Weekly Exercises - FYS-STK3155

Oscar Atle Brovold

Week 36

Exercise 1 - Analytical exercises

a) Expression for Ridge regression

To show that the optimal parameters $\hat{\beta}$ for ridge regression is

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

we can minimize the following cost-function

$$C(\mathbf{X}, \boldsymbol{\beta}) = \{(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We can rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We have that

$$(\boldsymbol{y}^T\mathbf{X}\boldsymbol{\beta})^T = \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{y} \tag{1}$$

And since (1) is a scaler we have that the transpose of (1) is itself.
We can therefore rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We can now optimize $C(\mathbf{X}, \boldsymbol{\beta})$, we can do this by setting $\frac{\partial C(\mathbf{X},\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = 0$.

$$\begin{aligned}\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial\boldsymbol{\beta}} &= -2\frac{\partial}{\partial\boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{y} + \frac{\partial}{\partial\boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\frac{\partial}{\partial\boldsymbol{\beta}}\boldsymbol{\beta}^T\boldsymbol{\beta} \\ &= -2\boldsymbol{y}^T\mathbf{X} + 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X} + 2\lambda\boldsymbol{\beta}^T\end{aligned}$$

Setting this equal to zero yields

$$\boldsymbol{y}^T\mathbf{X} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X} - \lambda\boldsymbol{\beta}^T = 0$$

This rewrites to

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

b) The singular value decomposition

Ordinary least square (OLS) part

To show that we can write the OLS solutions in terms of the eigenvectors of the orthogonal matrix $\mathbf{U}$ as

$$\tilde{\boldsymbol{y}}_{OLS} = \mathbf{X}\boldsymbol{\beta}_{OLS} = \sum_{j=0}^{p-1} \boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{y}$$

We can start with the OLS equation and combine it with the singular value decomposition, meaning

$$\begin{aligned} \tilde{\boldsymbol{y}}_{OLS} = \mathbf{X}\boldsymbol{\beta} &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y} \\ \mathbf{X} &= \mathbf{U\Sigma V}^T \end{aligned}$$

Combining these yields

$$\begin{aligned} &= \mathbf{U\Sigma V}^T(\mathbf{V}\tilde{\mathbf{\Sigma}}^2\mathbf{V}^T)^{-1}\mathbf{V\Sigma}^T\mathbf{U}^T\boldsymbol{y} \\ &= \mathbf{U\Sigma V}^T\mathbf{V}(\tilde{\mathbf{\Sigma}}^2)^{-1}\mathbf{V}^T\mathbf{V\Sigma}^T\mathbf{U}^T\boldsymbol{y} \\ &= \mathbf{U\Sigma}(\tilde{\mathbf{\Sigma}}^2)^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\boldsymbol{y} \end{aligned}$$

The expression $\mathbf{\Sigma}(\tilde{\mathbf{\Sigma}}^2)^{-1}\mathbf{\Sigma}^T$ is a matrix with the same dimensions as $\mathbf{\Sigma}$. If $\mathbf{\Sigma}$ has p singular values then $\mathbf{\Sigma}(\tilde{\mathbf{\Sigma}}^2)^{-1}\mathbf{\Sigma}^T$ has p ones along the diagonal, and n-p zero columns. This gives

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

Ridge regression part

For Ridge regression we want to show that the corresponsing equation is

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

We start with the optimized cost-function for Ridge and combine it with SVD.

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

Combining these yields

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

$\boldsymbol{\Sigma}(\tilde{\boldsymbol{\Sigma}}^2 + \lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}$ is the same situation as for OLS, but instead of ones along the diagonal we get $\frac{\sigma_j^2}{\sigma_j^2+\lambda}$ along the diagonal. In total we therefor have

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

Interpreting the results

The $\frac{\sigma_j^2}{\sigma_j^2+\lambda}$ indicates that smaller singular values have a smaller influence on the predicted data compared to large singular values.

We can also see that if λ approaches zero the ridge results converges to the OLS betas. Likewise, as λ increases the penalization effect becomes stronger.

Exercise 2 - Adding Ridge Regression

See code in the exercise2.py file, to see how I calculate the outputs and plots in its fullness.

