GROUP MEMBERS:

1. DAN SERBANOIU 244435207053
2. REGINA OCHONU 236022548645
3. MOATASIM MAHMOUD 235186856077

ASSIGNMENT 1

Note: Our **Python codes** have been emailed;
Mail subject:
File name:

PROBLEM 1

We know that Regression involves modeling the relationship of a dependent random variable, **y**, considered to be the response of a system, when its input is activated by a set of random variables, $\mathbf{x_1}$, $\mathbf{x_2}$,…, $\mathbf{x_l}$ (represented as the components of a feature (random) vector, **X)**. The relationship is modeled via an additive noise term, **η** (an unobserved random variable).

We also know that the goal of regression is to estimate the parameter vector, **θ**, which defines the input/output dependence, given a set of measurements, $\mathbf{(y_n, x_n)}$, where; n = 1, 2, . . ., N, and $y_n \in \mathbb{R}$, $x_n \in \mathbb{R}^l$, (called the training data set) that we have at our disposal.

For a linear model, we have that; $\mathbf{y = \theta_0 + \theta_1 x_1 + \ldots + \theta_l x_l + \eta = \theta_0 + \theta^T X + \eta}$, or for short;

$$\mathbf{Y = \theta^T X + \eta};$$

In the Exercises, we have been given:

- The general model; $y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_5 x^5 + \eta$
- η – white and Gaussian
- Components of the weight vectors; $\theta_0 = 0.2$, $\theta_1 = -1$, $\theta_2 = 0.9$, $\theta_3 = 0.7$, $\theta_4 = 0$, $\theta_5 = -0.2$
- Feature variables, $x_1, x_2, \ldots, x_n$; are equidistant points in the interval [0, 2], with n = 1, 2, . . ., N.
- Training sets; $y_n = \theta_0 + \theta_1 x_n + \theta_2 x_n^2 + \theta_3 x_n^3 + \theta_5 x_n^5 + \eta_n$, n = 1, 2, . . ., N, and η_n - i.i.d noise samples with mean = 0, Variance = σ_n^2.

Problem 1.1

Here, the Least-Squares (LS) method which involves minimizing the cost function (the squared-error loss over all data points) was used to estimate the parameter vector, **θ**, using a 5th degree polynomial (the structure of the correct model) and a noise variance, $\sigma_n^2 = 0.1$.

Results:

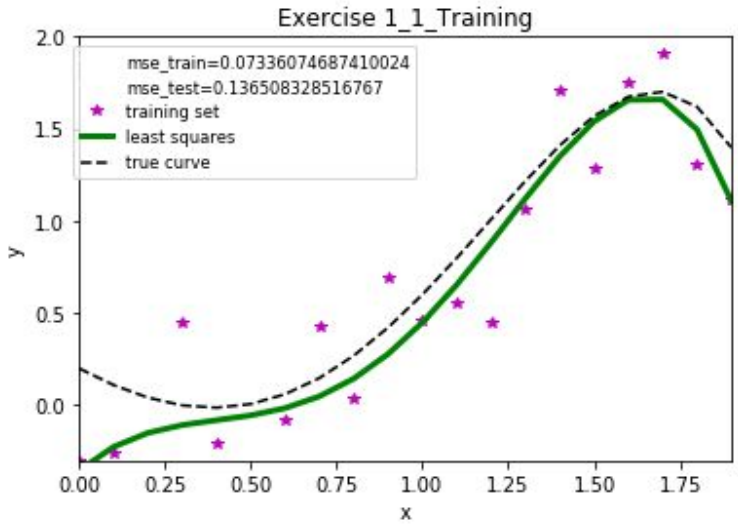

Fig1: Curve obtained using LS method

Observations:

- Mean squared error (MSE) obtained with the training points (N=20), $\mathbf{MSE_{Train}= 0.0734}$ and that obtained from the test sets (N=1000), $\mathbf{MSE_{Test}= 0.1365}$.

Conclusion:

The MSE is better over the training set than over the test set. This is expected since we used the training set to train the estimator. Also, since the structure of the correct model was used (5th degree polynomial with the coefficient of the 4th power equal to zero), we got a curve close to the true curve which is not over-fitted and thus, will give a good prediction.

Problem 1.2

We are aware that a reasonable measure to quantify the performance of an estimator is its mean-square deviation from the optimal one, which is given by; $E_D[(f(x; D) - E[y|x])^2]$, where the mean is taken with respect to all possible training sets (say D). Also, when changing from one training set to another, the mean-square deviation from the optimal estimate comprises two terms; the **Variance term** (contributed by the variance of the estimator around its own mean value) and the **Bias term** (which is the difference of the mean from the optimal estimate).

