컴퓨터 통계 방법론 HW3

20180490 이재헌

1. Ch6-Prob4: Suppose we estimate the regression coefficients in a linear regression model by minimizing $\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j\ x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}{\beta_j}^2$ for a particular value of λ. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

   (a) As we increase λ from 0, the training RSS will steadily increase since the model complexity decreases and it gets less fitted to the training dataset.

   (b) As we increase λ from 0, the test RSS will decrease initially, and then eventually start increasing in a U shape. At first, when λ is small, the reduction of variance is greater than the increment of bias thus the test RSS decreases. However, in later stages, the error due to bias tends to surpass the error due to variance. Then the test RSS start increasing.

   (c) As we increase λ from 0, the variance will steadily decrease since the number of effective predictors decreases. When λ goes to infinity, the model is a constant and the prediction is always the same for any dataset.

   (d) As we increase λ from 0, the bias will steadily increase since the model complexity decreases.

   (e) As we increase λ from 0, the irreducible error remains constant since it is independent from the model.

2. Ch6-Prob5: It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting. Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0\ and\ x_{11} + x_{21} = 0\ and\ x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

   (a) Write out the ridge regression optimization problem in this setting.

$$\min_{\hat{\beta}_1,\hat{\beta}_2}\left(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12}\right)^2 + \left(y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22}\right)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$$

   (b) Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.

$$\min_{\hat{\beta}_1,\hat{\beta}_2}\left(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12}\right)^2 + \left(y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22}\right)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$$

$$= \min_{\hat{\beta}_1,\hat{\beta}_2}\left(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{11}\right)^2 + \left(-y_1 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{11}\right)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$$

$$= \min_{\hat{\beta}_1,\hat{\beta}_2} 2\left(y_1 - (\hat{\beta}_1 + \hat{\beta}_2)x_{11}\right)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2) =: f(\beta)$$

$$Let\ \beta = \left[\hat{\beta}_1, \hat{\beta}_2\right]^T, x = [x_{11}, x_{11}]^T, y = [y_1, y_1]^T$$

$$\frac{\partial f}{\partial \beta} = 4(y - x^T\beta)x + 2\lambda\beta = 0$$

$$\beta = \frac{2}{\lambda}(x^T\beta\ x - y)$$

$$Since\ elements\ of\ x\ and\ y\ are\ identical, \hat{\beta}_1 = \hat{\beta}_2.$$

(c) Write out the lasso optimization problem in this setting.

$$\min_{\hat{\beta}_1,\hat{\beta}_2}\left(y_1-\hat{\beta}_1x_{11}-\hat{\beta}_2x_{12}\right)^2+\left(y_2-\hat{\beta}_1x_{21}-\hat{\beta}_2x_{22}\right)^2+\lambda(|\hat{\beta}_1|+|\hat{\beta}_2|)$$

$$=\min_{\hat{\beta}_1,\hat{\beta}_2}2\left(y_1-(\hat{\beta}_1+\hat{\beta}_2)x_{11}\right)^2+\lambda(|\hat{\beta}_1|+|\hat{\beta}_2|)$$

(d) Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique—in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions.

Suppose $\hat{\beta}_1,\hat{\beta}_2$ are the optimal solutions. Assume $\hat{\beta}_1+\hat{\beta}_2>0$. Then, $\hat{\beta}_1,\hat{\beta}_2>0$ to minimize $\lambda(|\hat{\beta}_1|+|\hat{\beta}_2|)$. Then, for any $0<\varepsilon<\hat{\beta}_1$, $\hat{\beta}_1-\varepsilon,\hat{\beta}_2+\varepsilon$ are also optimal solutions. Therefore, the optimal solution for lasso problem in this setting is not unique.

3. Ch6-Prob7: We will now derive the Bayesian connection to the lasso and ridge regression discussed in Section 6.2.2.

(a) Suppose that $y_i=\beta_0+\sum_{j=1}^{p}x_{ij}\beta_j+\epsilon_i\ where\ \epsilon_1,\dots,\epsilon_n$ are independent and identically distributed from a $N(0,\sigma^2)$ distribution. Write out the likelihood for the data.

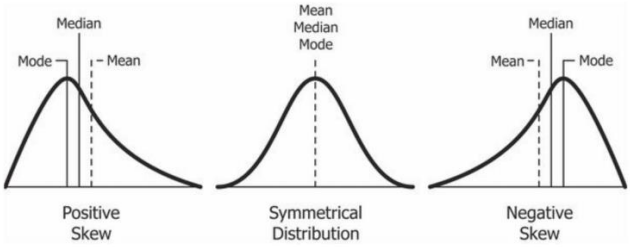
$$L(Y|X,\beta)=\prod_{i=1}^{p}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right)=\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}}\exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)$$

(b) Assume the following prior for β: $\beta_1,\dots,\beta_p$ are independent and identically distributed according to a double-exponential distribution with mean 0 and common scale parameter b: i.e. $p(\beta)=\frac{1}{2b}\exp\left(-\frac{|\beta|}{b}\right)$. Write out the posterior for β in this setting.

$$p(\beta|X,Y)\propto L(Y|X,\beta)p(X|\beta)=L(Y|X,\beta)p(\beta)$$

$$=\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}}\exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\frac{1}{2b}\exp\left(-\frac{|\beta|}{b}\right)$$

(c) Argue that the lasso estimate is the mode for β under this posterior distribution.



$$\operatorname*{argmax}_{\beta}p(\beta|X,Y)=\operatorname*{argmax}_{\beta}\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}}\exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\frac{1}{2b}\exp\left(-\frac{|\beta|}{b}\right)$$

$$=\operatorname*{argmax}_{\beta}\ln\left[\exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\exp\left(-\frac{|\beta|}{b}\right)\right]$$

$$=\operatorname*{argmax}_{\beta}-\frac{\sum_{i=1}^{p}\left(y_i-\left(\beta_0+\sum_{j=1}^{p}x_{ij}\beta_j\right)\right)^2}{2\sigma^2}-\frac{|\beta|}{b}$$

$$= \underset{\beta}{\operatorname{argmin}} \frac{\sum_{i=1}^{p} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right)^2}{2\sigma^2} + \frac{1}{b}\sum_{j=1}^{p} |\beta_j|$$

$$= \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{p} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$= \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{p} RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$where\ \lambda = \frac{2\sigma^2}{b}$$

Therefore, the lasso estimate is the mode for β under this posterior distribution.

