Model Assessment and Selection

Bin Li

IIT Lecture Series

Loss functions

- Typical choices for quantitative response Y:
  - Squared error: $L(Y, \hat{f}(X)) = (Y - \hat{f}(X))^2$
  - Absolute error: $L(Y, \hat{f}(X)) = |Y - \hat{f}(X)|$
- Typical choices for categorical response Y:
  - 0-1 loss: $L(Y, \hat{f}(X)) = I(Y \neq \hat{f}(X))$
  - Deviance (-2log-likelihood): $L(Y, \hat{p}(X)) = -2\sum_{k=1}^{K} I(Y = k) log \hat{p}_k = -2 log \hat{p}_Y$
- *Training error* is the average loss over the training sample (e.g. MSE and misclassification rate.)
- *Test error* is the average loss over an independent test sample.
- *In-sample error* is the *expected* prediction error conditioned on training set.
- *Generalization error* (or extra-sample error) is the *expected* prediction error over an independent test sample.

Performance assessment

- The *generalization* performance of a statistical model relates to its prediction capability on independent test data. Evaluation of this performance is *extremely important* in practice.
  - Model selection – estimating the performance of different models in order to choose the best one.
  - Model assessment – having chosen a final model, estimating its prediction error (generalization error) on new data.
- Three types of model assessment
  - In-sample error: performance on the sample used to develop model.
    - Without adjustment: in-sample *optimism*.
    - With adjustment: various information criteria: AIC, BIC, etc.
  - Internal: split sample, cross validation, bootstrapping.
  - External: independent test data.

Recap: a fundamental picture

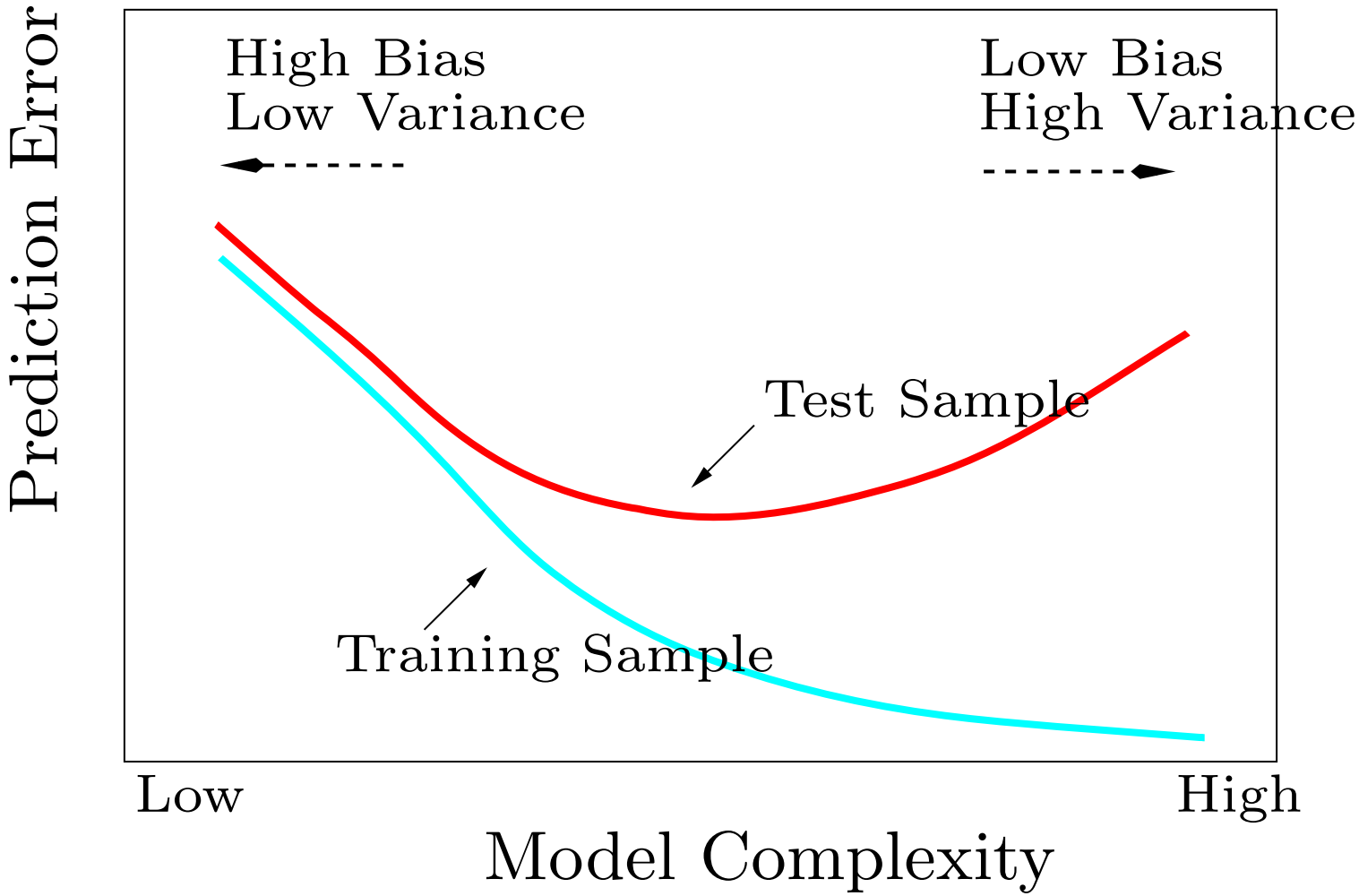


Figure from HTF 2001.

In-sample Optimism

- Let $\mathcal{T}$ be the training set and Y^0 the N new response values at each of the training points $\{x_i\}_{i=1}^N$.
- The *optimism* is defined as:

$$\text{op} \equiv \frac{1}{N}\sum_{i=1}^{N} E_{Y^0}\left[L(Y_i^0, \hat{f}(x_i))|\mathcal{T}\right] - \overline{err}$$

  where $\overline{err}$ is the training error for loss function $L(,)$.
- For squared error, 0-1 and other loss functions, we can show the expectation of optimism over training set is

$$E_{\mathbf{y}}(\text{op}) = \frac{2}{N}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- The amount of optimism depends on how strongly y_i affects its own prediction. The harder we fit the data, the greater $Cov(\hat{y}_i, y_i)$ will be.
- Assume additive model: $Y = f(X) + \epsilon$ and if $\hat{y}_i$ is from a linear fit with d inputs, it simplifies to $E_{\mathbf{y}}(\text{op}) = 2 \cdot \frac{d}{N}\sigma_\epsilon^2$

Mallow C_p, AIC and BIC

- Estimate the in-sample error by adding the *optimism* to the training error. Used for model selection.
  - Mallow C_p and AIC work in this way for a special class of estimates that are linear in their parameters.
- Mallow C_p statistic (squared error loss with d parameters):

$$C_p = \overline{err} + 2 \cdot \frac{d}{N} \hat{\sigma}_\epsilon^2$$

- Akaike information criterion (AIC) a more generally applicable estimate of in-sample error when a log-likelihood loss function is used:

$$\text{AIC} = -\frac{2}{N} \cdot \text{log-likehood} + \frac{2}{N} \cdot d$$

  For Gaussian model, AIC is equivalent to C_p statistic.
- Bayesian information criterion (BIC):

$$\text{BIC} = -2\text{log-likehood} + \log N \cdot d$$

Phoneme example

- Input features: log-periodograms measured at 256 frequencies.
- Response: two phonemes “aa” and “ao”.
- Training/test sets: 1000/717 samples.
- Figure below shows 15 periodograms for each class.

