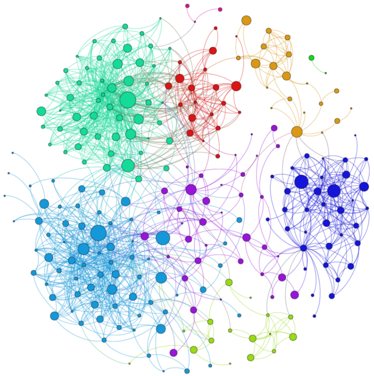
Model Selection and Regularization

Jiaming Mao

Xiamen University



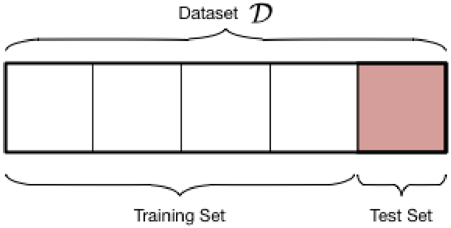
Model Assessment and Model Selection

- **Model assessment**: evaluating a model's performance
- **Model selection**: choosing the model with the best performance
- Both model assessment and model selection require estimate of a model's out-of-sample error.

Model Assessment

To evaluate the performance of a model $\mathcal{H}$, we can *randomly* split our data into two parts:

- **Training set**: used to fit the model (select g from $\mathcal{H}$)
- **Test set**: used to see how good the fit is (evaluate g's performance)



Training versus Testing

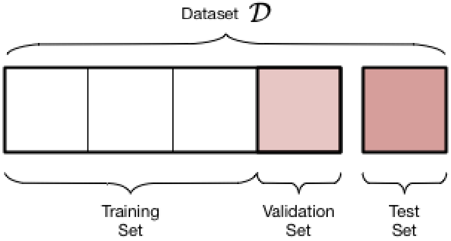
$$\text{Training:} \quad \Pr\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 4 m_{\mathcal{H}}(2N)\, e^{-\frac{1}{8}\epsilon^2 N} \qquad (1)$$

$$\text{Testing:} \quad \Pr\left(\left|E_{test}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N} \qquad (2)$$

- The generalization bound for test error $E_{test}(g)$ is much tighter.
- The test set is not biased, whereas the training set has an optimistic bias, since it is used to choose a hypothesis that looks good *on it*.
- The price for a test set is fewer data for training.

Model Selection

To choose the best model $\mathcal{H}_*$ among a set of models $\mathbb{H} = \{\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_n\}$, we can *randomly* split our data into three parts: a training set, a test set, and a **validation (hold-out) set**:



Model Selection

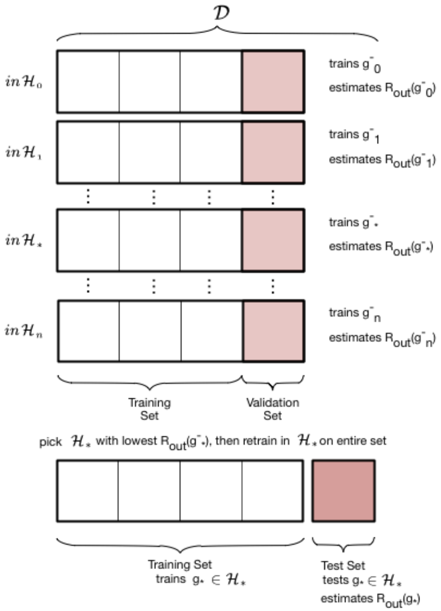
Then follow the following steps:

1. Fit each model $\mathcal{H}_i$ on the training set and obtain g_i for $i = 0, 1, \ldots, n$
2. Use the validation set to evaluate the performance of each g_i and select the hypothesis $g_* \in \mathcal{H}_*$ with the smallest validation-set error – $\mathcal{H}_*$ is our selected model.
3. Combine the training set and the validation set. Refit $\mathcal{H}_*$ on this larger set[1] to obtain g_{**} – this is our final selected hypothesis.
4. Assess the performance of g_{**} on the test set.

This is called the **validation set approach**.

[1] which corresponds to the training set in model assessment, since once we have picked a model – here $\mathcal{H}_*$ – what is left to do is model assessment.

Model Selection



$\mathcal{D}$
$in\,\mathcal{H}_0$
trains g^-_0
estimates $R_{out}(g^-_0)$
$in\,\mathcal{H}_1$
trains g^-_1
estimates $R_{out}(g^-_1)$
$in\,\mathcal{H}_*$
trains g^-_*
estimates $R_{out}(g^-_*)$
$in\,\mathcal{H}_n$
trains g^-_n
estimates $R_{out}(g^-_n)$
Training Set
Validation Set
pick $\mathcal{H}_*$ with lowest $R_{out}(g^-_*)$, then retrain in $\mathcal{H}_*$ on entire set
Training Set trains $g_* \in \mathcal{H}_*$
Test Set tests $g_* \in \mathcal{H}_*$ estimates $R_{out}(g_*)$

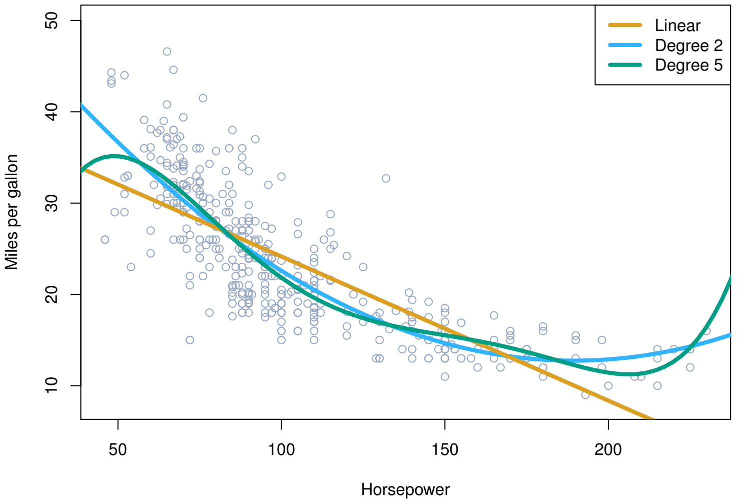
Model Selection

- A validation set is needed because we cannot use the test set to both help select the best model and assess the performance of the final hypothesis.
- Once a data set has been used in the learning process, it is "contaminated" – it obtains an *optimistic* bias, and the error calculated on the data set no longer has the tight bound in (2).

Model Selection

- Often times, the different models in $\mathbb{H}$ can be indexed by a complexity parameter λ, which we call a **hyperparameter**. Choosing the best model amounts to finding the best value of λ.
  - For example, if we are choosing among polynomial models of varying degrees, then λ is the degree of the polynomial.
- Using the validation set approach, the training set is used to fit each model with a given λ, the validation set is the set on which λ itself is fit, while the test set is used to estimate the true out-of-sample performance of the final hypothesis.

MPG and Horsepower



MPG and Horsepower

- Want to compare linear vs higher-order polynomial terms in a linear regression of miles per gallon (MPG) on horsepower on a data set of 392 vehicles.

- Suppose there exists an independent test data set somewhere else, so we only need this data set for model selection (without assessment). Then we can 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.

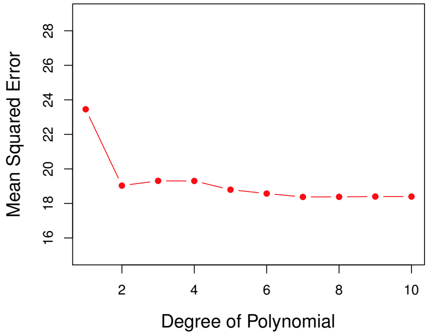
MPG and Horsepower

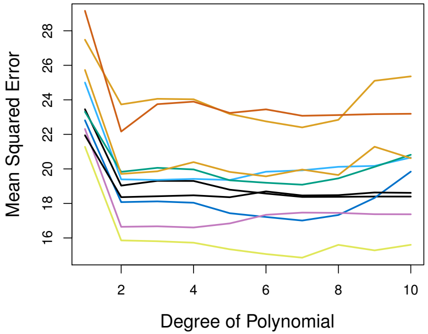
```r
require(ISLR) # contains the data set 'Auto'
attach(Auto)
train <- sample(392,196) # draw a random subset from the sample
fit <- lm(mpg ~ horsepower, subset=train)
yhat <- predict(fit,Auto)
V.e <- (mpg - yhat)[-train] # validation set error
V.MSE <- mean(V.e^2) # validation set MSE
V.MSE

## [1] 25.72651
```

MPG and Horsepower

```r
V.MSE <- rep(0,5)
for (i in 1:5){
  fit <- lm(mpg ~ poly(horsepower,i), subset=train)
  V.e <- (mpg - predict(fit,Auto))[-train]
  V.MSE[i] <- mean(V.e^2)
}
V.MSE

## [1] 25.72651 20.43036 20.38533 20.30902 19.79689
```

MPG and Horsepower





Left: Validation-set MSE for a single split into training and validation data sets. Right: The validation method was repeated ten times, each time using a different random split.

The Validation Set Approach

Problems:

- Results can be highly variable, depending on which observations are included in the training set and which are in the validation set.
- The larger the validation set is, the smaller the training set has to be, hence the better the validation set errors are as estimates of true out-of-sample errors, but the poorer the model fits produced by the training set are.
- Conversely, the smaller the validation set, the better the model fits on the training set, but the poorer the validation set errors are as estimates of true out-of-sample errors.

Cross Validation

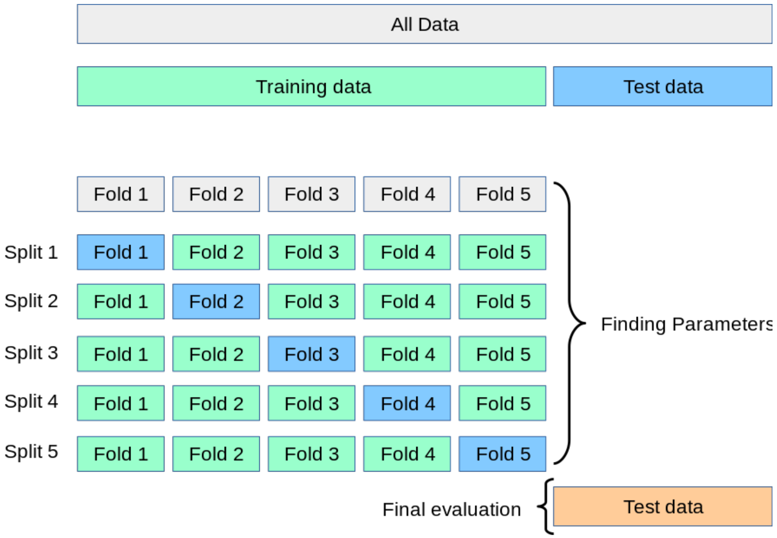
- The results of the validation set approach are variable due to the whims of a single random split. Thus, it might be better to randomly split the data multiple times and average the results – this is what **cross-validation** does.

- Like the bootstrap, cross-validation is a **resampling method**: these 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.

Cross Validation

- The cross-validation approach splits the original data into two parts: a training set and a test set, with no separate validation set.
- The training set is then randomly divided into K equal-sized folds.
- In turn, training is performed on $K-1$ folds (combined), and the remainder fold is used for evaluation. The K evaluation results are then averaged to produce an estimate of a model's out-of-sample error.

Cross Validation



source

Cross Validation

The $K-$fold cross-validation error of a model $\mathcal{H}$ is

$$CV_{(K)}\left(\mathcal{H}\right) = \frac{1}{K}\sum_{k=1}^{K} E_{in}^{(k)}\left(g^{(-k)}\left(\mathcal{H}\right)\right)$$

, where $g^{(-k)}\left(\mathcal{H}\right)$ is the hypothesis selected by fitting $\mathcal{H}$ on the training data with the k^{th} fold removed, and $E_{in}^{(k)}\left(.\right)$ is the error calculated on the k^{th} fold data[2].

[2]Note that $g^{(-k)}\left(\mathcal{H}\right)$ can be different for different k, hence the cross-validation error is obtained by averaging over the performance of different hypotheses.

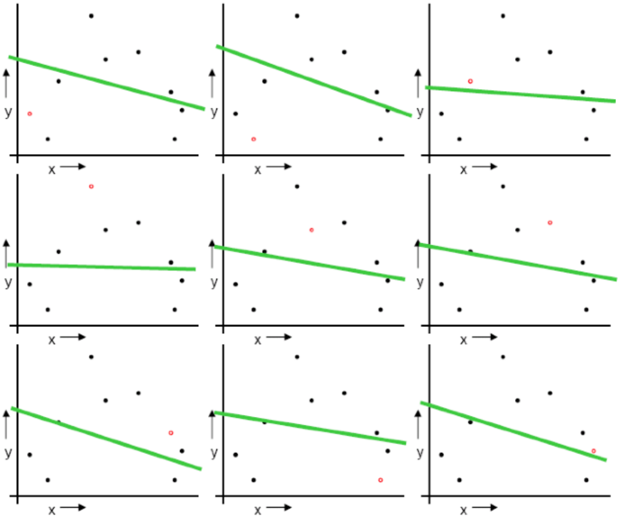
Cross Validation

Setting $K = N$ yields the **leave-one out cross-validation** (**LOOCV**) error:

$$CV_{(N)}(\mathcal{H}) = \frac{1}{N}\sum_{i=1}^{N} E_{in}^{(i)}\left(g^{(-i)}(\mathcal{H})\right)$$

, in which case $g^{(-i)}(\mathcal{H})$ is the result of fitting $\mathcal{H}$ on the entire training set with only the i^{th} data point removed, and $E_{in}^{(i)}(.)$ is the difference between the prediction made by $g^{(-i)}(\mathcal{H})$ on the i^{th} data point and its true value, according to the loss function used.

