LECTURE 13: Cross-validation

- **Resampling methods**
  - Cross Validation
  - Bootstrap
- **Bias and variance estimation with the Bootstrap**
- **Three-way data partitioning**

Introduction (1)

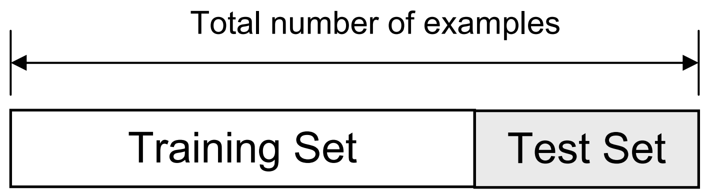
- **Almost invariably, all the pattern recognition techniques that we have introduced have one or more free parameters**
  - The number of neighbors in a kNN Classification Rule
  - The bandwidth of the kernel function in Kernel Density Estimation
  - The number of features to preserve in a Subset Selection problem
- **Two issues arise at this point**
  - **Model Selection**
    - How do we select the “optimal” parameter(s) for a given classification problem?
  - **Validation**
    - 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 would 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
- **However, in real applications only a finite set of examples is available**
  - This number is usually smaller than we would hope for!
  - Why? Data collection is a very expensive process

Introduction (2)

- **One may be tempted to use the entire training data to select the "optimal" classifier, *then* 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
- **The techniques presented in this lecture will allow you to make the best use of your (limited) data for**
  - Training
  - Model selection and
  - Performance estimation

The holdout method

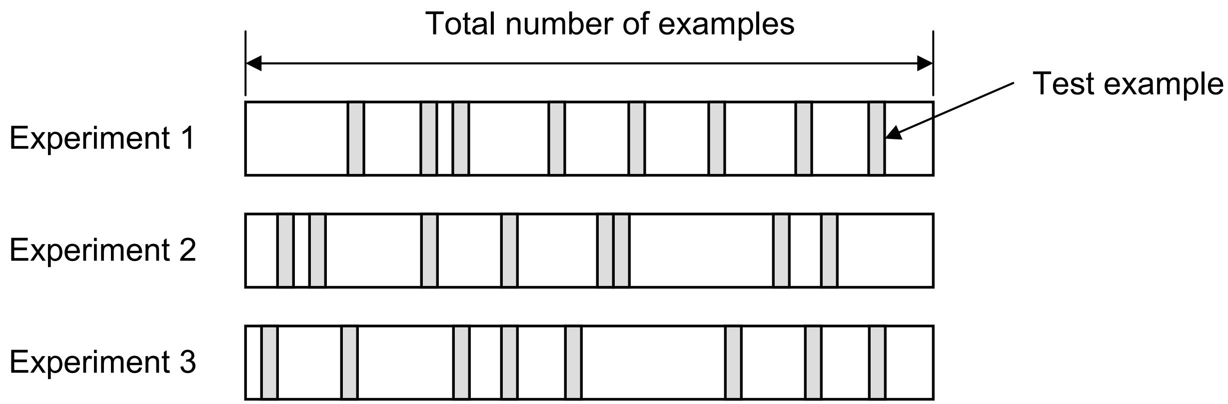
- **Split dataset into two groups**
  - **Training set**: used to train the classifier
  - **Test set**: used to estimate the error rate of the trained classifier



- **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 re-sampling 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

