Resampling Methods: Cross Validation

Section 5.1

Cathy Poliak, Ph.D.
cpoliak@central.uh.edu

Department of Mathematics
University of Houston

Resampling Methods

- **Resampling methods** involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model.
- Could potentially be computationally expensive, because they involve fitting the same statistical method multiple times using different subsets of the training data.
- Two most commonly used resampling methods:
  - Cross-validation - can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance.
  - Bootstrap - can be used to provide a measure of accuracy of a parameter estimate or of a given statistical learning method.

Recall

Recall the models:

- Linear Regression
- Logistic Regression

Model Validation:

- Training error
- Test error

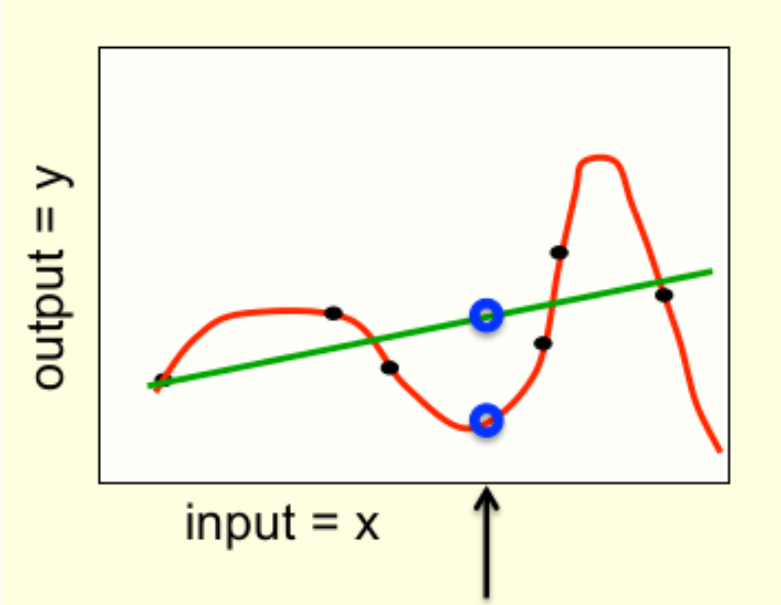
In this lecture - we delve deep into test error estimation.

Training error, Overfitting

- Training error may not be a good metric of model performance

Training error, Overfitting

- Training error may not be a good metric of model performance



- Sometimes good training test performance is more indicative of **overfitting**
- This results in fitting the **noise** instead of **true signal**

Validation Set Approach

- We may not always be able to get a random *test* set thus a **validation set** (or **hold-out set**) is used.
  1. Randomly divide data set into two parts:
     - ★ Training set
     - ★ Validation set
  2. Fit the model on the training data, and use the fitted model to predict responses for validation data.
  3. Note: The validation set error rate $\approx$ test error rate.

Validation Set Approach. *Auto* Dataset

Example. Recall studying the relationship between *mpg* and *horsepower* in *Auto* data set. It appeared to be non-linear, but unclear whether a quadratic or cubic regression would provide best model.

Let's try comparing

- **linear** regression,
- **quadratic** regression &
- **cubic** regression

models via validation set approach:

Validation Set Approach. *Auto* Dataset

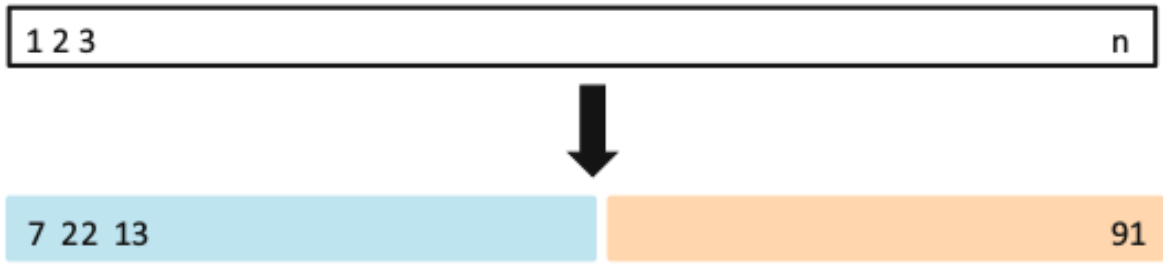
Example. Recall studying the relationship between *mpg* and *horsepower* in *Auto* data set. It appeared to be non-linear, but unclear whether a quadratic or cubic regression would provide best model.

