Assignment2 Report

Jianxiao Luo 300123069

October 2020

1 Introduction

Assignment 2 asks us to design and train image classifiers for two datasets. Those two datasets are MNIST and CIFAR-10. Three different models are required, including soft-max regression, MLP and CNN, and for each dataset, they should be trained respectively.

2 Dataset

2.1 Description

The first dataset MNIST contains 60,000 examples in training set and 10,000 examples in testing set. Each example is a (28×28) pixel matrix, representing a hand-writing number. The task is to recongize those numbers(0-9).

The second dataset CIFAR-10 contains 50,000 examples in training set and 10,000 examples in testing set. Each example is a $(3 \times 32 \times 32)$ pixel matrix, representing a colour image (with 3 channels). There are also 10 classes in total.

2.2 Load data

In PyTorch, there are some built-in functions to load popular datasets. For MNIST, we can use *dsets.MNIST()* and for CIFAR-10, we use *torchvision.datasets.CIFAR10()*. After downloading or loading the dataset, we should construct iterable training and testing dataset. *torch.utils.data.DataLoader()* is used to represents a Python iterable over a dataset.

3 Soft-max regression

Soft-max regression is a method to do multi-class classification, and it is a generalization of the logistic function.

$$p_{Y|X}(\cdot|x) = softmax(Wx)$$

W is a $K \times (m+1)$ matrix and x is a $(m+1)$ vector, where K equals to the size of input and m equals to the number of output nodes, which is also the number of classes.

Loss function of soft-max regression model is defined by cross-entropy. The learning goal is to find $\hat{W}$ that minimizes the cross-entropy loss.

$$l(W) = -\frac{1}{N}\sum_{i+1}^{N} log p_{Y|X}(y^i|x_i)$$

In pytorch, the regression can be implemented by using *torch.nn.Linear(input_dim, output_dim)*.

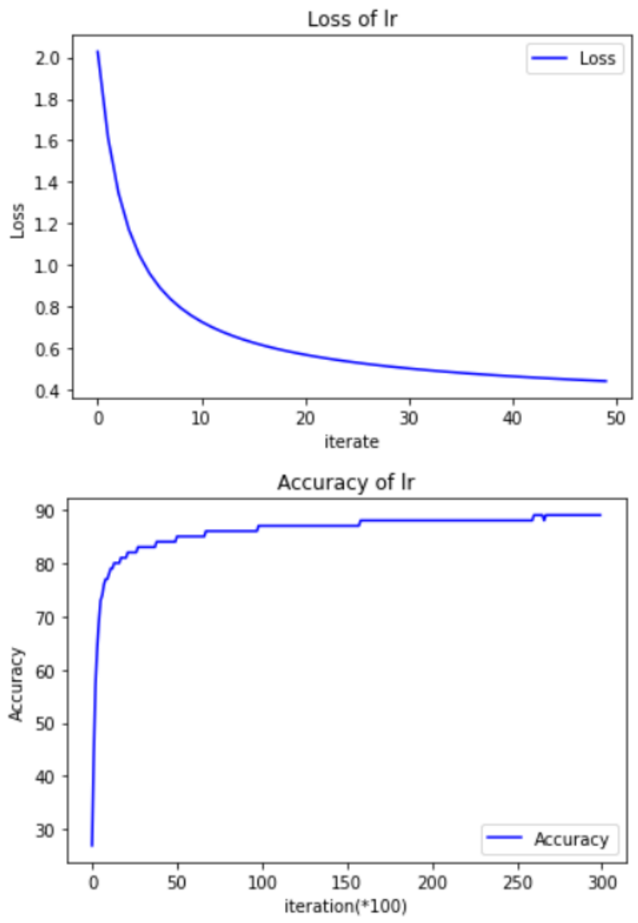

Figure 1: MNIST with soft-max regression and epoch=20

3.1 MNIST

For the MNIST dataset, its image size is (28×28) and the number of classes is 10, so the input size is 784 and output size is 10. I store the loss in each iteration and do the test each 100 iterations to see the change of accuracy. Figure 1 is the curve of training loss and the test accuracy during the training.

