Machine Learning: Symbolische Ansätze

Evaluation and Cost-Sensitive Learning

- Evaluation
  - Hold-out Estimates
  - Cross-validation
- Significance Testing
  - Sign test

- ROC Analysis
  - Cost-Sensitive Evaluation
  - ROC space
  - ROC convex hull
  - Rankers and Classifiers
  - ROC curves
  - AUC
- Cost-Sensitive Learning
Evaluation of Learned Models

- Validation through experts
  - a domain expert evaluates the plausibility of a learned model
    + but often the only option (e.g., clustering)
    - subjective, time-intensive, costly

- Validation on data
  - evaluate the accuracy of the model on a separate dataset drawn from the same distribution as the training data
    - labeled data are scarce, could be better used for training
    + fast and simple, off-line, no domain knowledge needed, methods for re-using training data exist (e.g., cross-validation)

- On-line Validation
  - test the learned model in a fielded application
    + gives the best estimate for the overall utility
    - bad models may be costly
### Confusion Matrix
(Concept Learning)

<table>
<thead>
<tr>
<th>Classified as +</th>
<th>Classified as -</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is +</strong></td>
<td><strong>false negatives (fn)</strong></td>
</tr>
<tr>
<td>true positives (tp)</td>
<td></td>
</tr>
<tr>
<td><strong>Is −</strong></td>
<td><strong>true negatives (tn)</strong></td>
</tr>
<tr>
<td>false positives (fp)</td>
<td></td>
</tr>
<tr>
<td><strong>tp + fp</strong></td>
<td><strong>fn + tn</strong></td>
</tr>
</tbody>
</table>

- the confusion matrix summarizes all important information
  - how often is class $i$ confused with class $j$
- most evaluation measures can be computed from the confusion matrix
  - accuracy
  - recall/precision, sensitivity/specificity
  - …
## Basic Evaluation Measures

- **true positive rate:** \[ tpr = \frac{tp}{tp + fn} \]
  - percentage of correctly classified positive examples
- **false positive rate:** \[ fpr = \frac{fp}{fp + tn} \]
  - percentage of negative examples incorrectly classified as positive
- **false negative rate:** \[ fnr = \frac{fn}{tp + fn} = 1 - tpr \]
  - percentage of positive examples incorrectly classified as negative
- **true negative rate:** \[ tnr = \frac{tn}{fp + tn} = 1 - fpr \]
  - percentage of correctly classified negative examples
- **accuracy:** \[ acc = \frac{tp + tn}{P + N} \]
  - percentage of correctly classified examples
  - can be written in terms of \( tpr \) and \( fpr \): \[ acc = \frac{P}{P + N} \cdot tpr + \frac{N}{P + N} \cdot (1 - fpr) \]
- **error:** \[ err = \frac{fp + fn}{P + N} = 1 - acc = \frac{P}{P + N} \cdot (1 - tpr) + \frac{N}{P + N} \cdot fpr \]
  - percentage of incorrectly classified examples
Confusion Matrix
(Multi-Class Problems)

- for multi-class problems, the confusion matrix has many more entries:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$n_{A,A}$</td>
<td>$n_{B,A}$</td>
<td>$n_{C,A}$</td>
<td>$n_{D,A}$</td>
</tr>
<tr>
<td>B</td>
<td>$n_{A,B}$</td>
<td>$n_{B,B}$</td>
<td>$n_{C,B}$</td>
<td>$n_{D,B}$</td>
</tr>
<tr>
<td>C</td>
<td>$n_{A,C}$</td>
<td>$n_{B,C}$</td>
<td>$n_{C,C}$</td>
<td>$n_{D,C}$</td>
</tr>
<tr>
<td>D</td>
<td>$n_{A,D}$</td>
<td>$n_{B,D}$</td>
<td>$n_{C,D}$</td>
<td>$n_{D,D}$</td>
</tr>
<tr>
<td></td>
<td>$n_A$</td>
<td>$n_B$</td>
<td>$n_C$</td>
<td>$n_D$</td>
</tr>
</tbody>
</table>

- accuracy is defined analogously to the two-class case:

$$\text{accuracy} = \frac{n_{A,A} + n_{B,B} + n_{C,C} + n_{D,D}}{|E|}$$
Out-of-Sample Testing

- Performance cannot be measured on training data
  - overfitting!

