F20DL Data Mining and Machine Learning

Diana Bental
(with material from David Corne and slides from http://www.cs.waikato.ac.nz/ml/weka/book.html)

This lecture…

- Main theme: testing the model
- Also
  - What if we don't have much data?
  - How to compare very different data mining schemes?

Testing the model

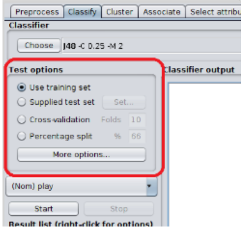
- We build a model (decision tree or set of rules etc) from some data – the *training data*
- We want to *predict* how well the model will perform on *new* data

Last time…

- We talked about some *measures* we can use
  - Accuracy and error rates
  - Sensitivity and specificity
  - Precision and recall
  - Confusion matrix and kappa statistic
  - Taking costs into account
  - Lift and ROC curves
- And noted that accuracy (success rate) is not always the best measure

Testing the model

- How to test the model?



1. Use the training set

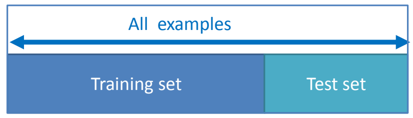
- Calculate the Resubstitution error:
  - The error rate obtained from running the model on the training data
- Resubstitution error is (hopelessly) optimistic!
- Error on the training data is *not* a good indicator of performance on new data
- Otherwise 1-NN would be the optimum classifier

2. Use a Test Set

- *Test* set: independent instances that have played no part in formation of the model
- 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 a classifier from town A in any completely new town, test it on data from B

3. Holdout method (Weka: Percentage split)

- Simple solution that can be used if lots of (labeled) data is available:
- Split data into a *training* set and *test* set



- Estimate the error rate of the trained classifier

Use of Hold-Out: Parameter optimization

- Some learning schemes operate in two stages:
  - Stage 1: build a basic structure
  - Stage 2: optimize parameter settings
- The parameter optimization data can't be used for testing!
- Use *three* sets:
  1. *training* data
  2. *validation* data
  3. *test* data
- *Validation* data is used to optimize parameters
- It is important that the test data is not used *in any way* to create the classifier

Holdout method

- Dilemma: ideally both training set and test set should be large!
  - Often use about 2/3 training to 1/3 testing
- Sparse dataset: may not have enough data to train while keeping data aside for testing

Limited amount of (labelled) data

- Screening personal loans
  - only 1000 appropriate examples (people with borderline credit ratings)
- Electricity supply data
  - 15 years and 5000 days but only 15 Christmas days and Thanksgivings
- Electromechanical fault diagnosis
  - 20 years of data but only 300 usable examples
- Detecting oil spills
  - A lot of human effort to classify images

Making the most of the data

- Once evaluation is complete, all the data (including test data) can be used to build the final classifier

Predicting performance

- Assume the estimated error rate (from testing) 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 correct / successful prediction
  - "Tail" is an incorrect prediction / 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

Use 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 trials
  - Estimated success rate: 75%
  - With 80% confidence p in [69.1,80.1]
- Note that really N needs to be large, e.g. > 100

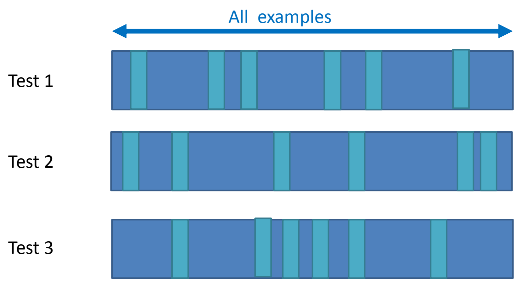
Stratification

- Problem: the sample might not be representative
  - maybe a class is not represented at all in the training or test data
- Stratification: Select the test sample so that each class is represented in (roughly) the same proportions in the test and training data

Repeated Holdout method

- Holdout is a single experiment – so our error estimate may not be very reliable
- 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 are averaged to yield an overall error rate
- This is called the *repeated holdout method*

Repeated Holdout method



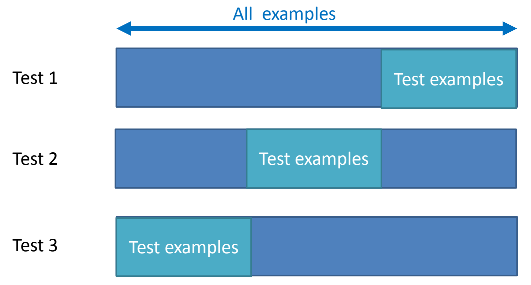
Repeated Holdout method

- Still not optimum: the different test sets overlap
- Can we prevent overlapping?

