The task of classification in Data Mining

Dr Beatriz de la Iglesia
email: b.Iglesia@uea.ac.uk

Session Outline

- Classification definition
- Pre-processing for classification
- Classification algorithms
  - Nearest Neighbour
  - Decision Trees
  - Artificial Neural Networks
- Evaluation of classification algorithms
Data mining

• Within this stage, the DM algorithms are applied to the operational database, suitably cleansed and pre-processed.
• The types of DM algorithm applied here relate to the HLTs (prediction/description) and LLTs (classification, clustering, etc.) described in the Problem Specification stage.

DM algorithms

• For each data mining task which is to be performed, there is a wide range of algorithms available. Some are similar in the way they work, while others work in very different ways.
• Each algorithm has its own strengths and weaknesses, in terms of efficiency, suitability for different data types, simplicity and usefulness of patterns formed etc.
• Depending upon resources, it is wise to use a variety of algorithms for comparison and reliability in the results produced.
Classification

- Each record/example in the DB belongs to a **pre-defined** class or category:
- Each example consists of
  - a single class attribute with some labels (output attribute)
  - the remaining predicting (input or dependant) attributes
- The aim is to predict the class of an example (existing or new) given its predicting attribute values.
- Consider the Customer Response data set, in which we are predicting Response using the other fields.

Marketing example

- The goal is to predict whether a customer will buy a product given sex, country and age.

<table>
<thead>
<tr>
<th>Sex</th>
<th>Country</th>
<th>Age</th>
<th>Buy? Goal/class</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>France</td>
<td>25</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>England</td>
<td>21</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>23</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>England</td>
<td>34</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>21</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>20</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>18</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>34</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>55</td>
<td>No</td>
</tr>
</tbody>
</table>

### Cleaning: data sampling

- Classification models should generalise to new data (no over-fitting).
- The **train data** (with known classes) is used to build the model.
- The **test data** (also with known classes) is used to evaluate the model.
- If parameter experimentation is required the **validation data** is used to make intermediary decisions.
- If the model is good enough, it may be applied to **new data**, i.e. examples with unknown class.

<table>
<thead>
<tr>
<th>Operational Database</th>
<th>Operational Database</th>
<th>Operational Database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Train dataset</strong></td>
<td><strong>Train dataset</strong></td>
<td><strong>Train dataset</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Build Model</strong></td>
<td><strong>Build Model</strong></td>
<td><strong>Build Model</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Adjust parameters</strong></td>
<td><strong>Adjust parameters</strong></td>
<td><strong>Adjust parameters</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validation dataset</strong></td>
<td><strong>Validation dataset</strong></td>
<td><strong>Validation dataset</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Final testing</strong></td>
<td><strong>Final testing</strong></td>
<td><strong>Final testing</strong></td>
</tr>
</tbody>
</table>
Cleaning: missing data

- If missing data is a problem
  - Treat all missing values consistently – e.g. blank, -999, ?
  - Impute using a variety of techniques – but beware of introducing bias.
  - Remove missing data from analysis – but sometimes not enough data left for learning.
  - Some algorithms (e.g. decision trees) can handle missing data so they can be left in the analysis untreated.

Cleaning: balancing

- Problematic when class distribution is very uneven: e.g. 2% respondents versus 98% non-respondents
- Algorithm may predict always a negative response.
- To improve on positive response detection attempt balancing the data:
  - Increase the proportion of the minority class by duplicating records (problematic).
  - Decrease the proportion of the majority class by discarding records.
- We only balance the training data.
Pre-processing

- Sometimes new, more powerfully predictive features need to be added, e.g.
  - Age instead of dob
  - Income/number of children instead of income and number of children separately
  - Aggregations such as total value of sales, total count, etc.
  - BMI instead of height and weight
- Feature selection may also be necessary before application of any algorithm – e.g. using ranking algorithm to rank features or genetic algorithm to select subgroups of features.