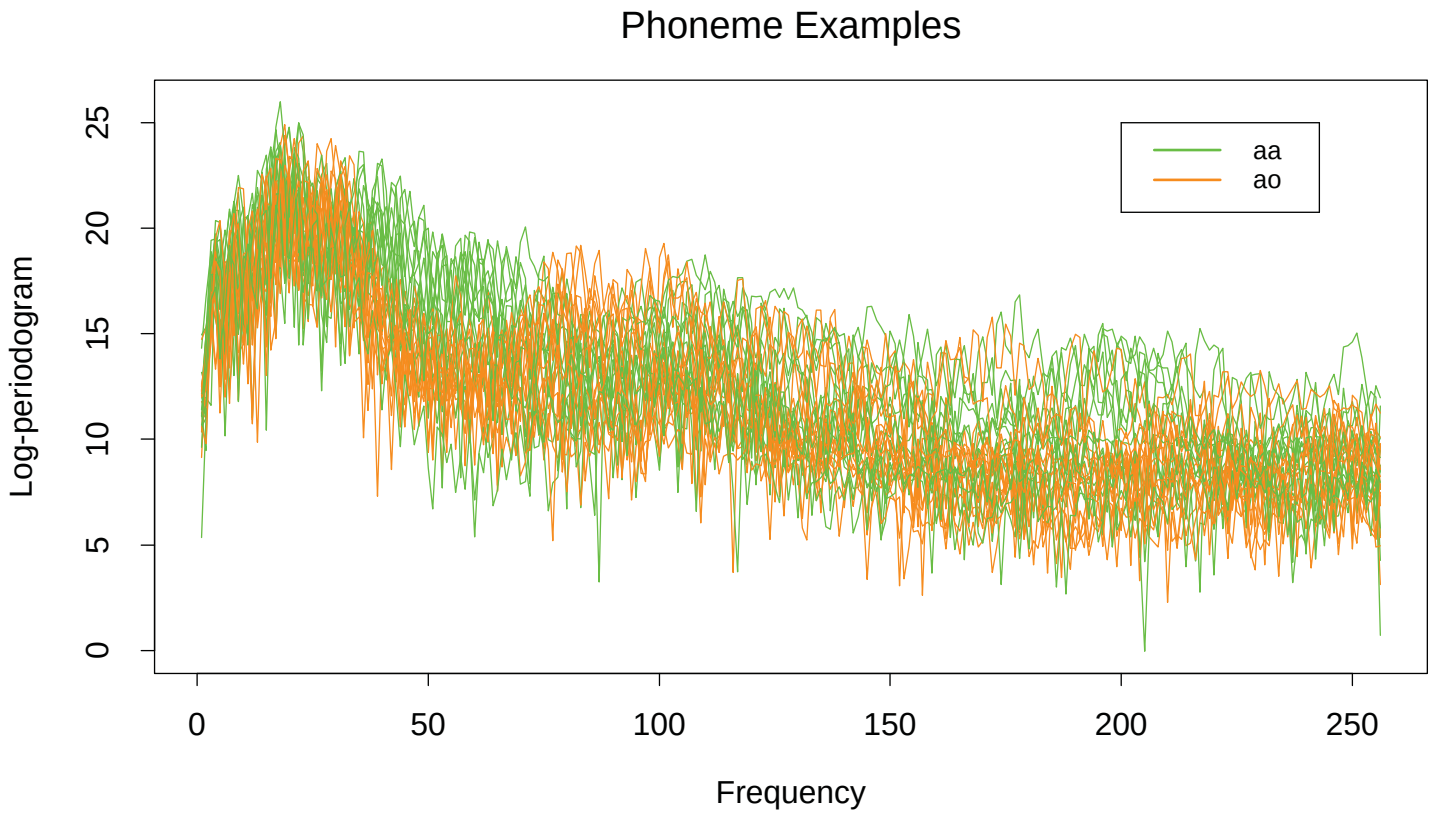


Figure from HTF 2009.

Training, test and AIC curves

- Logistic regression using spline basis functions.
- Left: Log-Likelihood loss is used for AIC. AIC is reasonably good except for the extremely over-parametrized case.
- Right: 0-1 loss is used for AIC, which also does a good job.

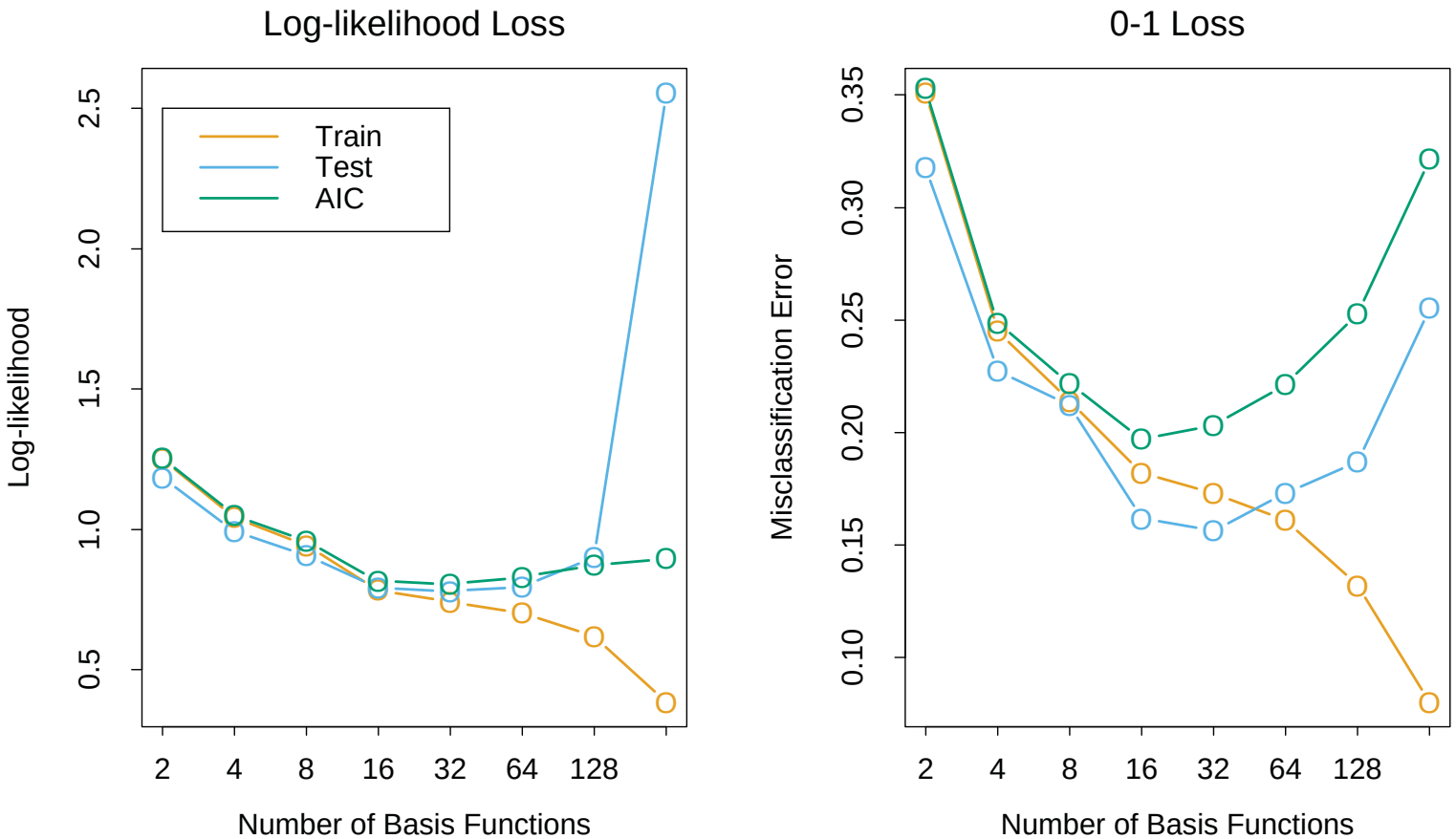


Figure from HTF 2009.

AIC or BIC?

- A lower AIC means a model is considered to be closer to the truth.
- A lower BIC means that a model is considered to be more likely to be the true model. Note BIC is an estimate of a function of the posterior probability of a model being true.
- BIC is asymptotically consistent as a selection criterion. That means, given a family of models including the true model, the probability that BIC will select the correct one approaches one as the sample size becomes large.
- AIC does not have the above property. Instead, it tends to choose more complex models as $N \to \infty$.
- For small or moderate samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity.

Validation set approach

- Suppose that we would like to find a set of variables that give the lowest test (not training) error rate.
- If we have a large data set, we can achieve this goal by randomly splitting the data into training and validation(testing) parts.
- We would then use the training part to build each possible model (i.e. the different combinations of variables) and choose the model that gave the lowest error rate when applied to the validation data.

