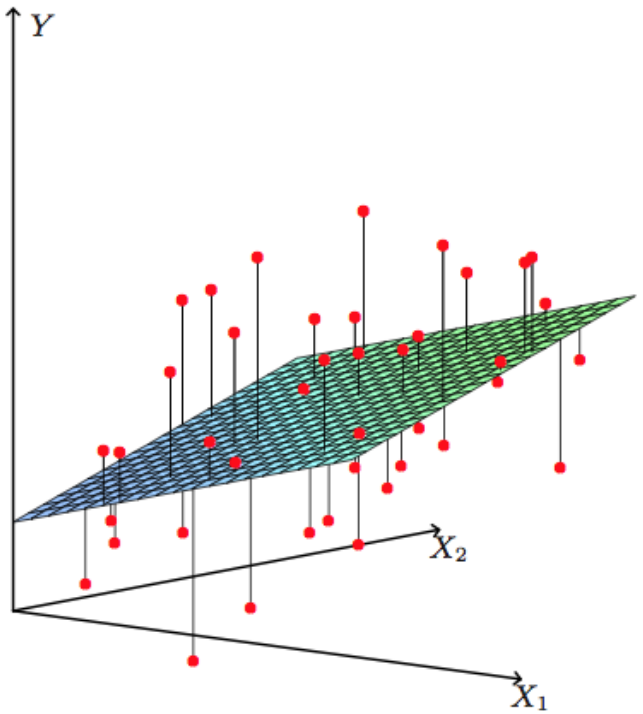
Data Mining and Machine Learning

Kuangnan Fang

Department of Statistics, Xiamen University
Email: xmufkn@xmu.edu.cn



Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.

Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.

Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.
- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates

Training Error versus Test error

- Recall the distinction between the *test error* and the *training error:*
- The *test error* is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- In contrast, the *training error* can be easily calculated by applying the statistical learning method to the observations used in its training.
- But the training error rate often is quite different from the test error rate, and in particular the former can *dramatically underestimate* the latter.

Assessing Model Accuracy

Suppose we fit a model $\hat{f}(x)$ to some training data $\mathsf{Tr} = \{x_i, y_i\}_1^N$, and we wish to see how well it performs.

- We could compute the average squared prediction error over Tr:

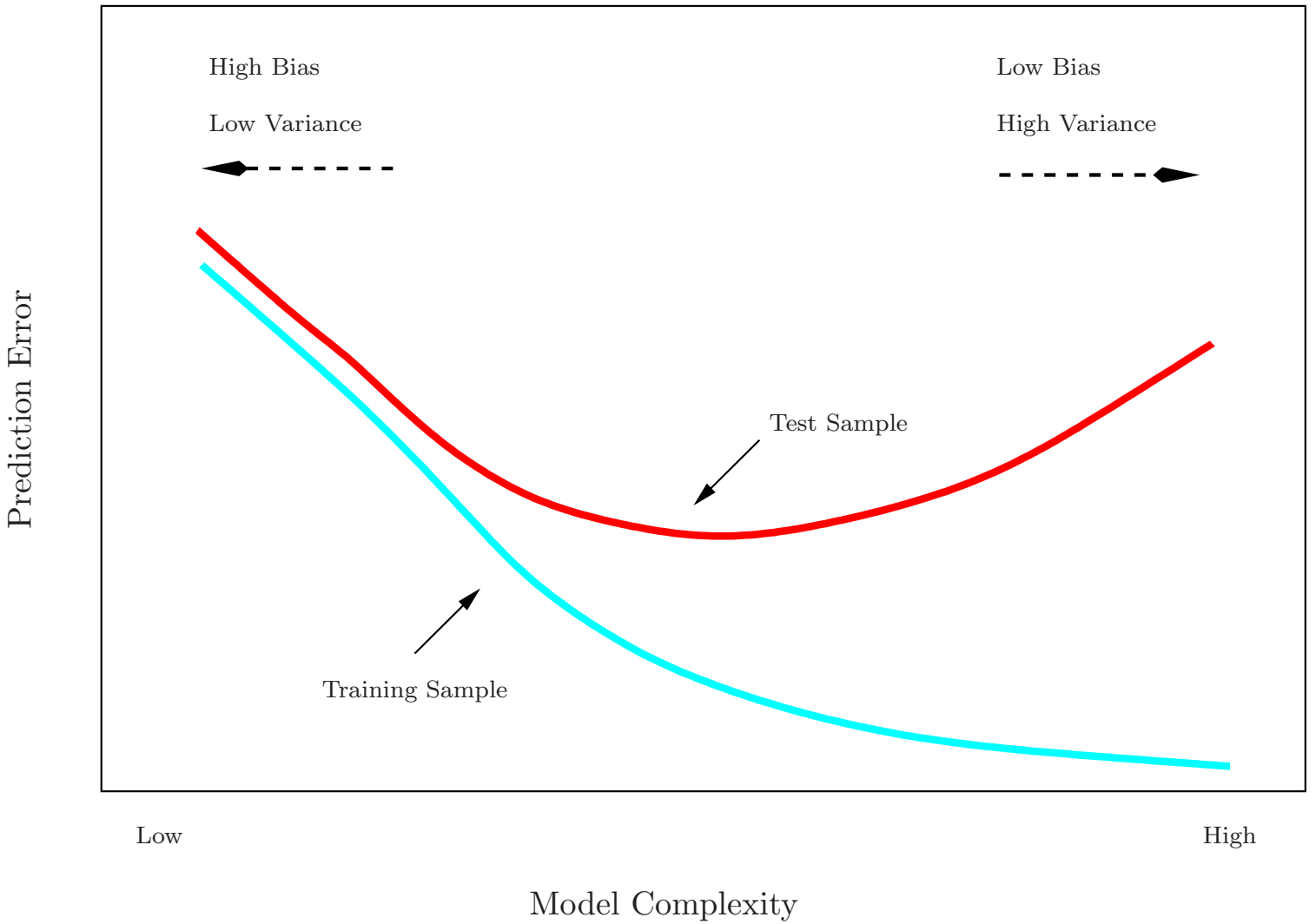
$$\text{MSE}_{\mathsf{Tr}} = \text{Ave}_{i \in \mathsf{Tr}}[y_i - \hat{f}(x_i)]^2$$

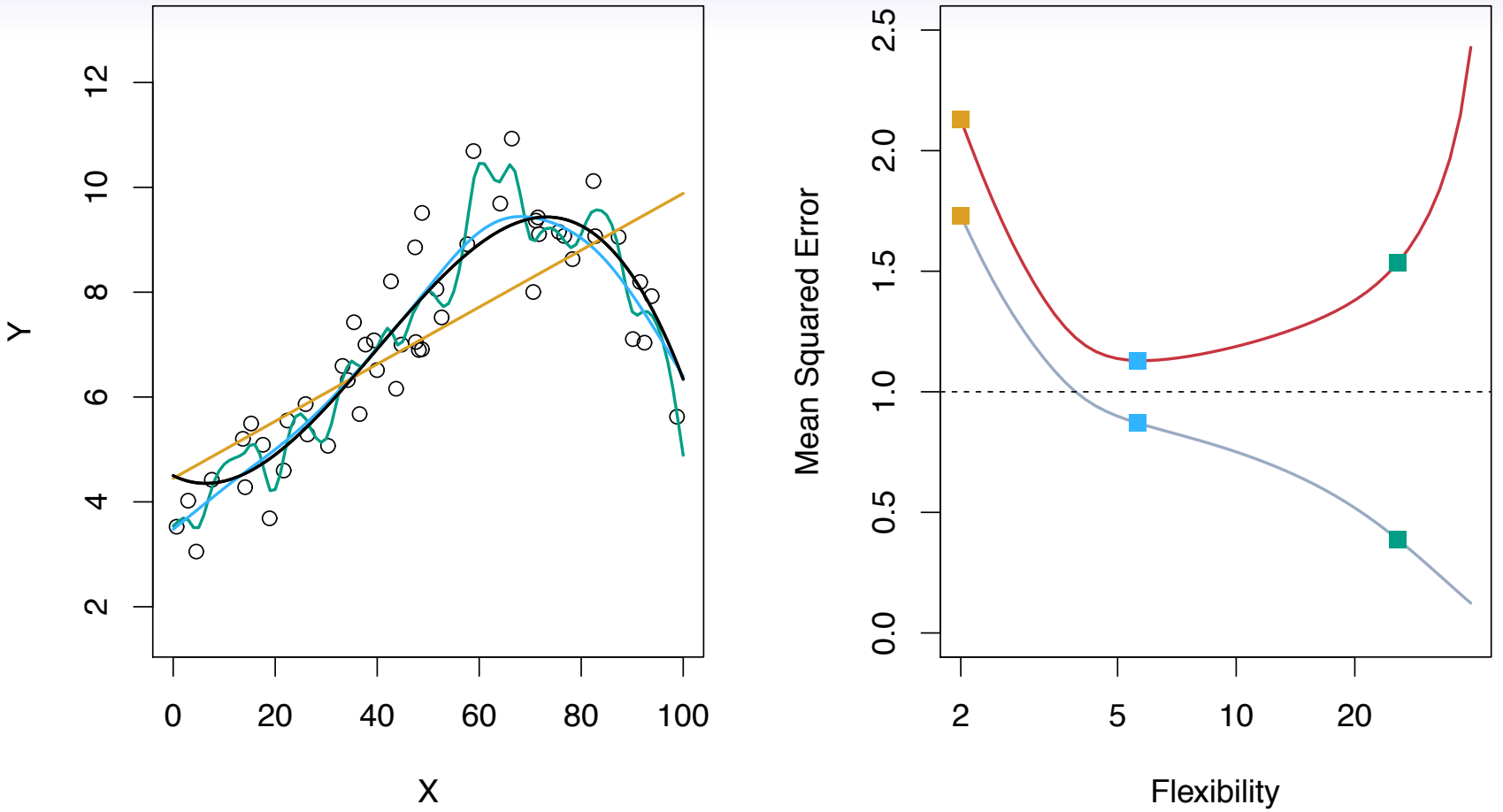
This may be biased toward more overfit models.

- Instead we should, if possible, compute it using fresh *test* data $\mathsf{Te} = \{x_i, y_i\}_1^M$:

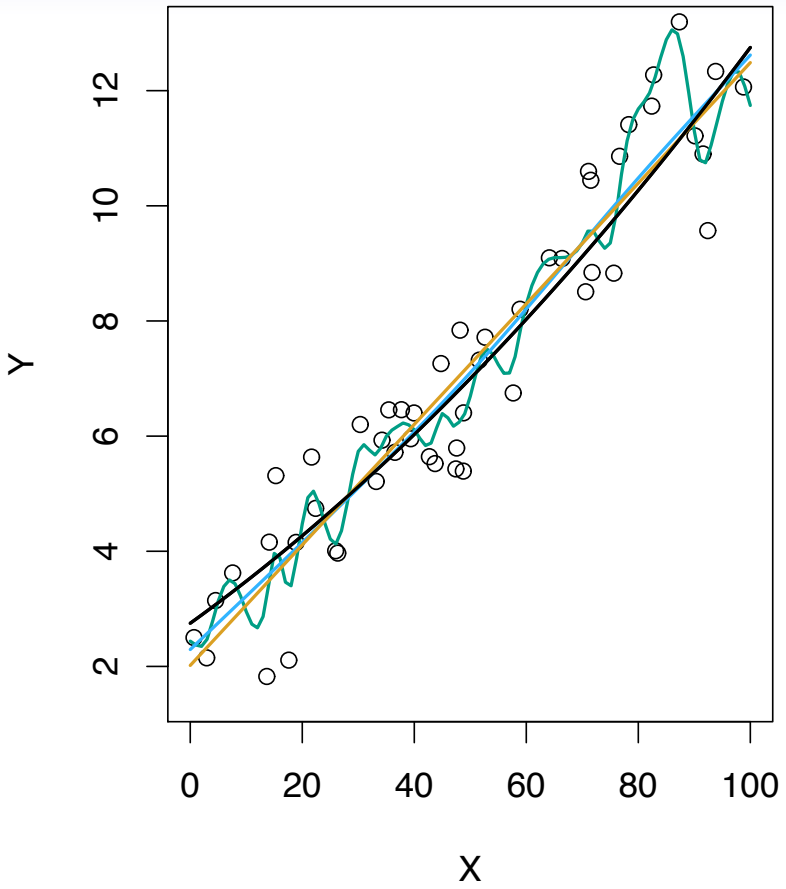
$$\text{MSE}_{\mathsf{Te}} = \text{Ave}_{i \in \mathsf{Te}}[y_i - \hat{f}(x_i)]^2$$

