Discuss approaches to combine techniques used by ensemble learning methods. Randomness which is used by Bagging and Random Forests is introduced into Adaboost to get robust performance under noisy situation. Declare that when the randomness introduced into AdaBoost equals to 100, the proposed algorithm turns out to be a Random Forests with weight update technique. Approaches are discussed to improve the performance of Random Forests with weight update technique introduced.

**Keywords:** AdaBoost; Random Forests; Bagging; Weight Update

## 1 Introduction

Several authors have noted that constructing ensembles of base learners can significantly improve the performance of learning. AdaBoost [1], Bagging [6] and Random forests (RF) [5] are the most popular examples of this methodology. The success of ensemble methods is usually explained with the margin and correlation of base classifiers [14].

We firstly introduce randomness into AdaBoost to improve its robustness under noisy situation. Then we declare that when the randomness level equals to 100, the proposed algorithm turns out to be a Random Forests with stump and weight update technique used. Under this situation, we further investigate approaches to combine weight update technique into Random Forests with Random Tree as its base classifier instead of stump. Then BootStrap technique is introduced and discussed under this situation.

## 2 Random-Adaboost

We consider introduce randomness into Adaboost to get better and more robust performance.

---

*Project supported by National Science Foundation of China (No. 10875027) and the Major State Basic Research Development Program of China (Program 973) (No. 2008CB717807).

*Corresponding author.

Email address: sbx19850629@msn.com (Binxuan SUN).
AdaBoost is a well-known ensemble learning algorithm. Also it shows its resistance to overfitting in low noise data cases, a lot of experiments\[2,3,4\] have shown that it is quite sensitive on noisy data. Several previous studies have focused on exploring the performance of ensemble methods in the presence of noise. Opitz and Maclin \[2\] illustrated the overfitting problem of AdaBoost by a simple experiment. Jiang \[8\] has studied the theoretical aspects of boosting on noisy data. Oza \[9\] proposed an approach called AveBoost2 to smooth noise. Kalai and Servedio \[12\] present a new boosting algorithm and prove that it can attain arbitrary accuracy when classification noise is present. An algorithm, Smooth Boosting \[13\], is proven to tolerate a combination of classification and feature noise. The data distribution skewness is penalized in the learning process to prevent several hardest examples from spoiling decision boundaries in \[7\]. Some other works focus on moving the noisy data from the training set \[10, 11\].

Bagging \[6\] and Random\[5\] forests have shown their robustness to noise. In this part we introduce the idea of randomness into AdaBoost and propose a new algorithm, which we call Random-AdaBoost in which the sub-optimal classifier is selected instead of the best one as in the original AdaBoost. Experiments show that the proposed method is superior to AdaBoost on both low and high noise dataset and is also resist to overfit. The algorithm is proposed below.

**Algorithm 1: Random-AdaBoost**

Given Training set $(x_m, y_m)$, $m = 1, 2, \ldots, M. x_i \in X ; y_i \in \{-1, +1\}$

1. Start with distribution $D_1(i) = 1/M$ ;
2. For $t = 1, \ldots, T$ :
3. Train weak learner using distribution $D_t$ and get a hypothesis set $\{h_t\}$
4. Get a hypothesis randomly from the best $r$ percent of the set $: ht : X \rightarrow R$
5. Choose $\alpha_i \in R$
6. Update:
7. $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_i y_i h_t(x_i))}{Z_t}$
8. where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).
9. Output the final hypothesis:
10. $H(x) = sign(\sum_{t=1}^{T} \alpha_i h_t(x))$

The main difference between Random-AdaBoost and the original AdaBoost is the new introduced parameter $r$ which is used to denote randomness level. In each iteration a hypothesis is selected randomly from the best $r$ percent of the hypothesis set. Later, this parameter will be discussed jointly with noise level and the number of the iterations.

Dietterich \[3\] introduced a method Randomization to grow the tree where at each node the split is selected at random from among the K best splits. Their idea is similar to ours but they didn’t introduce the idea of randomness into AdaBoost.