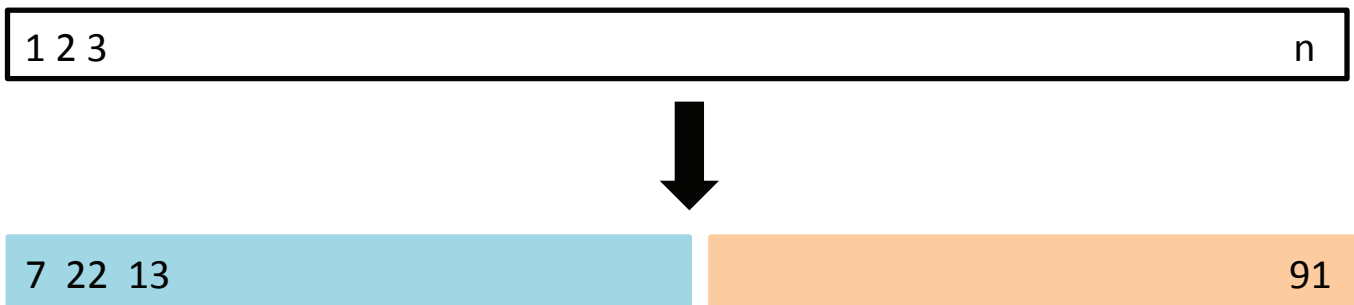


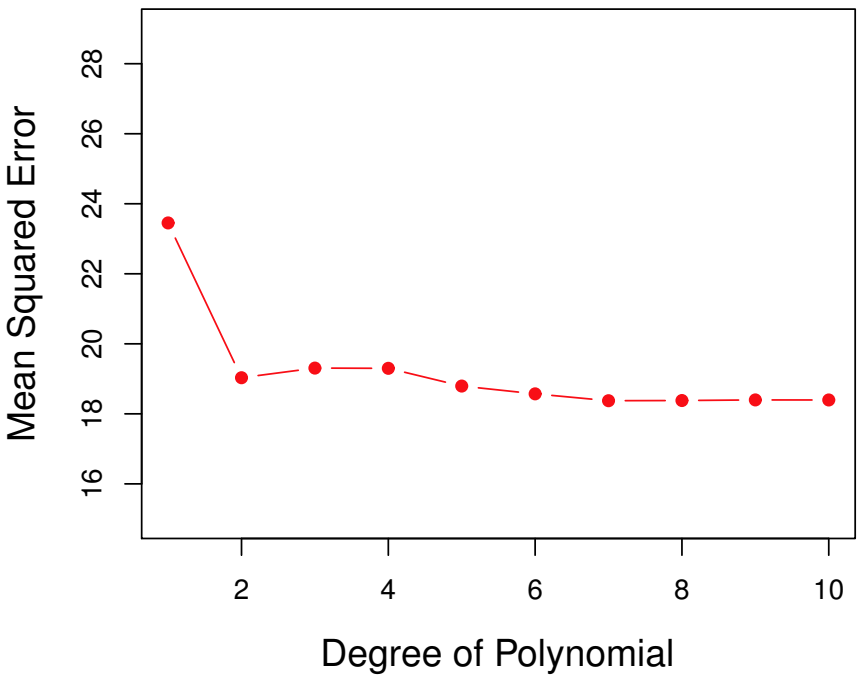
Figure from ISLR 2013.

Example: auto dataset

- Suppose that we want to predict "mpg" from "horsepower"
- Two models:
  - mpg $\sim$ horsepower
  - mpg $\sim$ horsepower + horsepower^2
- Which model gives a better fit?
  - Randomly split Auto data set into training (196 obs.) and validation data (196 obs.)
  - Fit both models using the training data set.
  - Evaluate both models using the validation data set.
  - The model with the lowest validation (testing) MSE is the winner!
- Pros: Simple and easy to implement.
- Cons: The validation MSE can be highly variable (see next slides). Only a subset of observations are used to fit the model (training data). Statistical methods tend to perform worse when trained on fewer observations.

Result: auto dataset

- Left: Validation error rate for a single split
- Right: Validation method repeated 10 times, each time the split is done randomly!
- There is a lot of variability among the MSE's ... Not good! We need more stable methods!



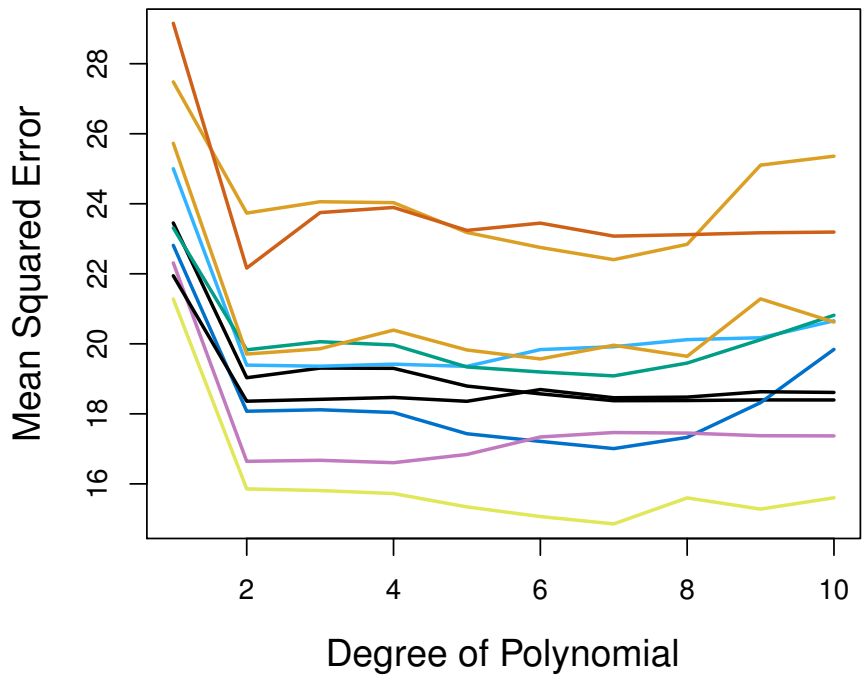


Figure from ISLR 2013.

Leave-one-out cross-validation (LOOCV)

- LOOCV repeats N times by using a single observation from the original data as the validation set, and the remaining as the training set.
- Address the second Cons but more computationally expensive!

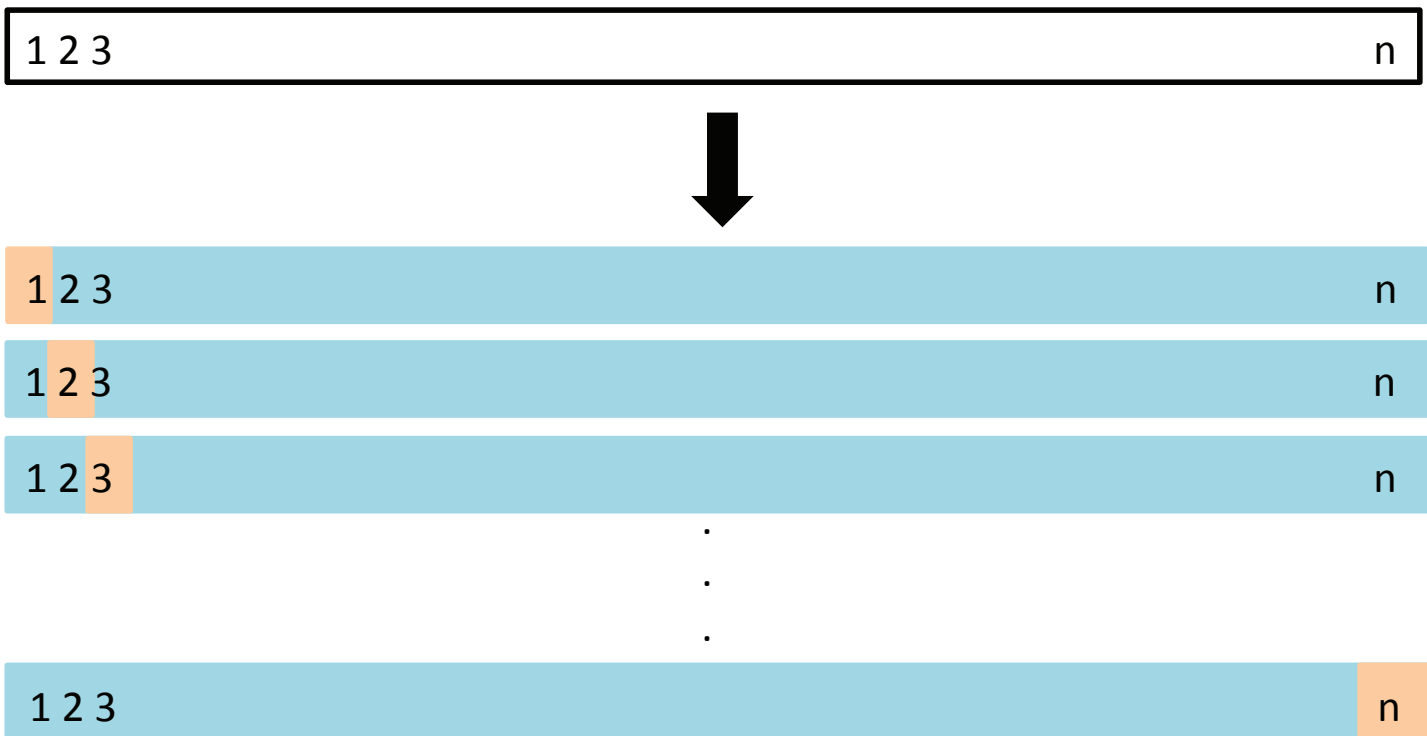


Figure from ISLR 2013.