Results:

Using a 2nd and 10th degree polynomial, we performed 100 experiments with different noise samples, then we plotted the mean, variance and the true model curve on the (x,y) plane as instructed and obtained the graphs in Fig.1.2 below;

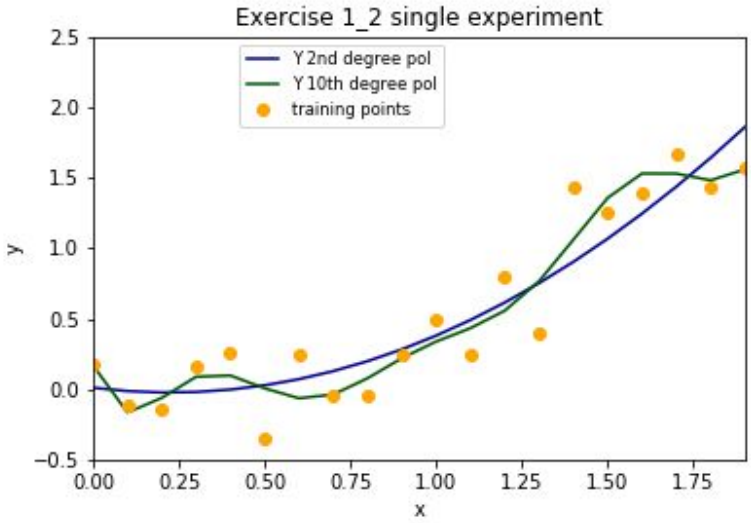

Figure1.2(a): Illustrate the results from fitting a 10th and 2nd order polynomial for a single experiments.

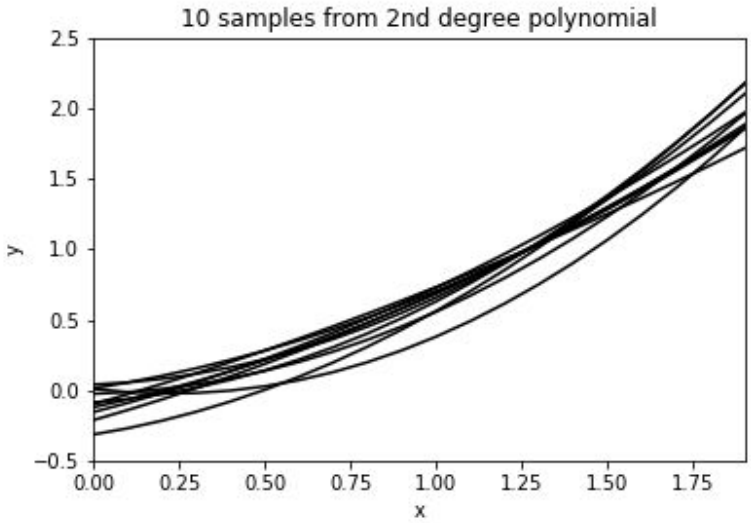

Figure1.2(b): Illustrate the results from fitting 10 sample experiments from the 2nd order polynomial

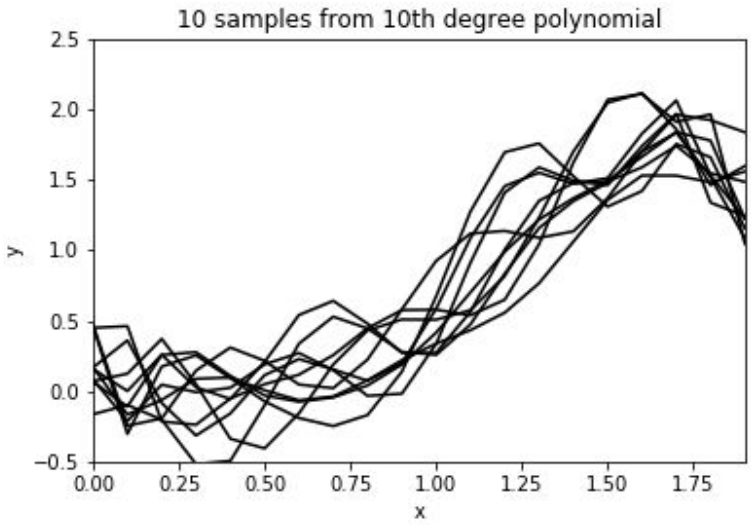

Figure1.2(c): Results from fitting 10 sample experiments from the 10th order polynomial