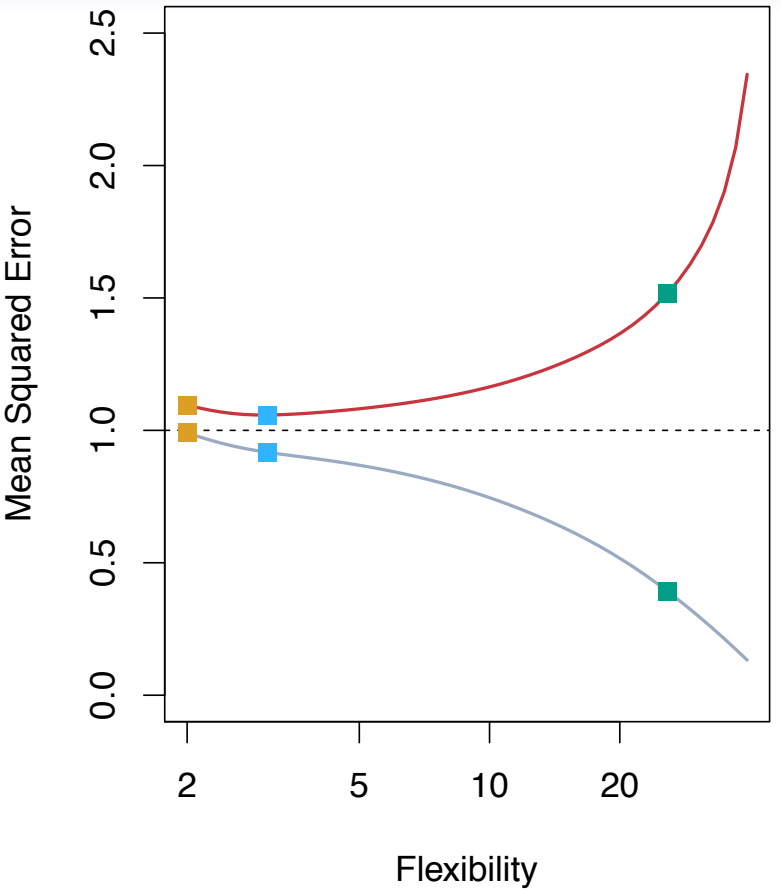
Training- versus Test-Set Performance



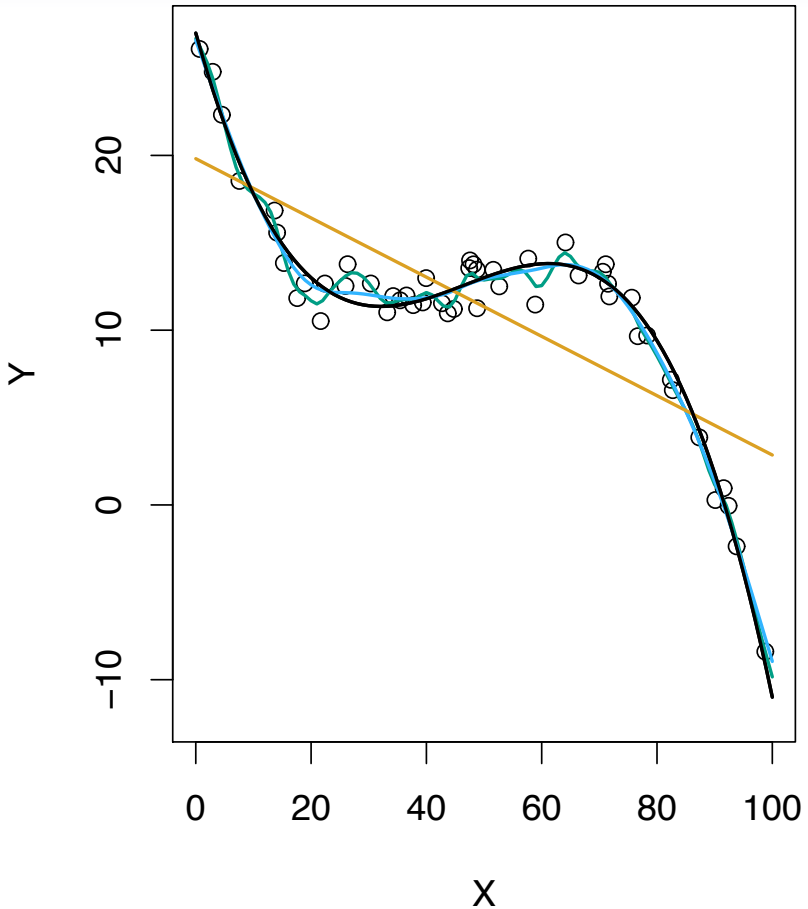


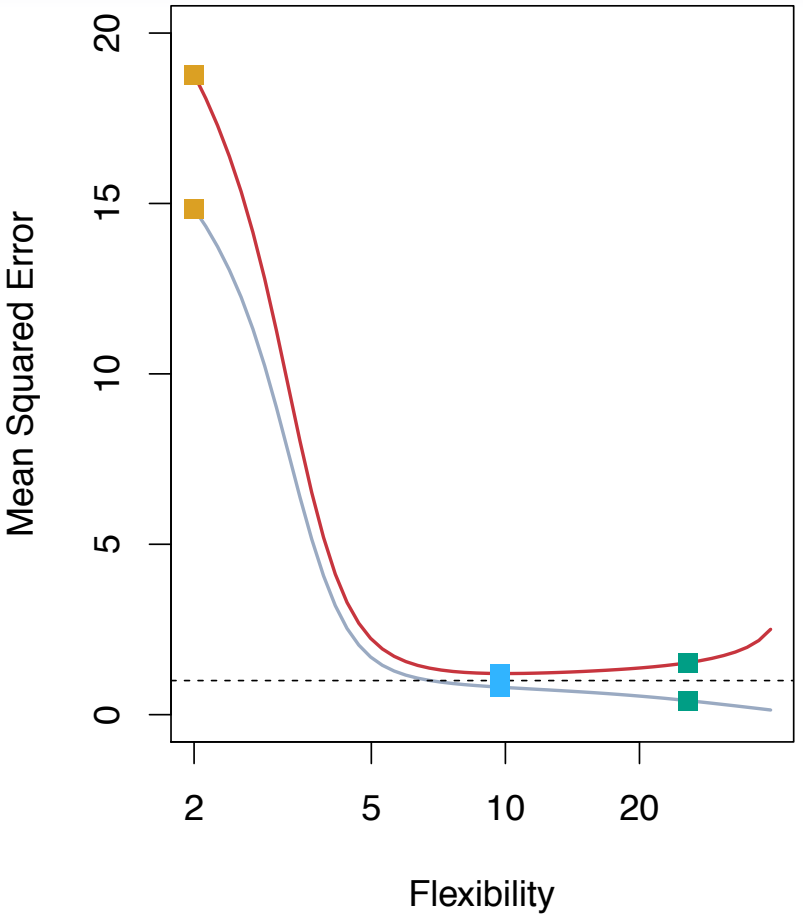
Black curve is truth. Red curve on right is MSE_{Te}, grey curve is MSE_{Tr}. Orange, blue and green curves/squares correspond to fits of different flexibility.





Here the truth is smoother, so the smoother fit and linear model do really well.





Here the truth is wiggly and the noise is low, so the more flexible fits do the best.

Bias-Variance Trade-off

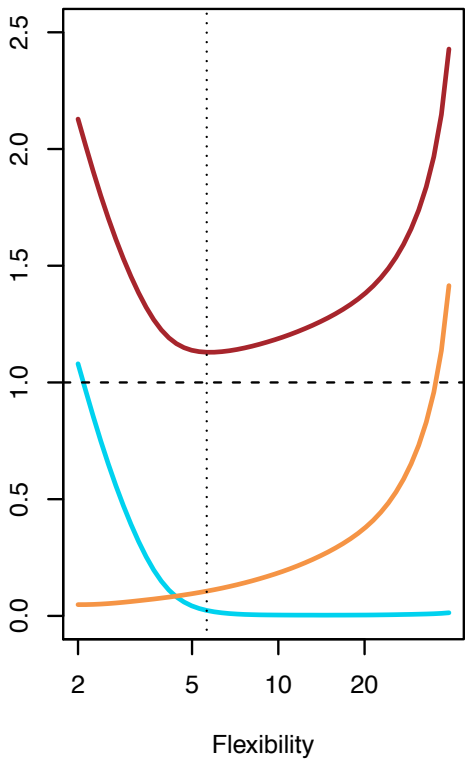
Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

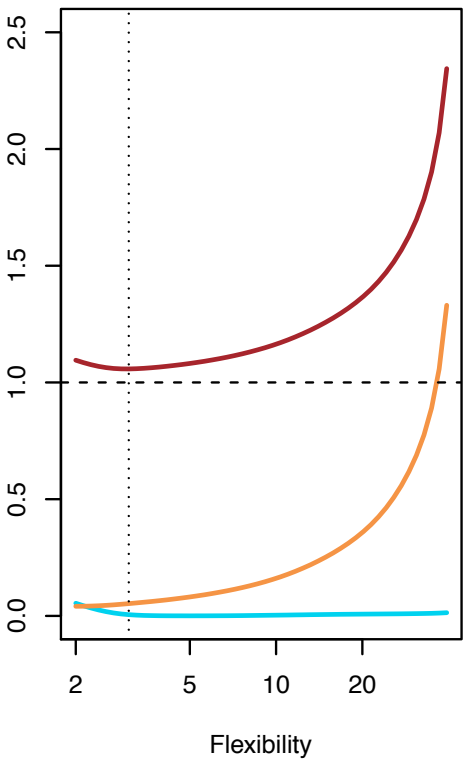
$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

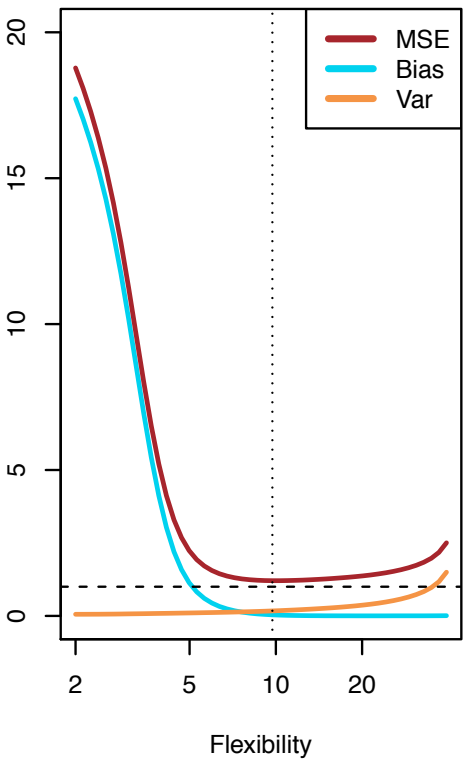
The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $\text{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$.

Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off.*

Bias-variance trade-off for the three examples







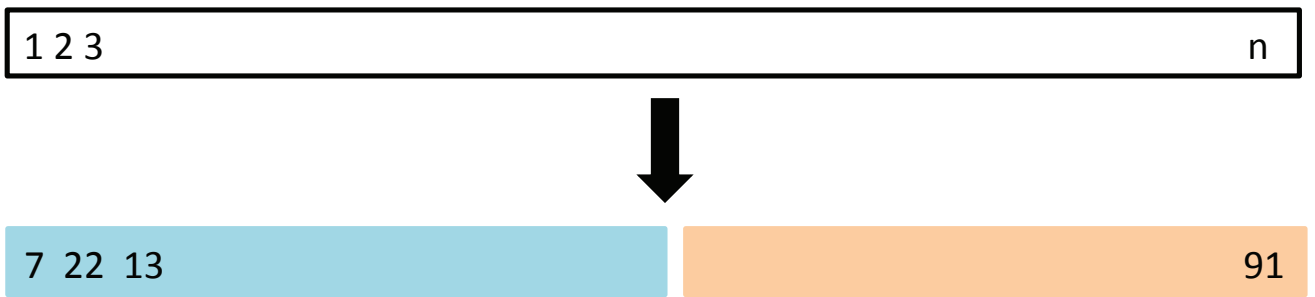
More on prediction-error estimates

- Best solution: a large designated test set. Often not available
- Some methods make a *mathematical adjustment* to the training error rate in order to estimate the test error rate. These include the *Cp statistic*, *AIC* and *BIC*. They are discussed elsewhere in this course
- Here we instead consider a class of methods that estimate the test error by *holding out* a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations

Validation-set approach

- Here we randomly divide the available set of samples into two parts: a *training set* and a *validation* 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 validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.