LOOCV vs. validation set approach

- LOOCV has less bias.
  - We repeatedly fit the statistical learning method using training data that contains $N-1$ obs., i.e. almost all the data set is used.
- LOOCV produces a less variable MSE.
  - The validation approach produces different MSE when applied repeatedly due to randomness in the splitting process, while performing LOOCV multiple times will always yield the same results
- LOOCV is computationally intensive (fit the model N times).
  - But not always! For many linear fitting methods (i.e. $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$), we have

$$\text{LOOCV} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2$$

where S_{ii} is the i^{th} diagonal elements of $\mathbf{S}$

k-fold cross validation

- LOOCV is computationally intensive, so we can run k-fold Cross Validation instead
- With k-fold Cross Validation, we divide the data set into k different parts (e.g. $k = 5$ or $k = 10$, etc.)
- We then remove the first part, fit the model on the remaining $k - 1$ parts, and see how good the predictions are on the left out part (i.e. compute the MSE on the first part).
- We then repeat this k different times taking out a different part each time.
- By averaging the k different MSE's we get an estimated validation (test) error rate for new observations.

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

k-fold cross validation (cont.)

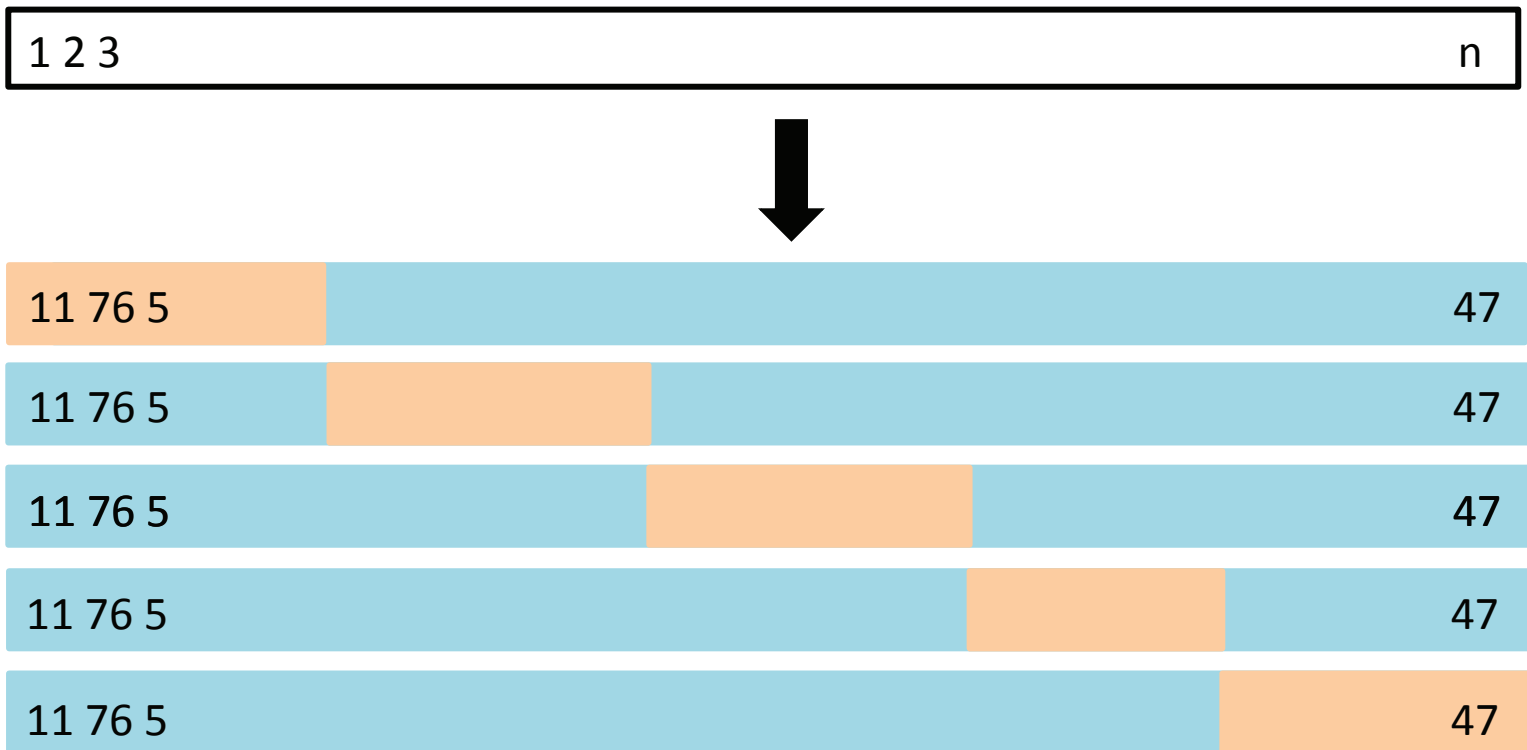


Figure from ISLR 2013.

Auto Data: LOOCV vs. k-fold CV

- Left: LOOCV error curve. Right: 10-fold CV was run many times, and the figure shows the slightly different CV error rates.
- LOOCV is a special case of k-fold, where $k = n$.
- They are both stable, but LOOCV is more computationally intensive!

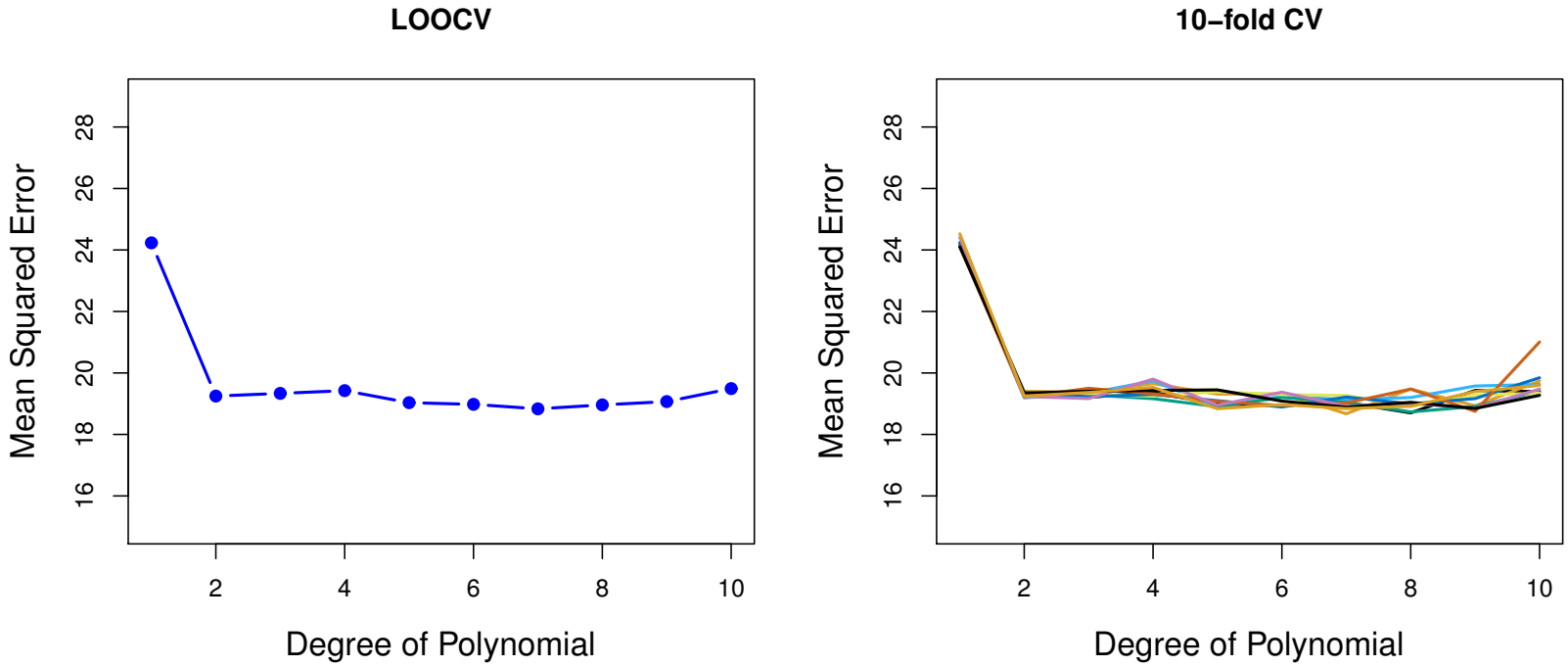


Figure from ISLR 2013.