Let's try comparing

- **linear** regression,
- **quadratic** regression &
- **cubic** regression

models via validation set approach:

1. Randomly subdivide data into training and testing subset.



   We split 392 total observations evenly - 196 for training, 196 for testing.

Validation set *Auto* dataset

2. We train the following models (*hpwr* for *horsepower*)

$$\text{linear} : mpg = \beta_0 + \beta_1 hpwr + \epsilon$$
$$\text{quadratic} : mpg = \beta_0 + \beta_1 hpwr + \beta_2 hpwr^2 + \epsilon$$
$$\text{cubic} : mpg = \beta_0 + \beta_1 hpwr + \beta_2 hpwr^2 + \beta_3 hpwr^3 + \epsilon$$

Validation set *Auto* dataset

2. We train the following models (*hpwr* for *horsepower*)

$$\text{linear} : mpg = \beta_0 + \beta_1 hpwr + \epsilon$$
$$\text{quadratic} : mpg = \beta_0 + \beta_1 hpwr + \beta_2 hpwr^2 + \epsilon$$
$$\text{cubic} : mpg = \beta_0 + \beta_1 hpwr + \beta_2 hpwr^2 + \beta_3 hpwr^3 + \epsilon$$

3. For each model, we calculate the Mean Squared Error (MSE)

$$MSE = \frac{1}{196} \sum_{i=1}^{196} (mpg_i - \widehat{mpg}_i)^2$$

on the validation set.

After running those steps 1, 2 & 3 in *R* with *set.seed*(10):

	Linear	Quadratic	Cubic
MSE	26.435	19.807	20.266

R Code

```
library(ISLR)
set.seed(10)
sample = sample(1:392,196)
train = Auto[sample,]
test = Auto[-sample,]
auto.lm = lm(mpg ~ horsepower, data = train)
auto.lm2 = lm(mpg ~ poly(horsepower,2), data = train)
auto.lm2 = lm(mpg ~ poly(horsepower,2), data = train)
auto.lm3 = lm(mpg ~ poly(horsepower,3), dat = train)

auto.pred = predict(auto.lm, newdata = test,se.fit = TRUE)
auto.pred2 = predict(auto.lm2, newdata = test, se.fit = TRUE)
auto.pred3 = predict(auto.lm3, newdata = test, se.fit = TRUE)

mean((Auto$mpg - predict(auto.lm,Auto))[-sample]^2)
mean((Auto$mpg - predict(auto.lm2,Auto))[-sample]^2)
mean((Auto$mpg - predict(auto.lm3,Auto))[-sample]^2)
```

Lab Question

1. In R repeat the code with set.seed(10). Do you get the same results?
   a) Yes
   b) No

2. Write in the chat box what you get. Is your different from others in the class?
   a) Yes
   b) No

Validation Set Approach: Drawbacks

Validation set approach has two big drawbacks:

1. **Inconsistency** (or split-to-split variability) of error estimates. Below are the results of running validation set approach for 10 different subdivisions of data into training and validation set.

	Linear	Quadratic	Cubic
1	26.14142	19.82259	19.78252
2	21.76211	16.36646	16.46181
3	22.50892	17.95000	18.16543
4	22.91684	18.33850	18.47677
5	24.49521	19.20276	19.25468
6	23.08818	18.37858	18.70315
7	21.97469	19.31556	19.51172
8	22.62499	17.27001	17.29571
9	22.86403	20.39818	20.95095
10	24.99476	19.24862	19.20272

2. If we use less data for the training data, this results in a worse performance which will overestimate the MSE.

Leave-One-Out Cross-Validation (LOOCV)

In LOOCV, for **each data point** i, $i = 1, \ldots, n$, we

1. Split data into two subsets:
   - Training set: $(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)$
   - Test "set" of just one observation: (x_i, y_i).

Leave-One-Out Cross-Validation (LOOCV)

In LOOCV, for **each data point** i, $i = 1, \ldots, n$, we

1. Split data into two subsets:
   - Training set: $(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)$
   - Test "set" of just one observation: (x_i, y_i).

