Read: ESL 7.4, 7.10
Training set, Validation set, Test Set
Hold-out method
K-fold Cross Validation
Model-based Selection Criteria
Given data \((x_i, y_i), i = 1, \cdots, n\), where \(x_i\) is the input and \(y_i\) is the response, we fit the model \(\hat{f}\).

Model assessment is extremely important, as it can
- measure the quality of a fitted model.
- compare different methods and make an optimal choice.
- select the best model with a model class \(\mathcal{F} = \{f_\alpha\}\), where \(\alpha\) is the model index
- choose optimal tuning parameters in regularization methods.
Two Goals:

- **Model Assessment**: after choosing a final model, estimate its prediction error (generalization performance) on new data.
- **Model Selection**: estimate the performance of different models and choose the best model

Different models:

- Is logistic \( \hat{f}_1 \) better than LDA \( \hat{f}_2 \)?

From the same model class

- Is 3-nn better than 5-nn?
- What is the best \( k \) for the model class \( \mathcal{F} = \{ \hat{f}_k, 1 \leq k \leq n \} \).
As complexity increases, the model tends to
- adapt to more complicated underlying structures (decrease in bias)
- have increased estimation error (increase in variance)

In between there is an optimal model complexity, giving the minimum test error.

Role of tuning parameter $\alpha$ ($k$ in knn, or $\lambda$ in LASSO)
- Tuning parameter $\alpha$ controls the model complexity
- Find the best $\alpha$ to produce the minimum of test error
How do we evaluate the performance of $\hat{f}$?

- its performance on the real-world data ("generalization" performance)
- we want to test $\hat{f}$ on the data that it has never seen before
  - this set of data is called the "test" set
  - the test data is an independent set of the training data

Why not training data?

- the risk of overfitting
Loss and Errors

If $Y$ is continuous, two commonly-used loss functions are

- squared error: $L(Y, \hat{f}(X)) = (Y - \hat{f}(X))^2$
- absolute error: $L(Y, \hat{f}(X)) = |Y - \hat{f}(X)|$

If $Y$ takes values $\{1, \ldots, K\}$, the common loss functions are

- 0–1 loss: $L(Y, \hat{f}(X)) = I(Y \neq \hat{f}(X))$
- log-likelihood: $L(Y, \hat{p}(X)) = -2 \sum_{k=1}^{K} I(Y = k) \log \hat{p}_k(X)$
  $= -2 \log \hat{p}_Y(X)$
Training error and Test error

- **Training error**: the average loss over the training samples

  \[
  \text{TrainErr} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i))
  \]

- **Test error**: Given a test data set \((x_i, y'_i)\) for \(i = 1, \ldots, n'\),

  \[
  \text{TestErr} = \frac{1}{n'} \sum_{i=1}^{n'} L(Y'_i, \hat{f}(X_i)).
  \]

  TestErr is an estimate of PE. The larger \(n'\), the better estimate.

- **Prediction error**: the expected error over all the input

  \[
  \text{PE} = E_{(X, Y', \text{data})} \left[ \frac{1}{n} \sum_{i=1}^{n} L(Y'_i, \hat{f}(X_i)) \right],
  \]

  where \(Y'_i = f(X_i) + \epsilon'_i, i = 1, \ldots, n\), are independent of \(y_1, \ldots, y_n\). Expectation is taken w.r.t. all random quantities.
Examples

Regression:

\[
\text{TrainErr} = \frac{1}{n} (y_i - \hat{f}(x_i))^2
\]

Classification:

- log-likelihood loss: TrainErr is the sample log-likelihood
  \[
  \text{TrainErr} = -\frac{2}{n} \sum_{i=1}^{n} \log \hat{p}_{y_i}(x_i),
  \]
  also called the \textit{cross-entropy loss} or \textit{deviance}.

- 0 – 1 loss: the error is the misclassification rate
  \[
  \text{TrainErr}_0 = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{f}(x_i)), \quad \text{TestErr} = \frac{1}{n'} \sum_{i=1}^{n'} I(y_i \neq \hat{f}(x_i)),
  \]
  TestErr is also called \textit{expected misclassification rate}.
TrainErr is not a good estimate of TestErr
- TrainErr is often quite different from TestErr.
- TrainErr can dramatically underestimate TestErr.

