An Empirical Comparison of Adaboost Using Three Different Weak Learners

Kai Wu, 78448140
Electrical and Computer Engineering Department
University of British Columbia
2332 Main Mall, Vancouver, BC Canada V6T 1Z4
imkaywu@gmail.com

Abstract

This project report investigates the performance of Adaboost using 3 different weak classifiers - decision stump, decision tree, and weakened SVM. The empirical comparison of convergence rate and training and test error rate are used to evaluate the algorithms.

1 Introduction

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining several relatively weak classifiers. The Adaboost algorithm proposed by Freund and Schapire [7] was the first practical boosting algorithm, and remains one of the most widely used and studied. The problem is, 1) most of the previous work has focused on comparing the performance between boosting and other classifiers, few work investigates the performance of boosting using different weak learners, thus 2) we don't know the optimal base learner in a particular or general case.

This is important because though boosting can generally yield a better result than the base learners, this might not always be the case. For instance, boosting a strong learner is generally counterproductive and results in worse results [9]. Therefore, it's crucial to identify what kind of base learners can be benefited from boosting.

The contribution of this project is the implementation of the weakened SVM and an empirical comparison of Adaboost algorithms using 3 different weak learners.

2 Background

Most of the literature investigates: **1)**. Compare the performance of the so called "voting classification algorithms", including bagging and boosting [1] or compare the performance of boosting to that of the other classifiers. [1] describe an empirical study comparing Bagging and Adaboost and explain why and when these algorithms, which use perturbation, reweighting and combination techniques, affect classification error, and [2] compares the effectiveness of randomization, bagging and boosting for improving the performance of the decision tree algorithm C4.5. Though it's not a empirical comparison between boosting using different base learners, I borrowed some of the test measurements in those papers to this project. **2)**. Explain or understand the classification power of boosting algorithm from numerous perspective [3]. [3] investigates various explanations to try to capture different aspects of Adaboost's behavior and form a rich and expansive theory for understanding this algorithm, including direct application of VC theory, margins explanation proposed by Schapire et al.[4], exponential loss function minimization, regularization, etc. [4] gives a general introduction of boosting, including the underlying theory, the mystery of avoiding overfitting and its

relationship with SVM. [10] explains the magic of boosting from the perspective of statistical principles, namely additive modeling and maximum likelihood. This helps to understand why boosting works, but doesn't clarify what kind of weak classifiers can be boosted or benefited from boosting.

3 Implementation

This section gives a background review of the Adaboost algorithm and the implementations of the weak classifiers.

3.1 Adaboost

Algorithm 1 Adaboost

Initialize weights to be equal $w_i = \frac{1}{N}$
for $t = 1...nBoosts$ **do**
  fit classifier h_t to the weighted data.
  compute the weighted error $\epsilon_t = w_i^t(y_i \neq h_t(x_i))$
  voting weight for base learner $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
  recalculate weights: $w_{t+1}(i) = \frac{w_t(i)exp(-\alpha_t y_i h_t(x_i))}{\sum_{i=1}^{N} w_t(i)}$.
end for
Majority vote classification: $sign\{\sum_{i=1}^{nBoosts} \alpha_i h_i(x)\}$

There are generally two ways to boosting a weak classifiers.

- Take into account the weights in the training stage.
- Simulate the weights by sampling. In each round, construct a new data set based on the old one, sample from the old data set k times with replacement. In each sampling, each data point x_i in the old data set has probability z_i of being chosen into the new dataset. k can be as large as the size of the old data set, or smaller. We only need to make sure there are enough data points sampled for a weak classifier to be trained reliably. Then we train a weak classifier without considering weights because weights are already considered in sampling. In this way, we don't need to modify the weak classifier. When the weak classifier is trained, the new data set is discarded. The only use of the newly constructed data set is in training the weak classifiers. Any other computation is based on the original data set.