The Validation process

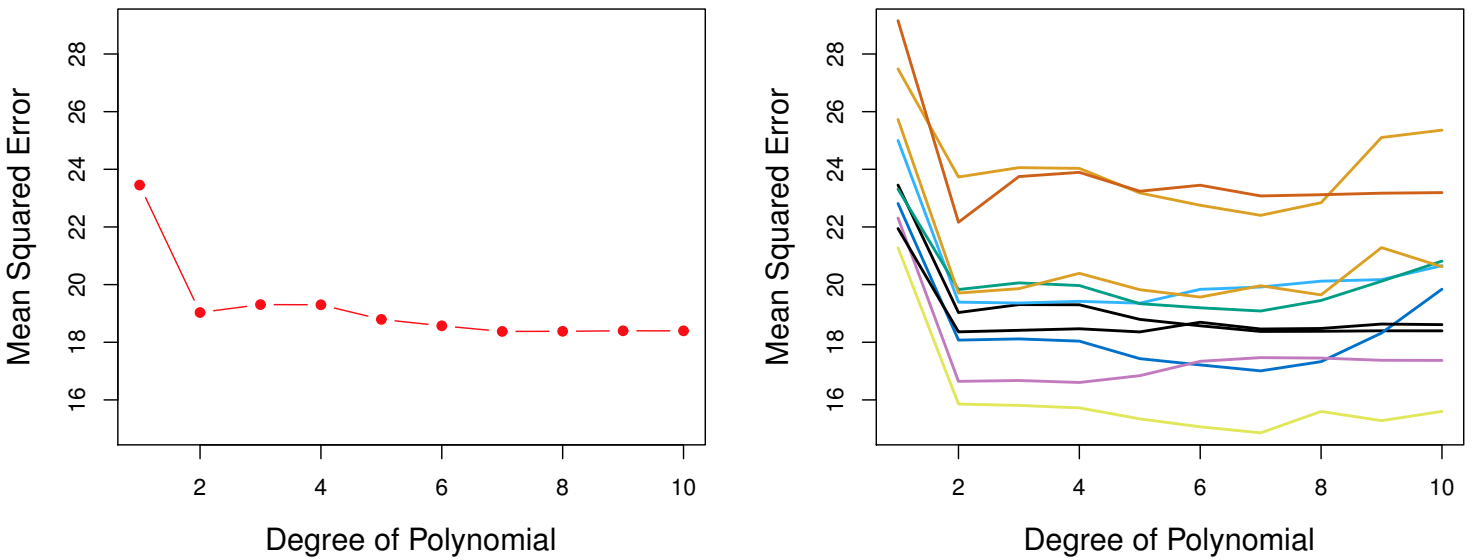


A random splitting into two halves: left part is training set, right part is validation set

Example: automobile data

- Want to compare linear vs higher-order polynomial terms in a linear regression
- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.

mpg=b0+b1*horsepower+b2*horsepower^2+e



Left panel shows single split; right panel shows multiple splits

Drawbacks of validation set approach

- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to *overestimate* the test error for the model fit on the entire data set.

Drawbacks of validation set approach

- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to *overestimate* the test error for the model fit on the entire data set. *Why?*

K-fold Cross-validation

- *Widely used approach* for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part.
- This is done in turn for each part $k = 1, 2, \ldots K$, and then the results are combined.

K-fold Cross-validation in detail

Divide data into K roughly equal-sized parts ($K = 5$ here)

1	2	3	4	5
Validation	Train	Train	Train	Train

K-fold cross validation

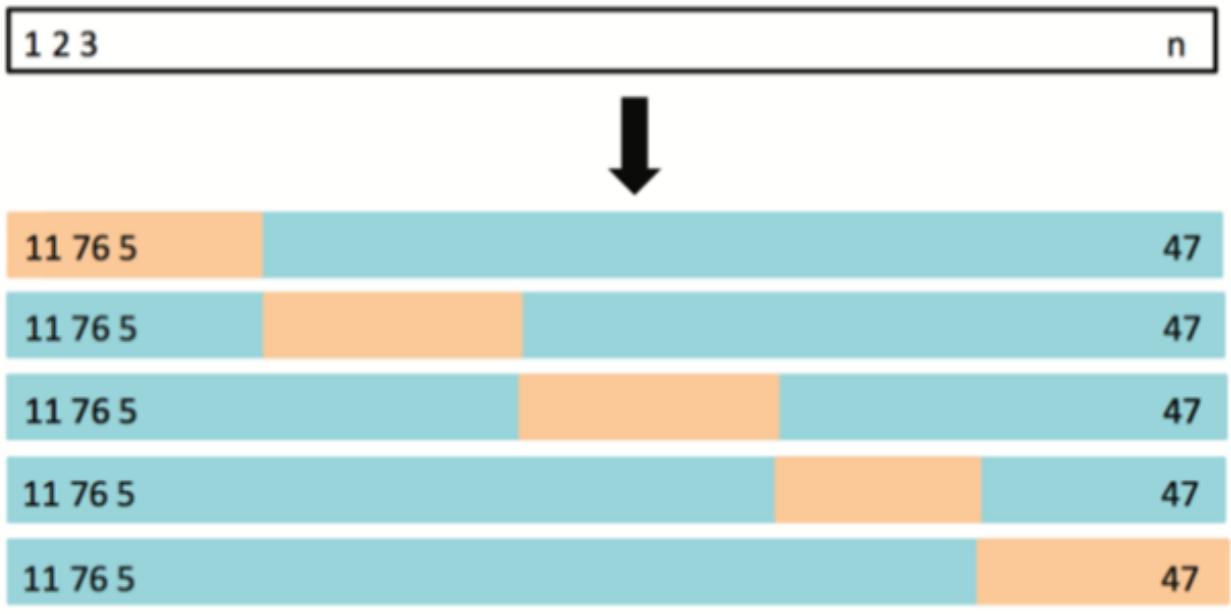


FIGURE 5.5. *A schematic display of 5-fold* CV. *A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting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

The details

- Let the K parts be $C_1, C_2, \ldots C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}_k$$

where $\text{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed.

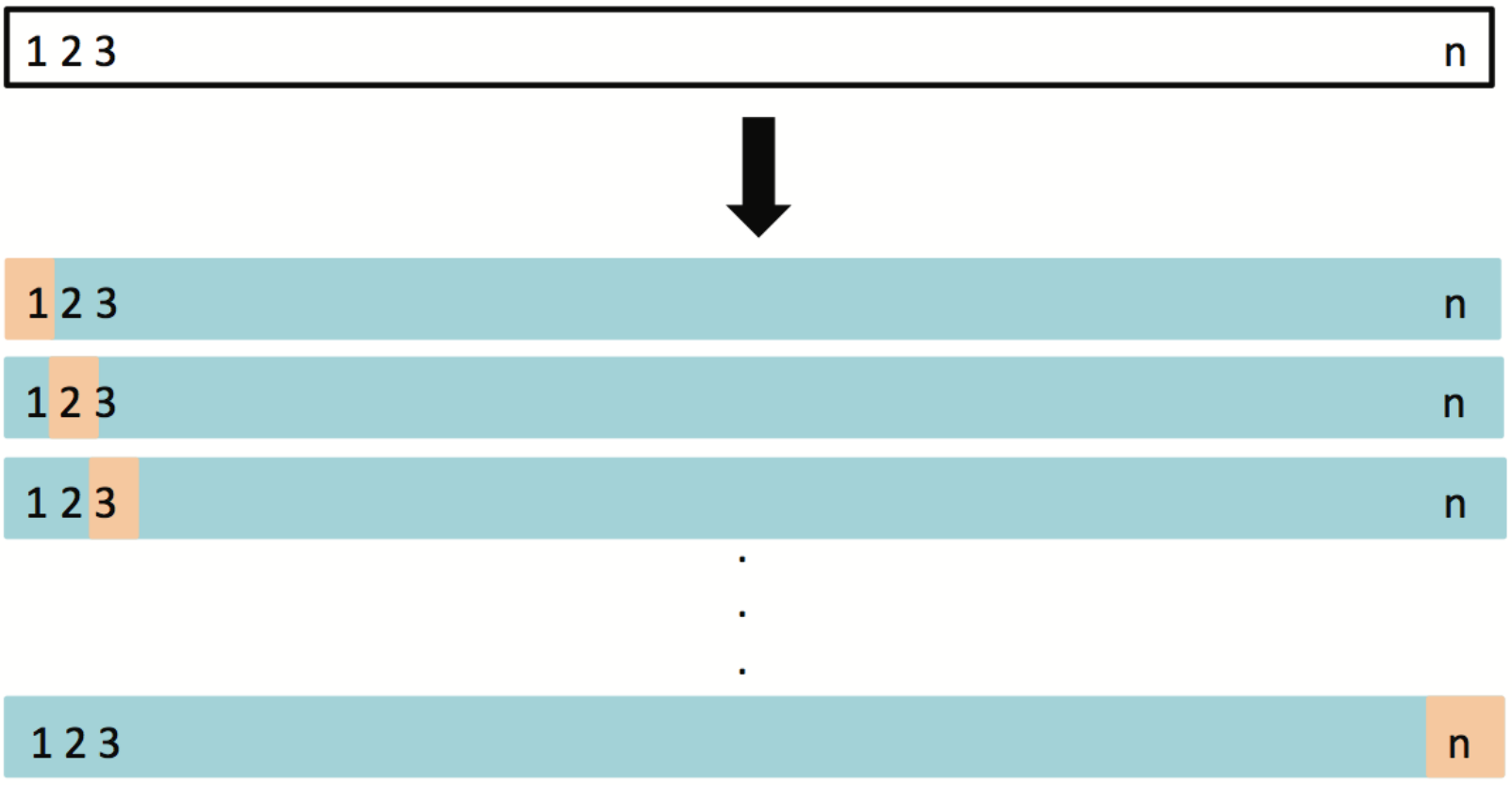
The details

- Let the K parts be $C_1, C_2, \ldots C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}_k$$

  where $\text{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed.
- Setting $K = n$ yields n-fold or *leave-one out cross-validation* (LOOCV).

leave one out cross validation



A nice special case!

- With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2,$$

  where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat" matrix; see book for details.) This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.

A nice special case!

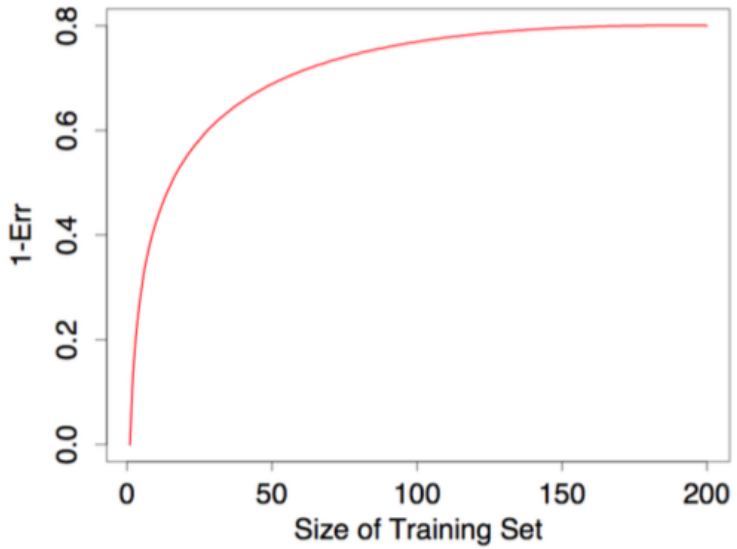
- With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2,$$

  where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat" matrix; see book for details.) This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.
- LOOCV sometimes useful, but typically doesn't *shake up* the data enough. The estimates from each fold are highly correlated and hence their average can have high variance.
- a better choice is $K = 5$ or 10.

How to choose K ?

- $K = N$, the cv estimator is approximately unbiased, but can have high variance because the N training sets are so similar. The computational burden is also considerable.
- $K = 5$, cv has lower variance. But bias could be a problem, depending on how the performance of the learning method varies with the size of the training set. (learning curve, see Fig 7.8)
- To summarize, if the learning curve has a considerable slope, five or tenfold cv will be overestimated.



