Data Mining: Advanced Concepts
Andrew Kusiak
Intelligent Systems Laboratory
2139 Seamans Center
The University of Iowa
Iowa City, Iowa 52242 - 1527
Tel: 319 - 335 5934 Fax: 319-335 5669
andrew-kusiak@uiowa.edu
http://www.icaen.uiowa.edu/~ankusiak

Increasing Classification Accuracy

- Bagging
- Boosting
- Meta-learning (stacking)

Bagging Procedure

Classifier generation
Step 1. Create $t$ data sets from a database applying the sampling with replacement scheme.
Step 2. Apply a learning algorithm to each sample training data set.

Classification
Step 3. For an object with unknown decision, make predictions with each of the $t$ classifiers.
Step 4. Select the most frequently predicted decision.

Boosting Procedure (1)

Classifier generation
Step 0. Set the weight value, $w = 1$, and assign it to each object in the training data set.
For each of $t$ iterations, perform:
Step 1. Apply a learning algorithm to the weighted training data set.
Step 2. Compute classification error $e$ for the weighted training data set. If $e = 0$ or $e >= .5$, then terminate the classifier generation process and go to Step 4, otherwise multiple the weight $w$ of each object by $e/(1 - e)$ and normalize the weights of all objects.

Classification
Step 4. Assign weight $q = 0$ to each decision (class) to be predicted.
Step 5. For each of $t$ (or less) classifiers, add $-\log e/(1 - e)$ to the weight of the decision predicted the classifier and output the decision with the highest weight.

Boosting Procedure (2)

For $e = 0$ all training examples (objects) are correctly classified (a perfect classifier) and therefore there is no reason to modify the object weights, i.e., for $e/(1 - e) = 0$ all new weights $w$ become 0.

For $e = .5$, the expression $-\log e/(1 - e) = 0$, and therefore the weights $q = 0$ are not be modified and therefore no decision is generated due to high classification error $e$. 
Meta-learning

Training data
Learning algorithm 1 Classifier 1

Test data
Predicted decisions

Training data
Learning algorithm 2 Classifier 2

Predicted decisions

Meta-classifier
Meta-learning algorithm
Meta-training data

Creating Meta-training Data

- Voting
  Each classifier gets one vote and the majority wins.

- Weighted voting
  Provides preferential treatment to some voting classifiers.

- Arbitration
  An arbitrator makes a selection, if the classifiers cannot reach a consensus.

- Combining
  Decisions produced by different classifiers are combined as one decision.

Example

Training data set

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Features</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vector 1</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Vector 2</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Vector 3</td>
<td>High</td>
</tr>
</tbody>
</table>

Predictions of classifiers 1 and 2 for the training data set

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Classifier 1 Prediction</th>
<th>Classifier 2 Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Example

Training set generated by the class-combiner scheme

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Classifier 1 Prediction</th>
<th>Classifier 2 Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Example

Training set generated by the class-attributecombiner scheme

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Features</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High, High, Vector 1</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>High, Low, Vector 2</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Low, Low, Vector 3</td>
<td>High</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Feature Vector</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes, No, Yes, No</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Yes, Yes, No, Yes</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>No, Yes, No, Yes</td>
<td>High</td>
</tr>
</tbody>
</table>

Training set generated by the binary class-attribute-combiner scheme

Example

<table>
<thead>
<tr>
<th>Object No.</th>
<th>Classifier 1 Prediction</th>
<th>Feature = High</th>
<th>Feature = Low</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
<td>High</td>
</tr>
</tbody>
</table>

Binary form of the predictions produced by classifier 1

Meta-learners

- Integration of knowledge learned from different and distributed databases.
- Elimination of inductive bias.
- Extraction of high level models.
- Scalability to hierarchical meta-learning.

Distributed Learning

- Distributed by partitioning
- Distributed by nature

Data Populations

- Homogeneous, $\Theta_i = \Theta_j$
- Heterogeneous, $\Theta_i \neq \Theta_j$

Learning from homogeneously distributed data sets
Learning from heterogeneously distributed data sets

Gini Index (1)

- $S$ = data set with $n$ objects
- $c$ = number of classes in $S$
- $j$ = relative frequency of class $j$ in $S$

$$gini(S) = 1 - \sum_{j=1}^{c} p_j^2$$

Gini Index (2)

- $S_1$ = partition 1 of $S$
- $n_1$ = number of objects in $S_1$
- $S_2$ = partition 2 of $S$
- $n_2$ = number of objects in $S_2$, where $n_2 = (n - n_1)$
- $a$ = splitting criterion

$$gini(S, a) = \frac{n_1}{n} gini(S_1) + \frac{n_2}{n} gini(S_2)$$