ACML-Assignment1

yumao.gu

November 2020

1 Model

The model of this assignment is a network with 3 layers: input - hidden - output. Both the input layer and the output layer will have 8 nodes, the hidden layer only 3 nodes (+biases). In hidden layer and output layer, we use sigmod function as our activation function.

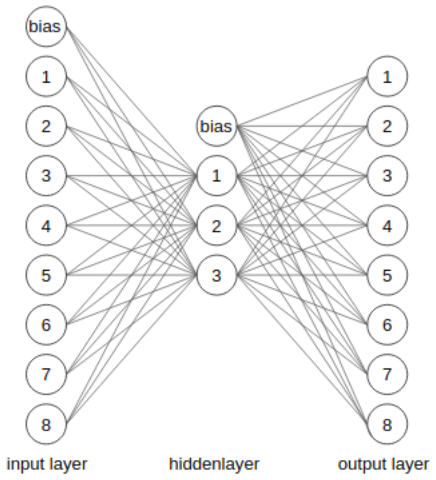


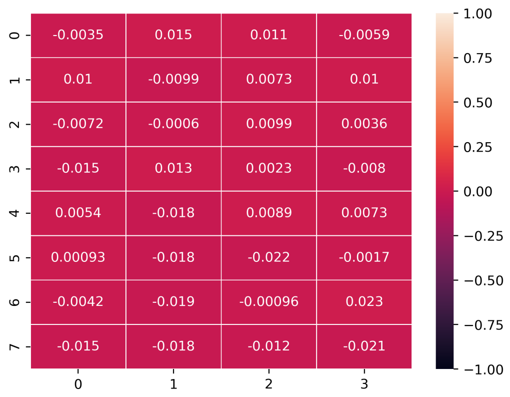
Figure 1: The Model of 9 * 4 * 8

2 Input and Output

The learning examples will each have 7 zeros and 1 one in them (so there will be only 8 different learning examples, and you will have to repeat them) and the output the network should learn is exactly the same as the input. So when the input layer is given $< 0, 0, 0, 1, 0, 0, 0, 0 >$ as input, the output to aim for is also $< 0, 0, 0, 1, 0, 0, 0, 0 >$.

3 Weight Initialization

The initialization of all weights is obtained according to the normal distribution. The variance of the normal distribution is 0.01.



	0	1	2	3
0	-0.0035	0.015	0.011	-0.0059
1	0.01	-0.0099	0.0073	0.01
2	-0.0072	-0.0006	0.0099	0.0036
3	-0.015	0.013	0.0023	-0.008
4	0.0054	-0.018	0.0089	0.0073
5	0.00093	-0.018	-0.022	-0.0017
6	-0.0042	-0.019	-0.00096	0.023
7	-0.015	-0.018	-0.012	-0.021

Figure 2: The initial weights from hidden layer to output layer

4 Loss Function

5 Weight Decay Effect

We fix the learning rate $\alpha = 0.01$ and change the weight decay λ (L2 regularization coefficient). We can see there is a plateau area in the graph which we will discuss later. In general, the smallerλ, the faster our loss converges to a very low value. And small λ make it earlier to go into the plateau area and easier to run out of this area.

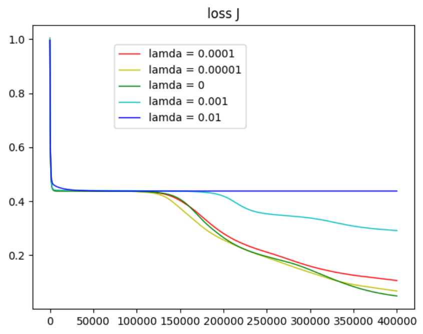


Figure 3: The convergence of error with different λ

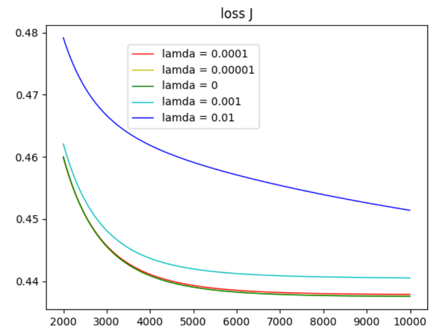


Figure 4: The early iterations. Our loss goes into the plateau area easily with small λ

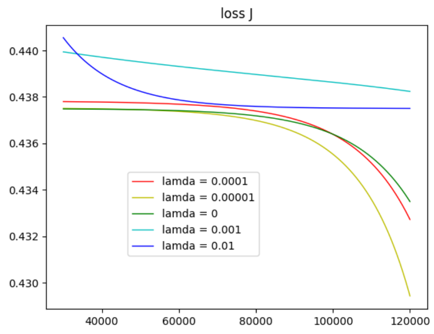


Figure 5: Our loss runs out of the plateau area easily with small λ,too

So in this model we can say the L2 regularization has a negative effect. The effect of L2 regularization is to scale each element of the original optimal solution in different proportions. So it is normally have lower convergence rate when we use higher λ. But it is equivalent to assuming that the prior distribution of weights is Gaussian, which will make it harder to run out of the plateau area.