- Reserve a portion of the available data for testing
  - typical scenario
    - 2/3 of data for training
    - 1/3 of data for testing (evaluation)
  - a classifier is trained on the training data
  - and tested on the test data
    - e.g., confusion matrix is computed for test data set

- Problems:
  - waste of data
  - labelling may be expensive
  - high variance
    - often: repeat 10 times or → cross-validation
Typical Learning Curves

Quelle: Winkler 2007, nach Mitchell 1997,
Cross-Validation

- Algorithm:
  - split dataset into $x$ (usually 10) partitions
  - for every partition $X$
    - use other $x-1$ partitions for learning and partition $X$ for testing
  - average the results

- Example: 4-fold cross-validation
Leave-One-Out Cross-Validation

- \( n \)-fold cross-validation
  - where \( n \) is the number of examples:
    - use \( n-1 \) examples for training
    - 1 example for testing
    - repeat for each example

- Properties:
  - makes best use of data
    - only one example not used for testing
  - no influence of random sampling
    - training/test splits are determined deterministically
  - typically very expensive
    - but, e.g., not for k-NN (Why?)
  - bias
    - example see exercises
Experimental Evaluation of Algorithms

- Typical experimental setup (in % Accuracy):
  - evaluate $n$ algorithms on $m$ datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Grading</th>
<th>Select</th>
<th>Stacking</th>
<th>Voting</th>
</tr>
</thead>
<tbody>
<tr>
<td>audiology</td>
<td>83.36</td>
<td>77.61</td>
<td>76.02</td>
<td>84.56</td>
</tr>
<tr>
<td>autos</td>
<td>80.93</td>
<td>80.83</td>
<td>82.20</td>
<td>83.51</td>
</tr>
<tr>
<td>balance-scale</td>
<td>89.89</td>
<td>91.54</td>
<td>89.50</td>
<td>86.16</td>
</tr>
<tr>
<td>breast-cancer</td>
<td>73.99</td>
<td>71.64</td>
<td>72.06</td>
<td>74.86</td>
</tr>
<tr>
<td>breast-w</td>
<td>96.70</td>
<td>97.47</td>
<td>97.41</td>
<td>96.82</td>
</tr>
<tr>
<td>colic</td>
<td>84.38</td>
<td>84.48</td>
<td>84.78</td>
<td>85.08</td>
</tr>
<tr>
<td>credit-a</td>
<td>86.01</td>
<td>84.87</td>
<td>86.09</td>
<td>86.04</td>
</tr>
<tr>
<td>credit-g</td>
<td>75.64</td>
<td>75.48</td>
<td>76.17</td>
<td>75.23</td>
</tr>
<tr>
<td>diabetes</td>
<td>75.53</td>
<td>76.86</td>
<td>76.32</td>
<td>76.25</td>
</tr>
<tr>
<td>glass</td>
<td>74.35</td>
<td>74.44</td>
<td>76.45</td>
<td>75.70</td>
</tr>
<tr>
<td>heart-c</td>
<td>82.74</td>
<td>84.09</td>
<td>84.26</td>
<td>81.55</td>
</tr>
<tr>
<td>heart-h</td>
<td>83.64</td>
<td>85.78</td>
<td>85.14</td>
<td>83.16</td>
</tr>
<tr>
<td>heart-statlog</td>
<td>84.22</td>
<td>83.56</td>
<td>84.04</td>
<td>83.30</td>
</tr>
<tr>
<td>hepatitis</td>
<td>83.42</td>
<td>83.03</td>
<td>83.29</td>
<td>82.77</td>
</tr>
<tr>
<td>ionosphere</td>
<td>91.85</td>
<td>91.34</td>
<td>92.82</td>
<td>92.42</td>
</tr>
<tr>
<td>iris</td>
<td>95.13</td>
<td>95.20</td>
<td>94.93</td>
<td>94.93</td>
</tr>
<tr>
<td>labor</td>
<td>93.68</td>
<td>90.35</td>
<td>91.58</td>
<td>93.86</td>
</tr>
<tr>
<td>lymph</td>
<td>83.45</td>
<td>81.69</td>
<td>80.20</td>
<td>84.05</td>
</tr>
<tr>
<td>primary-t.</td>
<td>49.47</td>
<td>49.23</td>
<td>42.63</td>
<td>46.02</td>
</tr>
<tr>
<td>segment</td>
<td>98.03</td>
<td>97.05</td>
<td>98.08</td>
<td>98.14</td>
</tr>
<tr>
<td>sonar</td>
<td>85.05</td>
<td>85.05</td>
<td>85.58</td>
<td>84.23</td>
</tr>
<tr>
<td>soybean</td>
<td>93.91</td>
<td>93.69</td>
<td>92.90</td>
<td>93.84</td>
</tr>
<tr>
<td>vehicle</td>
<td>74.46</td>
<td>73.90</td>
<td>79.89</td>
<td>72.91</td>
</tr>
<tr>
<td>vote</td>
<td>95.93</td>
<td>95.95</td>
<td>96.32</td>
<td>95.33</td>
</tr>
<tr>
<td>vowel</td>
<td>98.74</td>
<td>99.06</td>
<td>99.00</td>
<td>98.80</td>
</tr>
<tr>
<td>zoo</td>
<td>96.44</td>
<td>95.05</td>
<td>93.96</td>
<td>97.23</td>
</tr>
</tbody>
</table>

