Deep Learning in Data Science, DD2424

Short report on lab assignment 1

One-layer neural network with multiple outputs

Hamid Dimyati

1. Introduction

The goal of this lab assignment 1 is to train and test a single layer neural network to classify multiple classes of CIFAR-10 datasets. The network applies cross-entropy loss function and L_2 regularization with parameters learning using mini-batch gradient descent.

The tool used for this assignment is primarily `python 3.7.3`, along with several packages including `numpy 1.16.4`, `pandas 0.25.3`, and `matplotlib 3.1.0`.

2. Results and discussion

Before start building the network, a standardization (Z-score normalization) is performed. To validate whether the self-built codes, particularly the function to compute the gradient of parameters W and b, could run appropriately based on the mathematical functions, comparing the results with alternative ways using the numerical approach as provided in the instruction documents is necessary. A `numpy.allclose` function is used to ensure the difference between those two outputs are similar, which follows the equation (1):

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)} \approx absolute\ tolerance \tag{1}$$

Using that function with $absolute\ tolerance = 1e-4$, my codes (g_a) could successfully achieve similar results to the numerical approach (g_n) with an acceptable gap.

Once the codes were proven to yield reliable results, the full neural network is then trained based on different hyperparameters, as stated in Table 1.

Table 1. Hyperparameter settings for four models

Model	**Hyperparameters**			
	Total epoch	**Batch size**	**Lambda**	**Eta**
Model 1	40	100	0	0.1
Model 2	40	100	0	0.001
Model 3	40	100	0.1	0.001
Model 4	40	100	1	0.001

The datasets used for this assignment are CIFAR-10 batch 1 for training, CIFAR-10 batch 2 for validation, and CIFAR-10 test batch for testing. The detail results are seen in Table 2 and eight figures below:

Table 2. Accuracy for each model

Model	**Accuracy**			**Accuracy gap (train – test)**
	Training	**Validation**	**Testing**	
Model 1	40.85%	27.87%	27.72%	13.13%
Model 2	45.40%	38.14%	38.76%	6.64%
Model 3	44.75%	38.37%	39.20%	5.55%
Model 4	39.93%	36.41%	37.54%	2.39%

