Today:
- Ensemble learning
- Weighted majority algorithm
- Boosting
- AdaBoost and Logistic Regr.

Recommended reading:
- Shapire's tutorial on Boosting: see Piazza syllabus

some slides courtesy of Maria Balcan
some slides courtesy of Rob Shapire
some slides courtesy of Ziv Bar-Joseph

Mistake Bounds

So far: how many examples needed to learn?
What about: how many mistakes before convergence?

Let’s consider similar setting to PAC learning:
- Instances drawn at random from $X$ according to distribution $D$
- Learner must classify each instance before receiving correct classification from teacher
- Can we bound the number of mistakes learner makes before converging?
**Weighted Majority Algorithm**

\( a_i \) denotes the \( i \)th prediction algorithm in the pool \( A \) of algorithms. \( w_i \) denotes the weight associated with \( a_i \).

- For all \( i \) initialize \( w_i \leftarrow 1 \)
- For each training example \( \langle x, c(x) \rangle \)
  * Initialize \( q_0 \) and \( q_1 \) to 0
  * For each prediction algorithm \( a_i \)
    - If \( a_i(x) = 0 \) then \( q_0 \leftarrow q_0 + w_i \)
      If \( a_i(x) = 1 \) then \( q_1 \leftarrow q_1 + w_i \)
    * If \( q_1 > q_0 \) then predict \( c(x) = 1 \)
    * If \( q_0 > q_1 \) then predict \( c(x) = 0 \)
    * If \( q_1 = q_0 \) then predict 0 or 1 at random for \( c(x) \)
  * For each prediction algorithm \( a_i \) in \( A \) do
    - If \( a_i(x) \neq c(x) \) then \( w_i \leftarrow \beta w_i \)

---

**Weighted Majority**

[Relative mistake bound for Weighted-Majority] Let \( D \) be any sequence of training examples, let \( A \) be any set of \( n \) prediction algorithms, and let \( k \) be the minimum number of mistakes made by any algorithm in \( A \) for the training sequence \( D \). Then the number of mistakes over \( D \) made by the Weighted-Majority algorithm using \( \beta = \frac{1}{2} \) is at most

\[
2.4(k + \log_2 n)
\]
**Boosting**

- Weighted Majority algorithm learns weight for each predictor
  - It's an example of an ensemble method: combines predictions of *multiple* hypotheses

- Boosting learns weight, and also the hypotheses

- Leads to one of the most popular learning methods in practice: Decision Forests
Boosting: Key Idea

• Use a learner that produces better-than-chance \( h(x) \)’s
• Train it multiple times, on reweighted training examples
  – Each time, upweight the incorrectly classified examples, downweight the correctly classified examples
• Final prediction: weighted vote of the multiple \( h_i(x) \)’s

• Practically useful
• Theoretically interesting

---

**AdaBoost Algorithm**

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 weak learner using distribution \(D_t\).
• Get weak hypothesis \(h_t : X \rightarrow \{-1, +1\}\) with error
  \[\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].\]
• Choose \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
• Update:
  \[D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\]
  where \(Z_t\) is a normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final hypothesis:
\[H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).\]
AdaBoost: A toy example

Weak classifiers: vertical or horizontal half-planes (a.k.a. decision stumps)

[Rob Shapire]
AdaBoost: A toy example

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

- Train weak learner using distribution \(D_t\).
- Get weak hypothesis \(h_t : X \rightarrow \{-1, +1\}\) with error

\[
\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].
\]

- Choose \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
- Update:

\[
D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} 
    e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\
    e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i 
\end{cases}
= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

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

Output the final hypothesis:

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

Training error of final classifier is bounded by:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i))
\]

Where \( f(x) = \sum_{t} \alpha_t h_t(x); H(x) = \text{sign}(f(x)) \)

Theoretical Result 1: Training Error

Training error of final classifier is bounded by:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t} Z_t
\]

Where \( f(x) = \sum_{t} \alpha_t h_t(x); H(x) = \text{sign}(f(x)) \)

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_i y_i h_t(x_i))
\]
Theoretical Result 1: Training Error

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where

$$f(x) = \sum_t \alpha_t h_t(x); H(x) = sign(f(x))$$

If we minimize $\prod_t Z_t$ we minimize our training error.