MSE for polynomial of degree 5

The datapoints are produced with the following equation

```
x = np.linspace(-3, 3, n).reshape(-1,1)
y = np.exp(-x**2) + 1.5 * np.exp(-(x-2)**2) + np.random.normal(0, 0.1, x.shape)
```

The splitting is done via this function, where we include the intercept

```
def scale_split(x,y,degree):
    poly = PolynomialFeatures(degree=degree)
    design_matrix = poly.fit_transform(x)
    scaler_X = StandardScaler(with_std=False)
    scaler_y = StandardScaler(with_std=False)
    design_matrix = scaler_X.fit_transform(design_matrix)
    scaled_y = scaler_y.fit_transform(y)
    design_matrix[:, 0] = 1
    X_train, X_test, y_train, y_test =\
    train_test_split(design_matrix, scaled_y, test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test
```

For calculating MSE for a polynomial of degree 5, the following function is used

```
def poly5():
    X_train, X_test, y_train, y_test = scale_split(x,y,5)
    lambdas = [10**i for i in range(-4, 1)]
    OLS(X_train, X_test, y_train, y_test)
    for l in lambdas:
        ridge(X_train, X_test, y_train, y_test, l, 5)
```

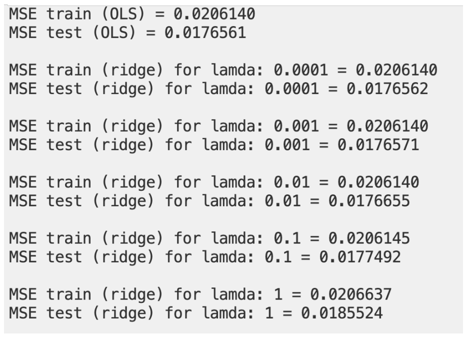
The poly5 calls again on the OLS and ridge function which calculates the related MSE

```
def OLS(X_train, X_test, y_train, y_test):
    beta_opt = np.linalg.inv(X_train.T@X_train)\
    @X_train.T@y_train
    MSE_train = MSE(y_train, X_train @ beta_opt)
    MSE_test = MSE(y_test, X_test @ beta_opt)
    print('MSE train (OLS) = {:.7f}'.format(MSE_train))
    print('MSE test (OLS) = {:.7f}'.format(MSE_test))

def ridge(X_train, X_test, y_train, y_test, l, degree):
    beta_opt = np.linalg.inv(X_train.T@X_train +\
    l * np.identity(degree + 1))@X_train.T@y_train
    MSE_train = MSE(y_train, X_train @ beta_opt)
    MSE_test = MSE(y_test, X_test @ beta_opt)
    print('\nMSE train (ridge) for lamda: {} =\
    {:.7f}'.format(l, MSE_train))
    print('MSE test (ridge) for lamda: {} =\
    {:.7f}'.format(l, MSE_test))
    return MSE_train, MSE_test

def MSE(y_data, y_predict):
    return mean_squared_error(y_data, y_predict)
```

This yields the following output



```
MSE train (OLS) = 0.0206140
MSE test (OLS) = 0.0176561

MSE train (ridge) for lamda: 0.0001 = 0.0206140
MSE test (ridge) for lamda: 0.0001 = 0.0176562

MSE train (ridge) for lamda: 0.001 = 0.0206140
MSE test (ridge) for lamda: 0.001 = 0.0176571

MSE train (ridge) for lamda: 0.01 = 0.0206140
MSE test (ridge) for lamda: 0.01 = 0.0176655

MSE train (ridge) for lamda: 0.1 = 0.0206145
MSE test (ridge) for lamda: 0.1 = 0.0177492

MSE train (ridge) for lamda: 1 = 0.0206637
MSE test (ridge) for lamda: 1 = 0.0185524
```

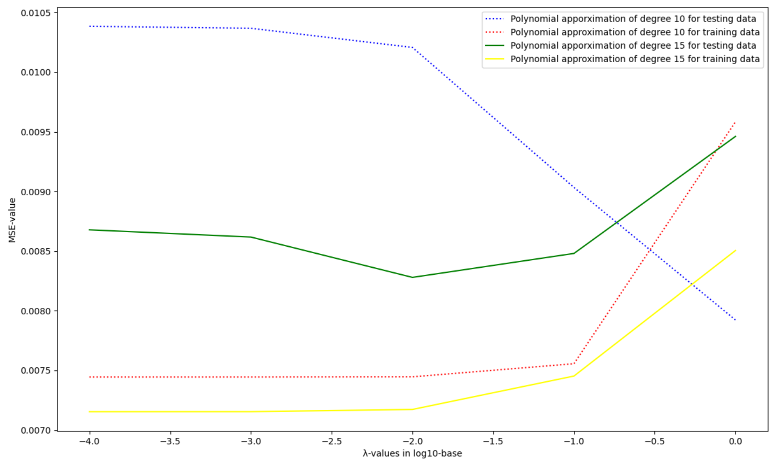
MSE for polynomial of degree 10 and 15

Now including a function that calculates MSE for degree 10 and 15 aswell.

