large number of iterations (1024), using decision stumps as the base classifiers, and generate a large test set from the same distribution as the (noise-free) training set to assess the classification error made by AdaBoost.

The results are shown in Figure 2. The dashed blue graph shows the error on the (noisy) training set, as a function of the number of boosting iterations. As can observed, this error decreases almost to zero, suggesting that AdaBoost is overfitting

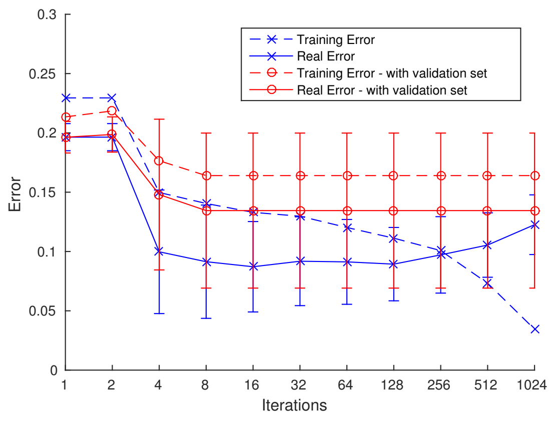

Fig. 3: Behavior of AdaBoost and the altered algorithm with half the dataset as training data, the other half as validation set. Averaged over 20 trials, with the standard deviation in error bars

on the training set. The solid blue graph, indicating the true (noise-free) test error, confirms this. This error increases with increasing number of iterations. Note that due to the noise in the training set, the error on the training set is a bit higher than on the test set.

We then make a small change to AdaBoost, we use the same training set to train the base classifiers, but now we use the test set to calculate the error ϵ and assign weights α_t to the base classifiers. The results are indicated by the red graphs in Figure 2. Clearly, the overfitting is avoided. This is highly artificial of course, but it shows what would happen if we had a perfect, unbiased estimate of the error a base classifier makes.

As we can see in figure 2, having a better estimate of the error a classifier make can certainly stop AdaBoost from overfitting. This specific example relies on having a noise-free validation set, which is an unfeasible requirement in practically all real-world cases. More realistically, we can split the noisy training set into two sets, and use one set to train the base classifiers and the other set to calculate ϵ and α. The results of this experiment on our same banana-shaped set are shown in figure 3. While this method seems to be effective in preventing overfitting, it does not give an error rate remotely as good as AdaBoost does. One obvious reason for this, is that we are using only half of our dataset.

III. ValidBoost

AdaBoost is known to reach high accuracy very quickly, which can also be observed in our graphs. To get the best of both worlds, we propose as next step to gradually increase the influence of our validation set. Initially, we want our method to be identical to AdaBoost, so that it can reach low error levels, but over time, we want to introduce a validation set (taken from our training set) to avoid overfitting like we did in the last few experiments. The question then becomes, how gradually?

We observe that the biggest changes occur in the first few iterations of AdaBoost. This is expected, as each iteration adds

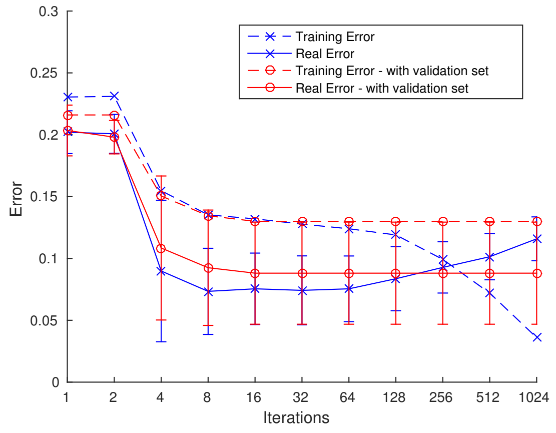

Fig. 4: Behavior of AdaBoost and the altered algorithm with half the dataset as training data, the other half as validation set, with logarithmically increasing influence of the validation set. Averaged over 20 trials, with the standard deviation in error bars.

a single classifier to a growing ensemble of classifiers. At iteration t, the 'impact' of a new classifier on the current ensemble is $1/t$. Because of this, we suggest a logarithmic increase for the introduction of the validation set, to take full advantage in the earlier iterations, when the impact of each iteration is larger. We define

$$\tau = \frac{\log t}{\log T} \quad (1)$$

$$\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t \quad (2)$$

Where t is the current iteration, T is the total number of iterations, ϵ_v is the weighted error on the validation set and ϵ_t is the weighted error on the training set. In the first few iterations, τ will be small and we will be relying heavily on the training error, akin to vanilla AdaBoost ([1]). As t increases, τ will tend towards 1 and the influence of the validation will quickly become larger, in order to avoid overfitting on the training set. The performance with this change on the banana-shaped dataset are shown in figure 4.

