Assignment2

Tong Zhang *4/22/2022*

0.1 Part I: PyTorch MLP

Run the part1.ipynb.

0.1.1 task1

I implement pytorch_mlp.py and pytorch_train_mlp.py.

We have implemented the structure in numpy and this task is easy because what we need to do is just use torch api.

However,I find that if I do not use nn.ModuleList but list,there will be a problem.I know this after reading many blogs.

Here is a distribution of make_moons.

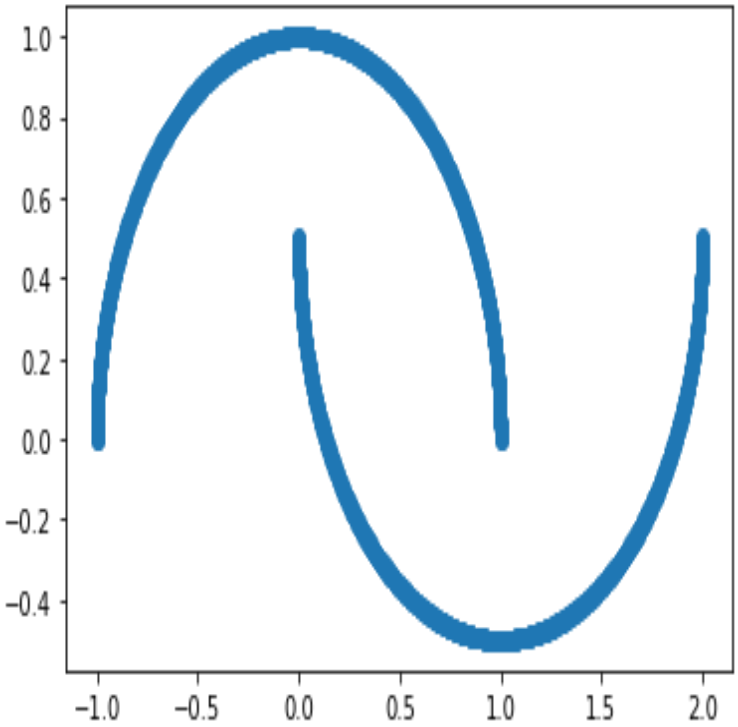

Fig. 1. make moon distribution

0.1.2 task2

I test both of two models on make moons dataset and make circle dataset with SGD, MAX EPOCHS= 100 and EVAL FREQ= 10. Below is the loss and accuracy of the torch_mlp model on the make_moon dataset.

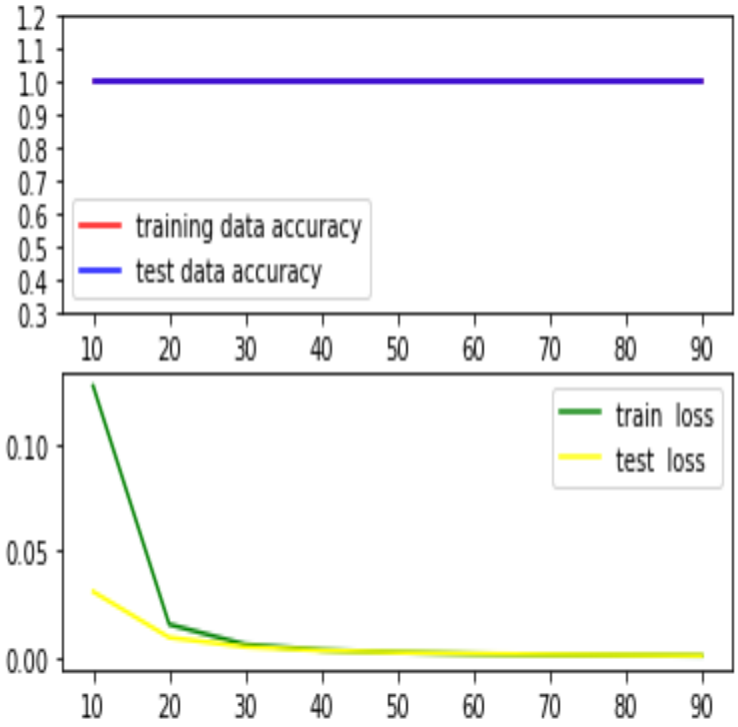

Fig. 2. performance of torch mlp model on the make moon dataset

Below is the loss and accuracy of the numpy_mlp model on the make_moon dataset.