LOOCV



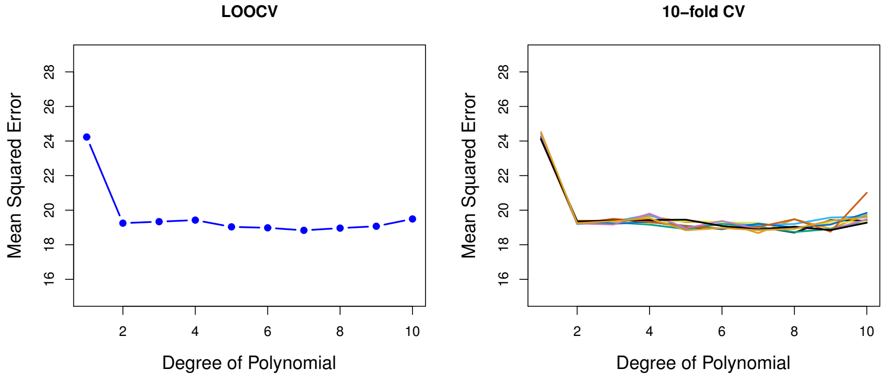
MPG and Horsepower

```r
## LOOCV
require(boot)
LOOCV.MSE <- rep(0,5)
for (i in 1:5){
  fit <- glm(mpg ~ poly(horsepower,i))
  LOOCV.MSE[i] <- cv.glm(Auto,fit)$delta[1]
}
LOOCV.MSE

## [1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

MPG and Horsepower

```r
## 10-fold CV
CV.MSE <- rep(0,5)
for (i in 1:5){
  fit <- glm(mpg ~ poly(horsepower,i))
  CV.MSE[i] <- cv.glm(Auto,fit,K=10)$delta[1]
}
CV.MSE

## [1] 24.17328 19.27789 19.43292 19.42324 19.03107
```

MPG and Horsepower

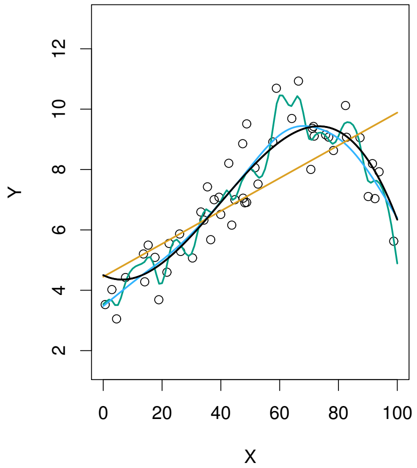


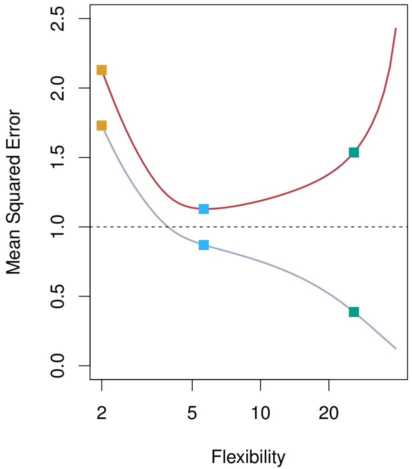
Left: The LOOCV error curve. Right: 10-fold CV was run nine separate times, each with a different random split of the data into ten folds. The figure shows the nine slightly different CV error curves.

Choice of K

- When N is large, LOOCV can be computationally expensive: need to fit the model N times!
- LOOCV gives almost unbiased estimate of the out-of-sample error, since its training set contains almost the entire data set.
- However, the LOOCV estimate has high variance: when we perform LOOCV, we are in effect averaging the outputs of N fitted models, each of which is trained on an almost identical set of observations. Therefore, these outputs are highly positively correlated with each other.
- There exists a *bias-variance tradeoff*: when K is small, bias is high and variance is low. When K is large, bias is low and variance is high.
- Rule of thumb: $K = 5$ or $K = 10$.

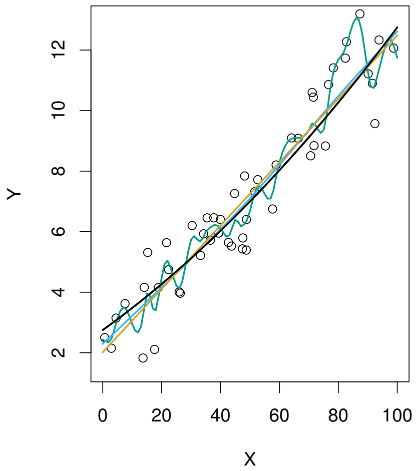
Cross-Validation for Regression

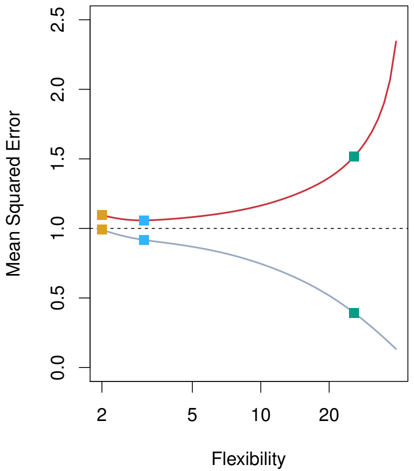




Left: target function (black), linear fit (orange), smoothing spline fits (blue & green). Right: training error (grey), prediction error (red), Var(e) (dashed).

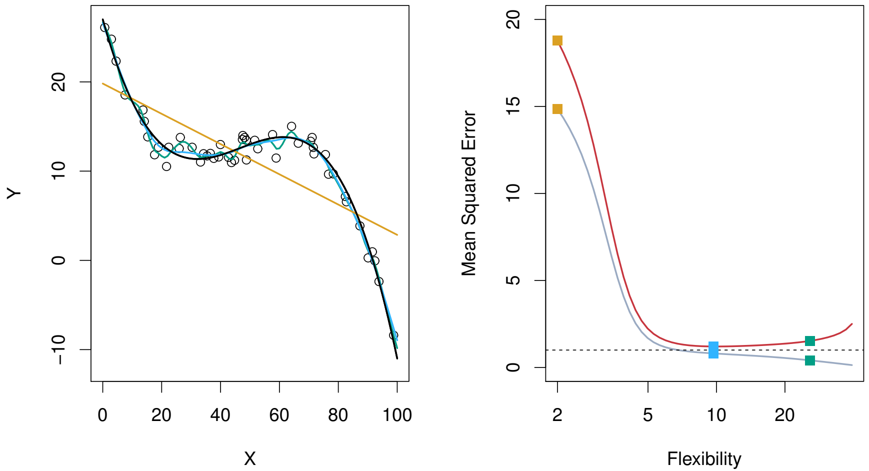
Cross-Validation for Regression





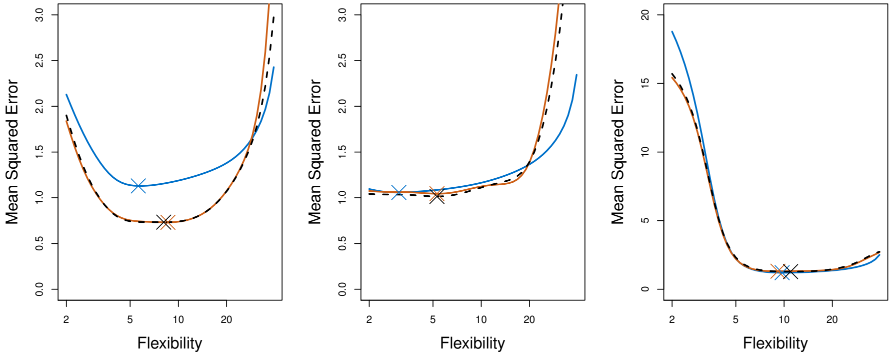
Left: target function (black), linear fit (orange), smoothing spline fits (blue & green). Right: training error (grey), prediction error (red), Var(e) (dashed).

Cross-Validation for Regression



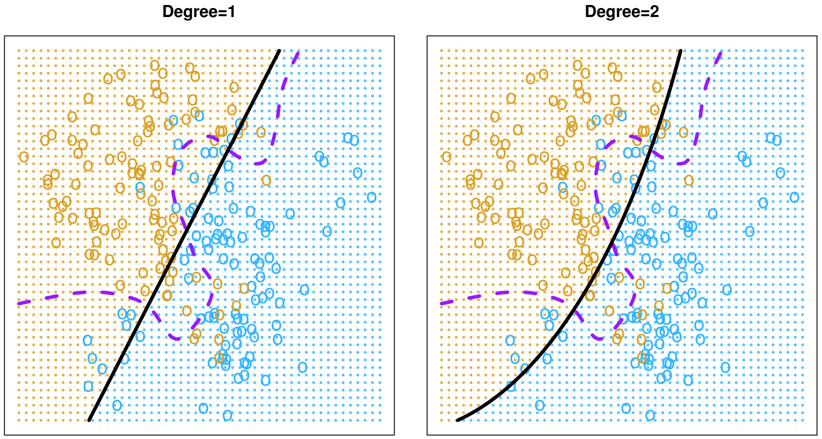
Left: target function (black), linear fit (orange), smoothing spline fits (blue & green). Right: training error (grey), prediction error (red), Var(e) (dashed).

Cross-Validation for Regression



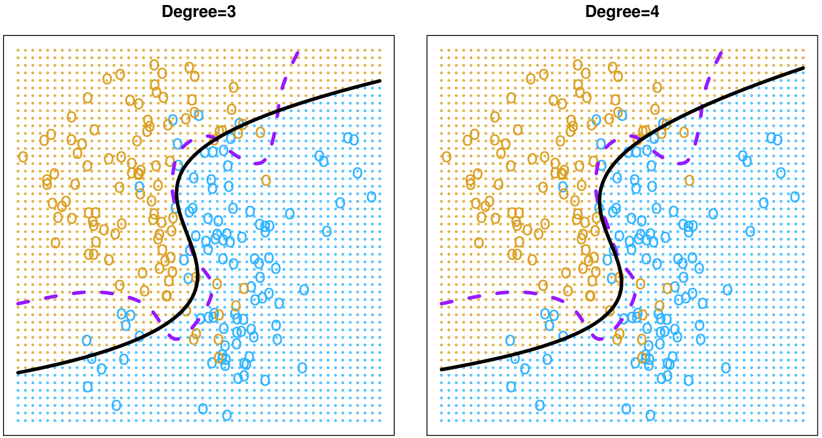
True and estimated prediction error for the simulated data sets. True prediction error (blue), LOOCV estimate (black dashed), 10-fold CV estimate (orange). Crosses indicate the minimum of each of the error curves.

Cross-Validation for Classification



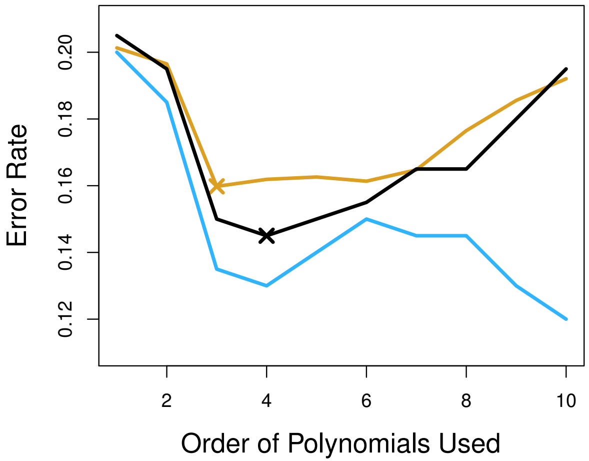
Two-dimensional classification data. Purple dashed: the Bayes decision boundary; Black: estimated decision boundaries from linear (left) and quadratic (right) logistic regressions.

Cross-Validation for Classification



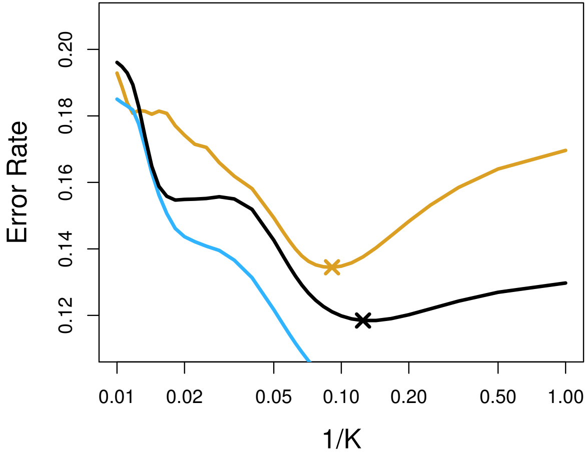
Two-dimensional classification data. Purple dashed: the Bayes decision boundary; Black: estimated decision boundaries from cubic (left) and quartic (right) logistic regressions.

Cross-Validation for Classification



Logistic regression using polynomial functions of the predictors. True misclassification rate (brown), training error (blue), 10-fold CV error (black).

Cross-Validation for Classification



KNN classifier. True misclassification rate (brown), training error (blue), 10-fold CV error (black).

Note on Cross-Validation

Note

In choosing the K folds at random, we are assuming that (x_i, y_i) are independently drawn from the underlying population $p(x, y)$, i.e., our data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ constitutes a *random sample*.

This would not be true in *structured problems* where (x_i, y_i) are not independent – for example, when $\mathcal{D}$ is time-series data[a].

[a] See here for discussions on cross-validation for time series data.

Alternatives to CV: Information Criteria

Suppose for dependent variable y, we have two potential predictors, x_1 and x_2. If we limit our attention to linear models without interactions, we could fit the following four models:

$$\begin{aligned}
\text{Model A: } & y_i = \beta_0 + e_i \\
\text{Model B: } & y_i = \beta_0 + \beta_1 x_{i1} + e_i \\
\text{Model C: } & y_i = \beta_0 + \beta_2 x_{i2} + e_i \\
\text{Model D: } & y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i
\end{aligned}$$

- Model A is the **null model**, which contains no predictors.
- Model D is the **full model**, which contains all potential predictors.

Alternatives to CV: Information Criteria

We could use CV to select the best model. Alternatively, we can use **information criteria** for model selection. These are metrics that make adjustment to the training error in order to account for the bias due to overfitting.

Akaike's Information Criterion (**AIC**)

$$\text{AIC} = -2\log\mathcal{L} + 2p = \text{Deviance} + 2p$$

, where $\mathcal{L}$ is the value of the likelihood function[3], and p is the number of parameters.

Bayesian Information Criterion (**BIC**)

$$\text{BIC} = -2\log\mathcal{L} + p\log N = \text{Deviance} + p\log N$$

, where N is the number of data points.

[3] achieved at the maximum-likelihood estimate.

Alternatives to CV: Information Criteria

- For both AIC and BIC, smaller values are better. We can use them for model selection by choosing the model with the minimum AIC or BIC.
- The term $2p$ in AIC and $p \log N$ in BIC can be regarded as **penalty** terms that favor simpler models over more complicated ones.
- Both AIC and BIC have theoretical justifications that rely on asymptotic arguments.
- Since $\log N > 2$ for $N > 7$, BIC generally increases with p faster than AIC. Thus BIC tends to lead to the selection of smaller models than AIC.

MPG and Horsepower

```r
require(foreach)
foreach (i = 1:5) %do% {
  model <- lm(mpg ~ poly(horsepower,i))
  c(AIC(model), BIC(model))
}
```

```
## [[1]]
## [1] 2363.324 2375.237
##
## [[2]]
## [1] 2274.354 2290.239
##
## [[3]]
## [1] 2275.531 2295.388
##
## [[4]]
## [1] 2276.108 2299.936
##
## [[5]]
## [1] 2268.663 2296.462
```

Subset Selection

- In general, given p potential predictors in a linear model, we can use CV or information criteria to choose the best subset of predictors. This is also called **variable selection**.
- However, given p potential predictors, there are 2^p possible models – each involves a subset of the p predictors. Comparing all of them becomes computationally infeasible when p is large.
- **Forward stepwise selection** is a computationally efficient alternative that begins with the null model, and then adds predictors one-at-a-time.

Subset Selection

Forward Stepwise Selection

1. Let $\mathcal{M}_0$ denote the null model.
2. Fit all univariate models. Choose the one with the best in-sample fit (smallest RSS, highest R^2) and add that variable – say $x_{(1)}$– to $\mathcal{M}_0$. Call the resulting model $\mathcal{M}_1$.
3. Fit all bivariate models that include $x_{(1)}$: $y \sim \beta_0 + \beta_{(1)}x_{(1)} + \beta_j x_j$, and add x_j from the one with the best in-sample fit to $\mathcal{M}_1$. Call the resulting model $\mathcal{M}_2$.
4. Continue until your model selection rule (cross-validation error, AIC, BIC) is lower for the current model than for any of the models that add one variable.

Subset Selection

- Similarly, we can also do **backward stepwise selection**: start with the full model and remove predictors one-at-a-time.
- Forward is in general better than backward:
  - The full model can be expensive to fit, while the null model is usually available in closed form.
  - The full model fit has high variance (because it is usually overfit). The null model is always the same.
  - Backward selection requires $N > p$ (so that the full model can be fit). Forward selection can be used even when $p > N$.
- Neither forward nor backward selection is guaranteed to find the best subset of predictors.

Credit Card Balance

