AdaBoost (Freund & Schapire '97)
"Ada Boost" is short for "Adaptive Boosting"
- very widely used, one of the most popular machine
  learning algorithms
- "An Empirical Comparison of Supervised Learning
  Algorithms" Caruana & Niculescu-Mizil '06
  1) Boosted DT
  2) Boosted Shpapes
  3) LDA

- Top 10 Algorithms in Data Mining (We at dhi)
  presents lists identified by IEEE Computer
  ICBO in Dec 2006 (Popularly ranked)
  1) SVM
  2) AdaBoost

- Freund & Schapire received the 2003 Cradel
  Prize for their work on AdaBoost.
- "meta" algorithm: can be used to improve
  ("boost") the performance of another learning
  algorithm
- very easy to implement, there are no
  standard software packages because you
  just do it yourself.

A Little History
AdaBoost came out of the PAC Learning
 community.
- PAC Learning Model was developed by Valiant 1985
- Kearns Valiant (1987, 1989) proved the
  guarantee of whether a "weak" PAC Learning
  algo, i.e., one that performs slightly better than
  a random guess could be used to construct
  an "strong" PAC Learning algorithm, i.e., one
  that performs arbitrarily well (not all polynomial
  other technical conditions)
- Freund & Schapire's example: betting strategies for
  horse racing:
  One expert gambler may not be able to
  easily learn betting strategy, but even
  prediction of data for a specific set of
  races, she can give us a "rule of thumb"
  - bet on the horse that had won the
    most recently
    - bet on the horse of the most favored odd
    - rules of thumb are not very accurate, but a
      little better than a random guess
  - boosting algorithms combine these rules of thumb into a
    single, highly accurate prediction rule.
- Schapire (1990) showed that the answer to Kearns
  & Valiant's question was "yes". Their method is called
  AdaBoost, and the algorithm is called
  AdaBoost algorithm (Short description)
- AdaBoost (1997) a few other boosting algo were
  invented but not used very practical
- Freund & Schapire (1995) a decision-theoretic
  generalization of online learning. i.e., adaptive
  learning (1995) - enhanced AdaBoost
- Schapire & Singer (1999) extended AdaBoost to more
  general, convex losses.
LinkedIn (2010) +40000 others
Feb 19-2:35 PM
Standard Classification Task:

Let \( X \in \mathbb{R}^m \times \mathbb{R}^n \), \( y \in \{-1, 1\} \)

We want to find a function \( f : X \rightarrow \{-1, 1\} \) such that \( f(x) \) agrees with \( y \) as often as possible.

Let \( \Delta \) denote the classification error, i.e., \( \Delta = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[f(x_i) \neq y_i]} \)

We want to minimize \( \Delta \).

Note: A solution to the above problem may not be unique. It may be difficult to compute the exact solution.

For some functions, we can obtain a global solution.

Some of these functions are convex, and thus nice properties.

Let's perform a trick, namely to use:

\[ L(y, \hat{y}) = \frac{1}{|y|} \sum_{i=1}^{n} \mathbb{1}_{[f(x_i) \neq y_i]} \]

Hence, the classification error at each step is:

\[ \Delta = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[f(x_i) \neq y_i]} = L(y(x), \hat{y})(x) \]

We hope that choosing \( \hat{y} \) to yield small values of \( L(y, \hat{y}) \) will yield small values of the classification error.

Next, let's focus on the form of \( \hat{y} \). The AdaBoost is a binary combination of weak classifiers or rules of thumb.

Let \( \hat{y} = \sum_{i=1}^{n} \lambda_i \mathbb{1}_{[x_i \in S_i]} \) where \( \lambda_i : \mathbb{R} \rightarrow [0, 1] \)

AdaBoost's Objective Function:

\[ L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[f(x_i) \neq y_i]} \]

Want to minimize this w.r.t. \( \lambda \).

