Boosting
Boosting

- Also an ensemble method: the final prediction is a combination of the prediction of multiple classifiers.

- What is different?
  - It’s iterative.
    - **Boosting**: Successive classifiers depends upon its predecessors - look at **errors from previous classifiers** to decide what to **focus** on for the next iteration over data
    - **Bagging**: Individual training sets and classifiers were independent.
  - All training examples are used in each iteration, but with different weights – more weights on difficult examples. (the ones we made mistakes before)
Adaboost: Illustration

**Final Classifier**

\[ H(x) = \text{sign} \left[ \sum_{m=1}^{M} \alpha_m h_m(x) \right] \]

- Original data: uniformly weighted
- Training Sample
- Weighted Sample
- Update weights
- \( h_1(x) \)
- \( h_2(x) \)
- \( h_3(x) \)
- \( h_M(x) \)
AdaBoost (High level steps)

- AdaBoost performs $M$ boosting rounds, creating one ensemble member in each round.

Operations in $l$’th boosting round:

1. Call Learn on data set $S$ and weights $D_l$ to produce $l$’th classifier $h_l$. Where $D_l$ is the weights of round $l$.

2. Compute the $(l+1)$’th round weights $D_{l+1}$ by putting more weight on instances that $h_l$ makes errors on.

3. Compute a voting weight $\alpha_l$ for $h_l$.

The ensemble hypothesis returned is:

$$H(x) = \text{sign} \left[ \sum_{m=1}^{M} \alpha_m \cdot h_m(x) \right]$$
Weighted Training Sets

- AdaBoost works by invoking the base learner many times on different weighted versions of the same training data set.
- Thus we assume our base learner takes as input a weighted training set, rather than just a set of training examples.

Learn:

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$ - Set of $N$ labeled training instances.</td>
</tr>
<tr>
<td>$D$ - A set of weights over $S$, where $D(i)$ is the weight of the $i^{th}$ training instance (interpreted as the probability of observing $i^{th}$ instance), and $\sum_{i=1}^{N} D(i) = 1$. $D$ is also called distribution of $S$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$ - hypothesis from hypothesis space $H$ with low weighted error</td>
</tr>
</tbody>
</table>
Definition: Weighted Error

• Denote the \(i^{th}\) training instance by \(<x_i, y_i>\).

• For a training set \(S\) and distribution \(D\) the weighed training error is the sum of the weights of incorrect examples

\[
error(h, S, D) = \sum_{i=1}^{N} D(i) \cdot [h(x_i) \neq y_i]
\]

• The goal of the base learner is to find a hypothesis with a ‘small’ weighted error.

• Thus \(D(i)\) can be viewed as indicating to Learn the importance of learning the \(i^{th}\) training instance.
Weighted Error

• Adaboost calls \textit{Learn} with a set of prespecified weights
• It is often straightforward to convert a base learner \textit{Learn} to take into account the weights in $D$.

Decision trees?

K Nearest Neighbor?

Naïve Bayes?

• When it is not straightforward we can resample the training data $S$ according to $D$ and then feed the new data set into the learner.
AdaBoost algorithm:

**Input:** $S$ - Set of $N$ labeled training instances.

**Output:** $H(x) = \text{sign}\left[\sum_{t=1}^{M} \alpha_m \cdot h_m(x)\right]$ 

Initialize $D_1(i) = \frac{1}{N}$, for all $i$ from 1 to $N$. (uniform distribution)

FOR $t = 1, 2, \ldots, M$ DO

$h_t = \text{Learn}(S, D_t)$

$\epsilon_t = \text{error}(h_t, S, D_t)$

$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ ;; if $\epsilon_t < 0.5$ implies $\alpha_t > 0$

$$D_{t+1}(i) = D_t(i) \times \begin{cases} e^{\alpha_t}, & h_t(x_i) \neq y_i \\ e^{-\alpha_t}, & h_t(x_i) = y_i \end{cases}$$ for $i$ from 1 to $N$

Normalize $D_{t+1}$ ;; can show that $h_t$ has 0.5 error on $D_{t+1}$

Note that $\epsilon_t < 0.5$ implies $\alpha_t > 0$ so weight is decreased for instances $h_t$ predicts correctly and increases for incorrect instances
AdaBoost using Decision Stump (Depth-1 decision tree)

Original Training set: Equal Weights to all training samples

Taken from “A Tutorial on Boosting” by Yoav Freund and Rob Schapire
AdaBoost (Example)

ROUND 1

What are the weight of these points?

