Ensemble Methods

- Use a number of base classifiers, and make a prediction based on the predictions of the base classifiers.

Ensemble Classifier Example

- Binary classification
- 3 classifiers, each with 30% error rate
- Classify by majority vote
- Error rate of the ensemble classifier??

Build An Ensemble Classifier

- Approach 1: use several different classification methods, and train each with a different training set
- Approach 2: use the one classification method and one training set

Get K Classifiers Out Of One ...

- By manipulating the training set
  - Use a different subset of the training set to train each classifier
  - E.g. Bagging and Boosting
- By manipulating the input features
  - Use a different subset of the attributes to train each classifier
  - E.g. Random Forest

...Get K Classifiers Out Of One

- By manipulating the class labels
  - E.g. ECOC.
- By manipulating the learning algorithm
  - E.g. use of different kernel functions, introducing randomness in attribute selection in decision tree induction
Manipulate the Training Set

- How can we use one training set to train k classifiers?
  - Use the same training set for each classifier??
  - Evenly divide the training set into k subsets??

Bagging

- Use a bootstrap sample for each classifier
- A bootstrap sample \( D_1 \)
  - Obtained by uniformly samples the training set \( D \) with replacement \( |D| \) times
  - Contains roughly 63.2% of the original records
    - \( 1 - (1 - 1/N)^N \to 1 - 1/e = 0.632 \)

Bagging Example – Dataset

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>0.4</td>
<td>-1</td>
</tr>
<tr>
<td>0.5</td>
<td>-1</td>
</tr>
<tr>
<td>0.6</td>
<td>-1</td>
</tr>
<tr>
<td>0.7</td>
<td>-1</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>1.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Bagging Example – Classifier

- Base classifier: decision tree with one level \( x \leq k \)
  - Best training accuracy possible??
- Ensemble classifier: 10 classifiers, majority vote

<table>
<thead>
<tr>
<th>Bound</th>
<th>( x \leq 1 )</th>
<th>( x \leq 2 )</th>
<th>( x \leq 3 )</th>
<th>( x \leq 4 )</th>
<th>( x \leq 5 )</th>
<th>( x \leq 6 )</th>
<th>( x \leq 7 )</th>
<th>( x \leq 8 )</th>
<th>( x \leq 9 )</th>
<th>( x \leq 10 )</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>
</tbody>
</table>

Classification

- Base classifier: decision tree with one level \( x \leq k \)
  - Best training accuracy possible??
- Ensemble classifier: 10 classifiers, majority vote

Bagging Example – Bagging

Figure 3.34. Example of use during classification with abstaining for bagging approach.
About Bagging
- Reduces the errors associated with random fluctuations in the training data for unstable classifiers, e.g. decision trees, rule-based classifiers, ANN
- May degrade the performance of stable classifiers, e.g. Bayesian network, SVM, k-NN

Intuition for Boosting
- Sample with weights
  - hard-to-classify records should be chosen more often
- Combine the prediction of the base classifiers with weights
  - Classifiers with lower error rates get more voting power

Boosting – Training
- Initialize the weight of each record
- For i=1 to k
  - Sample with replacement according to the weights
  - Train a classifier \( M_i \)
  - Calculate \( \text{error}(M_i) \), assign a weight to \( M_i \) based on \( \text{error}(M_i) \)
  - Update the weights of the records
    - Increase the weights of the misclassified records
    - Decrease the weights of the correctly classified records

Boosting – Classification
- For each class, sum up the weights of the classifiers that vote for that class
- The class that gets the highest sum is the predicted class

Boosting Implementation
- How the record weights are decided
- How the classifier weights are decided

Adaboost
- Error rate of classifier \( M_i \):
  \[
  \text{error}(M_i) = \sum w_i \times \text{err}(X_i)
  \]
- Update the weights of the correctly classified records:
  \[
  w_j \times \frac{\text{error}(M_i)}{\text{error}(M_j)}
  \]
- Weight of classifier \( M_i \):
  \[
  \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}
  \]
- Initial \( w_i = 1/|D| \)
- Classifiers with \( \text{error}(M_i) > 0.5 \) are dropped
- Normalize the weights of all records after updating the weights of the correctly classified records
Adaboost Example

- 5 records
- $M_1$ classification results:

<table>
<thead>
<tr>
<th>Record</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly classified</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Weight of $M_1$?? Updated record weights??

Random Forest

- A number of decision tree classifiers created from the same training set

Create a Random Forest

- Forest-RJ: randomly select $F$ attributes out of $d$ input attributes usually $F=\log_2 d + 1$
- Forest-RC: at each node, create $F$ new attributes, each is a random linear combination of $L$ randomly selected attributes
- At each node, randomly select one split out of the top $F$ best splits

Some Empirical Comparison of Ensemble Methods

- See Table 5.5 in Introduction to Data Mining by Tan, Steinbach, and Kumar

Readings

- Textbook 6.14