Ensemble Learning, Boosting & AdaBoost

Isaac Mackey

Some of the slides are borrowed from Derek Hoiem, Jan Sochman, Hongbo Deng.
Ensemble Learning ("Wisdom of Crowds")
Ensemble Learning ("Wisdom of Crowds")
Ensemble Learning ("Wisdom of Crowds")
How to make a crowd?

How to generate a set of (weak) classifiers?
Bagging

Step 1: Create Multiple Data Sets

D → D_1 → C_1
D → D_2 → C_2
... → D_t → C_t

Step 2: Build Multiple Classifiers

D_1 → C_1
D_2 → C_2
... → D_{t-1} → C_{t-1}
D_t → C_t

Step 3: Combine Classifiers

C^*
Bagging

Step 1: Create Multiple Data Sets
- $D_1$
- $D_2$
- $\cdots$
- $D_{t-1}$
- $D_t$

Step 2: Build Multiple Classifiers
- $C_1$
- $C_2$
- $C_{t-1}$
- $C_t$

Step 3: Combine Classifiers
- $C^*$

“Ensemble”
Boosting (Bagging + Memory)
Boosting (Bagging + Memory)
Boosting (Bagging + Memory)
Boosting (Bagging + Memory)
Boosting (Bagging + Memory)

BAG

1
2
3
4
5

BAG

6
3
3
5
5

BAG

6
6
7
8
9

BAG

10
11
8
8
9

C_1

C_2

C_{t-1}

C_t

C^*
Boosting (Bagging + Memory)

"Ensemble"
AdaBoost (Adaptive Boosting)

- Uses boosting to construct **weighted classifiers**

- Successful applications in facial recognition software
  - Combination of many indicative but weak features

- It won the Gödel Prize in 2003.
  - Combination of “weak” classifiers to produce “strong” classifier
AdaBoost (*Ada*ptive Boosting)

- Uses boosting to construct *weighted classifiers*
- Successful applications in facial recognition software
  - Combination of many indicative but weak features
- It won the Gödel Prize in 2003.
  - Combination of “weak” classifiers to produce “strong” classifier
AdaBoost (Adaptive Boosting)

- Intuition: Sample from distribution and update distribution after each classification based on performance, targeting errors
AdaBoost (Adaptive Boosting)

- Intuition: Sample from distribution and update distribution after each classification based on performance, targeting errors

  Did new classifier label this example correctly?

  - Decrease weight of example
  - Increase weight of example
AdaBoost (Adaptive Boosting)

- Intuition: Sample from distribution and update distribution after each classification based on performance, targeting errors

  Did new classifier produce low error?

  Increase weight of classifier  Decrease weight of classifier
AdaBoost (Adaptive Boosting)

weak classifier $h(x) = \text{perceptron}$
AdaBoost (Adaptive Boosting)

weak classifier $h(x) = \text{perceptron}$
AdaBoost (Adaptive Boosting)

weak classifier $h(x) = \text{perceptron}$
AdaBoost (Adaptive Boosting)

weak classifier \( h(x) = \text{perceptron} \)
AdaBoost (Adaptive Boosting)

weak classifier $h(x) = \text{perceptron}$
AdaBoost (Adaptive Boosting)

weak classifier $h(x) = \text{perceptron}$
weak classifier $h(x) =$ perceptron

The final classifier is a weighted sum of weak classifiers.

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$H(x)$ makes 0 errors as each point is classified correctly at least twice.
AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X\), \(y_i \in Y = \{-1, +1\}\)

 Initialise \(D_1(i) = \frac{1}{m}\).

For \(t = 1, \ldots, T\):

- Find the classifier \(h_t : X \to \{-1, +1\}\) that minimizes the error with respect to the distribution \(D_t\):
  \[
  h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j, \text{ where } \epsilon_j = \sum_{i=1}^{m} D_t(i) [y_i \neq h_j(x_i)]
  \]

- Prerequisite: \(\epsilon_t < 0.5\), otherwise stop.