(d) Now assume the following prior for β: $\beta_1, \ldots, \beta_p$ are independent and identically distributed according to a normal distribution with mean zero and variance c. Write out the posterior for β in this setting.

$$p(\beta|X,Y) \propto L(Y|X,\beta)p(X|\beta) = L(Y|X,\beta)p(\beta)$$

$$= \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\left(\frac{1}{2\pi c}\right)^{\frac{p}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\beta_i^2}{2c}\right)$$

(e) Argue that the ridge regression estimate is both the *mode* and the *mean* for β under this posterior distribution.

$$\underset{\beta}{\operatorname{argmax}}\, p(\beta|X,Y) = \underset{\beta}{\operatorname{argmax}} \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\left(\frac{1}{2\pi c}\right)^{\frac{p}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\beta_i^2}{2c}\right)$$

$$= \underset{\beta}{\operatorname{argmax}}\, \ln\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\epsilon_i^2}{2\sigma^2}\right)\left(\frac{1}{2\pi c}\right)^{\frac{p}{2}} \exp\left(-\frac{\sum_{i=1}^{p}\beta_i^2}{2c}\right)\right]$$

$$= \underset{\beta}{\operatorname{argmax}} -\frac{\sum_{i=1}^{p} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right)^2}{2\sigma^2} - \frac{\sum_{i=1}^{p}\beta_i^2}{2c}$$

$$= \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{p} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2$$

$$= \underset{\beta}{\operatorname{argmin}}\, RSS + \lambda \sum_{i=1}^{p} \beta_i^2$$

$$where\ \lambda = \frac{\sigma^2}{c}$$

Therefore, the ridge regression estimate is the mode for β under this posterior distribution.

Since the multiplication of two zero mean Gaussians is also Gaussian, $p(\beta|X,Y)$ is Gaussian. Therefore, the mean and the mode coincides, which means the ridge regression estimate is also the mean for β under this posterior distribution.

4. Ch6-Prob9: In this exercise, we will predict the number of applications received using the other variables in the College data set.

   (a) Split the data set into a training set and a test set.

   For the sake of simplicity we drop the variable 'private_yes'.

```
df = pd.read_csv('/content/drive/MyDrive/ISLP/College.csv')
df.head()
```

	Unnamed: 0	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
0	Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440
1	Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280
2	Adrian College	Yes	1428	1097	336	22	50	1036	99	11250
3	Agnes Scott College	Yes	417	349	137	60	89	510	63	12960
4	Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560

```
X = df.iloc[:,3:]
X = X.iloc[:,:-1]
y = df['Apps']
```

```
X.head()
```

	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal
0	1232	721	23	52	2885	537	7440	3300	450	2200
1	1924	512	16	29	2683	1227	12280	6450	750	1500
2	1097	336	22	50	1036	99	11250	3750	400	1165
3	349	137	60	89	510	63	12960	5450	450	875
4	146	55	16	44	249	869	7560	4120	800	1500

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

   (Default test size = 0.25)

   (b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
lr = LinearRegression()
lr.fit(X_train, y_train)
lr.coef_
```

```
array([ 1.64287449e+00, -1.03178491e+00,  5.85050643e+01, -1.71022448e+01,
        8.10279823e-02,  5.46024813e-02, -9.73462083e-02,  1.70821301e-01,
       -3.66767273e-02, -2.96038935e-02, -8.37796653e+00, -9.52492909e-01,
        1.14267523e+01,  1.74275049e+00,  4.79545160e-02])
```

```
lr.score(X_test, y_test)
```

```
0.8985132475318267
```

   Test Accuracy = 89.85%

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
rcv = RidgeCV(alphas=np.linspace(.01, 100, 1000),  cv=10)
```

```
rcv.fit(X_train_scaled, y_train)
```

```
RidgeCV
RidgeCV(alphas=array([1.00000000e-02, 1.10090090e-01, 2.10180180e-01, 3.10270270e-01,
       4.10360360e-01, 5.10450450e-01, 6.10540541e-01, 7.10630631e-01,
       8.10720721e-01, 9.10810811e-01, 1.01090090e+00, 1.11099099e+00,
       1.21108108e+00, 1.31117117e+00, 1.41126126e+00, 1.51135135e+00,
       1.61144144e+00, 1.71153153e+00, 1.81162162e+00, 1.91171171e+00,
       2.01180180e+00, 2.11189189e+0...
       9.76979279e+01, 9.77980180e+01, 9.78981081e+01, 9.79981982e+01,
       9.80982883e+01, 9.81983784e+01, 9.82984685e+01, 9.83985586e+01,
       9.84986486e+01, 9.85987387e+01, 9.86988288e+01, 9.87989189e+01,
       9.88990090e+01, 9.89990991e+01, 9.90991892e+01, 9.91992793e+01,
       9.92993694e+01, 9.93994595e+01, 9.94995495e+01, 9.95996396e+01,
```

```
rcv.alpha_
```

```
4.614144144144143
```

```
rcv.score(X_test_scaled,y_test)
```

```
0.9015429819480388
```

