TED UNIVERSITY

Machine Learning Assignment 1 Report

Arda ANDIRIN

17921934994

Q1)

I) In the first part of the question 1, I have applied polynomial regression on a randomly generated data. The degree of the polynomial model are as follows; 0,1,3,9. Our m, the size of the training data is 10 in this part. I ran the code 3 times to see if the algorithm was working correctly.

When d=0;

We have a simple straight line because the degree of x is 0 which is 1.

When d=1;

This is linear regression. The prediction fit is now a sloped line. However, it is not a good model.

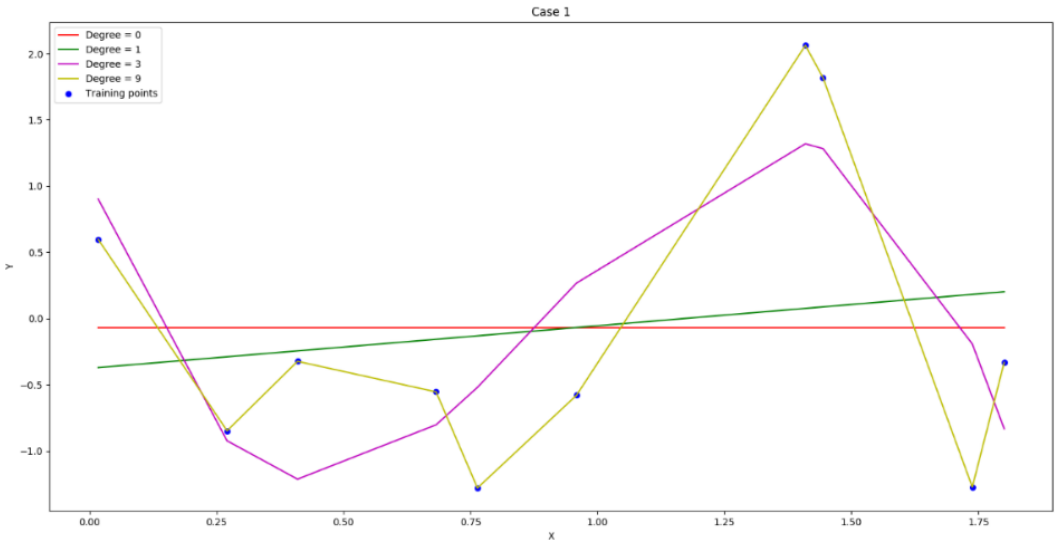
When d=3;

At this point our curve fits better to our training data, the degree of the polynomial is 3.

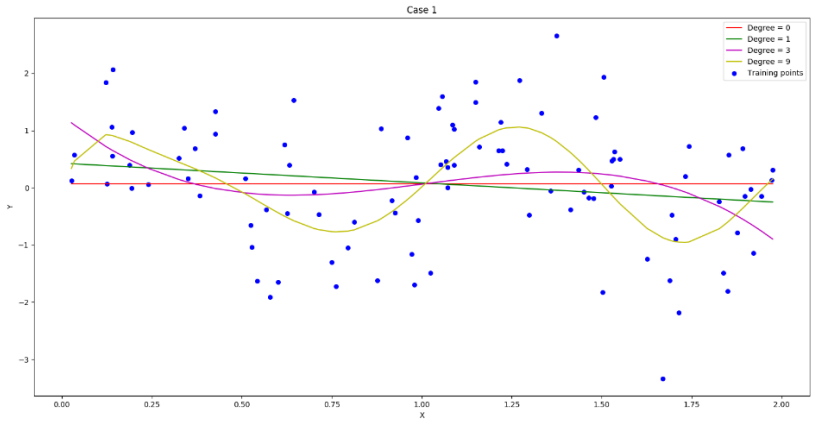
When d=9;

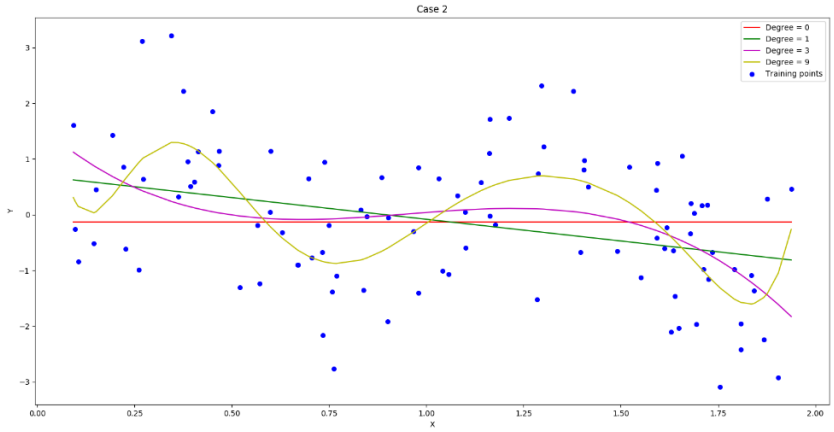
The order of the polynomial function is 9. It is as much as our training data.

We can see from the graphs that as the degree of the function increases, the curve is more alike to the training points. When the degree of the function is too low, we can observe underfitting. Because the lines don't mean anything. But for example, when the degree is 3 our model predicts the labels better. As the degree increases, we can observe overfitting. In the example of degree of 9 we have 10 labels. Our model learned the labels so well and so precise that when it tries to find y given an x point it matches exactly. However, if we were to give it an unknown test data different from the training data, we would observe that it won't predict very well because of overfitting.



II) In the second part of the question 1, I have increased the number of training labels points to 100. Here are the results. We can see that there is not much change for the degrees 0 and 1. Now, degree 3 nor 9 can predict well. Our training labels are too large and scattered it can't make good fit.





Q2) In the second question, I generate X and Y values and using batch gradient descent I find a theta that is iterated 1000 times with a learning rate of 0.1. With each iteration theta becomes more and more precise. At first theta is some random value. Then it finds a better and better fit to the training data. At the end the change of theta is so low that it is time to stop. In our example we update the value of theta 1000 times.

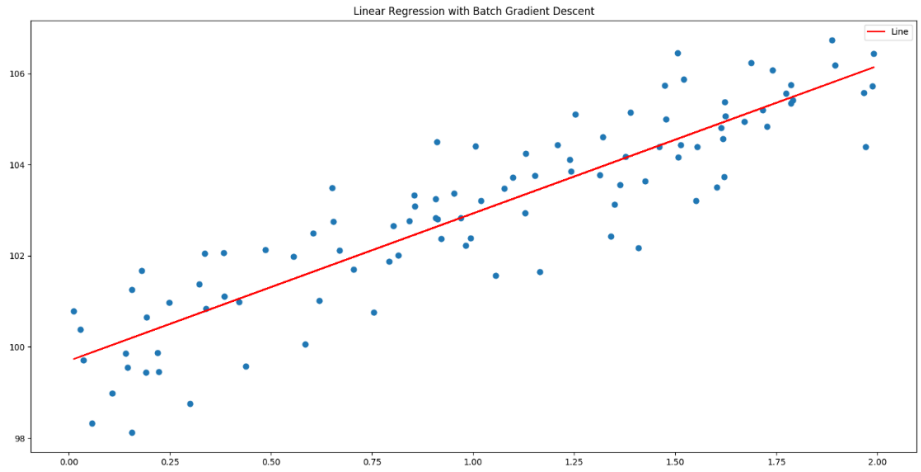
My function that was fitted to my data is

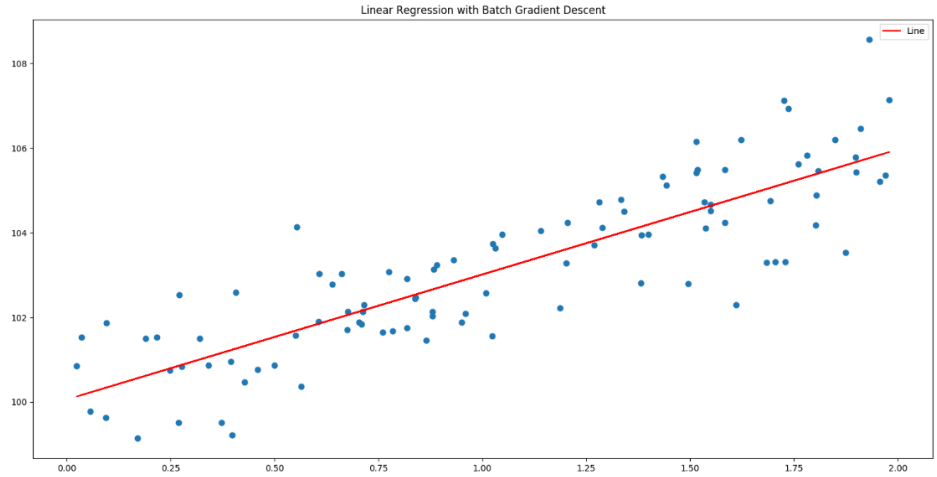
Gradient

$$\frac{\partial J(\theta)}{\partial \theta_j} = 1/m \sum_{i=1}^{m} (h(\theta^{(i)} - y^{(i)}).X_j^{(i)} \qquad \theta_j := \theta_j - \alpha.(1/m.\sum_{i=1}^{m} (h(\theta^{(i)} - y^{(i)}).X_0^{(i)})$$

y_hat in my code is h(theta)

We update the value of theta with each iteration.





Q3) In the 3rd question, we are expected to implement a locally weighted linear regression. It is basically same as the second question except this time I take weights into consideration.

For each training point I computed the weight of the function. If our bandwidth parameter tau is large such as 10, we should expect to see a straight line. But when the tau is small, such as 0.1, our line has some ups and downs. This is due to the closer points affecting our results. The algorithm will give a higher weight to the points closer to the point we are finding the thetas for. So, the predicted y value will depend on the location of those closer x values.

My weight function computes the weights correctly and I make a 100x2 matrix for each X value.

Weight is 100x2 matrix, X is 100x2 Matrix, theta is 2x1, Y_hat is dot product of X and theta matrices. 100x1 matrix, Y_hat is $x^{(I)}$*theta

When we take the derivative of the MSE function we get $w^{(I)}$*(y_hat – y) * $x^{(I)}$ this is for theta1.

If we take a holistic approach for the summation, we can use dot product of numpy to do all the computations in one go.

However, it seems impossible to get a 2x1 matrix for theta using this function. If we look at the sizes of the matrices we have (100x2) * (100x1) * (100x2). I couldn't get how to return a 2x1 matrix for theta.

So, I decided to use linalg method for the summation from scipy. When we take the derivative of the MSE w.r.t theta and equate it to zero we get the following;

$$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum w^{(i)} y^{(i)} \\ \sum w^{(i)} y^{(i)} x^{(i)} \end{bmatrix}$$

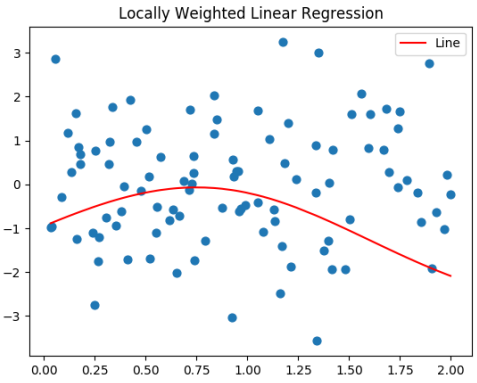


Figure 2 Tau = 1

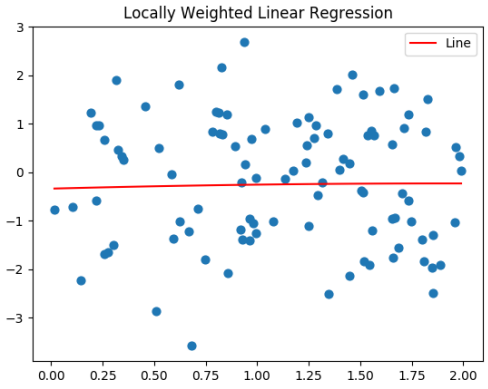


Figure 1 Tau = 10

Here are the results.

We can observe that as tau gets larger our line becomes flatter. The larger the bandwidth(tau) values the more weight will be given to more distant x values.