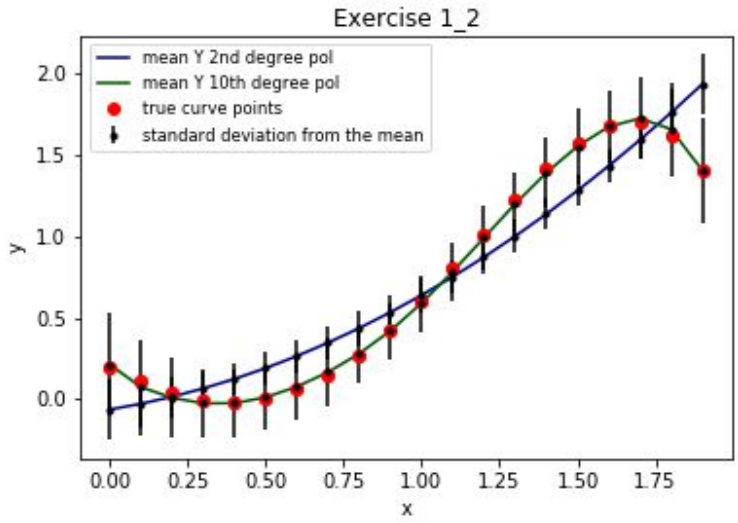

Figure 1.2(d)The resulting curve from fitting a 2nd and 10th degree polynomial averaged over 100 different experiments, together with the curve of the true model.

Observations:

- For every experiment using the 10th degree polynomial, the estimated curves follows too closely the training points (i.e., its overfitted) and the different experiments differ very much from each other. We can see that there is a very large variance compared to the curves obtained for the 2nd degree polynomial.
- From our experiment, we see that one cannot make both the mean and variance terms small simultaneously for a fixed number of training points in the data sets given, trying to minimize the variance term results in an increase of the bias term and vice versa.
- From Fig.1.2(d), we see that for the 2^{nd} degree polynomial, the value of the bias term is large and the value of the variance term is quite small compared to the 10^{th} degree polynomial.

- In order to reduce the bias term, we increase the complexity of the LS estimator from 2nd degree to 10th degree polynomial (i.e, more free parameters), but this resulted in a higher variance.
- This poses a dilemma (called the **Bias - Variance dilemma**). Thus, there has to be a tradeoff between the bias term and the variance term.
- We can reduce both the bias term and variance term simultaneously by increasing the number of training points, N and at the same time carefully increasing the model complexity. This will help resolve the Bias - Variance dilemma**.**

Conclusion:

We thus conclude that to obtain a low overall MSE, the complexity (number of parameters) of the model should be small enough with respect to the number of training points.

Problem 1.3

We know that one can improve the MSE performance of an estimator by shrinking the norm of the Minimum Variance Unbiased (MVU) estimator. We are also aware that the LS estimator is a MVU estimator under the assumption of linearity of the regression model and in the presence of a Gaussian white noise source (which is the case in this experiment). Ridge regression, thus involves regularising LS solution for the linear regression task by constraining the norm of the parameter vector. This is done by imposing **bias** (given by the value of lambda, $\boldsymbol{\lambda}$) on the LS estimator.

Results:

Fig1.3 below shows the different values of lambda, $\boldsymbol{\lambda}$ used for our experiment and the corresponding MSE of the biased estimator.

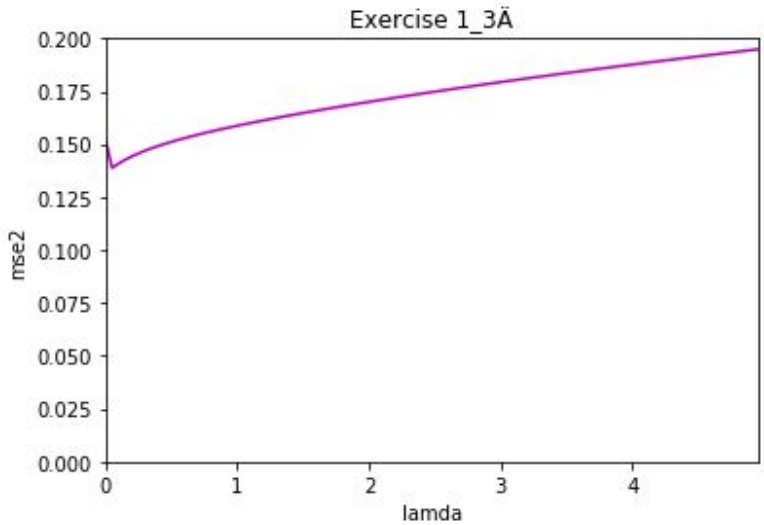

Fig 1.3(a): Mean squared error over a test set for different values of lambda in the range[0-4]

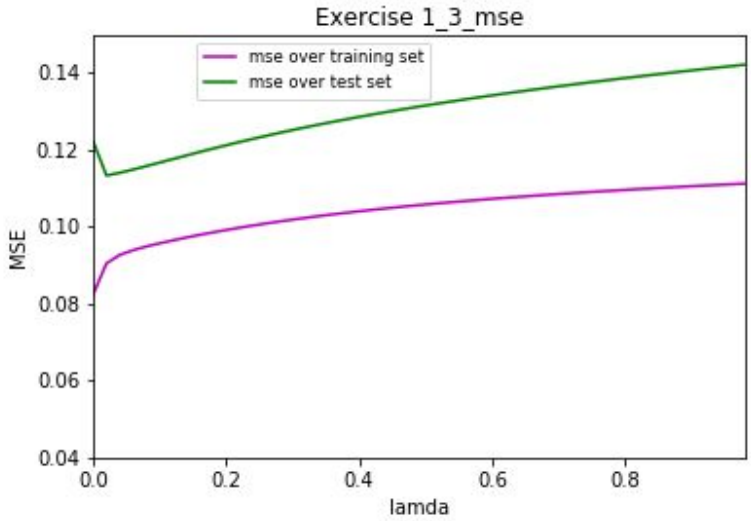

Fig 1.3(b): Mean squared error over the training set and test set for different values of lambda in the range[0-1]