λ = 4.61, Test Accuracy = 90.15%

(d) Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lcv = LassoCV(alphas=np.linspace(.01, 100, 1000),  cv=10)
```

```
lcv.fit(X_train_scaled, y_train)
```

```
LassoCV
       8.10720721e-01, 9.10810811e-01, 1.01090090e+00, 1.11099099e+00,
       1.21108108e+00, 1.31117117e+00, 1.41126126e+00, 1.51135135e+00,
       1.61144144e+00, 1.71153153e+00, 1.81162162e+00, 1.91171171e+00,
       2.01180180e+00, 2.11189189e+0...
       9.76979279e+01, 9.77980180e+01, 9.78981081e+01, 9.79981982e+01,
       9.80982883e+01, 9.81983784e+01, 9.82984685e+01, 9.83985586e+01,
       9.84986486e+01, 9.85987387e+01, 9.86988288e+01, 9.87989189e+01,
       9.88990090e+01, 9.89990991e+01, 9.90991892e+01, 9.91992793e+01,
       9.92993694e+01, 9.93994595e+01, 9.94995495e+01, 9.95996396e+01,
       9.96997297e+01, 9.97998198e+01, 9.98999099e+01, 1.00000000e+02]),
        cv=10)
```

```
lcv.alpha_
```

```
19.02711711711712
```

```
lcv.score(X_test_scaled,y_test)
```

```
0.9000907344458458
```

$\lambda = 19.03$, Test Accuracy = 90.00%

```
lcv.coef_
```

```
array([3946.72376664, -309.21416794,  760.69013302, -121.03404821,
          0.        ,   71.88222884, -286.94323605,  149.04247508,
          0.        ,   -0.        ,  -95.07924902,   -8.6767995 ,
          7.4683291 ,   -0.        ,  216.36666862])
```

Non-zero coefficient estimates are: Accept, Enroll, Top10%, Top25%, P.Undergrad, Outstate, Room.Board, PhD, Terminal, S.F.Ratio, Expend

Zero coefficient estimates are: F.Undergrad, Books, Personal, perc.alumni

5. Ch6-Prob10: We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

   (a) Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model

   $$Y = X\beta + \epsilon,$$

   where β has some elements that are exactly equal to zero.

```
np.random.seed(0)

# Dataframe with random numbers and the specified dimensions
n = 1000
p = 20
X = pd.DataFrame(np.random.normal(size=(n, p)))

# Epsilon
epsilon = np.random.normal(size=n)

# Coefficient b1
b1 = np.random.normal(size=p)
# Random number of b1 elements with value zero
for i in range(0, np.random.randint(0,p)):
    b1[np.random.randint(0,p)] = 0

# Final expression
# y must be a vector with 1000 rows.
y = np.dot(X, b1) + epsilon
```

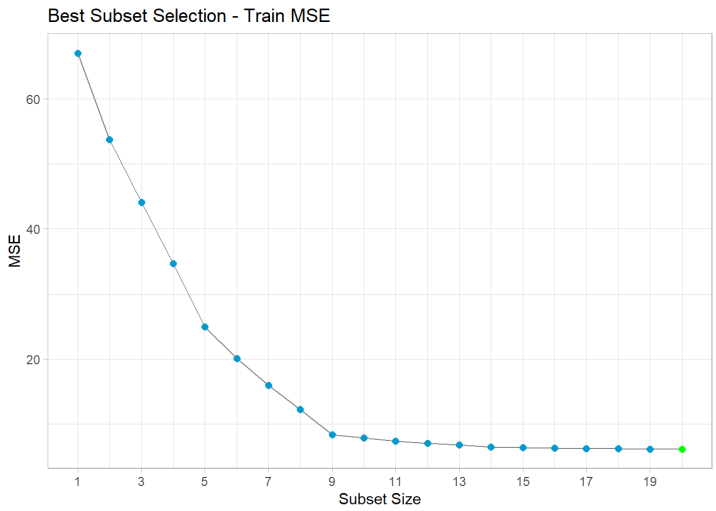
```
print(b1)
```

```
[-0.98566722 -1.6728118   1.18044823  1.78140295  0.0313383   0.12456057
  1.53409627  0.         -1.31750439  0.57227433  0.          1.04464875
 -0.97740152  1.40145792 -1.28768576  1.96530858 -0.25746227  0.87414065
  0.19235327  0.47711263]
```

   (b) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

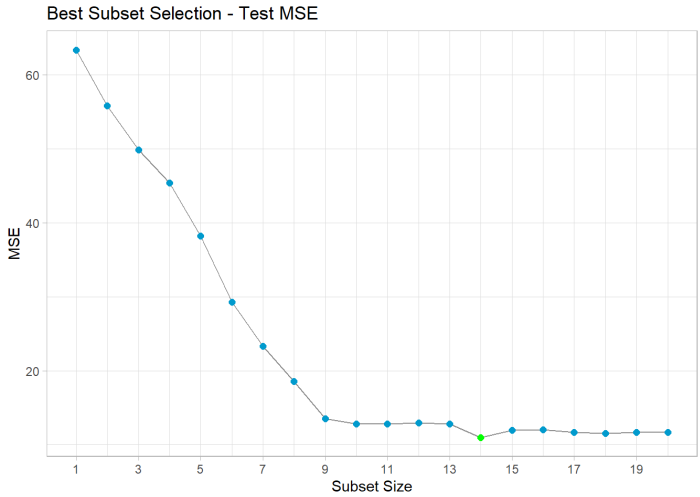
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .9)
```

(c) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.



