Assignment 3 Report

Course: MSBD 5002 Data Mining Course

Name: YANG Rongfeng

Data: 2020-04-26

1 Install and Learn Pytorch

- Install Anaconda
- Create a virtual environment in conda

```
# conda create -n (env-name) python=(version) [packages]
conda create -n pytorch python=3.7 anaconda
```

- Config Tsinghua channel to add up speed

```
# add channels
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch

# show the channels info
conda config --show channels
```

- Choose proper pytorch version by your computer on pytorch page
- run the command to install pytorch

```
conda install pytorch==1.2.0 torchvision==0.4.0 cudatoolkit=10.0 -c
pytorch
or
conda install pytorch==1.2.0 torchvision==0.4.0 cudatoolkit=10.0 -c
pytorch
```

Finally I successfully install the pytorch with cuda 10.0 on my computer.

Check if the cuda configuration available. I get True.

```
torch.cuda.is_available()
True
```

2 Train NN Model on Bi-Class Datasets

I use nn modules in pytorch to build a sequential neural network.

```
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
    torch.nn.Sigmoid()
).to(device)
```

The last layer of the network is a Sigmoid functions which squeeze the results from R to [0,1]. The output shows the probability of the positive class (0.5 as boundary). I use the Binary Cross Entropy Loss function as the loss function:

```
loss_fn = torch.nn.BCELoss()
```

It's expression is as follows:

$$l(x,y) = L = \{l_1, \cdots, l_N\}^T, l_n = -w_n[y_n \cdot \log x_n + (1-y_n) \cdot \log(1-x_n)]$$

In this task, we are required to build a single layer network. We should find the best H for every dataset. Hence, I use K-Folder Cross Validation method to help me do this. The detailed information please see the generate_k_folders() function in my source code. At each dataset partitions, the results are preserved in res_table and after K times adding, it takes the average by k.

```
k_folders = generate_k_folders(read_bi_data(breast_cancer_data), k)
for index in range(k):
    print("K-Folder index = %s" % index)
    res_table += train_bi_nn_model(k_folders[index], H_list, device, lr, iteration)
return res_table / k
```

In training process, at each iteration, I would compute the current model's accuracy on the validation set and save the best accuracy and the best model. At last, after all iterations done, we get the best model and we use it to predict on testing dataset.

```
if accuracy > best_accuracy_val:
    best_accuracy_val = accuracy
    best_iteration = t
    best_model = model
```

