HW 1 Solutions

Author:
FATEME NOORZAD

StudentID:
810198271

MACHINE LEARNING
Fall 2020

Contents

1 Question 1 **1**

1.1 Part A 1

1.2 B 1

2 Question 2 **3**

2.1 Part A 3

2.2 Part B 5

3 Question 3 **7**

4 Question 4 **8**

5 Question 5 **10**

5.1 Part A 10

5.2 Part B 11

6 Question 6 **12**

6.1 Method 1 12

6.2 Method 2 14

7 Question 7 **15**

8 Question 8 **19**

Chapter 1

Question 1

1.1 Part A

Generally, separating dataset into test and training datasets depends on many factors. Therefore, there is no general rule to separate dataset in an optimal manner. However, there are some separation, which are more popular than others. Among which are 80%- 20%, 67%-33%, and 50%-50%. The latter case is mostly optimal when we have large datasets.
In this problem, as the results suggests, the second way of separating data in which 95% of it is used as training dataset, achieves a better accuracy. Therefore, it is obvious that the second method is better than the first one, if our measurement is their accuracy.
In the first separation, it is mentioned that half of dataset is used as training. Based on the size of dataset, as well as the accuracy results, dividing dataset into half is not an optimal choice. In addition, it is essential to state the fact that fewer training data may not represent the whole pattern of data accurately. The given accuracy proves the stated fact.
Besides, few data for training erges the model to learn specific models. Therefore, the trained result suffers from being overfit to it and do not have the ability of being generalized.
The second kind of separation, seems to used enough data for training. Since the model is trained based on sufficient it can perform well while facing new data. Therefore, the trained model in this matter is a better choice for generalization. However, it is vital to note that too much training can be costly. In cases where cost is more crucial than accuracy, choosing a model to achieve this goal is essential. With keeping all of its merits in mind, it is worth mentioning here that few test data can be harmful in some cases. Although there are methods, such as cross-validation, to overcome this issue, using this method blindly is not suggested.
To sum up, based on the goal we want to achieve, which can be better accuracy, faster results, and so on, each of the separation can become practical, although they have merits and demerits.

1.2 Part B

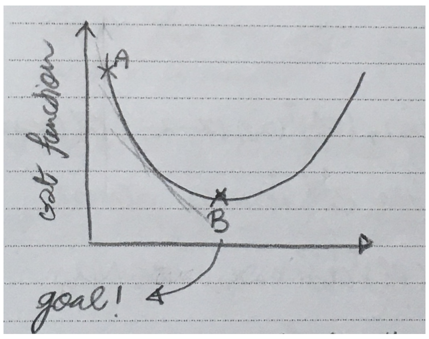
Inference is a conclusion reached based on evidance and reasoning. For instance, when we see someone running while wearing a formal cloth, we infere he/she is late for a meeting. Another example is that a detective observes blood stains on the floor leading to the back door, while there are no bodies in the room. He inferes that someone was killed or wounded in this room, but was carried out of it through the back door. On the other hand, since each phenomena in the nature can be represented by random variables, they are probabilistic. Meaning they occur according to a specific pattern depicted by their probability density function. If we narrow our view to statistical inference and probability theory, these two are dual of one another. In probability theory, PDF of random variables are known and various features are extracted from it. On the other hand, in statistical inference, the goal is to find a suitable distribution for given data.

Chapter 2

Question 2

2.1 Part A

Gradient descent is an optimization algorithm by which the values of parameters (coefficients) minimizing a cost function is found. To do so, it moves iteratively in the direction of steepest descent or in better words, negative of gradient.
It is best used in cases where parameters cannot be found analytically, such as using linear algebra, and they must be searched by an optimization algorithm.
How it works? A random point like A in the below plot is selected. This point represents the cost of current values of coefficients. The goal is to continue to try different values for the coefficients, evaluate theri cost and select new coefficients that have a slightly better and lower cost.



Repeating the stated procedure will result in reaching the minimum value. So first an initial value for the algorithm is chosen. Then its cost is evaluated. Next, the derivative is important so that by its sign we understand which direction we need to procede to get a lower cost in the next iteration. In the next step, a learning rate parameter is defined to control how much the coefficients change in each iteration.

$$coeff \rightarrow coeff - (learningrate * derivationofcost)$$

Writing Taylor series near x_0:

$$f(x) = f(x_0) \ + \ (\nabla f)^T(x_0)(x - x_0) \ + \ O(||x - x_0||^2)$$

Where based on the metioned assumption, the third term is almost zero. Therefore:

$$f(x) = f(x_0) \ + \ (\nabla f)^T(x_0)(x - x_0) \ \text{ for all } x \text{ near } x_0$$

Now in order to minimize $f(.)$ near x_0, the second term needs to be as small as possible. For this goal to come true, the value of $(\nabla f)^T(x_0)(x-x_0)$ needs to be minimized. This task is done when $(\nabla f)^T(x_0)$ and $(x-x_0)$ be in the exact opposite direction of one another. Mathematically speaking:

$$-\eta(\nabla f)^T(x_0) = (x-x_0)$$

where η is a positive number representing the step size. In order to find its value, we use the above equations to minimize $f(x_0-\eta\nabla f(x_0))$:

$$\frac{\partial}{\partial\eta} f(x_0 \ - \ \eta\nabla f(x_0)) = 0 \rightarrow (\nabla f)^T(x_0-\eta\nabla f(x_0))(\nabla f)(x_0) = 0 \rightarrow \nabla f(x) \perp \nabla f(x_0)$$