We can see that the accuracy get stable at 88% after 15,000 iterations. It actually already is a very good result for soft-max regression. The only parameter can be changed in this model is learning rate, so there is not much space to explore.

3.2 CIFAR-10

For CIFAR-10 dataset, each example is a $(3 \times 32 \times 32)$ image and the number of classes is also 10, so the input size is 3072 while output size is 10. The first three means 3 channels, because the images in CIFAR-10 dataset are colored, they have value in RGB. Since it has much larger input size, the training process is much slower accordingly.

Figure 2 is the curve of the training loss and the testing accuracy during the training process in 100,000 iterations. The accuracy becomes stable at 40% after 50,000 iterations, which is much slower and worse than the one of MNIST. It means that soft-max regression does not work well when there the input size is large, which is probably cause by the fact that without any hidden layer, the only weights cannot store that much information of the data.

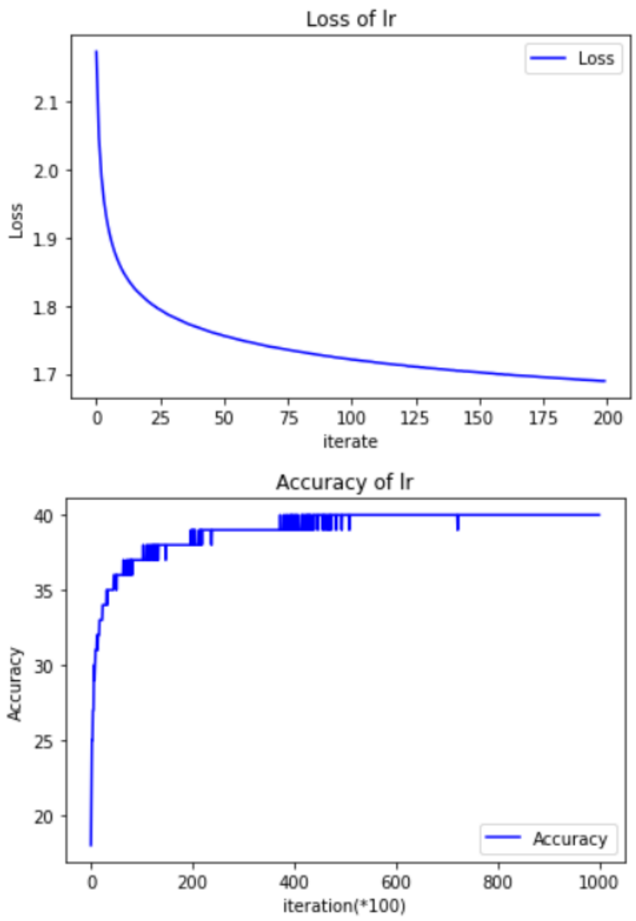

Figure 2: CIFAR-10 with soft-max regression and epoch=200

4 MLP

A simple and standard MLP consists of input layer, hidden layers, activation functions and output layers. Suppose that my network has two hidden layers and they use ReLU as the activation function and the output layer uses softmax as the activation function, then the model would be like this:

$$p_{Y|X}(\cdot|x) = \mathbf{softmax}(W \cdot g\mathbf{ReLu}(W^{(2)} \cdot g\mathbf{ReLu}(W^{(1)}x)))$$

MLP can be considered as a complex version of soft-max regression model. With more layers and weights, it can dig more rules hidden in the data.

4.1 MNIST

The input size of MNIST is also 784 in MLP. First of all, we need to define the network. This is the construct of the network I defined. It has two hidden layers, one has 512 nodes and the other one has 128 nodes. I used ReLu function in both of the hidden layer and softmax function in the output layer. The network is not very complex, so it runs very fast.

$$\textit{input layer} \rightarrow ReLU(fc1(512)) \ \rightarrow \ ReLU(fc2(128)) \rightarrow softmax(\textit{output layer}(10))$$

Figure 3 is the result of the training and testing. The curve of accuracy becomes flat after 1,300 iterations and finally reach a accuracy at 94%.

An interesting phenomenon is that although the loss of MLP does not decline to a tiny number, and it is even higher than soft-max regression, its accuracy for testing set is very high and it needs much less iteration to get there. It means that high loss does not always lead to a low accuracy.