- Can we conclude that algorithm X is better than Y? How?
Summarizing Experimental Results

- Averaging the performance
  - May be deceptive:
    - algorithm A is 0.1% better on 19 datasets with thousands of examples
    - algorithm B is 2% better on 1 dataset with 50 examples
    - A is better, but B has the higher average accuracy
  - In our example: “Grading” is best on average

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Grading</th>
<th>Select</th>
<th>Stacking</th>
<th>Voting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>85.04</td>
<td>84.59</td>
<td>84.68</td>
<td>84.88</td>
</tr>
</tbody>
</table>

- Counting wins/ties/losses
  - now “Stacking” is best
  - Results are “inconsistent”:
    - Grading > Select > Voting > Grading
  - How many “wins” are needed to conclude that one method is better than the other?

<table>
<thead>
<tr>
<th></th>
<th>Grading</th>
<th>Select</th>
<th>Stacking</th>
<th>Voting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grading</td>
<td>—</td>
<td>15/1/10</td>
<td>11/0/15</td>
<td>12/0/14</td>
</tr>
<tr>
<td>Select</td>
<td>10/1/15</td>
<td>—</td>
<td>10/0/16</td>
<td>14/0/12</td>
</tr>
<tr>
<td>Stacking</td>
<td>15/0/11</td>
<td>16/0/10</td>
<td>—</td>
<td>15/1/10</td>
</tr>
<tr>
<td>Voting</td>
<td>14/0/12</td>
<td>12/0/14</td>
<td>10/1/15</td>
<td>—</td>
</tr>
</tbody>
</table>
Sign Test

- **Given:**
  - A coin with two sides (heads and tails)
- **Question:**
  - How often do we need heads in order to be sure that the coin is not fair?
- **Null Hypothesis:**
  - The coin is fair ($P(\text{heads}) = P(\text{tails}) = 0.5$)
  - We want to refute that!
- **Experiment:**
  - Throw up the coin $N$ times
- **Result:**
  - $i$ heads, $N - i$ tails
  - What is the probability of observing $i$ under the null hypothesis?
Sign Test

- **Given:**
  - A coin with two sides (heads and tails)

- **Question:**
  - How often do we need heads in order to be sure that the coin is not fair?

- **Null Hypothesis:**
  - The coin is fair ($P(\text{heads}) = P(\text{tails}) = 0.5$)
  - We want to refute that!

- **Experiment:**
  - Throw up the coin $N$ times

- **Result:**
  - $i$ heads, $N-i$ tails
  - What is the probability of observing $i$ under the null hypothesis?

---

Two Learning Algorithms (A and B)

On how many datasets must A be better than B to ensure that A is a better algorithm than B?

Both Algorithms are equal.

Run both algorithms on $N$ datasets

$i$ wins for A on $N-i$ wins for B
Sign Test: Summary

We have a binomial distribution with \( p = \frac{1}{2} \)

- the probability of having \( i \) successes is \( P(i) = \binom{N}{i} p^i (1-p)^{N-i} \)

- the probability of having at most \( k \) successes is (one-tailed test)

\[
P(i \leq k) = \sum_{i=1}^{k} \binom{N}{i} \frac{1}{2^i} \cdot \frac{1}{2^{N-i}} = \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{i}
\]

- the probability of having at most \( k \) successes or at least \( N-k \) successes is (two-tailed test)

\[
P(i \leq k \lor i \geq N-k) = \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{i} + \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{N-i} = \frac{1}{2^{N-1}} \sum_{i=1}^{k} \binom{N}{i}
\]

- for large \( N \), this can be approximated with a normal distribution

Illustrations taken from http://www.mathsrevision.net/
Table
Sign Test

- Example:
  - 20 datasets
  - Alg. A vs. B
    - A 4 wins
    - B 14 wins
    - 2 ties (not counted)
  - we can say with a certainty of 95% that B is better than A
  - but not with 99% certainty!