2. Use training set to fit the model and produce prediction $\hat{y}_i$.

Leave-One-Out Cross-Validation (LOOCV)

In LOOCV, for **each data point** i, $i = 1, \ldots, n$, we

1. Split data into two subsets:
   - Training set: $(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)$
   - Test "set" of just one observation: (x_i, y_i).
2. Use training set to fit the model and produce prediction $\hat{y}_i$.
3. Calculate $MSE_i = (y_i - \hat{y}_i)^2$

The **LOOCV estimate** for test (squared) error is

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$

Leave-One-Out Cross-Validation (LOOCV)

Illustration of data subdivision for LOOCV as opposed to validation set approach (where training and test set were of comparable sizes).

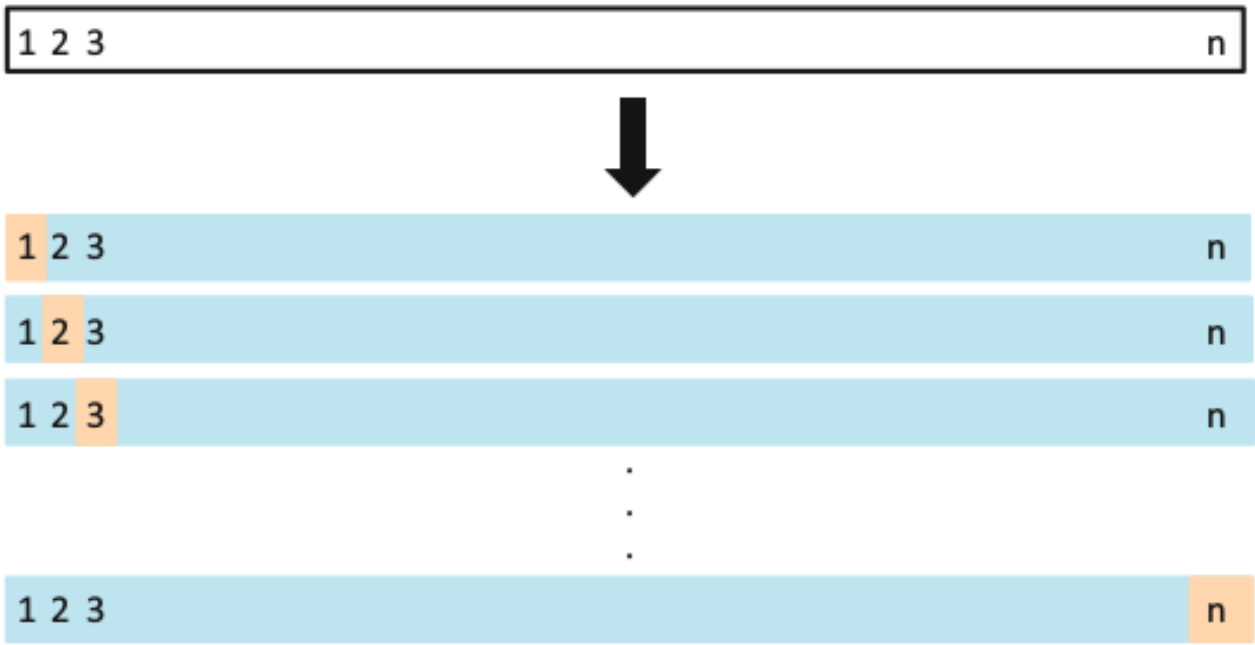


FIGURE 5.3. *A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.*

```
> #Leave-One-Out Cross Validation
> mse.loocv = matrix(nrow = nrow(Auto), ncol = 3)
> for (i in 1:nrow(Auto)) {
+   sample = i
+   #Creating the models
+   auto.lm = lm(mpg ~ horsepower, data = Auto[-sample,])
+   auto.lm2 = lm(mpg ~ poly(horsepower,2),data = Auto[-sample,])
+   auto.lm3 = lm(mpg ~ poly(horsepower,3),data = Auto[-sample,])
+
+   #Getting the MSE for each model
+   mse.loocv[i,1] = (Auto$mpg[sample] - predict(auto.lm,Auto[sample,]))^2
+   mse.loocv[i,2] = (Auto$mpg[sample] - predict(auto.lm2,Auto[sample,]))^2
+   mse.loocv[i,3] = (Auto$mpg[sample] - predict(auto.lm3,Auto[sample,]))^2
+ }
> colMeans(mse.loocv)
[1] 24.23151 19.24821 19.33498
```

Linear Quad Cubic

Leave-One-Out Cross-Validation: Pros & Cons.