For the optimizer, I use SGD

```
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

In this process, I also save the loss history and validation accuracy history so I can draw the curve for better analysis. Next, I would show my experiment results on five binary class datasets.

2.1 Breast cancer dataset

This dataset has 10 features, 547 training data and 136 testing data. I set learning rate to 1e-2 and I find the best H is 3.

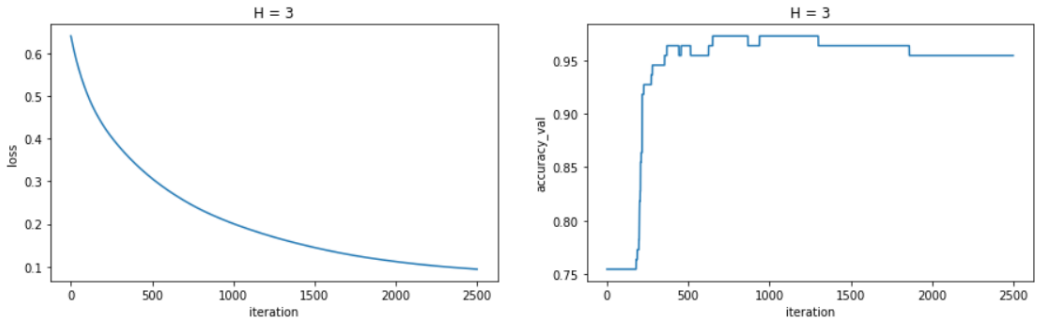

Figure 1. The loss curve and val-acc curve on breast cancer dataset (H=3)

H	Val_Accuracy	Test_Accuracy	AUC	Precision	Recall	F1	Training Time
1	0.892	0.901	0.892	0.838	0.866	0.848	9.782(s)
2	0.973	0.963	0.997	0.962	0.957	0.96	9.643(s)
3	0.973	0.969	0.997	0.967	0.966	0.966	9.4(s)
4	0.965	0.966	0.998	0.964	0.962	0.963	10.327(s)
5	0.963	0.968	0.997	0.965	0.964	0.965	11.125(s)
6	0.965	0.968	0.997	0.965	0.964	0.965	11.011(s)
7	0.973	0.968	0.998	0.965	0.964	0.965	10.401(s)

Table 1. The Comparison of performance by different H

2.2 Digit dataset

This dataset has 64 features, 615 training data and 153 testing data. This number of features of this dataset is the biggest among five datasets. Therefore, we can infer that the H should choose bigger. I make the learning rate smaller, setting it to 2e-3 and I find the best H is 10.

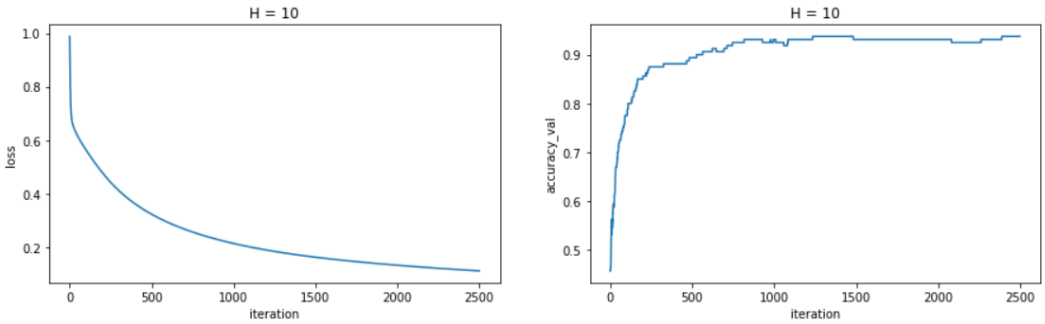

Figure 2. The loss curve and val-acc curve on digit dataset (H=10)

H	Val_Accuracy	Test_Accuracy	AUC	Precision	Recall	F1	Training Time
5	0.914	0.859	0.987	0.818	0.8	0.793	9.569(s)
6	0.947	0.931	0.998	0.951	0.903	0.92	9.327(s)
7	0.927	0.935	0.997	0.956	0.908	0.924	9.11 (s)
8	0.941	0.929	0.998	0.951	0.9	0.918	11.058(s)
9	0.919	0.922	0.997	0.948	0.89	0.907	11.388(s)
10	0.956	0.94	0.997	0.958	0.915	0.93	10.615(s)

Table 2. The Comparison of performance by different H

2.3 Diabetes dataset

This dataset has 8 features, 615 training data and 153 testing data. I set learning rate to 5e-2 and I find the best H is 4.

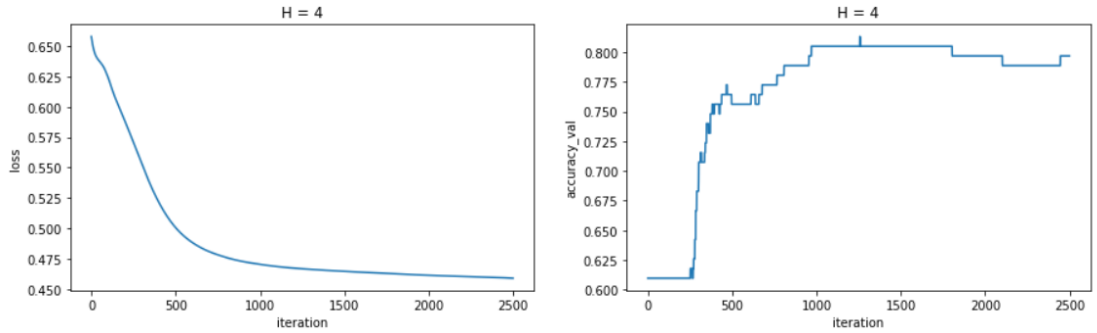