```
require(AER)
credit <- read.csv("Credit.csv")
fit <- lm(Balance~.,credit)
coeftest(fit)

##
## t test of coefficients:
##
##                 Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)  -484.017314   26.076322 -18.5616 < 2.2e-16 ***
## Income         -7.798884    0.233752 -33.3639 < 2.2e-16 ***
## Limit           0.191288    0.032555   5.8759 9.011e-09 ***
## Rating          1.129091    0.487398   2.3166   0.02104 *
## Cards          17.919886    4.329445   4.1391 4.274e-05 ***
## Age            -0.638650    0.293004  -2.1797   0.02988 *
## GenderFemale  -10.327215    9.896964  -1.0435   0.29737
## StudentYes    425.498590   16.608864  25.6188 < 2.2e-16 ***
## MarriedYes     -7.482805   10.249865  -0.7300   0.46580
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Credit Card Balance

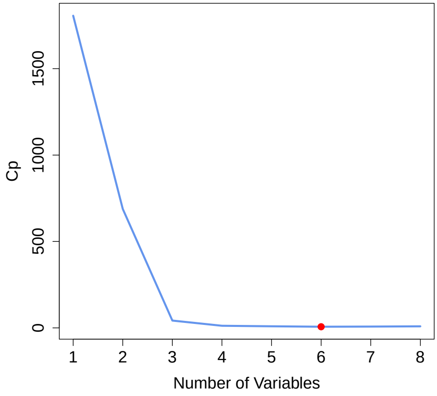
```
require(leaps)
all.fit <- regsubsets(Balance~., credit)
all <- summary(all.fit)
all$outmat
```

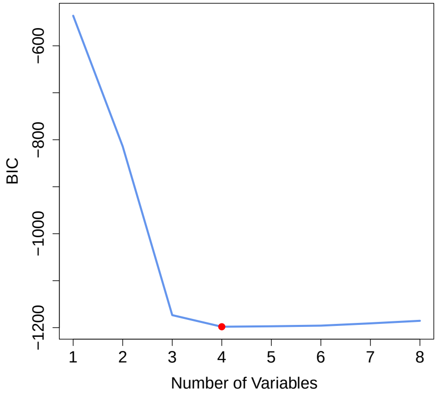
```
##          Income Limit Rating Cards Age GenderFemale StudentYes MarriedYe
## 1  ( 1 ) " "    " "   "*"    " "   " " " "          " "        " "
## 2  ( 1 ) "*"    " "   "*"    " "   " " " "          " "        " "
## 3  ( 1 ) "*"    " "   "*"    " "   " " " "          "*"        " "
## 4  ( 1 ) "*"    "*"   " "    "*"   " " " "          "*"        " "
## 5  ( 1 ) "*"    "*"   "*"    "*"   " " " "          "*"        " "
## 6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "          "*"        " "
## 7  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"          "*"        " "
## 8  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"          "*"        "*"
```

Credit Card Balance

```
# Mallow's Cp statistic is equivalent to AIC in the case of
# linear models with Gaussian errors
all$cp # Mallow's Cp

## [1] 1806.332193  688.156412   42.321818   12.249136    9.218038    6.648
## [7]    7.532958    9.000000

all$bic

## [1]  -535.9468  -814.1798 -1173.3585 -1198.0527 -1197.0957 -1195.7321 -1
## [8] -1185.4324
```

Credit Card Balance





Credit Card Balance

```
# best model according to AIC
coef(all.fit,6)

##  (Intercept)       Income        Limit       Rating        Cards          
## -493.7341870   -7.7950824    0.1936914    1.0911874   18.2118976   -0.62
##   StudentYes 
##  425.6099369 

# best model according to BIC
coef(all.fit,4)

##  (Intercept)       Income        Limit        Cards   StudentYes 
## -499.7272117   -7.8392288    0.2666445   23.1753794  429.6064203
```

Credit Card Balance

```r
## Forward stepwise selection using AIC
null <- lm(Balance~1,credit)
full <- lm(Balance~.,credit)
fwd <- step(null, scope=formula(full), direction="forward")
```

```
## Start:  AIC=4905.56
## Balance ~ 1
##
##             Df Sum of Sq      RSS    AIC
## + Rating     1  62904790 21435122 4359.6
## + Limit      1  62624255 21715657 4364.8
## + Income     1  18131167 66208745 4810.7
## + Student    1   5658372 78681540 4879.8
## + Cards      1    630416 83709496 4904.6
## <none>                   84339912 4905.6
## + Gender     1     38892 84301020 4907.4
## + Married    1      2715 84337197 4907.5
## + Age        1       284 84339628 4907.6
##
```

Credit Card Balance

```
##
## Step:  AIC=4359.63
## Balance ~ Rating
##
##           Df Sum of Sq      RSS    AIC
## + Income   1  10902581 10532541 4077.4
## + Student  1   5735163 15699959 4237.1
## + Age      1    649110 20786012 4349.3
## + Cards    1    138580 21296542 4359.0
## + Married  1    118209 21316913 4359.4
## <none>                 21435122 4359.6
## + Gender   1     16065 21419057 4361.3
## + Limit    1      7960 21427162 4361.5
##
```

Credit Card Balance

```
##
## Step:  AIC=4077.41
## Balance ~ Rating + Income
##
##           Df Sum of Sq      RSS    AIC
## + Student  1   6305322  4227219 3714.2
## + Married  1     95068 10437473 4075.8
## + Limit    1     94545 10437996 4075.8
## + Age      1     90286 10442255 4076.0
## <none>                 10532541 4077.4
## + Cards    1      2094 10530447 4079.3
## + Gender   1       948 10531593 4079.4
##
```

Credit Card Balance

```
##
## Step:  AIC=3714.24
## Balance ~ Rating + Income + Student
##
##           Df Sum of Sq     RSS    AIC
## + Limit    1    194718 4032502 3697.4
## + Age      1     44620 4182600 3712.0
## <none>                 4227219 3714.2
## + Married  1     13083 4214137 3715.0
## + Gender   1     12168 4215051 3715.1
## + Cards    1     10608 4216611 3715.2
##
```

Credit Card Balance

```
##
## Step:  AIC=3697.37
## Balance ~ Rating + Income + Student + Limit
##
##           Df Sum of Sq     RSS    AIC
## + Cards    1    166410 3866091 3682.5
## + Age      1     37952 3994549 3695.6
## <none>                 4032502 3697.4
## + Gender   1     13345 4019157 3698.0
## + Married  1      6660 4025842 3698.7
##
```

Credit Card Balance

```
##
## Step:  AIC=3682.52
## Balance ~ Rating + Income + Student + Limit + Cards
##
##           Df Sum of Sq     RSS    AIC
## + Age      1     44472 3821620 3679.9
## <none>                 3866091 3682.5
## + Gender   1     11350 3854741 3683.3
## + Married  1      3121 3862970 3684.2
##
```

Credit Card Balance

```
##
## Step:  AIC=3679.89
## Balance ~ Rating + Income + Student + Limit + Cards + Age
##
##            Df Sum of Sq     RSS    AIC
## <none>                  3821620 3679.9
## + Gender    1   10860.9 3810759 3680.7
## + Married   1    5450.6 3816169 3681.3
##
```

Credit Card Balance

```
coeftest(fwd)

##
## t test of coefficients:
##
##                Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept) -493.734187   24.824765 -19.8888 < 2.2e-16 ***
## Rating         1.091187    0.484804   2.2508   0.02495 *
## Income        -7.795082    0.233417 -33.3955 < 2.2e-16 ***
## StudentYes   425.609937   16.509565  25.7796 < 2.2e-16 ***
## Limit          0.193691    0.032383   5.9813 4.980e-09 ***
## Cards         18.211898    4.318650   4.2170 3.075e-05 ***
## Age           -0.624056    0.291817  -2.1385   0.03309 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Credit Card Balance

```r
## Forward stepwise selection using BIC
n <- nrow(credit)
fwd <- step(null, scope=formula(full), direction="forward", k=log(n))
```

```r
coeftest(fwd)
```

```
## 
## t test of coefficients:
## 
##                Estimate  Std. Error  t value  Pr(>|t|)    
## (Intercept) -526.155523   19.746614 -26.6454 < 2.2e-16 ***
## Rating         1.087901    0.486995   2.2339   0.02605 *  
## Income        -7.874924    0.231455 -34.0236 < 2.2e-16 ***
## StudentYes   426.850146   16.574025  25.7542 < 2.2e-16 ***
## Limit          0.194409    0.032527   5.9768 5.099e-09 ***
## Cards         17.851731    4.334889   4.1182 4.656e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Bias-Variance Trade-off

The goal of model selection is to choose the model (out of the set of candidate models) that can best balance the bias-variance tradeoff to achieve the smallest out-of-sample error.

In the case of linear models, we have:

$$E\left[\left(y - x'\widehat{\beta}\right)^2\right] = Var\left(x'\widehat{\beta}\right) + \left[\text{bias}\left(x'\widehat{\beta}\right)\right]^2 + Var\left(e\right)$$

- The OLS estimate $\widehat{\beta}^{LS}$ is unbiased: $E\left[\widehat{\beta}^{LS}\right] = \beta^*$. Hence $\text{bias}\left(x'\widehat{\beta}^{LS}\right) = E\left[x'\beta^* - x'\widehat{\beta}^{LS}\right] = 0$. What about $\text{Var}\Big(x'\widehat{\beta}^{LS}\Big)$?

The Gauss-Markov Theorem

Note that $\widehat{\beta}^{LS}$ is a linear function of Y: $\widehat{\beta}^{LS} = \left(X'X\right)^{-1} X'Y = \sum_i w_i y_i$, where $w_i = \left(X'X\right)^{-1} x_i$.

Gauss-Markov Theorem

Under the assumption that

1. The true CEF is linear: $E\left[y|x\right] = x'\beta^*$
2. Homoskedasticity: $E\left[\left.(e^*)^2\right| x\right] = \sigma^2$

The OLS estimator $\widehat{\beta}^{LS}$ is **BLUE**: **B**est **L**inear **U**nbiased **E**stimator, in the sense that among all *unbiased* linear estimators (estimators that are linear functions of Y), $\widehat{\beta}^{LS}$ has the *smallest* variance.

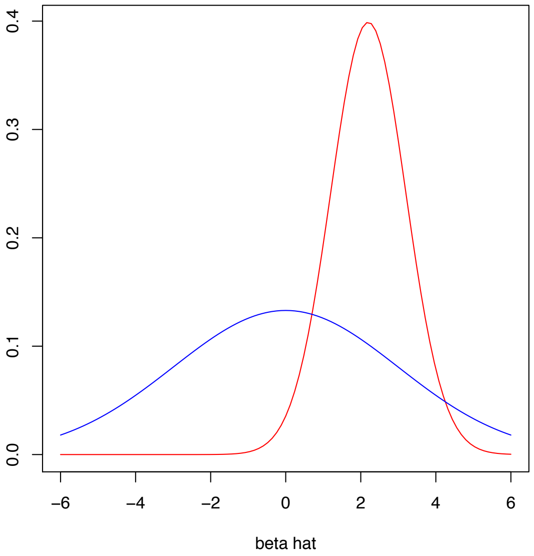
The Gauss-Markov Theorem

Under the assumptions of Gauss-Markov,

$$E\left[\left(y - x'\widehat{\beta}^{LS}\right)^2\right] = Var\left(x'\widehat{\beta}^{LS}\right) + \sigma^2 \tag{3}$$

- The theorem says that if we have any other *unbiased* linear estimator $\widehat{\beta}$, then $Var\left(x'\widehat{\beta}^{LS}\right) \leq Var\left(x'\widehat{\beta}\right)$.
- But can we do better – reduce the prediction error – if we are willing to allow a little bias in exchange for a larger reduction in variance?

Exploring the Bias-Variance Trade-off



Exploring the Bias-Variance Trade-off

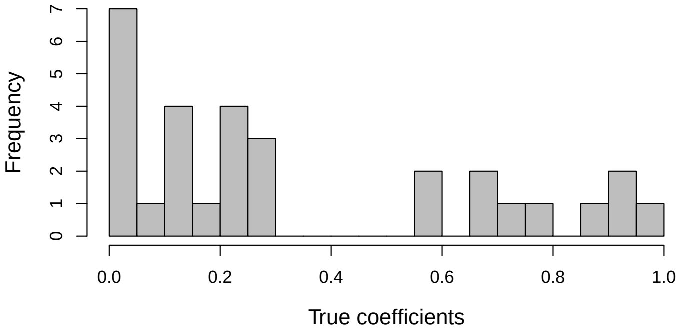
(3) $\Rightarrow$

$$
\begin{aligned}
E_{\mathcal{D},x}\left[\left(y - x'\widehat{\beta}^{LS}\right)^2\right] &\approx \frac{1}{N}\sum_{i=1}^{N} E_{\mathcal{D}}\left[\left(y_i - x_i'\widehat{\beta}^{LS}\right)^2\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} Var\left(x_i'\widehat{\beta}^{LS}\right) + \sigma^2 \\
&= \frac{1}{N}\text{trace}\left(Var\left(X\widehat{\beta}^{LS}\right)\right) + \sigma^2 \\
&\approx \frac{1}{N}\text{trace}\left(X\left(X'X\right)^{-1}X'\right)\sigma^2 + \sigma^2 \\
&= \frac{1}{N}\left(1+p\right)\sigma^2 + \sigma^2
\end{aligned}
$$

, i.e., the prediction error scales *linearly* with the number of predictors p.

Simulation 1

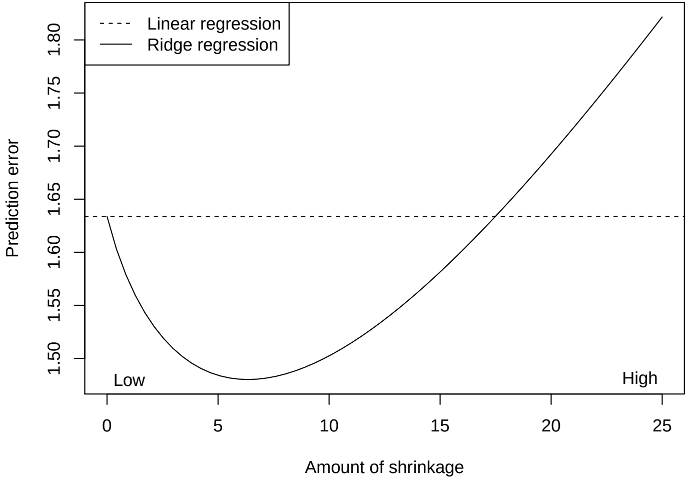
$N = 50$, $p = 30$. $y \sim \mathcal{N}(x'\beta^*, 1)$, where $\beta^* \in \mathbb{R}^{30}$ with 10 “large” coefficients (between 0.5 and 1) and 20 “small” ones (between 0 and 0.3).