Unsurprisingly, the model with the lowest train MSE is the model with 20 predictors. This is guaranteed, as adding additional predictors will only ever reduce the training RSS.

(d) Plot the test set MSE associated with the best model of each size.



(e) For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

The test MSE is minimized for the 14-variable model.

The test MSE graph paints quite a common picture - there is a lot of added benefit as the number of variables in the subset initially increases, then after a certain point larger subsets make very little difference (they actually begin to harm the model, as the increased variance from additional predictors outweighs the decrease in bias).

In practice, for very large datasets I would probably choose the 14-variable model, and for very small datasets such as this I would almost certainly choose the 9-variable model.

(f) How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.

```
##      parameter     actual  estimated
## 1  (Intercept)  0.0000000 -0.2263411
## 2           X1  1.4706970  0.5664113
## 3           X2  0.0000000  0.0000000
## 4           X3 -0.2843738  0.0000000
## 5           X4 -2.8696170 -2.8392890
## 6           X5  3.0773913  2.7177520
## 7           X6  0.4328998  0.0000000
## 8           X7 -1.8194582 -1.8207587
## 9           X8  2.4463609  2.8901977
## 10          X9  0.0000000  0.0000000
## 11         X10  2.4609806  3.3123983
## 12         X11  2.7583068  2.5165307
## 13         X12  2.1091128  1.8863706
## 14         X13  0.4308379  0.7166513
## 15         X14  0.0000000  0.0000000
## 16         X15 -0.4309777 -0.8653210
## 17         X16  0.4589441  0.6424018
## 18         X17  0.0000000  0.0000000
## 19         X18  4.3396136  4.0862468
## 20         X19  2.7845185  3.3514497
## 21         X20 -1.0309778 -0.8718763
```

It looks like best subset selection selected model estimates the true parameters pretty accurately, despite the fact that the true model is 16-variable and a 14-variable model was chosen.

Test R^2=0.8572

(g) Create a plot displaying $\sqrt{\sum_{j=1}^{p}\left(\beta_j - \hat{\beta}_j^r\right)^2}$ for a range of values of r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot from (d)?

What we observe is that the graph of parameter distances looks almost identical to the graph of the test MSE for different subset sizes, and that the model that minimizes this distance is also the 14-variable model.

To phrase this slightly differently: selecting a model based on the test MSE gave us a 14-variable model (can be applied to any dataset), and selecting a model based on how close the estimated parameters are to the true parameters (a luxury we can only afford when we know the true parameters) also led us to arrive at selecting the 14-variable model.

The fact that the graphs are similar is just showing that a model that has parameter estimates closer to the true parameters will also generally have a lower test MSE (makes sense).

6. Ch7-Prob1

It was mentioned in this chapter that a cubic regression spline with one knot at ξ can be obtained using a basis of the form x, x^2, x^3, $(x-\xi)^3_+$, where $(x-\xi)^3_+ = (x-\xi)^3$ if $x > \xi$ and equals 0 otherwise. We will now show that a function of the form

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$$

is indeed a cubic regression spline, regardless of the values of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(a) Find a cubic polynomial

$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$$

such that $f(x) = f_1(x)$ for all $x \leq \xi$. Express a_1, b_1, c_1, d_1 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(b) Find a cubic polynomial

$$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3$$

such that $f(x) = f_2(x)$ for all $x > \xi$. Express a_2, b_2, c_2, d_2 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$. We have now established that $f(x)$ is a piecewise polynomial.

(c) Show that $f_1(\xi) = f_2(\xi)$. That is, $f(x)$ is continuous at ξ.

(d) Show that $f_1'(\xi) = f_2'(\xi)$. That is, $f'(x)$ is continuous at ξ.

(e) Show that $f_1''(\xi) = f_2''(\xi)$. That is, $f''(x)$ is continuous at ξ.

Therefore, $f(x)$ is indeed a cubic spline.

(a) $a_1 = \beta_0,\ b_1 = \beta_1,\ c_1 = \beta_2,\ d_1 = \beta_3$

(b) $f_2(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3$

$$= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2) x + (\beta_2 - 3\beta_4 \xi) x^2 + (\beta_3 + \beta_4) x^3.$$

$$a_2 = (\beta_0 - \beta_4 \xi^3), \qquad b_2 = (\beta_1 + 3\beta_4 \xi^2), \qquad c_2 = (\beta_2 - 3\beta_4 \xi), \qquad d_2 = (\beta_3 + \beta_4)$$

(c) $f_1(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3,$

$$f_2(\xi) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\xi + (\beta_2 - 3\beta_4 \xi)\xi^2 + (\beta_3 + \beta_4)\xi^3 = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3$$

(d) $f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$

$$f_2'(\xi) = (\beta_1 + 3\beta_4 \xi^2) + 2(\beta_2 - 3\beta_4 \xi)\xi + 3(\beta_3 + \beta_4)\xi^2 = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2.$$

(e) $f_1''(\xi) = 2\beta_2 + 6\beta_3 \xi$

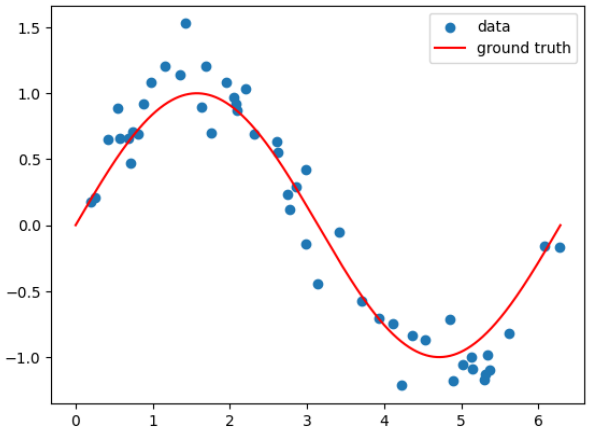
$$f_2''(\xi) = 2(\beta_2 - 3\beta_4 \xi) + 6(\beta_3 + \beta_4)\xi = 2\beta_2 + 6\beta_3 \xi\ .$$

7. Ch7-Prob2: Suppose that a curve $\hat{g}$ is computed to smoothly fit a set of n points using the following formula:

$$\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(m)}(x) \right]^2 dx \right)$$

where $g^{(m)}$ represents the m^{th} derivative of g (and $g^{(0)} = g$). Provide example sketches of $\hat{g}$ in each of the following scenarios.