Figure 3. The loss and val-acc curve on diabetes dataset (H=4)

H	Val_Accuracy	Test_Accuracy	AUC	Precision	Recall	F1	Training Time
1	0.926	0.9	0.897	0.833	0.867	0.846	9.412(s)
2	0.973	0.965	0.997	0.961	0.961	0.961	9.183(s)
3	0.967	0.968	0.997	0.964	0.966	0.965	9.584(s)
4	0.971	0.969	0.997	0.966	0.967	0.966	9.897(s)
5	0.973	0.968	0.997	0.965	0.965	0.965	9.909(s)
6	0.976	0.965	0.997	0.961	0.962	0.961	10.188(s)
7	0.971	0.966	0.997	0.963	0.963	0.963	10.127(s)
8	0.971	0.968	0.997	0.963	0.966	0.965	9.489(s)

Table 3. The Comparison of performance by different H

2.4 Iris dataset

This dataset has 4 features, 120 training data and 30 testing data. I set learning rate to 1e-2 and I find the best H is 3.

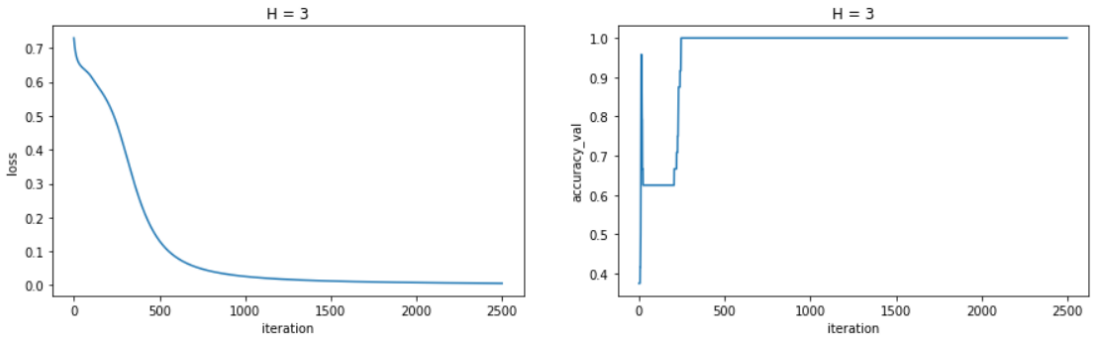

Figure 4. The loss and val-acc curve on iris dataset (H=3)

H	Val_Accuracy	Test_Accuracy	AUC	Precision	Recall	F1	Training Time
1	0.886	0.903	0.891	0.837	0.87	0.85	9.465(s)
2	0.962	0.965	0.997	0.963	0.959	0.961	9.192(s)
3	0.963	0.966	0.997	0.964	0.962	0.963	9.107(s)
4	0.965	0.963	0.998	0.962	0.957	0.96	9.198(s)

Table 4. The Comparison of performance by different H

2.5 Wine dataset

This dataset has 13 features, 142 training data and 36 testing data. I set learning rate to 1e-3 and I find the best H is 6.

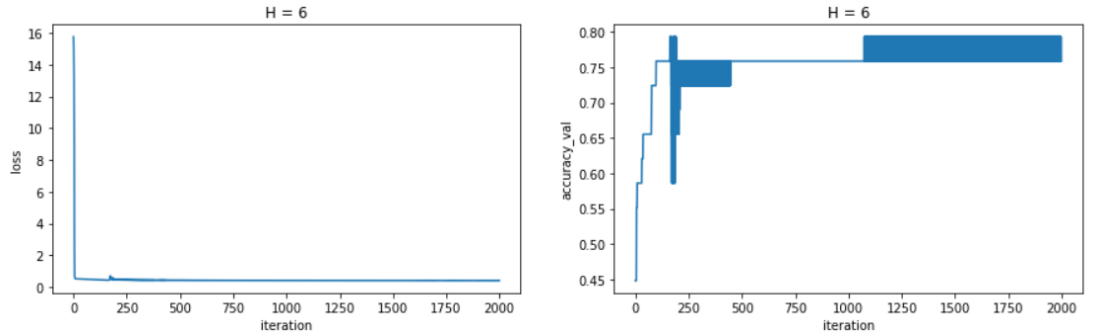

Figure 5. The loss and val-acc curve on wine dataset (H=6)

H	Val_Accuracy	Test_Accuracy	AUC	Precision	Recall	F1	Training Time
1	0.715	0.709	0.827	0.643	0.674	0.6	9.564(s)
2	0.822	0.781	0.979	0.678	0.691	0.66	9.17(s)
3	0.758	0.744	0.869	0.662	0.638	0.601	9.814(s)
4	0.864	0.841	0.979	0.901	0.778	0.791	10.202(s)
5	0.734	0.751	0.929	0.668	0.648	0.604	9.864(s)
6	0.861	0.834	0.997	0.902	0.765	0.773	9.523(s)
7	0.824	0.807	0.964	0.787	0.728	0.723	9.795(s)
8	0.87	0.815	0.996	0.892	0.737	0.748	9.733(s)
9	0.83	0.803	0.997	0.886	0.721	0.73	9.45(s)
10	0.813	0.799	0.997	0.885	0.715	0.716	10.286(s)

Table 5. The Comparison of performance by different H

3 Train NN Model on Multi-Classification Dataset

The structure of my neural network model is:

```
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, L1), # hidden layer1 = L1
    torch.nn.ReLU(),
    torch.nn.Linear(L1, L2), # hidden layer2 = L2
    torch.nn.ReLU(),
    torch.nn.Linear(L2, 10), # output probability on 10 classes
).to(device)
```

Because the dataset has 10 classes, the ouput dimension is set to 10 and L1, L2 are two hidden layers we need to tune.

In this task, I use Cross Entropy Loss function in Pytorch as my model's loss function. It's a combination of LogSoftmax and Negtive Log Likelihood.

```
loss_fn = torch.nn.CrossEntropyLoss()
```

It's mathematical formula is described as:

$$loss(x, class) = -\log\left(\frac{\exp(x[class])}{\Sigma_j \exp(x[j])}\right)$$

The K-Folders Cross Validation Method is the same as before which are used to help me choose the best combination of L1 and L2.

Again, I use SGD as the optimizer of my model and the learning rate is set to 1e-4.

```
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

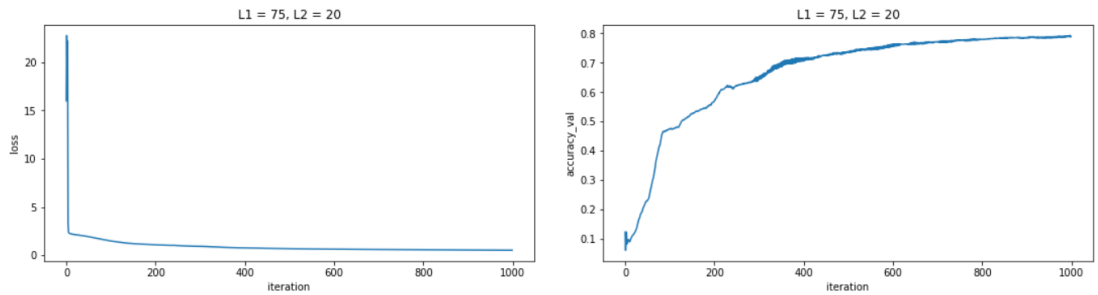

Figure 6. The loss and val-acc curve on multi-class dataset (L1=75, L2=20)

Analyze the model performance with different parameters in table 6, we can find that model with L1 = 75, L2 = 20 performs best. Therefore, we choose L1 = 75 and L2 = 20.

L1	L2	Val Accuracy	Test Accuracy	Training Time
50	10	0.454	0.221	11.726(s)
50	15	0.601	0.298	11.976(s)
50	20	0.704	0.352	12.125(s)
75	10	0.413	0.208	12.466(s)
75	15	0.615	0.304	12.493(s)
75	20	0.691	0.346	12.867(s)
100	10	0.529	0.256	13.829(s)
100	15	0.585	0.282	13.956(s)
100	20	0.651	0.32	13.946(s)

Table 6. The Comparison of performance by different combination of L1 and L2