Solution for Sheet 08 - Exercise 4

Contributors:

- Annika Bätz: annika.baetz@uni-potsdam.de
- Max Serra: maximilia-manuel.serra.lasierra@uni-potsdam.de
- Adrián de Miguel: adrian.de.miguel.palacio@uni-potsdam.de
- Ignacio Llorca: ignacio.llorca.rodriguez@uni-potsdam.de

In [1]:
```
import numpy as np
import pandas as pd
from numpy.linalg import inv
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [2]:
```
X = np.array(pd.read_csv('X.txt', header=None))
y = np.array(pd.read_csv('Y.txt', header=None))

# add column of 1's to allow bias
X_extend = np.hstack((np.ones((len(X),1)),X))
```

Standard Regression without Regularization: $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Ridge Regression: $\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}^T\mathbf{y}$

λ is the regularization parameter. It is a hyperparameter that can be tuned.

In [3]:
```
# Does X have full rank?
X_extend.shape[1] == np.linalg.matrix_rank(X_extend)
```

Out[3]:
```
True
```

In [4]:
```
def StandardRegression(X, y):
    return inv(X.T@X)@X.T@y

def RidgeRegression(X, y, lbd):
    return inv(X.T@X + lbd)@X.T@y
```

In [5]:
```
def mse(X, y, beta):
    """ Computes the mean squared error """
    return np.mean((X@beta - y)**2)
```

In [6]:
```
def emp_bias(X, y, beta):
    """ Computes the empirical bias: Mean residual """
    return np.mean(X@beta - y)
```

In [7]:
```
def emp_variance(X, y, beta):
    """ Computes the empirical variance using the true values as the "reference" """
    return (1/(len(y)-1))*np.sum((X@beta - y)**2)
```

In [8]:
```
def sample_variance(X, y, beta):
```

```
    """ Computes the sample variance using the mean predicted value as the "referenc
    return (1/(len(y)-1))*np.sum((X@beta - np.mean(X@beta))**2)
```

In [9]:

```
beta_stand = StandardRegression(X_extend, y)
print('Beta for Standard Regression:')
print(beta_stand)
beta_ridge = RidgeRegression(X_extend, y, lbd=1)
print('\nBeta for Ridge Regression:')
print(beta_ridge)
```

```
Beta for Standard Regression:
[[-0.00800698]
 [ 0.88161162]
 [-2.45938171]
 [-0.97715699]]

Beta for Ridge Regression:
[[ 0.00589151]
 [ 0.90744364]
 [-2.43410595]
 [-0.97683612]]
```

In [13]:

```
print('Standard Regression:')
mse_stand = mse(X_extend, y, beta_stand)
print('MSE:', mse_stand)
emp_bias_stand = emp_bias(X_extend, y, beta_stand)
print('Empirical Bias:', emp_bias_stand)
emp_variance_stand = emp_variance(X_extend, y, beta_stand)
print('Empirical Variance:', emp_variance_stand)
sample_variance_stand = sample_variance(X_extend, y, beta_stand)
print('Sample Variance:', sample_variance_stand)
```

```
Standard Regression:
MSE: 0.9548405627555108
Empirical Bias: -9.94229574291185e-17
Empirical Variance: 0.9596147655692883
Sample Variance: 10.959436916685817
```

In [14]:

```
print('Ridge Regression:')
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression:
MSE: 0.955652311528916
Empirical Bias: 0.012425905038097639
Empirical Variance: 0.9604305730865607
Sample Variance: 10.89270682280516
```

The empirical variance is quite similar for the Standard Regression and the Ridge Regression.

The empirical bias of the Standard Regression Model is very close to 0. That is expected since we saw in the lecture that the average residual for the Least Squares Estimator without Regularizer is 0. That the computed empircial bias is not exactly 0 is probably due to internal rounding operations when the computer has to handel long decimal numbers.

The empircial bias of the Ridge Regression Model is approximately 0.012. This means that on average we got slightly higher predicted values than the true value is. Let's try out different regularization parameters and see how these results change:

In [17]:
```
print('Ridge Regression with lambda=0.1:')
beta_ridge = RidgeRegression(X_extend, y, lbd=0.01)
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression with lambda=0.1:
MSE: 0.9548406481876277
Empirical Bias: 0.00012747581415444
Empirical Variance: 0.959614851428566
Sample Variance: 10.958745633135583
```

In [18]:
```
print('Ridge Regression with lambda=0.1:')
beta_ridge = RidgeRegression(X_extend, y, lbd=0.1)
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression with lambda=0.1:
MSE: 0.9548490658972969
Empirical Bias: 0.0012717651507996972
Empirical Variance: 0.9596233112267833
Sample Variance: 10.952546538291537
```

In [19]:
```
print('Ridge Regression with lambda=1:')
beta_ridge = RidgeRegression(X_extend, y, lbd=1)
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression with lambda=1:
MSE: 0.955652311528916
Empirical Bias: 0.012425905038097639
Empirical Variance: 0.9604305730865607
Sample Variance: 10.89270682280516
```

In [20]:
```
print('Ridge Regression with lambda=5:')
beta_ridge = RidgeRegression(X_extend, y, lbd=5)
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
```

```
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression with lambda=5:
MSE: 0.9715527201013028
Empirical Bias: 0.05638109000121817
Empirical Variance: 0.9764104837018093
Sample Variance: 10.667260379162668
```

In [21]:

```
print('Ridge Regression with lambda=10:')
beta_ridge = RidgeRegression(X_extend, y, lbd=10)
mse_ridge = mse(X_extend, y, beta_ridge)
print('MSE:', mse_ridge)
emp_bias_ridge = emp_bias(X_extend, y, beta_ridge)
print('Empirical Bias:', emp_bias_ridge)
emp_variance_ridge = emp_variance(X_extend, y, beta_ridge)
print('Empirical Variance:', emp_variance_ridge)
sample_variance_ridge = sample_variance(X_extend, y, beta_ridge)
print('Sample Variance:', sample_variance_ridge)
```

```
Ridge Regression with lambda=10:
MSE: 1.008547854671894
Empirical Bias: 0.10107269293922573
Empirical Variance: 1.0135905939452534
Sample Variance: 10.454987741640672
```

If we increase the regularization parameter, three of our metrics (MSE, emp. bias and emp. variance) increase, whereas the sample variance decreases slightly. Most interesting is the increase in the empirical bias. This shows that using a regularizer is a trade-off between a in general more stable solution (when using regularization; not shown here) and a small empirical bias/ small mean residual (less regularization).

1.Little Experiment:

If we use Regularization we are introducing a bias to our model. However, we can expect our model building to be more stable. This means, if we are slightly changing our training data, the trained models should still be quite similar for the regularized approach, whereas we can expect the models without regularization to differ more. The latter approach is in general also more prone to overfitting to the data.

The goal is to find a good balance between bias and variance, i.e. to build a stable model that generalizes well, doesn't overfit, but still catches the main structure of the data. This is also known as the "Bias-Variance-Tradeoff".

In order to show this here, we have constructed a little experiment:

- First we split the data into two parts: "inner" and "test" data
- Then we repeat for 10 iterations:
  - Split the "inner" part again into "train" and "remain". The "remain" part is not needed anymore (for this iteration). We train a model on the "train" part and make a prediction on the "test" set.
  - We compute the empirical bias for this model
- Now we have several lists of predictions on the test data. For each true label in the test data, we can compute the variance of all the different predicted values:

- Let n be the number of trained models
- Let y_i be the true label for the i-th instance in the test data.
- Let $\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{in}$ be the different predictions for y_i
- Let $\bar{y}_i$ be the mean value of $\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{in}$
- We define the variance for the predictions for one specific instance as: $var_i = \frac{1}{n-1} \sum_{k=1}^{n} (\hat{y}_{ik} - \bar{y}_i)^2$
- As a final step we can compute the mean value of var_i for all instances i in the test data.

The main difference to the first part of this notebook is that we now use different data to train and test our models and that we repeat the whole procedure several times to get a more general solution.

In [22]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
def bias_variance_estimate(X, y, rounds=10, model='simple', lbd=0.1):
    X_inner, X_test, y_inner, y_test = train_test_split(X, y, test_size=0.3, random_

    pred_list = []
    bias_list = []
    for i in range(rounds):
        # divide the data into training and test data
        X_train, X_remain, y_train, y_remain = train_test_split(X_inner, y_inner, te

        # compute beta:
        if model=='simple':
            beta = StandardRegression(X_train, y_train)
        elif model=='ridge':
            beta = RidgeRegression(X_train, y_train, lbd)

        # compute the empirical bias
        bias_list.append(emp_bias(X_test, y_test, beta))

        # make prediction on the "outer" data
        pred = X_test@beta
        pred_list.append(pred)

    # compute the mean variance between all predictions
    pred = np.array(pred_list)
    n = rounds
    variance = np.mean(1/(n-1) * np.sum((pred - np.mean(pred, axis=0))**2, axis=0))
    return variance, bias_list
```