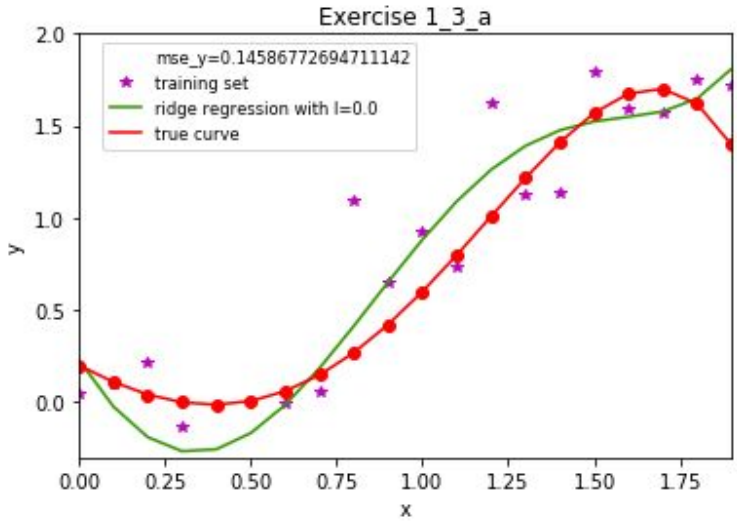

Fig 1.3(c): Curve obtained using lambda = 0 (same as LS method)

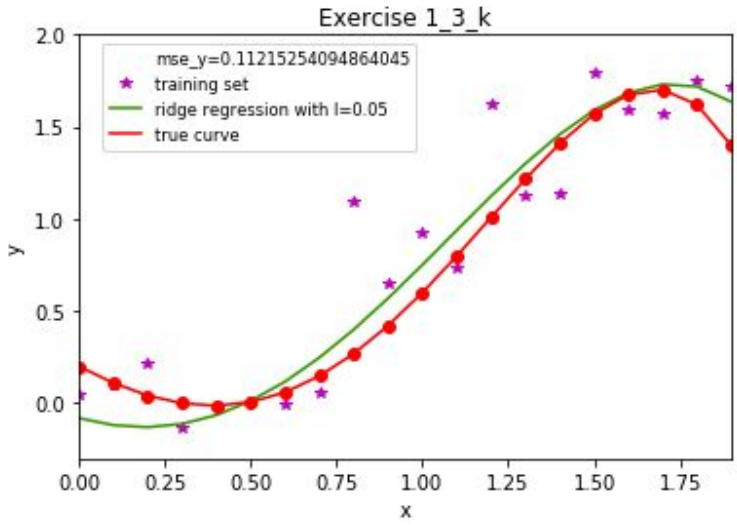

Fig 1.3(d): Curve obtained using Ridge regression with lambda = 0.05

Observations:

- As can be seen from the graph in Fig1.3(a), the MSE decreases with lambda close to 0.1 and then starts to increase as the value of lambda increase further.
- In fig 1.3(b), as we introduce more bias (i.e, shrinking the norm), the MSE over the training set is always worse than the LS estimator (where lambda = 0) and it keeps getting higher with increasing value of lambda. But the MSE over the test set shows to be better for certain values of lambda (around 0.05) than at lambda = 0 (LS method).
- From the Fig 1.3(c & d) above we see that the performance of the ridge regression estimator can offer an improvement to the MSE as compared to the LS estimator. The curve obtained from Ridge Regression is closer to the true curve.

Conclusion

Thus, we ascertained from our experiments that there always exists a $\boldsymbol{\lambda > 0}$, such that the ridge regression estimate gives an MSE lower than the MSE corresponding to MVU estimator (LS estimator).

Problem 1.4

In the full Bayesian Inference, when given an input vector *x*,one can predict the respective value of *y* using the most probable value, i.e., μ_y. Thus, more information concerning the predicted value is available, since we have an estimate of the respective variance, which quantifies the associated uncertainty of the prediction.