Advantages

- Always yields the same result. E.g., for *Auto* **data example**, LOOCV test error estimates are:

```
  Linear Quadratic     Cubic
24.23151  19.24821  19.33498
```

- Nearly the whole data set is used at each step (more data $\Longrightarrow$ less bias in test error estimate)

Leave-One-Out Cross-Validation: Pros & Cons.

Advantages

- Always yields the same result. E.g., for *Auto* **data example**, LOOCV test error estimates are:

```
  Linear Quadratic     Cubic
24.23151  19.24821  19.33498
```

- Nearly the whole data set is used at each step (more data $\Longrightarrow$ less bias in test error estimate)

Disadvantages

- computationally demanding
- bias-variance trade-off

Leave-One-Out Cross-Validation: Pros & Cons.

Advantages

- Always yields the same result. E.g., for *Auto* **data example**, LOOCV test error estimates are:

```
  Linear Quadratic     Cubic
24.23151  19.24821  19.33498
```

- Nearly the whole data set is used at each step (more data $\Longrightarrow$ less bias in test error estimate)

Disadvantages

- computationally demanding
- bias-variance trade-off

Both **validation set** and **LOOCV** are very general methods, and can be used with any predictive model.

LOOCV in R

- Since this method is very heavy computationally there is a function in `R` that can create the **MSE**, called cv.glm.
- Requires the package `boot`.
- Requires the models to be created using the function glm().

Information about `cv.glm`

Value
The returned value is a list with the following components

call
The original call to cv.glm.

K
The value of K used for the K-fold cross validation.

delta
A vector of length two. The first component is the raw cross-validation estimate of prediction error. The second component is the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation.

seed
The value of .Random.seed when cv.glm was called.

```
> #LOOCV
> #install.packages("boot")
> library(boot)
> auto.glm = glm(mpg ~ horsepower, data = Auto)
> cv.glm(Auto,auto.glm)$delta
[1] 24.23151 24.23114
```