Limited data availability

- In case of insufficient data for a train/test partition (say less than 1000 records) can use holdout method.
- Method reserves certain amount of data for testing and uses the remainder for training.
- To mitigate bias caused by poor sample choice, the technique is repeated several times with different random samples.
- In each iteration a certain proportion of the data is randomly selected for training, possibly with stratification, and the remainder used for testing.
- Error rates on different iterations are averaged to yield overall error rate.
Cross validation

- A simple variant using a predetermined number of folds or partitions in the data.
- For example, 10-fold cross validation partitions data into 10 equal partitions. Each in turn is used for testing while the remainder is used for training.
- If stratified sample then this is stratified tenfold cross-validation.
- In stratified sample we check that each of the 10 parts has the classes represented in approximately the same proportions as in the full dataset.
- Learning procedure executed 10 times and error estimates averaged to yield overall error estimate.

Cross-validation Example

Original Dataset

Random partitions

1
Test

2 3 4 5 6 7 8 9 10
Train  Train  Train  Train  Train  Train  Train  Train  Train

Model 1

...

Model n

Averaged Model
Classification

• There are many techniques used for classification, including:
  • Tree induction (decision trees)
  • Nearest Neighbour
  • Artificial Neural Networks (ANNs)
  • Support Vector Machines (SVMs)
  • Rule induction
  • Statistical models
    • Logistic model
    • Discriminant models, etc.
  • ....

kNN Algorithm

• The Nearest neighbour (kNN) approach is also known as case-based reasoning or memory based reasoning.
• Each example is perceived as a point on the representation space whose dimensions are descriptive attributes.
• There is no training! This is known as a lazy approach.
• To classify a record, the nearest neighbouring examples are located. Their class determines the class of the record.
**kNN Algorithm**

- Choice of k may be very important.
- Choice of distance metric may also be critical.
- Examples may be assigned a weight indicating their sphere of influence in the classification decision-making.
- Inputs may need to be normalised to avoid some inputs dominating the calculations.

**Tree induction**

- Decision trees are mainly predictive models that classify the data by asking a classification question at each branch.
- The *internal* nodes of the tree relate to the predicting attributes, while the *leaf* nodes are the predicted class.
- Most tree induction algorithms are *greedy* algorithms that *recursively* generate a tree.
Example

- Start at root node and select a single feature to split records
- Type of split depends on the type of feature

<table>
<thead>
<tr>
<th>Age</th>
<th>Engine</th>
<th>Type</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1100</td>
<td>estate</td>
<td>Low</td>
</tr>
<tr>
<td>19</td>
<td>1300</td>
<td>estate</td>
<td>High</td>
</tr>
<tr>
<td>18</td>
<td>1800</td>
<td>saloon</td>
<td>High</td>
</tr>
<tr>
<td>22</td>
<td>2000</td>
<td>sports</td>
<td>High</td>
</tr>
<tr>
<td>20</td>
<td>1400</td>
<td>saloon</td>
<td>Low</td>
</tr>
<tr>
<td>23</td>
<td>2000</td>
<td>estate</td>
<td>Low</td>
</tr>
</tbody>
</table>

Tree induction

- The tree is built recursively by selecting one attribute at a time - the one that 'best' separates the classes.
- The set of examples is then partitioned according to value of selected attributes.
- This is repeated at each branch node until segmentation is complete.
- Algorithm is greedy because space of all possible trees is enormous so not explored
  - For example, a simple database with 4 predictors (2 with 2 possible values, 2 with 3 possible values) and a binary output supports over $2.2 \times 10^{14}$ sensible decision trees.
### Marketing example

<table>
<thead>
<tr>
<th>Sex</th>
<th>Country</th>
<th>Age</th>
<th>Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>France</td>
<td>23</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>20</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>25</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>18</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>55</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>England</td>
<td>34</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>34</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>England</td>
<td>21</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>21</td>
<td>No</td>
</tr>
</tbody>
</table>

### A possible decision tree

- **Is the tree any good?**

![Decision tree diagram]

- **Internal Branching Node**
- **Leaf Node**
Tree induction

- The aim in building the tree is to ask the best possible question at each branch.
- A single question that would segment the examples would be ideal, but unlikely.
- Imagine the Marketing data in which all French customers also buy. In this case, a single split at the root level is sufficient.
  - If (Country = Germany) Then (Buy? = No);
  - If (Country ≠ Germany) Then (Buy? = Yes).
Performance measures

