Evaluation and Credibility

How much should we believe in what was learned?
Outline

• Introduction
• Classification with Train, Test, and Validation sets
  – Handling Unbalanced Data; Parameter Tuning
• Cross-validation
• Comparing Data Mining Schemes
Introduction

• How predictive is the model we learned?
• Error on the training data is \textit{not} a good indicator of performance on future data
  –\textit{Q: Why?}
  –\textit{A: Because new data will probably not be exactly the same as the training data!}
• Overfitting – fitting the training data too precisely - usually leads to poor results on new data
Evaluation issues

• Possible evaluation measures:
  – Classification Accuracy
  – Total cost/benefit – when different errors involve different costs
  – Lift and ROC curves
  – Error in numeric predictions

• How reliable are the predicted results?
Classifier error rate

• Natural performance measure for classification problems: error rate
  – *Success*: instance’s class is predicted correctly
  – *Error*: instance’s class is predicted incorrectly
  – Error rate: proportion of errors made over the whole set of instances

• *Training set error rate*: is way too optimistic!
  – you can find patterns even in random data
Evaluation on “LARGE” data

• If many (thousands) of examples are available, including several hundred examples from each class, then a simple evaluation is sufficient
  – Randomly split data into training and test sets (usually 2/3 for train, 1/3 for test)
• Build a classifier using the train set and evaluate it using the test set.
Classification Step 1:
Split data into train and test sets

THE PAST
Results Known

Data → Training set → Testing set
Classification Step 2: Build a model on a training set

THE PAST
Results Known

Data -> Model Builder

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Training set

Testing set
Classification Step 3:
Evaluate on test set (Re-train?)

Data

Results Known

Training set

Model Builder

Evaluate

Predictions

Testing set

Table:

<table>
<thead>
<tr>
<th>Data</th>
<th>Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Model Builder:

Evaluate:

Predictions:

+    -    +
Handling unbalanced data

- Sometimes, classes have very unequal frequency
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - Medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don’t buy, 1% buy
  - Security: >99.99% of Americans are not terrorists

- Similar situation with multiple classes
- Majority class classifier can be 97% correct
Balancing unbalanced data

• With two classes, a good approach is to build **BALANCED** train and test sets, and train model on a balanced set
  – randomly select desired number of minority class instances
  – add equal number of randomly selected majority class

• Generalize “balancing” to multiple classes
  – Ensure that each class is represented with approximately equal proportions in train and test
A note on parameter tuning

• It is important that the test data is not used *in any way* to create the classifier

• Some learning schemes operate in two stages:
  – Stage 1: builds the basic structure
  – Stage 2: optimizes parameter settings

• The test data can’t be used for parameter tuning!

• Proper procedure uses three sets: *training data, validation data, and test data*
  – Validation data is used to optimize parameters
Making the most of the data

• Once evaluation is complete, *all the data* can be used to build the final classifier

• Generally, the larger the training data the better the classifier (but returns diminish)

• The larger the test data the more accurate the error estimate
Classification:
Train, Validation, Test split

Data

Results Known

Training set

Model Builder

Evaluate

Predictions

Validation set

Final Test Set

Final Model

Final Evaluation
*Predicting performance*

- Assume the estimated error rate is 25%. How close is this to the true error rate?
  - Depends on the amount of test data
- Prediction is just like tossing a biased (!) coin
  - “Head” is a “success”, “tail” is an “error”
- In statistics, a succession of independent events like this is called a Bernoulli process
- Statistical theory provides us with confidence intervals for the true underlying
*Confidence intervals*

- We can say: $p$ lies within a certain specified interval with a certain specified confidence.
- Example: $S=750$ successes in $N=1000$ trials
  - Estimated success rate: 75%
  - How close is this to true success rate $p$?
    - Answer: with 80% confidence $p \in [73.2,76.7]$
- Another example: $S=75$ and $N=100$
  - Estimated success rate: 75%
  - With 80% confidence $p \in [69.1,80.1]$
*Mean and variance (also Mod 7)*