- Online: http://www.fon.hum.uva.nl/Service/Statistics/Sign_Test.html
Properties

- Sign test is a very simple test
  - does not make any assumption about the distribution

- Sign test is very conservative
  - If it detects a significant difference, you can be sure it is
  - If it does not detect a significant difference, a different test that models the distribution of the data may still yield significance

- Alternative tests:
  - two-tailed $t$-test:
    - incorporates magnitude of the differences in each experiment
    - assumes that differences form a normal distribution

- Rule of thumb:
  - Sign test answers the question “How often?”
  - $t$-test answers the question “How much?”
Problem of Multiple Comparisons

- **Problem:**
  - With 95% certainty we have
    - a probability of 5% that one algorithm appears to be better than the other
    - even if the null hypothesis holds!
  → if we make many pairwise comparisons the chance that a “significant” difference is observed increases rapidly

- **Solutions:**
  - Bonferroni adjustments:
    - **Basic idea:** tighten the significance thresholds depending on the number of comparisons
    - Too conservative
  - Friedman and Nemenyi tests
    - recommended procedure (based on average ranks)
    http://jmlr.csail.mit.edu/papers/v7/demsar06a.html
Cost-Sensitive Evaluation

- Predicting class \( i \) instead of the correct \( j \) is associated with a cost factor \( C(i \mid j) \)
  - 0/1-loss (accuracy):
    \[
    C(i \mid j) = \begin{cases} 
      0 & \text{if } i = j \\ 
      1 & \text{if } i \neq j 
    \end{cases}
    \]
  - general case for concept learning:

<table>
<thead>
<tr>
<th></th>
<th>Classified as +</th>
<th>Classified as −</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is +</td>
<td>( C(+</td>
<td>+) )</td>
</tr>
<tr>
<td>Is −</td>
<td>( C(+</td>
<td>−) )</td>
</tr>
</tbody>
</table>
Examples

- **Loan Applications**
  - rejecting an applicant who will not pay back → minimal costs
  - accepting an applicant who will pay back → gain
  - accepting an applicant who will not pay back → big loss
  - rejecting an applicant who would pay back → loss

- **Spam-Mail Filtering**
  - rejecting good E-mails (ham) is much worse than accepting a few spam mails

- **Medical Diagnosis**
  - failing to recognize a disease is often much worse than to treat a healthy patient for this disease
Cost-Sensitive Evaluation

- Expected Cost (Loss):
  \[ L = tpr \cdot C(+|+) + fpr \cdot C(+|-) + fnr \cdot C(-|+) + tnr \cdot C(-|-) \]

- If there are no costs for correct classification:
  \[ L = fpr \cdot C(+|-) + fnr \cdot C(-|+) = fpr \cdot C(+|-) + (1 - tpr) \cdot C(-|+) \]

  - note the general form:
    - this is essentially the relative cost metric we know from rule learning

- Distribution of positive and negative examples may be viewed as a cost parameter
  - error is a special case
    \[ C(+|-) = \frac{N}{P+N}, \quad C(-|+) = \frac{P}{P+N} \]
  - we abbreviate the costs with \( c_- = C(+|-), \quad c_+ = C(-|+) \)
ROC Analysis

- Receiver Operating Characteristic
  - origins in signal theory to show tradeoff between hit rate and false alarm rate over noisy channel

- Basic Objective:
  - Determine the best classifier for varying cost models
    - accuracy is only one possibility, where true positives and false positives receive equal weight

- Method:
  - Visualization in ROC space
    - each classifier is characterized by its measured $fpr$ and $tpr$
  - ROC space is like coverage space (→ rule learning) except that axes are normalized
    - x-axis: false positive rate $fpr$
    - y-axis: true positive rate $tpr$
Example ROC plot

ROC plot produced by ROCon (http://www.cs.bris.ac.uk/Research/MachineLearning/rocon/)
ROC spaces vs. Coverage Spaces