Exploring the Bias-Variance Trade-off

- Each additional predictor adds the same amount of variance $\frac{\sigma^2}{N}$, regardless of whether its true coefficient is large or small (or zero).
- So in this example, we are "spending" variance in trying to fit truly small coefficients—20 of them, out of 30 total.
- We can do better by shrinking small coefficients towards zero, incurring some bias, so as to reduce the variance. Methods that do this are called **shrinkage methods**.

Simulation 1



Simulation 1

Linear regression:

$$\begin{aligned} \text{Square bias} &\approx 0.006 \\ \text{Variance} &\approx 0.627 \\ \text{prediction error} &\approx 1 + 0.006 + 0.627 = 1.633 \end{aligned}$$

Ridge regression, at its best:

$$\begin{aligned} \text{Square bias} &\approx 0.077 \\ \text{Variance} &\approx 0.403 \\ \text{prediction error} &\approx 1 + 0.077 + 0.403 = 1.48 \end{aligned}$$

Ridge Regression

$$\widehat{\beta}^R = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\text{penalty}} \right\} \tag{4}$$

, where $\lambda \geq 0$ is a **tuning parameter** that controls the strength of the penalty term, which has the effect of shrinking the estimates towards zero.

- $\lambda = 0 \Rightarrow \widehat{\beta}^R = \widehat{\beta}^{LS}$[4]
- $\lambda = \infty \Rightarrow \widehat{\beta}_1^R = \cdots = \widehat{\beta}_p^R = 0$[5]

[4] $\widehat{\beta}^{LS}$ denotes the least square estimate.
[5] $\widehat{\beta}_0$ is not shrunk toward 0.

Ridge Regression

An equivalent way to write the ridge problem is:

$$\widehat{\beta}^R = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \tag{5}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq C$$

(4) is the Lagrangian form of (5). There is a one-to-one correspondence between λ in (4) and C in (5).

Ridge Regression

- Least square estimates are scale equivariant: multiplying x_j by a constant c leads to a scaling of $\widehat{\beta}_j$ by $1/c$. $\widehat{\beta}_j x_j$ remains the same.

- Ridge regression estimates are *not* scale equivariant. They can change substantially when multiplying a given predictor by a constant. Therefore, x_{ij} need to be *standardized* before applying ridge regression.

- In addition, note that β_0 is not penalized: we do *not* shrink $\widehat{\beta}_0$. Otherwise we could add some constant c to y and the solution would be different.

Ridge Regression

Let $\widetilde{x}_{ij} = \frac{x_{ij} - \overline{x}_j}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x}_j)^2}}$, so that x_{ij}s are both *centered* and *standardized*.

If we use $\widetilde{x}_{ij}$ as the regressors, then $\widehat{\beta}_0 = \overline{y}$[6] and we can estimate the the rest of the βs by a ridge regression without intercept:

$$\widehat{\beta}^R = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(\widetilde{y}_i - \widetilde{x}_i'\beta\right)^2 + \lambda \|\beta\|_2^2 \right\} \tag{6}$$

, where $\|.\|_2$ denotes the ℓ_2 norm, $\widetilde{y}_i = y_i - \overline{y}$, $\widetilde{x}_i = [\widetilde{x}_{i1}, \ldots, \widetilde{x}_{ip}]'$, and $\beta = [\beta_1, \ldots, \beta_p]'$.

[6] This is true as long as $\widetilde{x}_{ij}$ are centered.

Ridge Regression

(6) $\Rightarrow$

$$\widehat{\beta}^R = \left(\widetilde{X}'\widetilde{X} + \lambda I\right)^{-1}\widetilde{X}'Y \tag{7}$$

, where I is the $p \times p$ identity matrix, and $\widetilde{X}$ is centered and standardized.

In the case that $\widetilde{X}$ is *orthonormal*[7], (7) $\Rightarrow$

$$\widehat{\beta}_j^R = \frac{\widehat{\beta}_j^{LS}}{1+\lambda} \tag{8}$$

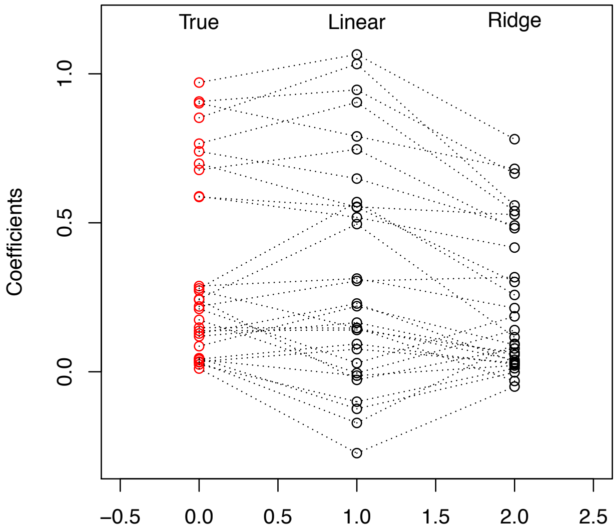
[7]so that $\widetilde{X}'\widetilde{X} = I$

Ridge Regression

To choose λ, we can treat λ is as a hyperparameter and select its value via cross-validation:

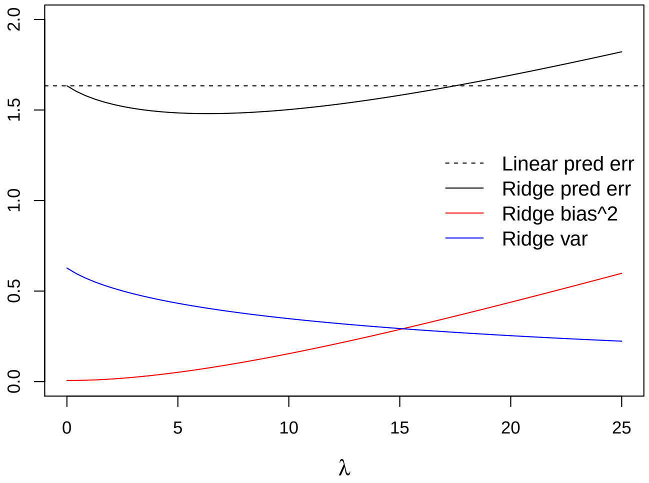
- Given a grid of λ values, perform cross-validation on the training set to pick the λ for which the cross-validation error is smallest.
- Re-fit the model with the selected λ using all the training data.
- Benchmark the performance of the final fitted model using a test set.

Simulation 1



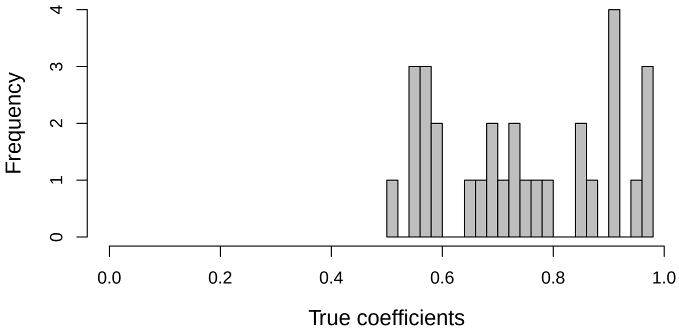
Ridge regression coefficients at $\lambda = 25$

Simulation 1

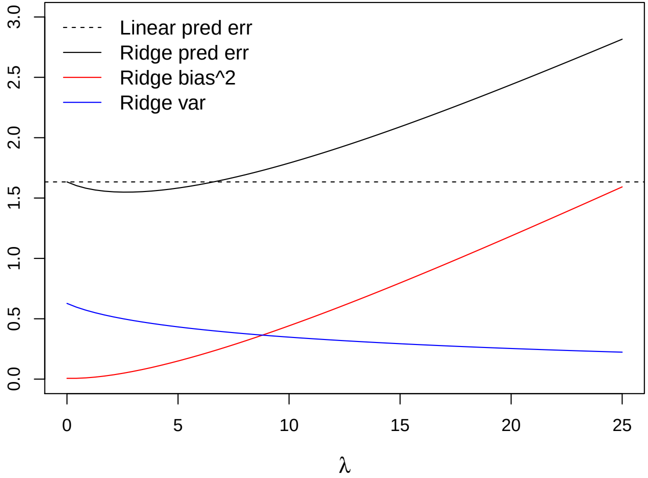


Simulation 2

$N = 50, p = 30$. $y \sim \mathcal{N}(x'\beta^*, 1)$, where $\beta^* \in \mathbb{R}^{30}$ with all 30 coefficients between 0.5 and 1.



Simulation 2



Credit Card Balance

```
require(glmnet)
x <- model.matrix(Balance~.,credit)[,-1] # no intercept
y <- credit$Balance
ridgefit <- glmnet(x,y,alpha=0,lambda=0.01)
coef(ridgefit)

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  -479.7851081
## Income         -7.8002927
## Limit           0.2049782
## Rating          0.9251803
## Cards          18.9116337
## Age            -0.6364639
## GenderFemale  -10.3247237
## StudentYes    426.0874930
## MarriedYes     -7.0597648
```

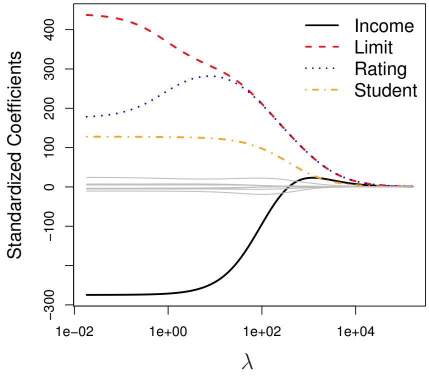
Credit Card Balance

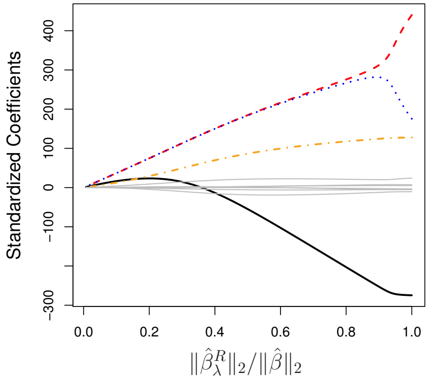
```
ridgefit <- glmnet(x,y,alpha=0,lambda=1e10)
coef(ridgefit)

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)   5.200149e+02
## Income        2.777310e-07
## Limit         7.881306e-09
## Rating        1.178376e-07
## Cards         1.331034e-06
## Age           2.245918e-09
## GenderFemale  9.061130e-07
## StudentYes    1.820460e-05
## MarriedYes   -2.455471e-07
```

Credit Card Balance

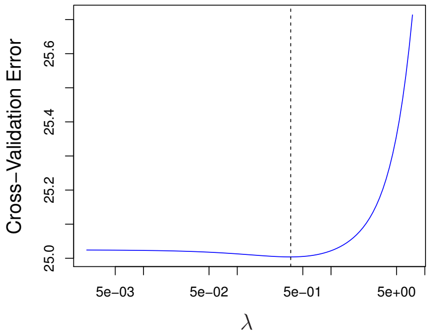
```
ridgefit <- glmnet(x,y,alpha=0) # automatically select a range of lambda
dim(coef(ridgefit))

## [1]   9 100

c(ridgefit$lambda[1],ridgefit$lambda[50],ridgefit$lambda[100])

## [1] 396562.69957   4154.45333     39.65627

coef(ridgefit)[,50]

##  (Intercept)       Income        Limit       Rating        Cards
## 339.02562071   0.44648135   0.01505380   0.22506908   2.77187416   -0.051
## GenderFemale   StudentYes   MarriedYes
##   1.76037503  39.44486798  -0.95049050
```

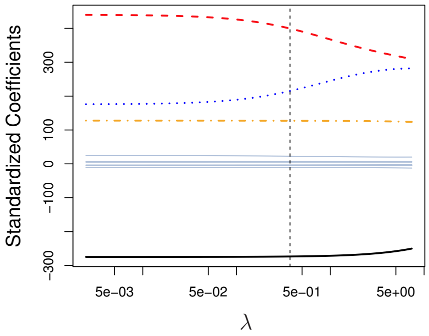
Credit Card Balance





Credit Card Balance





The Lasso

The lasso[8] is defined as:

$$\widehat{\beta}^L = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{9}$$

Equivalently,

$$\widehat{\beta}^L = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \tag{10}$$

$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq C$$

[8]acronym for: Least Absolute Selection and Shrinkage Operator. Not to be confused with wonder woman's weapon of choice.

The Lasso

Centering and standardizing $x_{ij} \Rightarrow$

$$\widehat{\beta}^L = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(\widetilde{y}_i - \widetilde{x}_i' \beta \right)^2 + \lambda \left\| \beta \right\|_1 \right\} \tag{11}$$

, where $\|.\|_1$ denotes the ℓ_1 norm, $\widetilde{y}_i = y_i - \overline{y}$, $\widetilde{x}_{ij} = \frac{x_{ij} - \overline{x}_j}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x}_j)^2}}$, and $\beta = [\beta_1, \ldots, \beta_p]'$.

The Lasso

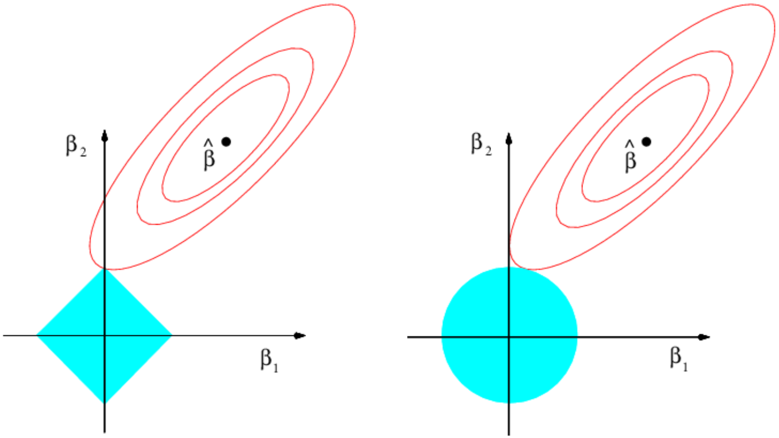
There is in general no closed-form solution to (11). In the case that X is *orthonormal*, we have:

$$\widehat{\beta}^{L} = \begin{cases} \widehat{\beta}^{LS} - \frac{\lambda}{2} & \text{if } \widehat{\beta}^{LS} > \frac{\lambda}{2} \\ \widehat{\beta}^{LS} + \frac{\lambda}{2} & \text{if } \widehat{\beta}^{LS} < -\frac{\lambda}{2} \\ 0 & \text{if } \left|\widehat{\beta}^{LS}\right| \le \frac{\lambda}{2} \end{cases}$$

The Lasso

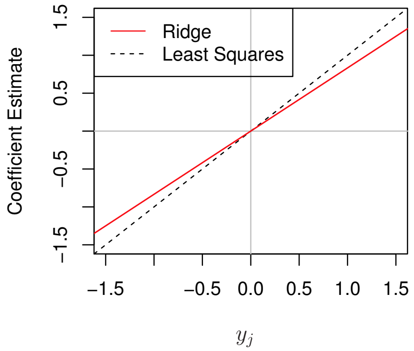
- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the ℓ_2 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter is sufficiently large.
- Hence, the lasso performs variable selection. We say that the lasso yields **sparse models** — that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical and is in practice done via cross-validation.

