4^{th} Homework

Chalkiopoulos Georgios | $p3352124$

February 10, 2022

Exercise 1.

(a) Use the Lagrangian function

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)^2 + \lambda\|\boldsymbol{\theta}\|^2$$

of the ridge regression (RR) problem and show that the RR solution satisfies the following equation:

$$\left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^T + \lambda I\right)\hat{\boldsymbol{\theta}} = \sum_{n=1}^{N} y_n\boldsymbol{x}_n$$

We first take the gradient of $L(\theta)$:

$$\begin{aligned}
L(\boldsymbol{\theta}) &= \sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)^2 + \lambda\|\boldsymbol{\theta}\|^2 \Rightarrow \\
\frac{\partial L(\boldsymbol{\theta})}{\partial\theta} &= \frac{\partial\left(\sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)^2 + \lambda\|\boldsymbol{\theta}\|^2\right)}{\partial\theta} \\
&= \frac{\partial\left(\sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)^2\right)}{\partial\theta} + \lambda\frac{\partial\left(\sum_{n=1}^{N}\|\boldsymbol{\theta}\|^2\right)}{\partial\theta} \\
&= -2\sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)\boldsymbol{x}_n + \lambda\frac{\partial\left(\sum_{n=1}^{N}\boldsymbol{\theta}^T\boldsymbol{\theta}\right)}{\partial\theta} \\
&= -2\sum_{n=1}^{N} y_n\boldsymbol{x}_n - (\boldsymbol{\theta}^T\boldsymbol{x}_n)\boldsymbol{x}_n - \lambda\boldsymbol{\theta} \\
&= -2\sum_{n=1}^{N} y_n\boldsymbol{x}_n - (\boldsymbol{\theta}^T\boldsymbol{x}_n)\boldsymbol{x}_n - \lambda\boldsymbol{\theta}
\end{aligned}$$

We then equate to 0 and solve:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta} = 0 \Rightarrow -2\sum_{n=1}^{N} y_n \boldsymbol{x}_n - (\boldsymbol{\theta}^T \boldsymbol{x}_n)\boldsymbol{x}_n - \lambda \boldsymbol{\theta} = 0 \Rightarrow$$

$$\sum_{n=1}^{N} y_n \boldsymbol{x}_n = \sum_{n=1}^{N} (\boldsymbol{\theta}^T \boldsymbol{x}_n)\boldsymbol{x}_n + \lambda \boldsymbol{\theta}$$

$$= \sum_{n=1}^{N} (\boldsymbol{x}_n \boldsymbol{x}_n^T)\boldsymbol{\theta} + \lambda I \boldsymbol{\theta} \Rightarrow$$

$$\sum_{n=1}^{N} y_n \boldsymbol{x}_n = \left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda I \right) \hat{\boldsymbol{\theta}} \tag{1.1}$$

(b) Prove that the above solution (1.1) can be expressed in matrix form as:

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

We have:

$$X^T X = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{21} & \dots & x_{N1} \\ x_{12} & x_{22} & \dots & x_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1l} & x_{2l} & \dots & x_{Nl} \end{bmatrix} \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1l} \\ 1 & x_{21} & x_{22} & \dots & x_{2l} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nl} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_N}^T \end{bmatrix}$$

$$= \sum_{n=1}^{N} \boldsymbol{x_n} \boldsymbol{x_n}^T \tag{1.2}$$

Similarly:

$$
\begin{aligned}
X^T\boldsymbol{y} &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{21} & \dots & x_{N1} \\ x_{12} & x_{22} & \dots & x_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1l} & x_{2l} & \dots & x_{Nl} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \\
&= \sum_{n=1}^{N} \boldsymbol{x_n} y_n = \sum_{n=1}^{N} y_n \boldsymbol{x_n} \qquad (1.3)
\end{aligned}
$$

Using (1.2) and (1.3) we can write (1.1) as:

$$
\begin{aligned}
\left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda I \right) \hat{\boldsymbol{\theta}} = \sum_{n=1}^{N} y_n \boldsymbol{x}_n \Rightarrow \\
(X^T X + \lambda I)\hat{\boldsymbol{\theta}} = X^T \boldsymbol{y} \Rightarrow \\
\hat{\boldsymbol{\theta}} = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}
\end{aligned}
$$

Exercise 2.

We define the following:
θ_0: True parameter
$\hat{\theta}_{MVU}$: minimum unbiased estimator of θ_0
F: Parametric set of all estimators:

$$\hat{\theta}_b = (1+\alpha)\hat{\theta}_{MVU}, \alpha \in R \tag{2.1}$$

Moreover:

$$MSE(\hat{\theta}) = E\left[(\hat{\theta} - E(\hat{\theta}))^2\right] + (E(\hat{\theta}) - \theta_0)^2 \tag{2.2}$$

(a) What can you infer from the fact that $\hat{\theta}_{MVU}$ is an unbiased estimator of θ_0?

If $\hat{\theta}_{MVU}$ is an unbiased estimator of θ_0 we can infer that:

$$E\left(\hat{\theta}_{MVU}\right) = \theta_0 \tag{2.3}$$

which diminishes the Bias term in (2.1) and results to the minimum variance.

(b) Prove that all $\hat{\theta}_b$'s of F, for $\alpha \neq 0$, are biased estimators of θ_0.

Suppose $\hat{\theta}_b$ is an unbiased estimator of θ_0. That should mean that:

$$\begin{aligned} E\left(\hat{\theta}_b\right) = \theta_0 \Rightarrow \\ E\left[(1+\alpha)\hat{\theta}_{MVU}\right] = \theta_0 \Rightarrow \\ E\left[(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU})\right] = \theta_0 \Rightarrow \\ E\left[\hat{\theta}_{MVU}\right] + \alpha E\left[\hat{\theta}_{MVU}\right] = \theta_0 \overset{(2.3)}{\Rightarrow} \\ (1+\alpha)\theta_0 = \theta_0 \end{aligned} \tag{2.3}$$

Given that $a \neq 0$ it's obvious that $\hat{\theta}_b$ cannot be an unbiased estimator of θ_0.

(c) Find the $MSE(\hat{\theta}_{MVU})$ using eq. (2.1). Explain why this value cannot be zero for finite N.

Suppose $\hat{\theta}_b$ is an unbiased estimator of θ_0. That should mean that:

$$\begin{aligned} MSE(\hat{\theta}_{MVU}) &= E\left[(\hat{\theta}_{MVU} - E(\hat{\theta}_{MVU}))^2\right] + (E(\hat{\theta}_{MVU}) - \theta_0)^2 \\ &= E\left[(\hat{\theta}_{MVU} - \theta_0)^2\right] + (\theta_0 - \theta_0)^2 \\ &= E\left[(\hat{\theta}_{MVU} - \theta_0)^2\right] \end{aligned}$$

The above equation suggests that in order, for this value, to be zero, the estimator would be exactly the same as the True Parameter, but we know that it is not possible to predict this, given that we don't have infinite data. As a matter of fact, the reason machine learning exists is this one (we cannot approximate the noise).