Generalized cross-validation

- Generalized cross-validation provides a convenient approximation to leave-one out cross-validation, for linear fitting under squared-error loss.a linear fitting method is one for which we can write

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y} \tag{7.50}$$

Now for many linear fitting methods,

$$\frac{1}{N}\sum_{i=1}^{N}\left[y_i - \hat{f}^{-i}(x_i)\right]^2 = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2, \tag{7.51}$$

where S_{ii} is the ith diagonal element of $\mathbf{S}$ (see Exercise 7.3). The GCV approximation is

$$\mathrm{GCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \mathrm{trace}(\mathbf{S})/N}\right]^2. \tag{7.52}$$

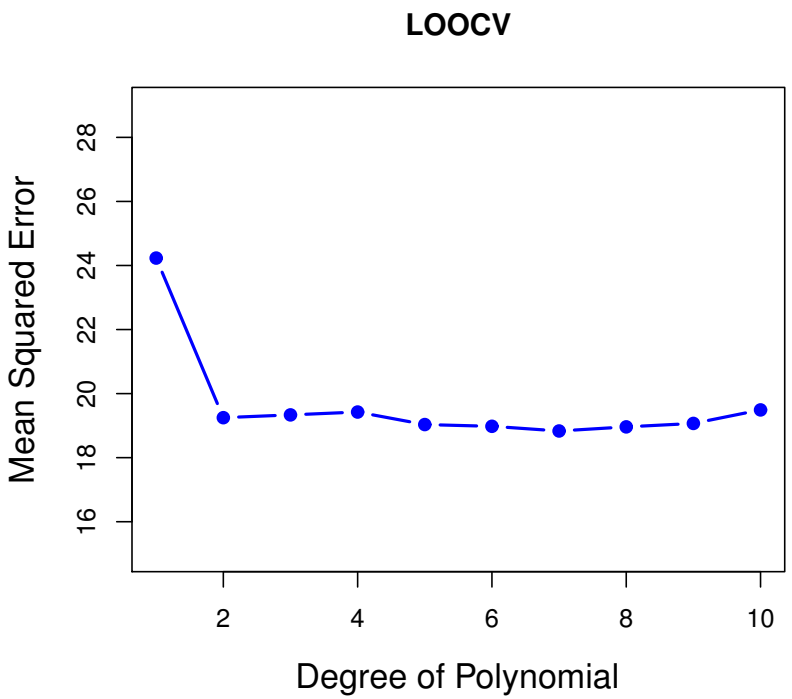
Cross-validation with tuning parameter

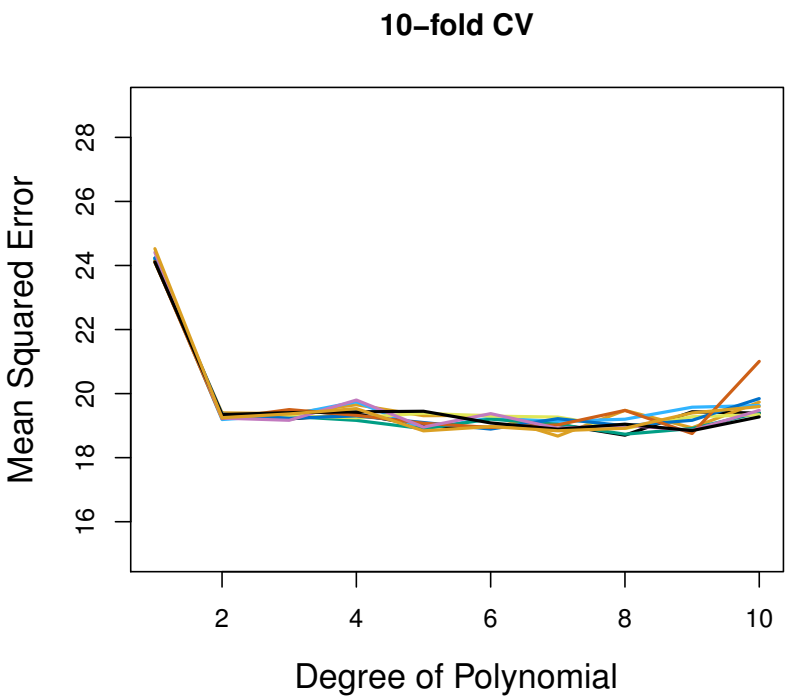
Given a set of models $f(x, \alpha)$ indexed by a tuning parameter α, denote by $\hat{f}^{-k}(x, \alpha)$ the α th model fit with the kth part of the data removed. Then for this set of models we define

$$\mathrm{CV}(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{f}^{-\kappa(i)}\left(x_i, \alpha\right)\right). \tag{7.49}$$

The function $\mathrm{CV}(\hat{f}, \alpha)$ provides an estimate of the test error curve, and we find the tuning parameter $\hat{\alpha}$ that minimizes it. Our final chosen model is $f(x, \hat{\alpha})$, which we then fit to all the data.

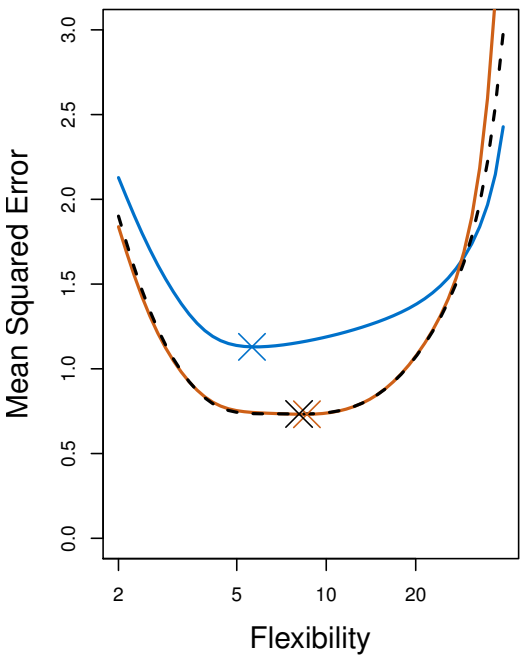
Auto data revisited

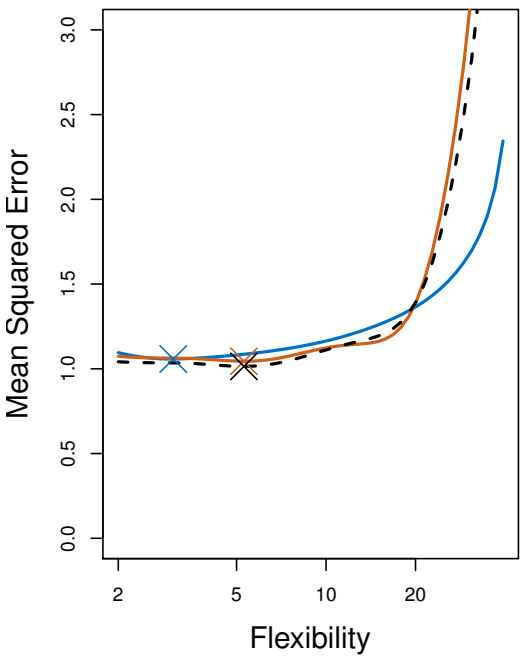


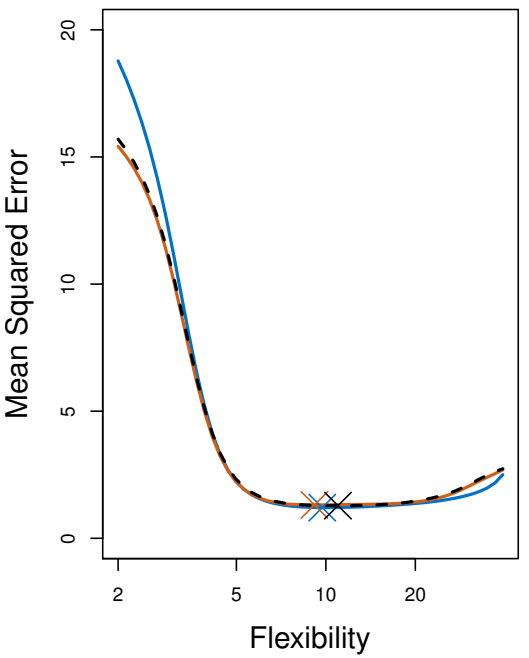


True and estimated test MSE for the simulated data

blue: true test MSE
black dashed: LOOCV
orange: 10-fold







Other issues with Cross-validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward.

Other issues with Cross-validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward. *Why?*

Other issues with Cross-validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward. *Why?*
- This bias is minimized when $K = n$ (LOOCV), but this estimate has high variance, as noted earlier.
- $K = 5$ or 10 provides a good compromise for this bias-variance tradeoff.

Cross-Validation for Classification Problems

- We divide the data into K roughly equal-sized parts $C_1, C_2, \ldots C_K$. C_k denotes the indices of the observations in part k. There are n_k observations in part k: if n is a multiple of K, then $n_k = n/K$.
- Compute

$$\mathrm{CV}_K = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{Err}_k$$

  where $\mathrm{Err}_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i)/n_k$.
- The estimated standard deviation of CV_K is

$$\widehat{\mathrm{SE}}(\mathrm{CV}_K) = \sqrt{\sum_{k=1}^{K} (\mathrm{Err}_k - \overline{\mathrm{Err}_k})^2/(K-1)}$$

- This is a useful estimate, but strictly speaking, not quite valid.

Cross-Validation for Classification Problems

- We divide the data into K roughly equal-sized parts $C_1, C_2, \ldots C_K$. C_k denotes the indices of the observations in part k. There are n_k observations in part k: if n is a multiple of K, then $n_k = n/K$.
- Compute

$$\mathrm{CV}_K = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{Err}_k$$

  where $\mathrm{Err}_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i)/n_k$.
- The estimated standard deviation of CV_K is

$$\widehat{\mathrm{SE}}(\mathrm{CV}_K) = \sqrt{\sum_{k=1}^{K} (\mathrm{Err}_k - \overline{\mathrm{Err}_k})^2/(K-1)}$$

- This is a useful estimate, but strictly speaking, not quite valid. *Why not?*

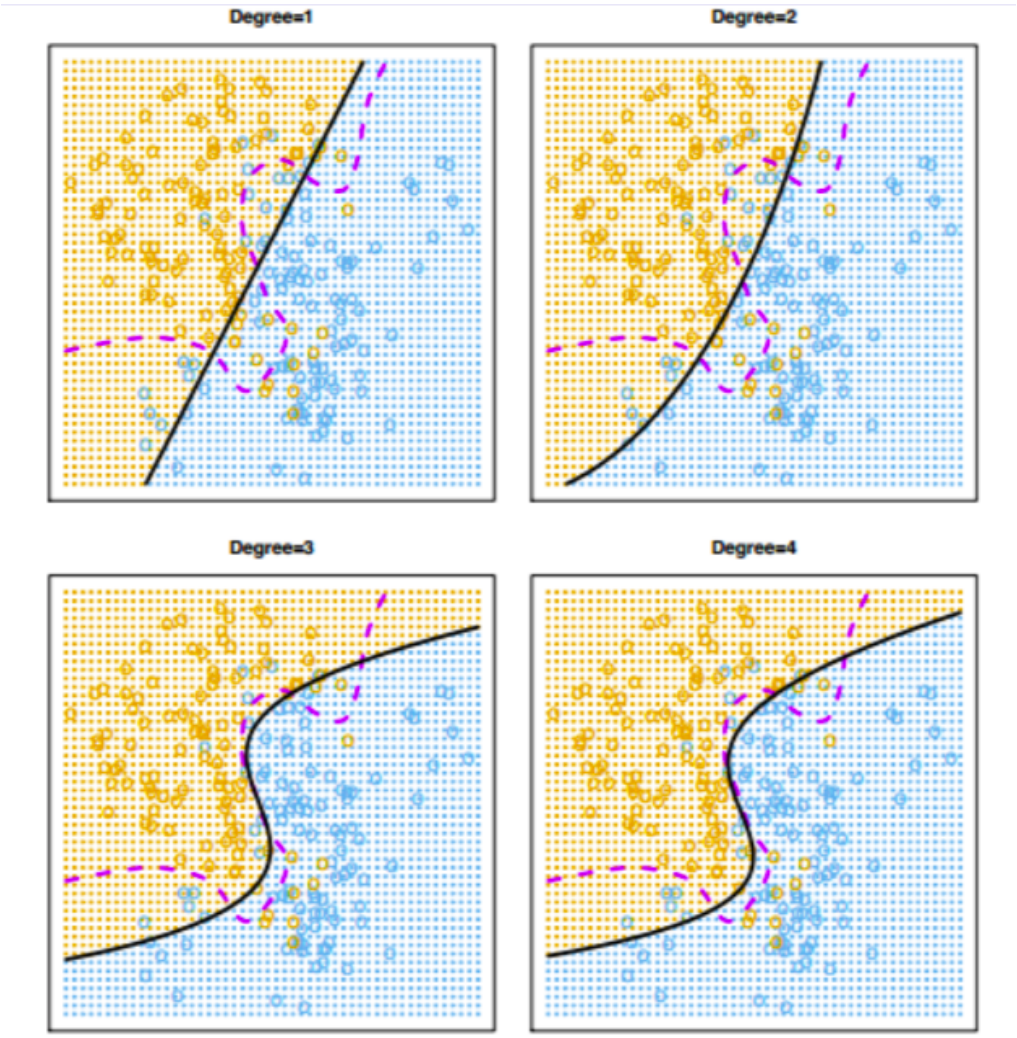


FIGURE 5.7. *Logistic regression fits on the two-dimensional classification data displayed in Figure 2.13. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from linear, quadratic, cubic and quartic (degrees 1-4) logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively* $0.201, 0.197, 0.160,$ *and* $0.162,$ *while the Bayes error rate is* $0.133.$