Therefore, the iterative moves are perpendicular to one another.

$$\begin{aligned}
f(x) = f(x_0) \ + \ (\nabla f)^T(x_0)(x-x_0) \ + \ \frac{1}{2}(x-x_0)^T H_f(x_0)(x-x_0)\ , \ x-x_0 = -\eta(\nabla f)(x_0) \\
\rightarrow f(x) = f(x_0) \ + \ (\nabla f)^T(x_0)(-\eta\nabla f(x_0)) \ + \ \frac{1}{2}(-\eta\nabla f(x_0))^T H_f(x_0)(-\eta(\nabla f)(x_0)) \\
= f(x_0) \ -\eta||(\nabla f)(x_0)||^2 \ + \ \frac{1}{2}\eta^2(\nabla f)^T(x_0)H_f(x_0)\nabla f(x_0) \\
\rightarrow \frac{\partial}{\partial\eta} f(x_0) = -||(\nabla f)(x_0)||^2 \ + \ \eta(\nabla f)^T(x_0)H_f(x_0)\nabla f(x_0) = 0 \\
\rightarrow \eta^{opt} = \frac{||(\nabla f)(x_0)||^2}{(\nabla f)^T(x_0)H_f(x_0)\nabla f(x_0)}
\end{aligned}$$

Therefore, as can be seen, the algorithm is compatible for many cases and in each iteration, the results do not get worsen. Although in cases near minimum, if the second degree approximation is used, convergance do not occur in a step. In fact, stochastic gradient is a slow algorithm for finding the optimum point.

Problem:

$$\begin{aligned}
J(\theta) = \frac{1}{2}\sum_{i=1}^{q}[h_\theta(x^{(i)}) - y^{(i)}]^2,\ h(x) = tanh(w^Tx+b) \\
\rightarrow J(w,b) = \frac{1}{2}\sum_{i=1}^{q}[tanh(w^Tx^{(i)})+b) - y^{(i)}]^2
\end{aligned}$$

So to find the SD formulation, partial derivation of w and b is required:

$$\begin{aligned}
J(w_j,\ b_j) = \frac{1}{2}\sum_{i=1}^{q}(tanh(w_j^Tx_j^{(i)} \ + \ b_j) - y_j^{(i)})^2 \\
\rightarrow \frac{\partial}{\partial w_j} J(w_j,\ b_j) = \sum_{i=1}^{q}[tanh(w_j^Tx_j^{(i)} \ + \ b_j) - y_j^{(i)}][1-tanh^2(w_j^Tx_j^{(i)} \ + \ b_j)][x_j^{(i)}] \\
\rightarrow \frac{\partial}{\partial w} J(w,\ b) = \sum_{i=1}^{q}[tanh(w^Tx^{(i)} \ + \ b) - y^{(i)}][1-tanh^2(w^Tx^{(i)} \ + \ b)][x^{(i)}]
\end{aligned}$$

By employing the same procedure on b_j, we get:

$$\rightarrow \frac{\partial}{\partial b_j} J(w_j,\ b_j) = \sum_{i=1}^{q} [tanh(w_j^T x_j^{(i)}\ +\ b_j) - y_j^{(i)}][1 - tanh^2(w_j^T x_j^{(i)}\ +\ b_j)]$$

$$\rightarrow \frac{\partial}{\partial b} J(w,\ b) = \sum_{i=1}^{q} [tanh(w^T x^{(i)}\ +\ b) - y^{(i)}][1 - tanh^2(w^T x^{(i)}\ +\ b)]$$

Using the gradients, the weights and bias is updated as below:

$$w_j := w_j - \alpha \sum_{i=1}^{q} [tanh(w_j^T x_j^{(i)}\ +\ b_j) - y_j^{(i)}][1 - tanh^2(w_j^T x_j^{(i)}\ +\ b_j)][x_j^{(i)}]$$

$$b_j := b_j - \alpha \sum_{i=1}^{q} [tanh(w_j^T x_j^{(i)}\ +\ b_j) - y_j^{(i)}][1 - tanh^2(w_j^T x_j^{(i)}\ +\ b_j)]$$

2.2 Part B

The rules regard updating each element of weights and biases were discussed in the last part. Now using them, these rules are formulated below in the vector form:

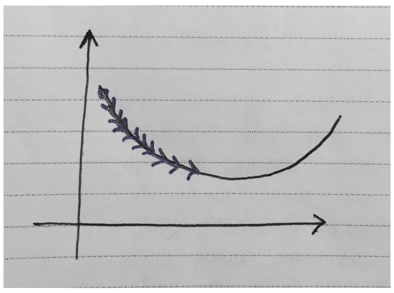
$$w := w - \alpha \sum_{i=1}^{q} [tanh(w^T x^{(i)}\ +\ b) - y^{(i)}][1 - tanh^2(w^T x^{(i)}\ +\ b)][x^{(i)}]$$

$$b := b - \alpha \sum_{i=1}^{q} [tanh(w^T x^{(i)}\ +\ b) - y^{(i)}][1 - tanh^2(w^T x^{(i)}\ +\ b)]$$

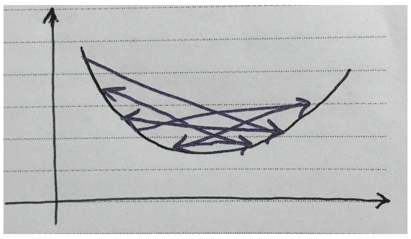
As was discussed,the learning rate controls how much to change the model in response to the estimated error each time model's parameters are updated. In other words, this parameter defines how fast or slow we will moce toward the optimal value of the model's parameters.

- **Small Learning Rate**: A smaller learning rate allows the model to learn a more optimal or even globally optimal set of parameters but ut requires more epochs to train the model to do so. Therefore, it can be an adverse effect in times where taking long time to learn can be costly.

- **Too Small Learning Rate**: Too small learning rates are not only the cause for slow learn, but they also may stuck the model with high training error. They may not even have the chance to convarge or end up to a sub-optimal solution.



- **Large Learning Rate** : A large learning rate allows the model to learn faster, however, since there is no free lunch, the cost is that the model may converge to a sub-optimal final set for its parameters.
- **Too Large Learning Rates** : A learning rate which is too large results in parameter updates that are too large. Hence, the performance of the model will oscillate during training epochs. In these cases, the gradient descent can increase the training error rather than decreasing it. It's also possible to encounter a positive feedback loop so that the large weights induce large gradients. This resutls in inducing a large update of weights as well. Continuing the same procedure will increase the size of weights and as a result parameters move away form the origin and overflow occurs.



Chapter 3

Question 3

Interpolation Theorem: Given $k+1$ distinct points $x_0,\ x_1,\ ...,\ x_{k+1}$ and corresponding values $y_0,\ y_1,\ ...,\ y_{k+1}$, there exists a unique polynomial of degree at most $k+1$ that interpolates the data $\{(x_0, y_0),\ ...,\ (x_{k+1}, y_{k+1})\}$.
As the above theorm suggests, in order the problem is asking to prove the "Interpolation Theorem". To do so, first we need to consider the Lagrange function as below:

$$L_j(x) = \prod_{i \neq j,\ i=1}^{k+1} \frac{x - x_i}{x_j - x_i}$$

It is crystal clear that L is a polynomial of degree k+1 and in x_i we have:

$$L_j(x_i) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

Therefore, it is obvious that $L_i(x)$ are linearly independant, so the below linear combination can interpolate a polynomial of degree k+1.

$$f(x) = \sum_{j=0}^{k+1} [y_i L_j(x)]$$

In order to prove that it is unique, assume there exists another polynomial g of degree at most $k+1$. Based on what has been discussed, $f(x_i) = g(x_i)$ for all $i = 0, ..., k+1$. Thus, the polynomial $f - g$ has $k+2$ unique zeros. However, based on the fundamental theorem of algebra, since $f - g$ is of degree k+1, its zeros are at most k+1. Therefore, $f = g$.

Chapter 4

Question 4

For linear regression in matrix form we have:

$$X = [\underline{x}^{(1)} \; ... \; \underline{x}^{(n)}],\ \underline{x}^{(i)} = [1 \; x_1^i \; ... \; x_d^i]^T,\ \underline{y} = [y^1 \; ... \; y^n]^T$$
$$\rightarrow J(\beta) = \frac{1}{2}(X\beta - \underline{y})^T(X\beta - \underline{y})$$
$$\rightarrow \nabla_\beta J(\beta) = \frac{1}{2}\{\beta(X^TX)\beta - 2(X^T\underline{y})\beta = X^TX\beta - X^T\underline{y} = 0$$
$$\rightarrow \beta = (X^TX)^{-1}X^T\underline{y}$$

Now if a noise is considered:

$$y = X^T\beta \;+\; \epsilon$$

In order to calculate the covariance of β we need its expected value, So:

$$E\{\beta\} = E\{(X^TX)^{-1}X\underline{y}\} = E\{(X^TX)^{-1}X^T(X\beta + \epsilon)\}$$

, supposing noise is zero mean so that it does not add bias $\rightarrow E\{\beta\} = (X^TX)^{-1}X^TX\beta$

Now based on the calculated expected value we have:

$$Cov(\beta) = E\{(\beta - \bar{\beta})(\beta - \bar{\beta})^T\}$$
$$= E\{[(X^TX)^{-1}X^T(X\beta + \epsilon) - (X^TX)^{-1}X^TX\beta][(X^TX)^{-1}X^T(X\beta + \epsilon) - (X^TX)^{-1}X^TX\beta]^T\}$$
$$= \{(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1}\} = (X^TX)^{-1}X^TE\{\epsilon\epsilon^T\}(X^TX)^{-1}X^T = (X^TX)^{-1}X^TVar(\epsilon)X(X^TX)^{-1}$$

Now if we suppose ϵ is a vector of zero mean, independent, normal elements, the results can be simplified as below:

$$Cov(\beta) = (X^TX)^{-1}X^TVar(\epsilon)X(X^TX)^{-1} = (X^TX)^{-1}X^T\sigma^2IX(X^TX)^{-1} = \sigma^2(X^TX)^{-1}$$

Based on the fact that we have two β values, we get:

$$X = \begin{bmatrix} 1 & x_1 \\ . & . \\ . & . \\ . & . \\ 1 & x_d \end{bmatrix} \rightarrow X^TX = \begin{bmatrix} d & \sum_{i=1}^d x_i \\ \sum_{i=1}^d x_i & \sum_{i=1}^d x_i^2 \end{bmatrix} \rightarrow Cov(\beta) = \frac{\sigma^2}{d\sum_{i=1}^d x_i^2 - (\sum_{i=1}^d x_i)^2} \begin{bmatrix} \sum_{i=1}^d x_i^2 & -\sum_{i=1}^d x_i \\ -\sum_{i=1}^d x_i & d \end{bmatrix}$$

For two random varaibles to be independent, the coveriance matrix needs to be dialgonal. Therefore we get:

$$\sum_{i=1}^{d} x_i = 0$$

This means data samples need to cancel each other out and be moslty desperesed near zero.

Chapter 5

Question 5

5.1 Part A

First of we need to calculate the variance of noise. In order to do that, we need to calculate sample variance of y: (Since based on the given relationship of these to variables, their varience is the same.)

$$Var\{\epsilon\} = Var\{y\} = E\{(y - \bar{y})^2\},$$
$$E\{y\} = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = \frac{1}{8}(40 + 41 + 42 * 3 + 43 * 2 + 44) = 42.125$$
$$\rightarrow Var\{\epsilon\} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 = \frac{1}{8}[(40 - 42.125)^2 + (41 - 42.125)^2 + 3 * (42 - 42.125)^2 +$$
$$2 * (42 - 42.125)^2 + (44 - 42.125)^2] = 1.36$$

Based on β's calclated value in notes as well as last question, its vetor, which is made up of β_0 and β_1 are evaluated:

$$\beta = (X^T X)^{-1} X^T y$$
$$X = \begin{bmatrix} 1 & 0.5 \\ 1 & 1 \\ 1 & 1.5 \\ 1 & 2 \\ 1 & 2.5 \\ 1 & 3 \\ 1 & 3.5 \\ 1 & 4 \end{bmatrix} \rightarrow X^T X = \begin{bmatrix} 8 & 18 \\ 18 & 51 \end{bmatrix} \rightarrow (X^T X)^{-1} = \begin{bmatrix} 0.6071 & -0.2143 \\ -0.2143 & 0.0952 \end{bmatrix}$$
$$\rightarrow (X^T X)^{-1} X^T y = \begin{bmatrix} 40.8929 \\ 0.5476 \end{bmatrix} \rightarrow \beta_0 = 40.8929,\ \beta_1 = 0.5476$$

Now in order to compute the variance of β_0 and β_1, we need to use the formula found in problem 4, for β's covariance matrix. In addition, σ^2 is needed, which is also computed in the begining of this problem.

$$cov\{\beta\} = \sigma^2 (X^T X)^{-1} = 1.36 \begin{bmatrix} 0.6071 & -0.2143 \\ -0.2143 & 0.0952 \end{bmatrix} = \begin{bmatrix} 0.8257 & -0.2914 \\ -0.2914 & 0.1295 \end{bmatrix}$$
$$\rightarrow var\{\beta_0\} = 0.8257,\ var\{\beta_1\} = 0.1295$$

5.2 Part B

Correlation is defined as below:

$$r = corr\{\beta_0,\ \beta_1\} = \frac{Cov\{\beta_0,\ \beta_1\}}{\sqrt{var\{\beta_0\}var\{\beta_1\}}} = \frac{-0.2914}{\sqrt{0.1295 * 0.8257}} = -0.9038$$

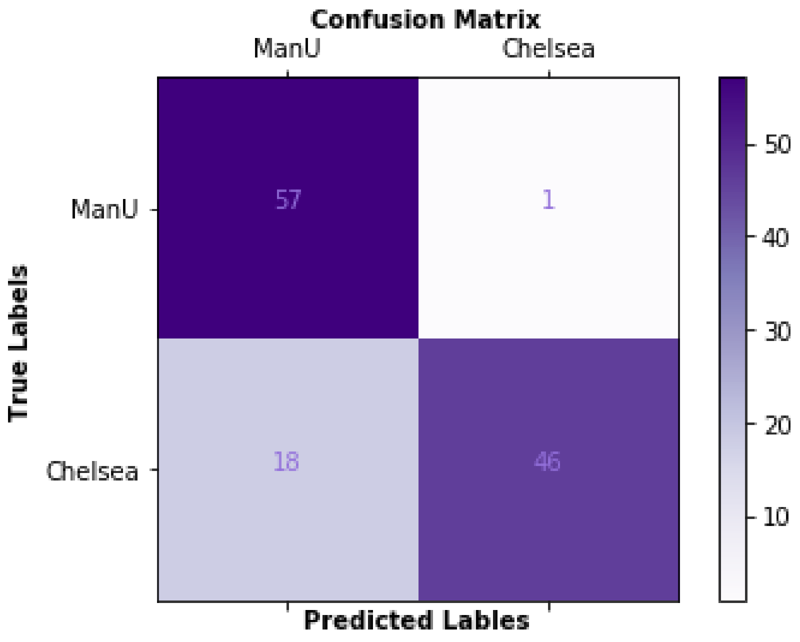
Chapter 6

Question 6

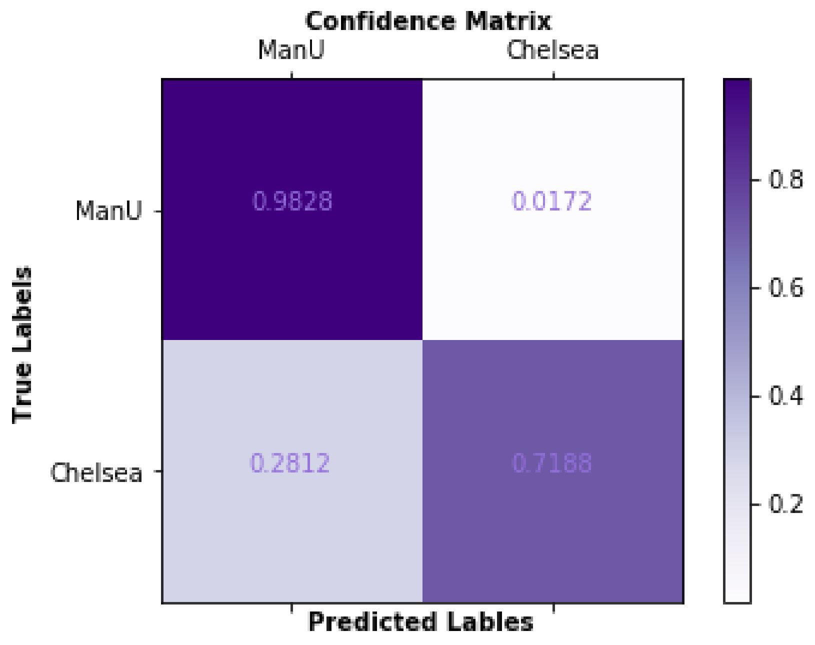
In order to classify each picture to "Manchester United" and "Chelsea", two methods are used.
In the method explained under the first section, no libraries are employed as well as no labels are used for training. Owing to the fact that the value of "Red" and "Blue" are clear, no training is needed in this part.
In the second method, two classifiers are used. One of which is KNN and the other is Naive Gaussian. Using train and test datasets, they are trained and then tested.

6.1 Method 1

In this part, the classifier is created based on the average value of RGB in each input picture. The distance between the value of red to the refrence is calculated. The same procedure is carried out on blue as well. Afterwards, based on which distance is minimum, the picture is classified. By doing so, the confusion matrix can be represented as below:

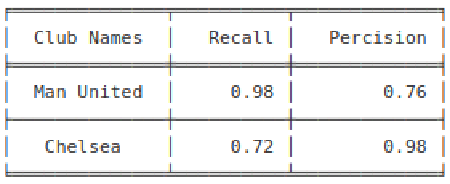


By normalizing this matrix in columns, we get the confidence matrix, which is depicted below:



Based on the confusion matrix, accuracy, percision, and recall values are calculated as well. In addition, CRR in computed. The results of these computations are shown below.

```
* Correct classification rate after employing this classifier is: 0.844
* Accuracy score after employing this classifier is: 0.844
```



Club Names	Recall	Percision
Man United	0.98	0.76
Chelsea	0.72	0.98

- **Accuracy and Correct Classification Rate**:
  As the above results suggest, the accuracy of this classifier, which is also its correct classification rate, is 84.4%. This means that among the total number of predictions by this classifier, 84.4% is correct.

- **Recall**:
  Based on the above table, recall values are 98% and 72% for Manchester United and Chelsea respectively. On the grounds of the fact that recall is the measure of the model's accuracy in identifying true positive, among all the times the picture is Manchester United, in 98% of the times, Manchester United is the classifiers's prediction as well. This value is 72% of the times for Chelsea.

- **Percision**:
  Based on the above table, percision values are 76% and 98% for Manchester United and Chelsea respectively. Owing to the fact that precision shows the

ratio between true positive and all positives, among all the times this classifier predicts Manchester United as a label, 76% of the times Manchester United is the true label as well. This value is 98% for Chelsea.

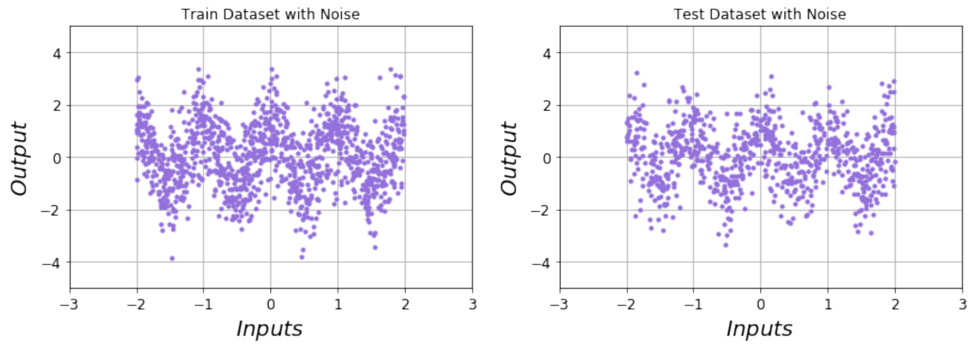
6.2 Method 2

In this method, by using the "sklearn" library, KNN classifier for $k = 5, 50$, and Naive Gaussian classifier are created. To do so, after the images are read, their average RGB is calculated. Next, each picture's labels are read and saved. As to create train and test datasets, the all the saved information is separated into two datasets. Among all data, 80% is used as training data and label, while the rest is used as test data and label. Afterward, the training datas are emplyed to train the model and then by using the test dataset, confusion matrix is computed. Using the confusion matrix, confidence matrix, recall, precision, and accuracy are evaluated as well.

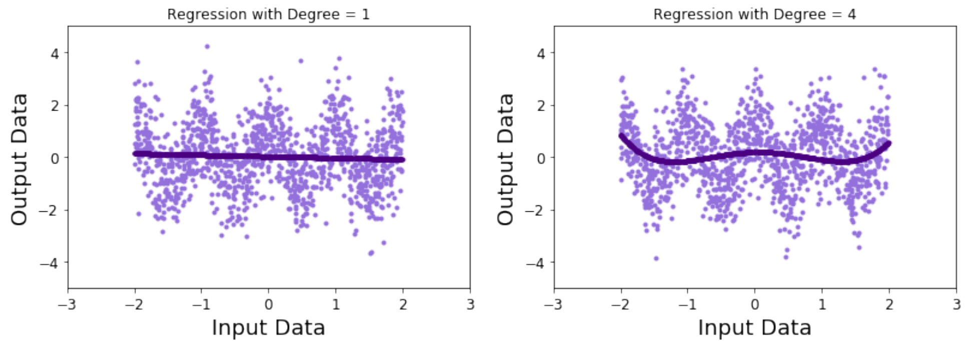
Chapter 7

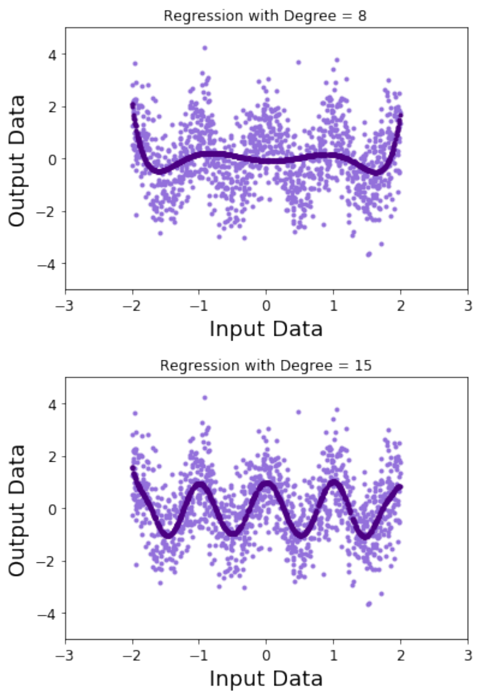
Question 7

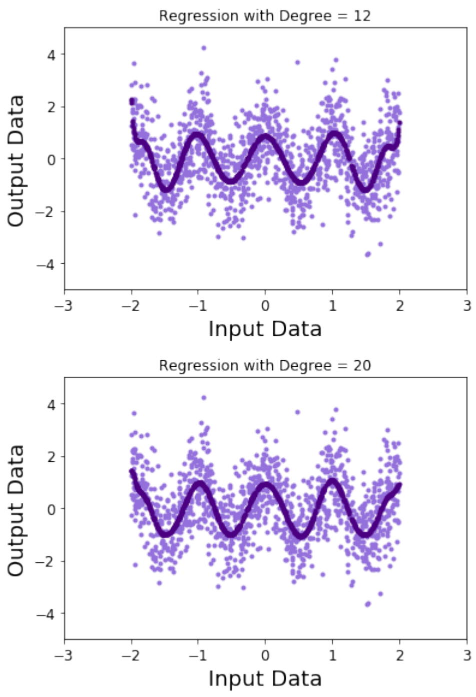
As the first step, a random data between $[-2, 2]$ is created as the input of cosine function. Afterward, the result is sumed up with zero mean, unit variance, normal noise.
In order to train the models, the simulated data is separated between test and training datasets. The separation is done randomly with 60% data for training and the others for testing. The resluts can be depicted as below:



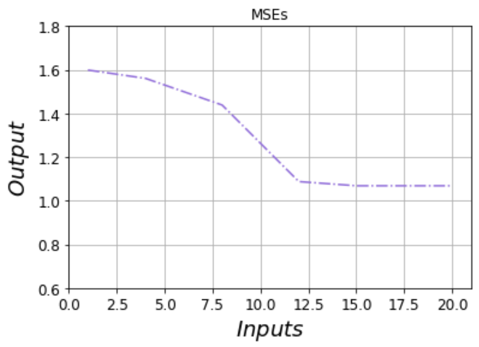
Now, for differnt degrees, the model is trained to fit the train data. Afterward, the trained model is tested with the test data. The result can be depected as below:

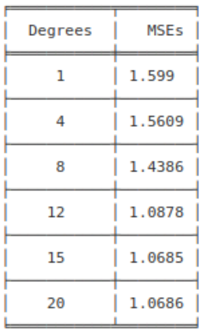






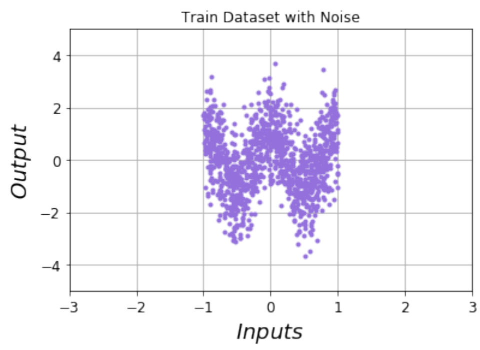
As can be seen, due to the numorous ups and downs of cosine function in the given interval, being fit requires high polynomial degree. This matter is proven by how MSE behaves as well, which is depicted as below:

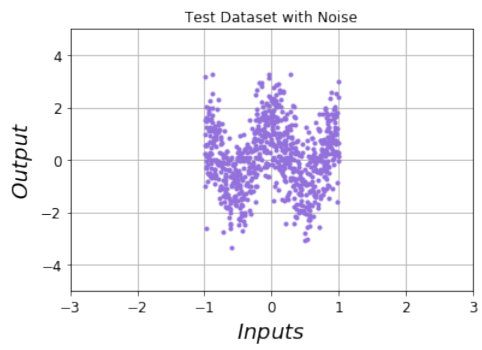


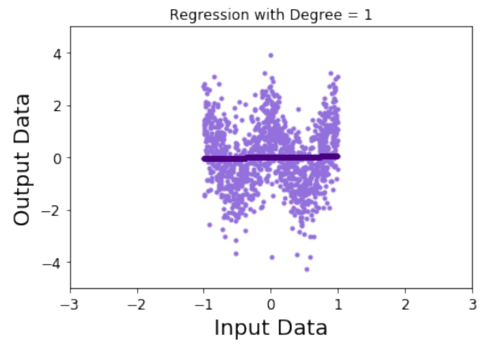


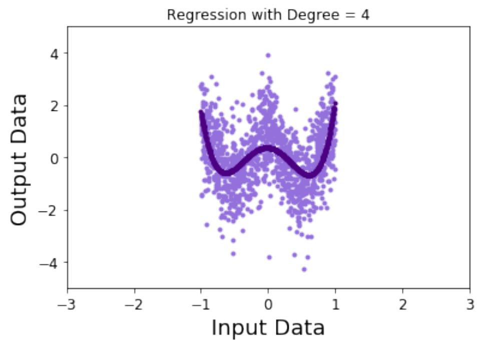
Degrees	MSEs
1	1.599
4	1.5609
8	1.4386
12	1.0878
15	1.0685
20	1.0686

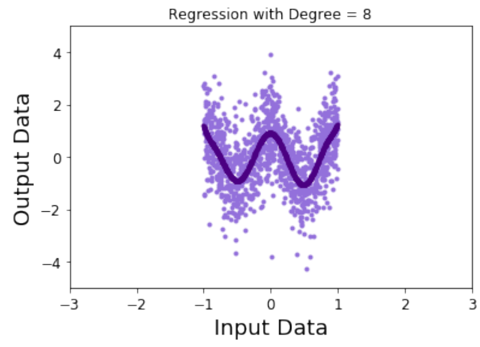
If we decrease the number of ups and downs, as depicted in the below plots, the degree fitting the noisy cosines, decreases too. This fact is obvious in plots as well as MSE's behavior. Therefore, people can fit different degrees suitable and others.

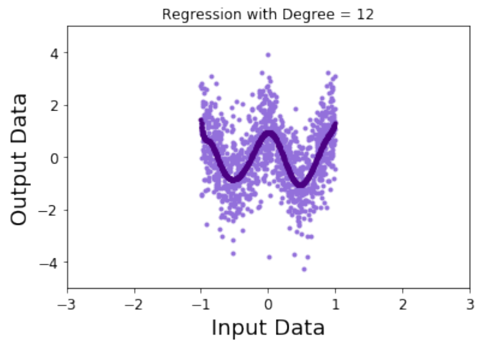


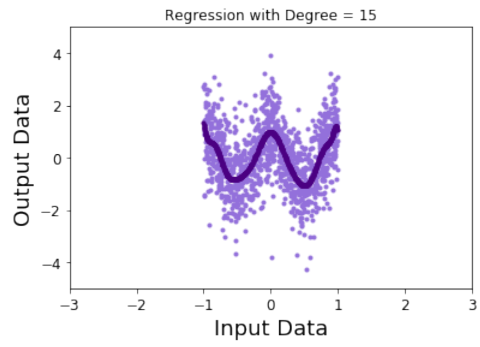


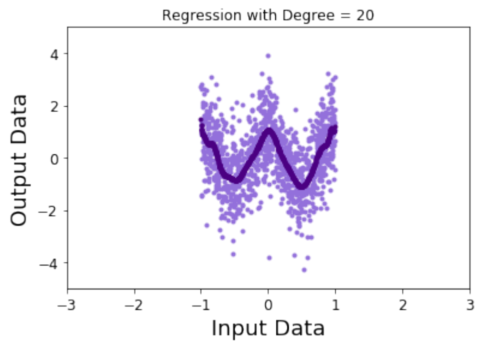


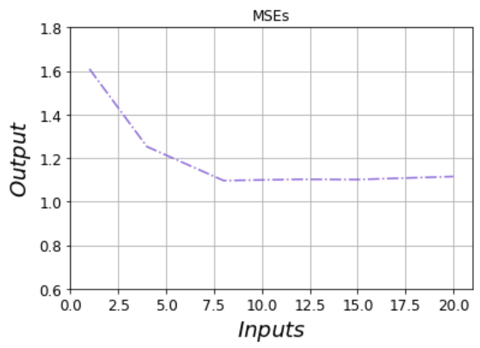


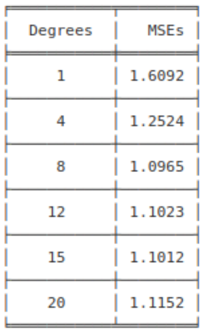












Degrees	MSEs
1	1.6092
4	1.2524
8	1.0965
12	1.1023
15	1.1012
20	1.1152

Now, for calcuating the value of bias and variance, we need to train the model in different epochs and average over them to reach the desired goal. For simulation, 500 train data, 1000 test data, and 500 epochs are set.
In each epoch, new generated data is attached to the data from previous ephoc. Then, using the same procedure for fitting, datas are fitted and labels are predicted. Now, in order to compute variance and bias, formulations from Bishop source book is used.
In Bishop, for computing the expected squared loss the below formulation is suggested:

$$expected\ loss = (bias)^2\ +\ variance\ +\ noise$$

where:

$$(bias)^2 = \int \{\mathbb{E}_D[y(\mathbf{x} : D)] - h(\mathbf{x})\}^2 p(\mathbf{x}) dx$$

$$variance = \int \mathbb{E}_D\{\{y(\mathbf{x} : D)] - h(\mathbf{x})\}\}^2 p(\mathbf{x}) dx$$

$$noise = \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x}\, dt$$

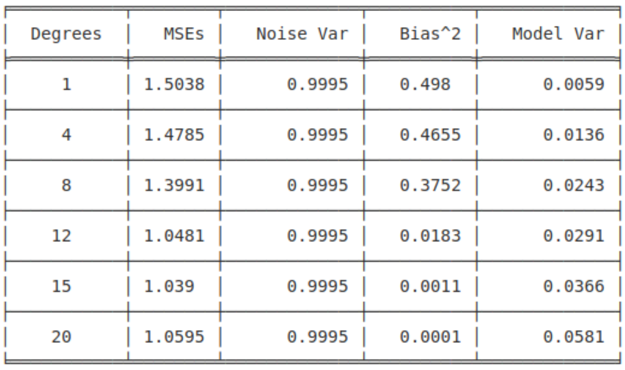
Now for the prediction case we have:

$$(bias)^2 = \frac{1}{N} \sum_{n=1}^{N} \{\bar{y}(x_n) - h(x_n)\}^2 dx$$

$$variance = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{L} \sum_{l=1}^{L} \{y^{(l)}(x_n) - \bar{y}(x_n)\}^2 dx$$

$$\bar{y} = \frac{1}{L} \sum_{l=1}^{L} y^{(l)}(x)$$

Using the above formulations for updating bias, variance, MSE, and noise value in each iteration we get:



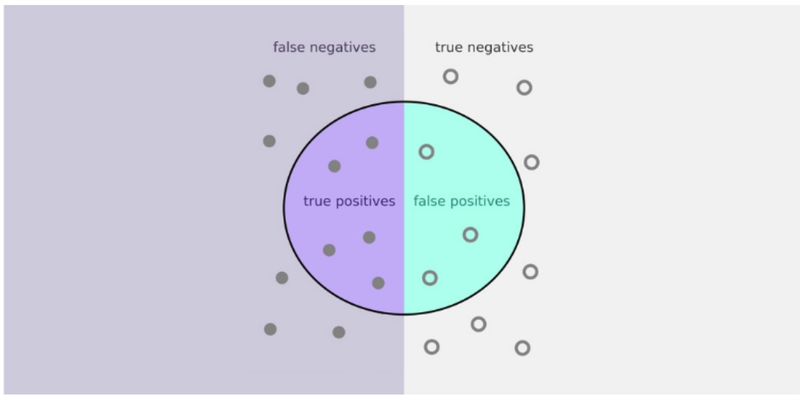
Degrees	MSEs	Noise Var	Bias^2	Model Var
1	1.5038	0.9995	0.498	0.0059
4	1.4785	0.9995	0.4655	0.0136
8	1.3991	0.9995	0.3752	0.0243
12	1.0481	0.9995	0.0183	0.0291
15	1.039	0.9995	0.0011	0.0366
20	1.0595	0.9995	0.0001	0.0581

It is crystal clear that as the model's degree increases, bias reache zero. Meaning the estimator of the true model moves toward being an unbiased estimator. Therefore, variance's value increases to compensate for the decreament of bias. The only boundry for the increment of variance, is the noise's variance's (modeled by the training labels) presence. Thus, the above table represents the behavior of the model as was expected and conveyed in the given formula.

Chapter 8

Question 8

In order to explain these concepts, the below picture can be helpful.



In addition, it is supposed that we are dealing with some pictures from space, and want to identify whether they are planets or stars. So that for the planet ones we come up with a plan to figure out whether there is life on it or not.

- **Recall**: Recall is the measure of our model correctly identifying **True Positives**. Thus, for all the pictures which were planets, recall tells us how many we correctly identified as planets. Mathematically speaking:

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP)\ +\ FalseNegative(FN)}$$

  Recall also gives a measurement of how accurately our model is able to identify the relevant data. It is refered as **Sensitivity** or **True Positive Rate**. It is analogous to a picture be a picture of planet but be detected otherwise due to the error of the model. Since we do not want to waste money and time, it is our goal to minimize this erronous detections. Therefore, unfortunatelly, it does not give an insight about how reliable our model is performing.

- **Precision**: Precision is the ratio between the **True Positives** and all the Positives. For our problem statement, that would be the measure of pictures that we correctly identify as planets out of all the pictures actually made up of it. Mathematically speaking:

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP)\ +\ FalsePositive(FP)}$$

  Precision also gives us a measure of the relevant data points. It is essential that we don't start spending all our time and sources on searching for life on an object which is a star, but our model predicted as a planet. However, it does not give us the vision to grasp how accurate our model is performing. In fact, as a comparision between precision and accuracy suggests, these two are dual of one another.

- **Accuracy**: Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

$$Accuracy = \frac{TruePositive(TP)\ +\ TrueNegative(TN)}{TruePositive(TP)\ +\ FalsePositive(FP)\ +\ FalseNegative(FN)\ +\ FalsePositive(FP)}$$

Using accuracy as a defining metric for our model does make sense intuitively, but more often than not, it is always advisable to use Precision and Recall too. There might be other situations where our accuracy is very high, but our precision or recall is low. Ideally, for our model, we would like to completely avoid any situations where the picture belongs to a star, but our model classifies the object as a planet it i.e., aim for high recall.

On the other hand, for the cases where the picture is not a star and our model predicts the opposite, we would also like to avoid losing chance of finding aliens.

Although we do aim for high precision and high recall value, achieving both at the same time is not possible. For example, if we change the model to one giving us a high recall, we might detect all the pictures who actually are planets, but we might end up wasting time and resources on stars as well.

Similarly, if we aim for high precision to avoid giving any wrong and unrequired effort, we end up getting a lot of pictures which were actually planets, but did not receive a chance to be ckecked.