k-fold cross validation on three simulated data

- Blue: True Test MSE
- Black: LOOCV MSE
- Orange: 10-fold MSE

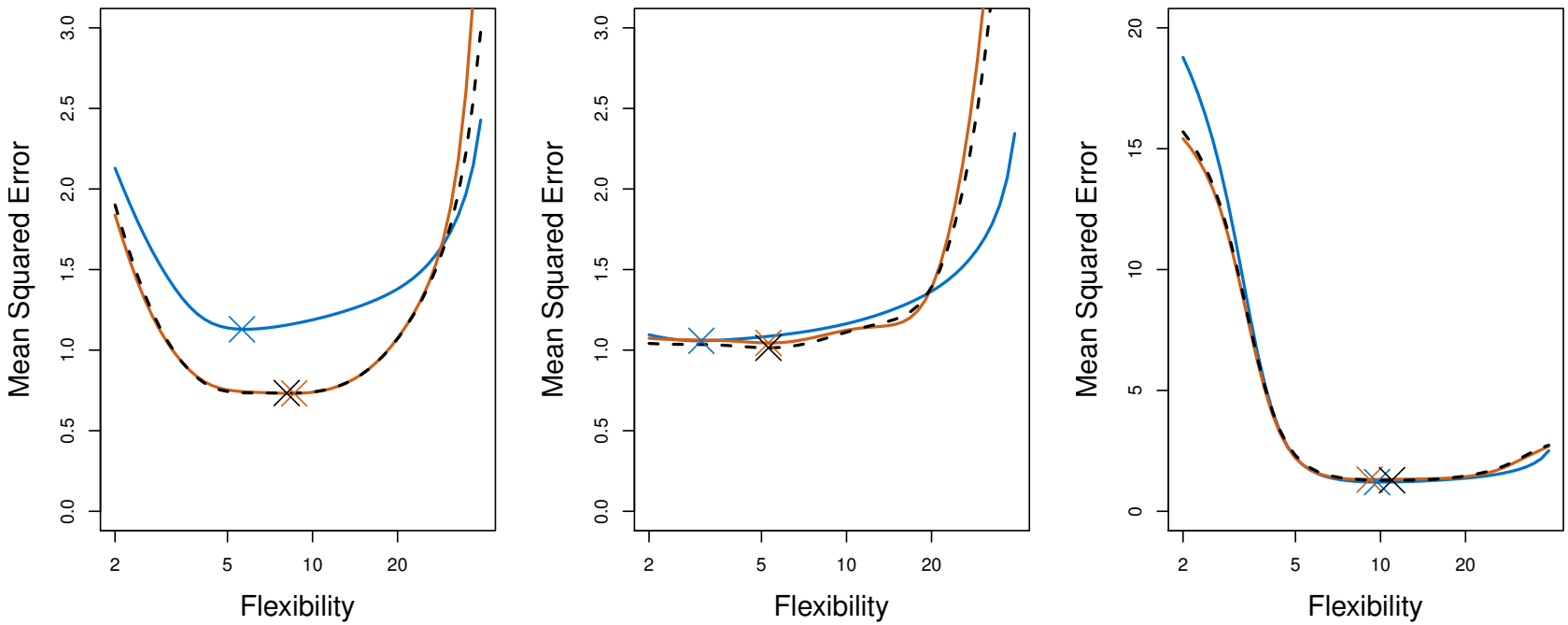


Figure from ISLR 2013.

Bias- variance trade-off for k-fold CV

- Putting aside that LOOCV is more computationally intensive than k-fold CV ... Which is better LOOCV or k-fold CV?
  - LOOCV is less bias than k-fold CV (when $k < n$)
  - But, LOOCV has higher variance than k-fold CV (when $k < n$)
  - Thus, there is a trade-off between what to use
- Conclusion:
  - We tend to use k-fold CV with ($k = 5$ and $k = 10$)
  - These are the magical k's
  - It has been empirically shown that they yield test error rate estimates that suffer neither from excessively high bias, nor from very high variance.

Cross validation on classification problems

- So far, we have been dealing with CV on regression problems.
- We can use cross validation in a classification situation in a similar manner.
  - Divide data into k parts.
  - Hold out one part, fit using the remaining data and compute the error rate on the hold out data.
  - Repeat k times.
  - CV error rate is the average over the k errors we have computed.

CV to choose order of polynomial

- The purple dashed line is the Bayes' boundary with Bayes error rate 0.133.
- Linear logistic regression (degree 1) is not able to fit the Bayes' decision boundary.
- Quadratic logistic regression does better than linear.

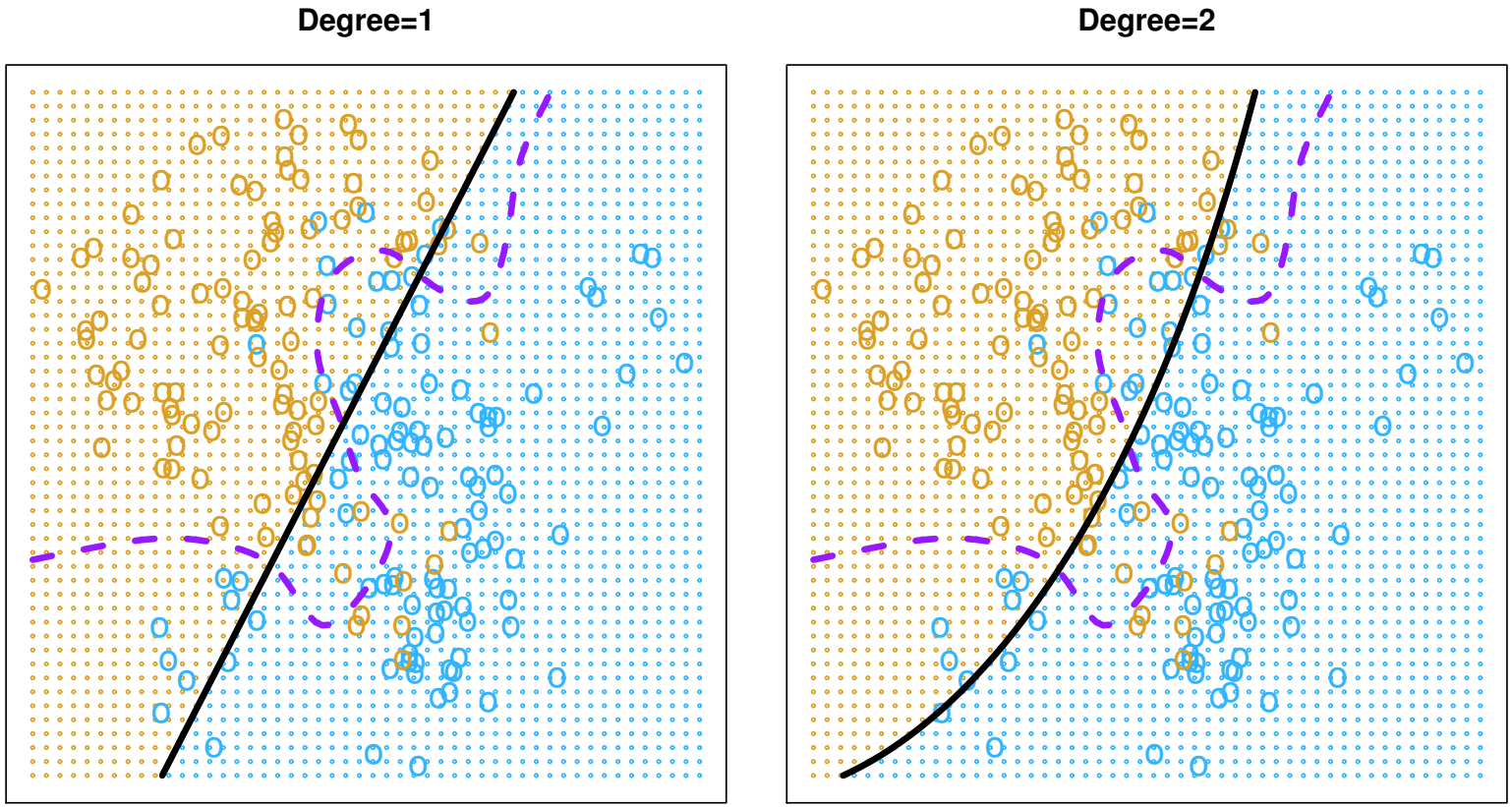


Figure from ISLR 2013.

