Exercises week 36 - Markus Bjørklund

September 8, 2023

1 Exercise 1

1.1 a)

We start with the mean squared error term

$$\mathbf{C}\left(\mathbf{X},\beta\right) = \Big(\left(\mathbf{y}-\mathbf{X}\beta\right)^T\left(\mathbf{y}-\mathbf{X}\beta\right)\Big) + \lambda\beta^T\beta, \tag{1}$$

where the trivial constant n has been baked into λ. Since we are searching for the optimal parameter $\hat{\beta}_{\text{Ridge}}$, what we want is to solve for β in the expression

$$\frac{\partial C}{\partial \beta} = 0 \tag{2}$$

Using the two results from last week,

$$\frac{\partial\left(\mathbf{x}-\mathbf{A}\mathbf{s}\right)^T\left(\mathbf{x}-\mathbf{A}\mathbf{s}\right)}{\partial \mathbf{s}} = -2\left(\mathbf{x}-\mathbf{A}\mathbf{s}\right)^T\mathbf{A} \tag{3}$$

and

$$\frac{\partial\left(\mathbf{b}^T\mathbf{a}\right)}{\partial \mathbf{z}} = \mathbf{a}^T\frac{\partial \mathbf{b}}{\partial \mathbf{z}} + \mathbf{b}^T\frac{\partial \mathbf{a}}{\partial \mathbf{z}} \tag{4}$$

we get that

$$\begin{aligned}
\frac{\partial C}{\partial \beta} =& -2\left(\mathbf{y}-\mathbf{X}\beta\right)^T\mathbf{X} + \lambda\left(\beta^T\frac{\partial\beta}{\partial\beta} + \beta^T\frac{\partial\beta}{\partial\beta}\right) = 0\\
&-2\left(\mathbf{y}-\mathbf{X}\beta\right)^T\mathbf{X} + 2\lambda\beta^T = 0\\
&-\mathbf{y}^T\mathbf{X} + \left(\mathbf{X}\beta\right)^T\mathbf{X} + \lambda\beta^T = 0\\
&\beta^T\mathbf{X}^T\mathbf{X} + \underbrace{\beta^T\lambda}_{\lambda\text{ scalar}} = \mathbf{y}^T\mathbf{X}\\
&\beta^T\left(\mathbf{X}^T\mathbf{X} + \mathbf{I}\lambda\right) = \mathbf{y}^T\mathbf{X}
\end{aligned}$$

Transposing both sides by $(AB)^T = \mathbf{B}^T\mathbf{A}^T$, and noting that $\mathbf{I}^T = \mathbf{I}, \lambda^T = \lambda$, and $\left(\mathbf{X}^T\mathbf{X}\right)^T = \mathbf{X}^T\left(\mathbf{X}^T\right)^T = \mathbf{X}^T\mathbf{X}$, we get

$$\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\beta = \mathbf{X}^T\mathbf{y}$$
$$\beta = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

1.2 b)

Using the singular value decomposition for $\mathbf{X}$,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5}$$

starts us off with transforming the expression for Ordinary Least Squares

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{X}\beta \tag{6}$$

into

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\beta \tag{7}$$

Now the Ordinary Least Squares solution for β is given by

$$\hat{\beta}_{\text{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{8}$$

Inserted, we get

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^{-1}\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T\mathbf{y} \tag{9}$$

Using $(\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_n)^T = \mathbf{A}_n^T\cdots\mathbf{A}_2^T\mathbf{A}_1^T$, we obtain

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}. \tag{10}$$

Since $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, and we get

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\right)^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}. \tag{11}$$

Using $(\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_n)^{-1} = \mathbf{A}_n^{-1}\cdots\mathbf{A}_2^{-1}\mathbf{A}_1^{-1}$, we get

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\mathbf{V}^T\right)^{-1}\left(\mathbf{\Sigma}^2\right)^{-1}(\mathbf{V})^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}. \tag{12}$$

Naturally, matrices cancel with their inverse, and all diagonal matrices commute with eachother, such that $\mathbf{\Sigma}\left(\mathbf{\Sigma}^2\right)^{-1}\mathbf{\Sigma}^T$ cancels, and we are left with

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{U}\mathbf{U}^T\mathbf{y}. \tag{13}$$

Note that if $\mathbf{\Sigma}$ contains p singular values, the relevant summation limit for this expression would be

$$\tilde{\mathbf{y}}_{\text{OLS}} = \sum_{j=0}^{p-1} \mathbf{u}_j \mathbf{u}_j^T \mathbf{y} \tag{14}$$