TrainErr is not an objective measure on the generalization performance
- we built \( \hat{f} \) based on the training data.
- TrainErr decreases with model complexity.
  - Training error drops to zero if the model is complex enough
  - A model with zero training error overfits the training data; over-fitted models typically generalize poorly
Figure 2.11: Test and training error as a function of model complexity.
How do we estimate TestErr accurately? Consider
- data-rich situation
- data-insufficient situation

Common Techniques (Efron 2004, JASA)
- Hold Out Method (data-rich situation)
- Resampling methods:
  - cross validation
  - bootstrap
- Analytical estimation criteria
**Key Idea:** Estimate the test error by
- hold out a subset of the training data from the fitting process
- apply the fitted model to those held out observations and calculate the prediction errors

![Diagram showing training and testing datasets](image-url)
Hold Out Approach: without tuning

- Randomly divide the data into two parts: a training set and a test or hold-out set.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the held-out set.
- The resulting error provides an estimate of the test error (e.g., MSE in regression, misclassification rate in classification).
Hold Out Approach: with tuning

Randomly divide the dataset into three parts:
- **training** set: to fit the models
- **tuning** (validation) set: to estimate the prediction error for model selection
- **test** set: to assess the generalization error of the final model

The typical proportions are respectively: 50%, 25%, 25%.

![Dataset division diagram](image-url)
The TestErr estimate can be highly variable, depending on which observations are included in the training set and which are held out.

Only a subset of the observations (those not held out) are used to fit the model.

The estimated error tend to overestimate the test error for the model fit on the entire data set.
Resampling Methods

These methods refit a model to samples formed from the training set:

- Cross-validation
- Bootstrap techniques

They provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates.
K-fold Cross Validation

The simplest and most popular way to estimate the test error.

1. Randomly split the data into $K$ roughly equal parts
2. For each $k = 1, \ldots, K$
   - leave the $k$th portion out, and fit the model using the other $K - 1$ parts. Denote the solution by $\hat{f}^{-k}(x)$
   - calculate the prediction error of the $\hat{f}^{-k}(x)$ on the $k$th (left-out) portion
3. Average the errors

Define the Index function (allocating memberships)

$$\kappa : \{1, \ldots, n\} \rightarrow \{1, \ldots, K\}.$$ 

Then the $K$-fold cross-validation estimate of prediction error is

$$CV = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

Typical choice: $K = 5, 10$
Choose $\alpha$

Given tuning parameter $\alpha$, the model fit $\hat{f}^{-k}(x, \alpha)$. The $CV(\alpha)$ curve is

$$CV(\alpha) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha))$$

- $CV(\alpha)$ provides an estimate of the test error curve
- We find the best parameter $\hat{\alpha}$ by minimizing $CV(\alpha)$.
- Final model is $f(x, \hat{\alpha})$. 
For regression problems, apply CV to
- shrinkage methods: select $\lambda$
- best subset selection: select the best model
- nonparametric methods: select the smoothing parameter

For classification methods, apply CV to
- kNN: select $k$
- SVM: select $\lambda$
If $K=n$, we have leave-out-one cross validation.

- $\kappa(i) = i$
- LOO-CV is approximately unbiased for the true prediction error
- LOO-CV can have high variance because the $n$ “training sets” are so similar to one another

Computation:

- Computational burden is generally considerable, requiring $n$ model fitting.
- For special models like linear smoothers, computation can be done quickly. For example: Generalized Cross Validation (GCV)
Cross-validation error is very nearly unbiased for test error.

- The slight bias is due to that the training set in cross-validation is slightly smaller than the actual data set (e.g. for LOOCV the training set size is $n - 1$ when there are $n$ observed cases).

- In nearly all situations, the effect of this bias is conservative, i.e., the estimated fit is slightly biased in the direction suggesting a poorer fit. In practice, this bias is rarely a concern.

The variance of CV-error can be large.