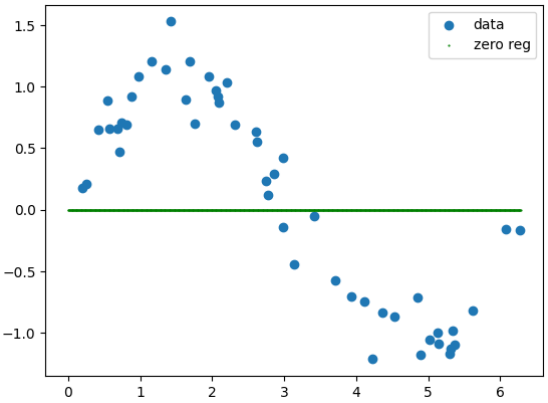
Let $Y = g(x) + \epsilon$ where $g(X) = \sin(X)$, $X \sim U[0, 2\pi]$, $\epsilon \sim N(0, 0.2^2)$



(a) λ = ∞, m = 0.

For m = 0, we have $\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int [g(x)]^2 dx \right)$.

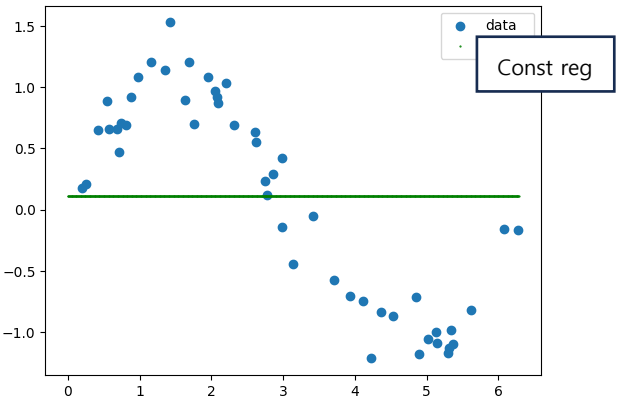
As λ goes to infinity, it forces $g(x) \to 0$.



(b) λ = ∞, m = 1.

For m = 1, we have $\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int [g'(x)]^2 dx \right)$.

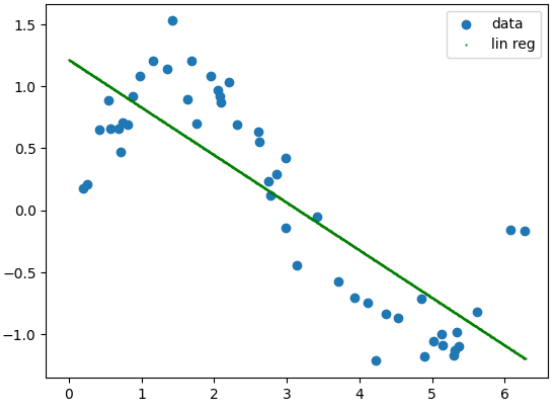
As λ goes to infinity, it forces $g'(x) \to 0$. Therefore, g must be constant.



(c) λ = ∞, m = 2.

For m = 2, we have $\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int [g''(x)]^2 dx \right)$.

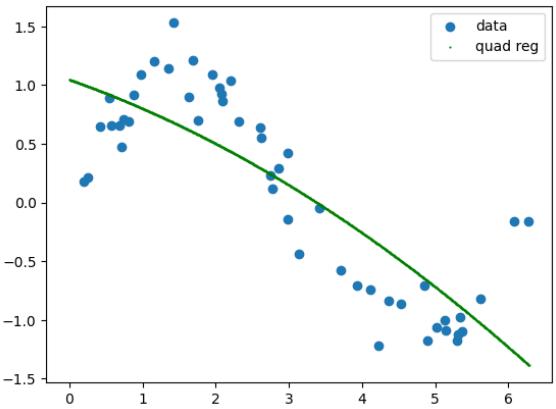
As λ goes to infinity, it forces $g''(x) \to 0$. Therefore, g must be linear.



(d) λ = ∞, m = 3.

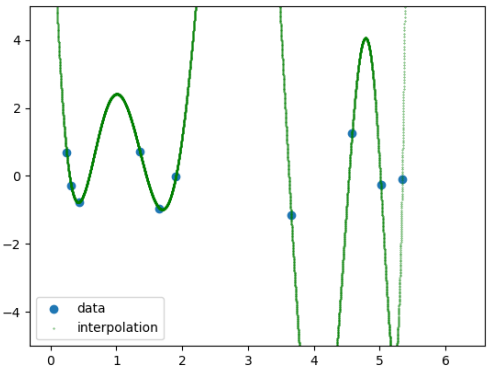
For m = 3, we have $\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int [g'''(x)]^2 dx \right)$.

As λ goes to infinity, it forces $g'''(x) \to 0$. Therefore, g must be quadratic.



(e) λ = 0, m = 3.

