Linear Combination as an AdaBoost Weak Learner

Patrick Grennan and Jiao Li
grennan@nyu.edu
jl3056@nyu.edu

12 December 2011

1 Introduction

Given a training set $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$, the meta-algorithm AdaBoost outputs a final classifier $H(x) \in \{-1, +1\}$ for any testing point x. The algorithm, given below, does this by defing the final classifier $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$, with T representing the number of weak learners h_t chosen by the algorithm, with $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$, and with $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$. At each step h_t, the algorithm updates the distribution D_t according to the points that are misclassified by the weak learner h_t.

```
AdaBoost($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      $D_1(i) \leftarrow \frac{1}{m}$
 3  for $t \leftarrow 1$ to $T$ do
 4      $h_t \leftarrow$ base classifier in $H$ with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$
 5      $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
 6      $Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$    $\triangleright$ normalization factor
 7      for $i \leftarrow 1$ to $m$ do
 8          $D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
 9  $f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
10  return $h = \text{sgn}(f)$
```

The AdaBoost algorithm doesn't specify what weak learners should be used; in fact, the weak learner can be everything from a simple decision tree stub to other learning algorithms like the SVM and Perceptron algorithms. So, inspired by the Perceptron algorithm, we sought to find an effective weak learner heuristic based on the linear combination of the feature vectors of the training set. This project is an empirical study of the question: what is the performance of the AdaBoost algorithm using a linear combination-based weak learner?

2 The Weak Learner

Given $\mathbf{x}_i, \mathbf{x}_j \in X$, with the labels $y_i = +1$ and $y_j = -1$, h_t is defined as:

$$h_t(s_c) = \min_{\mathbf{x}_i, \mathbf{x_j}}(\Pr_{D_t}[(\mathbf{x}_i + \mathbf{x}_j) \cdot s_c \neq y_c])$$

In line 4 of the AdaBoost algorithm, the base classifier h_t is defined as a classifier in H that has a small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$. Regarding this, the heuristic that we chose was to pick the weight vector $\mathbf{w}$ (which is returned as h_t) as the best performing linear combination of one feature vector $\mathbf{x}_i \in X$ with the label $y_i = +1$ and one feature vector $\mathbf{x}_j \in X$ with the label $y_j = -1$ with respect to the distribution D_t. To settle cases where multiple combinations of $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same error rate, $\mathbf{w}$ was chosen as the linear combination with the largest margin on the training data. The intuition behind the heuristic is that choosing the best linear combination of one point labeled $+1$ and one point labeled -1 would help to avoid the problem of overfitting only one class. Likewise, choosing the solution with the greatest margin also serves to help avoid overfitting.

3 Empirical Results

To test the performance of the algorithm, we implemented AdaBoost and our weak learner in python, and tested then on a number of benchmark data sets.

3.1 The Data Sets

To test the performance of the algorithm, we used 4 data sets that can be found in the UCI Machine Learning Repository [1]:

1. Balance Scale Data Set: This data set was generated to model psychological experimental results. There are 625 instances, with each instance classified as having the balance scale tip to the right, tip to the left, or be balanced. There are 4 attributes: the left weight, the left distance, the right weight, and the right distance.

2. Iris Data Set: This is the famous data set created by R.A. Fisher; the data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant, with 4 attributes.

3. Connectionist Bench (Sonar, Mines vs. Rocks) Data Set: This data was collected by bouncing sonar signals off a metal cylinder (the first class) and rocks (the second class) at various angles and under various conditions. The feature vectors are 208 sets of 60 numbers in the range 0.0 to 1.0.

Each number represents the energy within a particular frequency band, integrated over a certain period of time.

4. Wine Data Set: This data is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. There are 178 instances, and 13 attributes for each feature vector.

3.2 Results

The table below charts the 5-fold cross-validated classification accuracy of the algorithm on the 4 data sets for the different values of T. For reference, the performance of the Perceptron algorithm (after 500 passes if the data was not linearly seperable) is provided below. The Iris Data set does not have values for later rounds of boosting, because on the first round of boosting, the classifier h_t that was found in the first round of boosting returned a classification accuracy of 100%.

T	Balance	Iris	Sonar	Wine
10	79.5%	100.0%	52.0%	66.6%
25	93.2%	-	62.3%	67.2%
50	93.3%	-	73.0%	65.8%

	Balance	Iris	Sonar	Wine
Perceptron	90.4%	100%	73.1%	66.6%

3.3 Analysis

As seen in the results, the classification accuracy varied according to the data set, but the performance of the Boosting was either at or above the result returned by the Perceptron. What's interesting to note is that for each of the data sets the optimal T was reached quite quickly, and then leveled off; further rounds of boosting did not yield better results. While this could be a result of reported susceptibility to overfitting of the AdaBoost algorithm [2], empirical observation shows that on the smaller training data sets of Sonar and Wine, this could also have been a result of the function that found h_t continuously returning the same h_ts, meaning that furter rounds of boosting would not yield better h_ts.

Another important observation about the data sets (that was also brought up by Freund and Schapire [3]) is that due to the fact that the data sets are multi-class, a one vs. all random guess of y would not have a 50% probability of being correct. This is a key assumption when computing $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$; because the final classifier $H(x)$ is represented by $\text{sign}(\sum_{t=1}^{T}\alpha_t h_t(x))$, the hypotheses that have an ϵ close to .5 are unfairly penalized even though they produce better-than-random results.

One potential downside to our algorithm is that the running time of the heuristic is $O(n^2)$, because every positive point must be checked with every negative point to find the best hypothesis h_t. Our heuristic produced significantly better results (an improved classification accuracy of 25%) than linear time linear combination heuristics; however, for large data sets and larger rounds of T, computation time can quickly become an issue.

4 Conclusion

The results that we produced by our proposed weak learner heuristic were similar, if not better, in accuracy to results yielded from other "simple" weak learners used in the AdaBoost algorithm [3], and produced them in fewer rounds of boosting. Compared to the Perceptron algorithm, by using AdaBoost, our algorithm has the advantage of not requiring the data to be linearly seperable. In conclusion, linear combination heuristics are indeed effective and well-performing weak learners for the Ada-Boost algorithm.

References

[1] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Wenxin Jiang, *"Process consistency for adaboost:"* Annals oJ Statistics, vol. 32, no. 1, pp. 13-29, 2004.

[3] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. *In Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.