Classification

Exercise 3
Classification

"tree"  "tree"  "tree"

"not a tree"  "not a tree"  "not a tree"
The Classification Workflow

Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Unseen Records

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>

Learning algorithm

Induction

Learn Model

Apply Model

Deduction

Model
K-Nearest-Neighbour

- Calculate the distance to all other points
- Choose the nearest K neighbours
- Let them vote for a class

- Requires
  - All known records
  - Distance metric

- Often very accurate
- But also slow
Operators: Set Role

- **Input Port**
  - Example Set

- **Output Ports**
  - Changed Example Set
  - Original Example Set

- **Parameters**
  - Attribute Name
  - Target Role

- Classification Operators need an attribute of type ‘label’
Operators: K-NN

- **Input Port:**
  - Training data (Example Set)

- **Output Ports**
  - Classification Model
  - Training data (Example Set)

- **Parameters**
  - K
  - Weighted Vote
  - Similarity Measure
Operators: Apply Model

- Input Ports
  - Model
  - Unlabelled data (Example Set)
- Output Ports
  - Labelled data (Example Set)
  - Model

- Classification Operators do not apply the model they learn!

- You have to apply the model to *a different* example set
Process: Classification

- Learn the model from the training data
- Apply the model to the testing data
- Check the results
Decision Tree Classifiers

Training Data

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Model: Decision Tree

Terminal node = decision

Splitting Attributes

Refund
- Yes
  - NO
- No
  - MarSt
    - Single, Divorced
    - TaxInc
      - < 80K
        - NO
      - > 80K
        - YES
    - Married
      - NO
**Attribute Test Conditions**

- Depend on attribute types
  - Nominal values: Each partition contains a set of values

![Diagram of Size attributes](Diagram of Size attributes)

- Continuous values: Each partition contains a range of values

![Diagram of Taxable Income](Diagram of Taxable Income)
How to determine the best split?

• Measure the node **impurity** of the split
  • Gini Index
  • Information Gain
  • Gain Ratio
  • Classification Error

• The higher the measure, the less pure is the node
  • We want nodes to be as pure as possible

\[
\text{GINI}(t) = 1 - \sum \left[ p(i|t) \right]^2
\]

\[
\text{GINI}_{\text{split}} = \sum \frac{n_i}{n} \text{GINI}(i)
\]

\[
\text{GainRate}_{\text{split}} = \text{Entropy}(p) - \left( \sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i) \right)
\]

\[
\text{SplitINFO} = \sum \frac{n_i}{n} \text{GINI}(i) - \frac{n}{n} \text{GINI}(t) - 1 - \max_{i} I(t|t)
\]
Gini Index

• Gini Index for a given node \( t \)
  • \( 0 \) – all records belong to the same class
  • Max. (depends on number of classes) – records are equally distributed among classes

• Gini Index of a given split
  • The Gini index of each node in the split is weighted according to it’s size

\[
GINI(t) = 1 - \sum_j [p(j | t)]^2
\]

<table>
<thead>
<tr>
<th>C1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>1</td>
</tr>
<tr>
<td>Gini=0.278</td>
<td></td>
</tr>
</tbody>
</table>

\[
\frac{6}{12} \cdot 0.278 + \frac{6}{12} \cdot 0.278 = 0.278
\]

This is the better split!

<table>
<thead>
<tr>
<th>C1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>5</td>
</tr>
<tr>
<td>Gini=0.278</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>2</td>
</tr>
<tr>
<td>Gini=0.408</td>
<td></td>
</tr>
</tbody>
</table>

\[
\frac{7}{12} \cdot 0.408 + \frac{5}{12} \cdot 0.320 = 0.371
\]

\[
GINI_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)
\]
Operators: ID3

- **Input Port**
  - Training data (Example Set)

- **Output Ports**
  - Classification Model
  - Training data (Example Set)

- **Parameters**
  - Split criterion (measure)
  - Minimal size for split (examples)
  - Minimal leaf size (examples)
  - Minimal gain (for split)

- Can only handle nominal attributes!
Operators: Decision Tree

- **Input Port**
  - Training data (Example Set)

- **Output Ports**
  - Classification Model
  - Training data (Example Set)

- **Parameters**
  - Split criterion (measure)
  - Maximal depth (-1 = unlimited)
  - Minimal size for split (examples)
  - Minimal leaf size (examples)
  - Minimal gain (for split)
  - Pruning

![Decision Tree Parameters](image)
Pre-processing for Classification

• After learning a classifier our results can be bad

• What can we do?
  • Change parameters
  • Change pre-processing

• We add some of our knowledge to the dataset by pre-processing
  • Change the range of values (normalisation)
  • Transform value types
  • Manipulate how attributes are split for decision trees (discretisation)
Discretization Techniques

• Equally sized number of examples per bin
  • Discretize by Size: Specify the size of the bins
  • Discretize by Frequency: Specify the number of bins
Discretization Techniques

