Boosting & AdaBoost

Luis De Alba
Idealbar@cc.hut.fi
5.3.2008
Definition:
Method of producing a very accurate prediction rule by combining inaccurate rules of thumb.

Objective:
Improve the accuracy of any given learning algorithm (LMS, SVM, Perceptron, etc.)
How to boost in 2 steps:

- Select a Classifier with an accuracy greater than chance (random)
- Form an ensemble of the selected Classifiers whose joint decision rule has high accuracy.
Example 1: Golf Tournament
- Tiger Woods VS
- 18 average+ players
Example 2: Classification Problem

- 2 dimensions, 2 classes, 3 Classifiers
- \( D \) training sets \( \{x_1, y_1 \ldots x_m, y_m\} \)

a) Train \( C_1 \) with \( D_1 \) where \( D_1 \) is randomly selected.

b) Train \( C_2 \) with \( D_2 \) where only 50% of the selected set is correctly classified by \( C_1 \)
c) Train $C_3$ with $D_3$ where the classification, of selected $D_3$, disagrees between $C_1$ and $C_2$.

Classification task:
Duda, et al.
AdaBoost

- Adaptive Boosting aka “AdaBoost”
- Introduced by Yoav Freund and Robert Schapire in 1995. Both from AT&T Labs.

- Advantages:
  - As many weak learners as needed (wished)
  - “Focuses in” on the informative or “difficult” patterns
Algorithm Inputs:

- Training set: \((x_1, y_1), \ldots, (x_m, y_m)\)
  - \(Y=\{-1, +1\}\) (two classes)
- Weak or Base algorithm \(C\)
  - LMS, Perceptron, etc.
- Number or rounds (calls) to the weak algorithm \(k=1, \ldots, K\)
Each training tuple receives a weight that determines its probability of being selected.

At initialization all weights across training set are uniform:

\[ W_1(i) = \frac{1}{m} \]
For each iteration $k$ a random set is selected from the training set according to weights.

Classifier $C_k$ is trained with the selection.
- **Increase** weights of training patterns **misclassified** by $C_k$

- **Decrease** weights of training patterns **correctly** classified by $C_k$

- Repeat for all $K$
Algorithm:

initialize

\[ D = \{ x_1, y_1, \ldots, x_m, y_m \}, K, W_1(i) = 1/m, i = 1, \ldots, m \]

loop

For \( k = 1 \) to \( K \)

Train weak learner \( C_k \) using \( D_k \) sampled according to \( W_k(i) \)

\( E_k \leftarrow \) training error of \( C_k \) measured using \( D_k \)

\[ \alpha_k \leftarrow \frac{1}{2} \ln \left[ \frac{(1-E_k)/E_k}{1} \right] \]

\[ W_{k+1}(i) \leftarrow \frac{W_k(i) e^{(-\alpha_k y_i C_k(x_i))}}{Z_k} \]

return

\( C_k \) and \( \alpha_k \) for all \( K \)
● Output of the final classifier for any given input.

\[ H(x) = \text{sign} \left( \sum_{k=1}^{K} \alpha_k C_k(x) \right) \]
Analysis

- Error calculation:

$$E_k = \frac{\sum_{i=1}^{m} \text{if } (C_k(x_i) \neq y_i): 1 : 0}{m}$$

- Error is the summation and normalization of all wrongly classified sets by the weak learner.

- Weak learner shall do better than if only random guesses i.e., $E_k < 0.5$
- $\alpha$ measurement:
  
  $$\alpha_k \leftarrow \frac{1}{2} \ln \left[ \frac{(1-E_k)}{E_k} \right]$$

- It measures the importance assigned to $C_k$

- There are two things to note:
  - $\alpha \geq 0$ if $E \leq \frac{1}{2}$
  - $\alpha$ gets larger as $E$ gets smaller
Normalization factor:

- $Z_k$ is a normalization factor so that $W_{k+1}$ will be a distribution.
- Possible calculation:

$$Z_k = \sum_{i=1}^{m} W_{k+1}(i)$$
Hypothesis:

\[ H(x) = \text{sign} \left( \sum_{k=1}^{K} \alpha_{k} C_{k}(x) \right) \]

- \( H \) is a **weighted** majority vote of the \( K \) weak hypothesis where \( \alpha_{t} \) is the **weight** associated to \( C_{k} \).
- Each instance of \( x \) outputs a prediction whose **sign** represents the class \((-1, +1)\) and the magnitude \( |C_{k}(x)| \) the **confidence**.
Why better than random?

Suppose three classifiers:

- **Worse than random** \((E=0.75)\)
  - Alpha=-0.549  CC=1.731  IC=0.577

- **Random** \((E=0.50)\)
  - Alpha=0.0  CC=1.0  IC=1.0

- **Better than random** \((E=0.25)\)
  - Alpha=0.549  CC=0.577  IC=1.731

CC=Correctly Classified Weight Factor
IC=Incorrectly Classified Weight Factor
Generalization Error

- Based on sample size $m$, the VC-dimension $d$ and the boosting rounds $K$.

\[
\tilde{O}\left(\sqrt{\frac{Kd}{m}}\right) \to \tilde{O}\left(\sqrt{\frac{d}{m \theta^2}}\right)
\]

- Overfit when $K$ too large.
- Experiments showed that AdaBoost do not overfit.
- It continues improving after training error is zero.
Duda, et al.
In the multi-label case, each instance $\mathbf{x} \in X$ may belong to multiple labels in $Y$. 

$$(\mathbf{x}, Y) \text{ where } Y \subseteq Y$$
Decompose the problem into \( n \) orthogonal binary classification problems.

For \( Y \subseteq \mathcal{Y} \), define \( Y[\lambda] \) for \( \lambda \in \mathcal{Y} \) to be

\[
Y[\lambda] = \begin{cases} 
+1 & \text{if } \lambda \in Y \\
-1 & \text{if } \lambda \notin Y 
\end{cases}
\]

\[H : X \ast Y \to \{-1, +1\} \text{ defined by } H(x, \lambda) = H(x)[\lambda]\]
Then, replace each training example by $k$ examples.

$$(x_i, Y_i) \rightarrow ((x_i, \lambda), Y_i[\lambda]) \text{ for } \lambda \in \mathcal{Y}$$

The algorithm is called AdaBoost.MH [3]
Homework hints

- Weak learning function is a simple Perceptron
- Do not forget how $\text{Alpha}$ is computed.
- Do not forget what is the $\text{Weight's}$ vector and how it is computed.
- Do not be lazy!
Conclusion

- AdaBoost does not require prior knowledge of the weak learner.
- The performance is completely dependent on the learner and the training data.
- AdaBoost can identify outliers based on the weight's vector.
- It is susceptible to noise or to examples with very large number of outliers.
