Learning Ensembles
Outlines

- Learning Assembles
- Random Forest
- Adaboost
Training data: Restaurant example

- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>Bar</td>
<td>Fri</td>
</tr>
<tr>
<td>X1</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X2</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X3</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X4</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X5</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X6</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X7</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X8</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X9</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X10</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>X11</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X12</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- Classification of examples is positive (T) or negative (F)
A decision tree to decide whether to wait

- imagine someone talking a sequence of decisions.
Learning Ensembles

• Learn multiple classifiers separately
• Combine decisions (e.g. using weighted voting)
• When combing multiple decisions, random errors cancel each other out, correct decisions are reinforced.
Homogenous Ensembles

• Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m

• Methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data
  - DECORATE: Add additional artificial training data
Bagging

• Create ensembles by repeatedly randomly resampling the training data (Brieman, 1996).
• Given a training set of size $n$, create $m$ sample sets
  ▪ Each *bootstrap sample set* will on average contain 63.2% of the unique training examples, the rest are replicates.
• Combine the $m$ resulting models using majority vote.

• Decreases error by decreasing the variance in the results due to *unstable learners*, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.
Random Forests

- Introduce two sources of randomness: “Bagging” and “Random input vectors”
  - Each tree is grown using a bootstrap sample of training data
  - At each node, best split is chosen from random sample of \( m \) variables instead of all variables \( M \).
- \( m \) is held constant during the forest growing
- Each tree is grown to the largest extent possible
- Bagging using decision trees is a special case of random forests when \( m=M \)
Random Forests

Figure 5.40. Random forests.
Random Forest Algorithm

- Good accuracy without over-fitting
- Fast algorithm (can be faster than growing/pruning a single tree); easily parallelized
- Handle high dimensional data without much problem

- Simple with theoretical foundation
Adaboost - Adaptive Boosting

- **Use 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 sum of weak classifiers
AdaBoost: An Easy Flow

Original training set

Data set $1$ $\rightarrow$ Data set $2$ $\rightarrow$ \ldots $\rightarrow$ Data set $T$

Learner$_1$ $\rightarrow$ Learner$_2$ $\rightarrow$ \ldots $\rightarrow$ Learner$_T$

weighted combination

training instances that are wrongly predicted by Learner$_1$ will be weighted more for Learner$_2$
Adaboost Terminology

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

- Weak Classifier: < 50% error over any distribution
- Strong Classifier: thresholded linear combination of weak classifier outputs
And in a Picture

training case

correctly classified

training case has large weight in this round

this DT has a strong vote
AdaBoost.M1

• Given **training set** $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
• $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
• Initialize **distribution** $D_1(i) = 1/m$; (weight of training cases)
• for $t = 1, \ldots, T$:
  • Find a **weak classifier** (“rule of thumb”) $h_t : X \rightarrow \{-1, +1\}$ with small error $\varepsilon_t$ on $D_t$:
  • Update distribution $D_t$ on $\{1, \ldots, m\}$. $\alpha_t = \log(1/\varepsilon_t - 1)$

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

• **output final hypothesis** where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).

$$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$$
Reweighting

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha ty_i h_t(x_i))}{Z_t} \]

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

⇒ Increase (decrease) weight of wrongly (correctly) classified examples
Toy Example

$D_1$
Round 1

Weak classifier: if $h_1 < 0.2 \rightarrow 1$ else -1

$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$
Round 2

Weak classifier: if $h_2 < 0.8 \Rightarrow 1$ else -1
Round 3

Weak classifier: if $h_3 > 0.7 \rightarrow 1$ else $-1$
Final Combination

if \( h_1 < 0.2 \rightarrow 1 \) else -1

if \( h_2 < 0.8 \rightarrow 1 \) else -1

\[
H_{\text{final}} = \text{sign} \left( 0.42 \begin{pmatrix} +0.65 \end{pmatrix} + 0.92 \right)
\]

if \( h_3 > 0.7 \rightarrow 1 \) else -1
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, etc – *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”
Suppose \( f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \). Therefore, training error is:

\[
[H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)).
\]

Thus, \( \exp(-y_i f(x_i)) \geq 1 \).

Finally:

\[
\frac{1}{m} \sum_{i} D_{T+1}(i) = 1, \quad \frac{1}{m} \sum_{i} \exp\left(-\sum_{t} \alpha_t y_i i_t \right) = \prod_{t} Z_t
\]
AdaBoost: How to choose $\alpha_t$ 

- According to 
\[
\frac{1}{m} \left| \{i : H(x_i) \neq y_i \} \right| \leq \prod_{t=1}^{T} z_t
\]

This equation is obvious if we treat $u_i$ as a binary-valued variable.

- Let $u_i = y_i h_t(x_i)$. 

- The right term is minimized when 
\[
\alpha = \frac{1}{2} \ln \left( \frac{1+r}{1-r} \right)
\]

Minimize the error bound could be done by greedily minimizing $\alpha^*$ 

Actually AdaBoost can just minimize the training error. 

By setting 
\[
Z = \sum_i D(i)e^{-\alpha u_i}
\]

and considering 
\[
\sum D(i) = 1
\]

we can easily get this solution.

Then the training error of $H$ is at most 
\[
\prod_{t=1}^{T} \sqrt{1-r_t^2}
\]