• Equally sized data range per bin
  • **Discretize by Binning**: Specify the number of bins

<table>
<thead>
<tr>
<th>Bin</th>
<th>1-10</th>
<th>11-20</th>
<th>21-30</th>
<th>31-40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>1-10</td>
<td>11-20</td>
<td>21-30</td>
<td>31-40</td>
</tr>
<tr>
<td>Width</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

4 bins
Discretization Techniques

- Varying data range per bin
  - Discretize by User Specification: Specify the range per bin
  - Discretize by Entropy: Don’t Specify anything, minimise entropy

1-2
2-13
14-180
181-200
Operators: Discretize

• Input Port
  • Example Set

• Output Ports
  • Changed Example Set
  • Original Example Set
  • Pre-processing Model

• Transforms numerical attributes into nominal attributes
Evaluation

• How do we know the model actually works?
  • By counting the number of errors
    • On a *different* dataset

• What’s the purpose of a model?
  • To apply it to new data where we don’t know the label

• What happened if we used the same dataset?
  • How many errors for a K-NN classifier with K=1?
  • How good would that model be on a different dataset?

Important!!!
Evaluation: Confusion Matrix

- For every class in our dataset, the classifier can produce one of four possible results:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td>Class=No</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

- **True Positive (TP):** A case where the classifier correctly predicts the positive class.
- **False Negative (FN):** A case where the classifier incorrectly predicts the negative class when the actual class is positive.
- **False Positive (FP):** A case where the classifier incorrectly predicts the positive class when the actual class is negative.
- **True Negative (TN):** A case where the classifier correctly predicts the negative class.
Evaluation Measures: Accuracy

• A single measure that tells you the overall accuracy of the result

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

• “Number of correctly classified examples divided by the total number of examples.”

• Problem: Unbalanced data
  • If 99% belong to class “yes”
  • And classifier always says “yes” – 99% Accuracy
Evaluation Measures: Precision and Recall

- Measure two aspects of the result for every class

- Precision: How many of the examples that were labelled “yes” are really “yes”?
  - “the number of correctly labelled examples divided by the number of all examples that were labelled with this class”

- Recall: How many of the examples that are really “yes” were labelled “yes”?
  - “the number of correctly labelled examples divided by the number of all examples that actually belong to this class”

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}
\]
Evaluation Measures: Precision and Recall

• An example:

<table>
<thead>
<tr>
<th>ID</th>
<th>Prediction</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ \text{Precision} = \frac{TP}{TP + FP} \]
\[ \text{Recall} = \frac{TP}{TP + FN} \]

• For class “yes”
  • 1 true positive (ID 1)
  • 2 false positives (ID 2 & 4)
  • 1 true negative (ID 3)

\[ \text{Precision}_{yes} = \frac{1}{1 + 2} = \frac{1}{3} \]
\[ \text{Recall}_{yes} = \frac{1}{1 + 0} = 1 \]

• For class “no”
  • 1 true negative (ID 1)
  • 2 false negatives (ID 2 & 4)
  • 1 true positive (ID 3)

\[ \text{Precision}_{no} = \frac{1}{1 + 0} = 1 \]
\[ \text{Recall}_{no} = \frac{1}{1 + 2} = \frac{1}{3} \]
Operators: Performance (Classification)

• Input
  • Labelled Example Set
• Output
  • Performance
• Parameters
  • Performance Measures
Split-Validation / Cross-Validation

• What can you do if you only have one dataset?
  • Use one part of the data for training
  • Use *the other part* of the data for testing

• What if by accident all the easy examples are in the training set?
  • Then your model will not perform that good
  • Better to repeat the learning on different splits of the data

• X-Validation (Cross-Validation)
  • Split the dataset into X parts
  • Select one part for testing, use the rest for training
  • Repeat this until every part was used for training once
Just a reminder ...

- If you use the same data for training and evaluation...

- ... there will be no Easter!

http://s198.photobucket.com/user/FuzzuBunny/media/MeMes/0584342a-0a3e-4ff7-8319-9615604c1203_zps26f59e45.jpg.html
Operators: Split Validation / Cross-Validation

• Input Port
  • Training data (Example Set)

• Output Ports
  • Classification Model
  • Training data (Example Set)
  • Test data (with prediction)
  • Performance

• Parameters
  • Split type
  • Split ratio
  • Sampling type
Nested Processes in Rapid Miner

- Operators can have “inner” processes that define their behaviour
- Split/Cross-Validation Operators have a “Training” and a “Testing” phase
  - Training: This is where you learn your model
  - Testing: This is where you evaluate
From two datasets to Split-Validation
The Mannheim RapidMiner Toolbox

• A Rapid Miner Extension with many great operators

• Developed by researchers from the Data and Web Science Group

• Contains the nearest centroid classifier