- Tree performance is usually measured in terms of classification accuracy on the test data.
  - No. of records correctly predicted/no. of records

<table>
<thead>
<tr>
<th>Sex</th>
<th>Country</th>
<th>Age</th>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>England</td>
<td>40</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>25</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>30</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>30</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>England</td>
<td>25</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Classifier Accuracy (train) = 4/5 = 80%

Default Accuracy (train) = 3/5 = 60% (Guess the majority class)

Using a tree with new data

Test Set

<table>
<thead>
<tr>
<th>Sex</th>
<th>Country</th>
<th>Age</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>England</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>25</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>30</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>30</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>England</td>
<td>25</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Tree induction

- The most commonly used tree induction algorithms are:
  - ID3 family (including C4.5 and C5) - uses information gain or variants to split the cases.
  - CART (Classification and regression trees) - uses the Gini index to split the cases.
  - CHAID/XAID (CHI square Automatic Interaction Detector) - uses statistical clustering to split the cases.
  - We look in detail at ID3 and its family.

ID3/C4.5/C5

- ID3 picks predictors and splitting values on the basis of **information gain**.
- This is based upon the **entropy metric**, originated in 1949 (information theory) by Shannon and Weaver (how information could be carried more efficiently over telephone lines)!
- The information equation in this context is
  
  \[
  \text{Info}(D) = - \sum_j p_j \log_2(p_j), \text{ with a range of (0-1),}
  \]
  
  where \( p_j \) is probability of a record belonging to class \( j \).
Entropy metric

Consider an internal node of a DT in which there are two classes (T/F) to predict. At this node there are:
- 100 records
- 30 with T
- 70 with F
Then probability of T is 0.3 and of F is 0.7.
The entropy measure here is:
\[
\text{Info}(D) = -0.3 \log_2(0.3) - 0.7 \log_2(0.7) = 0.88
\]

Entropy metric

Consider now a leaf node of this DT, in which all cases belong to one class, say T. At this node there are:
- 100 records
  - 100 with T (probability of T is 1.0)
  - 0 with F (probability of F is 0.0)
The entropy measure here is:
\[
\text{Info}(D) = -1.0 \log_2(1.0) - 0.0 \log_2(0.0) = 0.0
\]
This is the best value of entropy possible, being a complete segmentation of the cases.
Finally, consider another internal node in which equal number of cases belong to each class. At this node there are:

- 100 records
- 50 with T (probability of T is 0.5)
- 50 with F (probability of F is 0.5)

The entropy measure here is:

\[ \text{Info}(D) = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5) = 1.0 \]

This is the **worst** value of entropy possible.

---

**ID3**

- At each internal node, ID3 decides on which field to use to split the cases, and the splitting value, by attempting to get the **biggest reduction in entropy**.
- The measure actually used to calculate the reduction in entropy is called the **information gain**, hence ID3 will chose at each node the split that produces the biggest information gain.
Information Gain

- For a particular split of the database, $D$, into $k$ subsets ($D_1, \ldots, D_k$), the total information is:

$$Info(D_{\text{split}}) = \sum_k p_k Info(D_k)$$

where $p_k$ is the probability of a record belonging to subset $k$.

- The gain is simply calculated as:

$$Gain = Info(D) - Info(D_{\text{split}})$$

Info Gain Example

- Calculate Information for the whole database
- $j = 2$ (2 classes, Y and N)
  - $j = Y$ \quad $p(j) = 4/10$
  - $j = N$ \quad $p(j) = 6/10$

$$\text{Info}(D) = - ((4/10 \times \log_2(4/10)) + (6/10 \times \log_2(6/10)))$$

$$= 0.970951$$

- Now try splitting the database to find a gain
Info Gain Example

- Splitting by Country creates three subsets of the database: $D_1$=Germany, $D_2$=France, $D_3$=England
- Calculate the sum of the info for each of these subsets
- $D_1$ = Germany, $p(D_1) = 3/10$
  
