Lecture 14:

- Classifier evaluation

Announcements:

- HW-5 out (due next Thurs)
Training and Testing: ... for estimating classifier (algorithm) performance

Building a classifier starts with a learning (training) phase
- based on predefined set of examples – the training set

The classifier is then evaluated for predictive performance
- based on another set of examples – the testing set
- we use the actual labels (ground truth) of the examples to test the predictions

In general, we want to be careful to avoid overfitting
- encoding particular characteristics/anomalies of training set in classifier
- i.e., the model is highly specialized to the training data set
- also underfitting: too simple of a model (e.g., linear instead of polynomial)

We’ll discuss different ways to select training and testing sets
1. Holdout method
2. Random subsampling
3. $k$-fold cross validation and variants
4. Bootstrap method
Note: Some things to keep in mind ...

- Our goal is to determine how good an algorithm is for a classification problem
- ... or to try to develop a good (specialized) algorithm for the problem
- This is different than building a model for deployment
- Some ML approaches also build/update models via testing (e.g., neural nets)
- The classification approaches we’ll look at don’t “learn from their mistakes”
- But still learn (a model) from data

(1). **Holdout**

- randomly divide data set into a training and test set
- lots of options, e.g., partition evenly, $\frac{2}{3}$ to $\frac{1}{3}$ (2:1) training to test set, etc.
- uses random selection without replacement ... instances selected once

Q: How can we randomly select $n$ instances w/out replacement from a data table?

- easiest approach is to copy the table
- then randomly select a row and remove it from the table copy
- repeat $n$ times

(2). **Random Subsampling**

- repeat the holdout method $k$ times
- performance estimate is the average of the performance of each iteration
(3). \textit{k-fold cross-validation}

- initial dataset partitioned into \( k \) subsets ("folds") \( D_1, D_2, \cdots , D_k \)
- each fold is approximately the same size
- training and testing is performed \( k \) times:
  - in iteration \( i \), \( D_i \) is used as the test set
  - and \( D_1 \cup \cdots \cup D_{i-1} \cup D_{i+1} \cup \cdots \cup D_k \) used as training set
- note each subset is used \underline{exactly once} for testing

Two ways to do the performance estimate:

\textit{i.} “holistically” over test sets ... e.g., for accuracy:
- total number of correct classifications over \( k \) iterations
- divided by number of rows in the dataset

\textit{ii.} average the performance results across each test set

(4). \textbf{Leave-one-out}

- special case of cross-validation where \( k \) is the number of instances
- thus, each \( D_i \) contains a single instance
(5). **Stratified Cross-Validation**

- class distribution within folds is approximately the same as in the initial data
- helps to prevent overfitting, assuming initial dataset has realistic distribution
- where overfitting here implies an inaccurate estimate of performance

Q: How might you go about generating stratified folds for cross validation?

One approach:

- partition dataset so each subset contains rows of a specific class
  - e.g., if class label is “yes” or “no”
  - then one partition has all “yes” rows
  - and the other all “no” rows
- generate folds by:
  - randomly selecting an instance from each partition
  - e.g., $D_1$ gets a random “yes” and “no” instance, then $D_2$, and so on
(6). **Bootstrap**

- like random subsampling but with replacement
- usually used for small datasets
- the basic "0.632" approach:
  - Given a dataset with $D$ rows
  - Randomly select $D$ rows with replacement (i.e., might select same row)
  - this gives a "bootstrap sample" (training set) of $D$ rows
  - the remaining rows (not selected) form the test set
- the sampling procedure is repeated $k$ times
  - each iteration uses the test set for a performance estimate
- note:
  - on average, 63.2% of original rows will end up in the training set
  - and 36.8% will end up in the test set
- why these percentages?
  - each row has a $1/D$ chance of being selected
  - each row has a $(1 - 1/D)$ chance of not being selected
  - we select $D$ times, so probability a row not chosen at all is $(1 - 1/D)^D$
  - for large $D$, the probability approaches $e^{-1} = 0.368$ (for $e = 2.718 \ldots$)