Boosting

Ying Wu

Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

http://www.eecs.northwestern.edu/~yingwu
Weak Learner and Features

- Weaker learner performs slightly better than random guessing.

- They are far from useful.

- We may have a large number of such weaker learners.

- For example, we may have a large number of features.

- Each feature leads to a weak classifier.

- Of course, it is natural to attempt to combine them.
Combining Weak Classifiers

- A feasible way of combining weak classifiers is the majority vote, or a linear combination.

- We weight individual weak classifier differently.

- A strategy of obtaining their weights is to construct an objective function of the weights and optimize it.

- Another strategy is to do it sequentially.

- Each time, we present a different data distribution.

- Based on which, we train a best weak classifier, which in turn change the data distribution.

- Finally, we have a linear combination for a strong classifier.
Outline

AdaBoost

Boosting and Logistic Regression

Boosting for Multiple Classes

Boosting for Feature Selection
The AdaBoost Algorithm

Algorithm 1: General AdaBoost Algorithm

Given: \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{X}, y_i \in \{1, -1\} \)

Init: \( W_1(i) = 1/n \)

for \( t = 1, \ldots, T \)

- Train base learner using distribution \( W_t \)
- Obtain base classifier \( h_t : \mathbb{X} \rightarrow \mathbb{R} \)
- Choose the voting weight \( \alpha_t \in \mathbb{R} \) for \( h_t \)
- Updating the data distribution

\[
W_{t+1}(i) = \frac{W_t(i)e^{-\alpha_y h_t(x_i)}}{Z_t}
\]

where \( Z_t \) normalizes \( W_{t+1} \) to be a distribution.

end

Return: a strong classifier \( H(x) = \text{sgn}\left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \)
How does it Work?

- A base (or weak) learner
- “Boost” the weak learner by feeding it different training data
- Call the weaker learner repetitively
- Each training cycle $t$ has a different distribution $W_t$ of the training data (by increasing the weights of the misclassified samples, or hard samples)
- Each time $t$, a weak classifier $h_t$ is trained and forced to focus on the hard samples
- We obtain a sequence of weak classifiers $\{h_t\}$ and their voting weights $\alpha_t$
- The final strong classifier is the linear combination of these weak classifiers
The Base Learner

- The base learner gives a base (or weak) classifier $h_t : \mathbb{X} \rightarrow \mathbb{R}$ at each training cycle $t$
- In the general framework, the output of the weak classifier is a real value
- Here the weak classifier $h_t$ achieves Bayesian error given the current data distribution $W_t$

$$h_t = \arg\min_{h} \epsilon_t(h) = \sum_{h(x_i) \neq y_i} W_t(x_i)$$

- The simplest case is $h_t : \mathbb{X} \rightarrow \{-1, +1\}$, i.e., binary output
- In this case, the optimal voting weights are

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$
AdaBoost for Binary Weak Learners

Algorithm 2: AdaBoost Algorithm for Binary Weak Learners

Given: \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{X}, y_i \in \{1, -1\} \)

Init: \( W_1(i) = 1/n \)

for \( t = 1, \ldots, T \)

- Train base learner using distribution \( W_t \)
- Obtain base classifier \( h_t : \mathbb{X} \rightarrow \{-1, +1\} \), and its error rate \( \epsilon_t \)
- Choose the voting weight \( \alpha_t \in \mathbb{R} \) for \( h_t \)

\[
\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]

- Updating the data distribution

\[
W_{t+1}(i) = \frac{W_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}
\]

end

Return: a strong classifier \( H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \)
Let’s define
\[
f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)
\]

The training error rate of \( H(x) \) is bounded by
\[
E_H \triangleq \frac{1}{n} \left| \{ i : H(x_i) \neq y_i \} \right| \leq \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(x_i)} \quad \text{prove it}
\]

More significantly, we have
\[
\frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(x_i)} = \prod_{t=1}^{T} Z_t \quad \text{prove it}^1
\]

\(^1\)hint: \( W_{t+1}(i) \propto \exp \left\{ -y_i \sum_{k=1}^{t} \alpha_k h_k(x_i) \right\} \)
A Greedy Strategy

- Let’s see it again

\[ E_H \leq \prod_{t=1}^{T} Z_t \]