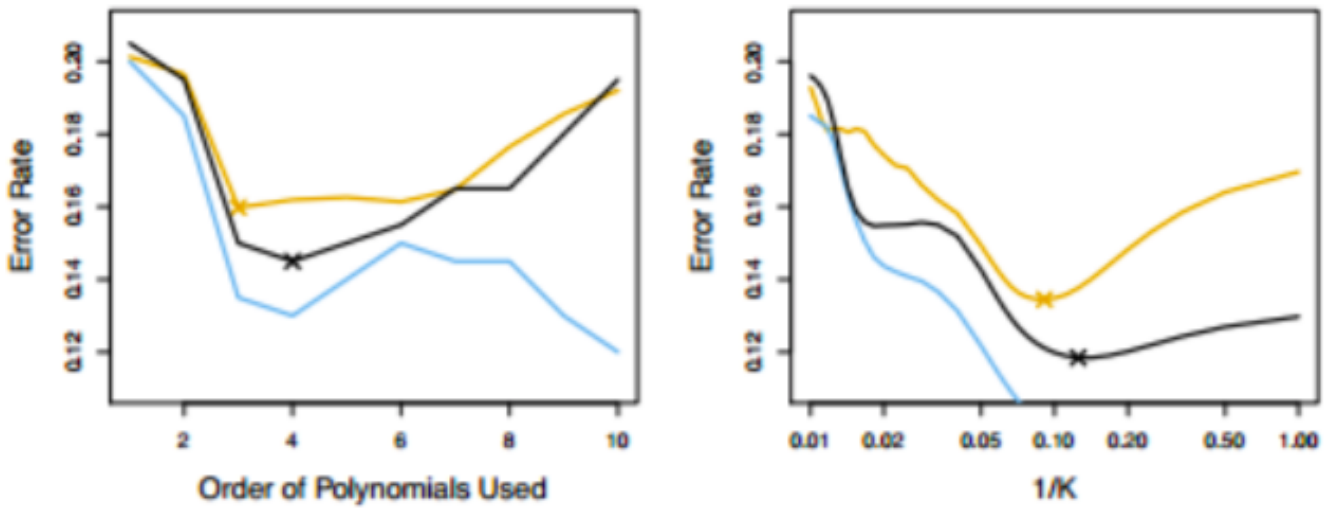


FIGURE 5.8. *Test error (brown), training error (blue), and 10-fold CV error (black) on the two-dimensional classification data displayed in Figure 5.7 . Left: Logistic regression using polynomial functions of the predictors. The onder of the polynomials used is displayed on the x-axis. Right: The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.*

Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.

  How do we estimate the test set performance of this classifier?

Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.

  How do we estimate the test set performance of this classifier?

  Can we apply cross-validation in step 2, forgetting about step 1?

NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero!

NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero! *Try to do this yourself*

NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero! *Try to do this yourself*
- We have seen this error made in many high profile genomics papers.

The Wrong and Right Way

- *Wrong:* Apply cross-validation in step 2.
- *Right:* Apply cross-validation to steps 1 and 2.

Cross-validation wrong or right?

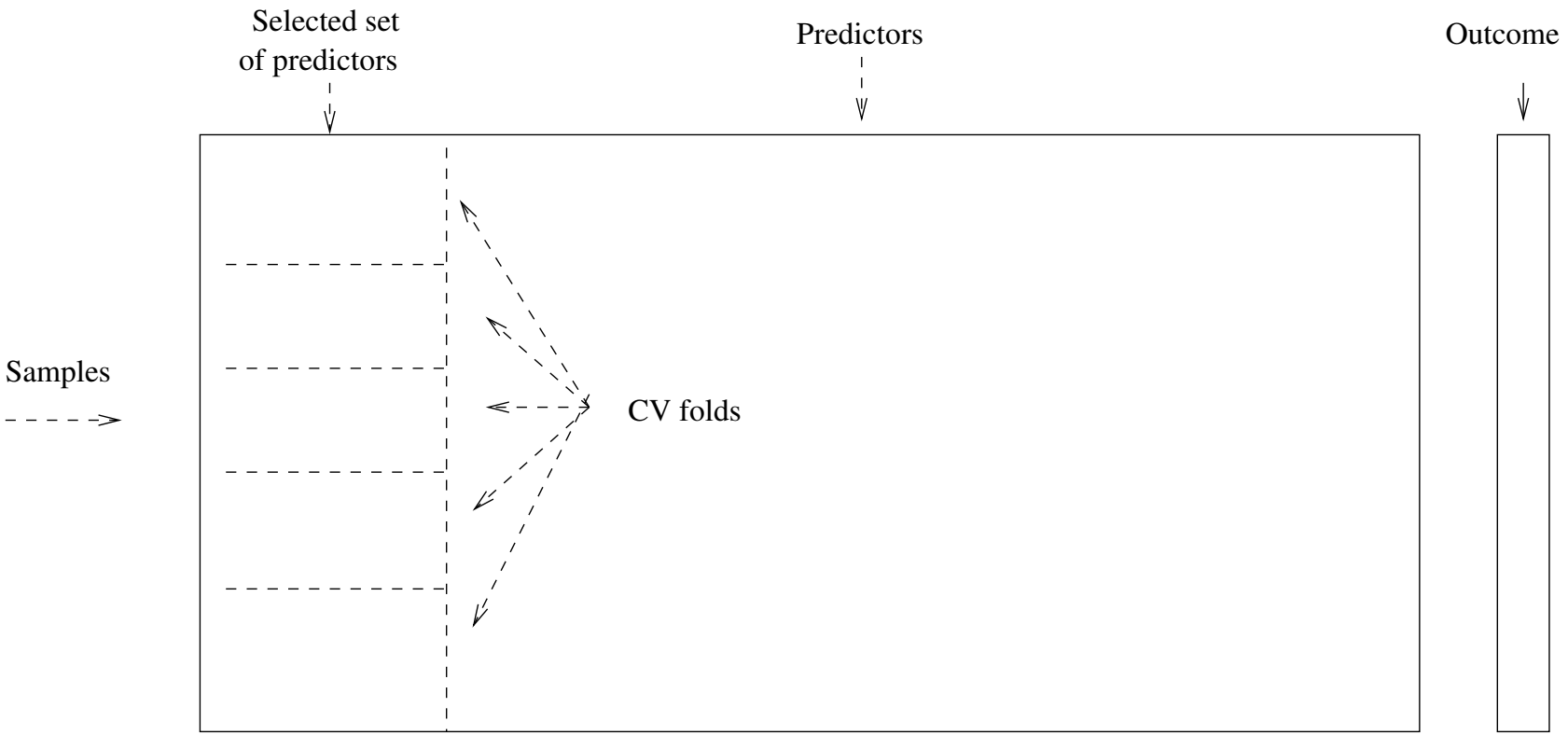
Consider a classification problem with a large number of predictors, as may arise, for example, in genomic or proteomic applications. A typical strategy for analysis might be as follows:

1. Screen the predictors: find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels

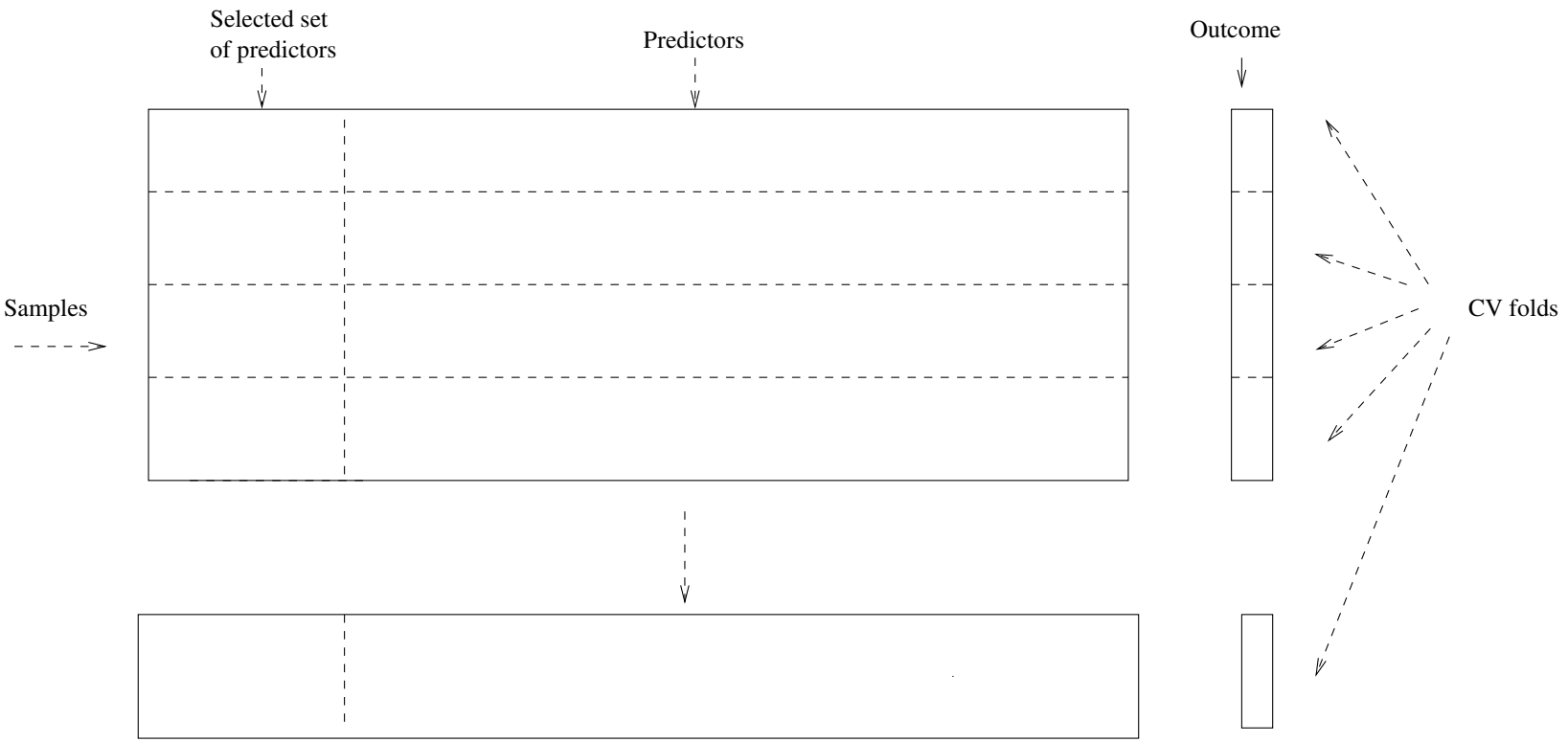
2. Using just this subset of predictors, build a multivariate classifier.