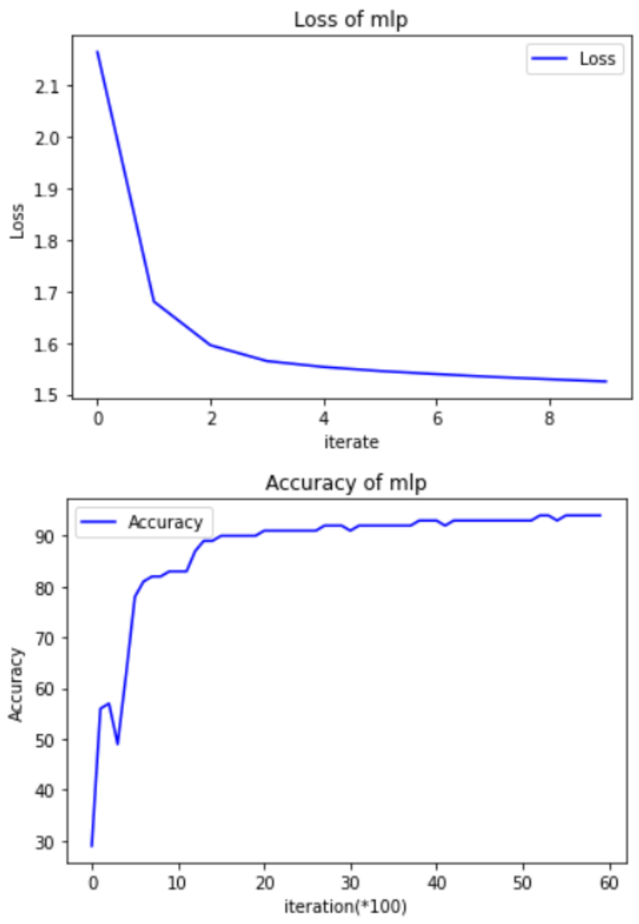

Figure 3: MNIST with MLP and epoch=10

4.2 CIFAR-10

For CIFAR-10 dataset, I used the same network mentioned above as MNIST in the first place. It has much more input than MNIST, so I increase the training epoch to 100.

Figure 4 is the result. It learns slowly and have a low accuracy 52%at last.

After analysis on the result, I think the low testing accuracy and high loss might be because of the simple network structure. So I modified the network a bit to see if anything changes.

- Experiment 1. The first experiment I did is to increase the width of the network. The new network has a structure like this: ($3072 \rightarrow 1024 \rightarrow 512 \rightarrow 10$), both of the two hidden layers are doubled. Theoretically, increasing the width of MLP can help sufficiently express the learning problem and also more possible to cause overfitting.

  Figure 5 is the accuracy curve during learning. It is almost the same as figure 4. As a result, it is an unsuccessful try.

- Experiment 2. The second experiment I did is to increase the depth of the network. The new network has a structure like this: $3072 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 10$. The final accuracy also does not have significant increase. Comparing the curve with the one with wider layer, we can see that this one goes up faster and is sharper.