6 Plateau Reason

The plateau area arises even if we don't use L2 regularisation, so it must be an internal reason of the model. As we can see, since the bias is 1 in the beginning,which is bigger than other node value after activation function, so back propagation would first add bias weight. It would lead to a saddle point of loss function. When learning rate is small, it can hardly to go through the saddle point so it shows as a plateau area.

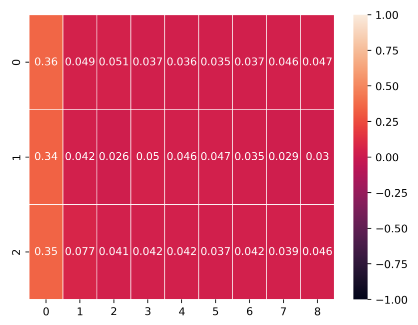


Figure 6: The weight in the plateau area

7 Learning Rate Effect

We fix the weight decay $\lambda = 0$ (L2 regularization coefficient) and change the learning rate α . We can see there is also a plateau area in the graph when α is small. But when α is big, it can easily run out of this local minimum area because of big update step.

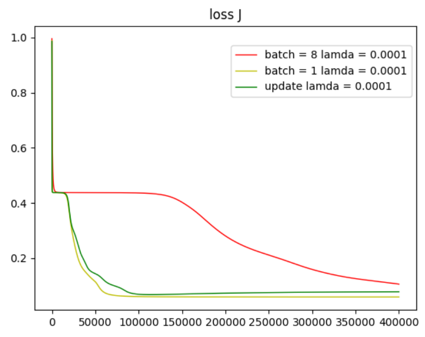


Figure 7: The different update batch size.

8 Batch Number Effect

When we update weights after each input instead of all 8 inputs, the convergence rate is much higher. It's easy to understand because different input has opposite influence of weight from hidden layer to output layer. And it is more easy to run out of the local minimum because we have bigger update step in each iterations. We also try a new update way. We use the loss function without L2 to update the weight using every input and update it via the L2 after all 8 inputs. The error decrease in the beginning but gradually started to rebound. It is normal

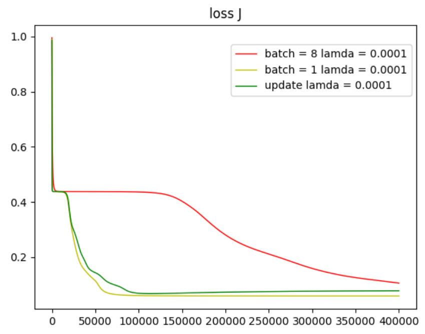
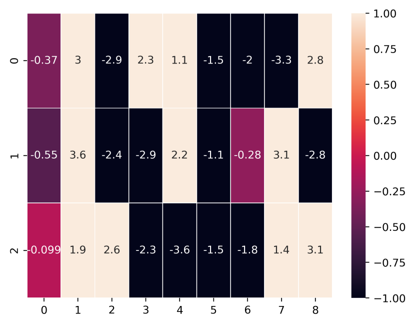


Figure 8: The different update batch size.

because the new loss function is no longer the original one.

9 Weight Interpretation

We just analyze the final weights with low error. Through the heat map below, if we set 0 if value ¿ 1 and 1 if value ¡ -1, we can see very interesting thing regardless of the bias weight. We can see that the weights could be a binary representation of 8 like 001,010,110...just because of $2^3 = 8$. It is obvious that if we want use 3 nodes to represent 8 nodes, the easiest way is to use binary number.



	0	1	2	3	4	5	6	7	8
0	-0.37	3	-2.9	2.3	1.1	-1.5	-2	-3.3	2.8
1	-0.55	3.6	-2.4	-2.9	2.2	-1.1	-0.28	3.1	-2.8
2	-0.099	1.9	2.6	-2.3	-3.6	-1.5	-1.8	1.4	3.1

Figure 9: The 3*9 weights from input to hidden, with the first column is bias. Look at each column as one binary number

References

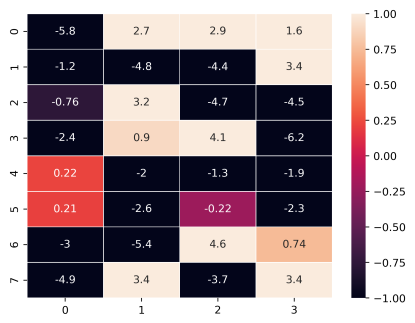


Figure 10: The 8*4 weights from hidden to output, with the first column is bias. Look at each row as one binary number