CV to choose order of polynomial (cont.)

- Using cubic and quartic predictors, the accuracy of the model improves.

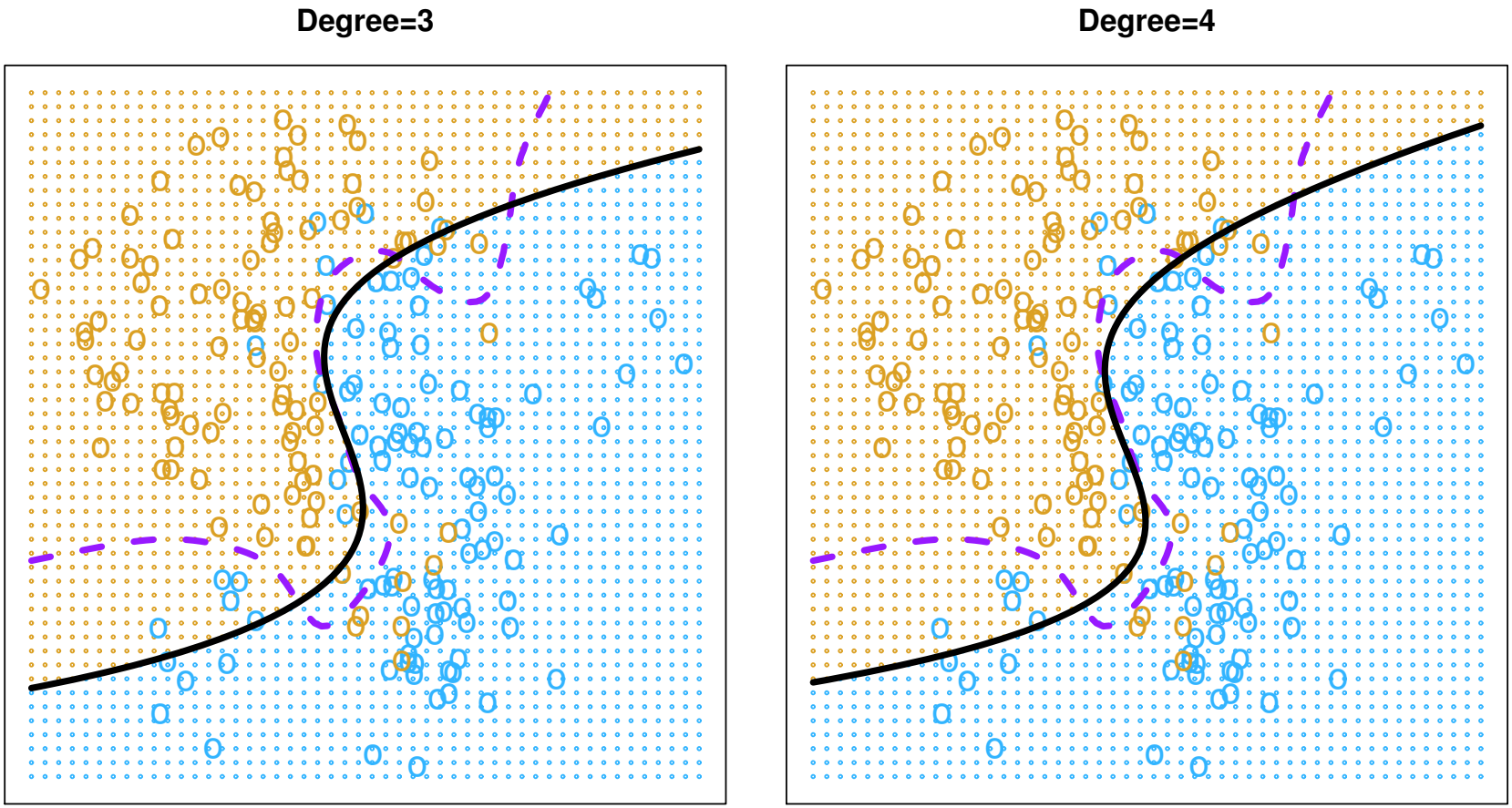


Figure from ISLR 2013.

CV to choose order of polynomial (cont.)

- Brown: test error
- Blue: training error
- Black: 10-fold CV error

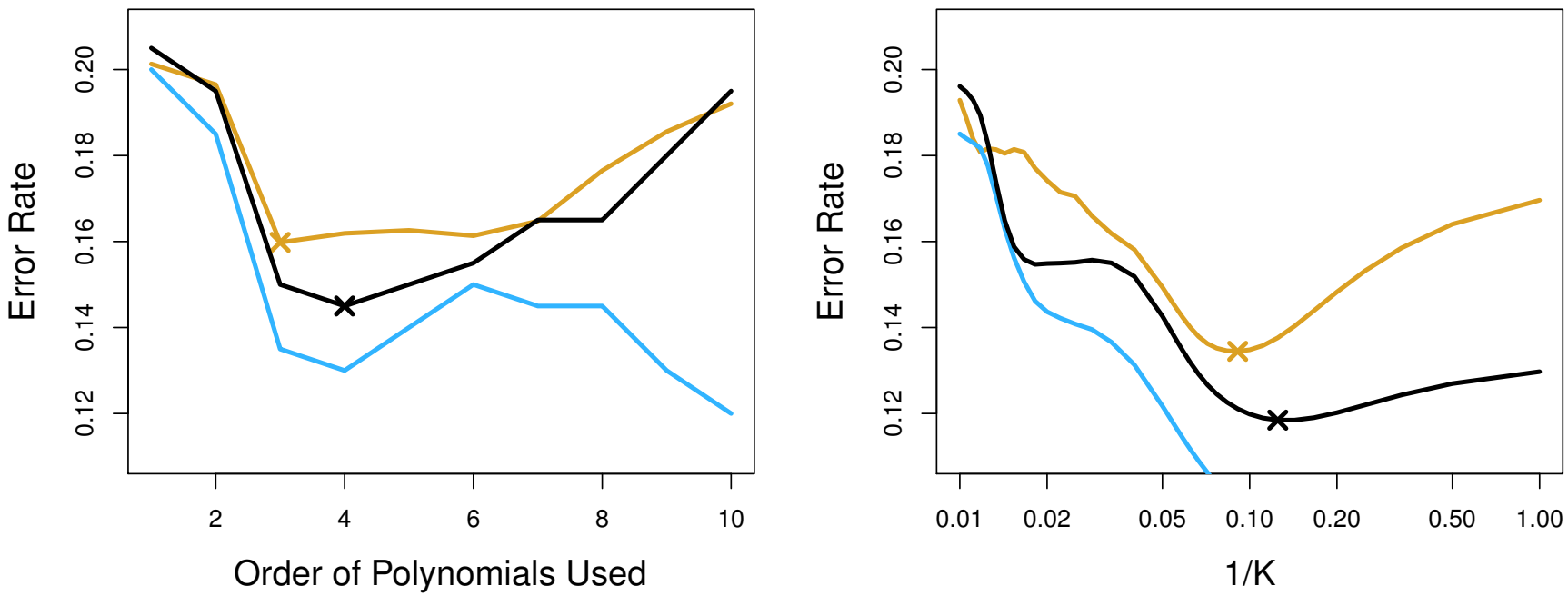
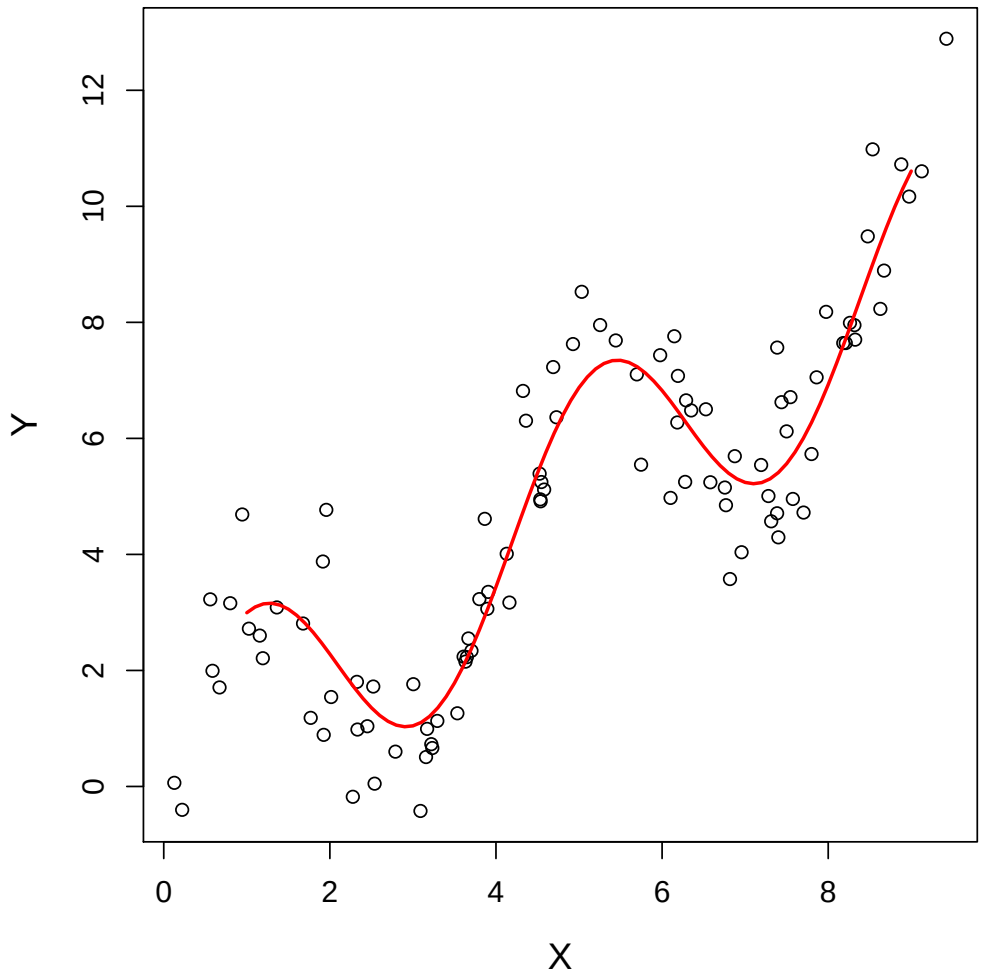