R Code

We can repeat this for multiple degrees for a polynomial regression.

```
> #LOOCV for Multiple Polynomials
> #For up to degreee 5
> cv.error = rep(0,5)
> for (i  in 1:5) {
+   glm.fit = glm(mpg ~ poly(horsepower,i),data = Auto)
+   cv.error[i] = cv.glm(Auto,glm.fit)$delta[1]
+ }
> cv.error
[1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

Lab Question

3. You run the code for the Leave-One-Out Cross Validation. Do you get the same results as in the previous slide?
   a) Yes
   b) No

4. Where is the highest difference in the MSE?
   a) Between 1st degree and 2nd degree
   b) Between 2nd degree and 3rd degree
   c) Between 3rd degree and 4th degree
   d) Between 4th degree and 5th degree

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.
2. The first **fold** is treated as a validation set.

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.
2. The first **fold** is treated as a validation set.
3. The method is fit on the remaining $k - 1$ folds.

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.
2. The first **fold** is treated as a validation set.
3. The method is fit on the remaining $k - 1$ folds.
4. The mean squared error, MSE_1 is then computed on the observations in the held-out fold.

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.
2. The first **fold** is treated as a validation set.
3. The method is fit on the remaining $k - 1$ folds.
4. The mean squared error, MSE_1 is then computed on the observations in the held-out fold.
5. This procedure is repeated k times; each time a different group of observations is treated as a validation set. Which results in k estimates of the test error, $\text{MSE}_1, \text{MSE}_2, \ldots, \text{MSE}_k$

K-fold Cross-Validation

K-fold Cross-Validation is an alternative to LOOCV where

1. Data is randomly divided into K **subsets** or *folds* of approximately equal size, n_K.
2. The first **fold** is treated as a validation set.
3. The method is fit on the remaining $k - 1$ folds.
4. The mean squared error, MSE_1 is then computed on the observations in the held-out fold.
5. This procedure is repeated k times; each time a different group of observations is treated as a validation set. Which results in k estimates of the test error, $\text{MSE}_1, \text{MSE}_2, \ldots, \text{MSE}_k$
6. The K-fold CV (squared) test error estimate is

$$CV_{(k)} = \frac{1}{K} \sum_{j=1}^{K} MSE_j$$

Lab Question

5. For what K does K-fold CV become a leave-one-out CV?
   a) 1
   b) 5
   c) 10
   d) Unlimited values (k = n, number of observations in the data).

K-fold Cross-Validation.

Illustration of random data subdivision into training and testing subsets for 5-fold CV (as opposed to LOOCV and validation set approaches):

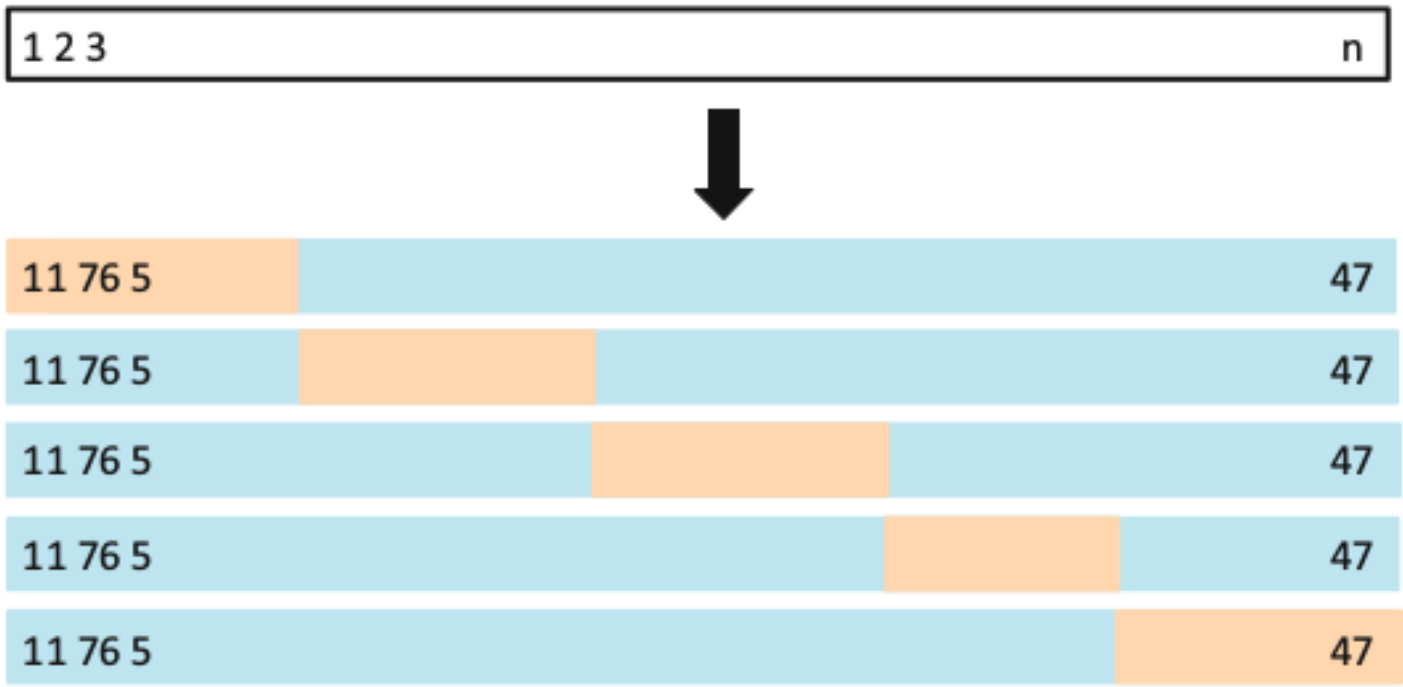


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

R Code