Model 1

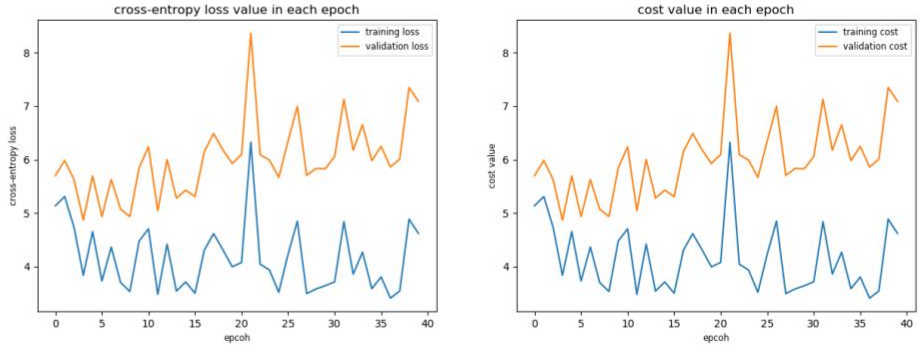

Figure 1. Cross-entropy loss (left) and cost (right) value by epoch from Model 1

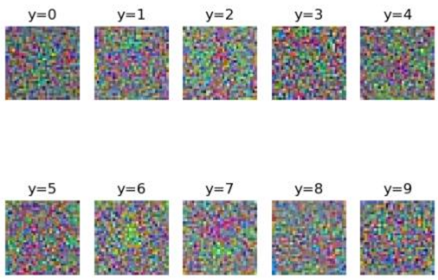

Figure 2. Image representation of learned weight from Model 1

Model 2

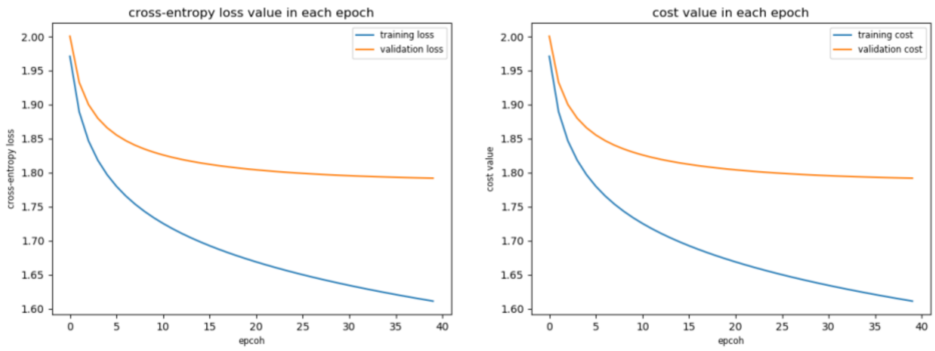

Figure 3. Cross-entropy loss (left) and cost (right) value by epoch from Model 2

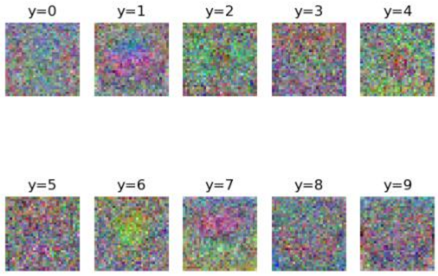

Figure 4. Image representation of learned weight from Model 2

Model 3

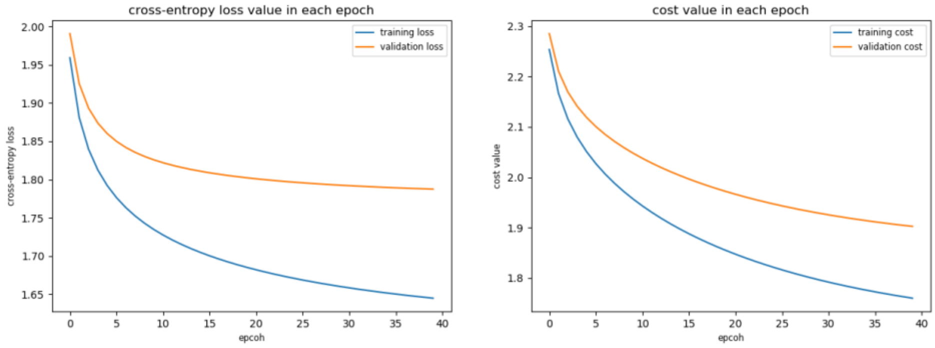

Figure 5. Cross-entropy loss (left) and cost (right) value by epoch from Model 3

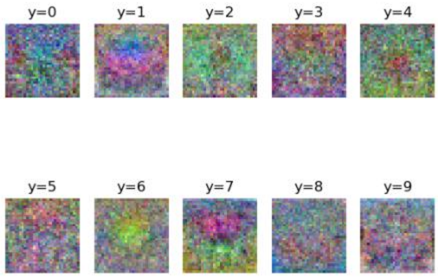

Figure 6. Image representation of learned weight from Model 3

Model 4

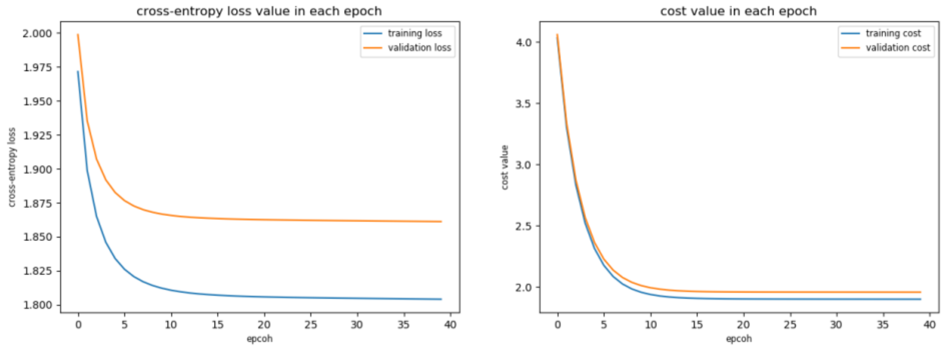

Figure 7. Cross-entropy loss (left) and cost (right) value by epoch from Model 4

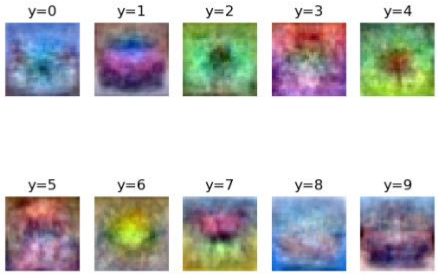

Figure 8. Image representation of learned weight from Model 4

2.1. The role of the learning rate

To investigate the role of the learning rate (eta) during training the model, we need to compare the result of Model 1 (eta=0.1) and Model 2 (eta=0.001). In terms of accuracy on testing data, Model 2 with smaller eta could outperform Model 1, achieving 38.76% accuracy compared to 27.72%. This fact is supported by the generated image from those two models, as seen in Figure 4 and Figure 2, respectively.

Figure 1 and Figure 3 show the difference between those two models based on their cross-entropy loss and cost reduction values. Since these two models did not employ regularization, thus the plot of the cross-entropy loss and cost values are identical. Model 1, with a higher value of eta, shows no significant cost reduction during training, while Model 2 performed much better with a quite small value of eta. Having a high learning rate makes the gradient descent overshot to the different local minimum, which causes the learning process not to make good progress. Otherwise, a quite small learning rate could drive the learning process toward the targeted local minimum confidently.

2.2. The role of lambda

To investigate the role of the lambda during training the model, we could compare the result of Model 2 (lambda=0), Model 3 (lambda=0.1), and Model 4 (lambda=1). The lambda parameter helps the model to control its complexity by reducing the value of the weight matrix. With a hope to avoid the overfitting, it reduces the variance and increases the bias due to shrinking some parameters' contribution. Hence, it is understandable to see a rise in the training error or decrease of the model accuracy, but it is expected to have better generalization power. As seen in Table 2, there happens a reduction in the training accuracy as the regularization parameter (lambda) increases: from Model 2 without regularization (45.40%) to Model 3 with lambda=0.1 (44.75%) to Model 4 with lambda=1 (39.93%).

Introducing a regularization parameter could help to enhance the model's generalization capability, which is proved by a small accuracy gap between training and testing accuracy, as seen in Table 2. In terms of the cost optimization process, the higher regularization parameter could reduce the gap between the cost value of training and validation data, as depicted in Figure 3, Figure 5, and Figure 7. Besides, a high regularization parameter may seem to help the learning algorithm converge faster, as seen in Figure 7.

However, too high lambda could also mean increasing the bias error. Since the high regularization parameter controls the weight matrix tightly, it could lead to producing minimal weight (close to zero). Therefore, in Figure 8, it can be seen as there are more points with black colors in the image generated by Model 4.

3. Final remarks

From this assignment, it could be learned that a small learning rate plays a role in leading the gradient descent converging to a local minimum smoothly. Further, the lambda as a regularization parameter contributes to control the model complexity achieving good generalization capability and avoiding overfitting problems. A quite small value of lambda is recommended to gain those advantages.