Figure from ISLR 2013.

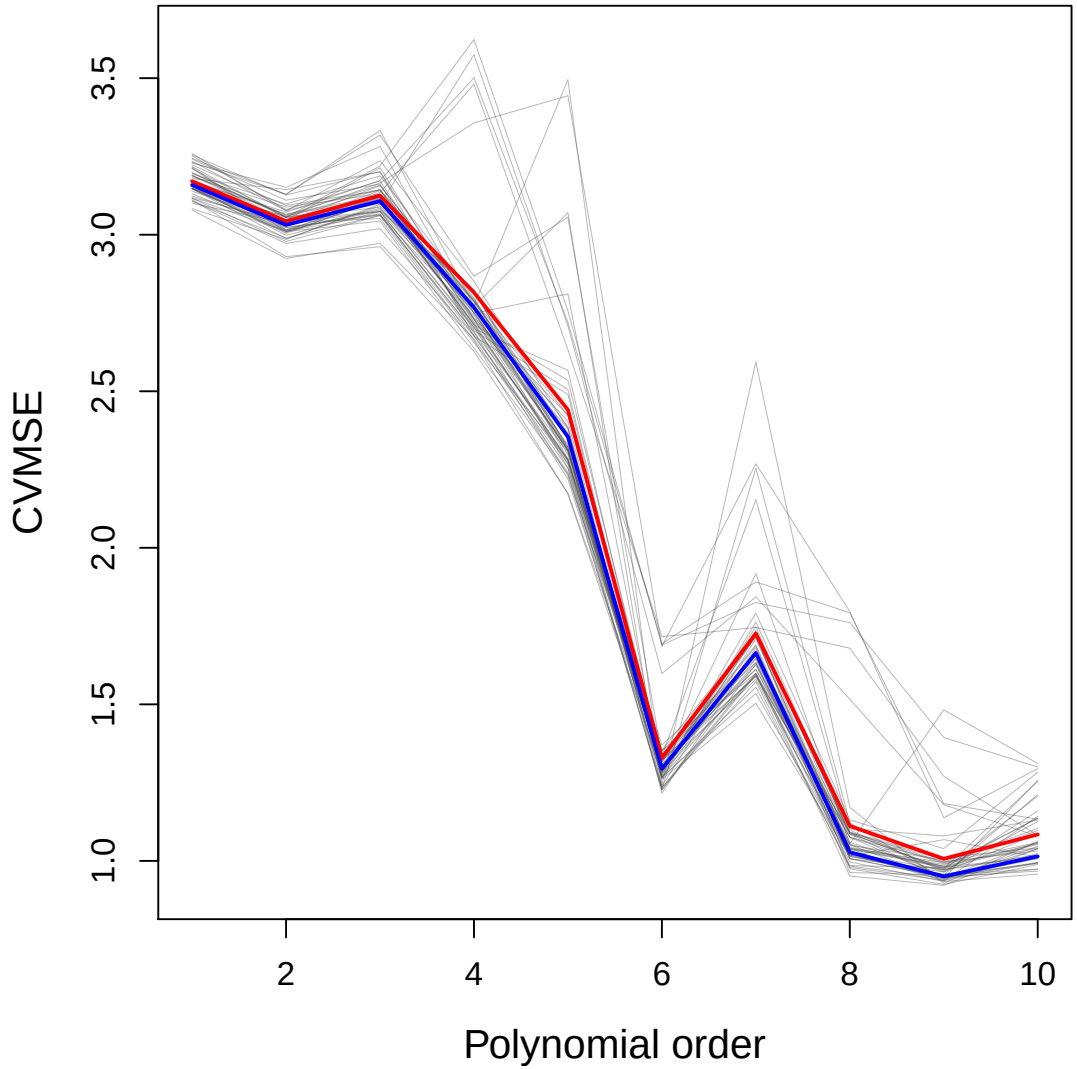
Revisit the regression simulation example

- Data:
  $y_i = 2sin(1.5x_i) + x_i + \epsilon_i$,
  where $\epsilon_i \sim N(0, 1)$
- Training set: `dat` has 100 observation.
- Fit the data using polynomial regressions.
- Range of polynomial order: from 1 (SLR) to 10.
- Use 10-fold CV and LOOCV to select the optimal polynomial order.



CV curves among 50 replications

- The grey lines on the backgrounds are the CV errors (in terms of MSE) based on 50 repetitions.
- For each repetition, the data is randomly split into 10 folds. Then 10-fold CV is applied on various orders of polynomial regression.
- The red line is the average CV errors from 50 repetitions. The optimal order is 9 although there is a kink around order=6.
- The blue line is the LOOCV error.



R code for cross-validation

```
> # K-fold CV function: return a list with
> # K components. One for each fold.
> cv.folds <- function(n,K,seed){
+  set.seed(seed)
+  split(sample(1:n),rep(1:K,length=n))
+ }
> # cv.pred: function estimate CV errors
> cv.preg<-function(data,ord,K,seed){
+  all.folds<-cv.folds(nrow(data),K,seed)
+  cv.err.mat<-matrix(0, length(ord), K)
+  for (i in seq(K)){
+   for (j in 1:length(ord)){
+    omit <- all.folds[[i]]
+    fit  <- lm(y~poly(x,j,raw=T),data[-omit,])
+    pred <- predict(fit,data[omit,])
+    cv.err.mat[j,i]<-mean((data$y[omit]-pred)^2)
+   }
+  }
+  out<-as.vector(apply(cv.err.mat,1,mean))
+  return(out)
+ }
```

```
> # 10-fold CV
> ord <- 1:10; K<-10; iter <- 50
> cv.mse <- matrix(0,iter,length(ord))
> for (it in 1:iter){
+   cv.mse[it,] <- cv.preg(dat,ord,K,it)
+ }
> # LOOCV
> loocv.mse <- rep(0,10)
> for (i in 1:10){
+  fit<-lm(y~poly(x,i,raw=T),dat)
+  loocv.mse[i]<-mean(((dat$y-fit$fit)/
+                 (1-hatvalues(fit)))^2)
+ }
> # Plot CV curves on the background
> plot(ord,cv.mse[1,],xlab="Polynomial order",
+ ylab="CVMSE",ylim=range(cv.mse),type="n")
> for (it in 1:iter){
+   lines(ord,cv.mse[it,],lwd=0.3)
+ }
> # Average 10-fold CV errors
> lines(ord,apply(cv.mse,2,mean),col="red",lwd=2)
> # LOOCV errors
> lines(ord,loocv.mse,col="blue",lwd=2)
```