- Mean and variance for a Bernoulli trial: $p, p (1-p)$
- Expected success rate $f = S/N$
- Mean and variance for $f$: $p, p (1-p)/N$
- For large enough $N$, $f$ follows a Normal distribution
- c% confidence interval $[-z \leq X \leq z]$ for random variable with 0 mean is given by:
  $$\Pr[-z \leq X \leq z] = c$$
- With a symmetric distribution:
  $$\Pr[-z \leq X \leq z] = 1 - 2 \times \Pr[X \geq z]$$

witten & eibe
*Confidence limits*

Confidence limits for the normal distribution with 0 mean and a variance of 1:

Thus:

\[
\Pr[-1.65 \leq X \leq 1.65] = 90\%
\]

To use this we have to reduce our random variable \(f\) to have 0 mean and unit variance
Transforming $f$

- Transformed value for $f$:
\[
\frac{f - p}{\sqrt{p(1 - p)/N}}
\]
(i.e. subtract the mean and divide by the standard deviation)

- Resulting equation:
\[
\Pr\left[-z \leq \frac{f - p}{\sqrt{p(1 - p)/N}} \leq z\right] = c
\]

- Solving for $p$:
\[
p = \left(f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}\right) / \left(1 + \frac{z^2}{N}\right)
\]
**Examples**

- \( f = 75\%, \ N = 1000, \ c = 80\% \) (so that \( z = 1.28 \)) \( \in [0.732, 0.767] \)

- \( f = 75\%, \ N = 100, \ c = 80\% \) (so that \( z = 1.28 \)):
  \[
p \in [0.691, 0.801]
  \]

- Note that normal distribution assumption is only valid for large \( N \) (i.e. \( N > 100 \))

- \( f = 75\%, \ N = 10, \ c = 80\% \) (so that \( z = 1.28 \)) \( p \in [0.549, 0.881] \)

  (should be taken with a grain of salt)
Evaluation on “small” data

• The *holdout* method reserves a certain amount for testing and uses the remainder for training
  – Usually: one third for testing, the rest for training

• For small or “unbalanced” datasets, samples might not be representative
  – Few or none instances of some classes

• *Stratified sample*: advanced version of balancing the data
  – Make sure that each class is represented with approximately equal proportions in both subsets
Repeated holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the repeated holdout method
- Still not optimum: the different test sets
Cross-validation

• *Cross-validation* avoids overlapping test sets
  – First step: data is split into $k$ subsets of equal size
  – Second step: each subset in turn is used for testing and the remainder for training

• This is called *$k$-fold cross-validation*

• Often the subsets are stratified before the cross-validation is performed

• The error estimates are averaged to yield an overall error estimate
Cross-validation example:

— Break up data into groups of the same size

— Hold aside one group for testing and use the rest to build model

— **Test**

— Repeat
More on cross-validation

• Standard method for evaluation: stratified ten-fold cross-validation
• Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
• Stratification reduces the estimate’s variance
• Even better: repeated stratified cross-validation
  E.g. ten-fold cross-validation is repeated ten
Leave-One-Out cross-validation

• Leave-One-Out:
  a particular form of cross-validation:
  – Set number of folds to number of training instances
  – I.e., for $n$ training instances, build classifier $n$ times
• Makes best use of the data
• Involves no random subsampling
• Very computationally expensive
  – (exception: NN)
Leave-One-Out-CV and stratification

• Disadvantage of Leave-One-Out-CV: stratification is not possible
  – It guarantees a non-stratified sample because there is only one instance in the test set!