```
> #K-fold Cross Validataion
> #Use K = 10
> set.seed(3)
> cv.error.10 = rep(0,5)
> cv.error = rep(0,5)
> for (i  in 1:5) {
+   glm.fit = glm(mpg ~ poly(horsepower,i),data = Auto)
+   cv.error.10[i] = cv.glm(Auto,glm.fit, K=10)$delta[1]
+ }
> cv.error.10
[1] 24.11129 19.21875 19.18225 19.49981 18.94992
```

K-fold Cross-Validation: Pros & Cons

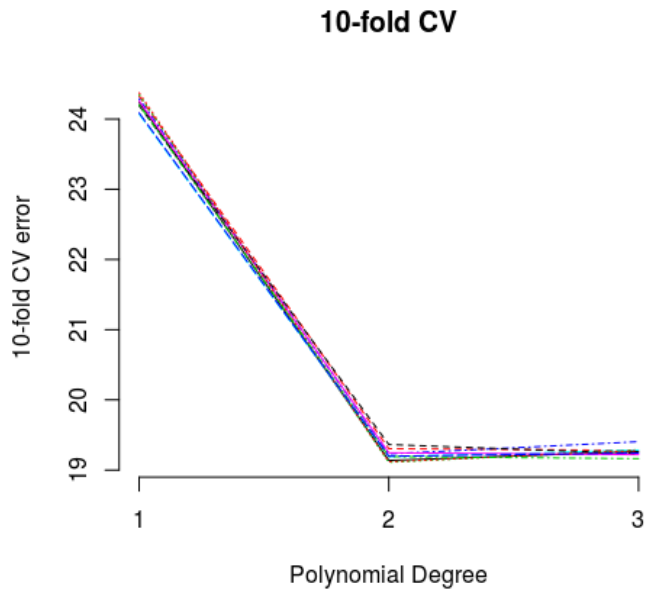
Advantages to K-fold CV over LOOCV:

- Computational: Doing *LOOCV* ($\equiv K$-fold CV for $K = n$) is tough for computationally intensive models, as opposed to 5- or 10-fold CV, especially for large n.
- The model is fit only $K << n$ times.
- K-fold CV doesn't lose in estimation quality to *LOOCV*.

K-fold Cross-Validation: Pros & Cons

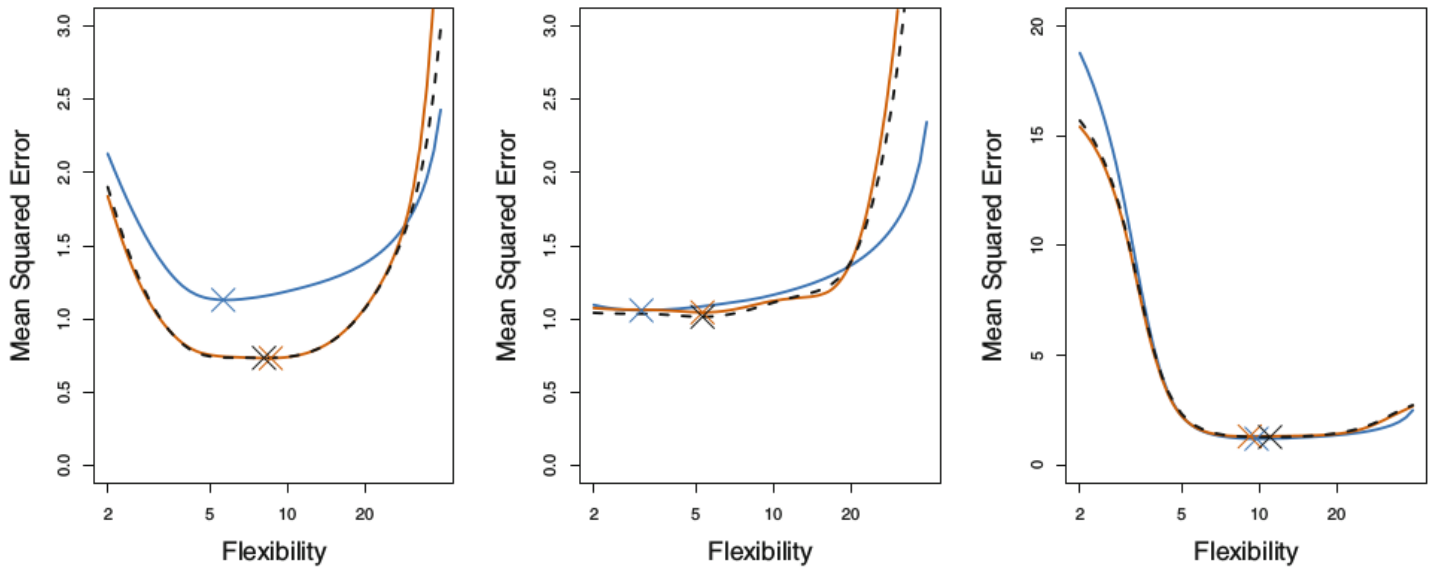
Advantages to K-fold CV over LOOCV:

- Computational: Doing *LOOCV* ($\equiv$ K-fold CV for $K = n$) is tough for computationally intensive models, as opposed to 5- or 10-fold CV, especially for large n.
- The model is fit only $K << n$ times.
- K-fold CV doesn't lose in estimation quality to *LOOCV*.
- The variability in K-fold error estimates is negligible. Below are the 10-fold CV results for polynomial fits to *Auto* data:



k-fold Cross-Validation: Pros & Cons

And it doesn't lose in estimate efficiency (show the plots of how close the LOOCV and *k*-fold estimates are for simulated data set).



Bias-Variance Tradeoff

K-fold CV also often gives more accurate estimates of the test error rate than LOOCV. This has to do with a **bias-variance trade-off**:

- *LOOCV* estimates model's test error with less bias, as it uses $\approx$ all observations ($n-1$ out of n) to obtain the estimate at each run.

Bias-Variance Tradeoff

K-fold CV also often gives more accurate estimates of the test error rate than LOOCV. This has to do with a **bias-variance trade-off**:

- *LOOCV* estimates model's test error with less bias, as it uses $\approx$ all observations ($n-1$ out of n) to obtain the estimate at each run.
- Nonetheless, the sample-to-sample variation of *LOOCV* is higher than that for K-fold CV

Bias-Variance Tradeoff

K-fold CV also often gives more accurate estimates of the test error rate than LOOCV. This has to do with a **bias-variance trade-off**:

- *LOOCV* estimates model's test error with less bias, as it uses $\approx$ all observations ($n-1$ out of n) to obtain the estimate at each run.
