Student Name: Jonathan Chandra
Student ID: S5161533

rijksuniversiteit groningen

Data Driven Optimization
Assignment 1 - Report

Problem 1 - Regression with basis functions

Firstly we define the input dataset as $u^{(i)}$. $u^{(i)}$ is generated using MATLAB `linspace` function. A total of 51 equally spaced points between 0 and 1 is generated. Since "0" is avoided as one of the points, $u^{(i)}$ is then defined to start from index 2 to index 51.

Using $u^{(i)}$ we define the output value $v^{(i)} = u^{(i)} \sin(2\pi u^{(i)}) + e^{(i)}$, where $e^{(i)}$ is an uncertainty drawn from from a uniform distribution over the interval [-0.1, 0.1]. $e^{(i)}$ is generated using `unifrnd` function. An addition of initialization of random number generator is made so the results is reproducible.

Now, it is assumed the regression problem where the function to be predicted is defined as:

$$h_\theta(u) = \theta_0 + \theta_1 u + \theta_2 u^2 + \ldots + \theta_M u^M \tag{1}$$

Where M is defined as the polynomial order of the model that will be predicted. Now the squared error is defined as:

$$E(\theta) := \frac{1}{50} \sum_{i=1}^{50} (h_\theta(u^{(i)}) - v^{(i)})^2. \tag{2}$$

In this Problem we will use three different value of M, that is $M = [1\ 3\ 9]$. Firstly we consider M to be 3, to find the minimizer of the squared error, one can define the linear cost function that is defined as

$$J(\theta) = \frac{1}{2m} (X\theta - y)^T (X\theta - y) \tag{3}$$

Since $\frac{1}{2m}$ is just a scaling factor, it won't affect the minimization result, one can omit the scaling factor. We can expand (3) and write it as

$$\begin{aligned} J(\theta) &= (X\theta)^T (X\theta) - 2y^T (X\theta) + y^T y \\ &= \theta^T X^T X\theta - 2y^T X\theta + y^T y \end{aligned} \tag{4}$$

Now, the derivative of $J(\theta)$ w.r.t θ is computed as below:

$$\frac{\partial J\theta}{\partial \theta} = X^T X\theta + X^T X\theta - 2X^T y = 2X^T X\theta - 2X^T y \tag{5}$$

To solve the variable, we have $\frac{\partial J\theta}{\partial \theta} = 0$, then:

$$\begin{aligned} 2X^T X\theta - 2X^T y &= 0 \\ X^T X &= X^T y \end{aligned} \tag{6}$$

Thus, the minimzer θ^* can be computed as:

$$\theta^* = (X^T X)^{-1} X^T y \tag{7}$$

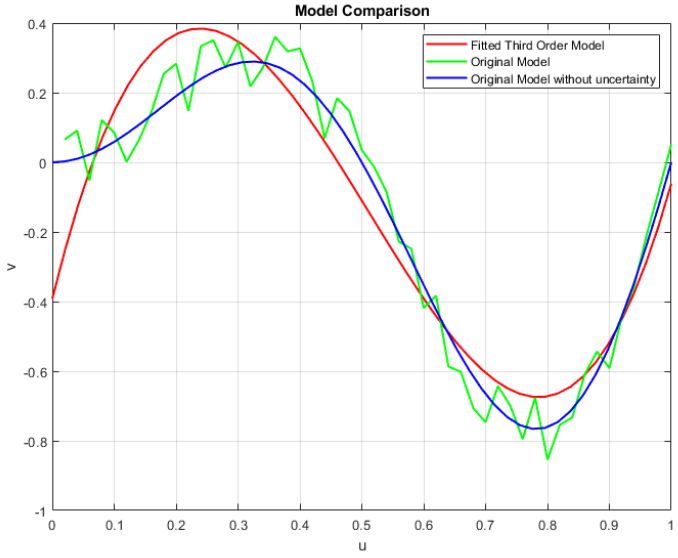


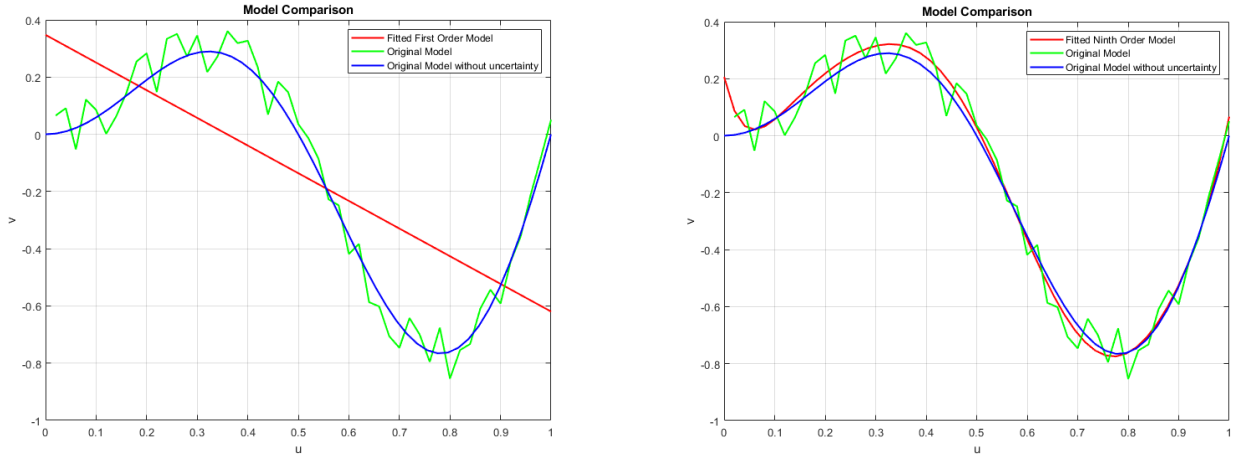
Figure 1: Model Comparison for $M = 3$

Where X is a $[(M+1) \times i]$ sized matrix and y is the output described earlier as $v^{(i)}$. Take an example where $M = 3$, then we have:

$$X = [\texttt{ones}(i, 1)\ u\ u^2\ u^3]. \tag{8}$$

Now for $M = 3$, we plot the function $u \mapsto h_{\theta^*}(u)$ and the function $u \mapsto u\sin(2\pi u)$. Figure 1 shows the comparison between the fitted model where $M = 3$ with the model using original data with and without uncertainty. We can see that the fitted model (red line) was able to follow most of the original model (blue line). It can also be observed that there's a gap between the initial data of the fitted and original model. This is due to the bias that was introduced in the initial value of $u^{(i)}$, which affects $v^{(i)}$. Another method to know the accuracy of the model, is to make use of the RMS error value. THE RMSE is calculated to be 0.1181 which can be said to be relatively small which means that the model is relatively able to fit the original data without having the uncertainty affect the predicted model.

Another model is needed to be used as a comparison. Now we try to fit the data using $M = 1$ and 3.



(a) $M = 1$ (b) $M = 9$

Figure 2: Model Comparison For $M = 1$ and 9

Figure 2a shows the model comparison for $M = 1$, while Figure 2 shows the model comparison for $M = 9$. We can observe that using lower M values results in underfitted model in Figure 2a which results the predicted model only a straight line since it is only first order polynomial function, while $M = 9$ results in a much better model of the predicted value but it tends to overfit the data. Overfitting occurs when the model is so complex that it starts learning the noise in the training data. As for the model RMS Error for $M = 1$ is 0.2531, while RMS Error for $M = 9$ is 0.0723. Both of those value is placed as expected since using higher value of M tends to bring the model predicted output close to the original output.