Experiments are implemented on dataset “ionosphere” which is a binary problem with 351 samples and 34 input variables. The Randomness Level, noise level and the number of iterations are discussed jointly. As our goal in this paper is to explore how this simple technique works, stump is used as the weak learning algorithm. The dataset is divided into training set and test set randomly on each running of the algorithm and test errors with fixed parameters are averaged on
Fig. 1: Comparison of AdaBoost and random-AdaBoost, indicated by solid line and dashed line respectively, on different randomness level (indicated by $r$), the horizontal axis indicates noise level and the vertical axis indicates the test error. The iteration number is fixed to 500.

100 running of the algorithm. The noisy data is generated by reversing the labels of the instances randomly under the noise level.

As Randomness Level increases, the ability of Random-AdaBoost to focus on the examples incorrectly classified is getting down and when randomness level is 100, the weight update technique used in the original AdaBoost no longer works and totally random classifiers are selected instead of the hardest ones. As the strength of the weak classifier decreases we were expect that the test error of the finally combined strong classifier to increase. But to our surprise, the test error gets even lower in such cases as indicated by Figure 1 where the gap between AdaBoost and Random-AdaBoost becomes larger as $r$ increases on every noise levels and when the randomness level equals to 100, the Random-AdaBoost domains AdaBost. This could be revealed more clearly in Figure 2 where the test error generally decreases on different noise levels as the Randomness level increases to 100. Notice that when random level equals zero, it is identical to the original AdaBoost.

3 RF with Weight Update Technique

When randomness level equals to 100, the proposed algorithm could not focus on the instances that were classified incorrectly. Under this situation the algorithm turns out to be a Random Forests with stump as its base classifier and weight update technique introduced. We then further investigate the influence of weight update technique on Random Forests with Random Tree as its base classifier and discuss the adjustment that should be made under this situation.

Efforts have been made to improve the performance of RF. [15] alternates the way the trees
in RF vote. [16] investigates some possibilities to increase strength or decrease correlation of individual trees in the forest. [17] shows that a continuous spectrum of randomization exists.

[18] [19] also work on combining the techniques used in bagging and AdaBoost into RF. [20] combines AdaBoost with RF to improve the performance of the final proposed algorithm. [21] combines different ensemble methods to get a robust algorithm on most cases. But they did not compare the influence of different weight update technique or discuss the restriction that should be applied to RF to get better performance. Instead they simply combine the weight update used in AdaBoost with RF in the simplest way which does not performs well in our experiments if the trees in RF is grown to the largest size and do not prune.

3.1 Introduce weight update technique

We firstly introduce the weight update technique into RF directly and propose the combined algorithm A-RF which is the acronym of AdaBoost-Random Forests. This algorithm is described in Algorithm 2.

**Algorithm 2: A-RF procedure**

Given Training set \((x_i, y_i), i = 1, 2, \ldots, m. x_i \in X, y_i \in \{-1, +1\}\)

1. Start with distribution \(D_1(i) = 1/M\);
2. For \(t = 1, \ldots, T\):

---

Fig. 2: Test error of random-AdaBoost on randomness level from 0 to 100 with different noise levels \(n=0, 10, 30\), indicated by dotted line, dashed line and solid line respectively, iteration number is fixed to 500.
3 Train a random tree using distribution $D_t$

4 Update the weights of the instances based on the classification result of the tree currently trained

5 End

6 Combine the trained random trees with weights to form the random forests.

As our goal in this part is to investigate the potential of improving Random Forests with techniques used in AdaBoost and bagging, the simplest setting of RF is used, and the random tree used by each algorithm is grown to the max size and do not prune.