- Of course, the best \( H(x) \) is the one that minimizes \( E_H \)

- To minimize \( E_H \), a **greedy** strategy is to minimize \( Z_t \) at **each** training cycle, i.e., choose \( \alpha_t \) and \( h_t \):

\[
\{\alpha_t, h_t\} = \arg\min_{\{\alpha, h\}} Z_t = \sum_{i=1}^{n} W_t(i) e^{-\alpha y_i h(x_i)}
\]

- Why is this greedy? Let’s discuss
Optimal $\alpha_t$ for Binary Weak Classifier

Based on this greedy strategy, we can find the optimal $\alpha_t$

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$

which was given in my previous slide (Let’s prove it)

Define

$$\gamma_t = \frac{1}{2} - \epsilon_t \quad \gamma_t > 0$$

The bound of training error is

$$E_H = \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \left[ 2 \sqrt{\epsilon_t (1 - \epsilon_t)} \right] = \prod_{t=1}^{T} \sqrt{1 - 4 \gamma_t^2}$$

$$\leq \exp \left( -2 \sum_{t=1}^{T} \gamma_t^2 \right)$$
Training Error Drops Quickly!

- We only need each weak classifier to be slightly better than random guess, i.e., $\epsilon_t$ is slightly smaller than $1/2$
- Assume $\gamma_t \geq \gamma > 0$, then we have

$$E_H \leq e^{-2T\gamma^2}$$

- $E_H$ drops exponentially fast in $T$!
Generalization Error

- It is observed that AdaBoost sometimes continue to reduce the generalization error after the training error = 0!
- WHY?
- Define the margin of training sample \( \{x, y\} \)

\[
\text{margin}_f(x, y) \triangleq \frac{y f(x)}{\sum_{t=1}^{T} |\alpha_t|} = \frac{y \sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} |\alpha_t|}
\]

- \( \text{margin}_f(x, y) \in [-1, +1] \), and is positive iff \( H(x) \) correctly classifies \( x \). It measures the confidence of prediction
- Schapire et al gave the generalization error

\[
E_g \leq \hat{P}(\text{margin}_f(x, y) \leq \theta) + O \left( \sqrt{\frac{d}{n\theta^2}} \right)
\]

for any \( \theta > 0 \). \( E_g \) is independent of \( T \)!
Outline

AdaBoost

Boosting and Logistic Regression

Boosting for Multiple Classes

Boosting for Feature Selection
Another Perspective of Boosting

- We said before that the training error of AdaBoost is bounded

\[ E_H \leq \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(x_i)} \]

- So, the training of \( H \) actually attempts to minimize the exponential loss

\[ J(\{\alpha_t, h_t\}) = \sum_{i=1}^{n} e^{-y_i f(x_i)} = \sum_{i=1}^{n} \exp \left\{ -y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) \right\} \]

- AdaBoost performs steepest descent search to minimize the exponential loss \( J \)
Logistic Regression

- For a classifier, it is useful to estimate the confidence (or the probability) of the classification.

- We may use the output of AdaBoost, i.e., $f(x)$ to make the estimation

$$P(y = +1|x) = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}}, \quad P(y = -1|x) = \frac{e^{-f(x)}}{e^{f(x)} + e^{-f(x)}}$$

- A unified formulae

$$P(y|x) = \frac{e^{yf(x)}}{e^{yf(x)} + e^{-yf(x)}} = \frac{1}{1 + e^{-2yf(x)}}$$

- We define the **logistic loss**, i.e., the negative log likelihood

$$L = \sum_{i=1}^{n} \ln \left( 1 + e^{-2y_if(x_i)} \right)$$
**J and L are Closely Related**

- The behavior of $J$ and $L$ are very similar near zero, as they have the same Taylor expansion around zero up to second order.

- It can be shown that

$$E \left[ \ln \left( 1 + e^{-2yf(x)} \right) \right], \text{ and } E \left[ e^{-yf(x)} \right]$$

are minimized by the same function $f$

$$f(x) = \frac{1}{2} \ln \frac{P(y = +1|x)}{P(y = -1|x)}$$

- Minimizing $J$ in AdaBoost can be viewed as approximation of minimizing $L$, so we can use the output of $f(x)$ to estimate the posterior.

