Boosting

Instructor: Pranjal Awasthi
Motivating Example

Want to predict winner of 2016 general election
Motivating Example

• Step 1: Collect training data
  – 2012,
  – 2008,
  – 2004,
  – 2000,
  – 1996,
  – 1992,
  – ........

What if instead, we have good “rules of thumb” that often work well?

• Step 2: Find a function that gets high accuracy (~99%) on training set.
Motivating Example

• Step 1: Collect training data
  – 2012, 
  – 2008, 
  – 2004, 
  – 2000, 
  – 1996, 
  – 1992, 
  – ........

  \[ h_1 = \text{If sitting president is running, go with his/her party,}\]
  \[ \text{Otherwise, predict randomly} \]

• \( h_1 \) will do well on a subset of train set, and will do better than half overall, say 51% accuracy.
Motivating Example

• Step 1: Collect training data
  - ,
  - 2008 🐘
  - ,
  - 2000, 🐘
  - ,
  - ,
  - ..........

$h_1 =$ If sitting president is running, go with his/her party,
Otherwise, predict randomly

What if only focus on a subset of examples: $h_1$ will do poorly.
Motivating Example

• Step 1: Collect training data
  – ,
  – 2008 🇺🇸
  – ,
  – 2000, 🇺🇸
  – ,
  – ,
  – ........

$h_2$ = If CNN poll gives a 20pt lead to someone, go with his/her party, Otherwise, predict randomly

What if only focus on a subset of examples: $h_1$ will do poorly. But $h_2$ will do well.

• $h_2$ will do well, say 51% accuracy.
Motivating Example

• Step 1: Collect training data
  – 2012, 
  – 2008, 
  – 2004, 
  – 2000, 
  – 1996, 
  – 1992, 
  – .......... 

• Suppose can consistently produce rules in $H$ that do slightly better than random guessing.
Motivating Example

- Step 1: Collect training data
  - 2012,
  - 2008,
  - 2004,
  - 2000,
  - 1996,
  - 1992,
  - ........

- Output a single rule that gets high accuracy on training set.
Boosting: A general method for converting “rules of thumb” into highly accurate predictions.
History of Boosting

• Originally studied from a purely theoretical perspective
  – Is weak learning equivalent to strong learning?
  – Answered positively by Rob Schapire in 1989 via the boosting algorithm.

• AdaBoost -- A highly practical version developed in 1995
  – One of the most popular ML (meta)algorithms
History of Boosting

• Has connections to many different areas
  – Empirical risk minimization
  – Game theory
  – Convex optimization
  – Computational complexity theory
Strong Learning

• To get strong learning, need to
  – Design an algorithm that can get any arbitrary error $\epsilon > 0$, on the training set $S_m$
  – Do VC dimension analysis to see how large $m$ needs to be for generalization
Theory of Boosting

• Weak Learning Assumption:
  – There exists algorithm $A$ that can consistently produce weak classifiers on the training set
    
    
    $S_m = (\tilde{X}_1, y_1), (\tilde{X}_2, y_2) ...$
  – Weak Classifier:
    • For every weighting $W$ of $S_m$, $A$ outputs $h_W$ such that
      
      
      $\text{err}_{S_m,W}(h_W) \leq \frac{1}{2} - \gamma, \quad \gamma > 0$
    • $\text{err}_{S_m,W}(h_W) = \sum_{i=1}^{m} \frac{w_i I(h_W(\tilde{X}_i) \neq y_i)}{\sum_{i=1}^{m} w_i}$

$e^{\text{err}}_{S_m}(h) = \sum_{i=1}^{m} w_i$
Theory of Boosting

• Weak Learning Assumption:
  – There exists an algorithm $A$ that can consistently produce weak classifiers on the training set $S_m$

• Boosting Guarantee: Use $A$ to get algorithm $B$ such that
  – For every $\epsilon > 0$, $B$ outputs $f$ with $err_{S_m}(f) \leq \epsilon$
AdaBoost

- Run $A$ on $S_m$ get $h_1$
- Use $h_1$ to get $W_2$
- Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
- Repeat $T$ times
- Combine $h_1, h_2, \ldots, h_T$ to get the final classifier.

Q1: How to choose weights?
Q2: How to combine classifiers?
Choosing weights

- Run $A$ on $S_m$ get $h_1$
- Use $h_1$ to get $W_2$
- Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
- Repeat $T$ times
- Combine $h_1, h_2, \ldots, h_T$ to get the final classifier.

Idea 1: Let $W_t$ be uniform over examples that $h_{t-1}$ got incorrectly

- $h_2$ could be $-h_1$
- $h_3$ could be $+h_1$
- No new information accumulated in each step.
Choosing weights

- Run $A$ on $S_m$ get $h_1$
- Use $h_1$ to get $W_2$
- Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
- Repeat $T$ times
- Combine $h_1, h_2, \ldots, h_T$ to get the final classifier.

