Outline

- Background
- Adaboost Algorithm
- Theory/Interpretations
What’s So Good About Adaboost

- Can be used with many different classifiers
- Improves classification accuracy
- Commonly used in many areas
- Simple to implement
- Not prone to overfitting
A Brief History

- Bootstrapping
- Bagging
- Boosting (Schapire 1989)
- Adaboost (Schapire 1995)
Bootstrap Estimation

- Repeatedly draw $n$ samples from $D$
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance
Bagging - Aggregate Bootstrapping

- For $i = 1 \ldots M$
  - Draw $n^* < n$ samples from $D$ with replacement
  - Learn classifier $C_i$
- Final classifier is a vote of $C_1 \ldots C_M$
- Increases classifier stability/reduces variance
Boosting (Schapire 1989)

- Consider creating three component classifiers for a two-category problem through boosting.
- Randomly select $n_1 < n$ samples from $D$ without replacement to obtain $D_1$
  - Train weak learner $C_1$
- Select $n_2 < n$ samples from $D$ with half of the samples misclassified by $C_1$ to obtain $D_2$
  - Train weak learner $C_2$
- Select all remaining samples from $D$ that $C_1$ and $C_2$ disagree on
  - Train weak learner $C_3$
- Final classifier is vote of weak learners
Adaboost - Adaptive Boosting

- Instead of resampling, uses training set re-weighting
  - Each training sample uses a weight to determine the probability of being selected for a training set.

- AdaBoost is an algorithm for constructing a “strong” classifier as linear combination of “simple” “weak” classifier
  
  \[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

- Final classification based on weighted vote of weak classifiers
Adaboost Terminology

- $h_t(x)$ … “weak” or basis classifier (Classifier = Learner = Hypothesis)
- $H(x) = \text{sign}(f(x))$ … “strong” or final classifier

- Weak Classifier: < 50% error over any distribution
- Strong Classifier: thresholded linear combination of weak classifier outputs
Discrete Adaboost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) \subseteq \{ -1, +1 \}

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

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

- Find the classifier \(h_t : X \rightarrow \{ -1, +1 \}\) 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:

\[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]
Find the Weak Classifier

**Loop step:** Call *WeakLearn*, providing it with the distribution \( D_t \); get back weak classifier \( h_t : \mathcal{X} \rightarrow \{-1, 1\} \) from \( \mathcal{H} = \{ h(x) \} \)

- Select a weak classifier with the smallest weighted error
  \[ h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)] \]

- Prerequisite: \( \epsilon_t < 1/2 \) (otherwise stop)

- *WeakLearn* examples:
  - Decision tree builder, perceptron learning rule – \( \mathcal{H} \) infinite
  - Selecting the best one from given *finite* set \( \mathcal{H} \)

**Demonstration example**

Training set

Weak classifier = perceptron

\[ \bullet \sim N(0, 1) \quad \bullet \sim \frac{1}{2\pi} e^{-1/2(r-4)^2} \]
Find the Weak Classifier

Loop step: Call \textit{WeakLearn}, providing it with the distribution $D_t$; get back weak classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  \[ h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)] \]

- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)

- \textit{WeakLearn} examples:
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ infinite
  - Selecting the best one from given finite set $\mathcal{H}$

Demonstration example

Training set \hspace{2cm} Weak classifier = perceptron

\[ \bullet \sim \mathcal{N}(0, 1) \hspace{1cm} \bullet \sim \frac{1}{2\pi} e^{-1/2(r-4)^2} \]
The algorithm core

- The main objective is to minimize \( \varepsilon_{tr} = \frac{1}{m} | \{ i : H(x_i) \neq y_i \} | \)

- It can be upper bounded by \( \varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t \)

How to set \( \alpha_t \)?

- Select \( \alpha_t \) to greedily minimize \( Z_t(\alpha) \) in each step

- \( Z_t(\alpha) \) is convex differentiable function with one extremum

  \( \Rightarrow h_t(x) \in \{-1, 1\} \) then optimal \( \alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right) \)

  where \( r_t = \sum_{i=1}^{m} D_t(i) h_t(x_i) y_i \)

- \( Z_t = 2 \sqrt{\varepsilon_t(1-\varepsilon_t)} \leq 1 \) for optimal \( \alpha_t \)

  \( \Rightarrow \) Justification of selection of \( h_t \) according to \( \varepsilon_t \)
Reweighting

Effect on the training set

Reweighting formula:

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

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

\[\Rightarrow\] Increase (decrease) weight of wrongly (correctly) classified examples
Reweighting

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

In this way, AdaBoost “focused on” the informative or “difficult” examples.
Reweighting

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

In this way, AdaBoost “focused on” the informative or “difficult” examples.
Algorithm recapitulation

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:

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

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

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

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

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]
- 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:

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

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

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

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

Output the final classifier:

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

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

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

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

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

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

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

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]
- 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:

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

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]
- 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:

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

Advantages
- Very simple to implement
- Does feature selection resulting in relatively simple classifier
- Fairly good generalization