  **$j = Y$**
  \[ p(j) = 0/3 \]
  
  **$j = N$**
  \[ p(j) = 3/3 \]

  **Info($D_1$) =**
  \[ H\left((0/3 \times \log(0/3) + 3/3 \times \log(3/3))\right) \]
  \[ = 0 \]

- Repeat for $D_2$ and $D_3$

---

Info Gain Example

- $D_2$ = France, $p(D_2) = 5/10$
  
  **$j = Y$**
  \[ p(j) = 2/5 \]
  
  **$j = N$**
  \[ p(j) = 3/5 \]

  **Info($D_2$) =**
  \[ H\left((2/5 \times \log(2/5) + 3/5 \times \log(3/5))\right) \]
  \[ = 0.970951 \]

- $D_3$ = England, $p(D_3) = 2/10$
  
  **$j = Y$**
  \[ p(j) = 2/2 \]
  
  **$j = N$**
  \[ p(j) = 0/2 \]

  **Info($D_3$) =**
  \[ H\left((2/2 \times \log(2/2) + 0/2 \times \log(0/2))\right) \]
  \[ = 0 \]
**Info Gain Example**

- Remember, for a particular split of the database, $D$, into $k$ subsets $(D_1, ..., D_k)$, the total information is:

$$Info(D_{split}) = \sum_k p_k Info(D_k)$$

where $p_k$ is the probability of a record belonging to subset $k$.

$Info(D_{1..3}) = -(((3/10)*0) + ((5/10) * 0.970951) + ((2/10)*0)))$

$Info(D_{1..3}) = 0.485475$

$Gain = 0.970951 - 0.485475 = 0.485476$

---

**Building the tree**
Split on numeric variables

- For numeric variables algorithm will try all potential splits, given data values
- It will then find best and compare with other attributes.

\[
\begin{align*}
\text{Potential root node} & \quad 4Y \quad 6N \\
\text{age?} & \\
\leq 25 & \quad 3Y \quad 3N \\
> 25 & \quad 1Y \quad 3N
\end{align*}
\]

Entropy = 0.97
Gain = 0.97 - 0.92 = 0.05

The next level in tree

- Hence, based on the gains available, the attribute that gives the biggest reduction in entropy (the biggest gain) is “Country”.
- This is chosen as the split at the root and three branches are created.

\[
\begin{align*}
\text{Country?} & \quad 4Y \quad 6N \\
\text{Germany} & \quad 0Y \quad 3N \\
\text{France} & \quad 2Y \quad 3N \\
\text{England} & \quad 2Y \quad 0N
\end{align*}
\]

- Germany and England have 0 entropy, cannot be improved
  France entropy = -(0.4(log₂0.4) + 0.6(log₂0.6)) = 0.97
- Algorithm executes recursively on France node.
C4.5 and C5

• C4.5 is an enhancement of ID3
  • A normalised form of information gain, the in Gain Ratio, is used

\[
gainRatio(D, A) = \frac{gain(D, A)}{splitInfo(D, A)}
\]

where
\[
splitInfo(D, A) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} \log_2 \left( \frac{|D_j|}{|D|} \right)
\]

• Fields with missing values can be used
• Fields with continuous values can be used
• Pruning is introduced
• Rules can be derived automatically

• C5 is a more recent enhancement of C4.5
  • Claims to be faster, more accurate and supports boosting.

Pruning the tree

• In some real-world DT cases, the tree gets very large and cumbersome, with some leaf nodes containing only one or two records. The tree has thus been over-trained on the train set and will not generalise very well, with worse performance likely on the test set.
• It is possible to cut off some leaf nodes (pruning) while testing the tree on the test data so that a similar performance is achieved on both data sets.
Pruning the tree

- The pruning algorithm examines the tree from the bottom up.
- Each sub-tree is replaced by a leaf or its most frequently used branch if to do so would lead to a lower estimated error rate.
- The estimated error rate includes a parameter used to alter the severity of pruning.

![Full Tree and Pruned Tree Diagram]

Rule derivation

Rules can be extracted directly from induction trees.

If (country = Germany) then (Buy? = No)

What are the other rules?
Finally, for decision trees

- They can be used for both description and prediction, but primarily the latter.
- Take a potential customer who is male, French and age 36. Will he be likely to buy?
- Then there is the English couple….
- Who will you spend more time on trying to persuade to buy your product?