(d) Express the $MSE(\hat{\theta})$ in terms of $MSE(\hat{\theta}_{MVU})$.

$$\begin{aligned} MSE(\hat{\theta}_b) &= E\left[(\hat{\theta}_b - E(\hat{\theta}_b))^2\right] + (E(\hat{\theta}_b) - \theta_0)^2 \\ &= E\left[((1+\alpha)\hat{\theta}_{MVU} - E((1+\alpha)\hat{\theta}_{MVU}))^2\right] \\ &\quad + (E((1+\alpha)\hat{\theta}_{MVU}) - \theta_0)^2 \\ &= E\left[(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU} - E(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU}))^2\right] \\ &\quad + (E(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU}) - \theta_0)^2 \\ &= E\left[(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU} - E(\hat{\theta}_{MVU}) - E(\alpha\hat{\theta}_{MVU}))^2\right] \\ &\quad + (E(\hat{\theta}_{MVU}) + E(\alpha\hat{\theta}_{MVU}) - \theta_0)^2 \\ &= E\left[(\hat{\theta}_{MVU} + \alpha\hat{\theta}_{MVU} - \theta_0 - \alpha\theta_0))^2\right] \\ &\quad + (\theta_0 + \alpha\theta_0 - \theta_0)^2 \\ &= E\left[((1+\alpha)\hat{\theta}_{MVU} - (1+\alpha)\theta_0)^2\right] \\ &\quad + (\alpha\theta_0)^2 \\ &= E\left[((1+\alpha)\hat{\theta}_{MVU} - (1+\alpha)\theta_0)^2\right] \\ &\quad + (\alpha\theta_0)^2 \end{aligned}$$

$$
\begin{aligned}
&= E\left[((1+\alpha)(\hat{\theta}_{MVU}-\theta_0))^2\right] + (\alpha\theta_0)^2 \\
&= (1+\alpha)^2 E\left[(\hat{\theta}_{MVU}-\theta_0)^2\right] + (\alpha\theta_0)^2 \\
&= (1+\alpha)^2 MSE(\hat{\theta}_{MVU}) + (\alpha\theta_0)^2
\end{aligned}
$$

(e) Determine the range of values of the parameter α that result to estimators with MSE lower than $MSE(\hat{\theta}_{MVU})$.

It is:

$$
\begin{aligned}
&MSE(\hat{\theta}_b) < MSE(\hat{\theta}_{MVU}) && \Rightarrow \\
&(1+\alpha)^2 MSE(\hat{\theta}_{MVU}) + (\alpha\theta_0)^2 < MSE(\hat{\theta}_{MVU}) && \Rightarrow \\
&MSE(\hat{\theta}_{MVU}) + 2MSE(\hat{\theta}_{MVU})\alpha + MSE(\hat{\theta}_{MVU})\alpha^2 + \alpha^2\theta_0^2 < MSE(\hat{\theta}_{MVU}) && \Rightarrow \\
&(MSE(\hat{\theta}_{MVU}) + \theta_0^2)\cdot\alpha^2 + 2MSE(\hat{\theta}_{MVU})\cdot\alpha < 0 && (2.4)
\end{aligned}
$$

We have a 2nd degree polynomial of α of which we will find the roots:

$$
\begin{aligned}
\alpha_{1,2} &= \frac{-2MSE(\hat{\theta}_{MVU}) \pm \sqrt{(2MSE(\hat{\theta}_{MVU}))^2}}{2\cdot(MSE(\hat{\theta}_{MVU}) + \theta_0^2)} \\
&= \frac{-2MSE(\hat{\theta}_{MVU}) \pm 2MSE(\hat{\theta}_{MVU})}{2MSE(\hat{\theta}_{MVU}) + 2\theta_0^2} \\
&= \begin{cases} 0 \\ -\dfrac{2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU}) + \theta_0^2} \end{cases}
\end{aligned}
$$

Since both $MSE(\hat{\theta}_{MVU})$ and θ_0^2 are positive values we conclude that the second root is a negative number, thus (2.4) is negative when:

$$
-\frac{2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU}) + \theta_0^2} < \alpha < 0 \qquad (2.5)
$$

The above equation (2.5) can also be written as $c(\theta_0) < a < 0$.

We will now try to find the bounds of $-2\frac{MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}$. It is:

$$-2\frac{MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2} = -2\frac{1}{1+\frac{\theta_0^2}{MSE(\hat{\theta}_{MVU})}} \tag{2.6}$$

The fraction in (2.6) is a $1/(1+x)$ function with $x >= 0$ and we know that the values of the function are bound between (0, 1]. Thus $c(\theta_0)$ takes values in [-2, 0) which, using (2.5), implies that:

$$-2 < \alpha < 0 \tag{2.7}$$

(f) Prove that for any value of α in the range defined in (e), it is $|\hat{\theta}_b| < |\hat{\theta}_{MVU}|$

Given that: $(-2 <)c(\theta_0) < a < 0$ it is:

$$\begin{aligned} -2 < a < 0 &\Leftrightarrow \\ -1 < 1+a < 1 &\Leftrightarrow \\ |1+a| < 1 &\Rightarrow \\ |1+a| \cdot |\hat{\theta}_{MVU}| < |\hat{\theta}_{MVU}| &\Leftrightarrow \\ |\hat{\theta}_b| < |\hat{\theta}_{MVU}| & \end{aligned}$$

(g) Determine the value α^* of the parameter α that corresponds to the estimator giving the lowest MSE.

1. Using $MSE(\hat{\theta}_b) = (1+\alpha)^2 MSE(\hat{\theta}_{MVU}) + (\alpha\theta_0)^2$ it is:

$$\begin{aligned} &\frac{\partial\left((1+\alpha)^2 MSE(\hat{\theta}_{MVU}) + (\alpha\theta_0)^2\right)}{\partial\alpha} = 0 && \Rightarrow \\ &2(1+\alpha)MSE(\hat{\theta}_{MVU}) + 2\alpha\theta_0^2 = 0 && \Rightarrow \\ &2MSE(\hat{\theta}_{MVU}) + 2\alpha MSE(\hat{\theta}_{MVU}) + 2\alpha\theta_0^2 = 0 && \Rightarrow \\ &(MSE(\hat{\theta}_{MVU}) + \theta_0^2)\alpha = -MSE(\hat{\theta}_{MVU}) && \Rightarrow \\ &\alpha = -\frac{MSE(\hat{\theta}_{MVU})}{(MSE(\hat{\theta}_{MVU}) + \theta_0^2)} \end{aligned}$$

(h) Explain why in practice α^* cannot be determined.

α^* cannot be determined since it contains θ_0 which is the True Parameter and is unknown (we are trying to estimate it's value).

Exercise 3.

(a) Derive the LS estimator of θ_0 from:

$$\left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T\right) \boldsymbol{\theta} = \sum_{n=1}^{N} y_n \boldsymbol{x}_n \tag{3.1}$$

We know that $\boldsymbol{x}_n$ is a scalar and $\boldsymbol{x}_n = 1$ for all n. Therefore (3.1) can be written as:

$$\begin{aligned}
\left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T\right) \boldsymbol{\theta} = \sum_{n=1}^{N} y_n \boldsymbol{x}_n \Rightarrow \\
\left(\sum_{n=1}^{N} 1 \cdot 1\right) \theta = \sum_{n=1}^{N} y_n \cdot 1 \Rightarrow \\
N \cdot \theta = \sum_{n=1}^{N} y_n \Rightarrow \\
\theta = \frac{1}{N} \sum_{n=1}^{N} y_n
\end{aligned} \tag{3.2}$$

(b) Prove that y_n is an unbiased estimator of θ_0.