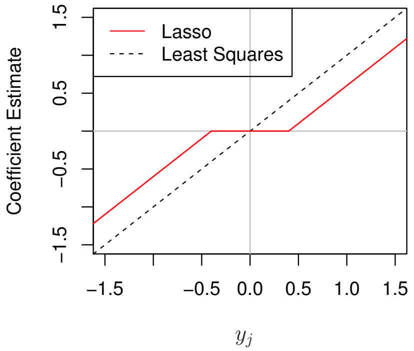
The Lasso



Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq C$ and $\beta_1^2 + \beta_2^2 \leq C$, while the red ellipses are the contours of the RSS.

The Lasso





The ridge regression and the lasso coefficient estimates for a simple setting with $N = p$ and X being orthonormal.

Credit Card Balance

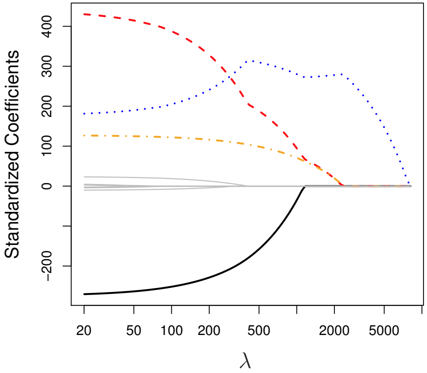
```
lassofit <- glmnet(x,y,alpha=1,lambda=0.01)
coef(lassofit)

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  -479.5814236
## Income         -7.7997555
## Limit           0.2056525
## Rating          0.9149775
## Cards          18.9546633
## Age            -0.6358133
## GenderFemale  -10.3038053
## StudentYes    426.0922168
## MarriedYes     -7.0184389
```

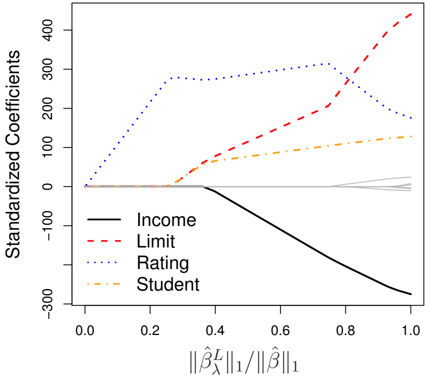
Credit Card Balance

```
lassofit <- glmnet(x,y,alpha=1,lambda=100)
coef(lassofit)

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  -157.92073077
## Income                .
## Limit            0.01822877
## Rating           1.64828689
## Cards                 .
## Age                   .
## GenderFemale          .
## StudentYes      65.68634041
## MarriedYes            .
```

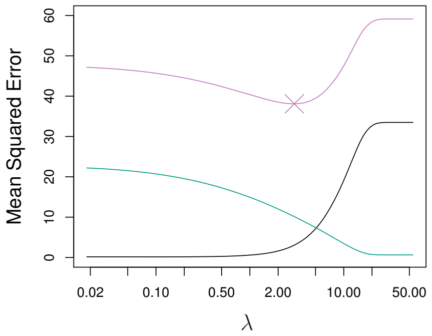
Credit Card Balance

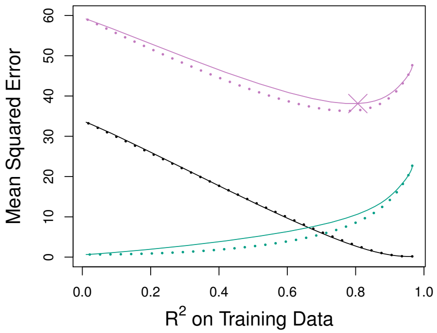
```
lassofit <- glmnet(x,y,alpha=1) # automatically select a range of lambda
```





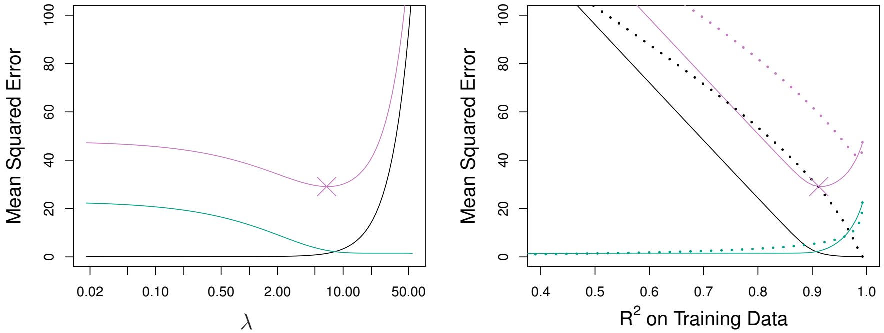
Comparing the Lasso and Ridge Regression





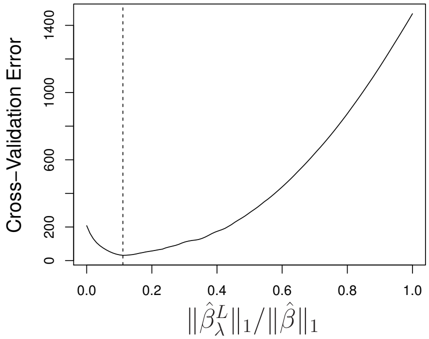
Plots of squared bias (black), variance (green), and prediction error (purple) for the lasso (solid) and ridge (dashed) on a simulated data set with $p = 45$ and $n = 50$. All 45 predictors are related to the response.

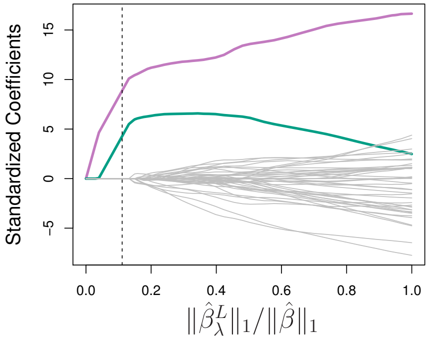
Comparing the Lasso and Ridge Regression



Comparison of squared bias, variance and prediction error between the lasso (solid) and ridge (dashed). Now only two predictors in the simulated data set are related to the response.

Comparing the Lasso and Ridge Regression





Ten-fold cross-validation error for the lasso, applied to the simulated data set with 2 out of 45 predictors related to the response.

U.S. Election

- We want to predict whether a person voted for Clinton or Trump in the 2016 U.S. presidential election using demographic variables taken from the 2018 General Social Survey (GSS).
- Selected variables include: age, sex, income, work, marriage, family information, political party affiliation, etc.

U.S. Election

```
gss <- read.csv("GSS2018.csv")
dim(gss)

## [1] 455  50

names(gss)

##  [1] "age"      "sex"      "sibs"     "size"     "adults"   "childs"
##  [7] "class"    "coninc"   "actssoc"  "attend"   "born"     "cappun"
## [13] "courts"   "degree"   "earnrs"   "ethnum"   "famgen"   "finalter"
## [19] "finrela"  "fund"     "hhrace"   "madeg"    "marital"  "mobile16"
## [25] "natmass"  "natpark"  "othlang"  "natchld"  "sexornt"  "partyid"
## [31] "phone"    "polviews" "pres16"   "prestg10" "natsci"   "quallife"
## [37] "race"     "raclive"  "rank"     "relig"    "reliten"  "satfin"
## [43] "satsoc"   "sei10"    "vetyears" "vote12"   "happy"    "weekswrk"
## [49] "wrkslf"   "wrkstat"
```

U.S. Election

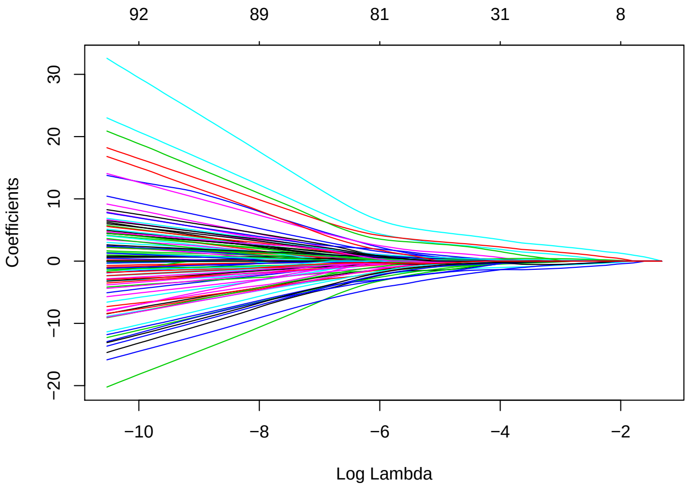
```
Y <- gss$pres16 # GSS Q: Did you vote for Clinton or Trump?
summary(Y)

## Clinton   Trump
##     260     195

X <- model.matrix(pres16 ~.,gss)[,-1] # create dummies for categorical vars
dim(X)

## [1] 455 128
```

U.S. Election

```r
##############################
# Logistic Lasso Regression #
##############################
require(glmnet)
# fit the lasso with a range of automatically selected lambdas
lasso.all <- glmnet(X,Y,alpha=1,family="binomial")
cv.lasso <- cv.glmnet(X,Y,alpha=1,family="binomial") # cross-validation
# choose lambda associated with min CV error (or plus 1se)
lambda.star <- cv.lasso$lambda.1se # Alternatively: cv.lasso$lambda.min
# fit the lasso with optimal lambda
fit = glmnet(X,Y,alpha=1,lambda=lambda.star,family="binomial")
```

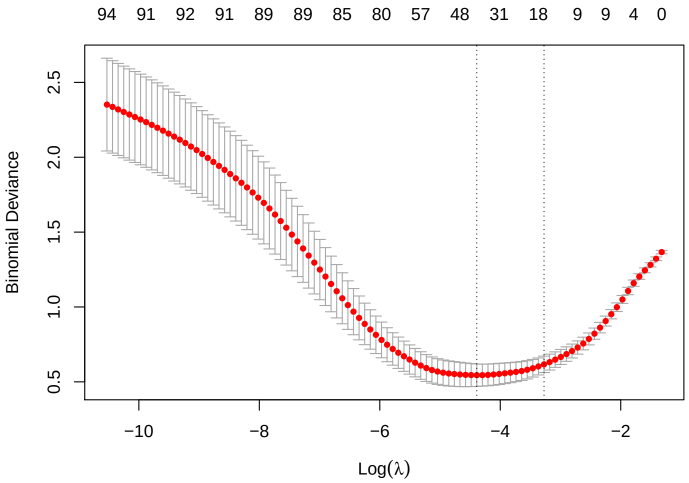
U.S. Election

```
plot(lasso.all,xvar="lambda")
```



U.S. Election

```
plot(cv.lasso)
```



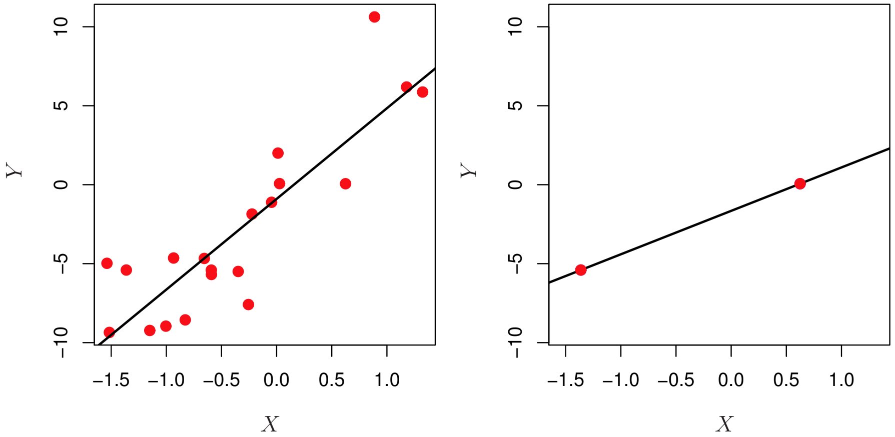
U.S. Election

```
source('lassosummary.R') # external function
lassosummary(fit)

##                                   beta
## sexmale                     0.01785683
## cappunoppose               -0.77766891
## hhraceblack                -0.73658259
## hhracewhite                 0.20921740
## maritalmarried              0.06177141
## maritalnever married       -0.21100431
## natchldtoo little          -0.11405573
## partyidind,near rep         1.22815699
## partyidnot str democrat    -0.54035075
## partyidnot str republican  1.63637730
## partyidother party          0.49325142
## partyidstrong democrat     -1.26057401
## partyidstrong republican   2.60413814
## polviewsliberal            -0.03666117
## polviewsslightly liberal   -0.04752788
## racewhite                   0.41245491
## religprotestant             0.05175686
## satfinsatisfied             0.02233638
```

Considerations in High Dimensions

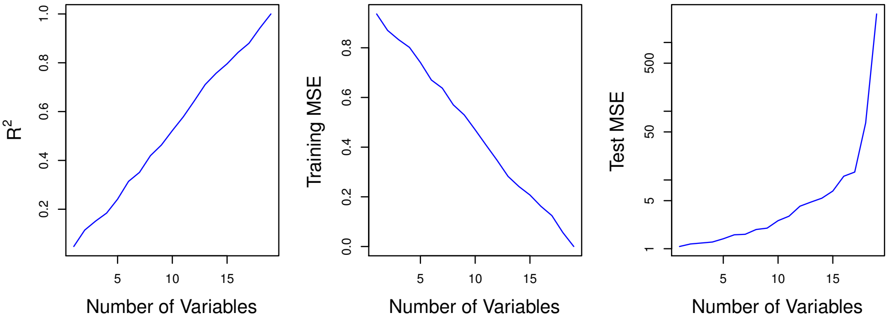
- Most traditional statistical techniques for regression and classification are intended for the low-dimensional settings in which $N \gg p$.
- Settings in which p is large relative to N are often referred to as high-dimensional. When $p > N$, classical approaches such as least squares linear regression can no longer be applied.

Considerations in High Dimensions



Left: Least squares regression in the low-dimensional setting. Right: Least squares regression with $N = 2$ observations and two parameters to be estimated.

Considerations in High Dimensions



R^2, training MSE, and test MSE on a sample with $N = 20$ observations, as more (unrelated) features are added to the model.

Considerations in High Dimensions

- In high dimensions, even linear models are *too flexible*.
- When $p > N$, information criteria such as AIC and BIC are *not* appropriate.
- Multicollinearity can be extreme in high-dimensional problems.
  - This means that we can never know exactly which variables (if any) truly are predictive of the outcome, and we can never identify the *best* coefficients for use in the regression. At most, we can hope to assign large regression coefficients to variables that are correlated with the variables that truly are predictive of the outcome.