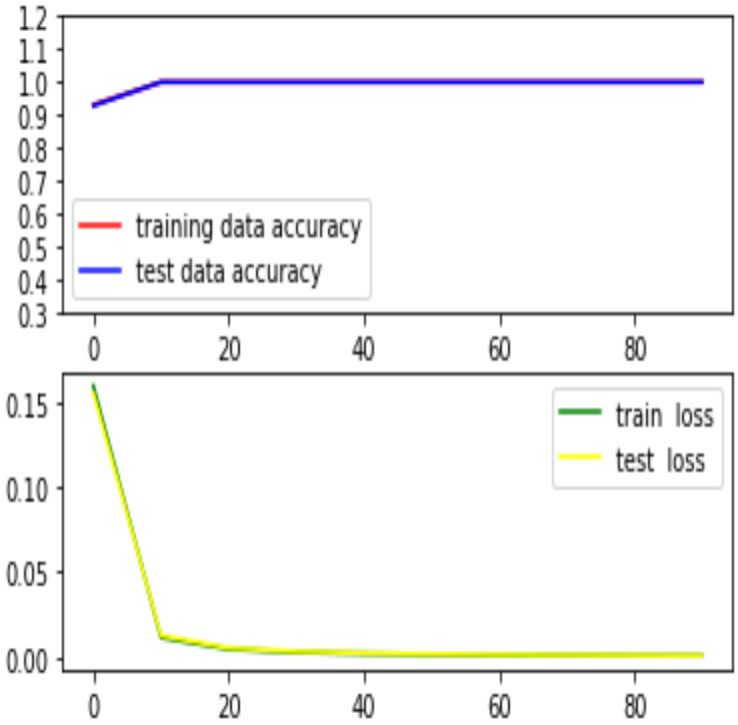

Fig. 3. performance of numpy mlp model on the make moon dataset

It is easy to see that both methods still have the same effect.The loss's shapes are the same and both quickly get 100% accuracy.

I also try make circle dataset.There is a big circle in another big circle.Below is the loss and accuracy of the torch_mlp model and numpy_mlp model on the make_circle dataset.