• Extreme example: random dataset split equally into two classes
  – Best inducer predicts majority class
  – 50% accuracy on fresh data
  – Leave-One-Out-CV estimate is 100% error!
The bootstrap

• CV uses sampling *without replacement*
  – The same instance, once selected, can not be selected again for a particular training/test set

• The *bootstrap* uses sampling *with replacement* to form the training set
  – Sample a dataset of $n$ instances $n$ times *with replacement* to form a new dataset of $n$ instances
  – Use this data as the training set
  – Use the instances from the original dataset that don’t occur in the new training set for testing
*The 0.632 bootstrap*

- Also called the 0.632 bootstrap
  - A particular instance has a probability of \(1 - \frac{1}{n}\) of not being picked
  - Thus its probability of ending up in the test data is:
    \[
    \left(1 - \frac{1}{n}\right) \approx e^{-1} = 0.368
    \]
  - This means the training data will contain approximately 63.2% of the instances
*Estimating error with the bootstrap*

- The error estimate on the test data will be very pessimistic
  - Trained on just ~63% of the instances
- Therefore, combine it with the resubstitution error:
  \[
  err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}
  \]
- The resubstitution error gets less weight than the error on the test data
- Repeat process several times with different replacement samples; average the results
*More on the bootstrap*

- Probably the best way of estimating performance for very small datasets
- However, it has some problems
  - Consider the random dataset from above
  - A perfect memorizer will achieve 0% resubstitution error and ~50% error on test data
  - Bootstrap estimate for this classifier:
    \[ \frac{0.632 \cdot 50\% + 0.368 \cdot 0\%}{1} = 31.6\% \]
  - True expected error: 50%
Comparing data mining schemes

- Frequent situation: we want to know which one of two learning schemes performs better
- Note: this is domain dependent!
- Obvious way: compare 10-fold CV estimates
- Problem: variance in estimate
- Variance can be reduced using repeated CV
- However, we still don’t know whether the results are reliable
Direct Marketing Paradigm

• Find most likely prospects to contact
• Not everybody needs to be contacted
• Number of targets is usually much smaller than number of prospects

• Typical Applications
  – retailers, catalogues, direct mail (and e-mail)
  – customer acquisition, cross-sell, attrition prediction
  – ...

Direct Marketing Evaluation

• **Accuracy on the entire dataset is not the right measure**

• **Approach**
  – develop a target model
  – score all prospects and rank them by decreasing score
  – select top P% of prospects for action

• **How to decide what is the best selection?**
**Model-Sorted List**

Use a model to assign score to each customer

Sort customers by decreasing score

Expect more targets (hits) near the top of the list

<table>
<thead>
<tr>
<th>No</th>
<th>Score</th>
<th>Target</th>
<th>CustID</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.97</td>
<td>Y</td>
<td>1746</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.95</td>
<td>N</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.94</td>
<td>Y</td>
<td>2478</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.93</td>
<td>Y</td>
<td>3820</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.92</td>
<td>N</td>
<td>4897</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>99</td>
<td>0.11</td>
<td>N</td>
<td>2734</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.06</td>
<td>N</td>
<td>2422</td>
<td></td>
</tr>
</tbody>
</table>

3 hits in top 5% of the list

If there are 15 targets overall, then top 5 has 3/15=20% of targets
Definition: CPH(P,M) = % of all targets in the first P% of the list scored by model M
CPH frequently called Gains

Q: What is expected value for CPH(P,Random) ?

A: Expected value for CPH(P,Random) = P
CPH: Random List vs Model-ranked list

5% of random list have 5% of targets, but 5% of model ranked list have 21% of targets

CPH(5%, model) = 21%.
Lift\((P,M)\) = \frac{\text{CPH}(P,M)}{P}

Lift (at 5%)
= 21\% / 5\%
= 4.2
better
than random

Note: Some (including Witten & Eibe) use “Lift” for what we call CPH.
Lift Properties

• $Q$: $\text{Lift}(P, \text{Random}) =$
  $-A$: 1 (expected value, can vary)

• $Q$: $\text{Lift}(100\%, M) =$
  $-A$: 1 (for any model M)