3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

Wrong Way


Selected set
of predictors
Predictors
Outcome
Samples
CV folds

Right Way

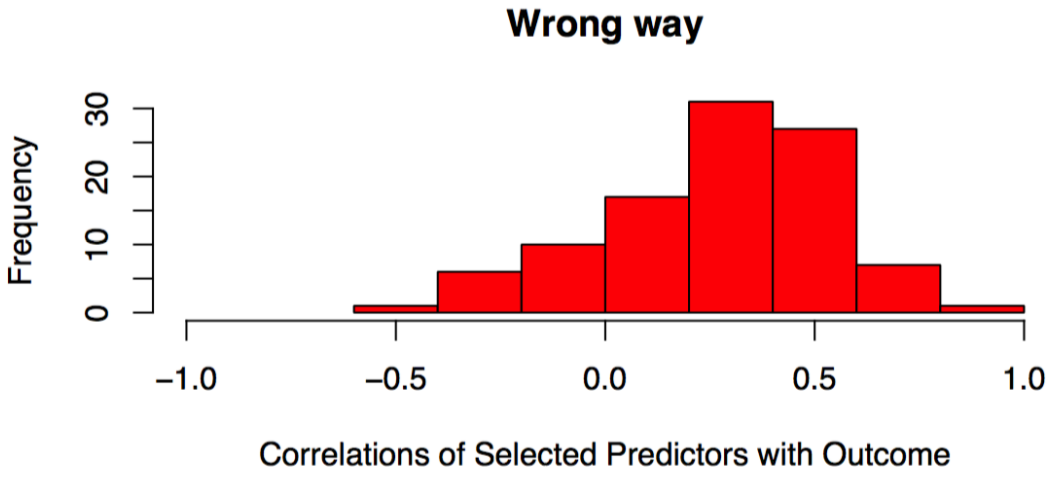

Selected set
of predictors
Predictors
Outcome
Samples
CV folds

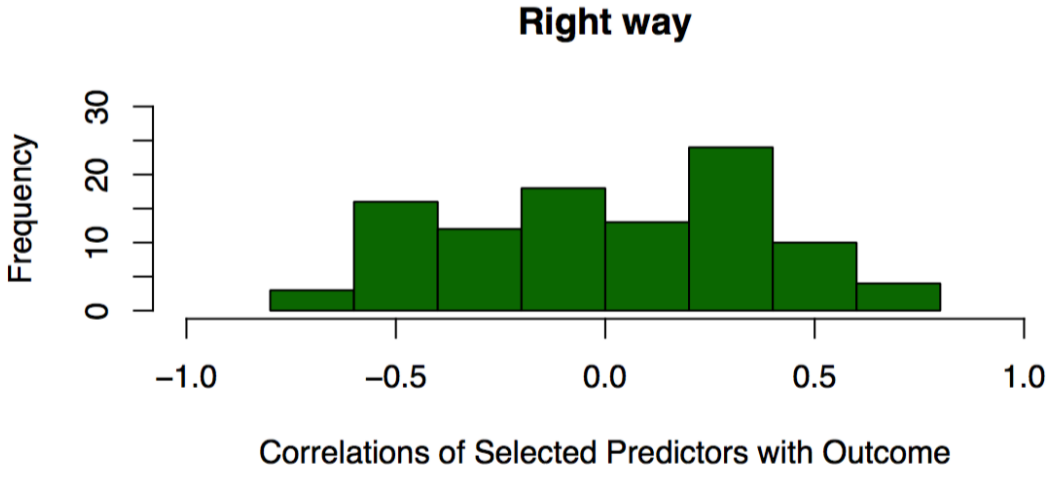
Cross-validation wrong or right?

Is this a correct application of cross-validation? Consider a scenario with $N = 50$ samples in two equal-sized classes, and $p = 5000$ quantitative predictors (standard Gaussian) that are independent of the class labels. The true (test) error rate of any classifier is 50%. We carried out the above recipe, choosing in step (1) the 100 predictors having highest correlation with the class labels, and then using a 1-nearest neighbor classifier, based on just these 100 predictors, in step (2). Over 50 simulations from this setting, the average CV error rate was 3%. This is far lower than the true error rate of 50%.

What has happened? The problem is that the predictors have an unfair advantage, as they were chosen in step (1) on the basis of *all of the samples*. Leaving samples out *after* the variables have been selected does not correctly mimic the application of the classifier to a completely independent test set, since these predictors "have already seen" the left out samples.

Cross-validation wrong or right?





Cross-validation wrong or right?

Here is the correct way to carry out cross-validation in this example:

1. Divide the samples into K cross-validation folds (groups) at random.
2. For each fold $k = 1, 2, \ldots, K$

(a) Find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k.

(b) Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.

(c) Use the classifier to predict the class labels for the samples in fold k.

Overall, samples must be "left out" before any selection or filtering steps are applied. But, initial unsupervised screening steps can be done before samples are left out. For example select with high variance.

The Bootstrap

- The *bootstrap* is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

Where does the name came from?

- The use of the term bootstrap derives from the phrase *to pull oneself up by one's bootstraps*, widely thought to be based on one of the eighteenth century "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:

  The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions, though the derivation is similar.

A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.
- We wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $\mathrm{Var}(\alpha X + (1 - \alpha)Y)$.

A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.
- We wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $\text{Var}(\alpha X + (1 - \alpha)Y)$.
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

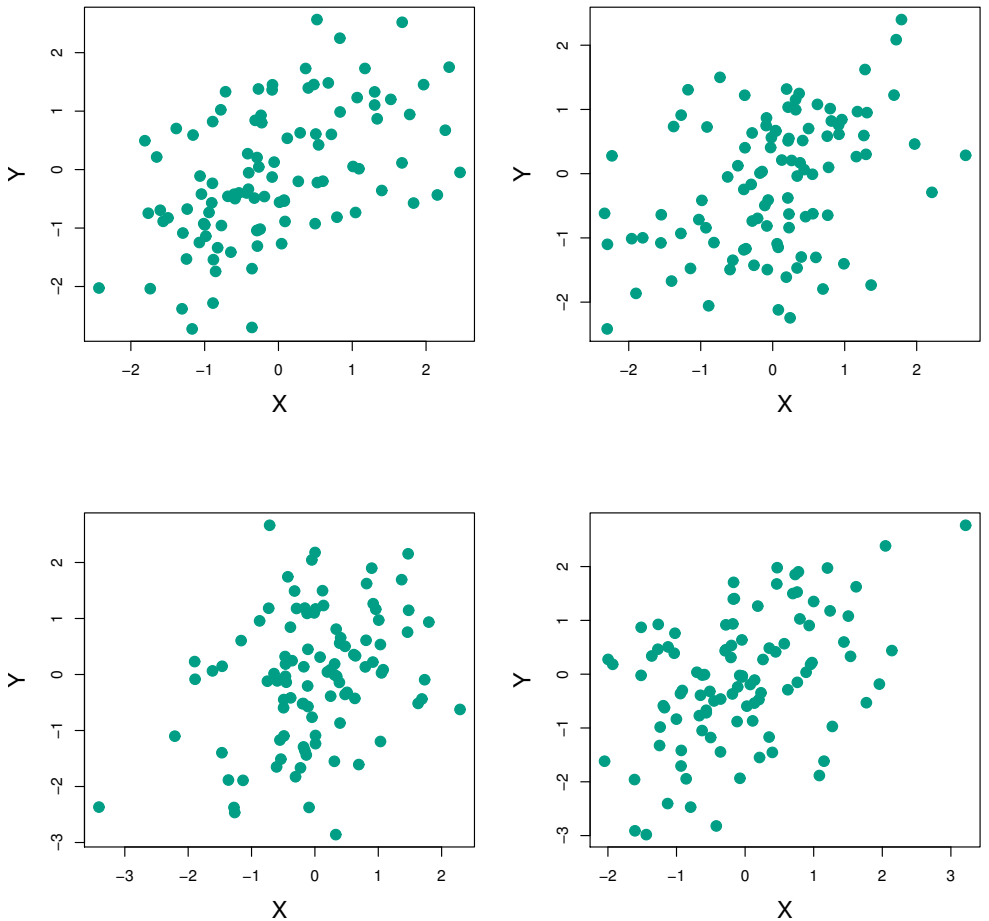
where $\sigma_X^2 = \text{Var}(X), \sigma_Y^2 = \text{Var}(Y)$, and $\sigma_{XY} = \text{Cov}(X, Y)$.

Example continued

- But the values of σ_X^2, σ_Y^2, and σ_{XY} are unknown.
- We can compute estimates for these quantities, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$, using a data set that contains measurements for X and Y.
- We can then estimate the value of α that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

Example continued



Each panel displays 100 *simulated returns for investments* X *and* Y. *From left to right and top to bottom, the resulting estimates for* α *are* 0.576, 0.532, 0.657, *and* 0.651.

Example continued

- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating α 1,000 times.
- We thereby obtained 1,000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$.
- The left-hand panel of the Figure on slide 29 displays a histogram of the resulting estimates.
- For these simulations the parameters were set to $\sigma_X^2 = 1, \sigma_Y^2 = 1.25$, and $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6 (indicated by the red line).

Example continued

- The mean over all 1,000 estimates for α is

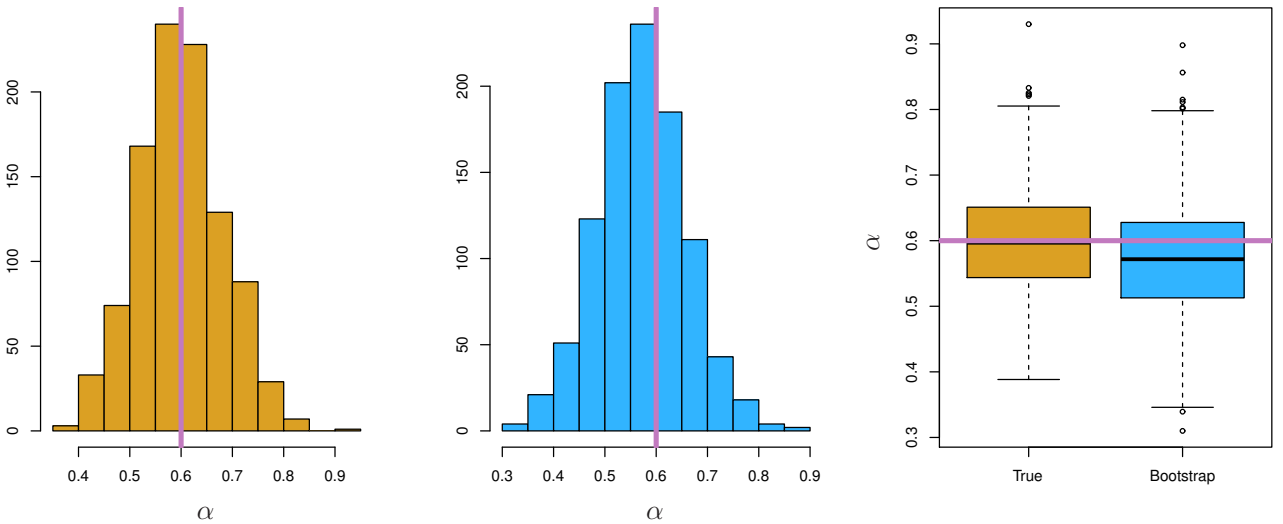
$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

very close to $\alpha = 0.6$, and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $\text{SE}(\hat{\alpha}) \approx 0.083$.
- So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

Results

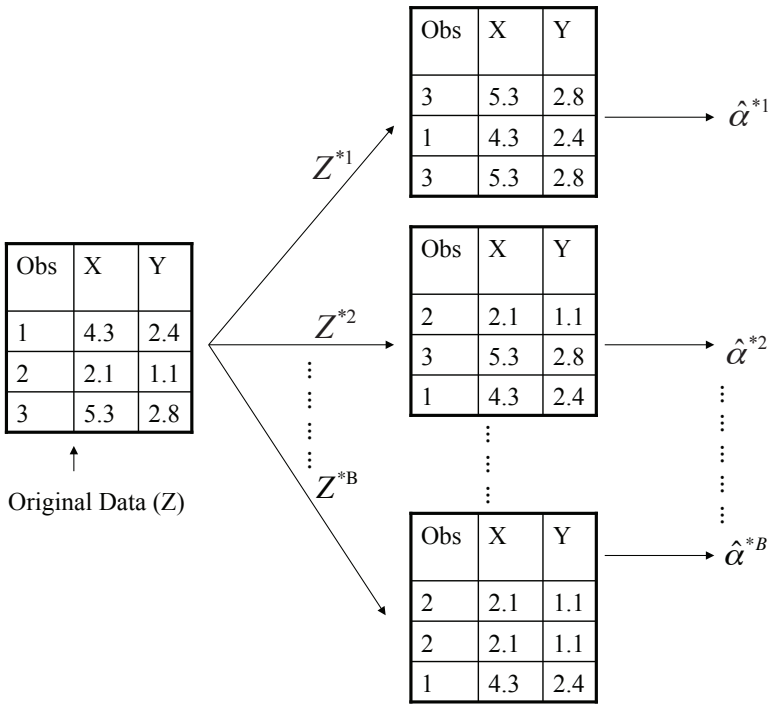


Left: A histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population. *Center:* A histogram of the estimates of α obtained from 1,000 bootstrap samples from a single data set. *Right:* The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α.