Multi-Stage Lasso

Consider the following linear model:

$$y = x'\beta + e \tag{12}$$

Let β^* be the population least squares coefficients:

$$\beta^* = \arg\min_{\beta} E\left[\left(y - \beta' x\right)^2\right]$$

When β^* is high-dimensional, but contains mostly zeros, i.e. when the number of non-zero population coefficients $\ll p$, we say there is **sparsity**.

Multi-Stage Lasso

In high-dimensional sparse settings, we *may* be able to improve upon the results of the lasso by running a multi-stage procedure:

Multi-Stage Lasso (incl. Post-Lasso OLS)

Stage 1 In the first stage, estimate (12) by the lasso.

Stage 2 Run the lasso again on the variables selected by the first stage.

Alternatively, run OLS on the selected variables in stage 2. Note that this corresponds to running the lasso in the second stage with $\lambda = 0$.

Multi-Stage Lasso

- Intuitively, the first stage focuses on variable selection, while the second stage focuses on (shrinkage) estimation[9].
- Since the variables in the second stage have less "competition" from noise variables, applying the lasso or OLS to these selected variables *could* result in better estimates.

[9] The idea originates from Efron et al. (2004), in which the authors propose running least angle regression (LARS, which incorporates the lasso) in the first stage and OLS in the second stage, a procedure that they call the **LARS-OLS hybrid**. Meinshausen (2007) proposes the two-stage procedure outlined above, which the author calls **relaxed Lasso** (Meinshausen discusses using OLS in the second stage as a special case of the relaxed lasso). Belloni and Chernozhukov (2013) propose the same two-stage procedure as Meinshausen (2007) but with OLS (rather than a lasso that incorporates the OLS) as the second stage, which they call the **post-Lasso OLS**.

Simulation 3

$$\begin{aligned} x_1, \ldots, x_{99}, e &\sim^{i.i.d.} \mathcal{N}(0,1) \\ y &= \beta_1 x_1 + \cdots + \beta_{99} x_{99} + e \end{aligned}$$

, where

$$\begin{aligned} \beta_1 = \cdots = \beta_5 = 5 \\ \beta_6 = \cdots = \beta_{99} = 0 \end{aligned}$$

Simulation 3

```r
# Training Set
n = 100 # sample size
p = 99 # number of variables
s = 5 # number of non-zero coefficients
X = matrix(rnorm(n*p),ncol=p)
beta = c(rep(5,s),rep(0,p-s))
Y = X%*%beta + rnorm(n)

# Test set
n = 1e5
Xnew = matrix(rnorm(n*p),ncol=p)
Ynew = Xnew%*%beta + rnorm(n)
```

Simulation 3

```
#######
# OLS #
#######
require(AER)
fit.ols = lm(Y ~ X-1)
coeftest(fit.ols)[1:10,]

##        Estimate Std. Error    t value   Pr(>|t|)
## X1    5.5742095  0.6359381  8.7653334 0.07231661
## X2    5.1908292  1.4299454  3.6300891 0.17112853
## X3    4.3040108  0.7556606  5.6956934 0.11064444
## X4    5.8280063  0.6142997  9.4872367 0.06685590
## X5    4.4215566  0.4772229  9.2651804 0.06844604
## X6    0.3826747  1.1566349  0.3308518 0.79659012
## X7    0.6595970  1.3420518  0.4914840 0.70918467
## X8   -0.3680858  1.2459579 -0.2954239 0.81712886
## X9   -0.2908295  1.3123265 -0.2216137 0.86116025
## X10 -1.6538125  2.5926638 -0.6378816 0.63852163
```

Simulation 3

```
# number of non-zero coefficients
sum(fit.ols$coefficients!=0)

## [1] 99

# number of statistically significant coefficients:
pv.ols = summary(fit.ols)$coefficients[,4]
sum(pv.ols<.05)

## [1] 0

# test err
pred.ols = predict(fit.ols,data.frame(X=Xnew))
mean((Ynew-pred.ols)^2)

## [1] 229.3538
```

Simulation 3

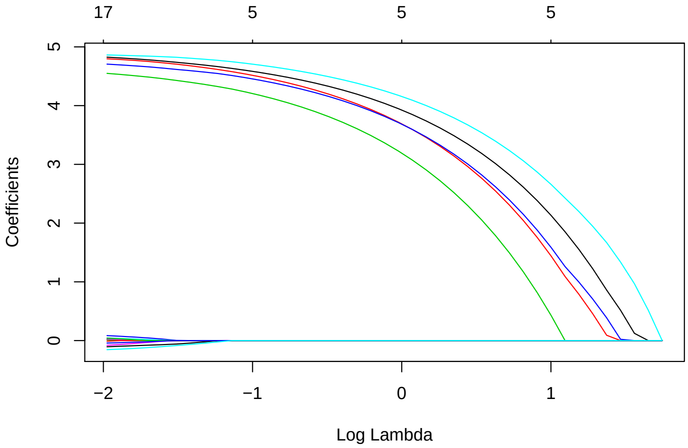
```
#########
# Lasso #
#########
require(glmnet)
cv.lasso = cv.glmnet(X,Y,intercept=FALSE,alpha=1)
lambda.star = cv.lasso$lambda.min
fit.lasso = glmnet(X,Y,intercept=FALSE,alpha=1,lambda=lambda.star)

# number of non-zero estimated coefficients:
sum(fit.lasso$beta!=0)

## [1] 17

# test err
pred.lasso = predict(fit.lasso,Xnew)
mean((Ynew-pred.lasso)^2)

## [1] 1.436501
```

Simulation 3

```
source('lassosummary.R')
lassosummary(fit.lasso)

##            beta
## V1   4.82447042
## V2   4.79685190
## V3   4.54908246
## V4   4.70615289
## V5   4.86261024
## V18 -0.08003794
## V23  0.01706472
## V37  0.01611246
## V39  0.01705680
## V40  0.08449569
## V45  0.04380224
## V77  0.02807225
## V90 -0.10516791
## V94 -0.01021298
## V96  0.04094897
## V97 -0.04232704
## V99 -0.15482818
```

Simulation 3

```r
lasso.all = glmnet(X,Y,intercept=FALSE,alpha=1)
plot(lasso.all,xvar="lambda")
```



Simulation 3

```
#################
# Relaxed Lasso #
#################
require(relaxo)
fit.relaxo = cvrelaxo(X,Y)

# estimated coefficients
fit.relaxo$beta[1:5]

## [1] 4.969345 4.963355 4.754283 4.865056 5.000046

sum(fit.relaxo$beta!=0)

## [1] 5

# test err
pred.relaxo = predict(fit.relaxo,Xnew)
mean((Ynew-pred.relaxo)^2)

## [1] 1.080831
```

Post-Selection Inference

- To conduct statistical inference for procedures that involve model selection, such as forward stepwise regression or the lasso, it is tempting to look only at the final selected model[10]. However, such inference is generally invalid.
- The problem is essentially the same as those of specification search and data-snooping: an observed correlation of 0.9 between x and y may be noteworthy. However, if x is found by searching over 100 variables looking for the one with the highest observed correlation with y, then the finding is no longer as impressive and could well be due to chance.
- If not taken into account, the effects of selection can greatly exaggerate the apparent strengths of relationships.

[10] For example, one may obtain $p-$values for the coefficients of interest by using OLS to refit the model selected by the lasso.

Post-Selection Inference

The problem of assessing the strength of the evidence after searching through a large number of models to find the best one is called **post-selection inference**[11].

Inference the old way (pre-1980?) :

1. Devise a model
2. Collect data
3. Test hypotheses

Classical inference

Inference the new way:

1. Collect data
2. Select a model
3. Test hypotheses

Post-selection inference

Source: Tibshirani (2015)

[11] or, **selective inference**.

Post-Selection Inference

One way to conduct post-selection inference is to make the inference *conditional* on the model selected[12].

Consider the following *variable selection* problem: we are interested in the linear relationship between a $p-$dimensional x and a target y. Using a selection algorithm, we arrive at the following model:

$$y = x_M' \beta_M + e \tag{13}$$

, where $M \subset \{1, \ldots, p\}$ indexes the subset of selected x variables.

[12]We have already discussed an approach designed to account for selection due to multiple-testing: the Bonferroni correction, which bounds the family-wise error rate.

Post-Selection Inference

Let β^*_M be the population coefficients of (13)[13]. How do we construct a confidence interval for $\beta^*_{M,j}$, $j \in M$[14]?

Classic CI: construct $\mathbb{C}_{M,j}$ such that

$$\Pr\left(\beta^*_{M,j} \in \mathbb{C}_{M,j}\right) \geq 1 - \alpha \tag{14}$$

[13] i.e.,

$$\beta^*_M = \arg\min_{\beta_M} E\left[\left(y - x'_M \beta_M\right)^2\right]$$

[14] $\beta^*_{M,j}$ is the population coefficient for $x_{M,j}$, which (1) is the j^{th} dimension of x; (2) belongs to the subset M. Note that the value of $\beta^*_{M,j}$ depends on what other variables are included in M.

Post-Selection Inference

Problem: here we do not start with model M. Instead, we start with $y = x'\beta + e$ and use a selection algorithm to find the optimal subset M. However, for every independent random sample $\mathcal{D} = \{x, y\}$ drawn from the underlying population, we may end up with a different selected model $M_{\mathcal{D}}$. $\beta^*_{M,j}$ is undefined when $M_{\mathcal{D}} \neq M$.

Solution: construct CI for $\beta^*_{M,j}$ *conditional* on M being selected:

$$\Pr\left(\beta^*_{M,j} \in \mathbb{C}_{M,j} \middle| M \text{ is selected}\right) \geq 1 - \alpha$$

Such an approach is called conditional coverage[15].

[15] Another approach is to assume the existence of a true model:

$$y = x'\beta_0 + e$$

, in which β_0 is sparse, and construct confidence intervals for β_0 when lasso-type estimators are applied. Such an approach requires assumptions about the correctness of the linear model and the sparsity of β_0.

Post-Selection Inference

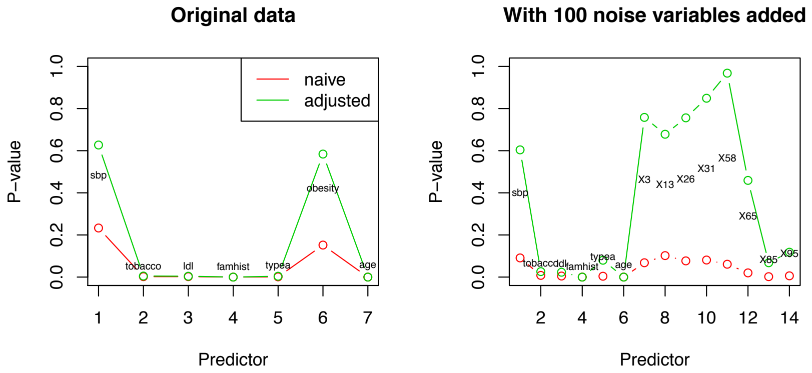


Figure 1: *South African Heart disease data. P-values from naive and selection-adjusted approaches, for original data (left) and data with 100 additional noise predictors (right). Each model was chosen by lasso-penalized logistic regression, choosing the tuning parameter by cross-validation.*

Taylor & Tibshirani (2017)

Simulation 3

```r
###########################################
# Post-selection inference for the lasso #
###########################################
# here we use a post-selection inference method
# for lasso-type estimators based on conditional coverage.
# See Tibshirani et al. (2016)
require(selectiveInference)
lassoInf = fixedLassoInf(X,Y,intercept=FALSE,
                         beta=fit.lasso$beta,
                         lambda=fit.lasso$lambda)
pv.lasso = lassoInf$pv

# compare with the naive method of
# running OLS on variables selected by the lasso
chosen.var = X[,which(fit.lasso$beta!=0)]
fit.olslasso = lm(Y ~ chosen.var-1)
pv.naive = summary(fit.olslasso)$coefficients[,4]
```

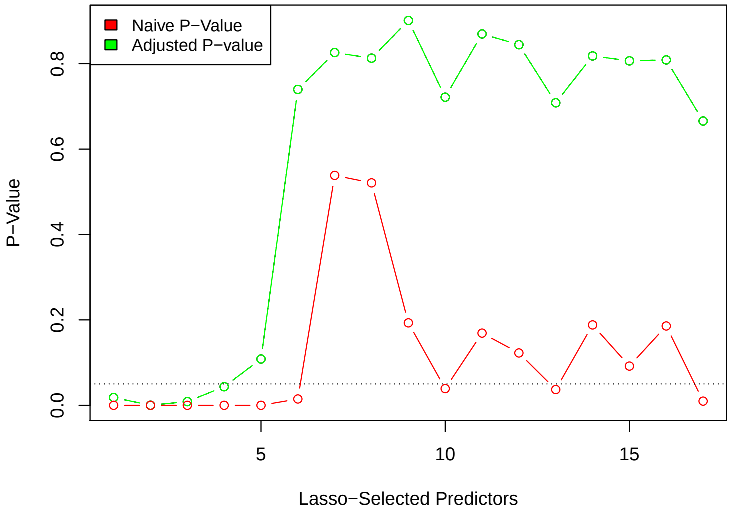
Simulation 3

```
coeftest(fit.olslasso)


##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## chosen.var1   4.964291   0.100367 49.4613 < 2.2e-16 ***
## chosen.var2   4.926740   0.091933 53.5907 < 2.2e-16 ***
## chosen.var3   4.744114   0.101547 46.7185 < 2.2e-16 ***
## chosen.var4   4.847510   0.096313 50.3308 < 2.2e-16 ***
## chosen.var5   4.926776   0.093042 52.9519 < 2.2e-16 ***
## chosen.var6  -0.223603   0.089858 -2.4884  0.014831 *
## chosen.var7   0.060767   0.098379  0.6177  0.538474
## chosen.var8   0.058576   0.090871  0.6446  0.520963
## chosen.var9   0.130765   0.099661  1.3121  0.193103
## chosen.var10  0.203057   0.096866  2.0963  0.039105 *
## chosen.var11  0.122677   0.088432  1.3873  0.169077
## chosen.var12  0.139490   0.089428  1.5598  0.122612
## chosen.var13 -0.199773   0.094191 -2.1209  0.036911 *
## chosen.var14 -0.117156   0.088313 -1.3266  0.188281
## chosen.var15  0.164892   0.096683  1.7055  0.091843 .
## chosen.var16 -0.117979   0.088426 -1.3342  0.185786
## chosen.var17 -0.258159   0.097588 -2.6454  0.009758 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 3