• $Q$: Can lift be less than 1?
  $-A$: yes, if the model is inverted (all the non-targets precede targets in the list)

• Generally, a better model has higher lift
ROC curves

- **ROC curves** are similar to gains charts
  - Stands for “receiver operating characteristic”
  - Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel

- Differences from gains chart:
  - \( y \) axis shows percentage of true positives in sample rather than absolute number
  - \( x \) axis shows percentage of false positives in sample rather than sample size
A sample ROC curve

- Jagged curve—one set of test data
- Smooth curve—use cross-validation

witten & eibe
Cross-validation and ROC curves

• Simple method of getting a ROC curve using cross-validation:
  – Collect probabilities for instances in test folds
  – Sort instances according to probabilities

• This method is implemented in WEKA

• However, this is just one possibility
  – The method described in the book generates an ROC curve for each fold and averages them
For a small, focused sample, use method A.
For a larger one, use method B.
In between, choose between A and B with appropriate probabilities.
The convex hull

- Given two learning schemes we can achieve any point on the convex hull!
- TP and FP rates for scheme 1: $t_1$ and $f_1$
- TP and FP rates for scheme 2: $t_2$ and $f_2$
- If scheme 1 is used to predict $100\times q \%$ of the cases and scheme 2 for the rest, then
  - TP rate for combined scheme: $q \times t_1 + (1-q) \times t_2$
  - FP rate for combined scheme: $q \times f_2 + (1-q) \times f_2$
Cost Sensitive Learning

- There are two types of errors

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>TP: True positive</td>
<td>FN: False negative</td>
</tr>
<tr>
<td>No</td>
<td>FP: False positive</td>
<td>TN: True negative</td>
</tr>
</tbody>
</table>

- Machine Learning methods usually minimize FP+FN
- Direct marketing maximizes TP
Different Costs

• In practice, true positive and false negative errors often incur different costs

• Examples:
  – Medical diagnostic tests: does X have leukemia?
  – Loan decisions: approve mortgage for X?
  – Web mining: will X click on this link?
  – Promotional mailing: will X buy the product?
  – …
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 decision tree learner

• Simple methods for cost-sensitive learning:
  – Re-sampling of instances according to costs
  – Weighting of instances according to costs

• Some schemes are inherently cost-sensitive, e.g. naïve Bayes
KDD Cup 98 – a Case Study

• Cost-sensitive learning/data mining widely used, but rarely published

• Well known and public case study: KDD Cup 1998
  – Data from Paralyzed Veterans of America (charity)
  – Goal: select mailing with the highest profit
  – Evaluation: Maximum actual profit from selected list (with mailing cost = $0.68)
    • Sum of (actual donation-$0.68) for all records with predicted/expected donation > $0.68

• More in a later lesson
*Measures in information retrieval

- Percentage of retrieved documents that are relevant: 
  \[ \text{precision} = \frac{TP}{TP+FP} \]
- Percentage of relevant documents that are returned: 
  \[ \text{recall} = \frac{TP}{TP+FN} \]
- Precision/recall curves have hyperbolic shape
- Summary measures: average precision at 20%, 50% and 80% recall (three-point average recall)
- \[ F\text{-measure} = \frac{(2 \times \text{recall} \times \text{precision})}{(\text{recall} + \text{precision})} \]
**Summary of measures**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Plot</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift chart</td>
<td>Marketing</td>
<td>TP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subset size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP/(TP+FP)/(TP+FP+TN+FN)</td>
</tr>
<tr>
<td>ROC curve</td>
<td>Communications</td>
<td>TP rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FP rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP/(TP+FN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FP/(FP+TN)</td>
</tr>
<tr>
<td>Recall-precision curve</td>
<td>Information retrieval</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP/(TP+FN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP/(TP+FP)</td>
</tr>
</tbody>
</table>

Witten & Eibe