Boston housing example

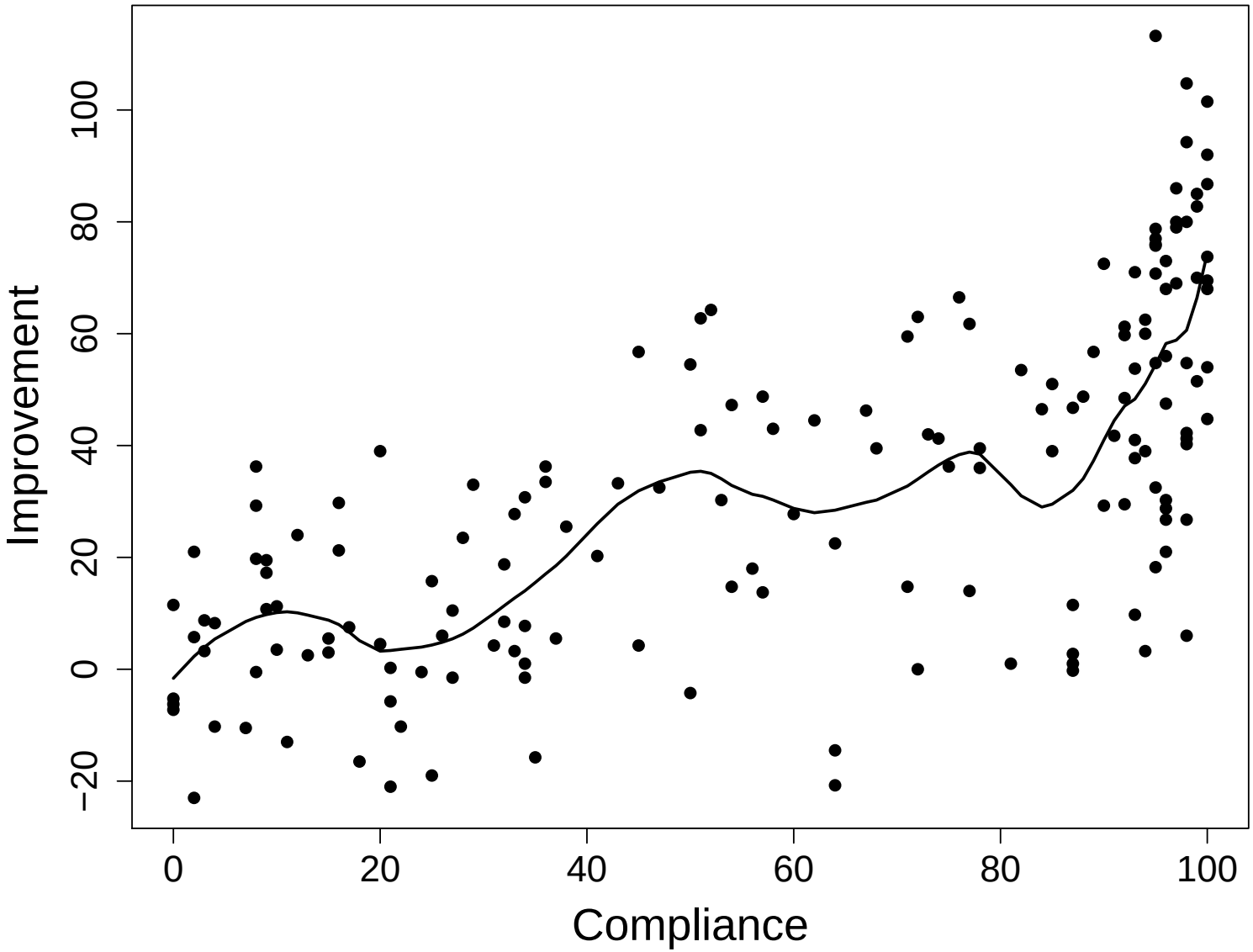
- Split the 506 observations into training set (100 obs.) and test set (406 obs.)
- Fit multiple linear regression on the training set
- Use 10-fold CV on the training set to estimate its performance on new data.
- Models `fit1` and `fit2` are the same.
- Function `cv.glm` in `boot` library calculates the estimated K-fold CV prediction error for generalized linear models.
- `delta[1]` is the raw CV estimate of prediction error. `delta[2]` is the bias-adjusted cross-validation estimate.

```
> library(mlbench)
> data(BostonHousing)
> bh <- BostonHousing
> set.seed(1)
> indx<-sample(1:506,size=506,replace=F)
> bh.train<-bh[indx[1:100],]
> bh.test<-bh[indx[101:506],]
> fit1 <- lm(medv~.,data=bh.train)
> fit2 <- glm(medv~.,data=bh.train,family=gaussian)
> summary(fit2)
    Null deviance: 8224.8 on 99 degrees of freedom
Residual deviance: 1943.8 on 86 degrees of freedom
AIC: 610.51
> sum(fit2$resi^2) #Residual deviance
[1] 1943.847
> mean(fit2$resi^2)
[1] 19.43847
> pred <- predict(fit2,newdata=bh.test)
> mean((pred-bh.test$medv)^2)
[1] 26.6657
> library(boot)
> val.10.fold<-cv.glm(data=bh.train,glmfit=fit2,K=10)
> val.10.fold$delta
[1] 29.67401 29.04337
```

Cholostyramine example

- $n = 164$ men took part in an experiment to see if the drug cholostyramine lowered blood cholesterol levels. The men were supposed to take six packets of cholostyramine per day, but many of them actually took much less.
- The explanatory variable, which we call z, is the *compliance*, as a percentage of the intended dose actually taken.
- The response variable, which we call y, is the *improvement* the decrease in total blood plasma cholesterol level from the beginning to the end of the experiment.
- We are interested the *nonlinear* relationship between the compliance and improvement.

Cholostyramine example (cont.)



Loess smoother

- For each point $\mathbf{x}_i = (z_i, y_i)$, the $\alpha \times n$ nearest points are identified based on the distance $|z_i - z|$. We call this neighborhood of $\alpha \times n$ points "$\mathcal{N}(z)$".
  - With $\alpha = 0.30$ and $n = 164$, the algorithm puts 49 points into $\mathcal{N}(z)$.
- A weighted least-squares linear regression

$$\hat{r}_z(Z) = \hat{\beta}_{z,0} + \hat{\beta}_{z,1} Z$$

  is fit to the $\alpha \times n$ points in $\mathcal{N}(z)$, where the *weights* $w_{z,j}$ are positive numbers which depend on $|z_j - z|$. Let

$$u_j = \frac{|z_j - z|}{\max_{\mathcal{N}(z)} |z_k - z|},$$

  the weights w_j equal $(1 - u_j^3)^3$.
- Finally, the loess estimate $\hat{r}_{loess}(z)$ is set equal to the value of $\hat{r}_z(Z)$ at $Z = z$.
- In `R`, `loess` is the function to fit Loess smoother. The value of the tuning parameter α can be specified through option `span` in `loess`.

Cholostyramine example (cont.)

- We use 10-fold CV to select the optimal value of α minimizing the MSE.
- The `crossval` function in `bootstrap` library is a generic function to calculate the estimated K-fold CV prediction error.
- We applied 10-fold CV 20 times. Each time, the crossvalidated MSE are calculated on a grid value of α from 0.2 to 0.8 by 0.1.
- Then we average the CV MSE based on 20 replications for each value of α.
- Based on the plot on next slide, we see the optimal value of α is at 0.5.

```
> library(bootstrap)
> data(cholost); x<-cholost$z; y<-cholost$y
> sp <- seq(from=0.2,to=0.8,le=7)
> cv.res <- matrix(0,20,7)
> for (i in 1:20){
+   set.seed(i)
+   for (j in 1:7){
+     lfit <- function(x,y){loess(y~x,span=sp[j])}
+     lpred <- function(fit,x){predict(fit,x)}
+     cv.out<-crossval(x,y,lfit,lpred,ngroup=10)
+     cv.res[i,j] <- mean((y-cv.out$cv.fit)^2)
+   }
+ }
> round(cv.res,d=0)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]  474  477  476  458  475  468  474
 [2,]  481  482  464  469  463  479  467
...
[19,]  519  472  466  460  462  471  484
[20,]  467  478  475  475  468  464  475
> plot(sp,cv.res[1,],type="n",xlab="Span value",
+      ylab="CV_MSE",ylim=range(cv.res))
> for (i in 1:20){
+   lines(sp,cv.res[i,],col=grey(0.5),lwd=0.5)
+ }
> lines(sp,apply(cv.res,2,mean),col="blue",lwd=2)
```

Cholostyramine example (cont.)