Model’s Performance Measures

- Evaluating the performance of a classifier
  
- Taking into account misclassification costs

- Class imbalance problem

Evaluation of a Classifier

- How predictive is the model we learned?
  - Which performance measure to use?

- Natural performance measure for classification problems: \textit{error rate} on a test set
  - \textit{Success}: instance’s class is predicted correctly
  - \textit{Error}: instance’s class is predicted incorrectly
  - \textit{Error rate}: proportion of errors made over the whole set of instances
  - \textit{Accuracy}: proportion of correctly classified instances over the whole set of instances

\[ \text{accuracy} = 1 - \text{error rate} \]
Evaluation of a Classifier

- Is the accuracy measure enough to evaluate the performance of a classifier?
  - Accuracy can be of little help, if classes are severely unbalanced
  - Classifiers are biased to predict well the majority class
    - e.g. decision trees based on information gain measure
    - e.g. prunning rules for the rare class can be difficult, resulting in rules involving large number of attributes and low coverage (overfitting)
  - Predicting correctly one of the classes may be more important

- How frequently records of class + are correctly classified?

Confusion Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=No</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

TPR (sensitivity) = \( \frac{TP}{TP+FN} \)  
FNR = \( \frac{FN}{TP+FN} \) = 1 - TPR

TNR (specificity) = \( \frac{TN}{TN+FP} \)  
FPR = \( \frac{FP}{TN+FP} \) = 1 - TNR
Other Metrics

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

- **a**: TP (true positive)
- **b**: FN (false negative)
- **c**: FP (false positive)
- **d**: TN (true negative)

Precision (p) = \( \frac{TP}{TP + FP} \)

Recall (r) = \( \frac{TP}{TP + FN} = TPR \)

F-measure (F) = \( \frac{2 \times r \times p}{r + p} \)

F tends to be closer to the smaller of precision and recall.

Example: A good medical test

TPR (sensitivity) = \( \frac{TP}{TP + FN} \)

TNR (specificity) = \( \frac{TN}{TN + FP} \)

- **Accuracy**: high
- **Sensitivity** (TPR): high
  - Good at classifying records of class +, i.e. few FNs
- **Specificity** (TNR): high
  - Good at classifying records of class -, i.e. few FPs
Example: A bad medical test

- + + + + + +
- + + + + + +
- + + + + + +

TPR (sensitivity) = \( \frac{TP}{TP+FN} \)
TNR (specificity) = \( \frac{TN}{TN+FP} \)

- + + + + + +
- + + + + + +
- + + + + + +

Classifier decision boundaries

• **Accuracy**: very high
• **Sensitivity** (TPR): 100%
  ✓ Good at classifying records of class +, i.e. zero FNs
• **Specificity** (TNR): 0%
  ✓ Classifies correctly half of the – records

Example: A better medical test

- + + + + + +
- + + + + + +
- + + + + + +

TPR (sensitivity) = \( \frac{TP}{TP+FN} \)
TNR (specificity) = \( \frac{TN}{TN+FP} \)

- + + + + + +
- + + + + + +
- + + + + + +

Classifier decision boundaries

• **Accuracy**: high
• **Sensitivity** (TPR): 100%
  ✓ Good at classifying records of class +, i.e. zero FNs
• **Specificity** (TNR): 50%
  ✓ Bad at classifying records of class –, FPR = 50%
Example: classify documents (about X or not)

TPR (sensitivity) = \( \frac{TP}{TP+FN} \)

TNR (specificity) = \( \frac{TN}{TN+FP} \)

- **Accuracy**: very high
- **Sensitivity** (TPR): 100%
  - Good at classifying records of class +, i.e. zero FNs
- **Specificity** (TNR): very high
  - Good at classifying records of class -, i.e. few FPs

Bad classifier: about half of the documents classified as being about X have nothing to do with X

Example: classify documents (about X or not)

Precision = \( \frac{TP}{(TP + FP)} \)

Recall (TPR) = \( \frac{TP}{(TP + FN)} \)

