‘Modest AdaBoost’ – Teaching AdaBoost to Generalize Better

Alexander Vezhnevets, Vladimir Vezhnevets
Moscow State University
vezhnick@mtu.ru
vvp@graphics.cs.msu.ru

Abstract

Boosting is a technique of combining a set weak classifiers to form one high-performance prediction rule. Boosting was successfully applied to solve the problems of object detection, text analysis, data mining and etc. The most and widely used boosting algorithm is AdaBoost and its later more effective variations Gentle and Real AdaBoost. In this article we propose a new boosting algorithm, which produces less generalization error compared to mentioned algorithms at the cost of somewhat higher training error.

Keywords: Boosting, AdaBoost, Real AdaBoost, Gentle AdaBoost, Generalization.

1. INTRODUCTION

The cornerstone of this paper is the technique of boosting – combining a set of “weak” classifiers in to one powerful “strong” classifier or “committee”. The first practical boosting algorithm, called AdaBoost, was proposed by Freund and Schapire [9] in 1996. Due good generalization capability, fast performance and low implementation complexity, boosting has become one of the most popular and effective classification tools in computer vision [6] and pattern recognition.

Let X denote the input data space and Y be the set of possible class labels. We consider the case of two classes $Y=\{-1, +1\}$ and assume that $X = R^n$. Our goal is to build a mapping function $F:X\to Y$ that given the feature vector $x \in X$ calculates the (correct) class label y. Also, we consider the case when a set of labeled data for training is available:

$$(x_1, y_1),..,(x_N, y_N); x_i \in X; y_i \in Y .$$

All AdaBoost based techniques can be considered as a greedy optimization method for minimizing exponential error function (see [1] for details):

$$\sum_{i=1}^{N} e^{-y_i \cdot F(x_i)}$$

where F(x) is the constructed classifier decision [2]. While good approximation capabilities of Boosting have been theoretically proven [1], its impressive generalization capabilities are still confirmed only experimentally, leaving enough space for future investigations for improvements.

In this article we propose a new boosting scheme, which is aimed to generalize better, sometimes at the cost of higher training error. Another advantage of our method is a natural stopping criterion, which other boosting technique lack. We implemented our and Gentle AdaBoost schemes in MatLab environment to compare the results. Our experiments show that our boosting method outperforms Gentle AdaBoost in terms of generalization error (measured on a control subset), but reduces training error much slower, sometimes not reaching zero. Also our method tends to overfit less. To sum up - this paper’s contribution is twofold. First, we propose a novel boosting technique, which in our experiments outperforms its predecessors; and second we hope to show that Boosting has a potential yet to be unlocked.

2. PROPOSED METHOD

In this section we will give a pseudocode and a detailed description of an algorithm of the new boosting scheme called *Modest AdaBoost*. In section 3 we show the experimental results and in section 4 we will give some comments on the proposed approach and discuss its proprieties.

Modest AdaBoost

```
1. Given training data $(x_1, y_1),..,(x_N, y_N)$, initialize data weights $D_0(i) = 1/N$.
2. For m = 1,…,M and while $f_m \neq 0$ :
   a. Train weak classifier $h_m(x)$ using distribution $D_m(i)$ by weighted least squares.
   b. Compute ‘inverted’ distribution $\overline{D}_m(i) = (1 - D_m(i))\overline{Z}_m$,
   c. Compute:
      $P_m^{+1}(x) = P_{D_m}(y = +1 \cap h_m(x))$
      $\overline{P}_m^{+1}(x) = P_{\overline{D}_m}(y = +1 \cap h_m(x))$
      $P_m^{-1}(x) = P_{D_m}(y = -1 \cap h_m(x))$
      $\overline{P}_m^{-1}(x) = P_{\overline{D}_m}(y = -1 \cap h_m(x))$
   d. Set
      $f_m(x) = (P_m^{+1}(1 - \overline{P}_m^{+1}) - P_m^{-1}(1 - \overline{P}_m^{-1}))(x)$
      and update the distribution:
      $D_{m+1}(i) = \frac{D_m(i)\exp(-y_i f_m(x_i))}{Z_m}$
3. Construct the final classifier $sign\left[\sum_{i=1}^{i=M} f_m(x)\right]$
```

As in the original AdaBoost scheme we have the set of training data: $(x_1, y_1),..,(x_N, y_N)$, where x_i is a an input vector and y_i is it's class label, $y = \{-1;+1\}$. Also we have H - a family of weak classification functions, which can either perform mapping to class labels space $h : X \rightarrow \{-1,+1\}$, or be real-valued functions, calculating "confidence-rated' predictions. In the latter case the sign of *h(x)* gives the classification, and |*h(x)*| a measure of classification "confidence". The only requirement for $h(x) \in H$ is that for any of its output values we should be able to estimate probability $P_D(y = 1 \cap h(x))$, where D is a weights distribution on the input data. The final classifier has the following form:

$$F(x) = sign\left[\sum_{m=1}^{M} f_m(h_m(x))\right]$$

where $h_m(x) \in H$ are the trained weak classifiers and f_m are real-valued functions. The process of building a boosted classifier is iterative: each iteration constructs a respective term $f_m(h_m(x))$. There are three key steps in boosting iteration: data weights recomputation, weak classifier fitting and computing $f_m(h_m(x))$.