For ridge regression, we have that

$$\beta = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}. \tag{15}$$

Inserted, we get the expression

$$\begin{aligned}
\tilde{\mathbf{y}}_{\text{Ridge}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y} \\
&= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \left(\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T + \lambda\mathbf{I}\right)^{-1} \left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T \mathbf{y} \\
&= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \left(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T + \lambda\mathbf{I}\right)^{-1} \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y} \\
&= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \left(\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T + \lambda\mathbf{I}\right)^{-1} \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}
\end{aligned}$$

Now, since $\lambda\mathbf{I}$ is commmutative with everything, and $\mathbf{V}\mathbf{V}^T = \mathbf{I}$, we can rewrite

$$\left(\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T + \lambda\mathbf{I}\right)^{-1} = \left(\mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbf{I}\right]\mathbf{V}^T\right)^{-1} = \left(\mathbf{V}^T\right)^{-1}\left(\mathbf{\Sigma}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{V}^{-1} \tag{16}$$

$$\begin{aligned}
\tilde{\mathbf{y}}_{\text{Ridge}} &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \left(\mathbf{V}^T\right)^{-1}\left(\mathbf{\Sigma}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{V}^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y} \\
&= \mathbf{U}\mathbf{\Sigma}\left(\mathbf{\Sigma}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}
\end{aligned}$$

Now since $\mathbf{\Sigma}$ and $\lambda\mathbf{I}$ are diagonal matrices, and $\mathbf{\Sigma}^2$ is a $p \times p$ matrix, we recognize that only one sum over p diagonal elements survive. So, in summation notation, the two remaining $\mathbf{\Sigma}$ and $\mathbf{\Sigma}^T$ give contribution σ^2 in the numerator, while the inverse $\left(\mathbf{\Sigma}^2 + \lambda\mathbf{I}\right)^{-1}$ gives contribution $\sigma^2 + \lambda$ in the denominator. This gives us

$$\tilde{\mathbf{y}}_{\text{Ridge}} = \sum_{j=0}^{p-1} \mathbf{u}_j \mathbf{u}_j^T \frac{\sigma_j^2}{\sigma_j^2 + \lambda}\mathbf{y} \tag{17}$$

2 Exercise 2

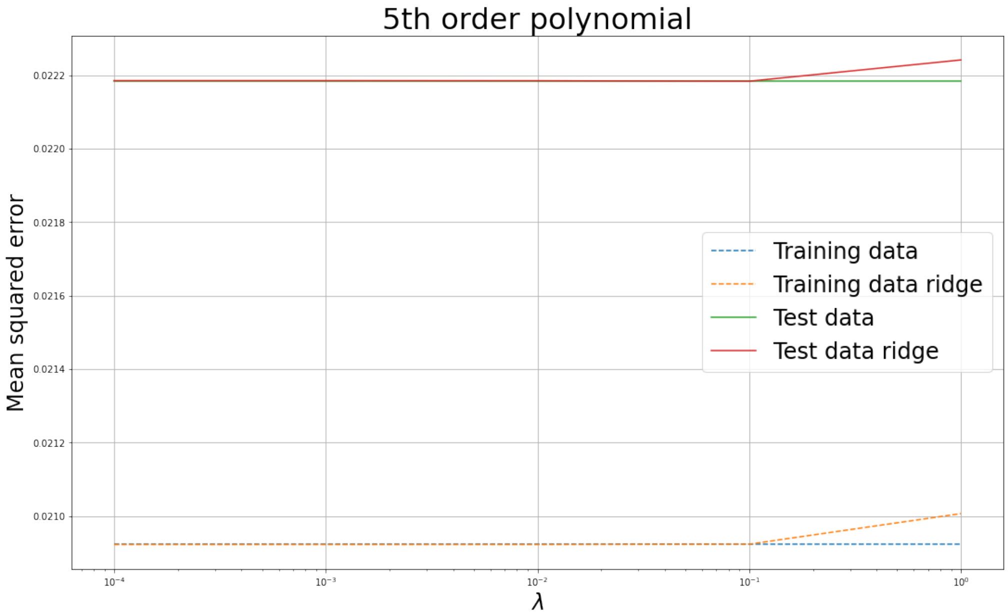
[69]:
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures


np.random.seed(1)
n = 100

noise_coeff2 = 1
x = np.linspace(-3, 3, n)
y = np.exp(-x**2) + 1.5 * np.exp(-(x-2)**2) + noise_coeff2*np.random.normal(0, 0.
↪1, x.shape)

def poly_train_func(x,y,p, lmbda_list):
    "Input p is polynomial order + 1"
    X = np.zeros((len(x), p-1))
    for i in range(p-1):
        X[:,i] = x**(i+1)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33)

    #Scaling
    y_train_mean = np.mean(y_train)
    X_train_mean = np.mean(X_train, axis=0)
    X_train = X_train - X_train_mean
    y_train = y_train - y_train_mean
    X_test = X_test - X_train_mean