Now back to the real world

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set *with replacement*.
- Each of these "bootstrap data sets" is created by sampling *with replacement*, and is the *same size* as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Example with just 3 observations



A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α

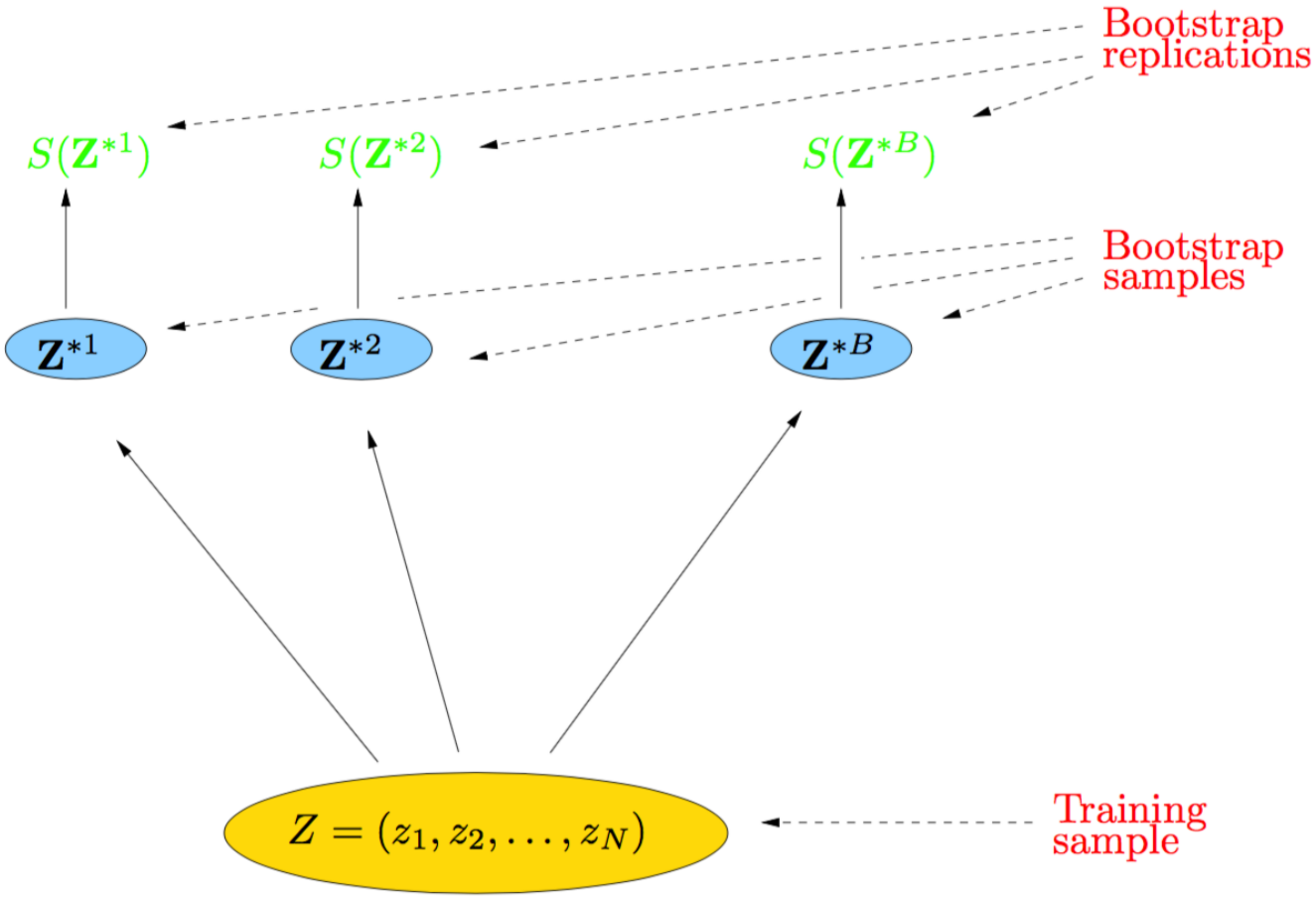
- Denoting the first bootstrap data set by Z^{*1}, we use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$
- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$, and B corresponding α estimates, $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$.
- We estimate the standard error of these bootstrap estimates using the formula

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^{*}\right)^2}.$$

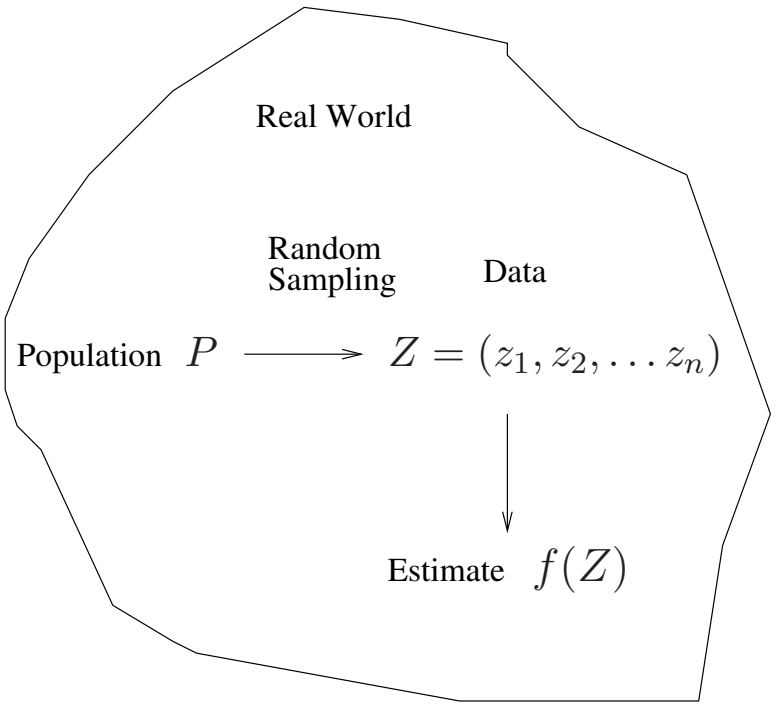
- This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set. See center and right panels of Figure on slide 29. Bootstrap results are in blue. For this example $\mathrm{SE}_B(\hat{\alpha}) = 0.087$.

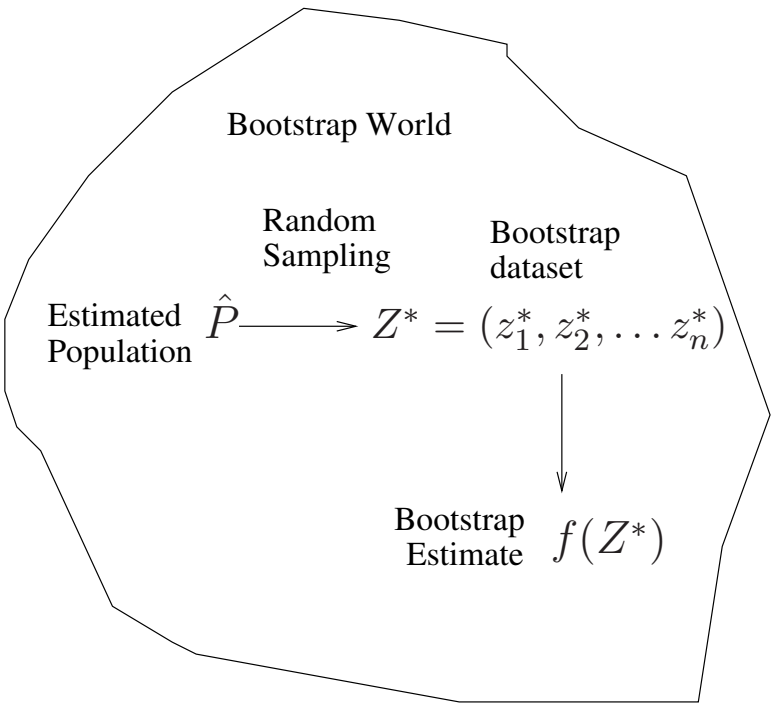
Bootstrap

- $\hat{Var}(S(Z)) = \frac{1}{B-1}\sum_{b=1}^{B}(S(Z^{*b}) - \bar{S}^*)^2$, where $\bar{S}^* = \sum_b (S(Z^{*b})/B)$



A general picture for the bootstrap





The bootstrap in general

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).

The bootstrap in general

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is $(.43, .72)$.
- This represents an approximate 90% confidence interval for the true α.

Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is $(.43, .72)$.
- This represents an approximate 90% confidence interval for the true α. *How do we interpret this confidence interval?*

Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is $(.43, .72)$.
- This represents an approximate 90% confidence interval for the true α. *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

test critical value

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success.

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample.

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error.

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*

Can the bootstrap estimate prediction error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around— with original sample = training sample, bootstrap dataset = validation sample— is worse!

Removing the overlap

- Can partly fix this problem by only using predictions for those observations that did not (by chance) occur in the current bootstrap sample.
- But the method gets complicated, and in the end, cross-validation provides a simpler, more attractive approach for estimating prediction error.

Bootstrap for prediction error

- Bootstrap error underestimate the true error.

$$\widehat{Err}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{*b}(x_i)\right)$$

- For example, y is class labels and independent with X. Then the true error rate is 0.5 , while the expecation of $\widehat{Err}_{boot}$ is $0.5 \times 0.368 = 0.184$. Underestimate.
- The leave one out bootstrap estimate of prediction error

$$\widehat{Err}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L\left(y_i, \hat{f}^{*b}(x_i)\right)$$

OOB:out of bag error

where C^{-i} is the set of indices of the bootstrap samples b that do not contain observation i.

- The leave one out bootstrap solves the overfitting problem suffered by $\widehat{Err}_{boot}$, but has the training set size bias, because the average number of distincet observations in each bootstrap sample is $0.632N$.

Bootstrap for prediction error

- The .632 estimator is designed to alleviate this bias.

$$\widehat{Err}^{(.632)} = 0.368\overline{err} + 0.632\widehat{Err}^{(1)}$$

- The .632 estimator works well in light fitting situations, but can break down in overfit ones. For example, Suppose we have two equal-size classes, with the targets independent of the class labels, and we apply a one-nearest neighbor rule. Then $\overline{err} = 0$, $\widehat{Err}^{(1)} = 0.5$, $\widehat{Err}^{(.632)} = 0.632 \times 0.5 = 0.316$. True error is 0.5.
- One can improve the .632 estimator by taking into account the amount of overfitting. Define γ to be the no-information error rate.

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} L\left(y_i, \hat{f}\left(x_{i'}\right)\right)$$

Bootstrap for prediction error

Using this, the *relative overfitting rate* is defined to be

$$\hat{R} = \frac{\widehat{\text{Err}}^{(1)} - \overline{\text{err}}}{\hat{\gamma} - \overline{\text{err}}}, \tag{7.60}$$

a quantity that ranges from 0 if there is no overfitting $\left(\widehat{\text{Err}}^{(1)} = \overline{\text{err}}\right)$ to 1 if the overfitting equals the no-information value $\hat{\gamma} - \overline{err}$. Finally, we define the".632+" estimator by

$$\begin{aligned} \widehat{\text{Err}}^{(.632+)} &= (1 - \hat{w}) \cdot \overline{\text{err}} + \hat{w} \cdot \widehat{\text{Err}}^{(1)} \\ \text{with } \hat{w} &= \frac{.632}{1 - .368\hat{R}}. \end{aligned} \tag{7.61}$$

The weight w ranges from .632 if $\hat{R} = 0$ to 1 if $\hat{R} = 1$, so $\widehat{\text{Err}}^{(.632+)}$ ranges from $\widehat{\text{Err}}^{(.632)}$ to $\widehat{\text{Err}}^{(1)}$. Again, the derivation of (7.61) is complicated: roughly speaking, it produces a compromise between the leave-one-out bootstrap and the training error rate that depends on the amount of overfitting. For the 1 -nearest-neighbor problem with class labels independent of the inputs, $\hat{w} = \hat{R} = 1$, so $\widehat{\text{Err}}^{(.632+)} = \widehat{\text{Err}}^{(1)}$, which has the correct expectation of 0.5. In other problems with less overfitting, $\widehat{\text{Err}}^{(.632+)}$ will lie somewhere between $\overline{err}$ and $\widehat{\text{Err}}^{(1)}$.