It is:

$$\begin{aligned}
E[y_n] &= E[\theta_0 + \eta_n] \\
&= E[\theta_0] + E[\eta_n] \overset{\eta_n \sim \mathcal{N}(0,\sigma^2)}{\Rightarrow} \\
E[y_n] &= \theta_0
\end{aligned}$$

(c) Prove that $\bar{y}$ is an unbiased estimator of θ_0.

$\bar{y}$ is the mean of N statistically independent rv's denoted by:

$$\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n \tag{3.3}$$

Moreover, knowing that the LS estimator is an unbiased estimator of θ_0, we can see that (3.2) and (3.3) are equal, thus $\bar{y}$ is an unbiased estimator of θ_0. Alternatively:

$$\begin{aligned} E[\bar{y}] &= E\left[\frac{1}{N}\sum_{n=1}^{N} y_n\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} E[\theta_0] + E[\eta_n] \\ &= \frac{1}{N}\sum_{n=1}^{N} \theta_0 \\ &= \frac{1}{N} N\theta_0 \Rightarrow \\ E[\bar{y}] &= \theta_0 \end{aligned}$$

(d) We denote: $\bar{y} = \hat{\theta}_{MVU}$.

(e) Prove that the ridge regression estimator for the present 1-dim. case is expressed as:

$$\hat{\theta} = \frac{1}{N+\lambda}\sum_{n=1}^{N} y_n$$

Using the RR solution, from (1.1) we can adjust the solution for the 1-dim case. Since all values are scalars, the identity matrix in the equation will be a scalar as well (I_1). It is:

$$\begin{aligned} \left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda I_1\right)\hat{\boldsymbol{\theta}} &= \sum_{n=1}^{N} y_n \boldsymbol{x}_n \Rightarrow \\ (N+\lambda)\,\hat{\theta} &= \sum_{n=1}^{N} y_n \Rightarrow \\ \hat{\theta} &= \frac{1}{N+\lambda}\sum_{n=1}^{N} y_n \end{aligned} \tag{3.4}$$

(f) Denote $\hat{\theta}$ in terms of $\hat{\theta}_{MVU}$.

It is:

$$\begin{aligned}\hat{\theta} &= \frac{1}{N+\lambda}\sum_{n=1}^{N} y_n \\ &= \frac{N}{N+\lambda}\frac{1}{N}\sum_{n=1}^{N} y_n \\ &= \frac{N}{N+\lambda}\hat{\theta}_{MVU}\end{aligned}$$

(g) Prove that $\hat{\theta}$ is a biased estimator.

It is:

$$\begin{aligned}E[\hat{\theta}] &= E\left[\frac{N}{N+\lambda}\hat{\theta}_{MVU}\right] \\ &= \frac{N}{N+\lambda}E\left[\hat{\theta}_{MVU}\right] \\ &\overset{\lambda\geq 0}{=} \frac{N}{N+\lambda}\theta_0 \neq \theta_0\end{aligned}$$

(h) Verify that $|\hat{\theta}| < |\hat{\theta}_{MVU}|$.

It is:

$$\begin{aligned}|\hat{\theta}| &= \left|\frac{N}{N+\lambda}\hat{\theta}_{MVU}\right| \\ &= \left|\frac{N}{N+\lambda}\right|\left|\hat{\theta}_{MVU}\right| \overset{\lambda\geq 0}{\Rightarrow} \\ |\hat{\theta}| &< \left|\hat{\theta}_{MVU}\right|\end{aligned}$$

(i) (optional) Given that $\hat{\theta}_b = (1+\alpha)\hat{\theta}_{MVU}, \alpha \in R$ express the quantity in terms of λ and derive the range of values of λ for which the MSE of the ridge regression estimator is less than that of the least squares estimator.

It is:

$$\begin{aligned} 1+\alpha &= \frac{N}{N+\lambda} \Rightarrow \\ \alpha &= \frac{N}{N+\lambda} - 1 \Rightarrow \\ \alpha &= \frac{N-N-\lambda}{N+\lambda} \Rightarrow \\ \alpha &= -\frac{\lambda}{N+\lambda} \end{aligned}$$

Using (2.5) we can define the fraction $\frac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}$ as $f(MSE, \theta_0)$. Therefore:

$$\begin{aligned} &\frac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2} < \alpha < 0 && \Rightarrow \\ &f(MSE,\theta_0) < \alpha < 0 && \Rightarrow \\ &f(MSE,\theta_0) < -\frac{\lambda}{N+\lambda} < 0 && \overset{N+\lambda>0}{\Rightarrow} \\ &f(MSE,\theta_0)(N+\lambda) < -\lambda < 0 && \Rightarrow \\ &f(MSE,\theta_0)N + \lambda f(MSE,\theta_0) < -\lambda < 0 && \Rightarrow \\ &f(MSE,\theta_0)N < -\lambda - \lambda f(MSE,\theta_0) < -\lambda f(MSE,\theta_0) && \Rightarrow \\ &-f(MSE,\theta_0)N > \lambda(f(MSE,\theta_0)+1) > \lambda f(MSE,\theta_0) && (3.5) \end{aligned}$$

Moreover $-2 < f(MSE, \theta_0) < 0$ thus $-1 < f(MSE, \theta_0) + 1 < 1$, and using (3.5) we have:

$$\begin{cases} -\frac{f(MSE,\theta_0)N}{f(MSE,\theta_0)+1} < \lambda, & f(MSE,\theta_0) < -1 \\ -\frac{f(MSE,\theta_0)N}{f(MSE,\theta_0)+1} > \lambda, & f(MSE,\theta_0) > -1 \end{cases}$$

$$\begin{cases} -\dfrac{\dfrac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}N}{\dfrac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}+1} < \lambda, & f(MSE,\theta_0) < -1 \\ -\dfrac{\dfrac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}N}{\dfrac{-2MSE(\hat{\theta}_{MVU})}{MSE(\hat{\theta}_{MVU})+\theta_0^2}+1} > \lambda, & f(MSE,\theta_0) > -1 \end{cases}$$

$$\begin{cases} \dfrac{2MSE(\hat{\theta}_{MVU})N}{\theta_0^2 - MSE(\hat{\theta}_{MVU})} < \lambda, & f(MSE,\theta_0) < -1 \\ \dfrac{2MSE(\hat{\theta}_{MVU})N}{\theta_0^2 - MSE(\hat{\theta}_{MVU})} > \lambda, & f(MSE,\theta_0) > -1 \end{cases}$$