Idea 2: Let $W_t$ be such that error of $h_{t-1}$ becomes $\frac{1}{2}$.

\[ \omega(A) = W_{t-1}(x, y) \]

\[ S \text{ (die } A \text{ by } \frac{1}{2} - y \text{ ) } \]

\[ \omega(B) = W_{t-1}(\frac{1}{2} - y) \]

$A = h_{t-1}$ is correct

$B = h_{t-1}$ is incorrect
Choosing weights

- Run $A$ on $S_m$ get $h_1$
- Use $h_1$ to get $W_2$
- Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
- Repeat $T$ times
- Combine $h_1, h_2, \ldots, h_T$ to get the final classifier
- If $\text{err}_{S_m,W_{t-1}}(h_{t-1}) = \frac{1}{2} - \gamma$, then
  - $W_{t,i} = W_{t-1,i}$, if $h_{t-1}$ is incorrect on $\vec{X}_i$
  - $W_{t,i} = \frac{W_{t-1,i}(\frac{1}{2} - \gamma)}{(\frac{1}{2} + \gamma)}$, if $h_{t-1}$ is correct on $\vec{X}_i$
Combining Classifiers

- Run $A$ on $S_m$ get $h_1$
- Use $h_1$ to get $W_2$
- Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
- Repeat $T$ times
- Combine $h_1, h_2, \ldots, h_T$ to get the final classifier.

Take majority vote $f(\vec{X}) = MAJ(h_1(\vec{X}), h_2(\vec{X}), \ldots, h_T(\vec{X}))$

$$= \arg\max_{y \in \mathbb{Y}} \left( \sum_{i=1}^{m} h_i(x) \right)$$
Combining Classifiers

• Run $A$ on $S_m$ get $h_1$
• Use $h_1$ to get $W_2$
• Run $A$ on $S_m$ with weights $W_2$ to get $h_2$
• Repeat $T$ times
• Combine $h_1, h_2, ... h_T$ to get the final classifier.

Take majority vote $f(\overrightarrow{X}) = MAJ(h_1(\overrightarrow{X}), h_2(\overrightarrow{X}), ... h_T(\overrightarrow{X}))$

Theorem:
If error of each $h_i$ is $\frac{1}{2} - \gamma$, then $err_{S_m}(f) \leq e^{-2T\gamma^2}$
Analysis

Theorem:
If error of each $h_i$ is $\frac{1}{2} - \gamma$, then $err_{sm}(f) \leq e^{-2T\gamma^2}$

Let's analyze total weight $W_t$ examples at each step:

Initially $W_t = m$

Given $W_{t-1} < h_{t-1}$:

- $A$: examples $h_{t-1}$ gets correctly
- $B$: examples $h_{t-1}$ gets incorrectly

$w(A) = W_{t-1} \left(\frac{1}{2} + \gamma\right)$

$w(B) = W_{t-1} \left(\frac{1}{2} - \gamma\right)$

$A$ is being scaled by $\frac{\frac{1}{2} - \gamma}{\frac{1}{2} + \gamma}$

$W_t = W_{t-1} \left(\frac{1}{2} \cdot \gamma\right) \left(\frac{1}{2} - \gamma\right) + W_{t-1} \left(\frac{1}{2} - \gamma\right) = W_{t-1} (1 - 2\gamma)$
Analysis

Theorem:
If error of each $h_i$ is $\frac{1}{2} - \gamma$, then $\text{err}_{sm}(f) \leq e^{-2TY\gamma^2}$
Analysis

Theorem:
If error of each $h_i$ is $\frac{1}{2} - \gamma$, then $err_{sm}(f) \leq e^{-2T\gamma^2}$
Analysis

Theorem:

If error of each $h_i$ is $\leq \frac{1}{2} - \gamma$, then $\text{err}_{sm}(f) \leq e^{-2T\gamma^2}$

Let $W_t$ be such that error of $h_{t-1}$ becomes $\frac{1}{2}$.

Take (weighted) majority vote

$$f = \text{sgn} \left( \frac{1}{2} \sum_{i=1}^{n} w_i \cdot h_i(x) \right)$$

$$\lambda_t = \ln \left( \frac{\frac{1}{2} + \gamma}{\frac{1}{2} - \gamma} \right)$$

$$\text{scdir} = \left( \frac{\frac{1}{2} - \gamma}{\frac{1}{2} + \gamma} \right)$$
Strong Learning

• To get strong learning, need to
  – Design an algorithm that can get any arbitrary error $\epsilon > 0$, on the training set $S_m$
  – Do VC dimension analysis to see how large $m$ needs to be for generalization
Generalization Bound

\[ h_1, h_2, h_3, \ldots \]

\[ VC \dim(H) = d \]

\[ f = \text{MAJ}(h_1, h_2, \ldots, h_T) \]

\[ VC \dim(f) \geq Td \logTd \]

\[ m \geq \frac{d^l}{\epsilon^2} > \frac{Td \log Td}{\epsilon^2} \]

What is \( T \)?

\[ e^{-2T \gamma^2} \leq \epsilon \]

\[ T > \frac{1}{2\gamma^2} \log \left( \frac{1}{\epsilon} \right) \]
Applications: Decision Stumps

\[ y = f(x_1 - - x_n) \text{ ; } x_i \in \{0, 1\} \]

Stumps:

\[ +1 \quad -1 \quad +1 \quad -1 \]

\text{Stumps + Boosting} \rightarrow \text{usually don't overfit.}
Advantages of AdaBoost

• Helps learn complex models from simple ones without overfitting
• A general paradigm and easy to implement
• Parameter free, no tuning
• Often not very sensitive to the number of rounds $T$

Cons:
  – Does not help much if weak learners are already complex.
  – Sensitive to noise in the data