$\epsilon_1 = 0.30$

$\alpha_1 = 0.42$
AdaBoost (Example)

ROUND 2

What are the weight of the points?

$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

$\frac{7}{66}$

$\frac{1}{22}$

$\frac{1}{6}$
AdaBoost(Example)

ROUND 3

\( h_3 \)

\( \varepsilon_3 = 0.14 \)
\( \alpha_3 = 0.92 \)
AdaBoost(Example)

\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]
Boosting Decision Stumps

Decision stumps: very simple rules of thumb that test condition on a single attribute.

Among the most commonly used base classifiers – truly weak!

Boosting with decision stumps has been shown to achieve better performance compared to unbounded decision trees.
Boosting Performance

• Comparing C4.5, boosting decision stumps, boosting C4.5 using 27 UCI data set
  – C4.5 is a popular decision tree learner
Boosting vs Bagging of Decision Trees
Overfitting?

• Boosting drives training error to zero, will it overfit?
• Curious phenomenon

![Graph showing training and test error over # of rounds (T)]

• Boosting is often robust to overfitting (not always)
• Test error continues to decrease even after training error goes to zero
Explanation with Margins

\[ f(x) = \sum_{l=1}^{L} w_l \cdot h_l(x) \]

Margin = \( y \cdot f(x) \)

Histogram of functional margin for ensemble just after achieving zero training error
Effect of Boosting: Maximizing Margin

Even after zero training error the margin of examples increases. This is one reason that the generalization error may continue decreasing.
Bias/variance analysis of Boosting

- In the early iterations, boosting is primarily a bias-reducing method.
- In later iterations, it appears to be primarily a variance-reducing method.
What you need to know about supervised ensemble methods?

• Bagging: a randomized algorithm based on bootstrapping
  – What is bootstrapping
  – Variance reduction
  – What learning algorithms will be good for bagging? - high variance, low bias ones such as un-pruned decision trees

• Boosting:
  – Combine weak classifiers (i.e., slightly better than random)
  – Training using the same data set but different weights
  – How to update weights?
  – How to incorporate weights in learning (DT, KNN, Naïve Bayes)
  – One explanation for not overfitting: maximizing the margin

• Other ensemble methods abound, often lead to improved performance
  – Random forest
  – Combining different classifiers
Unsupervised Ensembles

• Ensemble can also help clustering
• Ensemble can help
  – Identify the number of clusters in the data
  – Learn a more robust clustering solution
  – Integrate outputs obtained from different data sources
• I will present some very basic ideas
Ensemble methods for identifying K

- In kmeans clustering, when using incorrect K, different random runs (with different initialization, different subsets or bootstrapped sample of the data) may lead to highly variant solutions
- When using correct K, one expect greater stability
- Choose K to maximize the stability (aka the similarity between different clusterings)
Ensemble for improving clustering

• Given a data set, different clustering algorithms or different runs of the same algorithm may give different solutions
• Combining the different solutions can improve the clustering quality
• For example:
  – For every pair of instances we can measure how many times they were clustered together as a new measure of similarity
  – Perform clustering on this new similarity measure leads to more robust final solution
Ensemble for information fusion

• Data comes from different sources or modalities, each capturing different aspects of the object
  – E.g., a website can be described by the words present on the website, the images present on the site, the links present on the site etc

• They are not directly comparable
  – Different dimensions, different range of values

• Apply clustering separately for each source/modality

• Combine their clustering results to get an integrated understanding of the data