Exercise 4

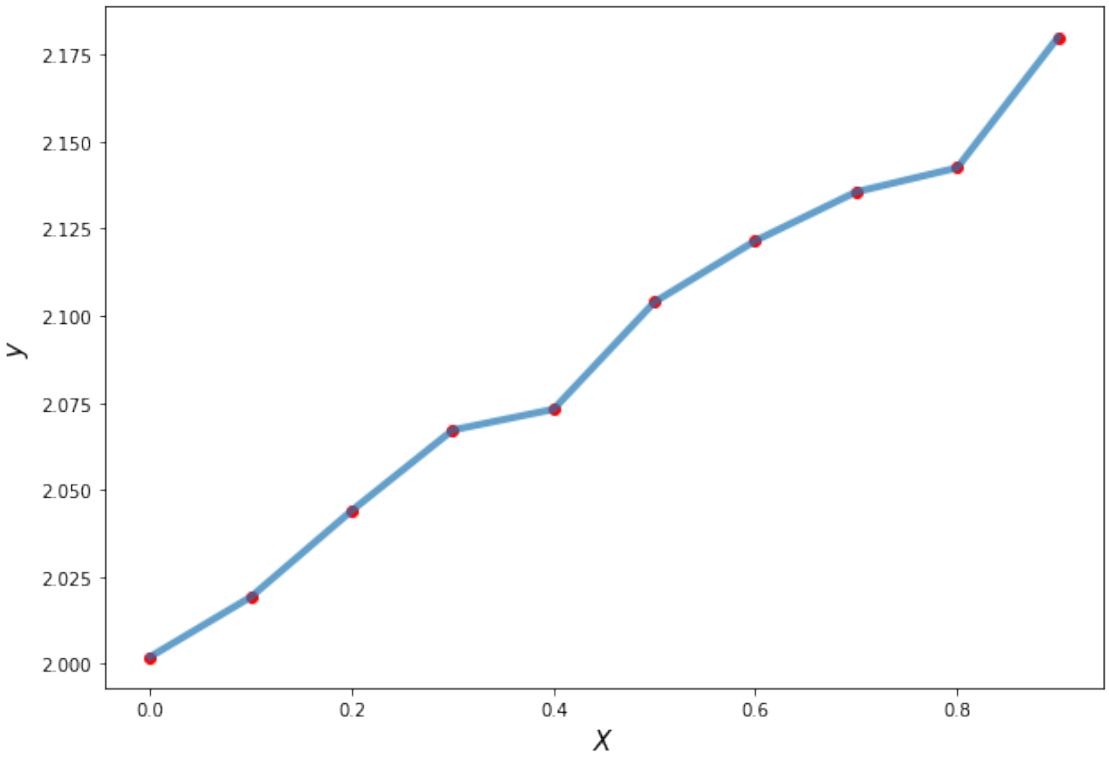
4.a Plot the data

[24]:
```
import numpy as np
import scipy.io as sio
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.cm as cm

import pandas as pd

from sklearn import linear_model

# for creating a responsive plot
%matplotlib inline
```

[25]:
```
# Load the data
Training_Set = sio.loadmat('Training_Set.mat')
X = Training_Set['X']
y = Training_Set['y']

# See the results
np.concatenate((X,y), axis=1)[:5]
```

[25]:
```
array([[0.        , 2.00195711],
       [0.1       , 2.01918334],
       [0.2       , 2.04400279],
       [0.3       , 2.06714191],
       [0.4       , 2.07310931]])
```

[26]:
```
# plot the data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.scatter(X,y,c='red',marker='o')
ax.plot(X,y,  linewidth=4, alpha=0.7)
ax.set_xlabel('$X$', fontsize=15)
_ = ax.set_ylabel('$y$', fontsize=15)
```



[27]:
```
# function to return 8th degree polynomial of given input
def calculate_polyn(x):
    x_polyn = x
    for i in range(2,9):
        x_polyn = np.append(x_polyn, np.power(x,i))
    return x_polyn.reshape(1,-1)

# create the X matrix
X_p = np.array(calculate_polyn(X[0]))
for i in X[1:,0:]:
    X_p = np.concatenate((X_p, np.array(calculate_polyn(i))))
X_p =  np.concatenate((np.ones((10, 1)), X_p), axis=1)

print("8th degree polynomial X matrix:\n")
print(X_p[:2])
```

```
8th degree polynomial X matrix:

[[1.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00]
 [1.e+00 1.e-01 1.e-02 1.e-03 1.e-04 1.e-05 1.e-06 1.e-07 1.e-08]]
```

4.b

Fit a 8th degree polynomial on the data using the LS estimator and plot the results (data points and the curve resulting from the fit). Output also the estimates of the parameters of the polynomial.

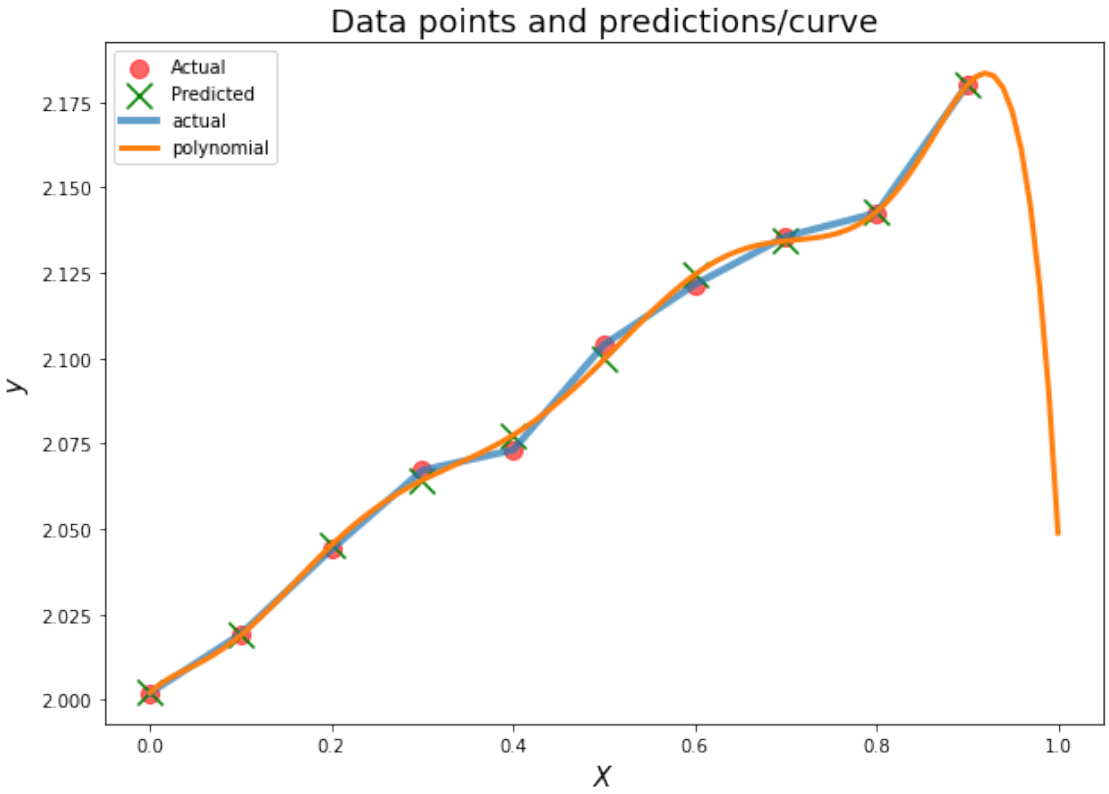
[28]:
```
# Calculate \theta LS
Xx_inv = np.linalg.inv(X_p.T.dot(X_p))

Xy = X_p.T.dot(y)

# Finally
theta_ls = Xx_inv.dot(Xy)

# data points:
y_hat = theta_ls.T.dot(X_p.T)
```