These results are again an improvement, but one detail has not changed, we are still using only half of the training set, even in the initial iterations where we are hardly using the validation set at all. This brings us to the final step in our train of thought, we change the size of the validation set, linear in τ. At the start of each iteration, we perform a stratified random split of the training set $\mathbf{x}$ into a validation set $\mathbf{x}_v$ of size $\tau N/2$, and the remainder, $\mathbf{x}_t$, which will be used as training set for this particular iteration. As such, we start out with an empty validation set and a full training set, and end up with a 50/50 split between the two at iteration T.

The final algorithm is shown in algorithm 2. The main difference with algorithm 1 is that it splits the training set into a new training set $\mathbf{x}_t$ and validation set $\mathbf{x}_v$ (line 3), and that the error ϵ is now a weighted combination of the training error ϵ_t and validation error ϵ_v.

The performance on the banana-shaped dataset can be seen in figure 5. It can be observed that ValidBoost converges as fast as AdaBoost to a low error, but in contrast to AdaBoost, the true error does not increase.

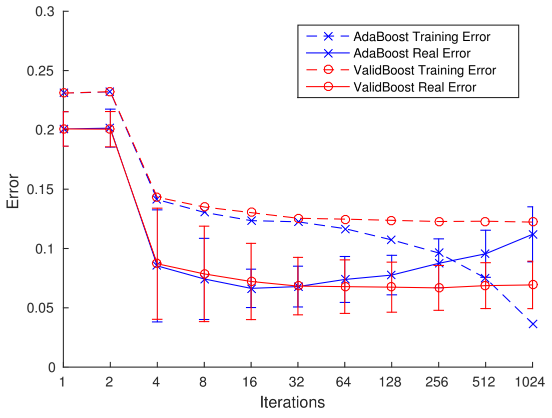

Fig. 5: Behavior of AdaBoost and ValidBoost. Averaged over 20 trials, with the standard deviation in error bars.

Algorithm 2: ValidBoost

```
    Input: Dataset $\mathbf{x}$, consisting of $N$ objects
           $\langle x_1, \ldots, x_N \rangle \in X$ with labels
           $\langle y_1, \ldots, y_N \rangle \in Y = \{1, \ldots, c\}$
    Input: Weak learning algorithm WeakLearn
    Input: Number of iterations $T$
    Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1, \ldots, N$
 1  for $t = 1$ to $T$ do
 2      Set $\tau = \frac{\log t}{\log T}$
 3      Split $\mathbf{x}$ into $\mathbf{x}_v$ of size $\tau N/2$ and $\mathbf{x}_t$
 4      Call WeakLearn on $\mathbf{x}_t$, providing it with weights
          $w_i^t$ for $\{i : x_i \in \mathbf{x}_t\}$
 5      Get back hypothesis $h_t : X \to Y$
 6      Compute $\epsilon_v = \sum_{\{i:x_i \in \mathbf{x}_v\}} w_i^t [h_t(x_i) \neq y_i]$
 7      Compute $\epsilon_t = \sum_{\{i:x_i \in \mathbf{x}_t\}} w_i^t [h_t(x_i) \neq y_i]$
 8      Set $\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t$
 9      if $\epsilon > \frac{1}{2}$ then
10          Set $\alpha_t = 0$
11          Set $w_i^{t+1} = w_i^t$
12      else
13          Compute $\alpha_t = \log((1-\epsilon)/\epsilon)$
14          Set $w_i^{t+1} = w_i^t \exp(\alpha_t [h_t(x_i) \neq y_i])$ for all $i$
15          Normalize $w_i^{t+1}$
16      end
17  end
    Output: Hypothesis $H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \alpha_t [h_t(x) = y]$
```

It has to be noted that ValidBoost has a higher reliance on the hyperparameter T than AdaBoost. If we run ValidBoost for a particularly short number of iterations, τ may increase too fast, leading to errors not as low as AdaBoost might give us. In the other extreme, if we pick a T that is too large, τ might not increase fast enough to prevent overfitting. But since τ scales with the logarithm of T, and not with T directly, these changes have to be rather extreme before an effect is observed.