- ROC spaces are normalized coverage spaces
  - Coverage spaces may have different shapes of the rectangular area \((0,P) \times (0,N)\)
  - ROC spaces are normalized to a square \((0,1) \times (0,1)\)

<table>
<thead>
<tr>
<th>property</th>
<th>ROC space</th>
<th>coverage space</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-axis</td>
<td>(\text{FPR} = \frac{n}{N})</td>
<td>(n)</td>
</tr>
<tr>
<td>y-axis</td>
<td>(\text{TPR} = \frac{p}{P})</td>
<td>(p)</td>
</tr>
<tr>
<td>empty theory (R_0)</td>
<td>((0,0))</td>
<td>((0,0))</td>
</tr>
<tr>
<td>correct theory (R)</td>
<td>((0,1))</td>
<td>((0,P))</td>
</tr>
<tr>
<td>universal theory (\tilde{R})</td>
<td>((1,1))</td>
<td>((N,P))</td>
</tr>
<tr>
<td>resolution</td>
<td>((\frac{1}{N}, \frac{1}{P}))</td>
<td>((1,1))</td>
</tr>
<tr>
<td>slope of diagonal</td>
<td>1</td>
<td>(\frac{P}{N})</td>
</tr>
<tr>
<td>slope of (p = n) line</td>
<td>(\frac{N}{P})</td>
<td>1</td>
</tr>
</tbody>
</table>
Costs and Class Distributions

- assume no costs for correct classification and a cost ratio 
  \[ r = c_-/c_+ \] for incorrect classifications
  - this means that false positives are \( r \) times as expensive as false negatives
- this situation can be simulated by increasing the proportion of negative examples by a factor of \( r \)
  - e.g. by replacing each negative example with \( r \) identical copies of the same example
  - each mistake on negative examples is then counted with \( r \), a mistake on positive examples is still counted with 1
- computing the error in the new set corresponds to computing a cost-sensitive evaluation in the original dataset

➔ the same trick can be used for cost-sensitive learning!
Example

- Coverage space with equally distributed positive and negative examples ($P = N$)

  - assume a false positive is twice as bad as a false negative (i.e., $c_- = 2c_+$)
  - this situation can be modeled by counting each covered negative example twice
Example

- Doubling the number of negative examples
  - changes the shape of the coverage space and the location of the points
Example

- Mapping back to ROC space
  - yields the same (relative) location of the original points

- but the angle of the isometrics has changed as well
- accuracy in the coverage space with doubled negative examples corresponds to a line with slope \( r = 2 \) in ROC space
Important Lessons

- Class Distributions and Cost Distributions are interchangable
  - cost-sensitive evaluation (and learning) can be performed by changing
    the class distribution (e.g., duplication of examples)
  - Therefore there is always a coverage space that corresponds to
    the current cost distribution
    - in this coverage space, the cost ratio $r = 1$, i.e., positive and negative
      examples are equally important
- The ROC space results from normalizing this rectangular
  coverage space to a square
  - cost isometrics in the ROC space are accuracy isometrics in the
    corresponding coverage space
- The location of a classifier in ROC space is invariant to changes
  in the class distribution
  - but the slope of the isometrics changes when a different cost model is used
**ROC isometrics**

- Iso-cost lines connects ROC points with the same costs $c$
  - $c = c_+ \cdot (1 - tpr) + c_- \cdot fpr$
  - $tpr = \frac{c_-}{c_+} \cdot fpr + \left( \frac{c}{c_+} - 1 \right)$

- Cost isometrics are parallel ascending lines with slope $r = \frac{c_-}{c_+}$
  - e.g., error/accuracy slope = $N/P$

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Selecting the optimal classifier

For uniform class distribution \( (r = 1) \), C4.5 is optimal.

Classifiers in ROC space

TP Rate

FP Rate

SVM

C4.5

NB

Ripper

CN2
Selecting the optimal classifier

With four times as many positives as negatives ($r = 1/4$), SVM is optimal.

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Selecting the optimal classifier

With four times as many negatives as positives ($r = 4$), CN2 is optimal

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Selecting the optimal classifier

- With less than 9% positives, predicting always negative is optimal
- With less than 11% negatives, predicting always positive is optimal
The ROC convex hull

Classifiers on the convex hull minimize costs for some cost model.

Any performance on a line segment connecting two ROC points can be achieved by interpolating between the classifiers.