As in the original AdaBoost the distribution over the training data is calculated in the following way:

$$D_{m+1}(i) = \frac{D_m(i)\exp(-y_i f_m(h_m(x_i)))}{Z_{m+1}},$$

where Z_m is a normalizing coefficient. Each step we compute an additional 'inverse' distribution:

$$\overline{D}_m(i) = (1 - D_m(i)) \cdot \overline{Z}_m ;$$

Z_m and $\overline{Z}_m$ are chosen so that:

$$\sum_{i=1}^{N} \overline{D}_m(i) = \sum_{i=1}^{N} D_m(i) = 1 \cdot$$

The initial weights are set to $D_0(i) = 1/N$. Weak classifier fitting at step 2.a is done by the weighted least squares:

$$h_m = \arg\min_h \left(\sum_{i=1}^{N} D_m(i) \cdot (y_i - h(x_i))^2 \right)$$

The steps 2.a-2.d a repeated M (pre-defined value) times, or while $f_m \neq 0$ - this situation can occur if both $(1 - \overline{P}_m^{\,+1}) = 0$ and $(1 - \overline{P}_m^{\,-1}) = 0$.

3. EXPERIMENTAL RESULTS

In this section we will present experimental results on UCI Machine Learning Repository database. The aim of our experiments was to compare Gentle AdaBoost, which is assumed to bee one of the best boosting algorithms used in practice with Modest AdaBoost. We used stumps as weak classifier for both methods. This choice was made because stumps are considered to be the "weakest of all" among commonly used weak learners, so we hope that using stumps lets us investigate the difference in performance resulting from different boosting schemes.

For now, we have tested four UCI datasets: ionosphere, Wisconsin breast cancer, Diabetes and Hepatitis. We used 5-fold cross validation to obtain the results presented below.

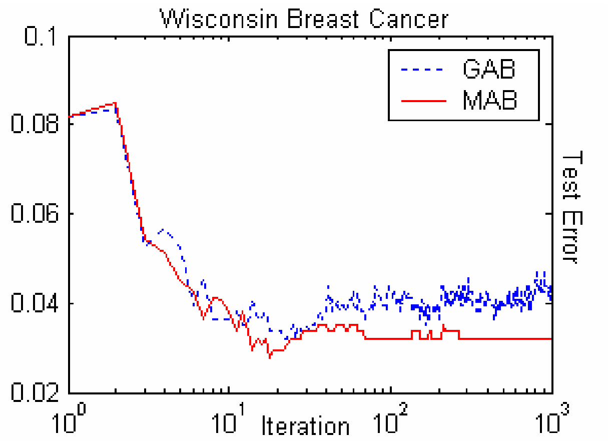

Figure 1. Modest AdaBoost (MAB) and Gentle AdaBoost (GAB) performance on test subset of UCI Wisconsin breast cancer dataset. Iteration axis is given in logarithmic scale.

As you can see on the figures 1-3, Modest AdaBoost performs noticeably better then Gentle AdaBoost on our test subset. Difference is about 1-4 percent, but in such applications like object detection (see [4] and [6]) it can be significant. In fact, our experiments show that our boosting method is resistant to overfitting more then Gentle AdaBoost (see Figures 2 - 3).

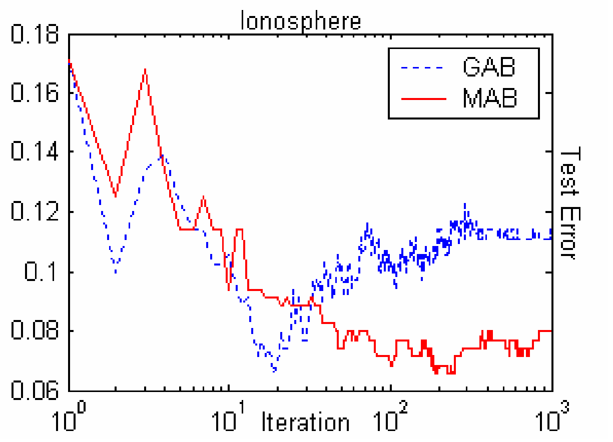

Figure 2 Modest AdaBoost (MAB) and Gentle AdaBoost (GAB) performance on test subset of Ionosphere. Iteration axis is given in logarithmic scale.