IV. Experiments

In this section we compare empirical results for different methods. In all tables, the best result for each dataset is written in bold, as are the results that are not significantly worse, determined by a paired value t-test at $p = 0.05$. All classification performances are measured in accuracy and the standard deviations are shown in parentheses.

TABLE I: Comparison between scaling τ for a few UCI dataset. $T = 1024$. As base classifier a tree with depth 4 is used, and except for the first dataset, no noise has been added.

Dataset	$\log(t)/\log(T)$	t/T
Banana 20% noise	**0.244 (0.169)**	**0.294 (0.185)**
breast-cancer	**0.047 (0.037)**	**0.039 (0.035)**
ecoli	**0.145 (0.064)**	**0.141 (0.069)**
glass	**0.318 (0.186)**	**0.313 (0.185)**
heart-disease	**0.212 (0.067)**	**0.215 (0.101)**
ionosphere	**0.080 (0.044)**	**0.066 (0.050)**
iris	**0.040 (0.056)**	**0.047 (0.045)**
liver-disorders	**0.307 (0.073)**	**0.316 (0.066)**
diabetes	**0.260 (0.041)**	**0.251 (0.043)**
satimage	**0.141 (0.038)**	**0.143 (0.036)**
sonar	**0.256 (0.172)**	**0.261 (0.162)**
soybean-large	**0.086 (0.039)**	**0.086 (0.043)**
soybean-small	**0.110 (0.069)**	**0.147 (0.096)**
wine	**0.113 (0.098)**	**0.132 (0.104)**

In the first experiment, we compare the logarithmic scaling of $\tau = \log(t)/\log(T)$ with a linear scaling t/T on 13 real-world datasets from the UCI repository ([10]). See appendix A for a complete list. Table I shows the classification performances on different datasets. A fairly large number of runs is used, $T = 1024$, and a slightly more complicated decision tree, a tree with depth 4. Sometimes a bit improvement is observed using the logarithmic scaling, although the differences are never significant. We chose to use the logarithmic scaling in the rest of this paper.

TABLE II: Comparison of ValidBoost with different values for T. As base classifier decision stumps are used and no noise had been added. Results are obtained using 10-fold crossvalidation.

Datasets	$T = 10$	$T = 100$	$T = 1000$	$T = 10000$
breast-cancer	**0.05 (0.02)**	**0.04 (0.02)**	**0.05 (0.02)**	**0.05 (0.02)**
ecoli	0.31 (0.06)	**0.22 (0.08)**	**0.20 (0.07)**	**0.26 (0.08)**
glass	**0.36 (0.07)**	**0.34 (0.11)**	**0.39 (0.11)**	**0.36 (0.10)**
heart-disease	**0.17 (0.02)**	**0.16 (0.03)**	**0.19 (0.02)**	**0.16 (0.04)**
ionosphere	0.13 (0.05)	**0.08 (0.04)**	**0.07 (0.05)**	**0.07 (0.03)**
iris	**0.05 (0.07)**	**0.07 (0.07)**	**0.07 (0.08)**	**0.05 (0.08)**
liver-disorders	**0.32 (0.07)**	**0.26 (0.08)**	**0.26 (0.08)**	**0.28 (0.08)**
diabetes	**0.26 (0.04)**	**0.24 (0.03)**	**0.25 (0.04)**	**0.24 (0.05)**
satimage	0.34 (0.07)	**0.21 (0.02)**	0.23 (0.01)	**0.21 (0.02)**
sonar	0.31 (0.11)	**0.18 (0.09)**	**0.18 (0.09)**	**0.20 (0.13)**
soybean-large	0.73 (0.04)	0.33 (0.05)	**0.27 (0.06)**	0.63 (0.08)
soybean-small	**0.20 (0.13)**	**0.18 (0.09)**	**0.18 (0.08)**	**0.12 (0.11)**
wine	**0.05 (0.06)**	**0.03 (0.04)**	**0.03 (0.04)**	**0.04 (0.04)**

For AdaBoost and most of its related algorithms, stopping early at iteration t_s is identical to the situation where we would have set $T = t_s$. For ValidBoost however, the initial choice of T has some more bearing, because it determines the speed with which the size and weight of the validation sets increases. Table II shows the classification error ValidBoost makes for different choices of T on several UCI datasets with decision stumps as base classifiers. As noted earlier, setting T too low might lead to performance worse than AdaBoost and setting T too high might lead to overfitting not being prevented. We note that for $T \geq 100$, these effects are almost not observable for these datasets, and $T = 1000$ seems to be a sensible choice.