[29]:
```
# Source https://stackoverflow.com/questions/37352098/
 ↪plotting-a-polynomial-using-matplotlib-and-coeffiecients
def polynomial_coefficients(xs, coeffs):
    """ Returns a list of function outputs (`ys`) for a polynomial with the␣
 ↪given coefficients and
    a list of input values (`xs`).

    The coefficients must go in order from a0 to an, and all must be included,␣
 ↪even if the value is 0.
    """
    order = len(coeffs)

    ys = np.zeros(len(xs))  # Initialise an array of zeros of the required␣
 ↪length.
    for i in range(order):
        ys += coeffs[i] * xs ** i
    return ys
```

[30]:
```
# plot the data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)

# Plot points
ax.scatter(X,y,c='red',marker='o', label = 'Actual', s=100, alpha=0.6)
ax.scatter(X,y_hat,c='green',marker='x', label = 'Predicted', s=200)

# Plot actual line
ax.plot(X,y,  linewidth=4, alpha=0.7, label='actual')

# plot polynomial
x_points = np.linspace(0, 1, 100)
```

```
ax.plot(x_points, polynomial_coefficients(x_points, theta_ls), linewidth=3,␣
 ↪label='polynomial')

ax.set_xlabel('$X$', fontsize=15)
ax.set_ylabel('$y$', fontsize=15)

ax.set_title('Data points and predictions/curve', fontsize=18)

_ = ax.legend()

print('The estimates are:')
print(theta_ls.flatten())
```

```
The estimates are:
[ 2.00200245e+00  2.59357661e-01 -3.83236599e+00  4.94205475e+01
 -2.62207916e+02  6.98307312e+02 -9.86594788e+02  7.06146301e+02
 -2.01451599e+02]
```



4.c

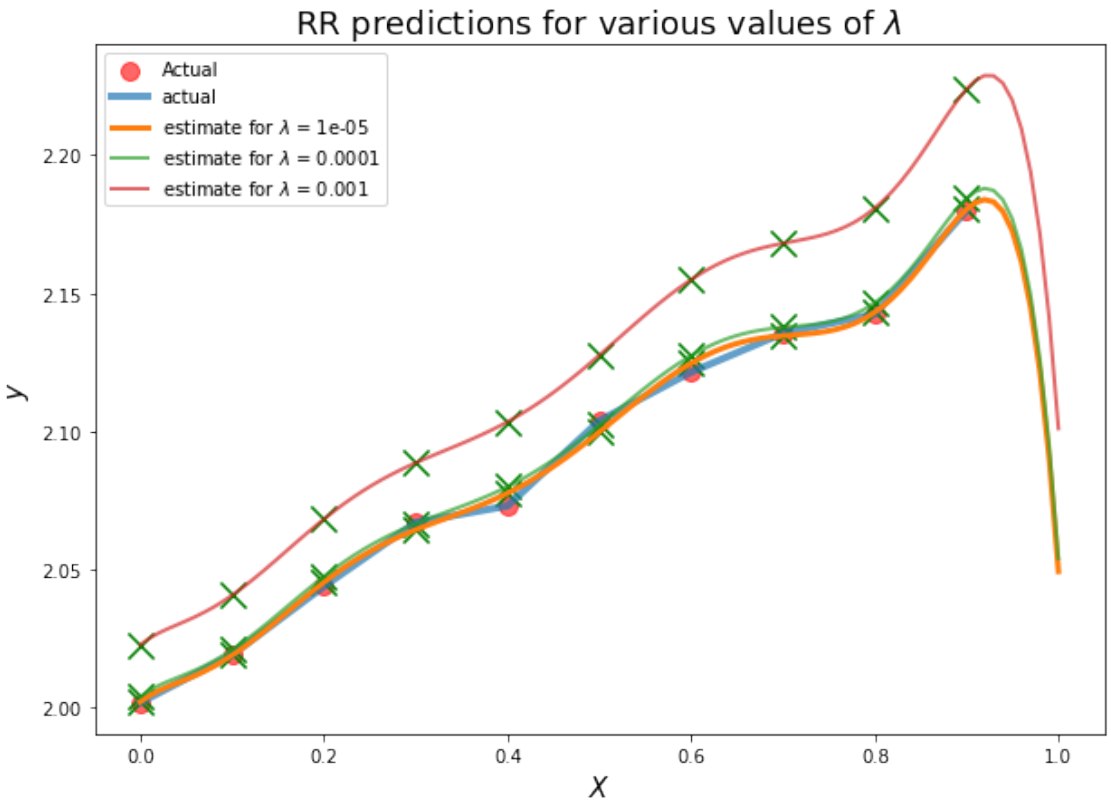
Fit a 8th degree polynomial on the data using the ridge regression estimator and plot the results (data points and the curve resulting from the fit). Output also the estimates of the parameters of the polynomial. Experiment with various values of λ.

We will first calculate the Ridge regression Estimators for small values of λ, since these give better results:

[36]:
```
def calculate_rr(X_p, lam, y):
    # Calculate \theta LS
    Xx_inv = np.linalg.inv(X_p.T.dot(X_p)) + lam * np.identity(9)

    Xy = X_p.T.dot(y)

    # Finally
    theta_rr = Xx_inv.dot(Xy)

    # data points:
    y_hat = theta_rr.T.dot(X_p.T)

    return theta_rr, y_hat


def plot_actual_vs_predicted_RR(X, X_p, y, l):
    # plot the data
    fig = plt.figure(figsize=(10,7))
    ax = fig.add_subplot(111)

    # Plot points
    ax.scatter(X,y,c='red',marker='o', label = 'Actual', s=100, alpha=0.6)


    # Plot actual line
    ax.plot(X,y,  linewidth=4, alpha=0.7, label='actual')

    ax.set_xlabel('$X$', fontsize=15)
    ax.set_ylabel('$y$', fontsize=15)


    for it, ld in enumerate(l):
        theta_rr, y_hat = calculate_rr(X_p, ld, y)

        ax.scatter(X,y_hat,c='green',marker='x', s=200)

        # plot polynomial
        x_points = np.linspace(0, 1, 100)
        if it==0:
```

```
            ax.plot(x_points, polynomial_coefficients(x_points, theta_rr),␣
 ↪linewidth=3, label=f'estimate for $\lambda$ = {ld}')
        else:
            ax.plot(x_points, polynomial_coefficients(x_points, theta_rr),␣
 ↪linewidth=2, label=f'estimate for $\lambda$ = {ld}', alpha=0.7)
        print(f'The estimates for lambda = {ld} are:')
        print(theta_rr.flatten())
        print()
    ax.set_title('RR predictions for various values of $\lambda$', fontsize=18)
    _ = ax.legend()


l = [10**i for i in range(-5,-2)]
plot_actual_vs_predicted_RR(X, X_p, y, l)
```