When deciding which M to use, What we want is to balance bias and variance. A model with too few features (e.g. $M = 1$) may have high bias and underfit the data, while a model with too many features (e.g. $M = 9$) may have high variance and overfit the data. In this case, choosing $M = 3$ may provide a good balance between bias and variance, which still assuming the underlying function is somewhat complex but not overly so.

Problem 2 - SVM with transformation

In this Problem, we are going to make use of SVM to solve a logistic regression problem of iris flowers within 'Iris.xlsx' dataset that was previously used in the second tutorial.

Firstly we load 'Iris.xlsx' dataset into MATLAB using either `readtable` or `xlsread` function in MATLAB. We define our input x_i with "SepalLengthCm" and "PetalLengthCm" while our output is defined with the "Class". Next we solve an optimization problem using CVX toolbox to acquire the optimal values of w and b, that is w_0^* and b_0^*. We formulate the optimization problem as:

$$\begin{aligned} &\min_{w,b} ||w||^2 \\ &\text{s.t} \;\; v^i(w^T u^{(i)} + b) \geq 1, \;\; \forall\, i \;\in \{1, \ldots, N\} \end{aligned} \tag{9}$$

Through CVX toolbox we acquire w_0^* and $b_0^* = [(-3.9477e-5) \quad -1.8181]$ and 4.4547 respectively. Using the acquired optimal values w_0^* and b_0^* we are able to create a decision boundary and the support vectors which are calculated using MATLAB. Figure 3 is a visualization of the dataset seperation of Setosa and

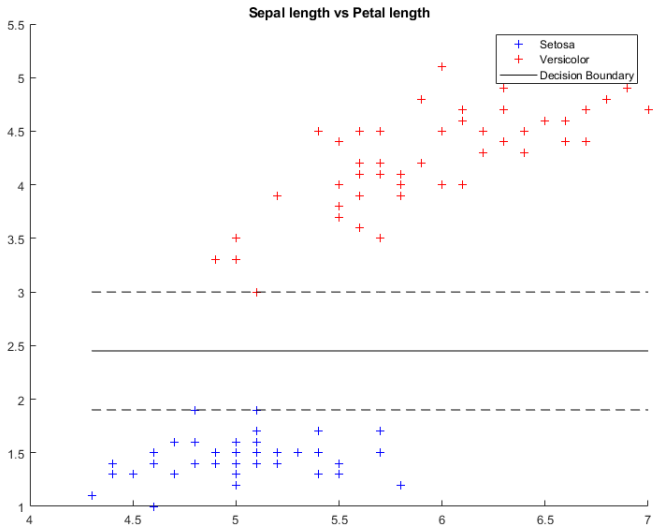


Figure 3: Logistic regression using SVM

Versicolor with its decision boundary and support vector. Using the computed w_0^* and b_0^*, we are able to create a decision boundary which is a hyperplane that seperates both of the classes.

Now we try to shift the input values x_i by a vector $[\pi \;\; \frac{\pi}{e}]$ while maintaining the output label. Using the same step as before we should be getting different value of the optimal values of b but not w. The optimal value of these shifted input is denoted as w_1^* and $b_1^* = [(-2.2764e-5) \quad -1.8181]$ and 6.5560 respectively. Using the acquired optimal values w_1^* and b_1^* we are able to create a decision boundary and the support vectors which are calculated using MATLAB.

Figure 4 is a visualization of the dataset seperation of Setosa and Versicolor with its decision boundary and support vector with its shifted input. Using the computed w_1^* and b_1^*, we are able to create a decision boundary which is a hyperplane that seperates both of the classes. It is to observed that both of the weight w_0^* and w_1^* have similar value (I say similar because w(1) of both weight have a very small difference where such a small difference might not be significant). This is due to that the weight in SVM only determines the orientation of the decision boundary (Hyperplane. The orientation is dependent to the relative positions