TABLE III: Classification error of the boosting algorithms using decision stumps (tree depth 1) after 1024 iterations. Results are obtained using 10-fold crossvalidation. (five times repeated).

		No noise		
Datasets	AdaBoost	LogitBoost	Bylander & Tate	ValidBoost
breast-cancer	**0.05 (0.02)**	**0.04 (0.03)**	**0.04 (0.02)**	**0.05 (0.02)**
ecoli	**0.20 (0.07)**	**0.19 (0.05)**	**0.21 (0.05)**	**0.19 (0.06)**
glass	**0.34 (0.09)**	**0.34 (0.09)**	**0.34 (0.09)**	0.39 (0.10)
heart-disease	**0.19 (0.06)**	0.20 (0.06)	**0.17 (0.06)**	**0.17 (0.07)**
ionosphere	**0.07 (0.03)**	**0.08 (0.03)**	**0.08 (0.04)**	**0.07 (0.04)**
iris	0.07 (0.05)	0.06 (0.04)	**0.04 (0.04)**	0.06 (0.05)
liver-disorders	0.29 (0.05)	0.29 (0.05)	**0.28 (0.05)**	**0.26 (0.07)**
diabetes	**0.24 (0.04)**	**0.24 (0.04)**	**0.25 (0.04)**	**0.24 (0.03)**
satimage	0.23 (0.02)	0.22 (0.02)	**0.20 (0.02)**	0.21 (0.02)
sonar	**0.12 (0.09)**	**0.12 (0.08)**	0.20 (0.08)	0.16 (0.08)
soybean-large	0.56 (0.08)	0.62 (0.09)	0.61 (0.10)	**0.33 (0.05)**
soybean-small	0.16 (0.09)	**0.12 (0.08)**	0.22 (0.11)	0.18 (0.09)
wine	0.06 (0.09)	0.07 (0.09)	0.07 (0.07)	**0.03 (0.05)**
		20% noise		
Datasets	AdaBoost	LogitBoost	Bylander & Tate	ValidBoost
breast-cancer	0.09 (0.04)	0.09 (0.04)	**0.06 (0.02)**	0.07 (0.02)
ecoli	0.33 (0.07)	0.29 (0.07)	**0.22 (0.04)**	**0.21 (0.05)**
glass	0.40 (0.10)	0.43 (0.10)	**0.34 (0.08)**	**0.33 (0.10)**
heart-disease	0.26 (0.07)	0.26 (0.06)	**0.21 (0.07)**	**0.22 (0.09)**
ionosphere	0.25 (0.06)	0.24 (0.07)	**0.12 (0.06)**	**0.13 (0.03)**
iris	0.20 (0.12)	0.18 (0.11)	0.09 (0.07)	**0.05 (0.04)**
liver-disorders	**0.32 (0.06)**	**0.32 (0.05)**	**0.34 (0.05)**	**0.34 (0.06)**
diabetes	**0.26 (0.05)**	0.27 (0.05)	**0.25 (0.03)**	0.26 (0.05)
satimage	**0.16 (0.01)**	**0.15 (0.01)**	0.17 (0.01)	0.17 (0.01)
sonar	0.34 (0.07)	0.35 (0.11)	**0.29 (0.10)**	**0.27 (0.08)**
soybean-large	0.40 (0.09)	0.33 (0.08)	**0.29 (0.07)**	0.34 (0.07)
soybean-small	0.30 (0.10)	0.29 (0.08)	**0.21 (0.13)**	**0.22 (0.12)**
wine	0.15 (0.07)	0.14 (0.06)	**0.09 (0.06)**	**0.10 (0.06)**

To assess the performance of ValidBoost, we compare it to AdaBoost*, to LogitBoost [2], and to the method by Bylander & Tate [7] on again the 13 UCI datasets. Experiments were done on a few more UCI datasets, but for some datasets no significant performance differences could be found for any setting. This happened for Abalone, Arrhythmia, and Letter-recognition. We compare performance on both the original datasets and the datasets with added label noise, as described in section I. As base classifiers we will use decision trees of varying depth, to account for different levels of complexity. All errors are calculated with 10-fold cross-validation.