- In practice, to reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.
Example: Two-class classification problem

- Linear model with best subsets regression of subset size $p$

In Figure 7.9, ($p$ is tuning parameter)

- both curves picks best $\hat{p} = 10$. (The true model)
- CV curve is flat beyond 10.
- Standard error bars are the standard errors of the individual misclassification error rates
Let $CV_k(\hat{f}_\alpha)$ denote the validation errors of $\hat{f}_\alpha$ in each fold for $\alpha$, or simply $CV_k(\alpha)$. Here $\alpha \in \{\alpha_1, \ldots, \alpha_m\}$, within a set of candidate values for $\alpha$. Then we have

- The cross validation error

$$CV(\alpha) = \frac{1}{K} \sum_{k=1}^{K} CV_k(\alpha).$$

- The standard deviation of $CV(\alpha)$ is calculated as

$$SE(\alpha) = \sqrt{\text{Var}(CV(\alpha))}.$$
Let

$$\hat{\alpha} = \arg \min_{\alpha \in \{\alpha_1, \ldots, \alpha_m\}} CV(\alpha).$$

First we find $\hat{\alpha}$; then we move $\alpha$ in the direction of increasing regularization as much as can as long as

$$CV(\alpha) \leq CV(\hat{\alpha}) + SE(\hat{\alpha}).$$

- choose the most parsimonious model with error no more than one standard error above the best error.
- Idea: “All equal (up to one standard error), go for the simplest model”.

In this example, one-SE rule CV would choose $p = 9$. 
Figure 7.9: Prediction error (red) and tenfold cross-validation curve (green) estimated from a single training set, from the scenario in the bottom right panel of Figure 7.3.
Consider a simpler classifier for microarrays:

1. Starting with 5,000 genes, find the top 200 genes having the largest correlation with the class label
2. Carry about the nearest-centroid classification using top 200 genes

How do we select the tuning parameter in the classifier?

- Way 1: apply cross-validation to step 2
- Way 2: apply cross-validation to steps 1 and 2

Which is right? – Cross validating the whole procedure.
Generalized Cross Validation

If linear fitting methods $\hat{y} = Sy$ satisfy

$$y_i - \hat{f}^{-i}(x_i) = \frac{y_i - \hat{f}(x_i)}{1 - S_{ii}},$$

then we do **NOT** need to train the model $n$ times. Using the estimation $S_{ii} = \text{trace}(S)/n$, we have the approximate score

$$GCV = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(S)/n} \right]^2$$

- computational advantage
- $GCV(\alpha)$ is always smoother than $CV(\alpha)$
Covariance penalties

- $C_p$
- AIC
- BIC
- Stein’s UBR

They can be used for both *model selection* and *test error estimation*. 
Generalized Information Criteria (GIC)

Information criteria:

$$\text{GIC}(\text{model}) = -2 \cdot \log\text{lik} + \alpha \cdot \text{df},$$

the degree of freedom (df) of \( \hat{f} \) as the number of effective parameters of the model.

Examples: In linear regression settings, we use

$$\text{AIC} = n \log(\text{RSS}/n) + 2 \cdot \text{df},$$
$$\text{BIC} = n \log(\text{RSS}/n) + \log(n) \cdot \text{df}$$

where $$\text{RSS} = \sum_{i=1}^{n} [y_i - \hat{f}(x_i)]^2$$ (the residual sum of squares)
training error:
\[
e\bar{\text{err}} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i))
\]

test error:
\[
\text{Err} = E_{(X,Y,\text{data})}[L(Y, \hat{f}(X))].
\]
in-sample error rate (an estimate of prediction error)
\[
\text{Err}_{\text{in}} = \frac{1}{n} \sum_{i=1}^{n} E_{y^{\text{new}}} L(y_i^{\text{new}}, \hat{f}(x_i))
\]

Optimism
\[
\text{op} = \text{Err}_{\text{in}} - E_y e\bar{\text{err}}
\]
Op is typically positive (Why?)

For the squared error, 0-1, and other loss function,

\[ op = \frac{2}{n} \sum_{i=1}^{n} Cov(\hat{y}_i, y_i). \]

- The amount by which err under-estimates the true error depends on the strength of correlation \( y_i \) and its production.