Exercise

Quilan’s golf database starts off as below.

<table>
<thead>
<tr>
<th>outlook</th>
<th>temp</th>
<th>humidity</th>
<th>windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>75</td>
<td>70</td>
<td>TRUE</td>
<td>play</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
<td>90</td>
<td>TRUE</td>
<td>don't play</td>
</tr>
<tr>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>FALSE</td>
<td>don't play</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>FALSE</td>
<td>don't play</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>FALSE</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
<td>90</td>
<td>TRUE</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>FALSE</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>TRUE</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>FALSE</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>TRUE</td>
<td>don't play</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>TRUE</td>
<td>don't play</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
<td>80</td>
<td>FALSE</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>FALSE</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>FALSE</td>
<td>play</td>
</tr>
</tbody>
</table>

Calculate Information Gain for Outlook and Windy and decide which should be use for the root of a DT.
Artificial neural networks (ANNs)

- There is little doubt that ANNs have played a formative part in the growth of data mining.
- Although they have a major disadvantage, in that they operate like a black box, making decisions that are almost impossible to track or audit, they are extremely flexible, with many different models of ANNs available for a wide range of DM tasks.
- We introduce the basic ANN here, the MLP, or multi-layer perceptron.

Artificial neural networks

- The MLP consists of input nodes (one for each predicting field), hidden nodes, and output nodes (usually one for each possible value of the class to be predicted).
- Each layer is fully connected to the preceding layer and to the following layer by weighted connections.
- The training cases are presented, one at a time, to the net and these weights are adjusted until the MLP model is sufficiently accurate.
During training, all cases with known classes (train set) are presented to the net many times. **Feed-forward stage:**

For each case, the predicting attribute values (suitably normalised) are fed into the input nodes. The input nodes simply pass the corresponding values to each node in the hidden layer. However, each link has an associated **weight**, which adjusts the value before it enters the hidden node.
Feed-forward (continued)

• Each hidden neuron sums the weighted inputs and then applies a transfer function (sigmoid) to the sum to produce a value (usually between 0 and 1) which is passed to the output node.

• The output nodes behave like hidden nodes, summing the weighted values from each hidden node, and outputting a value between 0 and 1.

• The ‘class’ that is chosen for this case is simply the largest of the two values from the output nodes “Yes” or “No”.

Back-propagation stage

• For each case, the predicted class is compared with the actual known class and the ‘error’ between predicted and actual class is propagated back through the network to adjust the weights of the links, firstly those from hidden to output layer, then from input to hidden layer, so as to reduce the error.

• This may be done after each case or after each entire data set (epoch).

• Training usually stops when no improvement is achieved.
Decision regions for MLP

- Single layer Perceptrons are capable of solving only linearly separable problems
- MLPs with three layers are capable to approximate any desired bounded continuous function
  - The units in the first hidden layer generate hyperplanes to divide the input space in half-spaces.
  - Units in the second hidden layer form convex regions as intersections of these hyperplanes.
  - Output units form unisons of the convex regions into arbitrarily shaped, convex, non-convex or disjoint regions
### Universal approximators

- A MLP can approximate any complex decision boundary to divide the input space with arbitrary accuracy, producing a (0) when the input is in one region and an output of (1) in the other.
- This property, known as a universal approximation capability, poses the essential problems of adequate model complexity in depth and size, i.e. the number of nodes and layers, and controlling the network training process to prevent overfitting.
The Bias-Variance Trade-off

- Training set error is not a good indicator of operational performance (generalisation)
- We can divide the generalisation error into two components
  - Bias - error resulting from a failure to reproduce the desired function.
  - Variance - error due to modelling noise in the training data.
- Bias and variance and complementary quantities.
- Optimal generalisation results from a compromise between low bias and low variance.

Illustrative problem

- Consider the problem of learning a sinusoid corrupted by noise.
- We want to learn underlying function and ignore noise
Unregularised solution

- The network has “overfitted” the data
- Bias is low, but the variance is high so generalisation is poor.