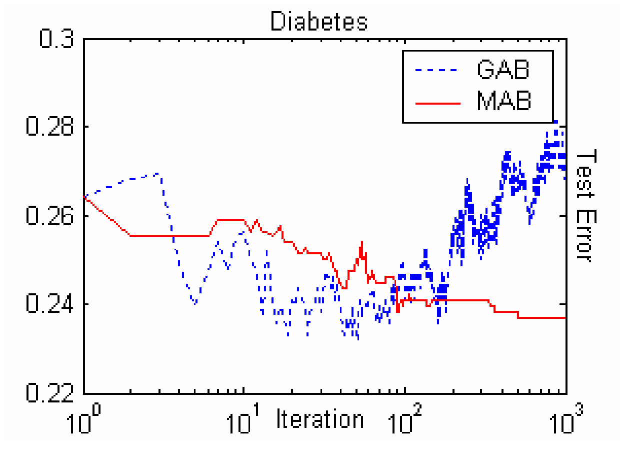

Figure 3. Modest AdaBoost (MAB) and Gentle AdaBoost (GAB) performance on test subset of UCI Diabetes dataset. Iteration axis is given in logarithmic scale.

Performance on Diabetes dataset (figure 3) is the most interesting. As you can see, Gentle AdaBoost outperforms Modest AdaBoost in terms of minimum test error during first 90 iterations, but after that Gentle AdaBoost starts to overfit, while Modest AdaBoost further decreases the test error.

The currently known drawback of the proposed method that is training error that decreases much slower then in Gentle AdaBoost scheme and often does not reach zero point, as displayed on figure 4.

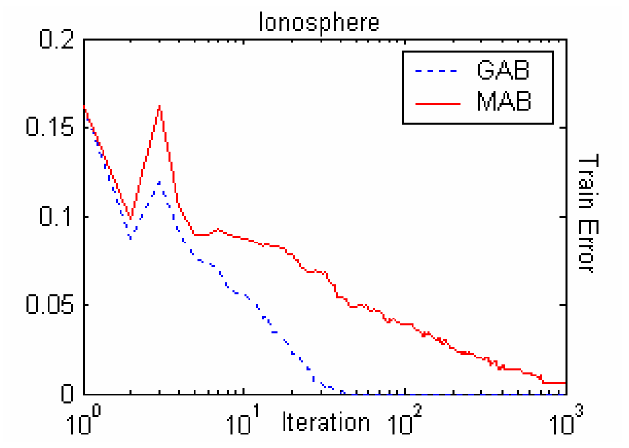

Figure 4. Modest AdaBoost (MAB) and Gentle AdaBoost (GAB) performance on training subset of Ionosphere dataset. Iteration axis is given in logarithmic scale.

4. DISCUSSION AND CONCLUSION

4.1 Comments on the algorithm structure

In this section we will give some insight on how and why our method works. The distribution at each step can also re-written as:

$$D_m(i) = \frac{\exp(-\mu(x_i))}{Z},$$

$$\mu(x) = y\sum_{k=0}^{m} f_m(h_m(x))$$

The value of $\mu(x)$ is proportional to margin (for more on margins see [9]), and it represents a kind of confidence of currently build classifier in a sample. This means that at each step distribution D_m assigns high weights to training samples misclassified by earlier stages, so the inverse distribution $\overline{D}_m$, on the contrary, gives higher weights to samples, which are already correctly classified by earlier steps. The values of

$$P_m^{+1} = P_{D_m}\left(y = +1 \cap h_m(x)\right)$$
$$P_m^{-1} = P_{D_m}\left(y = -1 \cap h_m(x)\right)$$

are the measurements of how good our current weak classifier is at predicting the class labels. If we chose $f_m = P_m^{+} - P_m^{-}$ as a term for current step, we will get a boosting scheme very similar, but not identical, to Gentle AdaBoost [1].

The values of

$$\overline{P}_m^{+1} = P_{\overline{D}_m}\left(y = +1 \cap h(x)\right)$$
$$\overline{P}_m^{-1} = P_{\overline{D}_m}\left(y = -1 \cap h(x)\right)$$

are the estimates of how good our current classifier $h_m(x)$ is working on the data, which has been correctly classified by previous steps. So when we use $f_m = P_m^{+1}(1-\overline{P}_m^{+1}) - P_m^{-1}(1-\overline{P}_m^{-1})$ as an update for current step, we decrease weak classifiers contribution, if it works "too good" on data that has been already correctly classified with high margin. That is why the method called *Modest* AdaBoost - it forces the classifiers to be 'modest' and work only in their domain, defined by D_m. A noticeable feature of such an update is that

$$1-\overline{P}_m^{\,y}, y \in \{-1,+1\}$$

can actually become zero, so then update will not occur. This feature provides us with a natural stopping criterion.