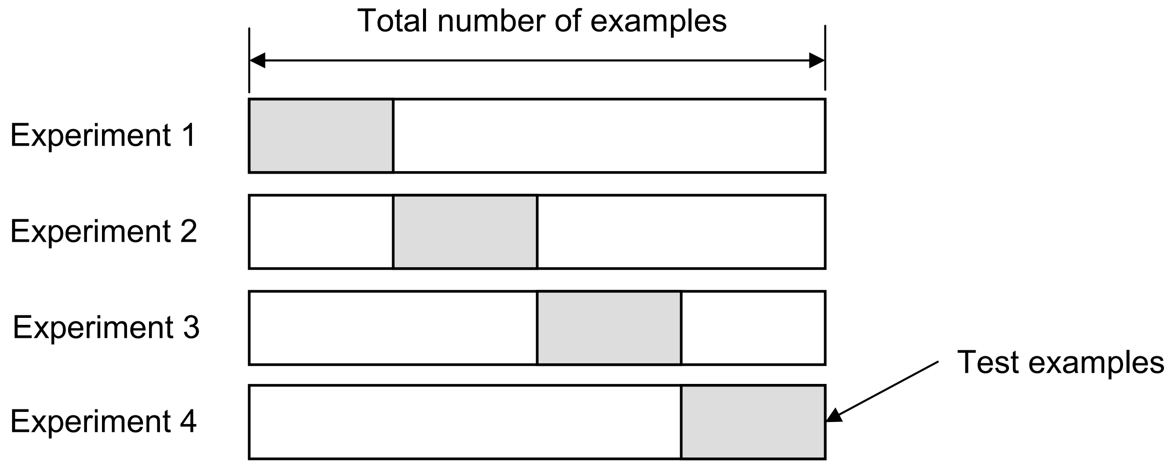


- **The true error estimate is obtained as the average of the separate estimates E_i**
  - This estimate is significantly better than the holdout estimate

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

K-Fold Cross-validation

- **Create a K-fold partition of the 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 figure for K=4

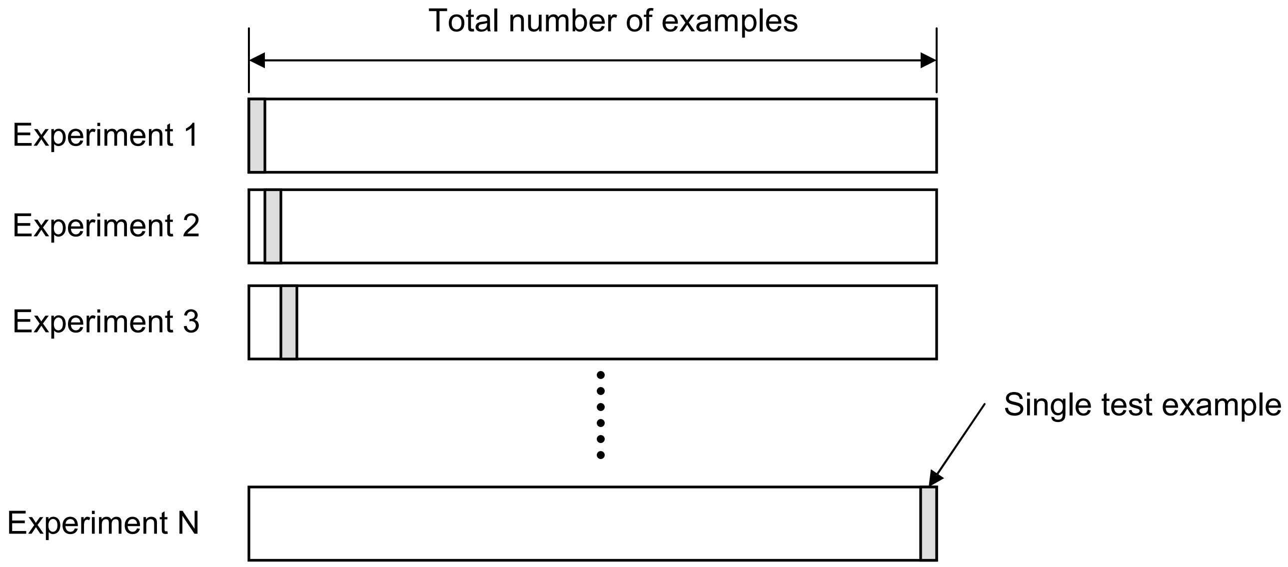


- **K-Fold Cross validation is similar to Random Subsampling**
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing
- **As before, the true error is estimated as the average error rate on test examples**

