Validation

Hongyu Li

School of Software Engineering
TongJi University
Fall, 2014
Today's topic

- Resampling methods
  - Holdout
  - Cross Validation
    - Random Subsampling
    - K-Fold Cross-Validation
    - Leave-one-out
  - The Bootstrap
    - Bias and variance estimation
- Three-way data partitioning
**Introduction**

- Almost invariably, all the pattern recognition techniques that we have introduced have one or more free parameters
  - The number of neighbors in a kNN classification rule (or the kNN density estimation method)
  - The bandwidth of the kernel function in kernel density estimation
  - The network size, learning parameters and weights in Multilayer Perceptrons

- Two questions arise at this point
  - How do we select the “optimal” parameter(s) for a given classification problem?
  - Once we have chosen a model, how do we estimate its true error rate?
    - The true error rate is the classifier’s error rate when tested on the ENTIRE POPULATION

- If we had access to an unlimited number of examples these questions have a straightforward answer
  - Choose the model that provides the lowest error rate on the entire population
    - And, of course, that error rate is the true error rate

- In real applications we only have access to a finite set of examples, usually smaller than we wanted
  - One approach is to use the entire training data to select our classifier and estimate the error rate
    - This naïve approach has two fundamental problems
      - The final model will normally overfit the training data: it will not be able to generalize to new data
        - The problem of overfitting is more pronounced with models that have a large number of parameters
      - The error rate estimate will be overly optimistic (lower than the true error rate)
        - In fact, it is not uncommon to have 100% correct classification on training data
  - A much better approach is to split the training data into disjoint subsets: the holdout method
The **holdout method** (1)

- **Split dataset into two groups**
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier

- A typical application the holdout method is determining a stopping point for the back propagation error
The holdout method (2)

- The holdout method has two basic drawbacks
  - In problems where we have a sparse dataset we may not be able to afford the “luxury” of setting aside a portion of the dataset for testing
  - Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an “unfortunate” split

- The limitations of the holdout can be overcome with a family of resampling methods at the expense of higher computational cost
  - Cross Validation
    - Random Subsampling
    - K-Fold Cross-Validation
    - Leave-one-out Cross-Validation
  - Bootstrap
Random Subsampling

- **Random Subsampling performs K data splits of the entire dataset**
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate $E_i$ with the test examples

  ![Diagram of Random Subsampling](image)

  - **The true error estimate is obtained as the average of the separate estimates $E_i$**
    - This estimate is significantly better than the holdout estimate

  $E = \frac{1}{K} \sum_{i=1}^{K} E_i$
**K-Fold Cross-validation**

- **Create a K-fold partition of the dataset**
  - For each of K experiments, use K-1 folds for training and a different fold for testing
  - This procedure is illustrated in the following diagram for K=4

![K-Fold Cross-validation diagram](image)

- **K-Fold Cross validation is similar to Random Subsampling**
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing

- **As before, the true error is estimated as the average error rate on test examples**

\[
E = \frac{1}{K} \sum_{i=1}^{K} E_i
\]
**Leave-one-out Cross Validation**

- Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples
  - For a dataset with N examples, perform N experiments
  - For each experiment use N-1 examples for training and the remaining example for testing

As usual, the true error is estimated as the average error rate on test examples

\[ E = \frac{1}{N} \sum_{i=1}^{N} E_i \]
**How many folds are needed?**

- **With a large number of folds**
  + The bias of the true error rate estimator will be small (the estimator will be very accurate)
  - The variance of the true error rate estimator will be large
  - The computational time will be very large as well (many experiments)

- **With a small number of folds**
  + The number of experiments and, therefore, computation time are reduced
  + The variance of the estimator will be small
  - The bias of the estimator will be large (conservative or smaller than the true error rate)

- **In practice, the choice of the number of folds depends on the size of the dataset**
  - For large datasets, even 3-Fold Cross Validation will be quite accurate
  - For very sparse datasets, we may have to use leave-one-out in order to train on as many examples as possible

- **A common choice for K-Fold Cross Validation is K=10**
The bootstrap (1)

- **The bootstrap is a resampling technique with replacement**
  - From a dataset with N examples
    - Randomly select (with replacement) N examples and use this set for training
    - The remaining examples that were not selected for training are used for testing
      - This value is likely to change from fold to fold
  - Repeat this process for a specified number of folds (K)

<table>
<thead>
<tr>
<th>Complete dataset</th>
<th>X₁</th>
<th>X₂</th>
<th>X₃</th>
<th>X₄</th>
<th>X₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>X₃</td>
<td>X₃</td>
<td>X₃</td>
<td>X₅</td>
<td>X₂</td>
</tr>
<tr>
<td>Experiment 2</td>
<td>X₅</td>
<td>X₃</td>
<td>X₁</td>
<td>X₂</td>
<td>X₄</td>
</tr>
<tr>
<td>Experiment 3</td>
<td>X₅</td>
<td>X₃</td>
<td>X₂</td>
<td>X₁</td>
<td>X₃</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiment K</td>
<td>X₄</td>
<td>X₄</td>
<td>X₄</td>
<td>X₁</td>
<td>X₂</td>
</tr>
</tbody>
</table>

Training sets Validation sets

- As usual, the true error is estimated as the average error rate on test examples

\[
E = \frac{1}{K} \sum_{i=1}^{K} E_i
\]
**The bootstrap (2)**

- Compared to basic cross-validation, the bootstrap increases the variance that can occur in each fold [Efron and Tibshirani, 1993]
  - This is a desirable property since it is a more realistic simulation of the real-life experiment from which we obtained our dataset.

- Consider a classification problem with $C$ classes, a total of $N$ examples and $N_i$ examples for each class $\omega_i$
  - The a priori probability of choosing an example from class $\omega_i$ is $N_i/N$
    - Once we choose an example from class $\omega_i$, if we do not replace it for the next selection, then the a priori probabilities will have changed since the probability of choosing an example from class $\omega_i$ will now be $(N_i-1)/N$
  - Sampling with replacement preserves the a priori probabilities of the classes throughout the random selection process.

- An additional benefit of the bootstrap is its ability to obtain accurate measures of BOTH the bias and variance of the true error estimate.