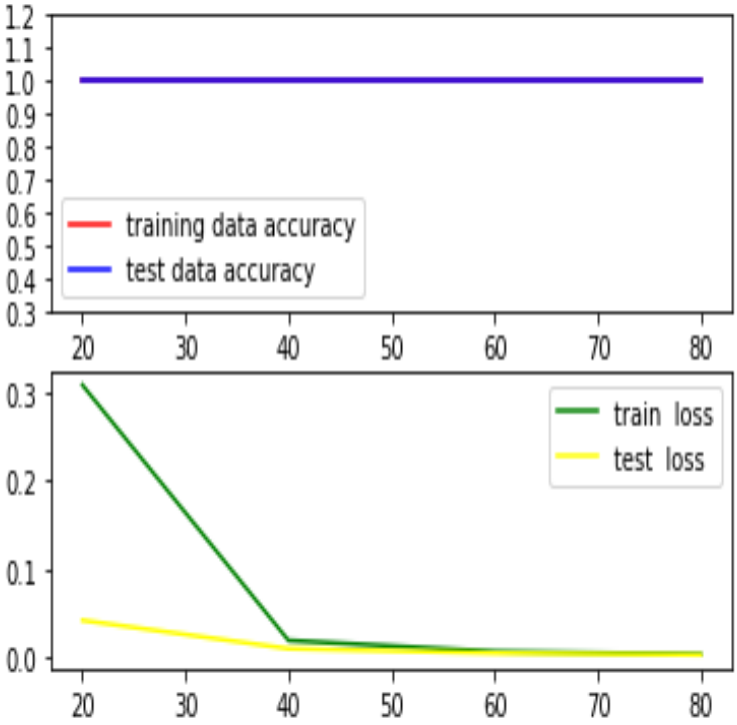

Fig. 4. performance of torch mlp model on the make circle dataset

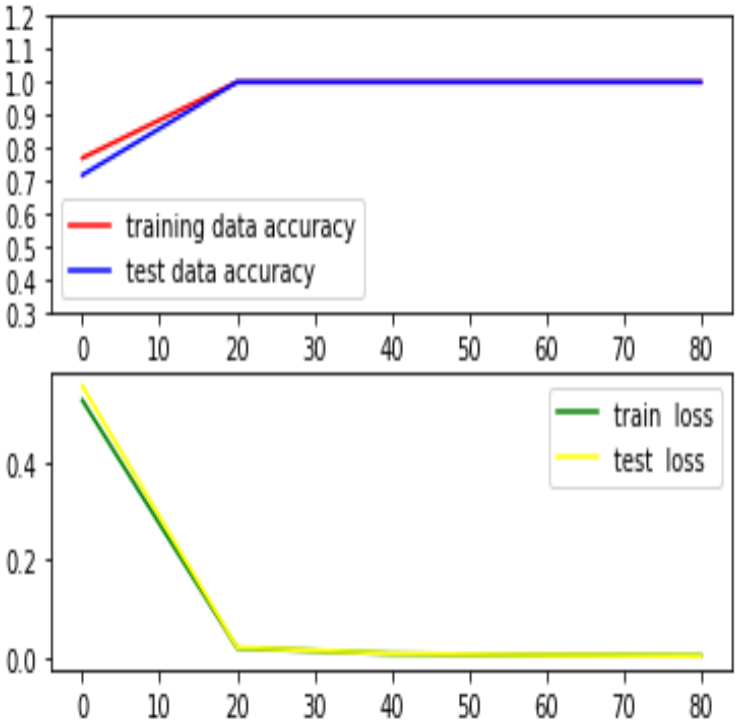

Fig. 5. performance of numpy mlp model on the make circle dataset

The result is still the same.Both of losses has similar shape.

0.1.3 task3

For this part, I wrote another file called torch_cifar.py and refined the MLP structure and training process. I do these improvements as below. 1. I use two hidden layers and each has 100 nodes. 2. I add dropout to the structure for better results. I need to use $os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'$or there will be bugs.

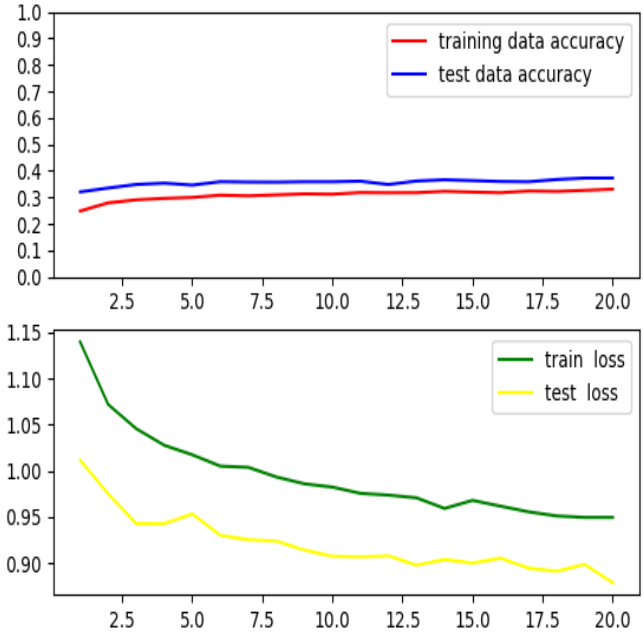

Fig. 6. performance of torch mlp model on the cifar10 dataset

As we can see,the result of mlp is bad .After 20 epoch training,its accuracy attains 40%.The test accuracy is highly lower than training data meaning that the model is overfit.

0.2 Part II: PyTorch CNN

run part2.ipynb

0.2.1 task1

1. order: convolution layer+ batch norm layer+ activation function.

2. Use torch.nn.Conv2d() to initialize the convolution layers and torch.nn.BatchNorm2d for batch norm layers. The size of batch norm layers should be the same as the output size of the before convolution layers.

3. The max pooling's size is (3, 2, 1). And the size of last linear layer should be (512 , 10).

4.Calculate the accuracy for each class and the total accuracy for all ten classes.

0.2.2 task2

Below is the loss and accuracy of the cnn model on the cifar10 dataset.

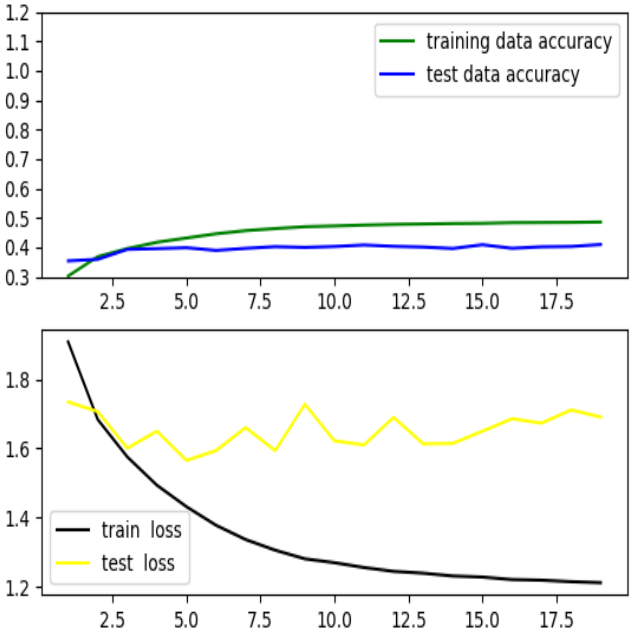

Fig. 7. performance of cnn model on the cifar10 dataset with default parameters

As we can see,its performance is much better than mlp.After 20 epoch training,its accuracy attains 50%. However, the test accuracy is still highly lower than training data meaning that the model is overfit.