Classifiers below the convex hull are always suboptimal.
**Interpolating Classifiers**

- Given two learning schemes we can reach any point on the convex hull!
  - TP and FP rates for scheme 1: \( tpr_1 \) and \( fpr_1 \)
  - TP and FP rates for scheme 2: \( tpr_2 \) and \( fpr_2 \)

- If scheme 1 is used to predict \( q \times 100\% \) of the cases and scheme 2 for the rest, then
  - TP rate for combined scheme: \( tpr_q = q \cdot tpr_1 + (1 - q) \cdot tpr_2 \)
  - FP rate for combined scheme: \( fpr_q = q \cdot fpr_1 + (1 - q) \cdot fpr_2 \)
Rankers and Classifiers

- A scoring classifier outputs scores $f(x,+)$ and $f(x,-)$ for each class
  - e.g. estimate probabilities $P(+ | x)$ and $P(- | x)$
  - scores don’t need to be normalised
- $f(x) = f(x,+)/f(x,-)$ can be used to rank instances from most to least likely positive
  - e.g. odds ratio $P(+ | x) / P(- | x)$
- Rankers can be turned into classifiers by setting a threshold on $f(x)$
- Example:
  - Naïve Bayes Classifier for two classes is actually a ranker
  - that has been turned into classifier by setting a probability threshold of 0.5 (corresponds to a odds ratio threshold of 1.0)
  - $P(+ | x) > 0.5 > 1 - P(+ | x) = P(- | x)$ means that class $+$ is more likely

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Drawing ROC Curves for Rankers

Performance of a ranker can be visualized via a ROC curve

- **Naïve method:**
  - consider all possible thresholds
  - only \(k+1\) thresholds between the \(k\) instances need to be considered
  - each threshold corresponds to a new classifier
  - for each classifier
    - construct confusion matrix
    - plot classifier at point \((\text{fpr}, \text{tpr})\) in ROC space

- **Practical method:**
  - rank test instances on decreasing score \(f(x)\)
  - start in \((0,0)\)
    - if the next instance in the ranking is +: move \(1/P\) up
    - if the next instance in the ranking is -: move \(1/N\) to the right
    - make diagonal move in case of ties

**Note:** It may be easier to draw in coverage space (1 up/right).
A sample ROC curve

Slide adapted from Witten/Frank, Data Mining
Properties of ROC Curves for Rankers

- The curve visualizes the quality of the ranker or probabilistic model on a test set, without committing to a classification threshold
  - aggregates over all possible thresholds

- The slope of the curve indicates class distribution in that segment of the ranking
  - diagonal segment → locally random behaviour

- Concavities indicate locally worse than random behaviour
  - convex hull corresponds to discretizing scores
  - can potentially do better: repairing concavities
Some example ROC curves

- Good separation between classes, convex curve
Some example ROC curves

- Reasonable separation, mostly convex
Some example ROC curves

- Fairly poor separation, mostly convex

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Some example ROC curves

- Poor separation, large and small concavities

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Some example ROC curves

- Random performance
Comparing Rankers with ROC Curves

If low $fpr$ is more important, use Method A.

Inbetween, interpolate between A and B.

If high $tpr$ is more important, use Method B.

Slide adapted from Witten/Frank, Data Mining.
AUC: The Area Under the ROC Curve

AUC

1 - AUC

True positives

0 20% 40% 60% 80% 100%

False positives

0 20% 40% 60% 80% 100%
The AUC metric

- The Area Under ROC Curve (AUC) assesses the ranking in terms of separation of the classes
  - all the positives before the negatives: AUC = 1
  - random ordering: AUC = 0.5
  - all the negatives before the positives: AUC = 0
- can be computed from the step-wise curve as:
  \[
  \text{AUC} = \frac{1}{P \cdot N} \sum_{i=1}^{N} (r_i - i) = \frac{1}{P \cdot N} \left( \sum_{i=1}^{N} r_i - \sum_{i=1}^{N} i \right) = \frac{S_- - N(N+1)/2}{P \cdot N}
  \]
  where \( r_i \) is the rank of a negative example and \( S_- = \sum_{i=1}^{N} r_i \)
- Equivalent to the Mann-Whitney-Wilcoxon sum of ranks test
  - estimates probability that randomly chosen positive example is ranked before randomly chosen negative example

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Multi-Class AUC

- ROC-curves and AUC are only defined for two-class problems (concept learning)
  - Extensions to multiple classes are still under investigation