Random lasso

The Annals of Applied Statistics
2011, Vol. 5, No. 1, 468–485
DOI: 10.1214/10-AOAS377

RANDOM LASSO

BY SIJIAN WANG, BIN NAN[1], SAHARON ROSSET[2] AND JI ZHU[3]

University of Wisconsin, University of Michigan, Tel Aviv University and University of Michigan

We propose a computationally intensive method, the random lasso method, for variable selection in linear models. The method consists of two major steps. In step 1, the lasso method is applied to many bootstrap samples, each using a set of randomly selected covariates. A measure of importance is yielded from this step for each covariate. In step 2, a similar procedure to the first step is implemented with the exception that for each bootstrap sample, a subset of covariates is randomly selected with unequal selection probabilities determined by the covariates' importance. Adaptive lasso may be used in the second step with weights determined by the importance measures. The final set of covariates and their coefficients are determined by averaging bootstrap results obtained from step 2. The proposed method alleviates some of the limitations of lasso, elastic-net and related methods noted especially in the context of microarray data analysis: it tends to remove highly correlated variables altogether or select them all, and maintains maximal flexibility in estimating their coefficients, particularly with different signs; the number of selected variables is no longer limited by the sample size; and the resulting

Random lasso

ALGORITHM ("Generate" and "Select"). *Step* 1. Generating importance measures for all coefficients:

1a. Draw B bootstrap samples with size n by sampling with replacement from the original training data set.

1b. For the b_1th bootstrap sample, $b_1 \in \{1, \ldots, B\}$, randomly select q_1 candidate variables, and apply lasso to obtain estimators $\hat{\beta}_j^{(b_1)}$ for β_j, $j = 1, \ldots, p$. Estimators are zero for coefficients of those unselected variables, either outside the subset of q_1 variables, or excluded by lasso.

1c. Compute the importance measure of x_j by $I_j = |B^{-1} \sum_{b_1=1}^{B} \hat{\beta}_j^{(b_1)}|$.

Step 2. Selecting variables.

2a. Draw another set of B bootstrap samples with size n by sampling with replacement from the original training data set.

2b. For the b_2th bootstrap sample, $b_2 \in \{1, \ldots, B\}$, randomly select q_2 candidate variables with selection probability of x_j proportional to its importance I_j obtained in step 1c, and apply lasso (or adaptive lasso) to obtain estimators $\hat{\beta}_j^{(b_2)}$ for β_j, $j = 1, \ldots, p$. Estimators are zero for coefficients of those unselected variables, either outside the subset of q_2 variables, or excluded by lasso.

2c. Compute the final estimator $\hat{\beta}_j$ of β_j by $\hat{\beta}_j = B^{-1} \sum_{b_2=1}^{B} \hat{\beta}_j^{(b_2)}$.

Random block lasso

358 Biometrical Journal **59** (2017) 2, 358–376 DOI: 10.1002/bimj.201600052

Analyzing large datasets with bootstrap penalization

Kuangnan Fang[1] and **Shuangge Ma***[,1,2]

[1] Department of Statistics, Xiamen University, Xiamen, Fujian, China
[2] Department of Biostatistics, Yale University, New Haven, CT 06520, USA

Received 27 February 2016; revised 31 August 2016; accepted 1 September 2016

Data with a large p (number of covariates) and/or a large n (sample size) are now commonly encountered. For many problems, regularization especially penalization is adopted for estimation and variable selection. The straightforward application of penalization to large datasets demands a "big computer" with high computational power. To improve computational feasibility, we develop bootstrap penalization, which dissects a big penalized estimation into a set of small ones, which can be executed in a highly parallel manner and each only demands a "small computer". The proposed approach takes different strategies for data with different characteristics. For data with a large p but a small to moderate n, covariates are first clustered into relatively homogeneous blocks. The proposed approach consists of two sequential steps. In each step and for each bootstrap sample, we select blocks of covariates and run penalization. The results from multiple bootstrap samples are pooled to generate the final estimate. For data with a large n but a small to moderate p, we bootstrap a small number of subjects, apply penalized estimation, and then conduct a weighted average over multiple bootstrap samples. For data with a large p and a large n, the natural marriage of the previous two methods is applied. Numerical studies, including simulations and data analysis, show that the proposed approach has computational and numerical advantages over the straightforward application of penalization. An R package has been developed to implement the proposed methods.

Keywords: Bootstrap; Computational feasibility; Large datasets; Penalization.

Random block lasso

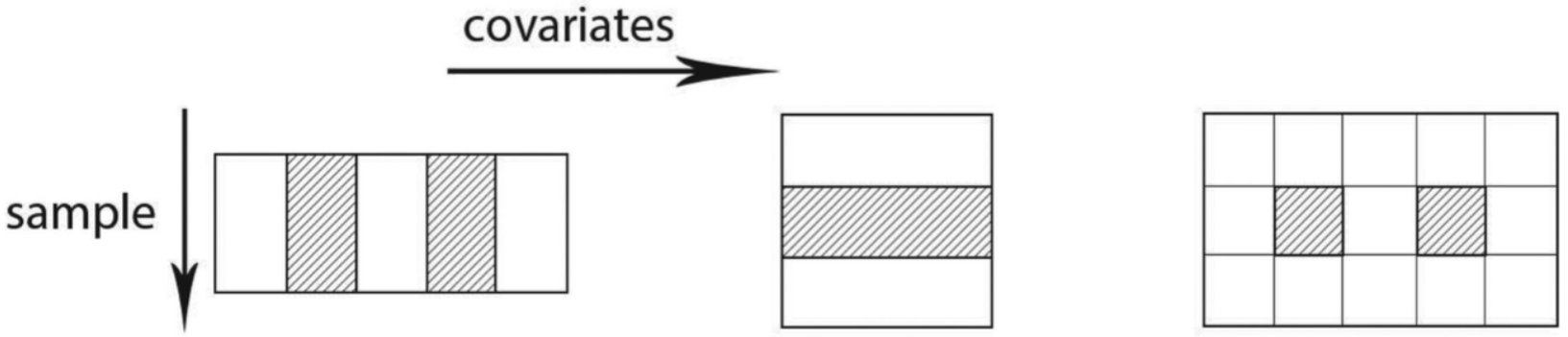


Figure 1 Analysis scheme for data with (a) a large p and a small to moderate n (left), (b) a large n and a small to moderate p (middle), and (c) a large p and a large n (right). The shaded areas represent data analyzed in one bootstrap run.

Random block lasso

3.1 Bootstrap penalization for data with a large p

With such data, the key is to reduce p to a more manageable level. The proposed method is realized in three steps.

Step 1: Cluster the p covariates into K nonoverlapping blocks.

With a specific clustering approach and number of blocks k, denote $(C_1, \ldots, C_k)$ as the index sets of resulted blocks. $C_i \cap C_j = \emptyset$ for $i \neq j$, and $\sum_j |C_j| = p$. K, the optimal number of blocks, is chosen by minimizing the Dunn Index (Dunn, 1974; Handl et al., 2005) defined as $\frac{\min_{j \neq l}(\min_{u \in C_j, v \in C_l} dist(u,v))}{\max_j diam(C_j)}$, where $dist(u, v)$ is the distance between covariates u and v, and $diam(C_j)$ is the maximum distance between any two covariates in C_j. With the Dunn Index, we generate nonoverlapping blocks with the highest degree of compactness and separation.

Random block lasso

Step 2: Generate an importance measure for each block.

1. Draw B_2 bootstrap samples—each with sample size n—by sampling with replacement from the original data.
2. For $b_2 = 1$ to B_2:
   (a) Select k_1 candidate blocks at random from the K blocks. Denote $E_1^{b_2} \subseteq \{1, 2, \ldots, K\}$ as the index set of selected blocks of the b_2-th bootstrap sample and $H_1^{b_2}$ as the corresponding index set of selected covariates.
   (b) Apply Lasso to the bootstrap sample and generate the estimate by minimizing

$$\sum_{i=1}^{n} \left(y_i^{b_2} - \alpha - \sum_{j \in H_1^{b_2}} \beta_j x_{ij}^{b_2} \right)^2 + \lambda \sum_{j \in H_1^{b_2}} |\beta_j|.$$

   Here we use the superscript "b_2" to denote the b_2-th bootstrap sample. Note that only coefficients for covariates in $H_1^{b_2}$ are estimated. Set the estimate for $\beta_j = 0$ for $j \in \{1, \ldots, p\} \backslash H_1^{b_2}$. Denote the resulted estimate as $\hat{\beta}_j^{(b_2)}$.
3. Compute the importance measure of block $k (= 1, \ldots, K)$ as

$$I_k = \sqrt{\sum_{j \in C_k} \left(B_2^{-1} \sum_{b_2=1}^{B_2} \hat{\beta}_j^{(b_2)} \right)^2}.$$

Random block lasso

Step 3: Generate the final selection and estimation results.

1. Draw B_3 bootstrap samples—each with sample size n—by sampling with replacement from the original data.
2. For $b_3 = 1$ to B_3:
   (a) Select k_2 candidate blocks from the K blocks with selection probabilities proportional to the importance measure I_k's obtained in Step 2. Denote $E_2^{b_3} \subseteq \{1, 2, \ldots, K\}$ as the index set of selected blocks of the b_3-th bootstrap sample and $H_2^{b_3}$ as the corresponding index set of selected covariates.
   (b) Apply Lasso to the bootstrap sample and generate the estimate by minimizing
   $$\sum_{i=1}^{n} \left(y_i^{b_3} - \alpha - \sum_{j \in H_2^{b_3}} \beta_j x_{ij}^{b_3} \right)^2 + \lambda \sum_{j \in H_2^{b_3}} |\beta_j|.$$
   Here we use the superscript "b_3" to denote the b_3-th bootstrap sample. Set the estimate for $\beta_j = 0$ for $j \in \{1, \ldots, p\} \backslash H_2^{b_3}$. Denote the resulted estimate as $\hat{\beta}_j^{(b_3)}$.
3. Conduct stability-based selection. Specifically, set $\hat{\beta}_j = 0$ if $B_3^{-1} \sum_{b_3=1}^{B_3} I(\hat{\beta}_j^{(b_3)} \neq 0) < \pi_1$ for $j = 1, \ldots, p$.
4. For $j = 1, \ldots, p$, if $\hat{\beta}_j \neq 0$ from Step 3, calculate the final estimate as $\hat{\beta}_j = B_3^{-1} \sum_{b_3=1}^{B_3} \hat{\beta}_j^{(b_3)}$.

Random block lasso

3.3 Bootstrap penalization for data with a large p and a large n

When both the number of covariates and sample size are large, the methods developed in the previous two sections can be naturally "combined" for analysis. The proposed method consists of the following steps.

1. Generate S bootstrap samples, each with size $m(<< n)$, by sampling without replacement from the original data. Denote the subject index of the s-th bootstrap sample as I_s.
2. For $s = 1$ to S:
   (a) Apply the block bootstrap penalization method developed in Section 3.1 to the s-th bootstrap sample, and obtain the estimate as $\hat{\boldsymbol{\beta}}^{(s)}$.
   (b) Compute the prediction mean squared error as
   $$PMSE^s = (\boldsymbol{y}_{(I\setminus I_s)} - \hat{\alpha}^{(s)} - X_{(I\setminus I_s)}\hat{\boldsymbol{\beta}}^{(s)})^T(\boldsymbol{y}_{(I\setminus I_s)} - \hat{\alpha}^{(s)} - X_{(I\setminus I_s)}\hat{\boldsymbol{\beta}}^{(s)})/|I \setminus I_s|.$$
3. Conduct stability-based selection. Specifically, set $\hat{\beta}_j = 0$ if $S^{-1}\sum_{s=1}^{S} I(\hat{\beta}_j^{(s)} \neq 0) < \pi_3$ for $j = 1, \ldots, p$. π_3 is chosen using a BIC-type criterion as previously described.
4. If $\hat{\beta}_j \neq 0$ from the previous step, then calculate the final estimate as $\hat{\beta}_j = \frac{1}{S}\sum_{s=1}^{S} w^s \hat{\beta}_j^{(s)}$. The weight w^s is computed as $w^s = \frac{max\left(\sqrt{PMSE_0^s} - \sqrt{PMSE^s}, 0\right)}{\sum_{s=1}^{S} max\left(\sqrt{PMSE_0^s} - \sqrt{PMSE^s}, 0\right)}$, where $PMSE_0^s = (\boldsymbol{y}_{(I\setminus I_s)} - \bar{\boldsymbol{y}}_{(I\setminus I_s)})^T(\boldsymbol{y}_{(I\setminus I_s)} - \bar{\boldsymbol{y}}_{(I\setminus I_s)})/|I \setminus I_s|$ is the prediction mean squared error of the null model without covariates.