Experiments are implemented on dataset "ionosphere". To construct training set and test set, the following procedure was used: a random 10% of the data was selected as test set, the remaining data was used as training set. After the final combined classifier was trained, the test set was put down each tree to get a test set error. This was repeated 20 times and the test set errors averaged. We experimented on three most popular versions of AdaBoost: Discrete-AdaBoost (DA), Real-AdaBoost (RA) and Gentle-AdaBoost (GA).

With the weight update technique, the weights of the incorrectly classified instances will be increased, then the weak learner will focus on these "hard to learn" instances. It is our expectation that this technique could decrease the correlation of the trained random trees. Figure 1 shows the test error of four algorithms on iterations from 1 to 1000: Discrete-AdaBoost-RF (DA-RF), Real-AdaBoost-RF (RA-RF), Gentle-AdaBoost-RF (GA-RF) and the original RF. The random tree used by each algorithm is grown to the max size and do not prune.

In Figure 2, we can see that only RA-RF performs better than RF but needs much more iterations. If we compare the techniques used in AdaBoost and RF, it is easy to notice that the tree grow technique is different. In RF the tree is grown to the max size and do not prune, and it is surely not suit for the weight update technique in which the tree size should be limited. This may explain the failure of GA-RF, DA-RF and the much more iteration needed by RA-RF.

Table 1 shows the average number of the split used by four algorithms to grow a tree to max size during their own procedure. It shows that more split is need by algorithms that apply various weight update technique into RF. We say that the weight update technique distorts the random tree grow process and make the grown random tree more complex.

### 3.2 Introduce bootstrap technique

Bootstrap is a technique used by Bagging. Taking a bootstrap of the training set $X$, one can avoid or get less misleading training objects in the bootstrap training set.

As discussed above the weight update technique is too aggressive for a max size random tree setting. Also the original idea of bagging is to generate different sub training set, here we use this technique to alleviate the aggressive weight update technique to achieve better performance in a max size random tree setting, and propose Algorithm 3 which we call Bootstrapped-AdaBoost-RF (B-AD-RF).

Experiments were implemented on the settings previously used. We still test on three most popular versions of AdaBoost: Discrete-AdaBoost (DA), Real-AdaBoost (RA) and Gentle-AdaBoost (GA), and propose the bootstrapped version of Bootstrapped-DA-RF (B-DA-RF), Bootstrapped-RA-RF (B-RA-RF) and Bootstrapped GA-RF (B-GA-RF). Figure 6 shows the test error of four
Fig. 3: Test error increases as the maximum number of split increases for DA-RF, RA-RF and GA-RF algorithms on iterations from 1 to 200. For all of them the trees are grown to the maximum size and do not prone.

As shown in Figure 6, the test error of the algorithms that involve weight update and bootstrap technique perform significantly better than the original RF, and also comparative with the previous algorithms using restricted number of split.

As discussed above, the weight update technique distorts the tree growing process and increases the number of split needed for growing a random tree. But with Bootstrap technique, the scope of the sub-training set that a tree fully grown on is restricted, which means that the size of the tree is restricted as shown in Table 2 where the average number of split with bootstrap technique is fewer.

Algorithm 3: Bootstrapped-AdaBoost-RF
Given Training set \((x_i, y_i), i = 1, 2, \ldots, m, x_i \in X, y_i \in \{-1, +1\}\)
1. Start with distribution \(D_1(i) = 1/M\);
2. For \(t = 1, \cdots, T\):
3. Create a bootstrapped sub training set
4. Train a random tree on this sub set
5. Update \(D_t(i)\) based on the classification result of the tree currently trained
6. end
7. Combine the trained random trees to form the weighted random forests.
4 Conclusion

We discuss approaches to combine techniques used by ensemble learning methods. Firstly randomness is introduced into Adaboost to get robust performance under noisy data. Then we declare that when the randomness introduced into AdaBoost equals to 100, the proposed algorithm turns out to be a Random Forests with stump and weight update technique. We then discussed approaches to improve the performance of Random Forests with Random Tree as its base classifier and weight update technique introduced. New algorithms are proposed under each situation.

References