3.2 Decision Tree

There are many measures that can be used to determine the best way to split the data set. The measure developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. Examples of impurity measures include

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2$$

$$Classification\_error(t) = 1 - \max_i p(i|t)$$

When training the boosted decision tree, we need to take into consideration the weights of the training set. I used bootstrap sampling in the current implementation.

Decision trees need to be pruned in order to prevent overfitting. There are generally two strategies:

- pre-prune: Tree growing is halted before generating a fully grown tree. Stop criteria: observed gain in impurity measure below a certain threshold.

- post-prune: Trim a fully grown tree by replacing a subtree with (1) a new leaf node whose label is determined by the majority class of records, or (2) the most frequently used branch of the subtree. (1) is used in the current implementation.

3.3 Decision Stump

Decision stump is an one-layer decision tree.

3.4 SVM

A simple way of boosting SVM is to use bootstrap samples of the original set drawn according to the discrete distribution. A serious drawback of this method is that the matrix Q may become ill conditioned because examples with large weight in the distribution may appear several times in a bootstrap sample.

A better option is to modify the original optimization problem to incorporate directly the distribution. I present two alternative solutions [12].

First alternative: Penalize the slack variable ξ_i proportional to z_i in the objective function.

The primal problem:

$$\begin{aligned} \min_{w} \quad & \mathcal{P}(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \boldsymbol{z_i}\xi_i \\ s.t. \quad & y_i(\langle w, \Phi(x_i)\rangle + b) \geq 1 - \xi_i, i = 1, ..., N \\ & \xi_i \geq 0, i = 1, ..., N \end{aligned}$$

where ξ_i is the slack variable and $C > 0$ is the regularization parameter. A positive slack variable $\xi_i > 0$ indicates a classification mistake.

The Lagrangian:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}w^T w + C\sum_{i=1}^{N} \boldsymbol{z_i}\xi_i - \sum_{i=1}^{N} \alpha_i[y_i(w^T\Phi(x_i) + b) + \xi_i - 1] - \sum_{i=1}^{N} \beta_i\xi_i.$$

Taking derivative w.r.t w and b

$$\begin{aligned} w &= \sum_{i=1}^{N} \alpha_i y_i \Phi(x_i) \\ 0 &= \sum_{i=1}^{N} \alpha_i y_i \end{aligned}$$

The dual problem:

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{D}(\alpha) = \vec{e}^T\alpha - \frac{1}{2}\alpha^T Q\alpha \\ s.t. \quad & 0 \leq \alpha_i \leq C\boldsymbol{z_i}, i = 1, ..., N \\ & y^T\alpha = 0 \end{aligned}$$

where $Q_{ij} = y_i y_j k(x_i, x_j)$, $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j)\rangle$ is a positive definite kernel, $\langle ., .\rangle$ is the dot product and $\vec{e}$ is a vector of ones.

Second alternative: Include the distribution in the objective function but also force examples with larger weights to have large margin.

The primal problem:

$$\begin{aligned} \min_{w} \quad & \mathcal{P}(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \boldsymbol{z_i}\xi_i \\ s.t. \quad & y_i(\langle w, \Phi(x_i)\rangle + b) \geq \boldsymbol{z_i}(1-\xi_i), i = 1, ..., N \\ & \xi_i \geq 0, i = 1, ..., N \end{aligned}$$

The Lagrangian:

$$\mathcal{L}(w,b,\xi,\alpha,\beta) = \frac{1}{2}w^T w + C\sum_{i=1}^{N} \boldsymbol{z_i}\xi_i - \sum_{i=1}^{N} \alpha_i[y_i(w^T\Phi(x_i)+b) + \boldsymbol{z_i}(\xi_i - 1)] - \sum_{i=1}^{N} \beta_i\xi_i.$$

The dual problem:

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{D}(\alpha) = \boldsymbol{z_i}^T\alpha - \frac{1}{2}\alpha^T Q\alpha \\ s.t. \quad & 0 \leq \alpha_i \leq C, i = 1, ..., N \\ & y^T\alpha = 0 \end{aligned}$$

The solution of this problem has the form:

$$h_{w,b}(x) = sign(\sum_{i=1}^{N} y_i\alpha_i k(x, x_i) + b)$$

Data points for which $\alpha_i \neq 0$, are called support vectors.