    #Betas
    beta = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train #OLS beta
    I = np.eye(p-1) #Identity matrix for ridge

    intercept = np.mean(y_train_mean - X_train_mean @ beta) #Manual intercept

    mse_train_list = []
    mse_test_list = []
    mse_train_ridge_list = []
    mse_test_ridge_list = []

    for lmbda in lmbda_list:
```

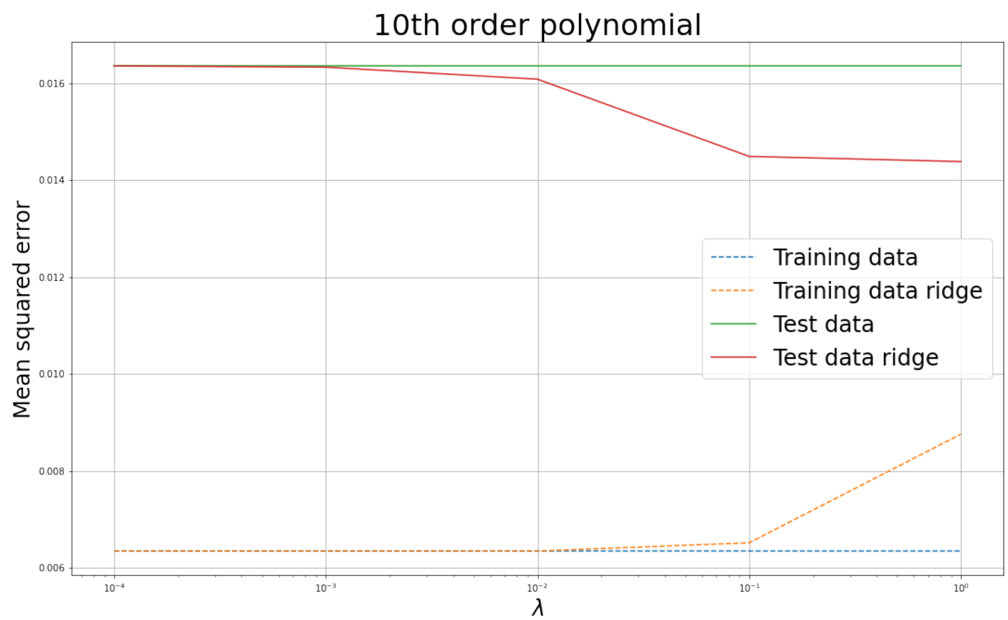
```
        #OLS (Not necessary to recalculate OLS everytime, but it's easy for␣
↪plotting)

        #Predictions are made with original X and y (non-scaled)
        y_tilde = (X_train + X_train_mean) @ beta + intercept
        y_predict = (X_test + X_train_mean) @ beta + intercept

        mse_train = mean_squared_error(y_train + y_train_mean, y_tilde)
        mse_test = mean_squared_error(y_test, y_predict)

        #Ridge
        beta_ridge = np.linalg.inv(X_train.T @ X_train + lmbda*I ) @ X_train.T @␣
↪y_train
        intercept_ridge = np.mean(y_train_mean - X_train_mean @ beta_ridge)

        #Predictions are made with original X and y (non-scaled)
        y_tilde_ridge = (X_train + X_train_mean) @ beta_ridge + intercept_ridge
        y_predict_ridge = (X_test + X_train_mean) @ beta_ridge + intercept_ridge

        mse_train_ridge = mean_squared_error(y_train + y_train_mean,␣
↪y_tilde_ridge)
        mse_test_ridge = mean_squared_error(y_test, y_predict_ridge)

        #Return lists
        mse_train_list.append(mse_train)
        mse_test_list.append(mse_test)
        mse_train_ridge_list.append(mse_train_ridge)
        mse_test_ridge_list.append(mse_test_ridge)


    return mse_train_list, mse_test_list, mse_train_ridge_list,␣
↪mse_test_ridge_list

mse_train_list = []
mse_test_list = []
mse_train_ridge_list = []
mse_test_ridge_list = []

lmbda_list = [0.0001, 0.001, 0.01, 0.1, 1.0] #Hyperparameter lambda

for deg in [6,11,16]:
    mse_train, mse_test, mse_train_ridge, mse_test_ridge =␣
↪poly_train_func(x,y,deg, lmbda_list)