It is important to notice why $P(y \cap x)$, rather then conditional probability $P(y \mid x)$ is used. This is justified by the fact that for any events x and y: $P(y \mid x) = P(y \cap x) / P(x)$, so comparing $P(y_1 \mid x) \vee P(y_2 \mid x)$ is equivalent to comparing $P(y_1 \cap x) \vee P(y_2 \cap x)$, which implicitly happens at each boosting step.

4.2 Discussion on generalization capabilities

Now we will try to provide our assumptions on the reasons of for outperforming than Real AdaBoost on the experimental data provided in section 3. We cannot yet provide a strict theoretical argumentation, so we will give some intuitive explanation that in future may lead us to a strict proof.

Every step we compute new distribution over the training set hence highlighting those samples, which were misclassified on previous steps (having low margin). So at each step boosting concentrates on increasing the lowest margins of training samples. Therefore, samples with already high margins can be misclassified by a current step, hence their margins would be decreased. If they get too low, they will be corrected by posterior steps. But *increasing* a margin of a sample, which is already high, has to be considered also. We suppose that it is not good for generalization capabilities of an algorithm to increase samples margin, when it is already high, and that is why we give an additional multipliers $\overline{P}_m^{\,y}, y \in \{-1,+1\}$. Those multipliers decrease weak learners' contribution proportionally to its probability to increase high margins. Why increasing high margins can be harmful for generalization? To put in few words – 'being unsure is not good, but being overconfident is not good either'. If an algorithm produces very high margins for a subset of training data, it becomes overconfident in some region of input space, which is represented in training set by those samples.

Another explanation can be given in terms of variance reduction. It is known, that for purpose of building a committee, we must use classifier that have maximally low variance [7]. Modest AdaBoost scheme favors weak classifiers that have maximally decorrelated error with each other. Another explanation may lay in Bayesian interpretation of boosting given in [9]. As Freund and Schapire mentions boosting is identical to a Bayesian classifier in case of independent weak learners (hypothesis). If we remove this constrain, then we should add another term, which takes account of learners correlation. Our guess is that $P_m^{\,y} \cdot \overline{P}_m^{\,y}, y \in \{-1,+1\}$ can be a good approximation of such term and hope to investigate this assumption in future work.

4.3 Conclusion

We have proposed a new boosting scheme, called Modest AdaBoost, which outperforms Gentle AdaBoost in terms of generalization error and overfitting experimentally measured by 5 fold cross validation on real world classification problems from UCI Machine Learning Repository database. Our method has the same performance speed as any of other AdaBoost based method and same implementation complexity. We hope that this algorithm will provide possibilities for improving performance of practically used boosting-based classification systems and also help theoretical investigations in this field.

5. REFERENCES

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. **Additive logistic regression: A statistical view of boosting.** *The Annals of Statistics*, 38(2):337–374, April 2000.

[2] Y Freund and R. E. Schapire. **Game theory, on-line prediction and boosting.** In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

[3] R. E. Schapire. **The Boosting Approach to Machine Learning. An Overview.** *MSRI Workshop on Nonlinear Estimation and Classification, 2002.*

[4] R. Lienhart, A. Kuranov, V. Pisarevsky. **Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection**. *MRL Technical Report, 2002*

[5] Pavlov, D. Mao, J. Dom, B. **Scaling-up Support Vector Machines Using Boosting Algorithm.** *Proceedings. 15th International Conference on Pattern Recognition, 2000.*

[6] P. Viola and M. Jones. **Robust Real-time Object Detection.** *In Proc. 2nd Int'l Workshop on Statistical and Computational Theories of Vision -- Modeling, Learning, Computing and Sampling, Vancouver, Canada, July 2001.*

[7] T. G. Dietterich. **Machine Learning Research: Four Current Directions** *AI Magazine. 18 (4), 97-136, 1997.*

[8] Rosset, Zhu and Hastie. **Boosting as a Regularized Path to a Maximum Margin Classifier.** *Journal of Machine Learning Research 5 (2004)* 941–973, 2004.

[9] 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*, 55(1):119–139,August 1997.

About authors

Alexander Vezhnevets is a fourth year student of Moscow State Lomonosov University, Faculty of Computational Mathematics and Cybernetics. His research interests lie in the fields of machine learning, pattern recognition, statistics and image processing. His email address is vezhnick@mtu.ru.

Vladimir Veznevets is a research fellow at Graphics and Media Laboratory of Moscow State Lomonosov University. He graduated with honors Faculty of Computational Mathematics and Cybernetics of Moscow State University in 1999. He received his PhD in 2002 in Moscow State University also. His research interests include image processing, pattern recognition, computer vision and adjacent fields. His email address is vvp@graphics.cs.msu.ru.