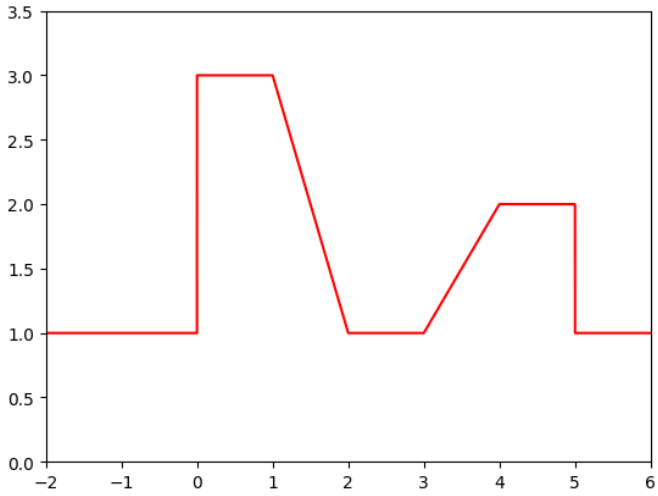
Since λ is 0, we minimize $\sum_{i=1}^{n} (y_i - g(x_i))^2$. $\sum_{i=1}^{n} (y_i - g(x_i))^2$ is minimized if g interpolates all y_i. The function would look like the below.



8. Ch7-Prob4: Suppose we fit a curve with basis functions $b_1(X) = I(0 \leq X \leq 2) - (X-1)I(1 \leq X \leq 2)$, $b_2(X) = (X-3)I(3 \leq X \leq 4) + I(4 < X \leq 5)$. We fit the linear regression model

   $Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon$, and obtain coefficient estimates $\hat{\beta}_0 = 1, \hat{\beta}_1 = 2, \hat{\beta}_2 = 3$. Sketch the estimated curve between $X = -2$ and $X = 6$. Note the intercepts, slopes, and other relevant information.

$$\hat{Y} = 1 + 2b_1(X) + 3b_2(X)$$



9. Ch7-Prob10: This question relates to the College data set.

   (a) Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

```
df = pd.read_csv('/content/drive/MyDrive/ISLP/College.csv')
df1 = pd.get_dummies(df.iloc[:,1:])
df = pd.concat([df.iloc[:,0],df1],axis=1)
df.head()
```

	Unnamed: 0	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books
0	Abilene Christian University	1660	1232	721	23	52	2885	537	7440	3300	450
1	Adelphi University	2186	1924	512	16	29	2683	1227	12280	6450	750
2	Adrian College	1428	1097	336	22	50	1036	99	11250	3750	400
3	Agnes Scott College	417	349	137	60	89	510	63	12960	5450	450
4	Alaska Pacific University	193	146	55	16	44	249	869	7560	4120	800

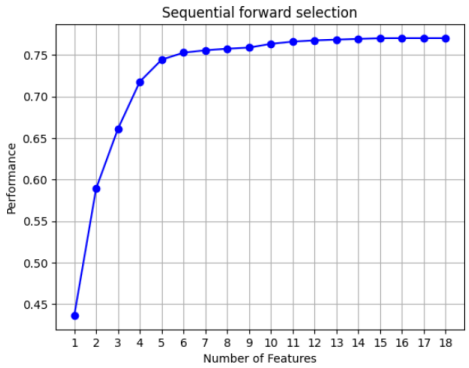
We need to make 'Private' as a dummy variable.

```
X.head()
```

	Unnamed: 0	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Private_No	Private_Yes
0	Abilene Christian University	1660	1232	721	23	52	2885	537	0	1
1	Adelphi University	2186	1924	512	16	29	2683	1227	0	1
2	Adrian College	1428	1097	336	22	50	1036	99	0	1
3	Agnes Scott College	417	349	137	60	89	510	63	0	1
4	Alaska Pacific University	193	146	55	16	44	249	869	0	1

Forward Selection:

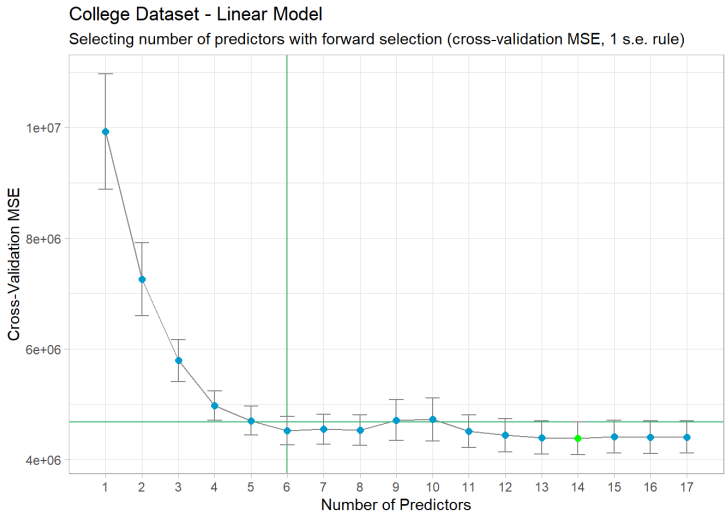
```
lr = LinearRegression()

from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
sfs = SFS(lr,
          k_features = 18,  # We have 18 features
          forward = True,
          floating = False,
          scoring = 'r2',
          cv = 0)

sfs = sfs.fit(X_train.iloc[:,1:], y_train)  # as_matrix() to be readable by sfs

fig = plot_sfs(sfs.get_metric_dict())

#plt.title('Sequential forward selection (w. StdDev)')
plt.title('Sequential forward selection')
plt.grid()
plt.show()
```



We will choose 6 features. The figure shows that a larger number of features will not increase the performance significantly.