In Table III the classification errors are shown for the four approaches on the collection of UCI datasets. On the top half, the original datasets are used, without noise. In the bottom

*For multi-class ($c > 2$) datasets, we use SAMME ([11]) instead of the original AdaBoost.M1, with its less strict requirements on ϵ ($\epsilon < 1 - \frac{1}{c}$ instead of $\epsilon < \frac{1}{2}$). With decision stumps as base classifiers, this is more or less a requirement, because our hypothesis will only output 2 different classes and the error of a single base classifier will never be lower than $1 - 2/c$ in a dataset with equal prior probabilities for each class. This change is applied to all other algorithms as well.

TABLE IV: Classification error of the boosting algorithms using tree depth 8 after 1024 iterations. Results are obtained using 10-fold crossvalidation (five times repeated).

		No noise		
Datasets	AdaBoost	LogitBoost	Bylander & Tate	ValidBoost
breast-cancer	**0.05 (0.02)**	**0.05 (0.02)**	**0.05 (0.02)**	**0.04 (0.03)**
ecoli	**0.14 (0.06)**	**0.14 (0.05)**	0.19 (0.06)	**0.12 (0.05)**
glass	0.28 (0.12)	0.27 (0.10)	0.28 (0.11)	**0.21 (0.08)**
heart-disease	**0.24 (0.07)**	**0.24 (0.08)**	**0.24 (0.07)**	**0.21 (0.07)**
ionosphere	0.12 (0.06)	0.12 (0.06)	**0.11 (0.06)**	**0.09 (0.05)**
iris	**0.06 (0.06)**	**0.06 (0.06)**	**0.06 (0.05)**	**0.05 (0.05)**
liver-disorders	**0.28 (0.06)**	**0.28 (0.05)**	0.31 (0.08)	**0.29 (0.06)**
diabetes	**0.25 (0.04)**	**0.25 (0.04)**	**0.24 (0.05)**	**0.26 (0.05)**
satimage	**0.07 (0.01)**	**0.07 (0.01)**	0.08 (0.01)	**0.08 (0.01)**
sonar	**0.28 (0.09)**	**0.28 (0.09)**	**0.30 (0.10)**	**0.26 (0.07)**
soybean-large	**0.08 (0.04)**	**0.08 (0.04)**	0.12 (0.06)	**0.08 (0.04)**
soybean-small	0.16 (0.09)	0.17 (0.10)	**0.16 (0.10)**	**0.11 (0.09)**
wine	**0.09 (0.06)**	**0.09 (0.05)**	**0.07 (0.06)**	**0.09 (0.06)**
		20% noise		
Datasets	AdaBoost	LogitBoost	Bylander & Tate	ValidBoost
breast-cancer	0.11 (0.04)	0.11 (0.04)	**0.09 (0.04)**	0.11 (0.04)
ecoli	**0.18 (0.06)**	**0.17 (0.07)**	**0.18 (0.06)**	**0.18 (0.05)**
glass	0.29 (0.10)	**0.27 (0.10)**	0.35 (0.12)	**0.25 (0.10)**
heart-disease	**0.27 (0.07)**	**0.27 (0.06)**	**0.30 (0.10)**	**0.28 (0.08)**
ionosphere	0.20 (0.07)	0.21 (0.08)	0.22 (0.07)	**0.16 (0.07)**
iris	**0.18 (0.10)**	0.19 (0.10)	**0.18 (0.10)**	**0.15 (0.10)**
liver-disorders	**0.36 (0.07)**	**0.38 (0.08)**	**0.37 (0.08)**	**0.35 (0.08)**
diabetes	0.30 (0.04)	0.30 (0.04)	**0.27 (0.04)**	0.31 (0.05)
satimage	**0.09 (0.01)**	**0.09 (0.01)**	0.09 (0.01)	0.10 (0.01)
sonar	0.38 (0.11)	0.38 (0.09)	0.35 (0.08)	**0.27 (0.08)**
soybean-large	0.24 (0.10)	0.25 (0.11)	**0.18 (0.10)**	**0.20 (0.09)**
soybean-small	0.28 (0.11)	0.29 (0.09)	**0.28 (0.13)**	**0.23 (0.11)**
wine	0.20 (0.11)	0.19 (0.09)	0.20 (0.10)	**0.09 (0.09)**