    #Nested lists
    mse_train_list.append(mse_train)
    mse_test_list.append(mse_test)
```

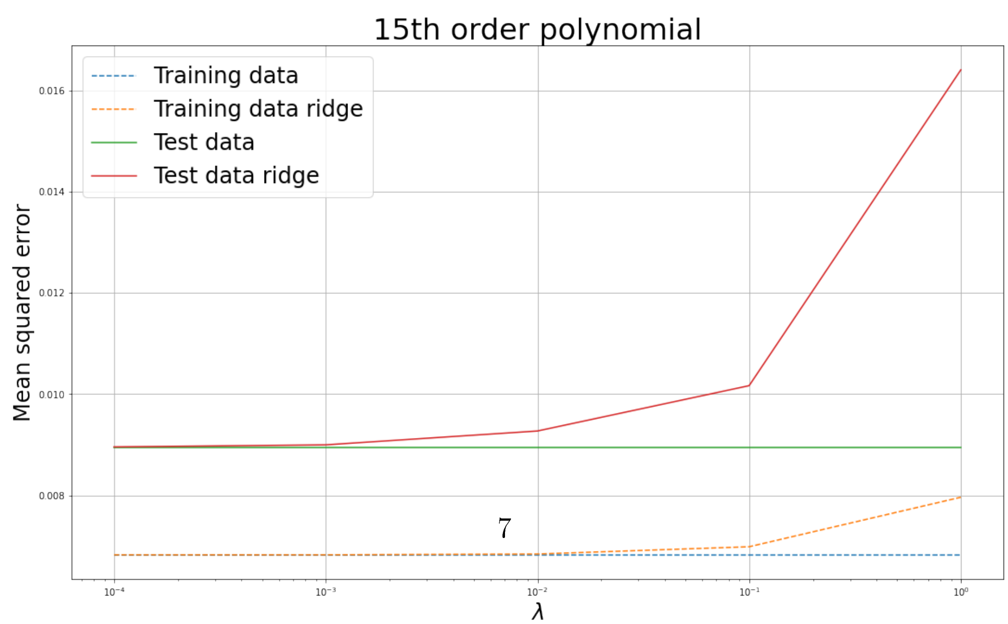
```
    mse_train_ridge_list.append(mse_train_ridge)
    mse_test_ridge_list.append(mse_test_ridge)
```

[70]:
```
#Plotting
fig = plt.figure(figsize=(18,36))

ax1 = fig.add_subplot(311)
ax1.semilogx(lmbda_list, mse_train_list[0], label="Training data",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_train_ridge_list[0], label="Training data ridge",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_test_list[0], label="Test data")
ax1.semilogx(lmbda_list, mse_test_ridge_list[0], label="Test data ridge")
ax1.set_xlabel(r"$\lambda$", fontsize=24)
ax1.set_ylabel("Mean squared error", fontsize=24)
ax1.legend(fontsize=24)
ax1.grid()
ax1.set_title("5th order polynomial", fontsize=32)

ax1 = fig.add_subplot(312)
ax1.semilogx(lmbda_list, mse_train_list[1], label="Training data",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_train_ridge_list[1], label="Training data ridge",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_test_list[1], label="Test data")
ax1.semilogx(lmbda_list, mse_test_ridge_list[1], label="Test data ridge")
ax1.set_xlabel(r"$\lambda$", fontsize=24)
ax1.set_ylabel("Mean squared error", fontsize=24)
ax1.legend(fontsize=24)
ax1.grid()
ax1.set_title("10th order polynomial", fontsize=32)

ax1 = fig.add_subplot(313)
ax1.semilogx(lmbda_list, mse_train_list[2], label="Training data",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_train_ridge_list[2], label="Training data ridge",␣
 ↪linestyle='dashed')
ax1.semilogx(lmbda_list, mse_test_list[2], label="Test data")
ax1.semilogx(lmbda_list, mse_test_ridge_list[2], label="Test data ridge")
ax1.set_xlabel(r"$\lambda$", fontsize=24)
ax1.set_ylabel("Mean squared error", fontsize=24)
ax1.legend(fontsize=24)
ax1.grid()
ax1.set_title("15th order polynomial", fontsize=32)
```

[70]:
```
Text(0.5, 1.0, '15th order polynomial')
```


5th order polynomial
Mean squared error
0.0222
0.0220
0.0218
0.0216
0.0214
0.0212
0.0210
Training data
Training data ridge
Test data
Test data ridge
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
λ


10th order polynomial
Mean squared error
0.016
0.014
0.012
0.010
0.008
0.006
Training data
Training data ridge
Test data
Test data ridge
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
λ


15th order polynomial
Mean squared error
0.016
0.014
0.012
0.010
0.008
Training data
Training data ridge
Test data
Test data ridge
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
λ

In my case, ridge regression generally does worse than Ordinary Least Squares, although the errors are still reasonably small. One could maybe expect the error to increase for ridge regression as you increase λ, at least for the training data, but maybe be a bit more robust on the test data. Although, in this case, it may seem like randomness is a bigger factor than any trends.