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

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**

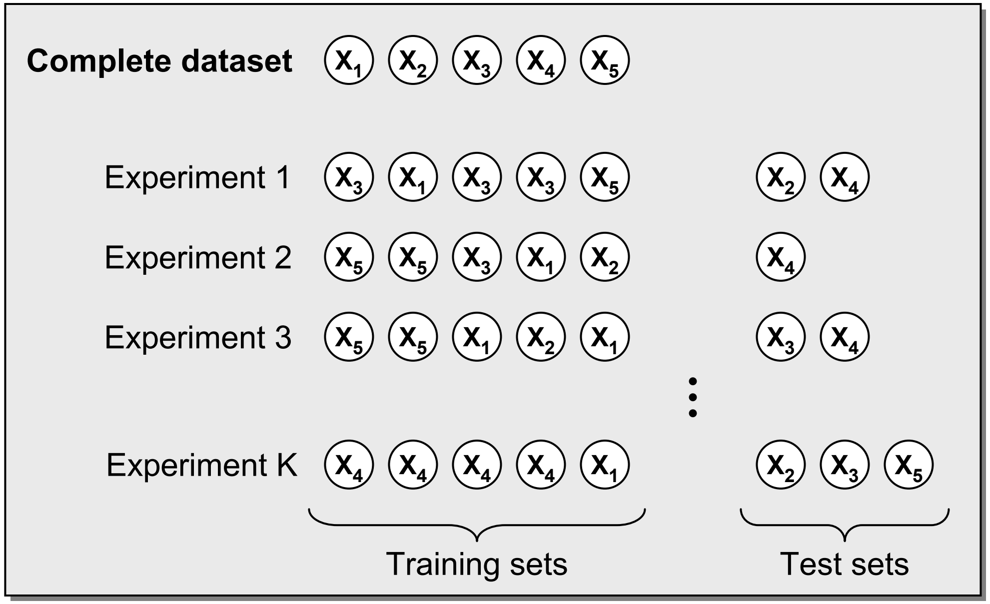
$$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

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



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 our dataset was obtained
- **Consider a classification problem with C classes, a total of N examples and N_i examples for each class ω_i**
  - The a priori probability of choosing an example from class ω_i is N_i/N
    - Once we choose an example from class ω_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 ω_i will now be $(N_i-1)/N$
  - Thus, 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**

Bias and variance of a statistical estimate

- **Consider the problem of estimating a parameter α of an unknown distribution G**
  - To emphasize the fact that α concerns G we will refer to it as $\alpha(G)$
- **We collect N examples $X=\{x_1, x_2, \ldots, x_N\}$ from the distribution G**
  - These examples define a discrete distribution G' with mass 1/N at each of the examples
  - We compute the statistic $\alpha'=\alpha(G')$ as an estimator of $\alpha(G)$
    - In the context of this lecture, $\alpha(G')$ is the estimate of the true error rate for our classifier
- **How good is this estimator?**
- **The "goodness" of a statistical estimator is measured by**
  - BIAS: How much it deviates from the true value

$$\boxed{\text{Bias} = E_G[\alpha'(G)] - \alpha(G)} \qquad \text{where } E_G[X] = \int_{-\infty}^{+\infty} x\,g(x)\,dx$$

  - VARIANCE: How much variability it shows for different samples $X=\{x_1, x_2, \ldots, x_N\}$ of the population G

$$\boxed{\text{Var} = E_G\left[(\alpha' - E_G[\alpha'])^2\right]}$$

- **Example: If we try to estimate the mean of the population with the sample mean**
  - The bias of the sample mean is know to be ZERO
  - From elementary statistics, the standard deviation of the sample mean is equal to

$$\text{std}(\bar{x}) = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

    - This term is also known in statistics as the STANDARD ERROR
  - Unfortunately, there is no such a neat algebraic formula for almost any estimate other than the sample mean