Results:

In this experiment, a Gaussian prior ($G(\boldsymbol{\theta})$) for the unknown θ was used with mean θ_0 equal to the previous true set of parameters given by; $\theta_0 = 0.2$, $\theta_1 = -1$, $\theta_2 = 0.9$, $\theta_3 = 0.7$, $\theta_5 = -0.2$, and $\Sigma_\theta = 0.1I$. Also, the true model structure was used to construct the matrix Φ. The following figures provide the graphical illustration of the results;

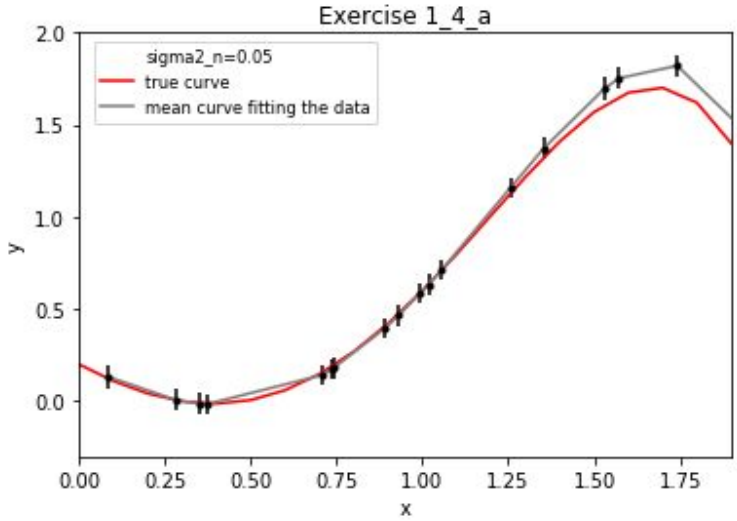

Figure 1.4 (a)

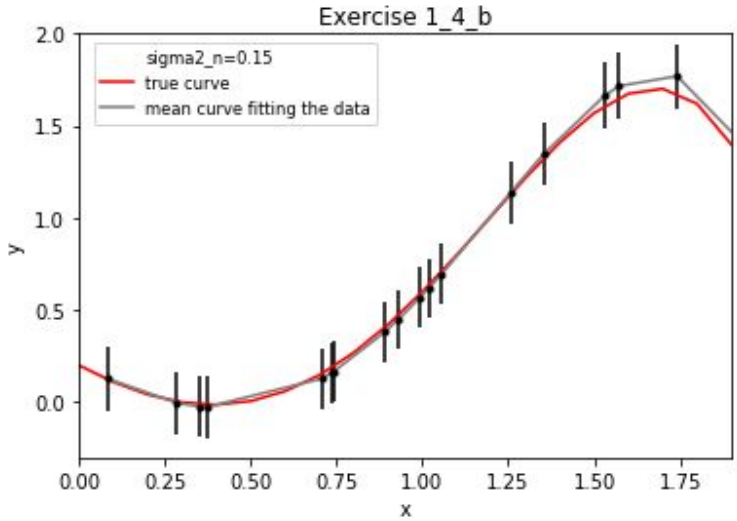

Figure 1.4 (b)

Each one of the black points, (y, x), indicates the prediction (*y*) corresponding to the input value, *x*. The error bars are dictated by the obtained variance, σ_y^2 . The mean values, $\boldsymbol{\theta_0}$, used in the Gaussian prior G($\boldsymbol{\theta}$) are equal to the true values of the unknown mode. Fig 1.4(a) $\sigma_\eta^2 = 0.05$, N = 20, $\sigma_\theta^2 = 0.1$ (b) $\sigma_\eta^2 = 0.15$, N = 20, $\sigma_\theta^2 = 0.1$.

Observations:

- We used our prior knowledge about the theta to find our estimate so we got results that are very close to the true curve for both values of the noise variance.
- We also observe that if we increase the noise variance our uncertainty about the output increases

Problem 1.5

Results:

For this experiment, we kept the correct model, however, the mean of the prior was given a different value to that of the true model, namely: θ_0 = [-10:54; 0:465; 0:0087; -0:093; -0:004]^T . The obtained curves are as shown below;

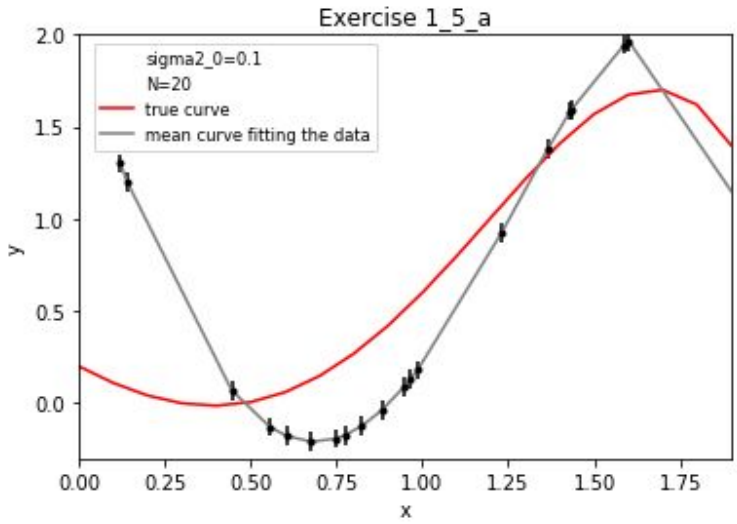

Figure 1.5 (a)