3.4.1 Weakened SVM

As mentioned in [2], Adaboost with strong base learners doesn't yield good results. These base classifiers must be appropriately weakened in order to benefit from Boosting. For RBFSVM(SVM with RBF kernel), the performance is affected by the regularization term C, and Gaussian width σ. As reported in [5], the performance of RBFSVM largely depends on the σ value if a roughly suitable C is given: a larger σ often leads to a lower complexity and lower performance classifier while a smaller σ often increases the learning complexity and leads to higher performance. As is shown in Fig. 1, which plots the test error of SVM against the values of C and σ on a non-separable data set used in [6]. We can see that changing σ leads to larger variation on test error than changing C. So with a roughly value of C, by choosing a relatively high value of σ, we can weaken the RBFSVM. Because the base classifier should perform better than random guessing, the RBFSVM with a certain σ is trained as many cycles as possible as long as more than half accuracy is met. Otherwise, σ is reduced slightly to increase the performance of RBFSVM. See Algo. 2 for details.

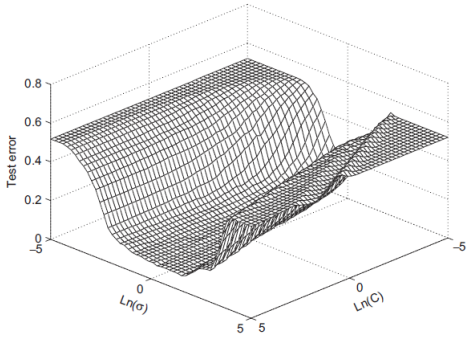

Figure 1: The test error of SVM vs. σ and C values.

Algorithm 2 Adaboost using RBFSVM as weak learner

Initialize weights to be equal $w_i = \frac{1}{N}$
for $t = 1...nBoosts$ **do**
  repeat
    Train RBFSVM h_t using weights z_i and σ.
    $\epsilon_t = \sum_{i=1}^{N} z_i^t (y_i \neq h_t(x_i))$.
    if $\epsilon_t \geq 0.5$ **then**
      decrease σ.
    end if
  until $\epsilon_t < 0.5$
  voting weight for base learner $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$.
  recalculate weights: $w_{t+1}(i) = \frac{w_t(i) exp(-\alpha_t y_i h_t(x_i))}{\sum_{i=1}^{N} w_t(i)}$.
end for
Majority vote classification: $sign\{\sum_{i=1}^{nBoosts} \alpha_i h_i(x)\}$

4 Experiments

I conducted a series of tests using 5 data sets summarized in Table 1. All of these data sets, with the exception of *binary* were taken from the *UCI Machine Learning Repository* [8]. The first data set is used in the assignment 6.

Table 1: Data Sets

Data Set	Entries	Attributes
Binary	250	2
statlog	270	13
sonar	208	60
liver	345	6
ionosphere	351	34

The *Statlog(Heart)* data set is a heart disease data set with both categorical and real-value attributes.

The *Sonar* data set contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions, and 97 patterns obtained from rocks under similar conditions. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

Each data example of *Liver* data set constitutes the record of a single male individual. The first 5 attributes are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption.

Ionosphere data set is the radar data collected by a system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not.

Experiment 1: We record the weighted error for each iteration of the training process and the final training error. The can show the convergence speed of the training process.

As is shown in Figure. 2, the weighted error of decision tree in each iteration is relatively low and after less than 10 iterations, the training error decreases to 0. Whereas in the case of decision stump, the weighted errors are relatively high and the training error converges to 0 slower, and for more complicated data set(feature dimension larger than 2), the training error doesn't even converges to 0. In the case of SVM, the weighted errors fluctuated wildly and the training error converges to 0 slower than that of decision tree, but faster than decision stump. We conclude that decision tree is the best in terms of convergence rate, and SVM is second for complicated data set.