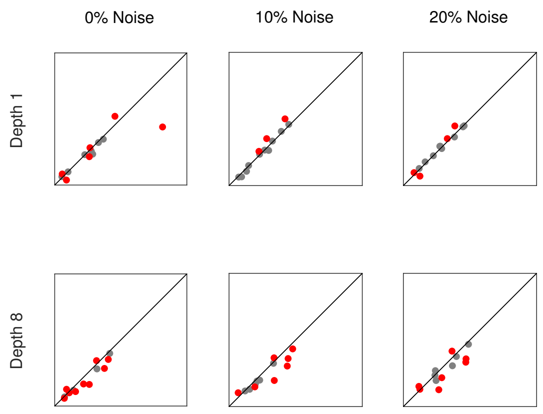

Fig. 6: Plots of ValidBoost errors (vertical axes) versus Bylander & Tate's errors (horizontal axes) on UCI datasets for different levels of noise and decision trees of different depths as base classifiers. All axes go from 0 to 0.75 error rate. Statistically significant differences are represented by the red dots.

part 20% label noise is added. The base classifier is in all situations a decision stump (or a decision tree with depth 1). Most of the original UCI datasets are relatively clean, and in most situations AdaBoost does not overfit. In these situations it does not pay off to introduce extra regularization to avoid overfitting. Only in soybean-large there is a significant improvement possible. The picture changes completely when noise is added. Then both the method of Bylander & Tate, and ValidBoost often significantly outperform AdaBoost or LogitBoost.

To see more difference between the method of Bylander & Tate and ValidBoost, the decision trees with depth 8 can be used as base classifiers. The results are shown in Table IV. Here in Sonar, Wine and Ionosphere we see significant differences. These are relatively high dimensional dataset, with a relatively small sample size. On these datasets the more flexible decision tree with depth 8 can easily overfit, though this does not guarantee that boosting these overfitted classifiers will fail, it might explain ValidBoost's improved performance on these datasets. On soybean-large one may also expect this behavior, but here the number of iterations T may be a bit too low for ValidBoost.

Figure 6 show the results of ValidBoost compared to Bylander & Tate's method [7]. The scatter plots show the pairwise errors of the two methods. When the two methods perform equally well on some UCI dataset, a dot appears on the diagonal line. We observe that especially with more complex base classifiers (decision trees at depth 8), ValidBoost compares favorably. In the other cases, neither seems to have an edge over the other.

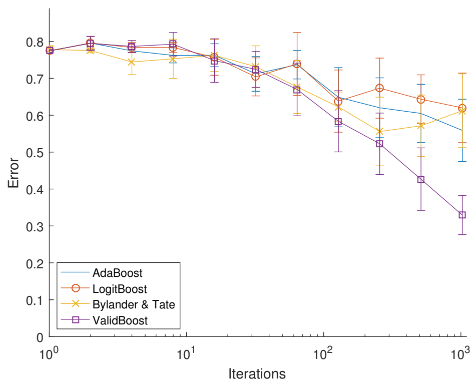

Fig. 7: Behavior on the Soybean-Large dataset with no added noise and tree depth 1.

Figure 7 shows the classification errors of the four compared algorithms on the Soybean-large dataset with no added noise and decision stumps as base classifiers at different intermediate iterations. All four methods perform similarly up to about iteration 128, after that ValidBoost keeps reducing its classification error while the other three methods seem to stagnate. It could be argued that Bylander & Tate's method even seems to start overfitting at that point, as it was initially performing better than AdaBoost and LogitBoost, but starts increasing in error again, eventually ending up with an error similar to AdaBoost and LogitBoost.

A similar but perhaps even more telling effect is observed on the Soybean-small dataset. Figure 8 shows the classification error on the Soybean-small dataset with no added noise and decision trees of depth 8 as base classifiers. When we observe AdaBoost, it seems little can be gained from boosting in this case, as after only four iterations the classification error no longer changes. This indicates that at this point, the base classifier that is found has an error in excess of the maximum