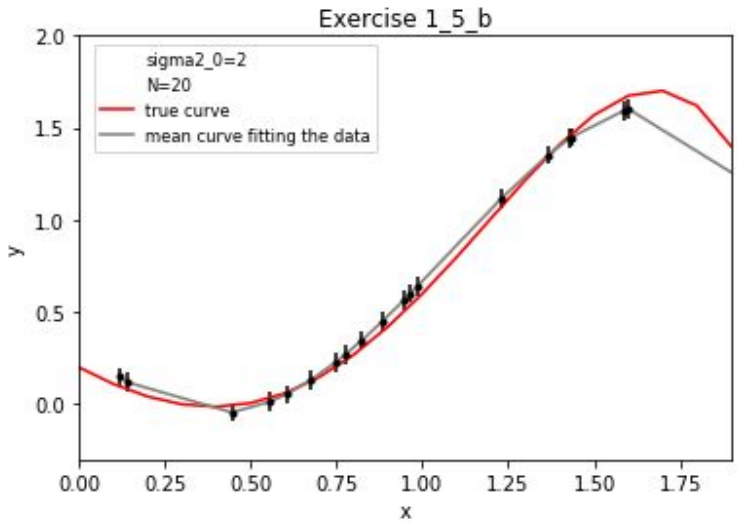

Figure 1.5 (b)

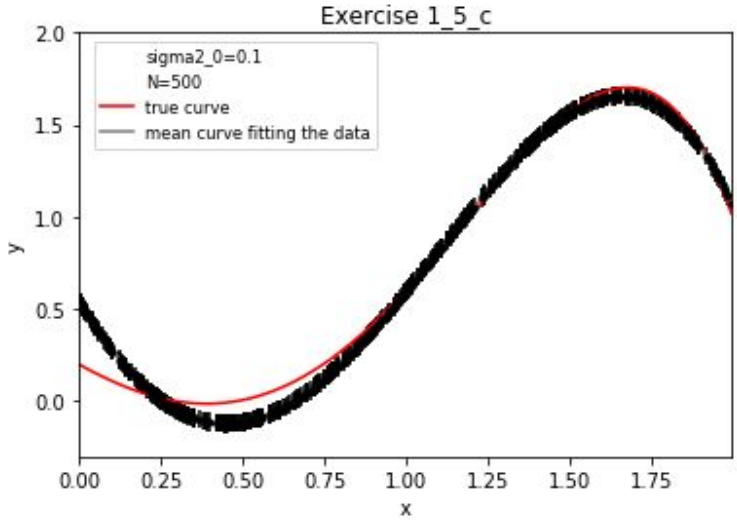

Figure 1.5 (c)

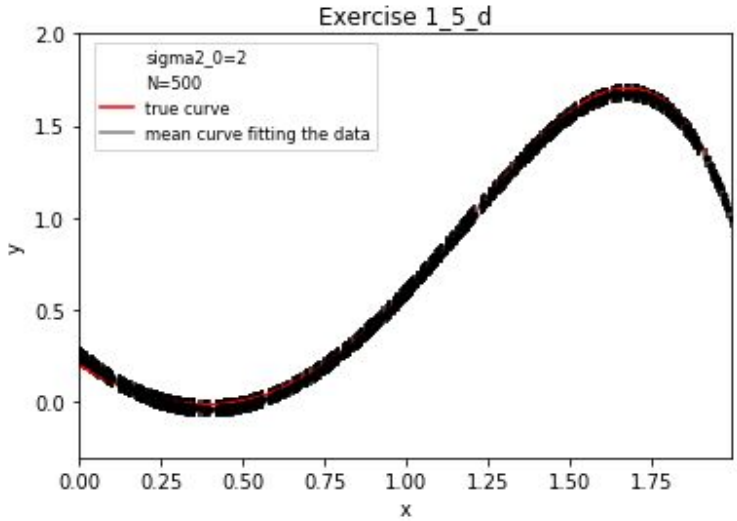

Figure 1.5 (d)