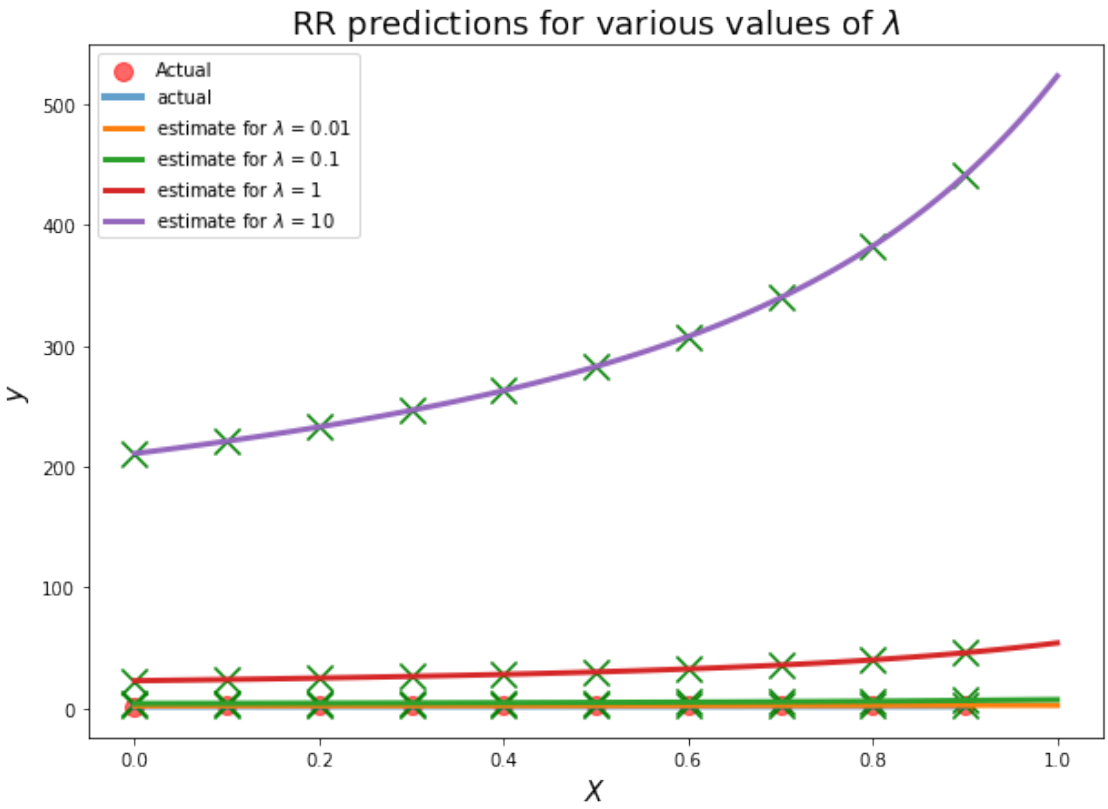
```
The estimates for lambda = 1e-05 are:
[ 2.00221134e+00  2.59453207e-01 -3.83230495e+00  4.94205933e+01
 -2.62207916e+02  6.98307312e+02 -9.86594727e+02  7.06146362e+02
 -2.01451584e+02]

The estimates for lambda = 0.0001 are:
[ 2.00409134e+00  2.60313243e-01 -3.83175659e+00  4.94209824e+01
 -2.62207611e+02  6.98307556e+02 -9.86594543e+02  7.06146484e+02
 -2.01451447e+02]

The estimates for lambda = 0.001 are:
[ 2.02289141e+00  2.68913567e-01 -3.82627296e+00  4.94248962e+01
 -2.62204620e+02  6.98309937e+02 -9.86592590e+02  7.06148071e+02
 -2.01450134e+02]
```



For large values of λ we have:

[32]:
```
l = [10**i for i in range(-2,2,1)]
plot_actual_vs_predicted_RR(X, X_p, y, l)
```

```
The estimates for lambda = 0.01 are:
[ 2.21089207e+00  3.54916871e-01 -3.77143669e+00  4.94640121e+01
 -2.62174927e+02  6.98333313e+02 -9.86573608e+02  7.06163818e+02
 -2.01436935e+02]

The estimates for lambda = 0.1 are:
[   4.09089866    1.2149497    -3.22307777   49.85523224 -261.87789917
  698.56787109 -986.38336182  706.32110596 -201.30496216]

The estimates for lambda = 1 are:
[  22.89096461    9.81527808    2.26051807   53.76735687 -258.90756226
  700.91320801 -984.48126221  707.8939209  -199.98524475]

The estimates for lambda = 10 are:
[ 210.89162412   95.81856176   57.09647274   92.88865662 -229.20419312
  724.36639404 -965.46038818  723.62225342 -186.78813171]
```



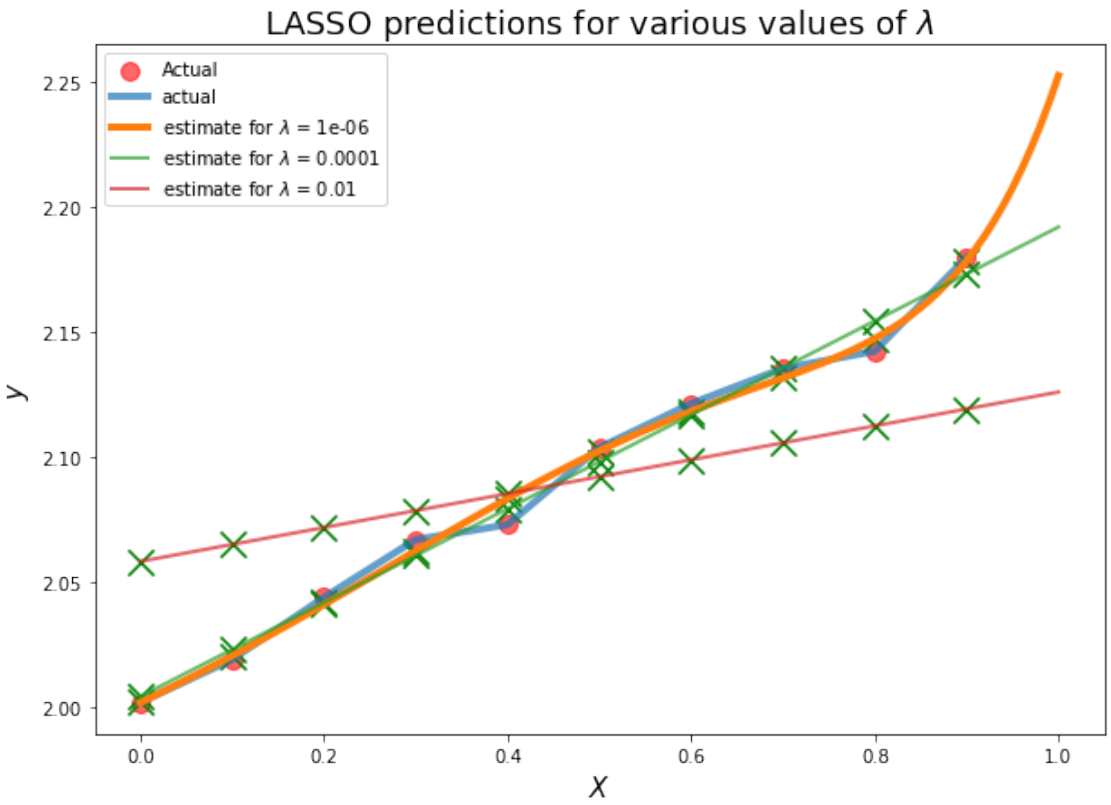
4.d

Fit a 8th degree polynomial on the data using the LASSO estimator and plot the results (data points and the curve resulting from the fit). Output also the estimates of the parameters of the polynomial. Experiment with various values of λ.