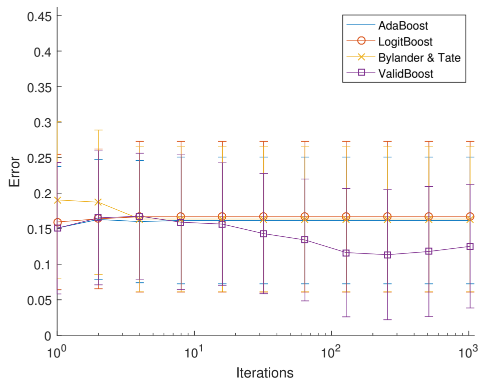

Fig. 8: Behavior on the Soybean-Small dataset with no added noise and tree depth 8.

error threshold for the multi-class case ($\epsilon > 1 - 1/c$). As such, the base classifier gets assigned a weight of zero, but in the next iteration, this exact same base classifier is found and every base classifier from this point on receives a weight of zero. ValidBoost however, keeps decreasing its error even after this point. We attribute this effect to the random sampling into training set and validation set that ValidBoost does. If a base classifier has a weight in excess of the threshold for the multi-class case, this does not mean we should stop the training process, because it is very likely we will not find the same base classifier in future iterations. In this case, we can see that this leads to a visible improvement.

V. Discussion

In this paper an adapted version of AdaBoost is proposed, that avoids the overfitting of AdaBoost. It has been shown that poor estimates of the error of the base classifiers lead to poor weights for the base classifiers and poor object weights. This in turn leads to an unwarranted focus on some objects with heigh weight. In particular when many noisy objects are present in the training set, the performance of the AdaBoost deteriorates significantly.

The proposed algorithm ValidBoost incorporates two adaptations to the basic AdaBoost algorithm. First, it extracts a validation set from the training set for a more reliable error estimate. Instead of estimating the error on the training set only, it uses a weighted combination of the training and validation error. Second, it increases the size of the validation set during the boosting iterations, and the relative weight of the validation error to the final error estimate. These two adaptations result in both a rapid error decrease during the boosting rounds, and the avoidance of overfitting on noisy data.

When data is well-sampled and clean, i.e. there are not many outlier objects, the differences between the standard AdaBoost, LogitBoost, Bylander & Tate and the proposed ValidBoost are small. Significant differences appear when the classification problem becomes hard: in high dimensional feature spaces, with noise and more flexible base classifiers. Experiments show that this might occur when some classes in a classification problem are small, and when boosting algorithms stop their boosting updates because their base classifiers obtain very high errors. In these situations ValidBoost is able to avoid overfitting and keep adding informative base classifiers to the ensemble.

An open issue is the stronger dependence of ValidBoost to the choice of T, the number of boosting rounds. This dependence of ValidBoost on T is stronger than the other boosting algorithms because both the size of the validation set and the computation of the error of the base classifier depends on this. When the number of boosting rounds is set too low, ValidBoost does not have enough time to converge. Experiments show that for most datasets $T = 1024$ is sufficient, although for some datasets it may be set a bit higher.

References

[1] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[2] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[3] A. J. Grove and D. Schuurmans, "Boosting in the limit: Maximizing the margin of learned ensembles," in *AAAI/IAAI*, 1998, pp. 692–699.

[4] D. Mease and A. Wyner, "Evidence contrary to the statistical view of boosting," *The Journal of Machine Learning Research*, vol. 9, pp. 131–156, 2008.

[5] R. A. Servedio, "Smooth boosting and learning with malicious noise," *The Journal of Machine Learning Research*, vol. 4, pp. 633–648, 2003.

[6] P. M. Long and R. A. Servedio, "Random classification noise defeats all convex potential boosters," *Machine Learning*, vol. 78, no. 3, pp. 287–304, 2010.

[7] T. Bylander and L. Tate, "Using validation sets to avoid overfitting in adaboost." in *FLAIRS Conference*, 2006, pp. 544–549.

[8] J. Torres-Sospedra, C. Hernández-Espinosa, and M. Fernández-Redondo, "Improving adaptive boosting with k-cross-fold validation," in *Intelligent Computing*. Springer, 2006, pp. 397–402.

[9] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space." NIPS, 1999.

[10] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[11] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

Appendix A
UCI Datasets

Dataset	**Objects**	**Dimensions**	**Classes**
breast-cancer-wisconsin	699	9	2
ecoli	336	7	8
glass	214	9	4
heart-disease	297	13	2
ionosphere	351	34	2
iris	150	4	3
liver-disorders	345	6	2
pima-indians-diabetes	768	8	2
satimage	6435	36	6
sonar	208	60	2
soybean-large	266	35	15
soybean-small	136	35	4
wine	178	13	3