- Nonetheless, the sample-to-sample variation of *LOOCV* is higher than that for K-fold CV
- That is if we were to use a different sample from the population, then LOOCV test error estimate would (on average) change more drastically compared to K-fold CV.

  Why?

Bias-Variance Tradeoff

K-fold CV also often gives more accurate estimates of the test error rate than LOOCV. This has to do with a **bias-variance trade-off**:

- *LOOCV* estimates model's test error with less bias, as it uses $\approx$ all observations ($n-1$ out of n) to obtain the estimate at each run.
- Nonetheless, the sample-to-sample variation of *LOOCV* is higher than that for K-fold CV
- That is if we were to use a different sample from the population, then LOOCV test error estimate would (on average) change more drastically compared to K-fold CV.

  Why? In *LOOCV* we train n models on an almost identical set of observations $\implies$ error estimates are highly correlated and are very sample-dependent.

Bias-Variance Tradeoff

K-fold CV also often gives more accurate estimates of the test error rate than LOOCV. This has to do with a **bias-variance trade-off**:

- *LOOCV* estimates model's test error with less bias, as it uses $\approx$ all observations ($n-1$ out of n) to obtain the estimate at each run.
- Nonetheless, the sample-to-sample variation of *LOOCV* is higher than that for K-fold CV
- That is if we were to use a different sample from the population, then LOOCV test error estimate would (on average) change more drastically compared to K-fold CV.

  Why? In *LOOCV* we train n models on an almost identical set of observations $\implies$ error estimates are highly correlated and are very sample-dependent.

  In contrast, for K-fold CV we average the outputs of K models trained on less correlated subsets (overlap is smaller). $K = 5$ or 10 were shown empirically to yield optimal test error estimates.

Cross-Validation for Classification

CV can easily be used for classification when the response variable is categorical.

The biggest difference of CV for classification as opposed to regression: to estimate a test error, we use

Err = # of misclassified observations,

instead of squared error $\sum_i (y_i - \hat{y}_i)^2$

E.g. LOOCV error rate for classification

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} Err_i, \quad \text{where } Err_i = \boldsymbol{I}(y_i \neq \hat{y}_i)$$

Same story for validation set approach and K-fold CV.