Each one of the black points, (y, x), indicates the prediction (*y*) corresponding to the input value, *x*. The error bars are dictated by the obtained variance, σ_y^2 . The mean values, $\boldsymbol{\theta_0}$, used in the Gaussian prior $G(\boldsymbol{\theta})$ are different from the true values of the unknown model, given by; $\theta_0 = -10.54$, $\theta_1 = 0.465$, $\theta_2 = 0.0087$, $\theta_3 = -0.093$, $\theta_5 = -0.004$). Fig 1.5(a) $\sigma_\eta^2 = 0.05$, N = 20, $\sigma_\theta^2 = 0.1$ (b) $\sigma_\eta^2 = 0.05$, N = 20, $\sigma_\theta^2 = 2$ (c) $\sigma_\eta^2 = 0.05$, N = 500, $\sigma_\theta^2 = 0.1$ (d) $\sigma_\eta^2 = 0.05$, N = 500, $\sigma_\theta^2 = 2$

Observations:

- Here we used a prior knowledge that is different from the true model. If the number of data points is small and the uncertainty about θ_0 is also small ($\sigma_\theta^2 = 0.1$), then the model follows our prior more than it follows the data points, thus, the model behaves badly.
- Using a larger variance ($\sigma_\theta^2 = 2$) for the prior, the model performance improves.
- The larger the data set is (N = 500) the better the model and hence the better the predictions are.

- When we use $\sigma_\theta^2 = 2$ and N = 500, we get the best model performance compared to all other cases and the curves follows the data points closely.

Conclusion for Problem 1.4 and 1.5:

When we use θ_0 equal to the correct θ, Bayesian Inference gives good results even when we do not have a large number of points. But when using θ_0 that is far from the true model (not certain about it), we must have a large number of points and use a higher σ_θ^2, so that the estimator pay more attention to the data points than to θ_0.

For smaller values of N, there is extra uncertainty, which is associated with the parameter θ, measured by σ_θ^2 and for a large number of observations, σ_y^2 tends towards σ_η^2, and our uncertainty is only contributed by the noise source, which cannot be reduced further.

Problem 1.6

The Expectation-Maximization (EM) algorithm is used to maximise the likelihood function for problems with latent variables. Here, we are using it for $\sigma_\eta^2 = 0.05$, N = 500. And we initialize $\alpha = \beta = 1$.

Results:

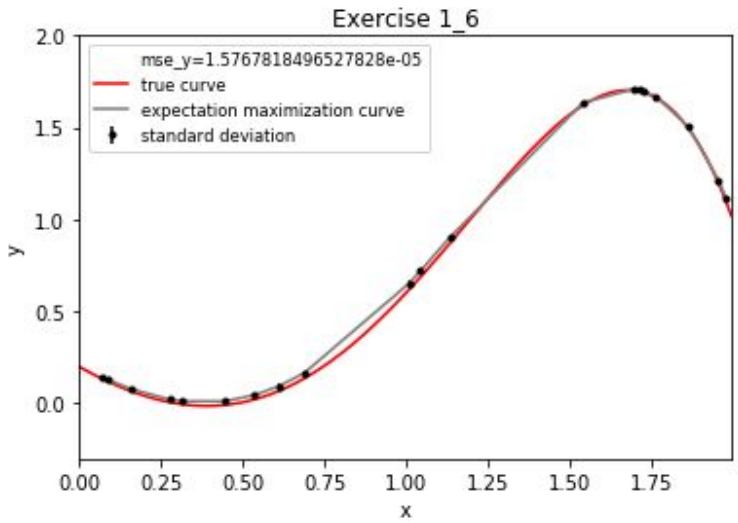

Fig1.6(a): Curve obtained from ME algorithm with comparison to the true curve.

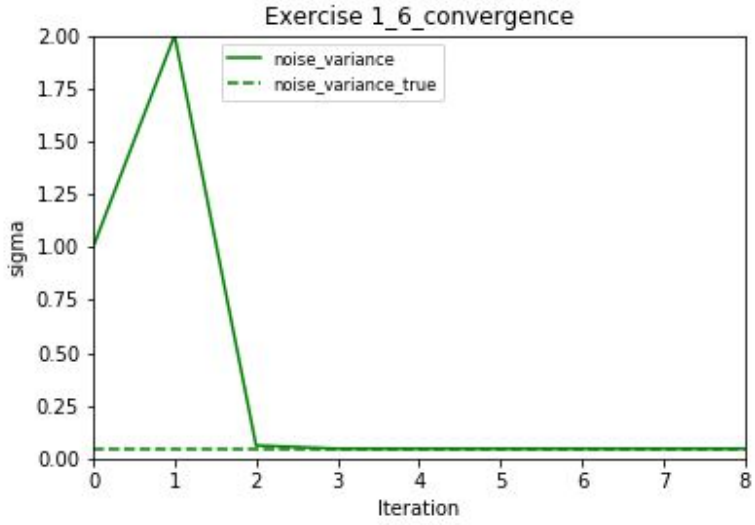

Fig1.6(b): The convergence curve of σ_η^2 to its true value

Observations:

-ME algorithm converged very quickly and gave good results (MSE = 1.58e-5).

-Fig1.6(b) shows how the value of σ_η^2 converges to the true value (0.05).

PROBLEM 2

The goal of a classification task is to partition the space into regions and associate each region with a specific class.