[33]:
```
def plot_actual_vs_predicted_LASSO(X, X_p, y, l):
    # plot the data
    fig = plt.figure(figsize=(10,7))
    ax = fig.add_subplot(111)

    # Plot points
    ax.scatter(X,y,c='red',marker='o', label = 'Actual', s=100, alpha=0.6)


    # Plot actual line
    ax.plot(X,y,  linewidth=4, alpha=0.7, label='actual')

    ax.set_xlabel('$X$', fontsize=15)
    ax.set_ylabel('$y$', fontsize=15)
```

```
    for it, ld in enumerate(l):
        clf = linear_model.Lasso(ld)
        clf.fit(X_p, y)

        theta_lasso = np.append(clf.intercept_,clf.coef_)
        ax.scatter(X,clf.predict(X_p),c='green',marker='x', s=200)

        # plot polynomial
        x_points = np.linspace(0, 1, 100)
        if it == 0:
            ax.plot(x_points, polynomial_coefficients(x_points, theta_lasso),␣
↪linewidth=4, label=f'estimate for $\lambda$ = {ld}')
        else:
            ax.plot(x_points, polynomial_coefficients(x_points, theta_lasso),␣
↪linewidth=2, label=f'estimate for $\lambda$ = {ld}', alpha=0.7)
        print(f'The estimates for lambda = {ld} are:')
        print(theta_lasso.flatten())
        print()
    ax.set_title('LASSO predictions for various values of $\lambda$',␣
↪fontsize=18)
    _ = ax.legend()

l = [10**i for i in range(-6,-1,2)]
plot_actual_vs_predicted_LASSO(X, X_p[:,1:], y, l)
```

```
C:\Users\jojoshulk\Anaconda3\lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:531: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 4.3395959829584926e-05, tolerance: 2.980546358500307e-06
  positive)

The estimates for lambda = 1e-06 are:
[ 2.00198987  0.17547491  0.11999218 -0.06109269 -0.12454477 -0.08486763
 -0.01077142  0.03825068  0.19828017]

The estimates for lambda = 0.0001 are:
[2.00441216 0.18774234 0.         0.         0.         0.
 0.         0.         0.        ]

The estimates for lambda = 0.01 are:
[2.05841216 0.06774234 0.         0.         0.         0.
 0.         0.         0.        ]
```



4.e

Discuss briefly on the results.

This excercise demonstrated the performance of various estimators on a given dataset. We approched he problem in three ways, with the following estimators: * LS estimator * Ridge Regression estimator * Lasso estimator

Initially we saw that the LS estimator is passing through all data, but it seems that after that it doesn't follow the "trend" of the data (which can be though of as a line). Then using the Ridge Regression, we see that for small values of λ the performance of the performance is similar, but as λ increases the predicted data points are moving away from the true values. Finally, using Lasso Regression, we can see that for small values of λ the predictions are the best, but as lambda increases, the predictions diverge from the actual values, and the predictions, then follow a line.

Exercise 5

5.i

Determine the range of values of λ where the MSE is smaller than that of the unbiased LS estimator,

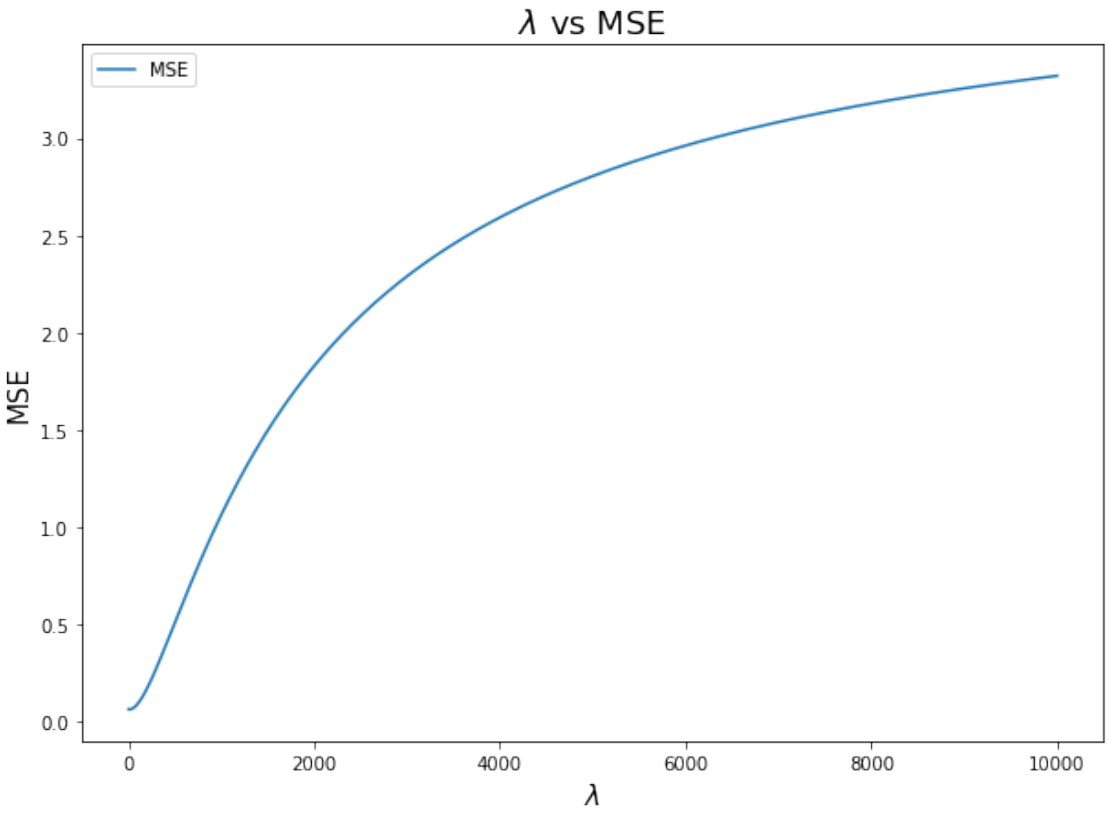
[239]:
```
def generate_data(state):

    r = np.random.RandomState(state)
    # Construct X matrix [1, x1, x2, x1*x2]
    X = r.uniform(low=0,high=10,size=(30,1))

    # define theta
    theta = 2

    # define normal error
    n = r.normal(0,np.sqrt(64),len(X))

    # Define y using only x1, x2
    y = theta * (X.T) + n

    #prin X and y
    return(np.concatenate((X, y.reshape(-1,1), n.reshape(-1,1)), axis=1))

def yield_index(ds, num):
    """
    Function to return a virtual dataset from a np array with 30 data points per␣
 ↪dataset
    Input:  an array containing all dataset, in order
            the requested dataset point (ex. in order to fetch dataset 30 num␣
 ↪should be 30)
    Output: the range in which the specific dataset can be found
    """
    return ds[num*30-30: num*30]

# Geneerate 50 datasets
data = np.empty((1,3))
for i in range(50):
    data = np.concatenate((data, generate_data(i)))
data = data[1:]

X_all = data[:,0]
y_all = data[:,1]
data[:,:5]
```