Bias and variance estimates with the bootstrap

- **The bootstrap, with its elegant simplicity, allows us to estimate bias and variance for practically any statistical estimate, be it a scalar or vector (matrix)**
  - Here we will only describe the estimation procedure
    - If you are interested in more details, the textbook "*Advanced algorithms for neural networks*" [Masters, 1995] has an excellent introduction to the bootstrap
- **The bootstrap estimate of bias and variance**
  - Consider a dataset of N examples $X=\{x_1, x_2, \ldots, x_N\}$ from the distribution G
    - This dataset defines a discrete distribution G'
  - Compute $\alpha'=\alpha(G')$ as our initial estimate of $\alpha(G)$
  - Let $\{x_1^*, x_2^*, \ldots, x_N^*\}$ be a bootstrap dataset drawn from $X=\{x_1, x_2, \ldots, x_N\}$
    - Estimate the parameter α using this bootstrap dataset $\alpha^*(G^*)$
  - Generate K bootstrap datasets and obtain K estimates $\{\alpha^{*1}(G^*), \alpha^{*2}(G^*), \ldots, \alpha^{*K}(G^*)\}$
  - **The rationale in the bootstrap method is that the effect of generating a bootstrap dataset from the distribution G' is similar to the effect of obtaining the dataset $X=\{x_1, x_2, \ldots, x_N\}$ from the original distribution G**
    - In other words, the distribution $\{\alpha^{*1}(G^*), \alpha^{*2}(G^*), \ldots, \alpha^{*K}(G^*)\}$ is related to the initial estimate α' in the same fashion as multiple estimates α' are related to the true value α, so the bias and variance estimates of α' are:

$$\mathrm{Bias}(\alpha') = \left[\alpha^{*\bullet} - \alpha'\right] \text{ where } \alpha^{*\bullet} = \left(\frac{1}{K}\sum_{i=1}^{K}\alpha^{*i}\right)$$

$$\mathrm{Var}(\alpha') = \frac{1}{K-1}\sum_{i=1}^{K}\left(\alpha^{*i} - \alpha^{*\bullet}\right)^2$$

Example: estimating bias and variance

- **Assume a small dataset x={3,5,2,1,7}**
  - We want to compute the bias and variance of the sample mean $\alpha'=3.6$
- **We generate a number of bootstrap samples (three in this case)**
  - Assume that the first bootstrap yields the dataset {7,3,2,3,1}
    - We compute the sample mean $\alpha^{*1}=3.2$
  - The second bootstrap sample yields the dataset {5,1,1,3,7}
    - We compute the sample mean $\alpha^{*2}=3.4$
  - The third bootstrap sample yields the dataset {2,2,7,1,3}
    - We compute the sample mean $\alpha^{*3}=3.0$
  - We average these estimates and obtain an average of $\alpha^{*\bullet}=3.2$
- **What are the bias and variance of the sample mean α'**
  - **Bias(α') = 3.2 - 3.6 = -0.4**
    - Therefore, we conclude that the re-sampling process introduces a downward bias on the mean, so we would be inclined to use 3.6 + 0.4 = 4.0 as an unbiased estimate of α
  - **Variance(α') = $\frac{1}{2}$*[(3.2-3.2)^2+(3.4-3.2)^2+(3.0-3.2)^2] = 0.04**
- **NOTES**
  - We have done this exercise for the sample mean (so you could trace the computations), but α could be any other statistical operator. Here lies the real power of this procedure!!
  - How many bootstrap samples should we use? As a rule of thumb, several hundred re-samples will be sufficient for most problems

Adapted from [Masters,1995]

Three-way data splits (1)

- **If model selection *and* true error estimates are to be computed simultaneously, the data needs to be divided into three disjoint sets [Ripley, 1996]**
  - **Training set**: a set of examples used for learning: to fit the parameters of the classifier
    - In the MLP case, we would use the training set to find the "optimal" weights with the back-prop rule
  - **Validation set**: a set of examples used to tune the parameters of a classifier
    - In the MLP case, we would use the validation set to find the "optimal" number of hidden units or determine a stopping point for the back-propagation algorithm
  - **Test set**: a set of examples used <u>only</u> to assess the performance of a fully-trained classifier
    - In the MLP case, we would use the test to estimate the error rate after we have chosen the final model (MLP size and actual weights)
    - <u>After assessing the final model on the test set, YOU MUST NOT tune the model any further!</u>
- **Why separate test and validation sets?**
  - The error rate estimate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model
  - <u>After assessing the final model on the test set, YOU MUST NOT tune the model any further!</u>
- **Procedure outline**

1. Divide the available data into training, validation and test set
2. Select architecture and training parameters
3. Train the model using the training set
4. Evaluate the model using the validation set
5. Repeat steps 2 through 4 using different architectures and training parameters
6. Select the best model and train it using data from the training and validation sets
7. Assess this final model using the test set

  - This outline assumes a holdout method
    - If Cross-Validation or Bootstrap are used, steps 3 and 4 have to be repeated for each of the K folds

Three-way data splits (2)