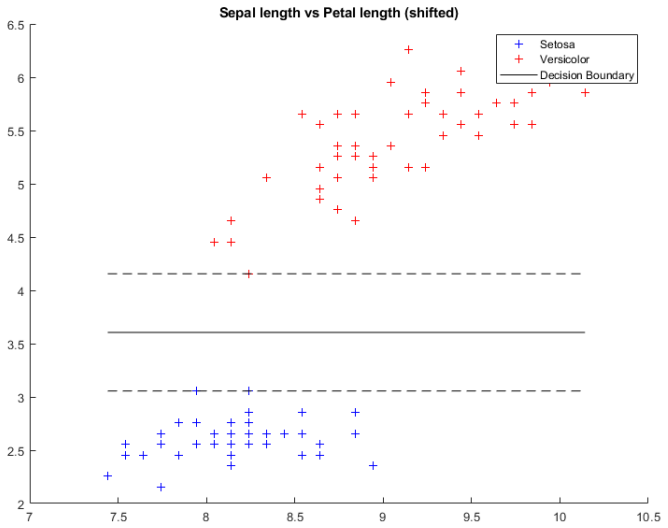


Figure 4: Logistic regression using SVM with shifted input

of the data points of each other, not their absolute position in the space. The bias term, on the other hand, helps position the hyperplane in the space relative to the origin, so it does change when the input is shifted.

Lastly, we rotate the input data x_i counter-clockwise by $\theta = \frac{\pi}{3}$ using the rotation matrix defined as:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{10}$$

where x_i becomes

$$x_i = R(\theta)x_i. \tag{11}$$

Using the same step as before we should be getting different value of the optimal values of w, but not b. The optimal value of these rotated input is denoted as w_2^* and $b_2^* = [-1.5746 \quad -0.9090]$ and 4.4547 respectively. Using the acquired optimal values w_2^* and b_2^* we are able to create a decision boundary and the support vectors which are calculated using MATLAB.

Figure 5 is a visualization of the dataset seperation of Setosa and Versicolor with its decision boundary and support vector with its rotated input. Using the computed w_2^* and b_2^*, we are able to create a decision boundary which is a hyperplane that seperates both of the classes. We can observe that w_2^* is different from w_0^* and w_1^*. This is due to the definition of the weight in the SVM determines the orientation of the decision boundary (hyperplane), and this orientation is based on the relative positions of the data points. When we rotate the data, it affects these relative positions and angles, so the orientation of the optimal hyperplane also changes. As a result, the weights (which encode the orientation) will change.

Moreover, b_2^* have the same value as b_0^*. This suggests that the rotated data and original data share the same distance from the origin to the decision boundary along the direction of the weight vector.

***Note**

A copy of mfiles used to make calculation and the plots of all Figure is provided in the submission folder of Brightspace. The name of each mfiles indicates the problem number in the assignment.

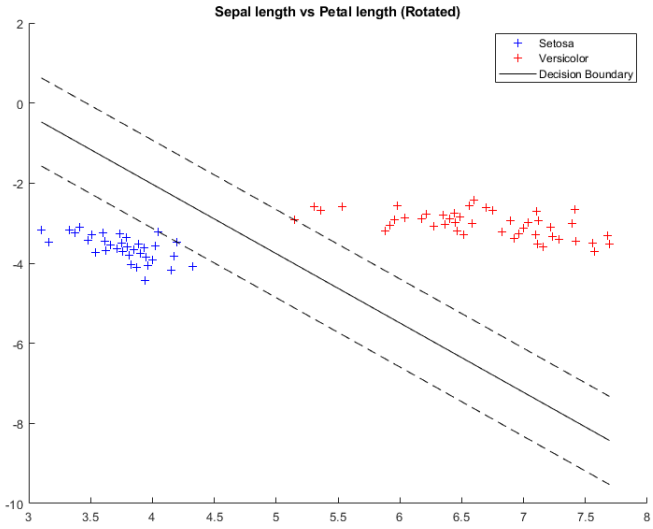


Figure 5: Logistic regression using SVM with rotated input