4. Cross-validation

- Cross-validation avoids overlapping test sets
- *K-fold cross-validation*
  - First step: split data into *k* subsets of equal size (*folds)*
  - Second step: use each subset in turn for testing, the remainder for training
  - The error estimates are averaged to yield an overall error estimate
- Perform *k* tests, using *k*-1 *folds* for training and 1 *fold* for testing each time

Example: 3-fold cross-validation



Cross-validation

- All the instances in the dataset are used for both training and testing
- *But not at the same time*

Cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
  - Subsets may be stratified before the cross-validation is performed
    - Stratification reduces the estimate's variance
  - Why ten?
    - Extensive experiments have shown that this is the best choice to get an accurate estimate
    - There is also some theoretical evidence for this
  - Even better: repeated (stratified) cross-validation
    - ten-fold cross-validation is repeated ten times and the results are averaged (reduces the variance)

So far: Main methods for testing

1. Use the training set
2. Supply a new test set
3. Hold-out methods
4. Cross-validation

Two methods for small datasets

- Leave-one-out cross-validation
- The 0.632 bootstrap

Small datasets: Leave-One-Out cross-validation

- Cross-validation
- Set the number of folds = number of training instances
  - Leave one instance out, train on the others, and test on that one instance
  - So for *n* training instances, build the classifier *n* times
  - Average the results to get an overall error
- Makes best use of the data
  - Largest possible training sets give best possible classifier
- Involves no random subsampling
  - No need to repeat
- But: very computationally expensive

Disadvantage of Leave-One-Out-CV: no stratification

- It *guarantees* a non-stratified test sample because there is only one instance in the test set!
- Extreme (pathological) example: random dataset split equally into two classes
  - "Perfect" learning method predicts the majority class each time
  - 50% accuracy on fresh data set but
  - Leave-One-Out-CV estimate is 100% error!
  - Then the majority class is always the *opposite* of the test case

Small datasets: The bootstrap

- Cross validation 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
  - Forms a new dataset of *n* instances
  - Some are repeated
  - Use this data as the training set.
  - Use the leftover instances from the original dataset (that weren't put into the new training set) for testing

Small datasets: The 0.632 bootstrap

- Why 0.632?
- A particular instance has a probability of $1 - 1/n$ of *not* being picked
- So the probability of an instance ending up in the test data is $\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$
- And the probabilty of an instance ending up in the training data is $1 - 0.368 = 0.632$
- So the training data will contain approx 63.2% of the data instances

Small datasets: The 0.632 bootstrap

- The error estimate with the test data will be pessimistic
  - Smaller training set (63%, compared to 90% for 10-fold CV)
- So combine with the *resubstitution* error (error on the training instances)
- But give the resubstitution error less weight
- *error* = $0.632 \times error_{bootstrap} + 0.378 \times error_{resub}$
- Repeat the whole process with different samples for the training set, and take an average.

Small datasets: The 0.632 bootstrap

- Best way to test learning mechanisms for very small datasets (usually)
- But consider again the (pathological) completely random two-class dataset example
- A perfect learning mechanism would give 0% resubstitution error and 50% error on test data
- True error ≈ 50%
- Bootstrap error ≈ 30%
- err = 0.632 x 50 + 0.368 x 0 ≈ 30%
- Too optimistic!

Comparing data mining schemes

- Which of two learning schemes performs better for some application?
  - Note: this is domain dependent!
  - Obvious way: compare 10-fold CV estimates for both schemes
  - Generally sufficient in applications
  - What about machine learning research?
  - Need to show convincingly that a new method works better on *many* different data sets and applications

Comparing data mining schemes

- In a particular domain
  - Compare two schemes A and B …
  - For a given amount of training data
  - On average, across all possible training sets
- If we could have an infinite amount of data from the domain:
  - Sample infinitely many datasets of specified size
  - Obtain cross-validation estimate on each dataset for each scheme
- Check if mean accuracy for scheme A is better than mean accuracy for scheme B

Comparing schemes: paired t-test

- In practice we have limited data and a limited number of estimates for computing the mean
- Student's t-test tells whether the means of two samples are significantly different
- In our case the samples are cross-validation estimates for different datasets from the domain
- Use a paired t-test because the individual samples are paired
- The same cross-validation is applied twice

Comparing schemes: unpaired t-test

- If the cross-validation estimates are from different datasets, they are no longer paired (or maybe we have different numbers of c-v estimates for the two schemes)
- Then we have to use an un-paired t-test

Summary

- Main theme:
  - Four approaches to testing a model
- Also
  - Two methods for very small datasets
  - Statistical comparisons for different data mining schemes
- Take-away message:
  - Separate datasets for training , tuning, and testing