- **Accuracy**: very high
- **Recall** (TPR): 100%
  - Good at classifying records of class +, i.e. zero FNs
- **Precision**: \( \leq 50\% \) (low)
  - Too many FPs compared to the TPs
Example: classify documents (about X or not)

- +
- + + + + +
- + + + +
- + +

Classifier decision boundaries

<table>
<thead>
<tr>
<th>TPs</th>
<th>TNs</th>
<th>FNs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Precision = \(\frac{TP}{TP + FP}\)
Recall (TPR) = \(\frac{TP}{TP + FN}\)

- **Accuracy**: very high
- **Recall (TPR)**: very high
  - Good at classifying records of class +, i.e. few FNs
- **Precision**: 100%
  - No FPs

Classification Costs

- Associate a cost with misclassification
  - Class +: *with brain tumour*
  - Class -: *without brain tumour*
    - A **FN** error can be highly costly: a patient with brain tumour is missed and may die
    - A **FP** error is not so serious: a healthy patient is sent to extra screening unnecessarily
    - A classifier biased toward to the + class is preferred
Cost Sensitive Learning

- How to build a classifier that takes into account the misclassification costs?

![Diagram of FN errors cost more than FP errors](C(- | +) > C(+ | -))

Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>C(Yes</td>
</tr>
<tr>
<td>Class=No</td>
<td>C(Yes</td>
</tr>
</tbody>
</table>

\( C(i|j) \): Cost of misclassifying class \( j \) example as class \( i \)

- How to use the cost matrix?
  - Cost-sensitive evaluation of classifiers (average cost of the model)
  - Cost-sensitive classification
  - Cost-sensitive learning
Cost-sensitive Evaluation

<table>
<thead>
<tr>
<th></th>
<th>PREDICTED CLASS</th>
<th>Cost Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUAL CLASS</td>
<td></td>
<td>C(ij)</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>- 1</td>
<td>0</td>
</tr>
</tbody>
</table>

Model M₁

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 150 40</td>
<td>+ -</td>
</tr>
<tr>
<td>- 60 250</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy = 80%
Cost = 3910

Model M₂

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 250 45</td>
<td>+ -</td>
</tr>
<tr>
<td>- 5 200</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy = 90%
Cost = 4255

Cost-sensitive Learning

- Cost information can be incorporated into classification
  - Decision trees
    - Cost-sensitive splitting criteria
    - Cost-sensitive pruning
- General way to make any learning method cost-sensitive
  - Generate data samples with different proportions of Yes (+) and No (-) instances
  - Ex: + instances are increased by a factor of 10
    - Oversample the Yes class
    - Classifier skewed toward avoidance of errors on the Yes instances
    - Less false negatives than false positives
- Read paper about “MetaCost”
  - Available in WEKA
Cost Sensitive Learning

- Oversample the positive class

Find models that describe a rare class

- Why? Correct classification of the rare class may have greater value than correct classification of the majority class

- Problems:
  - Classifiers tend to be biased to the majority class
  - Highly specialized patterns susceptible to the presence of noise
  - Accuracy (error rate) measure can be misleading

- Example: consider a 2-class problem
  - Number of class - examples = 9990
  - Number of class + examples = 10
  - If model predicts everything to be class -, then accuracy is 9990/10000 = 99.9%

How can we tackle the problem?
Class Imbalance Problem

- **Oversampling** of the rare class (+)
  - **Problem**: more sensitive to noise and overfitting

- **Undersampling** of the majority class
  - **Problem**: useful cases of the + class may not be chosen for training

- **Cost-sensitive learning**
  - $C(- | +) > C(+ | -)$ (cost of FN higher than cost of FP)
  - Use **MetaCost** algorithm

Oversampling
**Undersampling**

- Preprocess
  - Supervised
    - Instance
      - Resample
        - NoReplacement = false
        - biasToUniform = 1
      - spreadSubSample
        - distributionSpread = 1
(undersamples the majority class)

Weka: Sampling

- Preprocess
  - Supervised
    - Instance
      - Resample
        - NoReplacement = false
        - biasToUniform = 1
      - spreadSubSample
        - distributionSpread = 1
(undersamples the majority class and oversamples the rare class)