IST597 Deep Learning Foundations HW1

Guanjie Zheng

October 2, 2017

Note: there are more plots inside the folders that were used to tune the parameters. Due to the space limit, I did not show them in the report.

1 Problem 1

1.1 Data statistics and plot

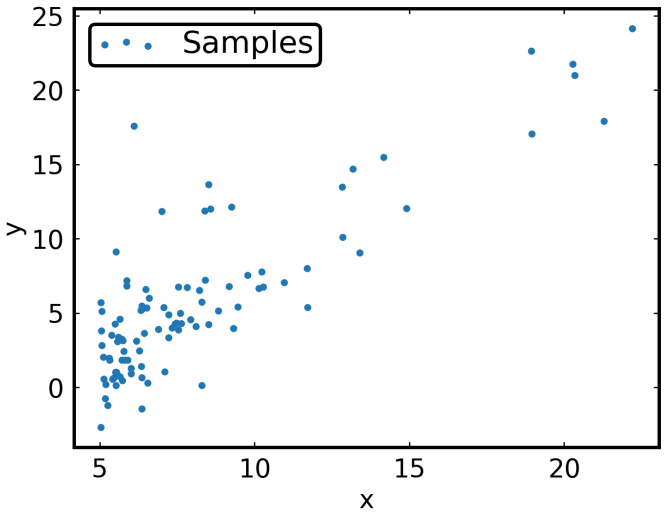

Figure 1: Data

1.2 Result

With the parameter as $alpha = 0.001$, $eps = 10^{-8}$, $n_{epoch} = 10000$, we can get the regression with final loss as 4.516. The learned parameters are

$$w = 1.127, b = -3.240 \tag{1}$$

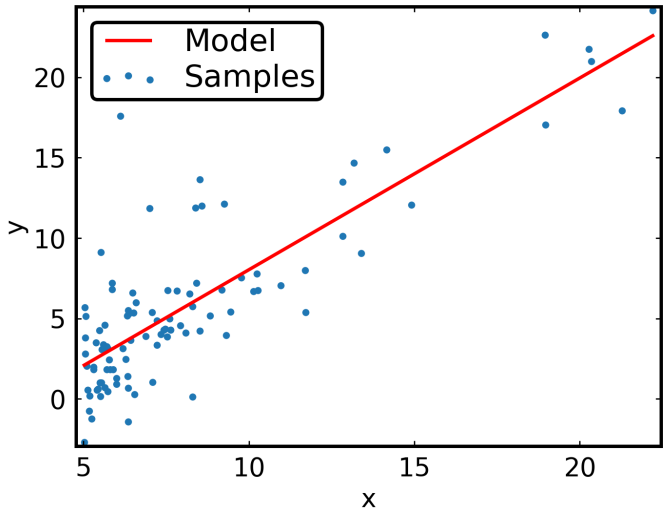

Figure 2: Fitting result v.s. data

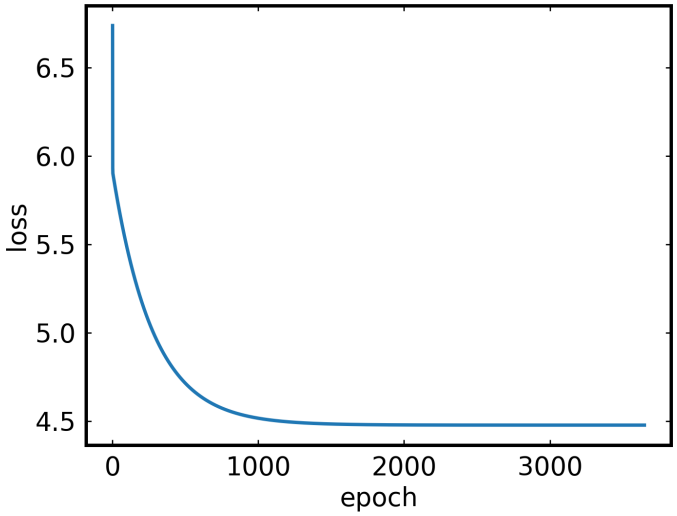

Figure 3: Loss v.s. epochs

Here is the process of tuning the parameter alpha.
Small learning rate will make the learning process very slow. For example,

Table 1: Loss for different learning rate alpha

α	loss
0.0001	5.48
0.001	4.51
0.01	4.47
0.02	4.47
0.05	nan
0.1	nan

when α is set to 0.0001, after 10,000 epochs, the loss is still decreasing. Larger learning rate will speed up the learning process, but may lead to unstable results. Usually, it takes several hundreds (500) epochs to converge to a reasonable solution.

2 Problem 2

2.1 Fitting Result

With the parameter as $alpha = 0.001$, $eps = 10^{-8}$, $n_{epoch} = 500000$, we can get the regression with the following results.

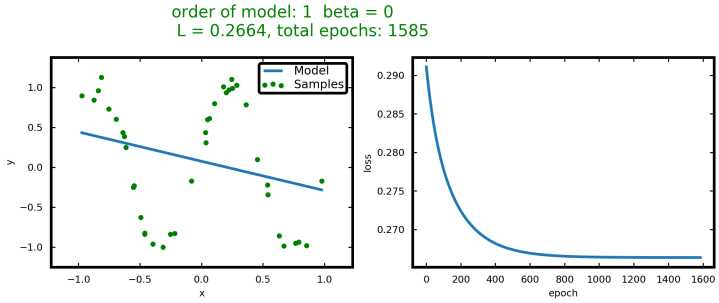

Figure 4: 1st order

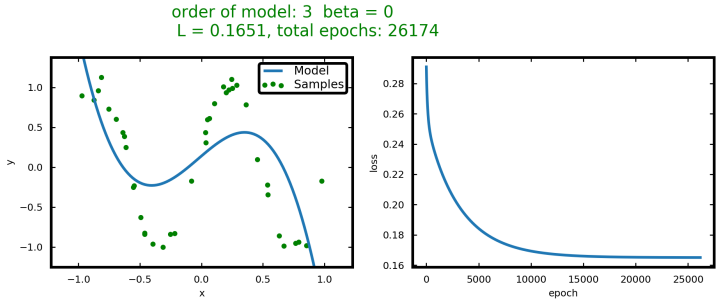

Figure 5: 3rd order

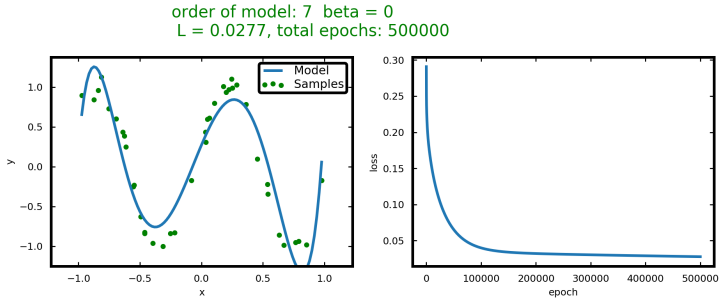

Figure 6: 7th order

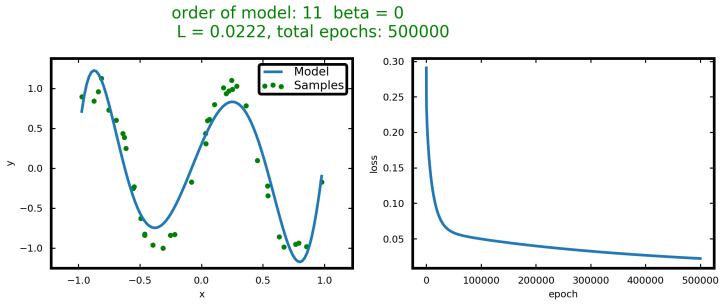

Figure 7: 11th order

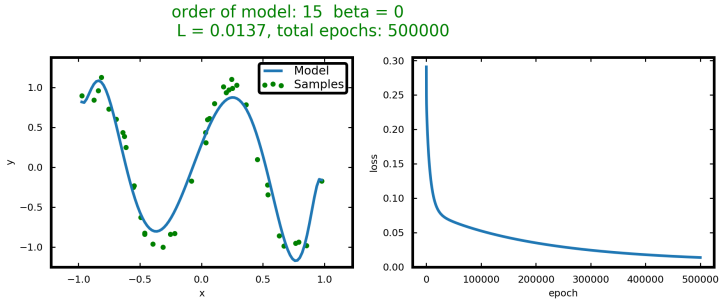

Figure 8: 15th order

- With respect to the feature mapping, what is a potential problem that the scheme we have designed above might create? What is one solution to fixing any potential problems created by using this scheme (and what other problems might that solution induce)?

  The created features (x, x^2, x^3, ...) are highly correlated. In this way, the coefficient of each feature will highly depend on what features are included in the model. (i.e., including x^2 will have strong influence on the coefficient of x). This is not the correct relationship between x and y that we want to model.

  One possible solution is to just feed one x input into the computation graph, and make the x, x^2, x^3 as a hidden layer. Then, when computing the gradient, the change in x will cause the change in all the hidden layers.

- What do you observe as the capacity of the model is increased? Why does this happen?

  As the capacity of the model increase, the model is able to fit the data better (the loss is decreasing). This is due to the reason that higher order polynomial can fit more details than lower order polynomial. If we push the order to infinity, this will become taylor series, which is able to fit any function form.

2.2 Tuning the regularization

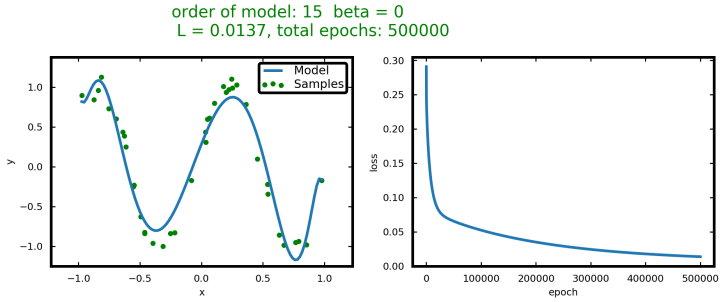

Figure 9: 15th order, $\beta = 0$

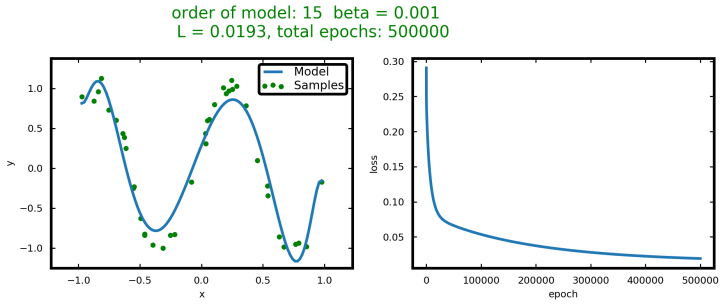

Figure 10: 15th order, $\beta = 0.001$

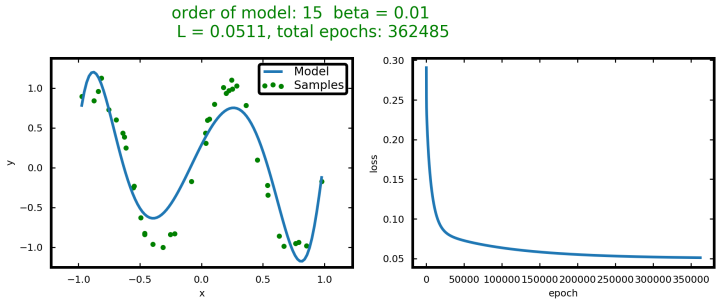

Figure 11: 15th order, $\beta = 0.01$

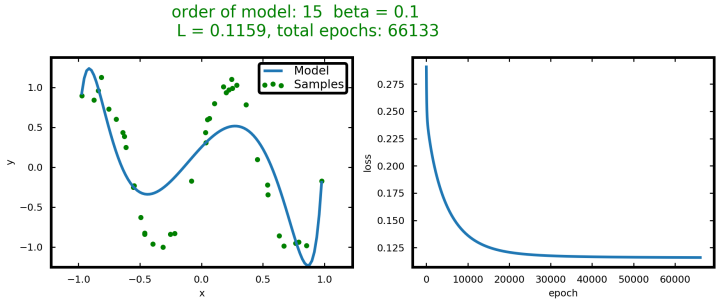

Figure 12: 15th order, $\beta = 0.1$

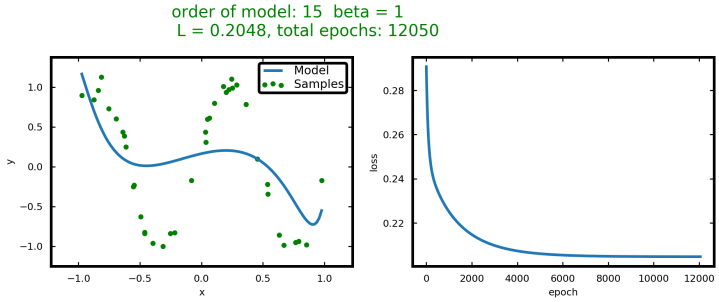

Figure 13: 15th order, $\beta = 1$

- What do you observe as you increase the value of β? How does this interact with the general model fitting process (such as the step size α and number of epochs needed to reach convergence)?

  As β increase, the model tends to be more general (have less detailed information). The general model fitting process becomes faster and will converge with fewer epochs.

- What might be a problem with a convergence check that compares the current cost with the previous cost (i.e., looks at the deltas between costs at time t and t 1), especially for a more complicated model? How can we fix this?

  I already added this convergence check in the code. If the function is not convex everywhere, then this convergence check might lead to a local optimum if the step size is not very big. Because the optimizer might get trapped in the local optimum for several epochs. We can wait for several epochs. Within these several epochs, if the loss decrease is always smaller than a threshold value, we stop the optimiation.

3 Problem 3

3.1 Fitting result without regularization

I tuned the α from [0.001, 0.01, 0.1] and n_{epoch} from [100000, 500000, 1000000]. But actually, with the help of early stopping criterion, we only needs the largest number of epochs. So I only show the result for $n_{epoch} = 1000000$, with varying α. Different learning rates do not make the results that different. The lowest error rate is 16.1%.

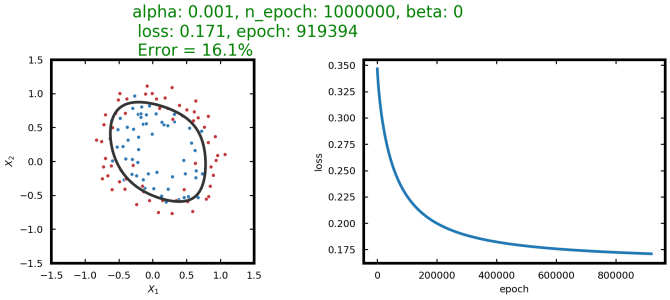

Figure 14: $\alpha = 0.001$, $epoch = 1000000$

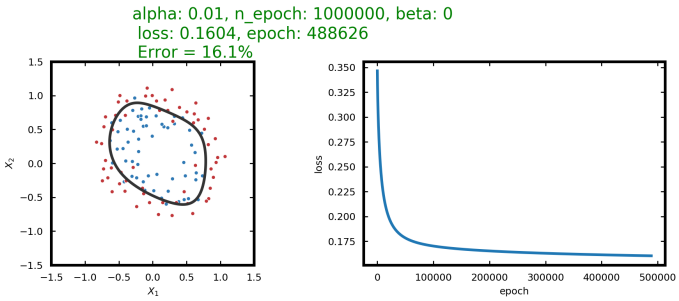

Figure 15: $\alpha = 0.01$, $epoch = 1000000$

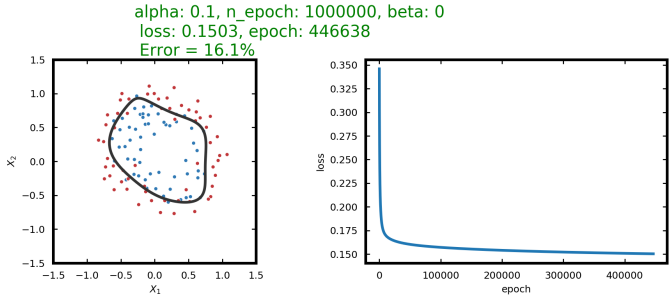

Figure 16: $\alpha = 0.1$, $epoch = 1000000$

3.2 Tuning the regularization

I further tune the β to change the regularization strength. With some proper strength of regularization (when $\beta \leq 0.1$), the classification accuracy stays the same. When larger regularization is added, the accuracy starts to decrease. With increasing regularization, the decision boundary tends to be more smooth.

Regularizing model will make the model more general and prevent it from overfitting.

The results is as below.

Table 2: Loss for different regularization

β	error rate
0.	16.1%
0.1	16.1%
1	17.8%
10	27.97%
100	49.15%

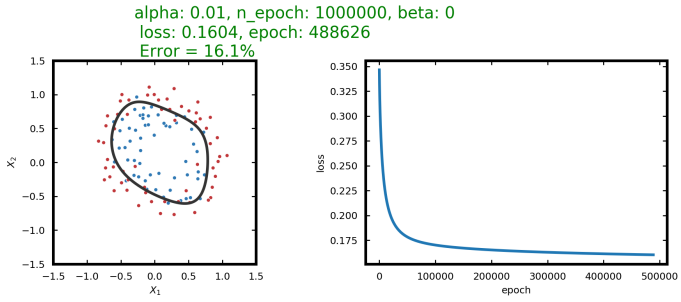

Figure 17: No regularization

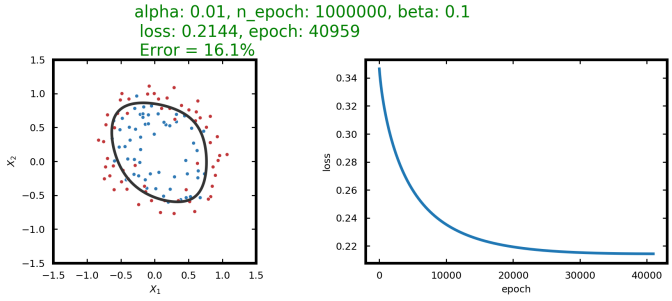

Figure 18: $\beta = 0.1$

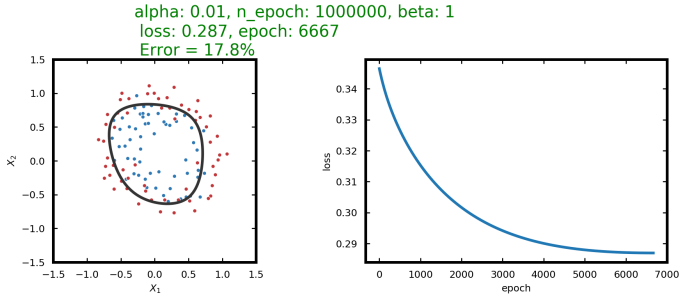

Figure 19: $\beta = 1$

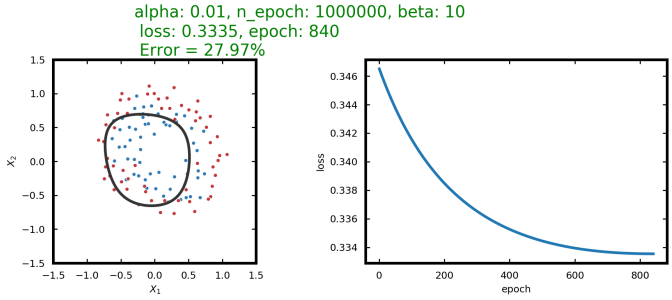

Figure 20: $\beta = 10$

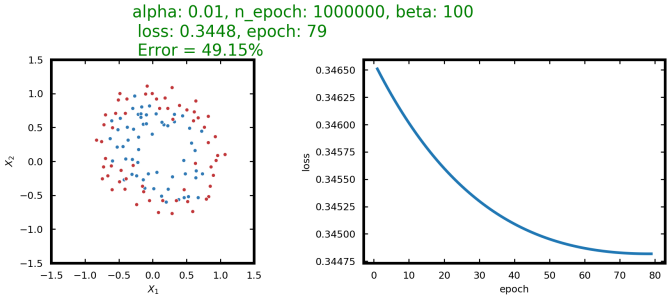

Figure 21: $\beta = 100$