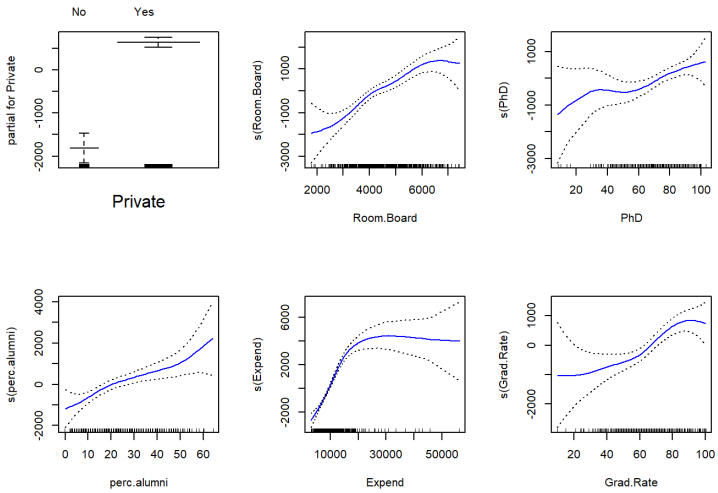
Variables: Private_No, Room.Board, PhD, perc.alumni, Expend, Grad.Rate

Also, we can check test MSE with cross validation.



The model with 14 predictors has the lowest test MSE. However, considering simplicity, the model with 6 predictors shows a good performance.

(b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.



It seems that there is obvious evidence of a nonlinear effect of Expend, whereas a linear relationship seems more reasonable for perc.alumni.

(c) Evaluate the model obtained on the test set, and explain the results obtained.

$$R^2 = 0.7712156$$

(d) For which variables, if any, is there evidence of a non-linear relationship with the response?

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board) + s(PhD) + s(perc.alumni) +
##     s(Expend) + s(Grad.Rate), data = train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7511.91 -1172.59    38.71  1267.44  7827.85
##
## (Dispersion Parameter for gaussian family taken to be 3617918)
##
##     Null Deviance: 9263945675 on 543 degrees of freedom
## Residual Deviance: 1888551843 on 521.9997 degrees of freedom
## AIC: 9782.515
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private         1 2402224634 2402224634 663.980 < 2.2e-16 ***
## s(Room.Board)   1 1721929895 1721929895 475.945 < 2.2e-16 ***
## s(PhD)          1  620590394  620590394 171.532 < 2.2e-16 ***
## s(perc.alumni)  1  456743095  456743095 126.245 < 2.2e-16 ***
## s(Expend)       1  746743434  746743434 206.401 < 2.2e-16 ***
## s(Grad.Rate)    1   99786036   99786036  27.581 2.197e-07 ***
## Residuals     522 1888551843    3617918
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                Npar Df Npar F   Pr(F)
## (Intercept)
## Private
## s(Room.Board)        3  2.680 0.04629 *
## s(PhD)               3  1.179 0.31718
## s(perc.alumni)       3  0.937 0.42233
## s(Expend)            3 32.503 < 2e-16 ***
## s(Grad.Rate)         3  1.995 0.11380
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looking at p-values in the bottom table ('Anova for Nonparametric Effects') we are performing tests to determine whether smooth/non-linear effects are present, where significance ($p<0.05$) indicates the presence of a non-linear effect.

The smallest p-value is for the smooth effect of Expend, with Room.Board also being

significant. This is the evidence that 'Expend' and 'Room.Board' has a non-linear relationship.

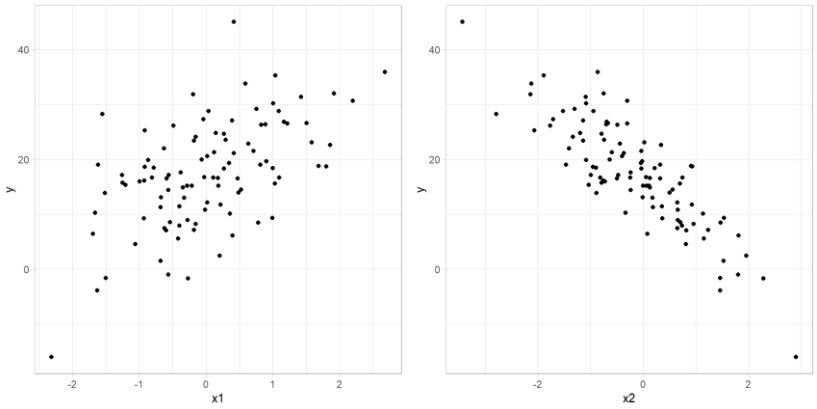
10. Ch7-Prob11: In Section 7.7, it was mentioned that GAMs are generally fit using a backfitting approach. The idea behind backfitting is actually quite simple. We will now explore backfitting in the context of multiple linear regression.

    Suppose that we would like to perform multiple linear regression, but we do not have software to do so. Instead, we only have software to perform simple linear regression. Therefore, we take the following iterative approach: we repeatedly hold all but one coefficient estimate fixed at its current value, and update only that coefficient estimate using a simple linear regression. The process is continued until convergence—that is, until the coefficient estimates stop changing.

    We now try this out on a toy example.

    (a) Generate a response Y and two predictors X1 and X2, with n = 100.

$$
\begin{aligned}
Y &= 16 + 5.1X_1 - 7.3X_2 + \epsilon \\
X_1 &\sim \mathcal{N}(0,1) \\
X_2 &\sim \mathcal{N}(0,1) \\
\epsilon &\sim \mathcal{N}(0,1)
\end{aligned}
$$



    (b) Write a function simple_reg() that takes two arguments outcome and feature, fits a simple linear regression model with this outcome and feature, and returns the estimated intercept and slope.