About the weak classifier \( \mathbb{1}_{[x_i \in S_i]} \):

- AdaBoost can be used in 2 ways:
  - To do most of the work is a wrapper to improve the accuracy of an already trained base learner algorithm (e.g., bagged model).
  - To do most of the work is a wrapper to improve the accuracy of an already trained base learner algorithm (e.g., boosted model).

- We assume that the \( \mathbb{1}_{[x_i \in S_i]} \) is not hard to do, so we just add to the loss.

Tactic: Derivation of Logistic Regression's Objective:

\[ \theta \approx \log \left( 1 + e^{-\theta x} \right) \]

Thus, we would derive the objective for the logistic regression model.

\[ L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + e^{-\hat{y}_i y_i} \right) \]

Tactic: Risk of SVM's Objective:

\[ \frac{1}{n} \sum_{i=1}^{n} \max(1 - y_i \hat{y}_i, 0) \]

Thus, we would have derived part of the objective for SVM.
Back to AdaBoost!

Since \( \mathcal{L}(h) \) is convex in \( \lambda \), we can use simple techniques to minimize \( \mathcal{L}(h) \) w.r.t. \( \lambda \) in \( \mathbb{R}^m \). We'll use "coordinate descent."

\[ F(x) = \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \]

for \( t = 1 \ldots T 

Step 1: Find the deepest decision \( \mathcal{d}_t \) (i.e., choose a weak classifier)

Step 2: move along that decision until \( \mathcal{L}(h) \) is minimized (i.e., choose \( \alpha \) to minimize \( \mathcal{L}(h_t + \alpha \mathcal{d}_t) \) where \( \mathcal{d}_t = \left( \frac{1}{T} \right) (x_i - \bar{x}) \))

Finally, 

**Objective** 

\[ \min_{\lambda, \alpha} \sum_{i=1}^{m} \mathcal{L}(h_t + \alpha \mathcal{d}_t) + \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \]

**Gradient Descent** 

\[ \frac{d}{d\alpha} \sum_{i=1}^{m} \mathcal{L}(h_t + \alpha \mathcal{d}_t) + \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \]

\[ = \sum_{i=1}^{m} \frac{d}{d\alpha} \mathcal{L}(h_t + \alpha \mathcal{d}_t) + \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \]

\[ = \sum_{i=1}^{m} \frac{d}{d\alpha} \mathcal{L}(h_t + \alpha \mathcal{d}_t) \]

**Step 1:** 

\[ \begin{align*}
&\lambda^{(t)} : \arg\min_{\lambda} \sum_{i=1}^{m} \mathcal{L}(h_t + \alpha \mathcal{d}_t) + \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \\
&\text{subject to:} \quad \sum_{i=1}^{m} \lambda_i \cdot h_i(x) \leq 1
\end{align*} \]

**Step 2:** 

\[ \alpha^{(t)} = \arg\min_{\alpha} \sum_{i=1}^{m} \mathcal{L}(h_t + \alpha \mathcal{d}_t) \]

Pseudocode for AdaBoost:

Given \( \{(x_i, y_i)\}_{i=1}^{n} \quad \{\lambda_i\}_{i=1}^{m} \quad T \quad \lambda_0 = 0 \)

For \( t = 1 \ldots T 

\[ \begin{align*}
&\mathcal{d}_t = \sum_{i=1}^{m} \frac{d_i}{\sum_{i=1}^{m} d_i} \quad \\text{"train weak learner with weight \( d_i \"\) for \( \mathcal{d}_t \"\)}
\end{align*} \]

\[ \delta_t = \sum_{i=1}^{m} \mathcal{d}_t(i) \quad \\text{"error of weak classifier \( \"\)}
\end{align*} \]

\[ \lambda_t = \lambda_0 + \frac{1}{2} \ln \left( \frac{1 - \delta_t}{\delta_t} \right) \]

\[ \text{weak weights for next round in terms of weights for this round} \]

\[ \text{end} \]

To calculate \( \mathcal{F} \), 

\[ \mathcal{F}(x) = \sum_{t=0}^{T} \lambda_t \cdot h_t(x) \]