Ensemble Learning: An Introduction

Adapted from Slides by Tan, Steinbach, Kumar
General Idea

Original Training data

\[ D \]

\[ D_1 \quad D_2 \quad \cdots \quad D_{t-1} \quad \cdots \quad D_t \]

\[ C_1 \quad C_2 \quad \cdots \quad C_{t-1} \quad \cdots \quad C_t \]

Step 1: Create Multiple Data Sets

Step 2: Build Multiple Classifiers

Step 3: Combine Classifiers
Why does it work?

• Suppose there are 25 base classifiers
  – Each classifier has error rate, $\varepsilon = 0.35$
  – Assume classifiers are independent
  – Probability that the ensemble classifier makes a wrong prediction:
    $$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1 - \varepsilon)^{25-i} = 0.06$$
Examples of Ensemble Methods

• How to generate an ensemble of classifiers?
  – Bagging
  – Boosting
Bagging

- Sampling with replacement

- Build classifier on each bootstrap sample
- Each sample has probability \((1 - 1/n)^n\) of being selected as test data
- Training data = \(1 - (1 - 1/n)^n\) of the original data

<table>
<thead>
<tr>
<th>Data ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Bagging (Round 1)</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Bagging (Round 2)</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Bagging (Round 3)</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
The 0.632 bootstrap

• This method is also called the 0.632 bootstrap
  – A particular training data has a probability of 1-1/n of not being picked
  – Thus its probability of ending up in the test data (not selected) is:

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

  – This means the training data will contain approximately 63.2% of the instances
Example of Bagging

Assume that the training data is:

0.3 to 0.7: -1
0.3

Goal: find a collection of 10 simple thresholding classifiers that collectively can classify correctly.
-Each simple (or weak) classifier is:
  \[(x \leq K \rightarrow \text{class} = +1 \text{ or } -1 \text{ depending on which value yields the lowest error; where } K \text{ is determined by entropy minimization)}\]
<table>
<thead>
<tr>
<th>Bagging Round 1:</th>
<th>Bagging Round 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bagging Round 3:</th>
<th>Bagging Round 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bagging Round 5:</th>
<th>Bagging Round 6:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bagging Round 7:</th>
<th>Bagging Round 8:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bagging Round 9:</th>
<th>Bagging Round 10:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.35. Example of bagging.
Bagging  (applied to training data)

<table>
<thead>
<tr>
<th>Round</th>
<th>$x=0.1$</th>
<th>$x=0.2$</th>
<th>$x=0.3$</th>
<th>$x=0.4$</th>
<th>$x=0.5$</th>
<th>$x=0.6$</th>
<th>$x=0.7$</th>
<th>$x=0.8$</th>
<th>$x=0.9$</th>
<th>$x=1.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-6</td>
<td>-6</td>
<td>-6</td>
<td>-6</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Sign</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>True Class</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.36.  Example of combining classifiers constructed using the bagging approach.

Accuracy of ensemble classifier: 100% 😊
Bagging- Summary

• Works well if the base classifiers are unstable (complement each other)
• Increased accuracy because it reduces the variance of the individual classifier
• Does not focus on any particular instance of the training data
  – Therefore, less susceptible to model over-fitting when applied to noisy data
• What if we want to focus on a particular instances of training data?
In general,
- **Bias** is contributed to by the training error; a complex model has low bias.
- **Variance** is caused by future error; a complex model has High variance.
- Bagging reduces the variance in the base classifiers.

**Figure 5.32.** Bias-variance decomposition.
Figure 5.33. Two decision trees with different complexities induced from the same training data.
(a) Decision boundary for decision tree.  

(b) Decision boundary for 1-nearest neighbor.

**Figure 5.34.** Bias of decision tree and 1-nearest neighbor classifiers.
Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of a boosting round
Boosting

• Records that are wrongly classified will have their weights increased
• Records that are classified correctly will have their weights decreased

```
Original Data  1  2  3  4  5  6  7  8  9  10
Boosting (Round 1) 7  3  2  8  7  9  4  10  6  3
Boosting (Round 2) 5  4  9  4  2  5  1  7  4  2
Boosting (Round 3) 4  4  8  10 4  5  4  6  3  4
```

• Example 4 is hard to classify
  • Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds
Boosting

• Equal weights are assigned to each training instance \((1/d)\) for round 1) at first
• After a classifier \(C_i\) is learned, the weights are adjusted to allow the subsequent classifier \(C_{i+1}\) to “pay more attention” to data that were misclassified by \(C_i\).
• Final boosted classifier \(C^*\) combines the votes of each individual classifier
  – Weight of each classifier’s vote is a function of its accuracy
• Adaboost – popular boosting algorithm
Adaboost (Adaptive Boost)

• Input:
  – Training set D containing N instances
  – T rounds
  – A classification learning scheme