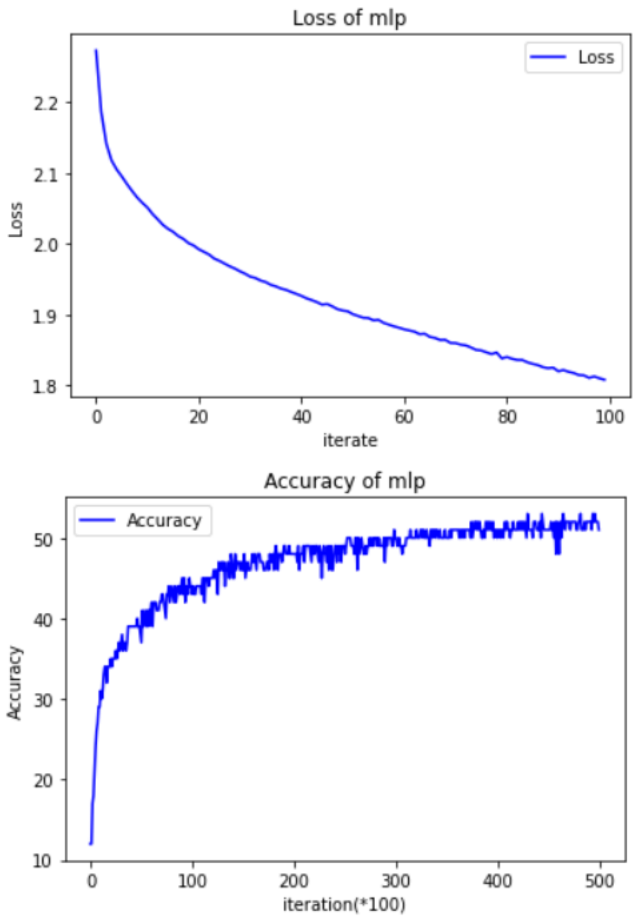

Figure 4: CIFAR with MLP and epoch=100

5 CNN

A normal CNN consists of input layer, convolutional layers, pooling layers, ReLU layers, fully-connected layers and output layer. This is the network I firstly used while doing experiments.

$$input\ layer \rightarrow Conv1 \rightarrow maxpooling \rightarrow ReLU \rightarrow Conv2 \rightarrow maxpooling \rightarrow ReLU$$

$$\rightarrow maxpooling \rightarrow ReLU \rightarrow ReLU(fc1) \rightarrow softmax(fc2) \rightarrow output\ layer$$

5.1 MNIST

For the MNIST dataset, the input size is $(1 \times 28 \times 28)$. The parameters of the layers are:

(conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
(fc1): Linear(in_features=320, out_features=50, bias=True)
(conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
(fc2): Linear(in_features=50, out_features=10, bias=True)

Figure 8 is the learning process of CNN. As we can see, it has the highest accuracy in three models (98%). CNN is born for image processing. Figure 7 is the visualization of the kernels in the first convolution layer.

There is not too much to analyze and explore, the current result is already impressive enough.

5.2 CIFAR-10

When it comes to CIFAR-10 dataset, there is a lot to explore. When I use the same network as MNIST, the result is really bad, as usual, which is shown in figure 9.

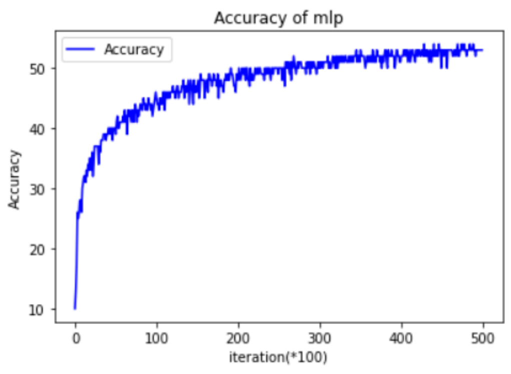

Figure 5: Experiment 1, MLP with wider hidden layer

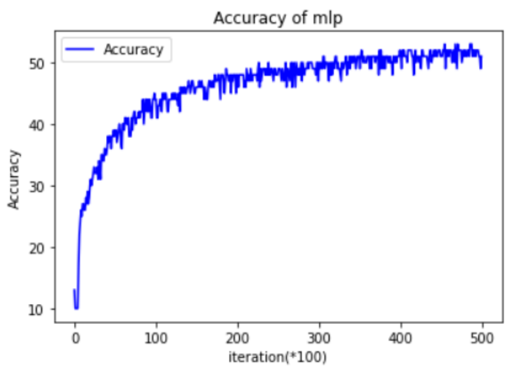

Figure 6: Experiment 2, MLP with deeper hidden layer

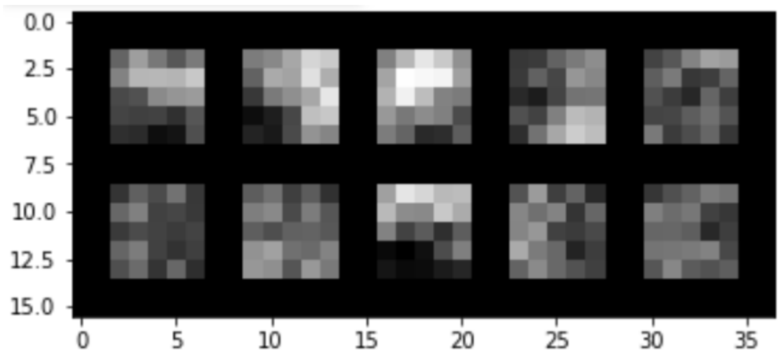

Figure 7: Visualization of kernels in Conv1

The performance is bad, with a long training time and very low accuracy (56%), shown in figure 9 So I did some experiment to try to enhance the performance.