Disadvantages
- Suboptimal solution
- Sensitive to noisy data and outliers
References

- Duda, Hart, ect – *Pattern Classification*
- Freund – “An adaptive version of the boost by majority algorithm”
- Freund – “Experiments with a new boosting algorithm”
- Freund, Schapire – “A decision-theoretic generalization of on-line learning and an application to boosting”
- Friedman, Hastie, etc – “Additive Logistic Regression: A Statistical View of Boosting”
- Jin, Liu, etc (CMU) – “A New Boosting Algorithm Using Input-Dependent Regularizer”
- Li, Zhang, etc – “Floatboost Learning for Classification”
- Opitz, Maclin – “Popular Ensemble Methods: An Empirical Study”
- Ratsch, Warmuth – “Efficient Margin Maximization with Boosting”
- Schapire, Freund, etc – “Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods”
- Schapire, Singer – “Improved Boosting Algorithms Using Confidence-Weighted Predictions”
- Zhang, Li, etc – “Multi-view Face Detection with Floatboost”
Appendix

- Bound on training error
- Adaboost Variants
Bound on Training Error (Schapire)

\[
\frac{1}{m} \sum_i \left[ H(x_i) \neq y_i \right] \leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) \\
= \sum_i \left( \prod_t Z_t \right) D_{T+1}(i) \\
= \prod_t Z_t .
\]

\[
Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))
\]
Discrete Adaboost (DiscreteAB) (Friedman’s wording)

Discrete AdaBoost (Freund & Schapire 1996b)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.

2. Repeat for $m = 1, 2, \ldots, M$:

   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   (b) Compute $err_m = E_w[1_{y \neq f_m(x)}]$, $c_m = \log((1 - err_m)/err_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{y_i \neq f_m(x_i)}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.

3. Output the classifier $\text{sign}[\sum_{m=1}^M c_m f_m(x)]$
Discrete Adaboost (DiscreteAB) (Freund and Schapire’s wording)

Algorithm AdaBoost

Input: sequence of $N$ labeled examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$

- distribution $D$ over the $N$ examples
- weak learning algorithm WeakLearn
- integer $T$ specifying number of iterations

Initialize the weight vector: $w_i^t = D(i)$ for $i = 1, \ldots, N$.

Do for $t = 1, 2, \ldots, T$

1. Set
   $$p_i^t = \frac{w_i^t}{\sum_{i=1}^{N} w_i^t}$$

2. Call WeakLearn, providing it with the distribution $p_i^t$; get back a hypothesis $h_t : X \to [0, 1]$.

3. Calculate the error of $h_t$: $\epsilon_t = \sum_{i=1}^{N} p_i^t |h_t(x_i) - y_i|$.

4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.

5. Set the new weights vector to be
   $$w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \left( \log \frac{1}{\beta_t} \right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \geq 0, \\ 0 & \text{otherwise} \end{cases}$$
Adaboost with Confidence Weighted Predictions (RealAB)

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)
Initialize \(D_1(i) = 1/m\).
For \(t = 1, \ldots, T:\)

- Train base learner using distribution \(D_t\).
- Get base classifier \(h_t : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- 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 normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final classifier:

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

Figure 1: The boosting algorithm AdaBoost.
Adaboost Variants Proposed By Friedman

- **LogitBoost**
  - Solves
    \[
    \min_{f(x)} E_w(x) \left( F(x) + \frac{1}{2} \frac{y^* - p(x)}{p(x)(1 - p(x))} - (F(x) + f(x)) \right)^2
    \]
  - Requires care to avoid numerical problems

- **GentleBoost**
  - Update is \( f_m(x) = P(y=1 \mid x) - P(y=0 \mid x) \) instead of \( f_m(x) = \frac{1}{2} \log \frac{P_w(y=1|x)}{P_w(y=-1|x)} \)
    - Bounded [0, 1]
Adaboost Variants Proposed By Friedman

- LogitBoost

LogitBoost (2 classes)
1. Start with weights $w_i = 1/N$ for $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights
       $$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}$$
       $$w_i = p(x_i)(1 - p(x_i))$$
   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of $z_i$ to $x_i$ using weights $w_i$.
   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}$.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

**Algorithm 3**: An adaptive Newton algorithm for fitting an additive logistic regression model.
Adaboost Variants Proposed By Friedman

- GentleBoost

**Gentle AdaBoost**

1. Start with weights \( w_i = 1/N, \ i = 1, 2, \ldots, N, \ F(x) = 0 \).
2. Repeat for \( m = 1, 2, \ldots, M \):
   (a) Fit the regression function \( f_m(x) \) by weighted least-squares of \( y_i \) to \( x_i \) with weights \( w_i \).
   (b) Update \( F(x) \leftarrow F(x) + f_m(x) \)
   (c) Update \( w_i \leftarrow w_i e^{-y_if_m(x_i)} \) and renormalize.
3. Output the classifier \( \text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)] \)

**Algorithm 4:** A modified version of the Real AdaBoost algorithm, using Newton stepping rather than exact optimization at each step
Thanks!!!
Any comments or questions?