- Choose \(\alpha_t \in \mathbb{R}\), typically \(\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}\) where \(\epsilon_t\) is the weighted error rate of classifier \(h_t\).

- Update:
  \[
  D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]

  where \(Z_t\) is a normalisation factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialise \(D_1(i) = \frac{1}{m}\).

For \(t = 1, \ldots, T\):

- **Find the classifier** \(h_t : X \to \{-1, +1\}\) that minimizes the error with respect to the distribution \(D_t\):
  \[
  h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j, \text{ where } \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]
  \]

  - Prerequisite: \(\epsilon_t < 0.5\), otherwise stop.

- **Choose** \(\alpha_t \in \mathbb{R}\), typically \(\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}\) where \(\epsilon_t\) is the weighted error rate of classifier \(h_t\).

- **Update**:
  \[
  D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]

  where \(Z_t\) is a normalisation factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
AdaBoost Algorithm (essentially)

To get $t$ classifiers for distribution $D$:

1) Generate a classifier $h(t)$ for $D(t)$

2) Assign weight $\alpha(t)$ to $h(t)$ based on error

3) Adjust $D(t)$ based on error of $h(t)$ to produce $D(t+1)$

Weighted sum of $\alpha(t) * h(t)$ produces classification
AdaBoost Algorithm (essentially)

To get $t$ classifiers for distribution $D$: 

1) Generate a classifier $h(t)$ for $D(t)$

2) Assign weight $\alpha(t)$ to $h(t)$ based on error

3) Adjust $D(t)$ based on error of $h(t)$ to produce $D(t+1)$

Weighted sum of $\alpha(t) * h(t)$ produces classification
The algorithm core (essentially) \( \alpha(t) \) will converge, but not too quickly. This allows for generalization.
Adjusting Distribution

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q} \]

\[
\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}
\]

⇒ Increase (decrease) weight of wrongly (correctly) classified examples
Adjusting Distribution

Effect on the training set

Reweighting formula:

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}
\]

\[
\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}
\]

⇒ Increase (decrease) weight of wrongly (correctly) classified examples
Adjusting Distribution
Adjusting Distribution

misclassified points increase weight
Adjusting Distribution

Really hard points
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- If $\epsilon_t \geq 1/2$ then stop

- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$

- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) \cdot [y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- If $\epsilon_t \geq 1/2$ then stop

- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1 + r_t}{1 - r_t} \right)$

- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Example

Initialization...
For \( t = 1, \ldots, T \):

- Find \( h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)] \)
- If \( \epsilon_t \geq 1/2 \) then stop
- Set \( \alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right) \)
- Update
  \[
  D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Initialization...

For $t = 1, \ldots, T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i)e^{\alpha_t y_i h_t(x_i)}}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$
What’s So Good About AdaBoost

- Can be used with many different, weak classifiers
What’s So Good About AdaBoost

- Simple to implement

Initialization...
For $t = 1, \ldots, T$:

- Find $h_t = \arg \min_{h_j \in H} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

To get $t$ classifiers for distribution $D$:

1) Generate a classifier $h(t)$ for $D(t)$
2) Assign weight $\alpha(t)$ to $h(t)$ based on error
3) Adjust $D(t)$ based on error of $h(t)$ to produce $D(t+1)$

Simple
What’s So Good About AdaBoost

- Not prone to overfitting: it uses weak learners and re-weighting of distribution can be minimal.
What’s So Good About AdaBoost

- Test error can improve after training error reaches 0
References

- Freund – “An adaptive version of the boost by majority algorithm”
- Freund – “Experiments with a new boosting algorithm”
- Friedman, Hastie, etc – “Additive Logistic Regression: A Statistical View of Boosting”
- Opitz, Maclin – “Popular Ensemble Methods: An Empirical Study”
- Ratsch, Warmuth – “Efficient Margin Maximization with Boosting”
- Schapire, Singer – “Improved Boosting Algorithms Using Confidence-Weighted Predictions”
- Zhang, Li, etc – “Multi-view Face Detection with Floatboost”