    (c) Initialize beta1 to take on a value of your choice. It does not matter what value you choose.

    Let $\hat{\beta}_1 = 1$.

    (d) Keeping beta1 fixed, use your function simple_reg() to fit the model:

$$Y - \texttt{beta1} \cdot X_1 = \beta_0 + \beta_2 X_2 + \epsilon.$$

    Store the resulting values as beta0 and beta2.

    We can do this as follows:

$$\hat{\beta}_2 = -7.3969$$

    (e) Keeping beta2 fixed, fit the model

$$Y - \texttt{beta2} \cdot X_2 = \beta_0 + \beta_1 X_1 + \epsilon.$$

Store the result as beta0 and beta1 (overwriting their previous values).

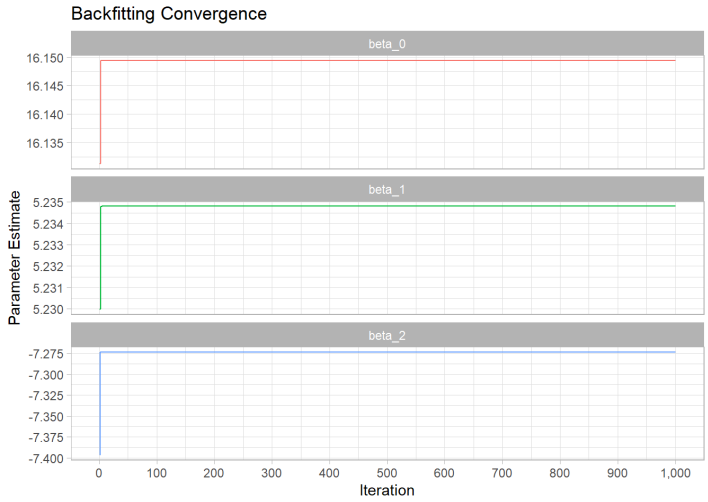
$\hat{\beta}_1 = 5.23$

$r_1 = \hat{\beta}_0 + \hat{\beta}_1 X + \epsilon$

$\hat{\beta}_0 = 16.13$

(f) Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of beta0, beta1, and beta2 at each iteration of the for loop. Create a plot in which each of these values is displayed, with beta0, beta1, and beta2.

The question says that this process continues until convergence, where the estimated coefficients no longer change. It appears from the graph that this happens very quickly:

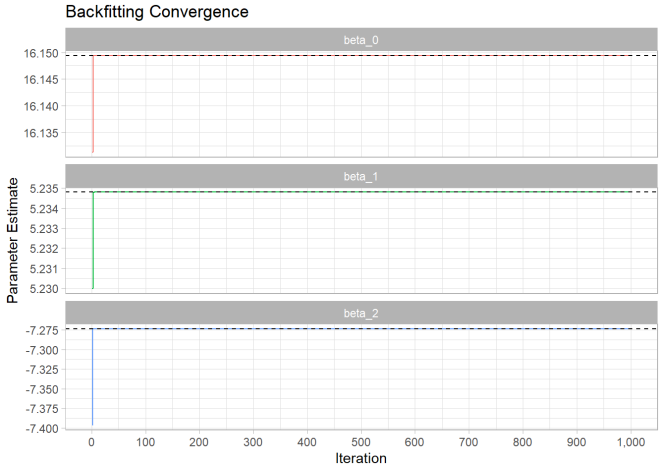


(g) Compare your answer in (e) to the results of simply performing multiple linear regression to predict Y using X1 and X2. Use axline() method to overlay those multiple linear regression coefficient estimates on the plot obtained in (e).

The fitted coefficients:

```
## (Intercept)          x1          x2
##   16.149420    5.234804   -7.273397
```

Overlaying these coefficients onto the previous plot (with a dashed line), it appears that the looped simple linear regressions via backfitting gave the same (or very close) parameter estimates to those provided by multiple regression, once they had converged:



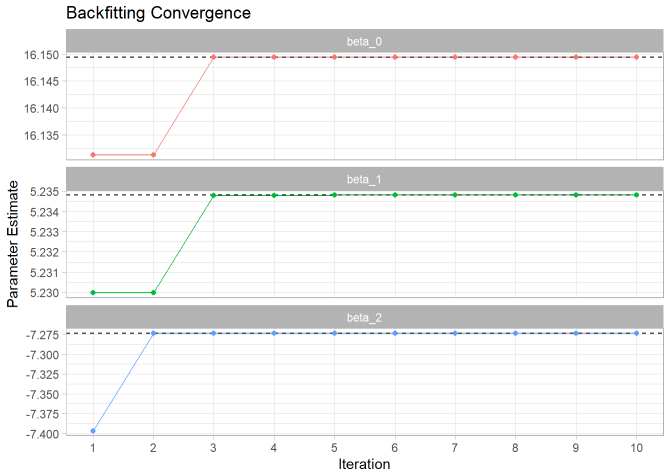
Extracting the backfitting coefficients at iteration 1,000 and comparing to those of multiple regression:

```
##      simple_reg multiple_reg
## 1    16.149420     16.149420
## 2     5.234804      5.234804
## 3    -7.273397     -7.273397
```

It certainly looks like they are identical.

(h) On this data set, how many backfitting iterations were required in order to obtain a "good" approximation to the multiple regression coefficient estimates?

For this simple example, just one iteration gave us a 'good' approximation of multiple regression coefficients:



It looks like convergence happens at iteration 3 from the graph, and that all subsequent iterations give the same estimates.

A closer inspection shows that perhaps the coefficients haven't completely converged by this point: