Data Mining

Practical Machine Learning Tools and Techniques

Slides for Sections 5.1-5.4
Testing and Predicting Performance

Evaluation: the key to success

- How predictive is the model we learned?
- Error on the training data is *not* a good indicator of performance on future data
  - Otherwise nearest neighbor would be the optimum classifier!
- Simple solution that can be used if lots of (labeled) data is available:
  - Split data into training and test set
- However: (labeled) data is usually limited
  - More sophisticated techniques need to be used

Issues in evaluation

- Statistical reliability of estimated differences in performance ($\rightarrow$ significance tests)
- Choice of performance measure:
  - Number of correct classifications
  - Accuracy of probability estimates
  - Error in numeric predictions
- Costs assigned to different types of errors
  - Many practical applications involve costs

Training and testing I

- 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
- *Resubstitution error:* error rate obtained from training data
- Resubstitution error is usually quite optimistic!

Training and testing II

- *Test set*: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem
- Test and training data may differ in nature
  - Example: classifiers built using customer data from two different towns *A* and *B*
    - To estimate performance of classifier from town *A* in completely new town, test it on data from *B*

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: build the basic structure
  - Stage 2: optimize 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
- The larger the test data the more accurate the error estimate
- *Holdout* procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set *and* test set should be large!

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 – How?
- 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 proportion

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

- 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:

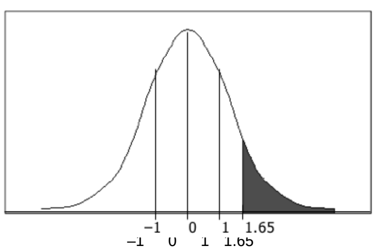
$$Pr[-z \leq X \leq z] = c$$

- With a symmetric distribution:

$$Pr[-z \leq X \leq z] = 1 - 2 \times Pr[x \geq z]$$

Confidence limits

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



$Pr[X \geq z]$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

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

Examples

- f = 75%, N = 1000, c = 80% (so that z = 1.28):

$$p \in [0.732, 0.767]$$

- 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):

$$p \in [0.549, 0.881]$$

not really meaningful

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
  -
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses *stratification*
  - Ensures 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 stratificiation)
  - 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 overlap
  - Can we prevent overlapping?

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: split data into k subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
- 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

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 times and results are averaged (reduces the variance)

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

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 1–1/*n* of *not* being picked
  - Thus its probability of ending up in the test data is:

$$(1-\frac{1}{n})^n \approx e^{-1} \approx 0.368$$

  - 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:

$$err = 0.632 \times e_{\text{test instances}} + 0.368 \times e_{\text{training\_instances}}$$

- 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:

$$err = 0.632 \times 50\% + 0.368 \times 0\% = 31.6\%$$

  - True expected error: 50%