• Output:
  – A composite model
Adaboost: Training Phase

• Training data D contain N labeled data \((X_1, y_1), (X_2, y_2), (X_3, y_3), \ldots (X_N, y_N)\)
• Initially assign equal weight \(1/N\) to each data
• To generate \(T\) base classifiers, we need \(T\) rounds or iterations
• Round \(i\), data from D are sampled with replacement, to form \(D_i\) (size \(n\))
• Each data’s chance of being selected in the next rounds depends on its weight
  – Each time the new sample is generated directly from the training data D with different sampling probability according to the weights; these weights are not zero
Adaboost: Training Phase

• Base classifier $C_i$, is derived from training data of $D_i$
• Error of $C_i$ is tested using $D$
• Weights of training data are adjusted depending on how they were classified
  – Correctly classified: Decrease weight
  – Incorrectly classified: Increase weight
• Weight of a data indicates how hard it is to classify it (directly proportional)
Adaboost: Testing Phase

- The lower a classifier error rate, the more accurate it is, and therefore, the higher its weight for voting should be.
- Weight of a classifier $C_i$’s vote is
  $$\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$
- Testing:
  - For each class $c$, sum the weights of each classifier that assigned class $c$ to $X$ (unseen data)
  - The class with the highest sum is the WINNER!

$$C^*(x_{test}) = \arg \max_y \sum_{i=1}^{T} \alpha_i \delta(C_i(x_{test}) = y)$$
Example: AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_T$

- Error rate: $(i =$ index of classifier, $j =$ index of instance $)$

  $\varepsilon_i = \frac{1}{N} \sum_{j=1}^{N} w_j \delta(C_i(x_j) \neq y_j)$

- Importance of a classifier:

  $\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_i}{\varepsilon_i} \right)$
Example: AdaBoost

- Assume: $N$ training data in $D$, $T$ rounds, $(x_j, y_j)$ are the training data, $C_i$, $a_i$ are the classifier and weight of the $i^{th}$ round, respectively.
- Weight update on all training data in $D$:

\[
w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \left\{ \begin{array}{ll}
\exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\
\exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j
\end{array} \right.
\]

where $Z_i$ is the normalization factor

\[
C^*(x_{test}) = \arg \max_y \sum_{i=1}^{T} \alpha_i \delta(C_i(x_{test}) = y)
\]
Illustrating AdaBoost

Original Data

0.1

0.1

0.1

Data points for training

Boosting Round 1

0.0094

0.0094

0.4623

B1

\( \alpha = 1.9459 \)
Illustrating AdaBoost

Boosting Round 1

Boosting Round 2

Boosting Round 3

Overall

\[ \alpha = 1.9459 \]

\[ \alpha = 2.9323 \]

\[ \alpha = 3.8744 \]
Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Random Forests grows many trees
  - Ensemble of unpruned decision trees
  - Each base classifier classifies a “new” vector of attributes from the original data
  - Final result on classifying a new instance: voting. Forest chooses the classification result having the most votes (over all the trees in the forest)
Random Forests

• Introduce two sources of randomness: “Bagging” and “Random input vectors”
  – Bagging method: each tree is grown using a bootstrap sample of training data
  – Random vector method: At each node, best split is chosen from a random sample of $m$ attributes instead of all attributes
Random Forests

Figure 5.40. Random forests.
Methods for Growing the Trees

- Fix a $m \leq M$. At each node
  - Method 1:
    - Choose $m$ attributes randomly, compute their information gains, and choose the attribute with the largest gain to split
  - Method 2:
    - (When $M$ is not very large): select $L$ of the attributes randomly. Compute a linear combination of the $L$ attributes using weights generated from $[-1,+1]$ randomly. That is, new $A = \sum (W_i \times A_i)$, $i=1..L$.
  - Method 3:
    - Compute the information gain of all $M$ attributes. Select the top $m$ attributes by information gain. Randomly select one of the $m$ attributes as the splitting node.
Random Forest Algorithm: method 1 in previous slide

• M input features in training data, a number \( m << M \) is specified such that at each node, \( m \) features are selected at random out of the \( M \) and the best split on these \( m \) features is used to split the node. (In weather data, \( M=4 \), and \( m \) is between 1 and 4)

• \( m \) is held constant during the forest growing

• Each tree is grown to the largest extent possible (deep tree, overfit easily), and there is no pruning
Generalization Error of Random Forests (page 291 of Tan book)

- It can be proven that the generalization Error $\leq \frac{\rho(1-s^2)}{s^2}$,
  - $\rho$ is the average correlation among the trees
  - $s$ is the strength of the tree classifiers
    - Strength is defined as *how certain* the classification results are on the training data on average
    - How certain is measured $\Pr(C_1|X) - \Pr(C_2|X)$, where $C_1, C_2$ are class values of two highest probability in decreasing order for input instance $X$.
- Thus, higher diversity and accuracy is good for performance