```
lassoInf

##
## Call:
## fixedLassoInf(x = X, y = Y, beta = fit.lasso$beta, lambda = fit.lasso$lambda,
##     intercept = FALSE)
##
## Standard deviation of noise (specified or estimated) sigma = 10.217
##
## Testing results at lambda = 0.139, with alpha = 0.100
##
##  Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
##    1  4.964   4.157   0.018     1.401   12.830       0.049       0.05
##    2  4.927   4.504   0.000     3.315   13.136       0.049       0.05
##    3  4.744   3.927   0.009     2.182   17.443       0.050       0.05
##    4  4.848   4.230   0.044     0.222   11.423       0.050       0.05
##    5  4.927   4.450   0.108    -1.999   12.436       0.050       0.05
##   18 -0.224  -0.209   0.740    -4.113   15.239       0.050       0.05
##   23  0.061   0.052   0.826   -68.087    9.614       0.050       0.05
##   37  0.059   0.054   0.813   -60.133    9.719       0.050       0.05
##   39  0.131   0.110   0.901   -32.436    1.198       0.050       0.05
##   40  0.203   0.176   0.722   -19.556    6.150       0.050       0.05
##   45  0.123   0.117   0.870   -27.107    2.008       0.050       0.05
##   77  0.139   0.131   0.845   -24.394    2.573       0.050       0.05
##   90 -0.200  -0.178   0.709    -6.415   18.764       0.050       0.05
```

Simulation 3



Shrinkage as Continuous Model Selection

- Consider a series of polynomial models: $\mathcal{H}_0 \subset \mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots$, where $\mathcal{H}_q = \{h(x) = \beta_0 + \beta_1 x + \cdots \beta_q x_q^q\}$, then estimating, say $\mathcal{H}_2$, is equivalent to estimating, say $\mathcal{H}_{10}$, with the constraint that $\beta_3 = \cdots = \beta_{10} = 0$[16].
- Instead of imposing such "hard constraints", shrinkage methods impose "soft constraints" by giving β a "budget" (e.g., $\|\beta\|_2^2 \leq C$ or $\|\beta\|_1 \leq C$) that have the effect of shrinking β toward 0.
- In this sense, shrinkage can be thought of as selecting the final hypothesis from a continuous set of models rather than a discrete one (constant, linear, quadratic ...).

[16] Technically, this is only true if the polynomials are expressed as linear combinations of Legendre polynomials in x rather than consecutive powers of x.

Regularization

- Shrinkage is a form of **regularization**: methods that constrain the complexity of a model in order to avoid overfitting and improve out-of-sample error.
- In least squares regression, given a model $\mathcal{H}$, we choose $g \in \mathcal{H}$ to minimize the in-sample error E_{in}. This is called **empirical risk minimization** (**ERM**).
- Regularization methods choose g by solving the following problem:

$$\min_{h \in \mathcal{H}} E_{in}(h) \quad \text{subject to} \quad \Omega(h) \leq C \tag{15}$$

, where $\Omega(h)$ is a measure of the complexity of h and is called a **regularizer**[17].

[17] When $\mathcal{H}$ is parametrized by β, $\Omega(h)$ can be written as $\Omega(\beta)$.

Regularization

- Equivalently, (15) can be written as

$$\begin{aligned} &\min_{h\in\mathcal{H}} \{E_{in}(h) + \lambda\Omega(h)\} \\ =&\min_{h\in\mathcal{H}} E_{aug}(h) \end{aligned} \tag{16}$$

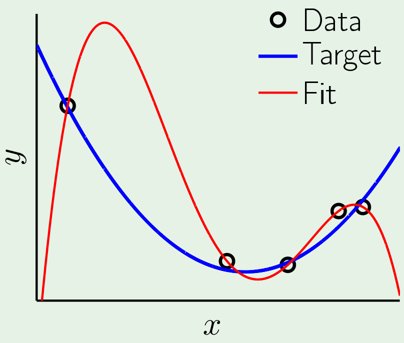
  , where we define **augmented error** $E_{aug}(h) \equiv E_{in}(h) + \lambda\Omega(h)$.

- (15) is called the **Ivanov form** and expresses regularization as constrained ERM.

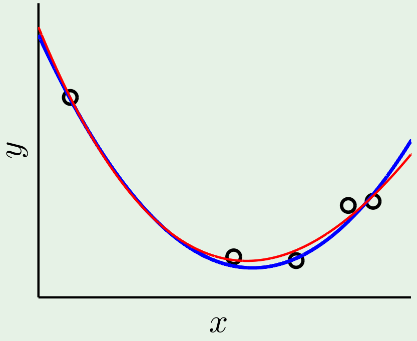
- (16) is called the **Tikhonov form** and expresses regularization as penalized ERM.

Regularization

Regularization as Constrained ERM



free fit



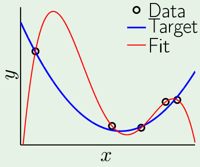
restrained fit

Regularization

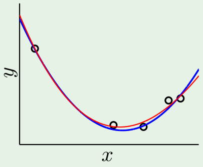
Regularization as Constrained ERM

Minimizing $E_{in}(\beta) + \lambda \|\beta\|_2^2$ for different λs

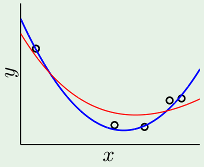
$\lambda = 0$



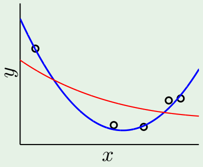
$\lambda = 0.0001$



$\lambda = 0.01$



$\lambda = 1$



overfitting $\longrightarrow$ $\longrightarrow$ $\longrightarrow$ $\longrightarrow$ underfitting

Regularization

Bias-Variance Decomposition

$f : [-1, 1] \to \mathbb{R} \qquad f(x) = \sin(\pi x)$

Only two training examples! $\quad N = 2$

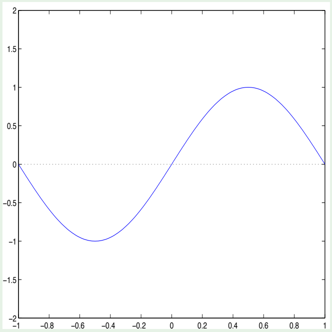
Two models used for learning:

$\mathcal{H}_0$: $\quad h(x) = b$

$\mathcal{H}_1$: $\quad h(x) = ax + b$

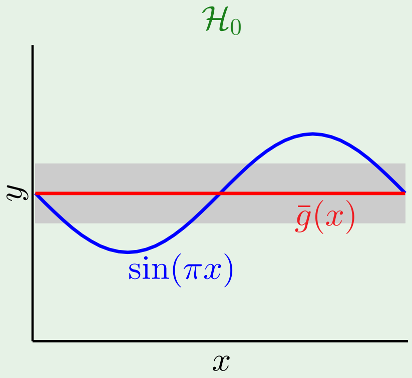
Which is better, $\mathcal{H}_0$ or $\mathcal{H}_1$?

f

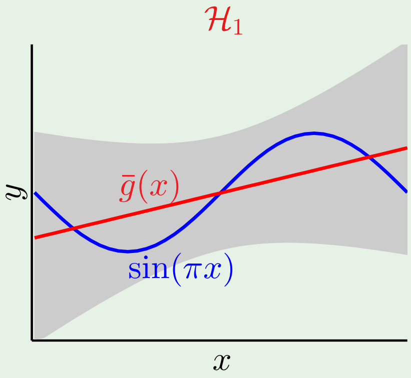


Regularization

Bias-Variance Decomposition



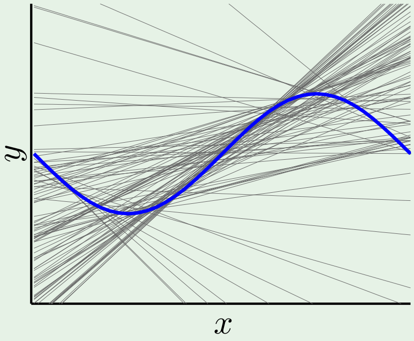
bias = $\mathbf{0.50}$ var = $\mathbf{0.25}$



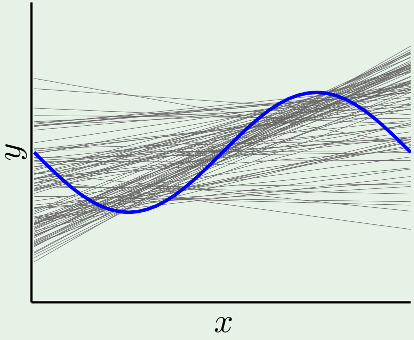
bias = $\mathbf{0.21}$ var = $\mathbf{1.69}$

Regularization

Bias-Variance Decomposition



without regularization

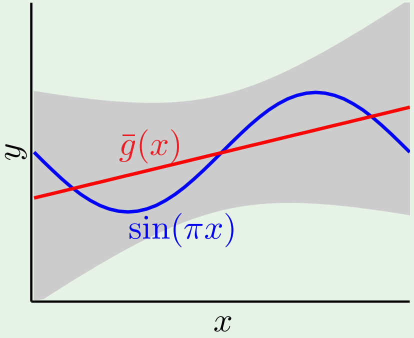


with regularization

Regularization

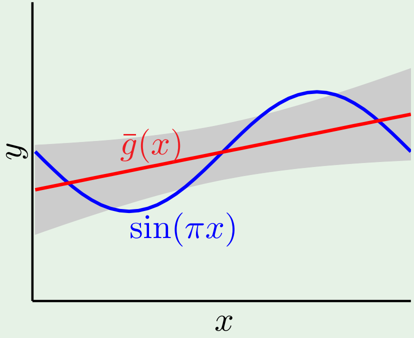
Bias-Variance Decomposition

without regularization



bias = $\mathbf{0.21}$ var = $\mathbf{1.69}$

with regularization



bias = $\mathbf{0.23}$ var = $\mathbf{0.33}$

Regularization

E_{aug} is a better proxy for E_{out} than E_{in}.

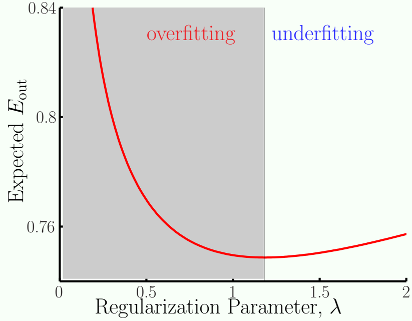
VC generalization bound:

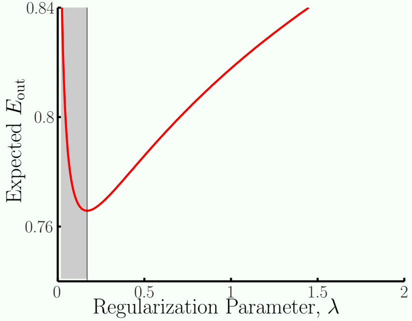
$$E_{out}(h) \leq E_{in}(h) + \mathcal{O}\left(\sqrt{d_{VC}(\mathcal{H})\frac{\ln N}{N}}\right)$$

Augmented error:

$$E_{aug}(h) = E_{in}(h) + \lambda\Omega(h)$$

Choosing a Regularizer

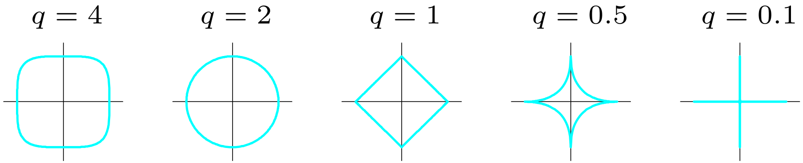




Simulated data set with the target function being a 15th degree polynomial.
Left: $\Omega(\beta) = \sum_{j=0}^{15} \beta_j^2$; Right: $\Omega(\beta) = \sum_{j=0}^{15} j\beta_j^2$

Choosing a Regularizer

L_0 **regularizer:** $\Omega(\beta) = \|\beta\|_0 = \sum_j \mathcal{I}\{\beta_j \neq 0\}$
L_1 **regularizer:** $\Omega(\beta) = \|\beta\|_1 = \sum_j |\beta_j|$
L_2 **regularizer:** $\Omega(\beta) = \|\beta\|_2^2 = \sum_j \beta_j^2$
L_q **regularizer:** $\Omega(\beta) = \|\beta\|_q^q = \sum_j |\beta_j|^q$



Contours of constant value of L_q regularizer for different values of q

Elastic Net

In the presence of highly correlated features, the results of the lasso can be arbitrary and unstable.

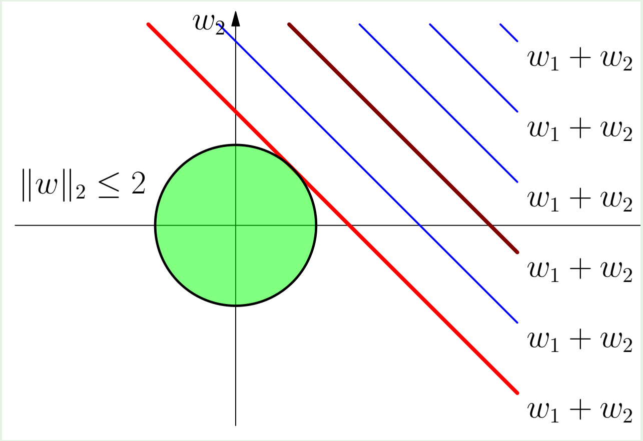
- Target function: $y = f(x) = 4x_1$; Model: $h(x) = w_1x_1 + w_2x_2$
- Suppose $x_1 = x_2$, then any $h(x)$ with $w_1 + w_2 = 4$ minimizes E_{in}, but different regularizers penalize them differently:

w_1	w_2	$\lVert w \rVert_1$	$\lVert w \rVert_2^2$
4	0	4	16
2	2	4	8
1	3	4	10
-1	5	6	26

- $\lVert w \rVert_1$ does not discriminate (as long as all have same sign)
- $\lVert w \rVert_2^2$ minimized when w is spread equally

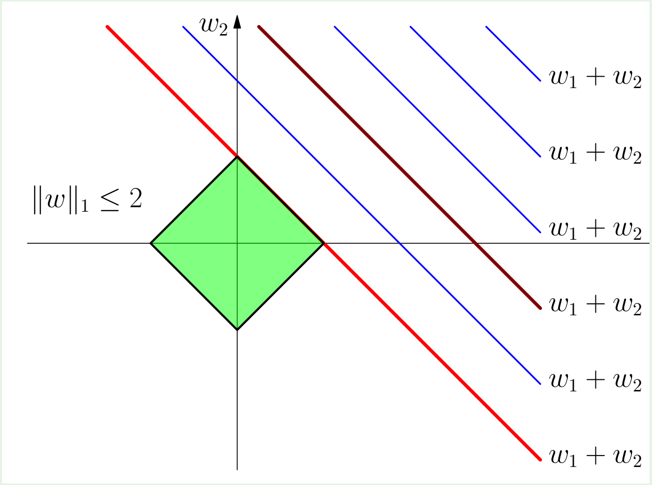
Elastic Net

Correlated Features (cont.)



Elastic Net

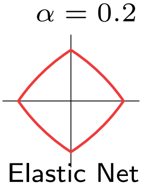
Correlated Features (cont.)



Elastic Net

The **elastic net** combines the lasso and ridge penalties:

$$\Omega(\beta) = \alpha \|\beta\|_1 + (1-\alpha)\|\beta\|_2^2$$



- The elastic net shrinks together the coefficients of correlated features like ridge regression, while encouraging sparse solutions like the lasso.

Elastic Net

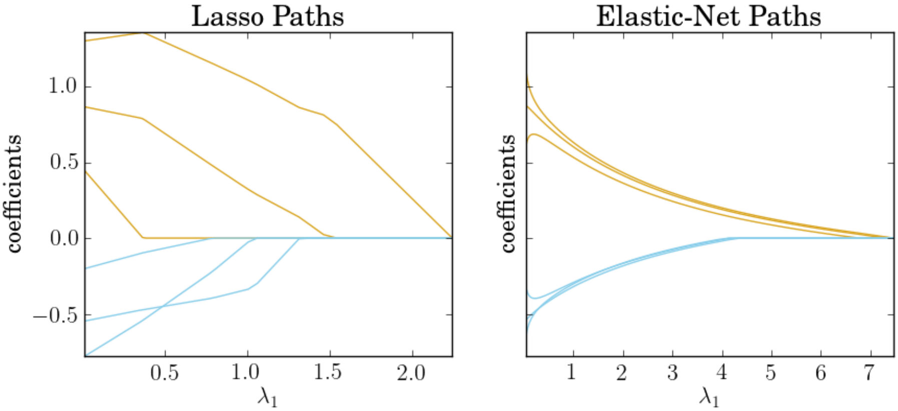
Simulation:

$$\begin{aligned} z_1, z_2, \epsilon_1, \ldots, \epsilon_6, e &\sim^{i.i.d.} \mathcal{N}(0,1) \\ y &= 3z_1 - 1.5z_2 + 2e \\ x_j &= \begin{cases} z_1 + 0.2\epsilon_j & j = 1,2,3 \\ z_2 + 0.2\epsilon_j & j = 4,5,6 \end{cases} \end{aligned}$$

Elastic Net