[239]:
```
array([[ 5.48813504, 17.89175967,  6.91548959],
       [ 7.15189366,  8.36646716, -5.93732016],
```

```
       [ 6.02763376, 30.21330451, 18.15803699],
       ...,
       [ 2.51929002,  7.97437456,  2.93579451],
       [ 3.02920439, 15.41652257,  9.35811379],
       [ 0.76047628,  7.45203207,  5.93107951]])
```

[240]:
```
# initialize values
MSE = []
min, pos = 1, 0
min_range = []
lambda_range = np.arange(0, 10000, 0.1)
# start iteration for various values of lambda
for j, ld in enumerate(lambda_range):

    # Calculate theta
    theta = []
    for i in range(50):
        # Fetch the dataset located in position i+1 of the X_all table
        X = yield_index(X_all, i+1)
        y = yield_index(y_all, i+1)

        # Calculate scalar tables
        XX = X.dot(X.T) + ld
        Xy = X.dot(y.T)

        # Append Theta
        theta.append(Xy/(XX))

    # append MSE
    cur_MSE = np.power((np.full((50), 2) - theta),2).mean()
    MSE.append(cur_MSE)

    # keep track of the index with the lowest MSE
    if cur_MSE < min:
        min, pos = cur_MSE, j


    if j == 0:
        MSE_LS = cur_MSE
    elif cur_MSE <= MSE_LS:
        min_range.append(j)

# Plot the results
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.plot(lambda_range, MSE, label='MSE')
```

```
ax.set_xlabel('$\lambda$', fontsize=15)
ax.set_ylabel('MSE', fontsize=15)

ax.set_title('$\lambda$ vs MSE', fontsize=18)

_ = ax.legend()
```

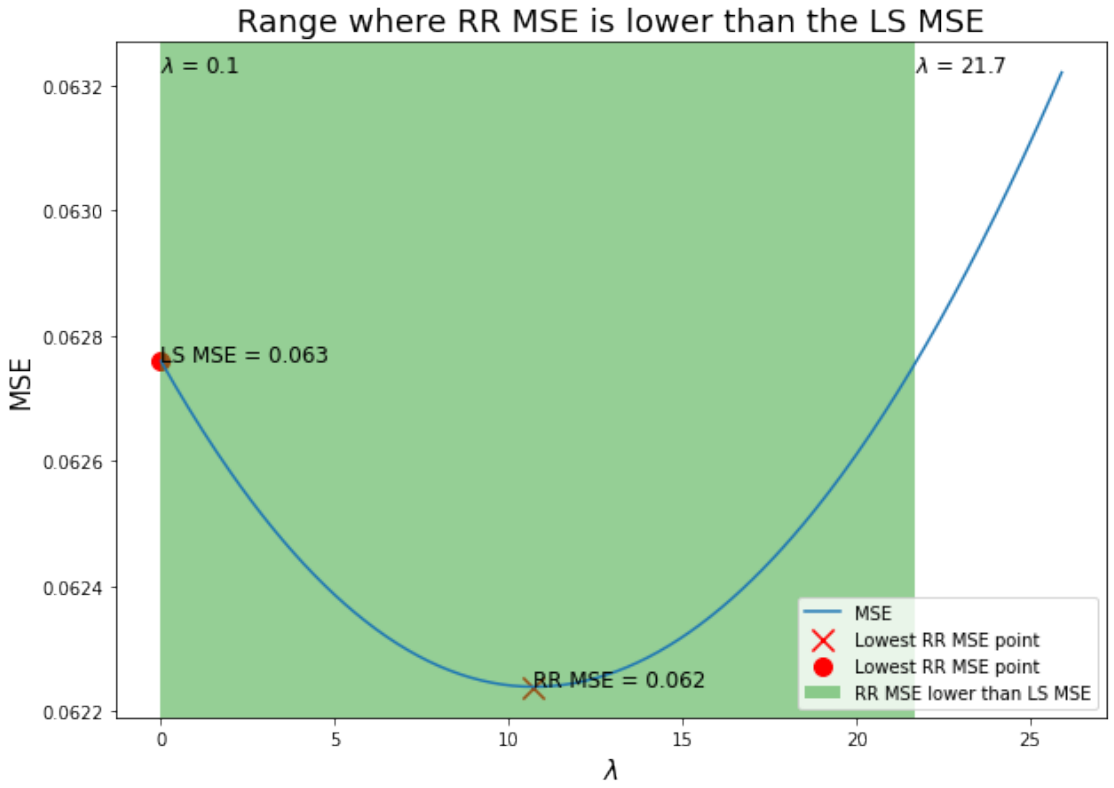


- Let's zoom in the data to get a better view

[241]:
```
plot_range = int(min_range[-1]*1.2)
lambda_range_zoom, MSE_zoom = lambda_range[:plot_range], MSE[:plot_range]

print(f"Range where the RR MSE is smaller than that the the LS MSE:␣
 ↪{lambda_range_zoom[min_range[0]]:.2f} <=  <= {lambda_range_zoom[min_range[-1]]:
 ↪.2f}")

# Plot the results
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)

ax.plot(lambda_range_zoom, MSE_zoom, label='MSE')
```

```
ax.scatter(lambda_range_zoom[pos], MSE_zoom[pos], c='r', label='Lowest RR MSE␣
 ↪point', marker='x', s=150)
ax.annotate(f"RR MSE = {MSE_zoom[pos]:.3f}", xy=(lambda_range_zoom[pos],␣
 ↪MSE_zoom[pos]), fontsize=12)

ax.scatter(0, MSE_zoom[0], c='r', label='Lowest RR MSE point', marker='o', s=100)
ax.annotate(f"LS MSE = {MSE_zoom[0]:.3f}", xy=(0,MSE_zoom[0]), fontsize=12)

ax.axvspan(0, lambda_range_zoom[min_range[-1]], facecolor='#2ca02c', alpha = 0.
 ↪5, label='RR MSE lower than LS MSE')
ax.annotate(f"$\lambda$ = {lambda_range_zoom[min_range[0]]:.1f}",␣
 ↪xy=(0,MSE_zoom[-1]), fontsize=12)
ax.annotate(f"$\lambda$ = {lambda_range_zoom[min_range[-1]]:.1f}",␣
 ↪xy=(lambda_range_zoom[min_range[-1]],MSE_zoom[-1]), fontsize=12)

ax.set_xlabel('$\lambda$', fontsize=15)
ax.set_ylabel('MSE', fontsize=15)

ax.set_title('Range where RR MSE is lower than the LS MSE', fontsize=18)

_ = ax.legend(loc='lower right')
```

```
Range where the RR MSE is smaller than that the the LS MSE: 0.10 <=  <= 21.70
```



5.ii

Comment on the results.

The above examples captures the essense of the theory shown so far. Knowing that the LS estimator is a MVU estimator, we tried various values of lambda and plotted the change in the RR MSE. As expected, there are λ s that have an MSE lower than the MVU. For large values of λ tough the MSE is getting close to 1.