- Experiment 1. Since the CIFAR-10 has much more data points compared with MNIST, I add one more layer to the mlp.

  Figure 10 and are the learning curves of the experiments. The performance increase a little bit, not too much.

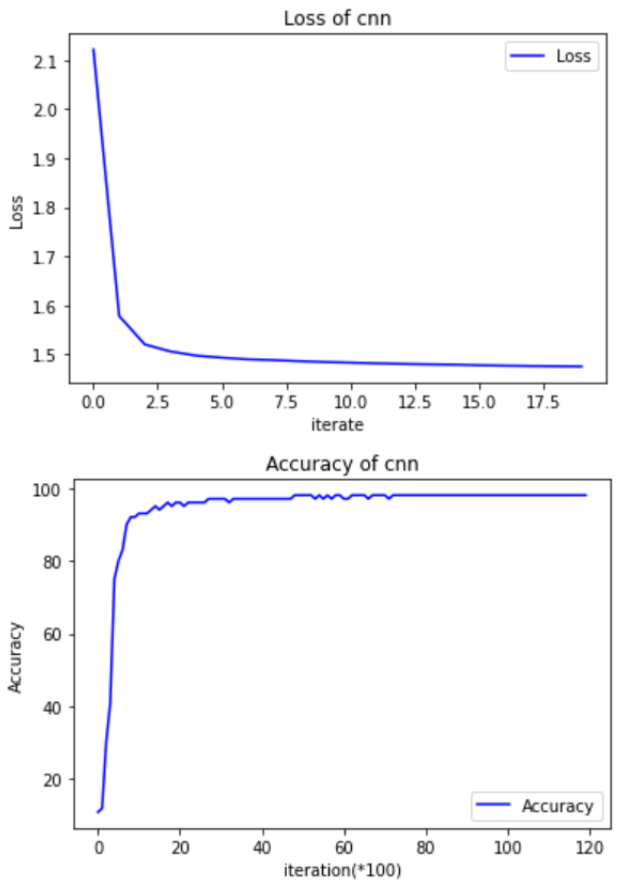

Figure 8: MNIST with CNN and epoch=20

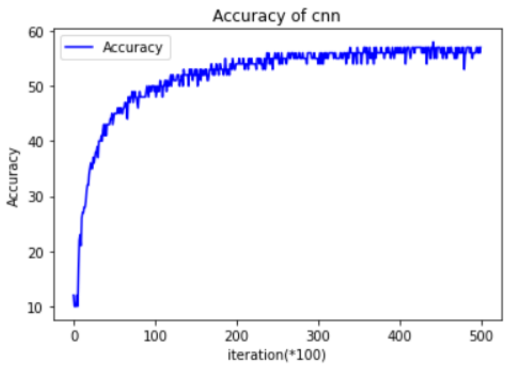

Figure 9: CIFAR with the same CNN and epoch=100

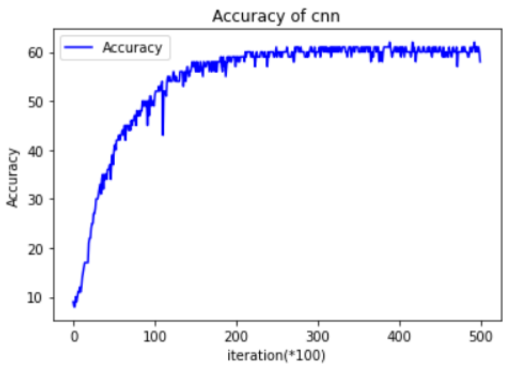

Figure 10: Experiment 1