Problem 2.1

KNN is a simple learning algorithm that that does not make any assumptions about the underlying data set, that is it is non - parametric. It makes its selection based on the proximity to other data points regardless of what feature the numerical values represent and we can immediately classify new data points as they are presented.

If the number of training samples is large enough, then as the value of k increases, the classification error probability tends to the optimal Bayesian error, P_B.

Results:

Using the leave-one-out (LOO) cross validation method (i.e., each time one sample is left for testing), we obtained the following results;

K Values	Percentage Accuracy for Iris Plant Database	Percentage Accuracy for Pima Indians Diabetes Database
1	96.00	67,67
3	96.00	69.40
5	96.67	71.48
7	96.67	72.79
9	96.67	73.44
11	97.33	73.31
13	96.66	74.74
15	97.33	74.09
17	97.33	75.26
19	98.00	76.04

The graph below also shows the percentage of correct classification (Accuracy) as a function of the number of nearest neighbours (K-values).

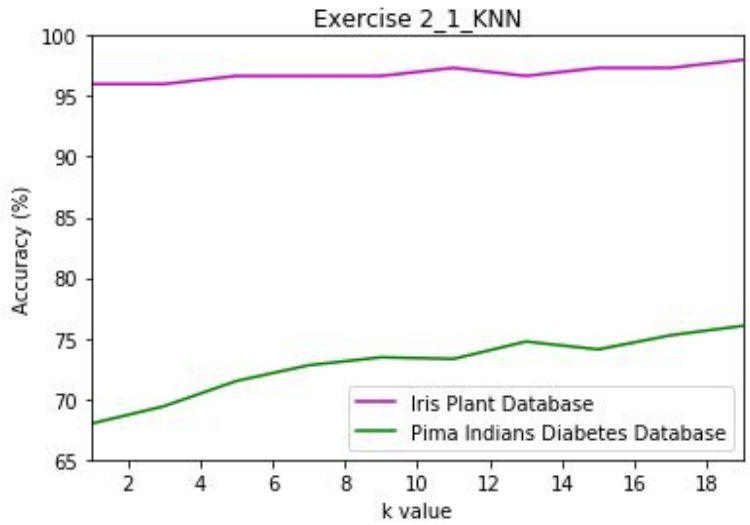

Figure 2.1

Observations:

- As shown in table 2.1 and figure 2.1, the percentage of correct classification increases as the number of nearest neighbors increases.

Problem 2.2:

The Bayes classifier is an optimal classifier. And the probability of error, $\mathbf{P_B <= P_{NN} >= 2P_B}$ **,** where P_{NN} is the probability of error for the KNN classifier at K = 1.

Here we also used the Leave - one - out cross validation method to evaluate the performance of the Bayes classifier.

Results:

- The percentage accuracy (correct classification) of the Bayes classifier for Pima Indians Diabetes Database is; **73.96%**. Implying that the probability of error, $\mathbf{P_B = 0.2604.}$

Observations:

- From Problem 2.1 above, the percentage accuracy at K = 1 for the KNN model is **67.97%,** that means the probability of error $\mathbf{P_{NN} = 0.3203.}$
- Therefore, the relation; $\mathbf{P_B <= P_{NN} >= 2P_B}$ is valid from our experiments [$\mathbf{0.26 < 0.32 > 2(0.26)}$]
- From our experiment in Problem 2.1, even with the best accuracy (76.04%) at K = 19, the KNN classifier still performed better than the Bayes classifier.

Conclusion:

For large number of training points the Bayes classifier always performs better than the KNN classifier.

Problem 2.3

We know that the Naive Bayes classifier is a means of coping with the curse of dimensionality and to exploit more efficiently the available training set because it assumes that individual features x_i, for i = 1, 2,...l are statistically independent. Thus, a number of points becomes of the order $N \cdot \ell$.

It turns out that

Result:

- The Percentage of correct classification of the Naive Bayes classifier for Pima Indians Diabetes Database is; **75.40%**.
- The Naive Bayes classifier performed better at classifying the Pima Indians Diabetes Data than the Bayes classifier and the KNN classifier.
- Since each dimension is independent of all the other dimensions, the curse of dimensionality does not affect the result of the Naive Bayes classifier.

Conclusion:

The Naive-Bayes classifier seem to be robust and has shown to perform well for real world datasets even when the independence assumption is violated.

Problem 2.4

Result:

A separating hyperplane was found, that seperates the Iris-setosa class from the other two classes. The hyperplane parameters were [1.3, 4.1, -5.2, -2.2, -1.]. And the algorithm did not converge for the other two classes.

Conclusion:

The Iris-setosa class is the only class that is linearly separable from the other classes.The perceptron algorithm requires for the classes to be linearly separable in order to converge. Hence, one is advised to use it only when the data can be linearly separated into the desired classes.