0.3 Part III: PyTorch RNN

run part3.ipynb

0.3.1 task1

Complete the vanilla_rnn.py. 1. Use three linear layer hx,hh,oh 2. Use tanh to activate the sum of the output of hx and hh 3. Input last linear layer and a softmax

0.3.2 task2

Below is my experiment when input size=5,10,15

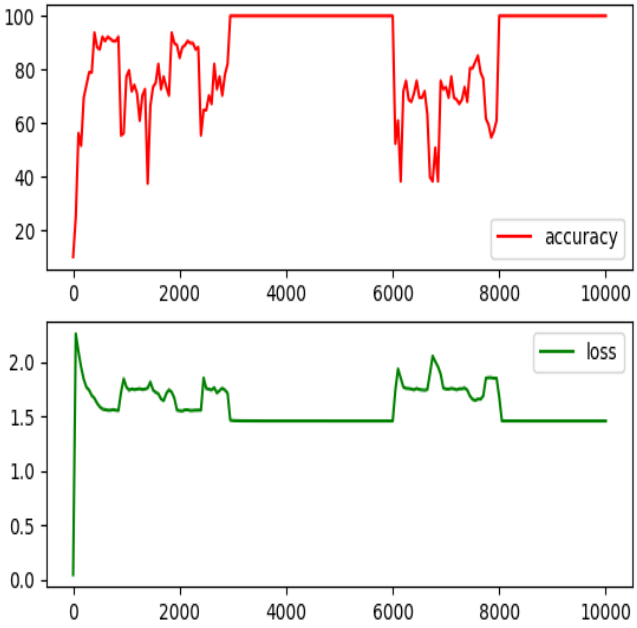

Fig. 8. performance of vrnn when inputsize =5

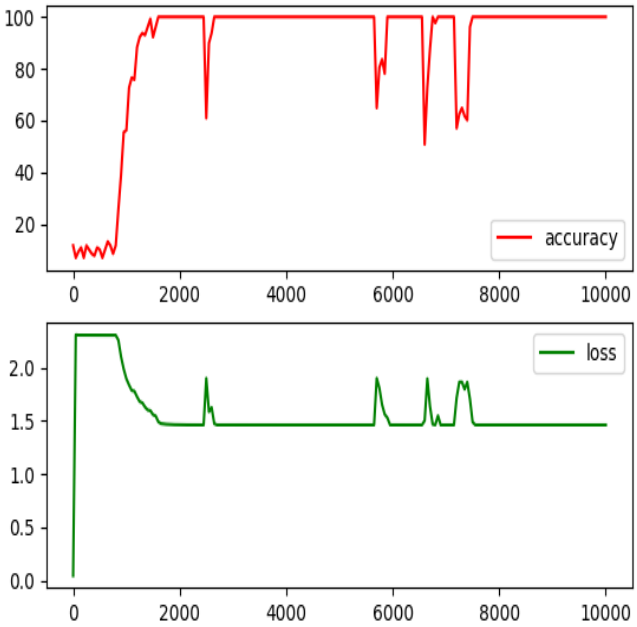

Fig. 9. performance of vrnn when inputsize =10

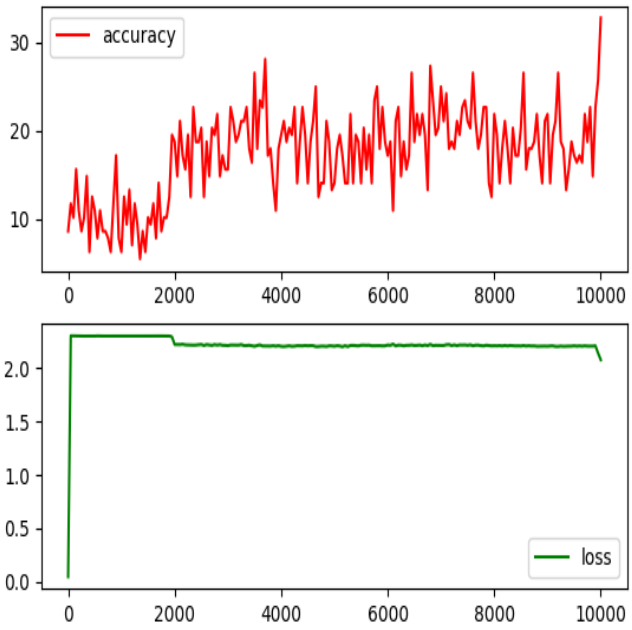

Fig. 10. performance of vrnn when inputsize =15

From the above figures, we can easily see that when the length is 5, our model can achieve the best result where the curves are smooth and it reached 100% accuracy quickest