Some Proposals for extensions:
- In the most general case, we want to calculate Volume Under ROC Surface (VUS)
  - number of dimensions proportional to number of entries in confusion matrix
- Projecting down to sets of two-dimensional curves and averaging
  - MAUC (Hand & Till, 2001): \[ \text{MAUC} = \frac{2}{c \cdot (c - 1)} \sum_{i < j} \text{AUC}(i, j) \]
    - unweighted average of AUC of pairwise classification (1-vs-1)
  - (Provost & Domingos, 2001):
    - weighted average of 1-vs-all AUC for class \( c \) weighted by \( P(c) \)
Cost-sensitive learning

- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard rule or decision tree learner

- Simple methods for cost-sensitive learning:
  - If classifier is able to handle weighted instances
    - weighting of instances according to costs
    - covered examples are not counted with 1, but with their weight
  - For any classifier
    - resampling of instances according to costs
    - proportion of instances with higher weights/costs will be increased
  - If classifier returns a score $f$ or probability $P$
    - varying the classification threshold
Costs and Example Weights

- The effort of duplicating examples can be saved if the learner can use example weights
  - positive examples get a weight of $c_+$
  - negative examples get a weight of $c_-$
- All computations that involve counts are henceforth computed with weights
  - instead of counting, we add up the weights
- Example:
  - Precision with weighted examples is
    \[
    prec = \frac{\sum_{x \in Cov \cap Pos} w_x}{\sum_{x \in Cov} w_x}
    \]
    - $w_x$ is the weight of example $x$
    - $Cov$ is the set of covered examples
    - $Pos$ is the set of positive examples
  - if $w_x = 1$ for all $x$, this reduces to the familiar
    \[
    prec = \frac{p}{p+n}
    \]
Minimizing Expected Cost

- Given a specification of costs for correct and incorrect predictions
  - an example should be predicted to have the class that leads to the lowest expected cost
  - not necessarily to the lowest error
- The expected cost (loss) for predicting class \( i \) for an example \( x \)
  - sum over all possible outcomes, weighted by estimated probabilities
  \[
  L(i, x) = \sum_j C(i|j) P(j|x)
  \]
- A classifier should predict the class that minimizes \( L(i, x) \)
  - If the classifier can estimate the probability distribution \( P(i | x) \) of an example \( x \)
Minimizing Cost in Concept Learning

- For two classes:
  - predict positive if it has the smaller expected cost:
    \[
    C(\text{+|+}) \cdot \Pr(\text{+|x}) + C(\text{+|}-) \cdot \Pr(\text{-|x}) \leq C(\text{-|+}) \cdot \Pr(\text{+|x}) + C(\text{-|}-) \cdot \Pr(\text{-|x})
    \]

    - cost if we predict positive
    - cost if we predict negative

  - as \(\Pr(\text{+|x}) = 1 - \Pr(\text{-|x})\):
    - predict positive if \(\Pr(\text{+|x}) \geq \frac{C(\text{-|+}) - C(\text{-|}-)}{C(\text{+|+}) - C(\text{-|+}) - C(\text{-|}-)}\)

- Example:
  - Classifying a spam mail as ham costs 1, classifying ham as spam costs 99, correct classification cost nothing:
    \(\Rightarrow\) classify as spam if spam-probability is at least 99%
Calibrating a Ranking Classifier

- What is the right threshold of the ranking score $f(x)$ if the ranker does not estimate probabilities?
  - classifier can be calibrated by choosing appropriate threshold that minimizes costs
  - may also lead to improved performance in accuracy if probability estimates are bad (e.g., Naïve Bayes)

- Easy in the two-class case:
  - calculate cost for each point/threshold while tracing the curve
  - return the threshold with minimum cost

- Non-trivial in the multi-class case

**Note:** threshold selection is part of the classifier training and must therefore be performed on the training data!
Example: Uncalibrated threshold

True and false positive rates achieved by default threshold (NB. worse than always predicting majority class!)

Accuracy isometric for this domain

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
Example: Calibrated threshold

Optimal achievable accuracy

Slide adapted from P. Flach, ICML-04 Tutorial on ROC
References

  http://www-cse.ucsd.edu/users/elkan/rescale.pdf

  http://www.csee.usf.edu/~candamo/site/papers/ROCintro.pdf

- Peter Flach: *The many faces of ROC analysis in machine learning*, Tutorial held at ICML-04. 
  http://www.cs.bris.ac.uk/~flach/ICML04tutorial/