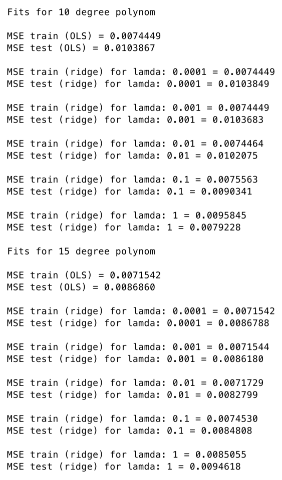
```
def polyN():
    N = [10, 15]
    lambdas = [10**i for i in range(-4, 1)]
    xplot = [10**i for i in range(-4, 1)]
    MSE_ridge_train_deg10, MSE_ridge_test_deg10 = [], []
    MSE_ridge_train_deg15, MSE_ridge_test_deg15 = [], []
    for n in N:
        print('\nFits␣for␣{}␣degree␣polynom\n'.format(n))
        X_train, X_test, y_train, y_test =\
        scale_split(x,y,n)
        OLS(X_train, X_test, y_train, y_test)
        for l in lambdas:
            if n == 10:
                MSE_train, MSE_test = ridge\
                (X_train, X_test, y_train, y_test, l, n)
                MSE_ridge_train_deg10.append(MSE_train)
                MSE_ridge_test_deg10.append(MSE_test)
            if n == 15:
                MSE_train, MSE_test = ridge\
                (X_train, X_test, y_train, y_test, l, n)
                MSE_ridge_train_deg15.append(MSE_train)
                MSE_ridge_test_deg15.append(MSE_test)

    plt.plot(np.log10(xplot), MSE_ridge_test_deg10,\
    linestyle=':', color='blue', label='Polynomial␣\
␣␣␣␣apporximation␣of␣degree␣10␣for␣testing␣data')
    plt.plot(np.log10(xplot), MSE_ridge_train_deg10,\
    linestyle=':', color='red', label='Polynomial␣\
␣␣␣␣approximation␣of␣degree␣10␣for␣training␣data')
    plt.plot(np.log10(xplot), MSE_ridge_test_deg15,\
    color='green', label='Polynomial␣\
␣␣␣␣apporximation␣of␣degree␣15␣for␣testing␣data')
    plt.plot(np.log10(xplot), MSE_ridge_train_deg15,\
    color='yellow', label='Polynomial␣approximation\
␣␣␣␣of␣degree␣15␣for␣training␣data')
    plt.xlabel('\lambda-values␣in␣log10-base')
    plt.ylabel('MSE-value')
    plt.legend()
    plt.show()
```

This gives the following plot:



and the following output:



```
Fits for 10 degree polynom

MSE train (OLS) = 0.0074449
MSE test (OLS) = 0.0103867

MSE train (ridge) for lamda: 0.0001 = 0.0074449
MSE test (ridge) for lamda: 0.0001 = 0.0103849

MSE train (ridge) for lamda: 0.001 = 0.0074449
MSE test (ridge) for lamda: 0.001 = 0.0103683

MSE train (ridge) for lamda: 0.01 = 0.0074464
MSE test (ridge) for lamda: 0.01 = 0.0102075

MSE train (ridge) for lamda: 0.1 = 0.0075563
MSE test (ridge) for lamda: 0.1 = 0.0090341

MSE train (ridge) for lamda: 1 = 0.0095845
MSE test (ridge) for lamda: 1 = 0.0079228

Fits for 15 degree polynom

MSE train (OLS) = 0.0071542
MSE test (OLS) = 0.0086860

MSE train (ridge) for lamda: 0.0001 = 0.0071542
MSE test (ridge) for lamda: 0.0001 = 0.0086788

MSE train (ridge) for lamda: 0.001 = 0.0071544
MSE test (ridge) for lamda: 0.001 = 0.0086180

MSE train (ridge) for lamda: 0.01 = 0.0071729
MSE test (ridge) for lamda: 0.01 = 0.0082799

MSE train (ridge) for lamda: 0.1 = 0.0074530
MSE test (ridge) for lamda: 0.1 = 0.0084808

MSE train (ridge) for lamda: 1 = 0.0085055
MSE test (ridge) for lamda: 1 = 0.0094618
```

Discussion of the results for the training MSE and test MSE with Ridge regression and ordinary least squares

The overall results suggest that both models, ridge and OLS, predict the data well. There is a trend that a higher polynomial degree gives a better fit without overfitting. Based on the plot a polynomial degree of 15 with a $\lambda = 0.01$ appears to give the best fit. The MSE from both the test and training set is low, and relatively close. It is also slightly better than the MSE test for the OLS of degree 15.

However, It is importent to note that there has only been used 100-datapoints, and the results are likely affected by randomness. More data would likely yield more accurate results.

— End of Weekly Exercise —