```r
# generate data
require(MASS)
require(glmnet)
n <- 100
z1 <- rnorm(n)
z2 <- rnorm(n)
e <- mvrnorm(n,mu=rep(0,6),Sigma=diag(6))
x1 <- z1 + e[,1:3]/5
x2 <- z2 + e[,4:6]/5
x <- cbind(x1,x2)
y <- 3*z1 - 1.5*z2 + 2*rnorm(n)

# lasso
lassofit <- glmnet(x,y,alpha=1)

# elastic net
enetfit <- glmnet(x,y,alpha=0.3) # 30% L1 and 70 %L2
```

Elastic Net



Lasso Paths
Elastic-Net Paths
coefficients
coefficients
1.0
0.5
0.0
−0.5
0.5
1.0
1.5
2.0
λ_1
1
2
3
4
5
6
7
λ_1

Bayesian Interpretation of Regularization

The lasso and ridge regression estimates correspond to the *maximum a posteriori* (*MAP*) estimates with Laplace and Gaussian priors[18].

- Ridge regression is the posterior mode[19] for β under a normal prior with mean zero.
- The lasso is the posterior mode for β under a double-exponential (Laplace) prior with mean zero.
- The standard deviation of the prior distribution corresponds to regularization strength (λ).

[18] See Appendix I for a simple derivation

[19] as well as the posterior mean

Bayesian Interpretation of Regularization

The **maximum à posteriori** (**MAP**) estimate is defined as

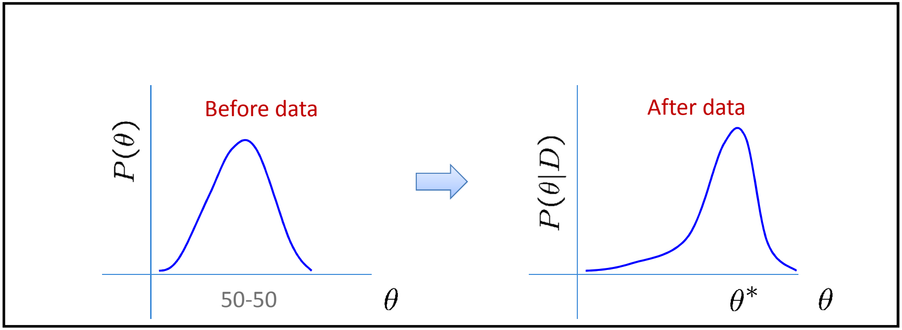
$$\begin{aligned}\widehat{\theta}_{MAP} &= \arg\max_{\theta} p(\theta|\mathcal{D}) \\ &= \arg\max_{\theta} p(\mathcal{D}|\theta)\, p(\theta)\end{aligned}$$

, i.e., $\widehat{\theta}_{MAP}$ is the mode of the posterior distribution of θ.

- Given a uniform prior $p(\theta) =$ constant, $\widehat{\theta}_{MAP} = \widehat{\theta}_{MLE}$[20].

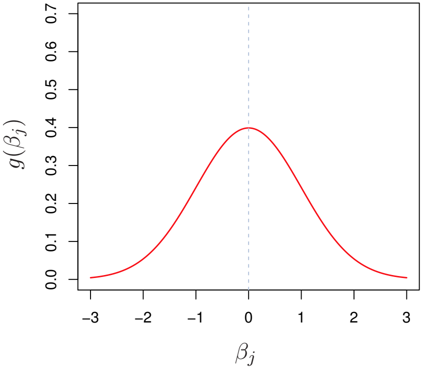
[20] In this case, the Bayesian estimate equals the frequentist estimate because the prior is *uninformative* (hence uniform priors are sometimes called **uninformative priors**). As a result, $\mathcal{D}$ becomes the only source of our knowledge for θ, as in the frequentist approach. If $p(\theta) =$ constant over the entire real line, then it is also an **improper prior**, since it does not integrate to one.

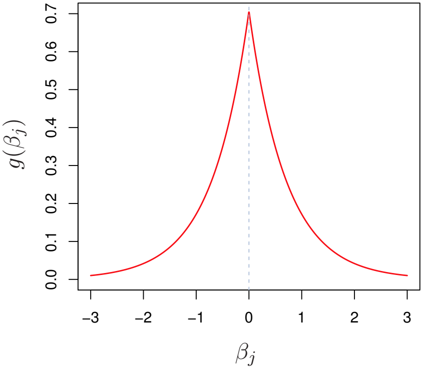
Bayesian Interpretation of Regularization



The Bayesian Way

Bayesian Interpretation of Regularization





Left: Gaussian prior for Ridge; Right: Laplace prior for the Lasso

Bayesian Interpretation of Regularization

- From a Bayesian perspective, regularization amounts to the use of *informative priors*, where we introduce our knowledge or belief about the target function in the form of priors, and use them to "regulate" the behavior of the hypothesis we choose[21].

[21] Recall that in choosing a model, our choice should be based both on our belief about the true target function *and* on the amount of data we have (matching model complexity to data sources). Similarly, the choice of regularizers is based both on our belief about the target function (such as its smoothness), and the necessity of tempering down the variability in our estimates when p is large relative to N.

Smoothing Splines

A smoothing spline is a function $g(x)$ that solves the following problem:

$$\min_h \left\{ \sum_{i=1}^{N} (y_i - h(x_i))^2 + \lambda \int \left[h''(t)\right]^2 dt \right\} \tag{17}$$

, where $\lambda \geq 0$ is a tuning parameter.

- The regularizer $\Omega(h) = \int [h''(t)]^2 dt$ is a roughness penalty. Hence λ regulates the smoothness of the spline.
- $\lambda = 0$: $g(x)$ *interpolates* every (x_i, y_i).
- $\lambda \to \infty$: $g(x)$ becomes the linear least squares line.

Smoothing Splines

The solution to (17) is a natural cubic spline with knots at *every* x_i[22,23,24].

Recall that the least squares solution for natural cubic splines is $\widehat{\beta}^{LS} = (\Phi'\Phi)^{-1}\Phi' Y$, where $\Phi(x)$ is the set of basis functions for natural cubic splines. Solving (17) gives the *shrinkage* solution:

$$\widehat{\beta} = \left(\Phi'\Phi + \lambda\Psi\right)^{-1}\Phi' Y$$

, where Ψ is an $N \times N$ matrix with $\Psi_{ij} = \int \Phi_i''(t)\,\Phi_j''(t)\,dt$.

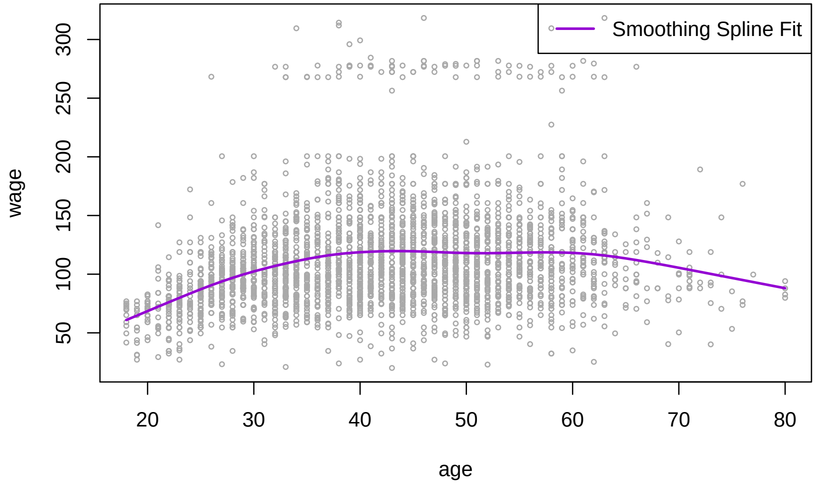
[22]Hence, we no longer have to choose knot placement!

[23]See Appendix II for proof.

[24]More generally, penalizing the squared k^{th} derivative leads to a natural spline of degree $2k-1$.

Wage Profile

```r
fit = smooth.spline(Wage$age,Wage$wage)
```



Appendix I: Gaussian Prior and L2 Regularization

Consider a normal linear model with Gaussian prior[25]:

$$\beta_j \sim \mathcal{N}\left(0, \tau^2\right), \;\; j = 0, \ldots, p$$
$$y \sim \mathcal{N}\left(\beta' x, \sigma^2\right)$$

Then we have:

$$\widehat{\theta}_{MAP} = \arg\max_{\beta} \left\{ \left(\prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\left(y_i - \beta' x_i\right)^2}{2\sigma^2}} \right) \left(\prod_{j=1}^{p} \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2\tau^2}} \right) \right\} \tag{18}$$

[25] Assume we know σ.

Appendix I: Gaussian Prior and L2 Regularization

(18) $\Rightarrow$

$$\begin{aligned}
\widehat{\theta}_{MAP} &= \arg\max_{\beta} \left\{ \log \left(\prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\beta' x_i)^2}{2\sigma^2}} \right) + \log \left(\prod_{j=1}^{p} \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2\tau^2}} \right) \right\} \\
&= \arg\max_{\beta} \left\{ -\sum_{i=1}^{N} \frac{(y_i-\beta' x_i)^2}{2\sigma^2} - \sum_{j=0}^{p} \frac{\beta_j^2}{2\tau^2} \right\} \\
&= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i-\beta' x_i)^2 + \lambda \sum_{j=0}^{p} \beta_j^2 \right\}
\end{aligned}$$

, where $\lambda = \frac{\sigma^2}{\tau^2}$[26].

[26] Hence smaller values of τ correspond to stronger regularization

Appendix I: Laplace Prior and L1 Regularization

Consider a normal linear model with Laplace prior[27]:

$$\beta_j \sim \text{Laplace}(\mu, b)\,, \;\; j = 0, \ldots, p$$
$$y \sim \mathcal{N}\left(\beta' x, \sigma^2\right)$$

Then we have:

$$\begin{aligned}\widehat{\theta}_{MAP} &= \arg\max_{\beta} \left\{ \left(\prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\left(y_i - \beta' x_i\right)^2}{2\sigma^2}} \right) \left(\prod_{j=1}^{p} \frac{1}{2b} e^{-\frac{\left|\beta_j\right|}{2b}} \right) \right\} \\ &= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta' x_i\right)^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right\}\end{aligned}$$

, where $\lambda = \frac{\sigma^2}{b}$[28].

[27] Assume we know σ.

[28] Smaller values of b correspond to stronger regularization

Appendix II: Smoothing Splines

Theorem

Let h be any differentiable function on $[a, b]$ for which $h(x_i) = z_i$ for $i = 1, \ldots, n$. Suppose $n > 2$, and that g is the natural cubic spline interpolant to the values $z_1, \ldots, z_n$ at points $x_1, \ldots, x_n$ with $a < x_1 < \cdots < x_n < b$. Then $\int (h'')^2 \geq \int (g'')^2$ with equality if and only if $h = g$.

Appendix II: Smoothing Splines

Proof.

Let $\ell = h - g$. So $\ell(x_i) = 0$ for $i = 1, \ldots, n$.

$$\int (h'')^2 = \int (g'' + \ell'')^2 = \int (g'')^2 + 2\int g''\ell'' + \int (\ell'')^2 \qquad (19)$$

Since

$$\begin{aligned}\int_a^b g''(x)\,\ell''(x)\,dx &= g''(x)\,\ell'(x)\big|_a^b - \int_a^b \ell'(x)\,g^{(3)}(x)\,dx \\ &= -\sum_{i=1}^{n-1} g^{(3)}\left(x_j^+\right)\int_{x_j}^{x_{j+1}} \ell'(x)\,dx \\ &= -\sum_{i=1}^{n-1} g^{(3)}\left(x_j^+\right)\left(\ell(x_{j+1}) - \ell(x_j)\right) = 0\end{aligned}$$

, (19) $\Rightarrow \int (h'')^2 \geq \int (g'')^2$. □

Acknowledgement I

Part of this lecture is based on the following sources:

- Abu-Mostafa, Y. S., M. Magdon-Ismail, and H. Lin. 2012. *Learning from Data*. AMLBook.
- Blei, D. M. *Interacting with data*. Lecture at Princeton University, retrieved on 2017.01.01. [link]
- Dave, R. *Advanced Scientific Computing*, Lecture at Harvard Institute for Applied Computational Science, retrieved on 2019.01.01. [link]
- Hastie, T., R. Tibshirani, and J. Friedmand. 2008. *The Elements of Statistical Learning* (2nd ed.). Springer.
- James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. *An Introduction to Statistical Learning: with Applications in R*. Springer.

Acknowledgement II

- Liang, F. *Statistical Learning: Nonparametric and Regularization Approaches*, Lecture at Duke University, retrieved on 2019.01.01. [link]
- Rosenberg, D. S. *Machine Learning and Computational Statistics*, Lecture at NYU Center for Data Science, retrieved on 2019.01.01. [link]
- Tibshirani, R. *Data Mining*, Lecture at Carnegie Mellon University, retrieved on 2017.01.01. [link]

Reference

Belloni, A. and V. Chernozhukov. 2013. "Least squares after model selection in high-dimensional sparse models," *Bernoulli*, 19(2).

Efron B., T. Hastie, I. Johnstone, and R. Tibshirani. 2004. "Least Angle Regression," *The Annals of Statistics*, 32(2).

Lee, J. D., D. L. Sun, Y. Sun, and J. Taylor. 2016. "Exact Post-selection Inference with Application to the Lasso," *The Annals of Statistics*, 44(3).

Meinshausen, N. 2007. "Relaxed Lasso," *Computational Statistics & Data Analysis*, 52(1).

Taylor, J. and R. Tibshirani. 2017. "Post-selection inference for ℓ_1-penalized likelihood models," *The Canadian Journal of Statistics*, 46(1).

Tibshirani, R. 2015. "Recent Advances in Post-Selection Statistical Inference," Breiman lecture, NIPS 2015.

Tibshirani, R. J., J. Taylor, R. Lockhart, and R. Tibshirani. 2016. "Exact Post-Selection Inference for Sequential Regression Procedures," *Journal of the American Statistical Association*, 111(514).