- In addition, one can directly minimize $L$, instead of $J$.
Let’s redefine the objective function to be a logistic loss

\[
L(\{\alpha_t, h_t\}) = \sum_{i=1}^{n} \ln \left( 1 + e^{-y_i f(x_i)} \right)
\]

Define \( f_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x) \). The weights in AdaBoost is

\[
W_t(i) \propto e^{-y_i f_{t-1}(x_i)}
\]

The only modification in LogitBoost is to refine \( W_t(i) \)

\[
W_t(i) \propto \frac{1}{1 + e^{y_i f_{t-1}(x_i)}}
\]
Outline

- AdaBoost
- Boosting and Logistic Regression
- Boosting for Multiple Classes
- Boosting for Feature Selection
AdaBoost for Multiple Classes

**Algorithm 3: AdaBoost.M1**

**Given:** \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{X} \), \( y_i \in \mathbb{Y} = \{1, \ldots, k\} \)

**Init:** \( W_1(i) = 1/n \)

**for** \( t = 1, \ldots, T \)

- Train a weak classifier \( h_t : \mathbb{X} \rightarrow \mathbb{Y} \)
- Compute its error \( \epsilon_t \). If \( \epsilon > 1/2 \), set \( T = t - 1 \) and abort loop
- Set \( \beta_t = \frac{\epsilon_t}{1-\epsilon_t} \) and \( \alpha_t = -\log \beta_t \)
- Updating the data distribution

\[
W_{t+1}(i) = \begin{cases} 
W_t(i)\beta_t & \text{if } h_t(x_i) = y_i \\
W_t(i) & \text{otherwise}
\end{cases}
\]

- Normalize \( W_{t+1} \)

**end**

**Return:** a strong classifier \( H(x) = \arg \max_{y \in \mathbb{Y}} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \)
Algorithm 4: AdaBoost.M2

**Given:** \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{X}, y_i \in \mathbb{Y} = \{1, \ldots, k\} \)

Let \( B = \{(i, y) : i = 1, \ldots, n, y \neq y_i\} \)

**Init:** \( W_1(i, y) = 1/|B| \), for \((i, y) \in B\)

for \( t = 1, \ldots, T \)

- Train a weak classifier \( h_t : \mathbb{X} \times \mathbb{Y} \rightarrow [0, 1] \)
- Pseudo-loss \( \epsilon_t = \frac{1}{2} \sum_{(i, y) \in B} W_t(i, y)(1 - h_t(x_i, y_i) + h_i(x_i, y)) \)
- Set \( \beta_t = \frac{\epsilon_t}{1-\epsilon_t} \) and \( \alpha_t = -\log \beta_t \)
- Updating the data distribution

\[
W_{t+1}(i) = W_t(i)\beta_t^{(1-h_t(x_i,y_i)+h_i(x_i,y))/2}
\]

- Normalize \( W_{t+1} \)

end

**Return:** a strong classifier \( H(x) = \arg \max_{y \in \mathbb{Y}} \left( \sum_{t=1}^{T} \alpha_t h_t(x, y) \right) \)
Outline

AdaBoost

Boosting and Logistic Regression

Boosting for Multiple Classes

Boosting for Feature Selection
Let’s have a Real Example: Face Detection

How many features do we have for face detection?

2 Viola&Jones, CVPR’01
Algorithm 5: AdaBoost for Feature Selection

Given: \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{X} \), \( y_i \in \{1, -1\} \)

Init: \( W_1(i) = 1/n \)

for \( t = 1, \ldots, T \)

- Normalize the weights \( W_t \)
- \( \forall \) feature \( j \), train \( h_j \) and obtain its error rate \( \epsilon_j \)
- Choose the classifier \( h_t \) with the lowest \( \epsilon_t \)
- Set \( \beta_t = \frac{\epsilon_t}{1-\epsilon_t} \) and \( \alpha_t = -\log \beta_t \)
- Updating the data distribution

\[
W_{t+1}(i) = \begin{cases} 
W_t(i)\beta_t & \text{if } h_t(x_i) = y_i \\
W_t(i) & \text{otherwise}
\end{cases}
\]

end

Return: a strong classifier \( H(x) = sgn \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \)