Homework 2

Guanjie Zheng

Problem 1

Problem 1a

I tuned the parameter “step_size” and “reg” a little bit.

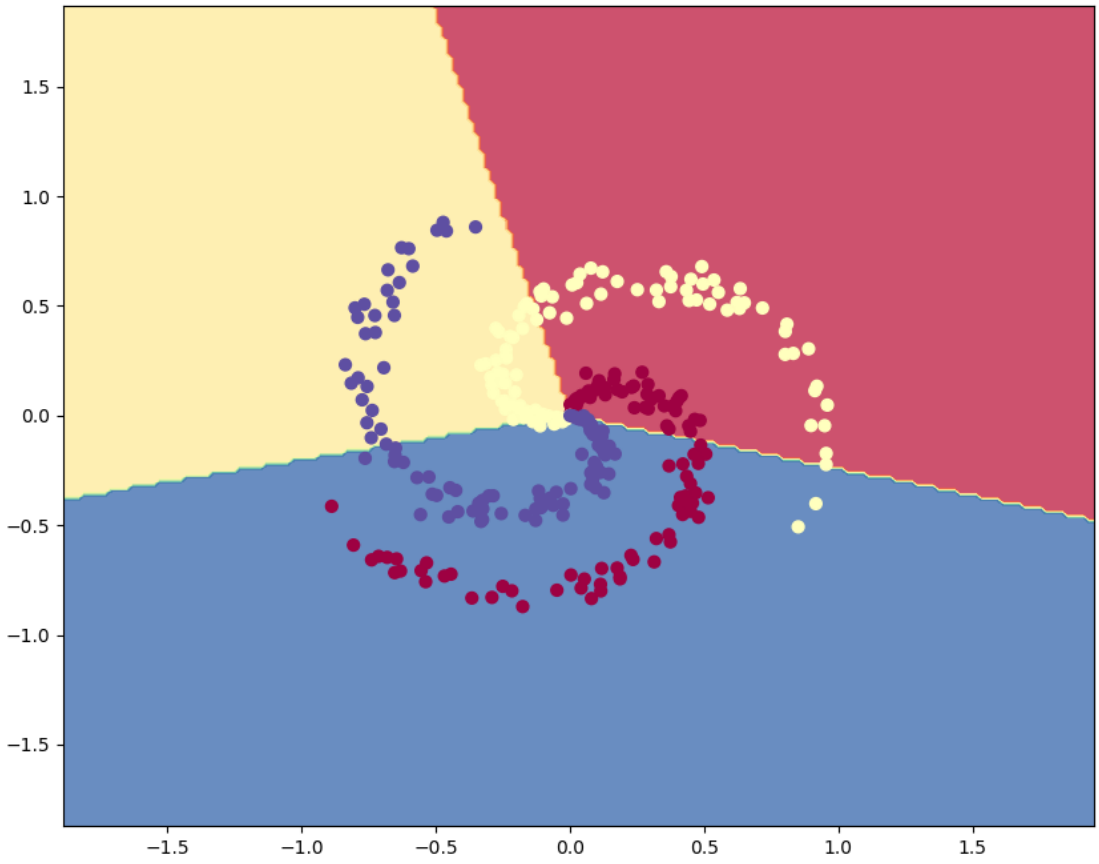
step_size	reg	acc
0.001	0.001	0.25
0.001	0.01	0.25
0.001	0.1	0.5
0.01	0.001	0.5
0.01	0.01	0.5
0.01	0.1	0.75
0.1	0.001	0.75
0.1	0.01	0.75
0.1	0.1	0.75

We can observe that when the step_size is small (e.g., 0.001), it seems the model is not fitted well, so the accuracy is very low. As the step_size becomes larger and regularization is larger, the accuracy becomes higher. I also tried increase the number of iterations, however, the highest accuracy is still 0.75. This is probably because this one-layer softmax regression can not model the non-linear relationship in this problem.

Problem 1b

step_size	reg	acc
0.001	0.001	0.54
0.001	0.01	0.54
0.001	0.1	0.55
0.01	0.001	0.49
0.01	0.01	0.49
0.01	0.1	0.49
0.1	0.001	0.49
0.1	0.01	0.49
0.1	0.1	0.49

I tune the step_size and reg as usual. The best accuracy I can achieve is 0.55. From the figure we can see, the three classes are in a highly non-linear pattern within these two displayed dimensions. Therefore, they are not linear separatable. Even if I tried to tune the parameters, the accuracy of this linear classifier is still very low.



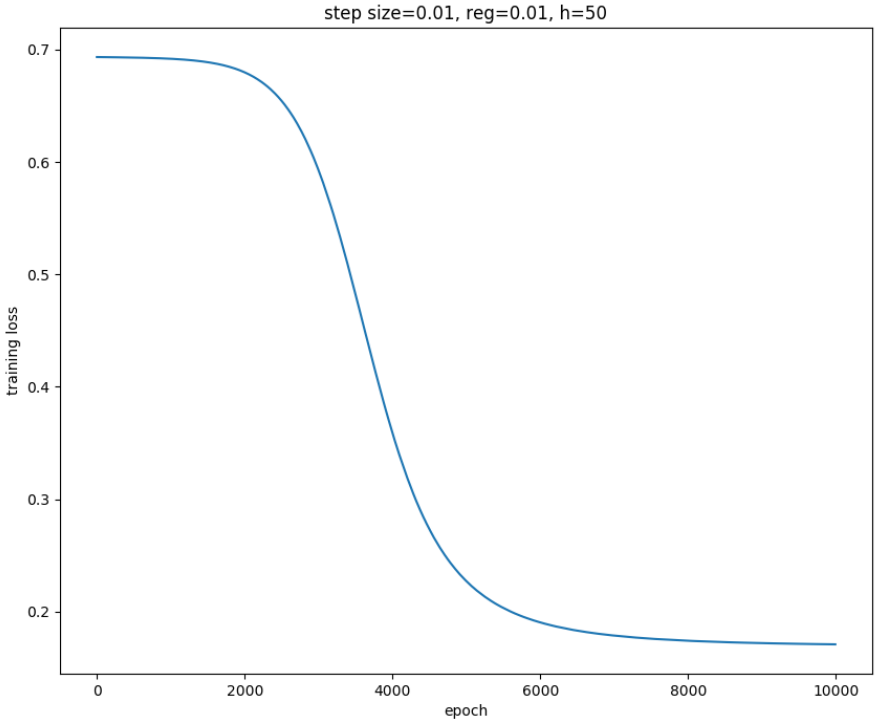
Problem 1c

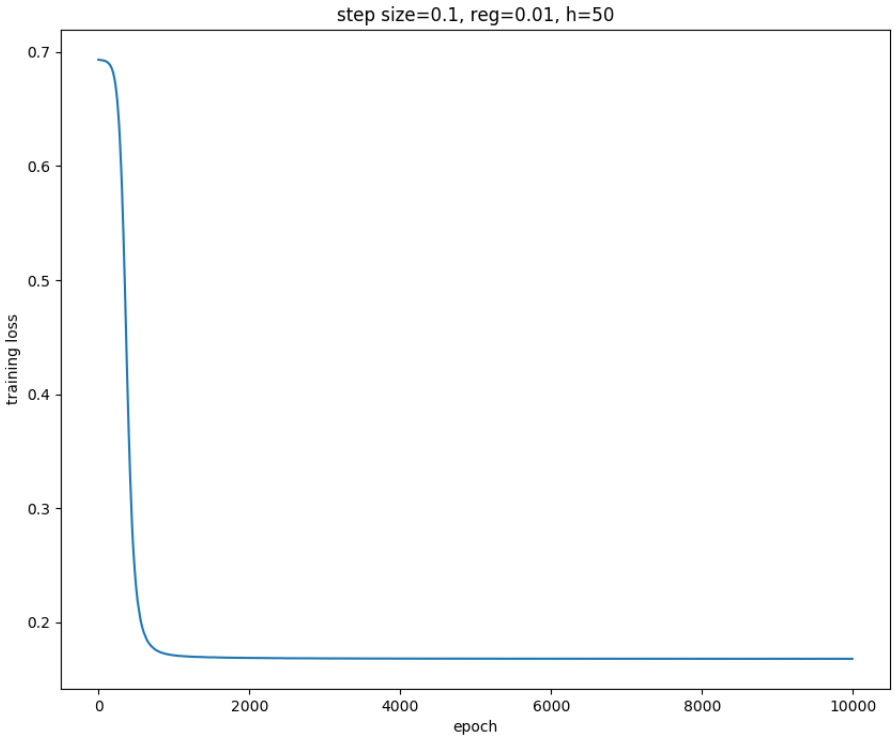
We can see that during the process I tune the step_size, reg, and hidden layer size, we can achieve higher accuracy (accuracy = 1) when larger hidden layer size is used.

Was your MLP model better able to fit the XOR data?
Why would this be the case, when compared to your softmax regressor?

We achieve better performance compared to the softmax regressor. This is because in this model, we have two layers of neurons, and we have the ReLU which can model the non-linear relationship in the XOR function.

step_size	reg	h	acc
0.001	0.001	10	0.75
0.001	0.001	50	1
0.001	0.001	100	1
0.001	0.01	10	0.75
0.001	0.01	50	1
0.001	0.01	100	1
0.001	0.1	10	0.75
0.001	0.1	50	1
0.001	0.1	100	1
0.01	0.001	10	0.75
0.01	0.001	50	1
0.01	0.001	100	1
0.01	0.01	10	0.75
0.01	0.01	50	1
0.01	0.01	100	1
0.01	0.1	10	0.75
0.01	0.1	50	1
0.01	0.1	100	1
0.1	0.001	10	0.75
0.1	0.001	50	1
0.1	0.001	100	1
0.1	0.01	10	0.75
0.1	0.01	50	1
0.1	0.01	100	1
0.1	0.1	10	0.75
0.1	0.1	50	1
0.1	0.1	100	1





We can see that, the training loss is getting smaller with increasing epoch. When we use larger step size, the converge process is faster.

Problem 1d

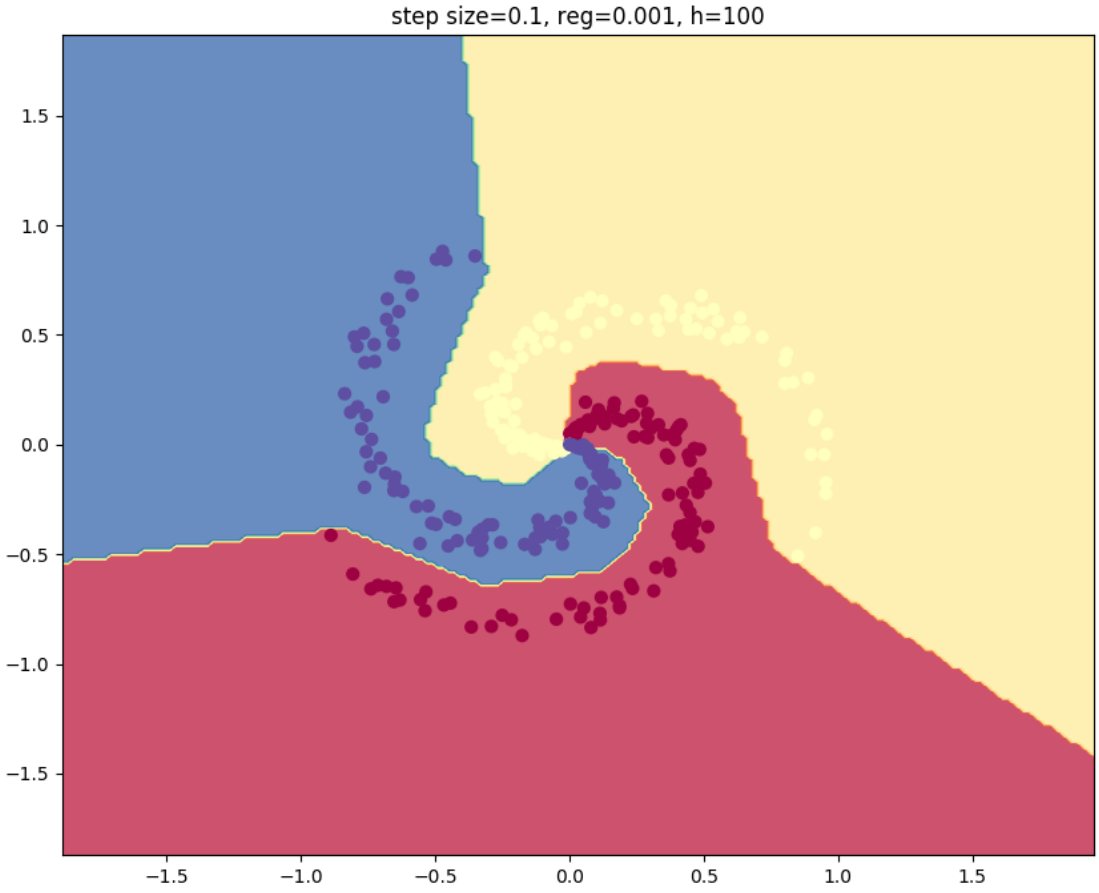
What you did to tune the MLP, including what settings of the hyper-parameters you explored (such as learning rate/step-size, number of hidden units, number of epochs, value of regularization coefficient, etc.). Do not forget to write your observations and thoughts/insights.

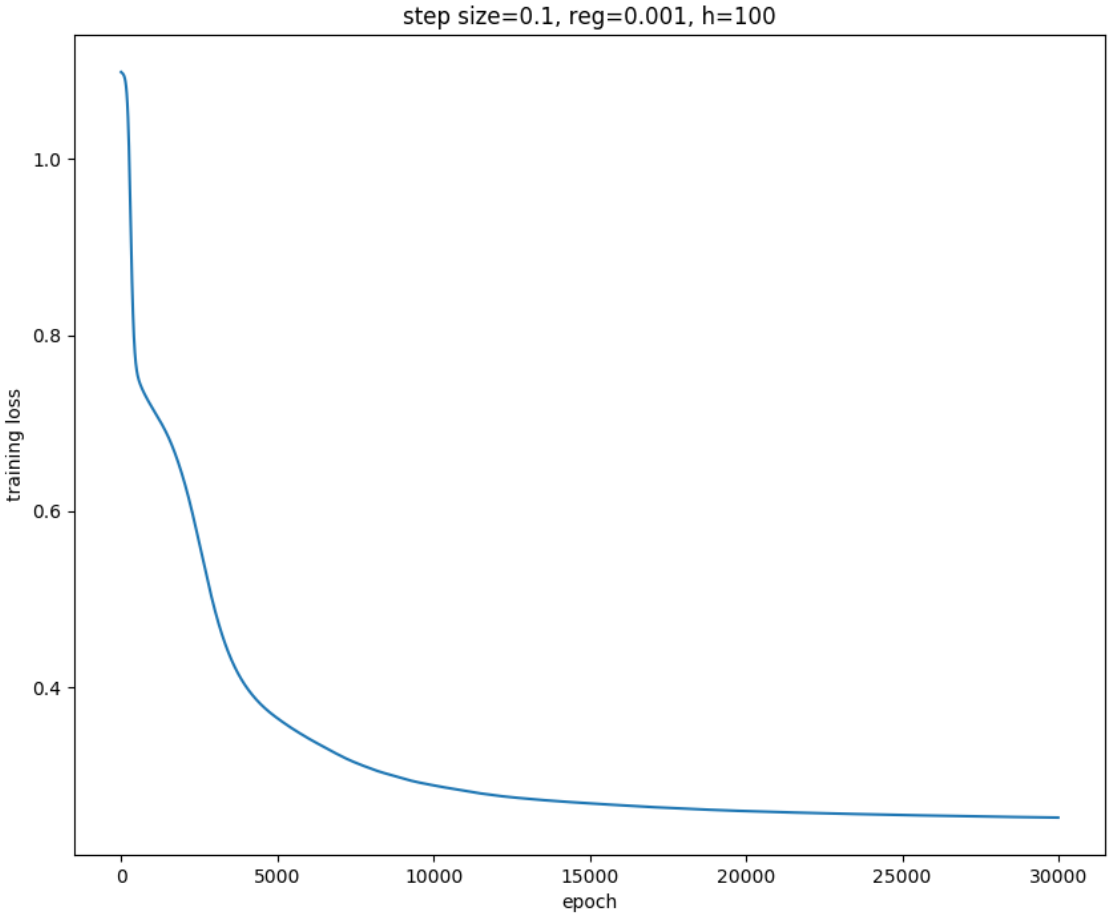
First, I tune the step size, reg, and hidden size. I found that step_size = 0.1, reg = 0.001, and hidden size = 100 gives me the best result. (Further tuning these three parameters does not change the result too much.) However, from the loss-epoch result, it seems the model is still not fully fitted, so I tune the n_epoch to be larger. Finally, I can get an accuracy of 0.98.

step_size	reg	h	acc
0.001	0.001	10	0.56
0.001	0.001	50	0.51
0.001	0.001	100	0.51
0.001	0.01	10	0.56
0.001	0.01	50	0.51
0.001	0.01	100	0.51
0.001	0.1	10	0.54
0.001	0.1	50	0.52
0.001	0.1	100	0.5
0.01	0.001	10	0.54
0.01	0.001	50	0.55
0.01	0.001	100	0.56
0.01	0.01	10	0.52
0.01	0.01	50	0.53
0.01	0.01	100	0.53
0.01	0.1	10	0.46
0.01	0.1	50	0.43
0.01	0.1	100	0.44
0.1	0.001	10	0.89
0.1	0.001	50	0.95
0.1	0.001	100	0.95
0.1	0.01	10	0.57
0.1	0.01	50	0.72
0.1	0.01	100	0.76
0.1	0.1	10	0.52
0.1	0.1	50	0.52
0.1	0.1	100	0.51

Generate the decision boundary plot of your tuned MLP and paste it into your answer document. Comment (in your answer document) on the decision boundary plot:

IST 597 Homework 2 **Guanjie Zheng**





What is different between the MLP's decision boundary and the multinoulli regressor's decision boundary?

This decision boundary is far more complex than the multinoulli regressor's decision boundary and is highly non-linear.

Does the MLP decision boundary accurately capture the data distribution?

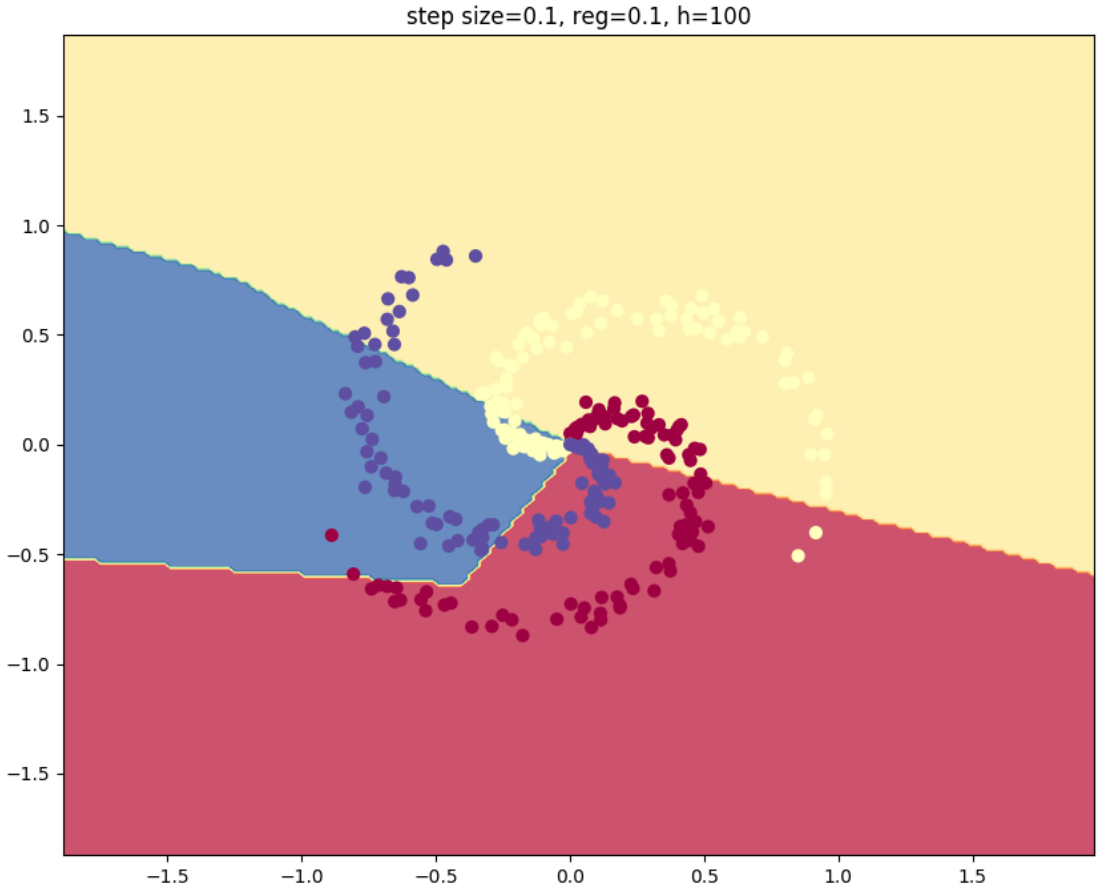
Yes.

How did the regularization coefficient affect your model's decision boundary?

Larger regularization will keep the model from overfitting (more rough decision boundary). For example, when the reg = 0.1, the decision boundary is almost linear. Compared with the previous

IST 597 Homework 2 **Guanjie Zheng**

decision boundary, this is very simple.

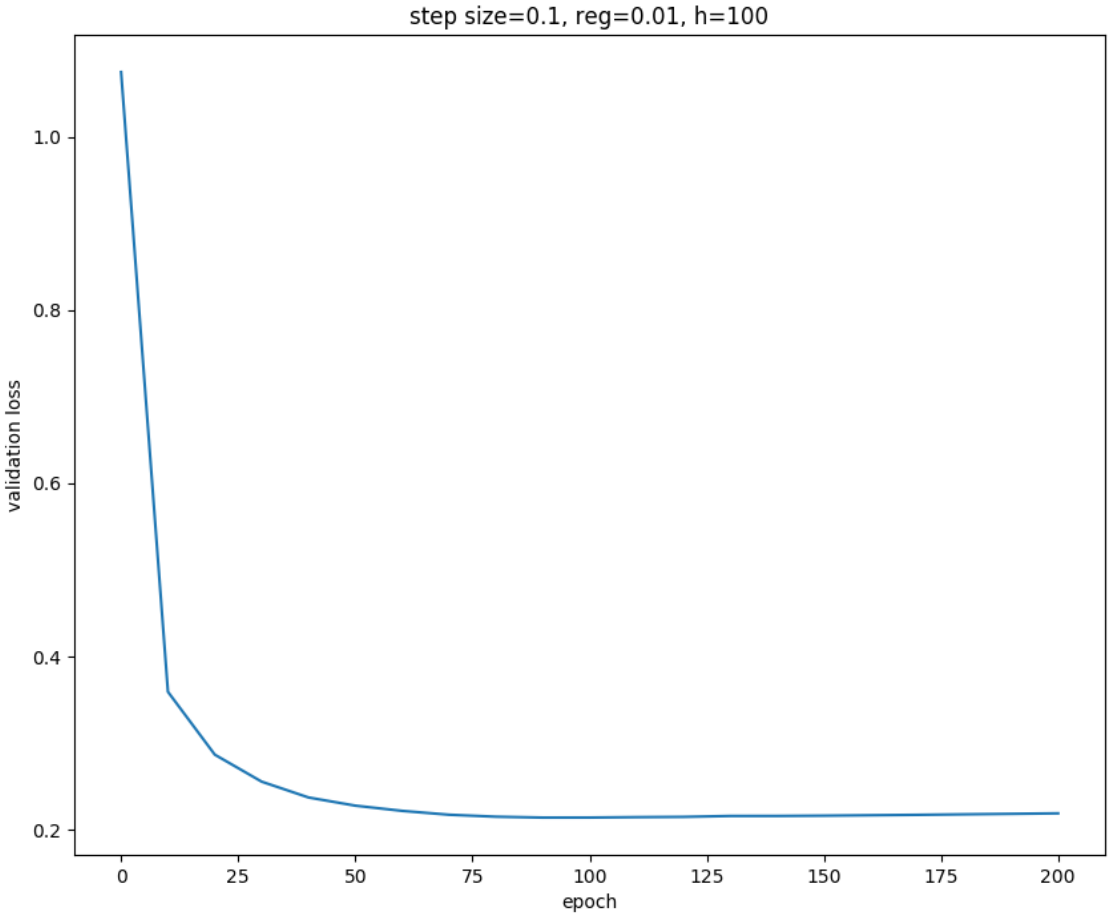


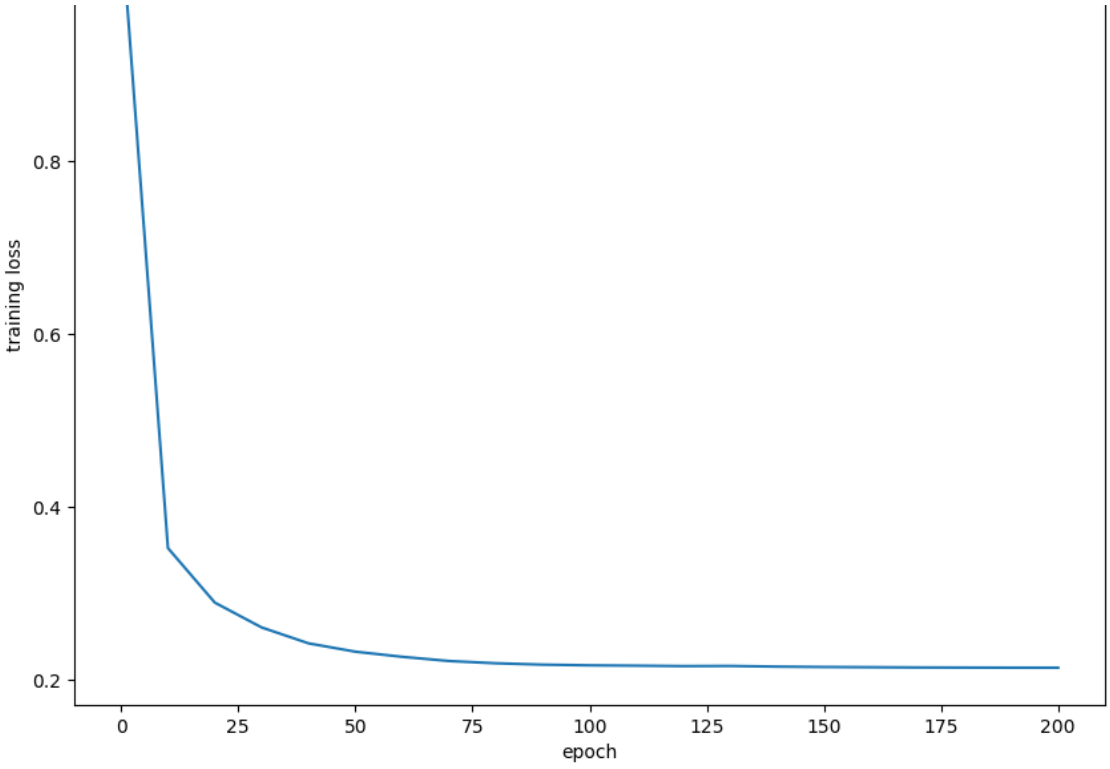
Problem 2

Problem 2a

Besides recording your accuracy for both your training and development sets, track your loss as a function of epoch. Create a plot with both of these curves superimposed. Furthermore, consider the following questions:

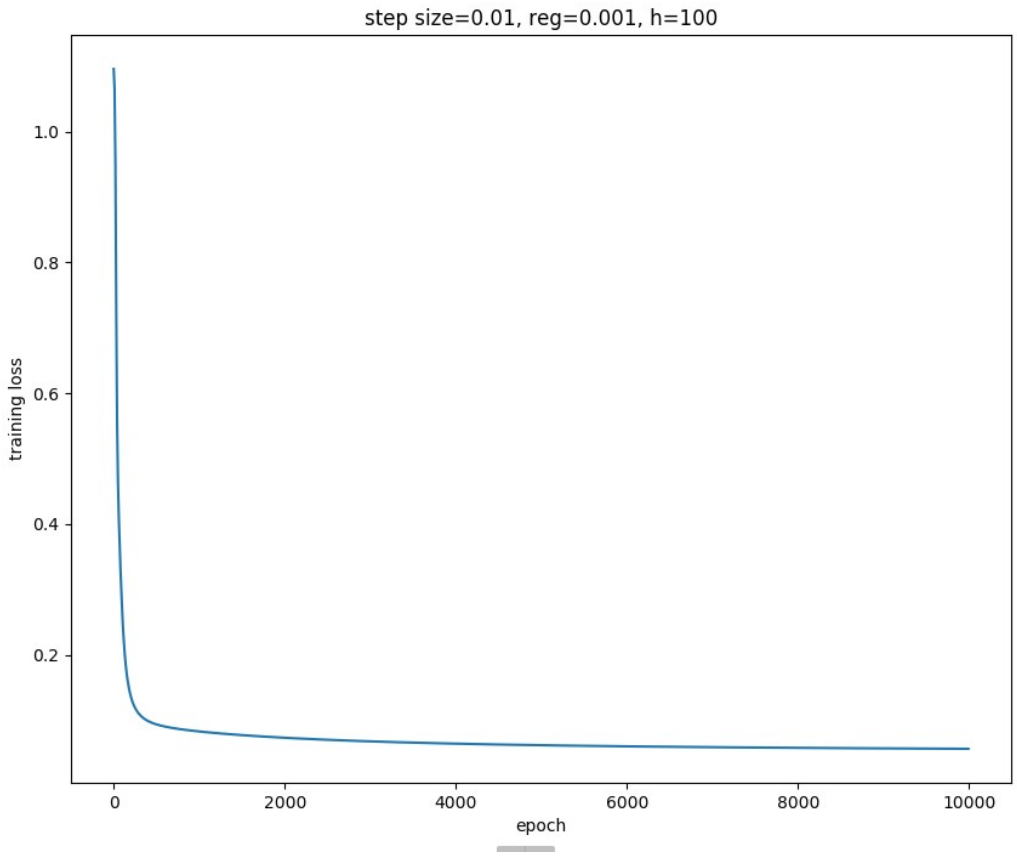
step_size	reg	h	train_acc	validation_acc
0.001	0.001	10	0.99	0.97
0.001	0.001	100	0.99	0.97
0.001	0.01	10	0.99	0.97
0.001	0.01	100	0.99	0.97
0.01	0.001	10	0.99	0.97
0.01	0.001	100	0.99	0.97
0.01	0.01	10	0.99	0.97
0.01	0.01	100	0.99	0.97
0.1	0.001	10	0.96	0.97
0.1	0.001	100	0.96	0.97
0.1	0.01	10	0.96	0.97
0.1	0.01	100	0.96	0.97

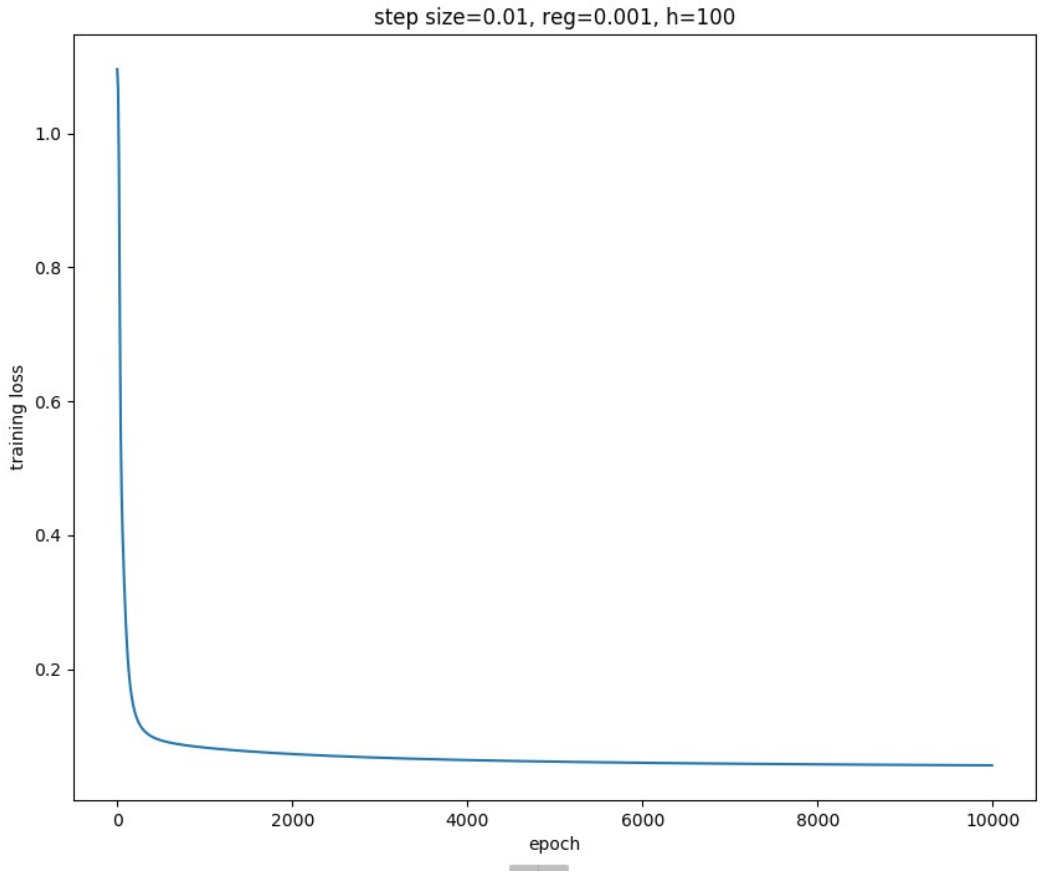

step size=0.1, reg=0.01, h=100
validation loss
1.0
0.8
0.6
0.4
0.2
0
25
50
75
100
125
150
175
200
epoch


training loss
0.8
0.6
0.4
0.2
0
25
50
75
100
125
150
175
200
epoch

What ultimately happens as you train your MLP for more epochs?

The training loss will become smaller and smaller, and the validation loss will first decrease and then increase.





What phenomenon are you observing and why does this happen?

When the validation loss begins to increase, overfitting happens. This is because the model has parameter size comparable to the size of the data, and then the model is over trained to overfit the data.

What are two ways you can control for this?

(1) We can use a smaller number of epochs, or use an early stop criterion. For example, in my experiment, if the validation loss has not been decreasing in the last 10 epoch checks, then I will do an early stop.
(2) We can also choose to use a larger regularization coefficient.

Problem 2b

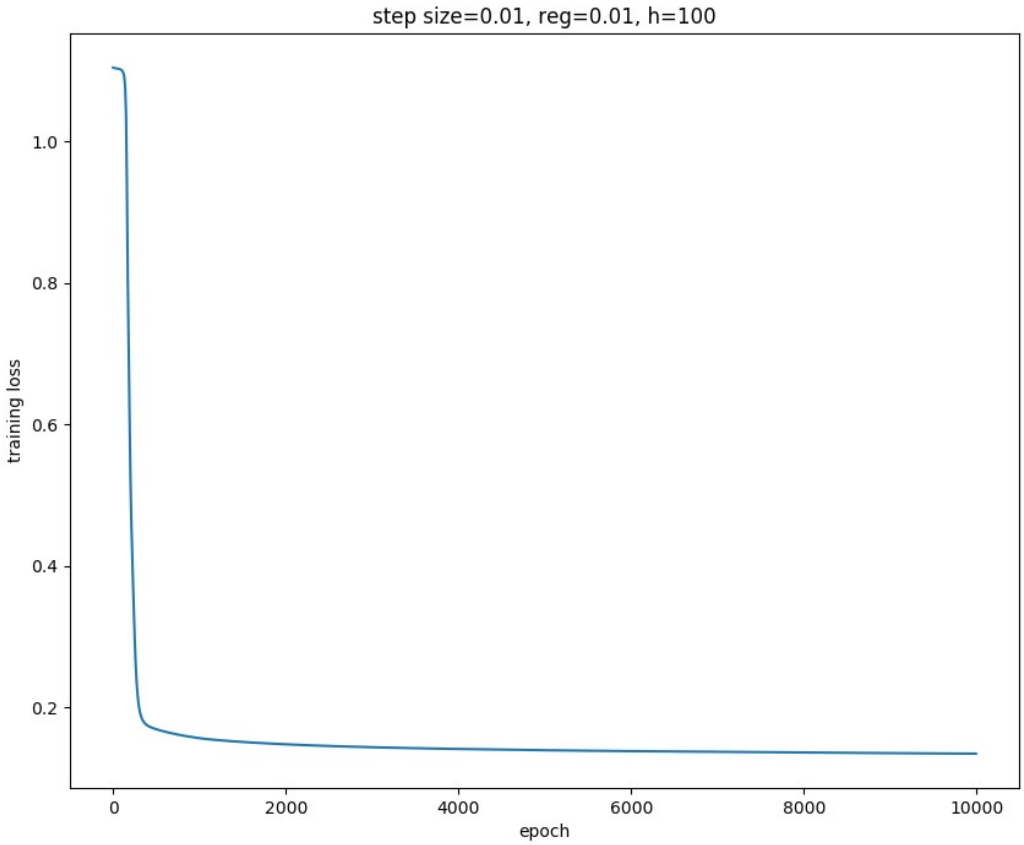
Create plots of your loss-versus-epoch curves as you did in Problem #1a. Put all of this in your answer document and write down any

IST 597 Homework 2 **Guanjie Zheng**

thoughts/comments of your tuning process and the differences observed between training a single and two-hidden layer MLP.

step_size	reg	h	train_acc	validation_acc
0.001	0.001	10	0.35	0.3
0.001	0.001	100	0.35	0.3
0.001	0.01	10	0.35	0.3
0.001	0.01	100	0.35	0.3
0.01	0.001	10	0.35	0.3
0.01	0.001	100	0.98	0.97
0.01	0.01	10	0.35	0.3
0.01	0.01	100	0.98	0.97
0.1	0.001	10	0.97	0.97
0.1	0.001	100	0.96	0.97
0.1	0.01	10	0.96	0.97
0.1	0.01	100	0.96	0.97

I tune the above three parameters. The best training accuracy is 0.98, and validation accuracy is 0.97. It is expected that this should achieve better accuracy than the single layer MLP, but interestingly, it is not. Probably it is because the single layer MLP is already powerful enough?


step size=0.01, reg=0.01, h=100
training loss
epoch
1.0
0.8
0.6
0.4
0.2
0
2000
4000
6000
8000
10000

IST 597 Homework 2 **Guanjie Zheng**