Generalisation

- NN can be too powerful!
- Variance depends on two factors:
  - Network size - a complex mapping requires large numbers of hidden units/weights.
  - Magnitude of weights - a mapping with a high degree of curvature requires weights with a large magnitude.
- The two approaches to the bias/variance tradeoff are:
  - Regularisation - add a term to the error function to penalise complex mappings.
  - Structural stabilisation - control the complexity of the network (growing/pruning algorithms).
Heavily regularised solutions

- The network has “underfitted” the data.
- Variance is low, but the bias is high so generalisation is poor.

Optimally Regularised Solution

- Network learns underlying function, but ignores noise
- Compromise between bias and variance optimises generalisation
• One of the drawbacks of using a neural network is that it is a black box - classification is made by a very complex set of mathematical transfer functions.

• Organisations who wish to understand the database more may be disinclined to use them.

• However, if the model has high accuracy and applicability it can be used as a benchmark for other methods.

Discussion time

• For each of the following scenarios estate what DM techniques may be suitable for application:
  • Develop a model to determine the characteristics of individuals that default on a mortgage loan. They are a minority in the database and need to be predicted accurately. The model must make transparent decisions.
  • Develop a model to select recipients of a mail-shot campaign for a new product. The response rate for mail-shot campaigns is usually low but the cost of sending the mail is also low. The model does not need to be transparent.
Evaluation: confusion matrix

- Most evaluation metrics for classification are obtained from the confusion matrix

<table>
<thead>
<tr>
<th>Confusion Matrix</th>
<th>Target</th>
<th>Positive</th>
<th>Negative</th>
<th>Positive Predictive Value</th>
<th>a/(a+b)</th>
<th>Negative Predictive Value</th>
<th>d/(c+d)</th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Positive</td>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a/(a+c)</td>
<td>d/(b+d)</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accuracy = (a+d)/(a+b+c+d)

Error rate/default accuracy

- Error rate is complementary to Accuracy.
- Error Rate is percent of misclassified records out of the total records in the validation data

\[
ER = \frac{c+b}{a+b+c+d}
\]

Accuracy = 1 - ER

- Default accuracy is the percentage of records that belong to the majority class.
- In the absence of a model, classify all records as majority class.
- Gives a benchmark for models.
ROC curve

- Receiver operating characteristic (ROC) illustrates the performance of a binary classifier system as its discrimination threshold is varied.
- The curve is created by plotting the true positive rate (TPR) or sensitivity against the false positive rate (FPR) at various threshold settings (1-specificity).
- Visualises relationship between specificity and sensitivity and shows the performance of classifiers.
- Generated by plotting the cumulative distribution function of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability in x-axis.

Example from Wikipedia

- ROC space represent trade-offs between true positive (benefits) and false positive (costs).
- Each instance of a confusion matrix represents one point in ROC space.
- Perfect classification in (0,1) coordinate.
- Random guess in diagonal line. Points above are good results. Points below are poor.
ROC curves

- In prediction, there is often a probability associated with prediction, a score, X.
- Given a threshold, T, an instance is classified as positive if $X > T$.
- X follows a probability density function $f_1(x)$ if instance belongs to + class, $f_0(x)$ otherwise.
- $\text{TPR}(T) = \int_T^\infty f_1(x) \, dx$ and $\text{FPR}(T) = \int_T^\infty f_0(x) \, dx$
- ROC curve plots parametrically TPR(T) against FPR(T) with T as the varying parameter.

Example ROC Curve
AUC

- Summarising the ROC curve into a single number loses information about the pattern of tradeoffs of the particular algorithm.
- However, discrimination power of model can be measured by area under the roc curve (AUC).
- AUC can be computed by various methods. Could be interpreted as follows:
  - 90-1 = excellent
  - .80-.90 = good
  - .70-.80 = fair
  - .60-.70 = poor
  - .50-.60 = fail

Learning outcomes

- What is classification?
- Cleaning/Pre-processing operations for classification including sampling, balancing and missing data handling.
- Some understanding of the following algorithms:
  - Decision trees,
  - ANNs,
  - kNN.
- Evaluation for classification
• C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1005.