Experiment 2: Each algorithm consisted of 50 trials. At each trial, one third of the data examples were selected at random and set as the test set. The remaining two thirds of examples were used to train the algorithm. I record both the training error and test error to compare the classification power of each algorithm. I only tested the data set *statlog*, results are in Figure. 3.

We concluded from this experiment that Adaboost with decision tree and decision stump yield similar test error rate while Adaboost with RBFSVM has the worst results.

5 Discussion and Future Work

Adaboost with decision tree as base learner has the fastest convergence rate as well as the best test error rate. Although the convergence rate of RBFSVM is faster than that of decision stump, the training time and test error rate of RBFSVM are worse than that of decision stump. So the conclusion is that Adaboost with decision tree is the best in terms of convergence rate and test error rate, decision stump comes next and SVM is the worst of all three.

Strength:

1. The experiment results are consistent among different data sets.

Weakness

1. The types of base learners are limited;
2. The results are based on the experimental results, with no theoretical justifications.

We should implement more base learners to decide the best base learner in various situations and apply statistical analysis to explain the classification power of Adaboost with different base learners.

References

[1] Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants[J]. Machine learning, 1999, 36(1-2): 105-139.

[2] Dietterich T G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization[J]. Machine learning, 2000, 40(2): 139-157.

[3] Schapire R E. Explaining AdaBoost[M]//Empirical Inference. Springer Berlin Heidelberg, 2013: 37-52.

[4] Schapire, R.E., Freund, Y., Bartlett, P., Lee,W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of Statistics 26(5), 16511686 (1998).

[5] Valentini, G., Dietterich, T.G., 2004. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research 5, 725775.

[6] Baudat, G., Anouar, F., 2000. Generalized discriminant analysis using a kernel approach. Neural Computation 12, 23852404.

[7] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119139, August 1997

[8] UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[9] Wickramaratna J, Holden S, Buxton B. Performance degradation in boosting[M]//Multiple Classifier Systems. Springer Berlin Heidelberg, 2001: 11-21.

[10] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)[J]. The annals of statistics, 2000, 28(2): 337-407.

[11] McDonald R A, Hand D J, Eckley I A. An empirical comparison of three boosting algorithms on real data sets with artificial class noise[M]//Multiple Classifier Systems. Springer Berlin Heidelberg, 2003: 35-44.

[12] Garcia E, Lozano F. Boosting Support Vector Machines[C]//MLDM Posters. 2007: 153-167.

Figure 2: Weighted error and training error using different weak classifiers as weak learners, the 1st column uses decision tree as base learners, the 2nd column uses decision stumps as base learners and the 3rd column uses the weakened RFBSVM as weak learners. The data sets are: Binary(1st row), statlog(2nd row), sonar(3th row), liver(4th row), ionosphere(5th row).

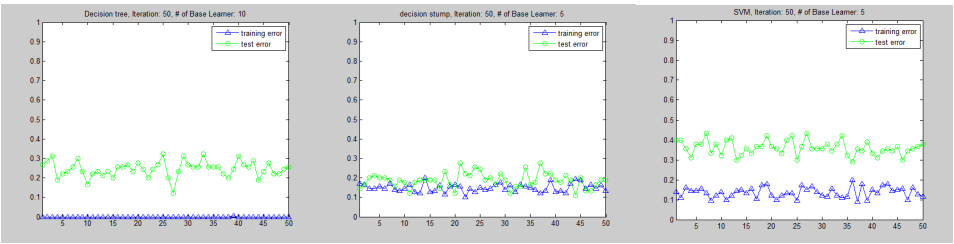

Figure 3: Training and test errors of Adaboost using different base learners tested on data set 'statlog'.