In [24]:
```
print('Normal Regression:')
variance_stand, bias_list_stand = bias_variance_estimate(X_extend, y, rounds=10, mod
print('Variance among the model predictions: ', np.round(variance_stand,4))
print('Mean empirical bias: ', np.round(np.mean(bias_list_stand), 4))
```

```
Normal Regression:
Variance among the model predictions:  0.0097
Mean empirical bias:  0.0082
```

In [25]:
```
print('Ridge Regression (lambda = 0.1):')
variance, bias_list = bias_variance_estimate(X_extend, y, rounds=10, model='ridge',
print('Variance among the model predictions: ', np.round(variance,4))
print('Mean empirical bias: ', np.round(np.mean(bias_list), 4))
```

```
Ridge Regression (lambda = 0.1):
Variance among the model predictions:  0.0097
Mean empirical bias:  0.0096
```

In [26]:
```
print('Ridge Regression (lambda = 1):')
variance, bias_list = bias_variance_estimate(X_extend, y, rounds=10, model='ridge',
print('Variance among the model predictions: ', np.round(variance,4))
print('Mean empirical bias: ', np.round(np.mean(bias_list), 4))
```

```
Ridge Regression (lambda = 1):
Variance among the model predictions:  0.0094
Mean empirical bias:  0.0216
```

In [27]:
```
print('Ridge Regression (lambda = 10):')
variance, bias_list = bias_variance_estimate(X_extend, y, rounds=10, model='ridge',
print('Variance among the model predictions: ', np.round(variance,4))
print('Mean empirical bias: ', np.round(np.mean(bias_list), 4))
```

```
Ridge Regression (lambda = 10):
Variance among the model predictions:  0.0081
Mean empirical bias:  0.1012
```

In [28]:
```
print('Ridge Regression (lambda = 50):')
variance, bias_list = bias_variance_estimate(X_extend, y, rounds=10, model='ridge',
print('Variance among the model predictions: ', np.round(variance,4))
print('Mean empirical bias: ', np.round(np.mean(bias_list), 4))
```

```
Ridge Regression (lambda = 50):
Variance among the model predictions:  0.0083
Mean empirical bias:  0.2056
```

The results fit our expectations quite well:

By introducing regularization, we introduce a bias, the computed mean empirical bias increases. If we regularize stronger (increase the regularization parameter lambda), the bias increases even more.

On the other hand, the variance among the different model predictions decreases when we use regularization. This means the model building is less effected by the difference in training data.

In our specific case, we can see that the variance decreases only to a certain point. If the hyperparameter is quite high (lambda = 50) the variance starts to increase again.

In [ ]:
```
```