We can tighten this bound greedily, by choosing $\alpha_t$ and $h_t$ on each iteration to minimize $Z_t$:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

Theoretical Result 1: Training Error

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

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

Where:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$
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 weak learner using distribution \(D_t\).
- Get weak hypothesis \(h_t : X \rightarrow \{-1, +1\}\) with error
  \[\epsilon_t = \Pr_{x \sim D_t} [h_t(x) \neq y].\]
- Choose \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
- Update:
  \[D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\]

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

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

**Analyzing the training error**

- **Theorem:**
  - write \(\epsilon_t\) as \(1/2 - \gamma_t\)
  - then
    \[
    \text{training error}(H_{\text{final}}) \leq \prod_{t} \left[ 2 \sqrt{\epsilon_t(1 - \epsilon_t)} \right] = \prod_{t} \left[ 1 - 4 \gamma_t^2 \right] \leq \exp \left( -2 \sum_{t} \gamma_t^2 \right)
    \]
  - so: if \(\forall t : \gamma_t \geq \gamma > 0\)
    then \(\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}\)

- **AdaBoost is adaptive:**
  - does not need to know \(\gamma\) or \(T\) a priori
  - can exploit \(\gamma_t \gg \gamma\)

[Rob Shapire]
Bound on True Error

[Freund & Shapire, 1999]

With high probability:

\[
\text{error}_{\text{true}} \left( \text{sign} \left( \sum_t \alpha_t h_t(x) \right) \right) \leq \text{error}_{\text{train}} \left( \text{sign} \left( \sum_t \alpha_t h_t(x) \right) \right) + O \left( \sqrt{\frac{T \cdot V C \text{dim}(H)}{m}} \right)
\]

**Actual Typical Run**

- test error does not increase, even after 1000 rounds
  - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
</tbody>
</table>

[Rob Shapire]
Figure 3: Comparison of C4.5 versus boosting stumps and boosting C4.5 on a set of 27 benchmark problems as reported by Freund and Schapire [21]. Each point in each scatterplot shows the test error rate of the two competing algorithms on a single benchmark. The y-coordinate of each point gives the test error rate (in percent) of C4.5 on the given benchmark, and the x-coordinate gives the error rate of boosting stumps (left plot) or boosting C4.5 (right plot). All error rates have been averaged over multiple runs.
A Better Story: Theory of Margins [with Freund, Bartlett & Lee]

- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- recall: $H_{\text{final}}$ is weighted majority vote of weak classifiers
- measure confidence by margin = strength of the vote
  $= (\text{fraction voting correctly}) - (\text{fraction voting incorrectly})$

Empirical Evidence: The Margin Distribution

- margin distribution
  $= \text{cumulative distribution of margins of training examples}$

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
<tr>
<td>% margins $\leq 0.5$</td>
<td>7.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>minimum margin</td>
<td>0.14</td>
<td>0.52</td>
<td>0.55</td>
</tr>
</tbody>
</table>
Bounds on Generalization Error in Boosting

[Freund & Shapire, 1999]
With high probability

$$
\text{error}_{true}\left(\text{sign}\left(\sum_{t} \alpha_t h_t(x)\right)\right) \leq \text{error}_{\text{train}}\left(\text{sign}\left(\sum_{t} \alpha_t h_t(x)\right)\right) + O\left(\sqrt{\frac{T \cdot \text{VCdim}(H)}{m}}\right)
$$

[Shapire, et al., 1999]
For all $\theta > 0$, with high probability:

$$
\text{error}_{true}\left(\text{sign}\left(\sum_{t} \alpha_t h_t(x)\right)\right) \leq \text{P}_{\text{train}}[\text{margin}_f(x, y) \leq \theta] + O\left(\sqrt{\frac{\text{VCdim}(H)}{m \theta^2}}\right)
$$

Margin based: Independent of $T$ !!

Boosting and Logistic Regression

Logistic regression assumes:

$$
P(Y = 1|X) = \frac{1}{1 + \exp(f(x))}
$$

And tries to maximize data likelihood:

$$
P(D|H) = \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i f(x_i))}
$$

Equivalent to minimizing log loss

$$
\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i)))
$$
Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss
\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]

Boosting minimizes similar loss function!!
\[ \frac{1}{m} \sum_i \exp(-y_if(x_i)) = \prod_t Z_t \]

Both smooth approximations of 0/1 loss!

Logistic regression and Boosting

Logistic regression:
- Minimize loss fn
  \[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]
- Define
  \[ f(x) = \sum_j w_j x_j \]
  where \( x_j \) predefined

Boosting:
- Minimize loss fn
  \[ \sum_{i=1}^{m} \exp(-y_if(x_i)) \]
- Define
  \[ f(x) = \sum_t \alpha_t h_t(x) \]
  where \( h_t(x_i) \) defined dynamically to fit data
  (not a linear classifier)
- Weights \( \alpha_t \) learned incrementally
What You Should Know

• Ensemble methods

• Weighted Majority
  – Learns weights for a given pool of hypotheses
  – Mistake bound relative to best hypothesis in the pool
  – …

• Boosting
  – Learns weights and hypotheses
  – Theory: training error, true error, correspondence to Log. Regression
  – Practice: Boosted decision trees (and stumps) very popular!

• Many variants of ensemble methods
  – Resample training data to generate variety
  – Randomize learning algorithm to generate variety
  – Active learning – choose examples where vote is closest to tie