Ve593
Jia Shi
Prof. Weng
716370990049

In this project, I construct a neural network to realize the classification of hand-written picture in the Mnist database.

The level of network is [784, 30,10], with three activation function sigmoid, ReLU and tanh. Beside that, I also implement L1 and L2 regularization and early stopping for better performance.

In general, the testing accuracy is slightly lower to equal to the training accuracy, I didn't encounter the problem of overfitting or underfitting in my network.

Worth to mention, the program will stop when the accuracy reach 93% thus it will not finish all the iteration most of the time.
Plus, in my program, the time of iteration stands for the continue increment of a model(when it drop, it print restart, and the iteration start from 0 again). When it restart for 5 times, it return the final result.

At the end, I receive an average training accuracy rate of 92.6% and testing accuracy rate of 91.5% over 10 performance, and 8 of them are greater than 90%. It's pretty stable.

In my final network, I have number of node in each level [784, 30, 10], with activation function [None, sigmoid, sigmoid], batch size 10, number of iteration 20, and learning rate 0.5, with early stopping but NO L1 and L2 regularization. I will explain how I choose those hyper parameter in the next part.

For batch size, in my network, if the batch_size is so big, like 100, the network will start training at a really high accuracy (like 75%), but will not increase much at the end. Thus, my choice of batch size allow me to start at 70% for the first time update and reach 85-95%eventually.

For number of iteration, it doesn't really matter in the training of network. Enough iteration is fine. I stop the run-in when it reach 93 percent.

For learning rate, if the learning rate is too low, my accuracy rate will be stuck in a local minima and never get a good score eventually. If I use 0.01, it will only be 45-55% eventually. If it's too big, like 1, the score will drop frequently, and also never excel 70%. Thus, I choose 0.1.

For L1 regularization, it's highly unstable and very slow, the performance is good, it will start at 85% right away, but it's very slow. So I deleted it eventually.

For L2 regularization, I found that for such a small and shallow network, it will be unnecessary to add regularization, and when I did, the result is not well. It reach 77 percent at first and stop at 78 percent all the time. Thus I decide to delete it at the end.

For activation function, sigmoid perform the best and it's more stable. But sometime(rarely) when the number of training increase, it may cause "overflow error", because it lead to the Vanishing Gradient problem and it's hard to find a solution.

For tanh, the accuracy rate is very low at first, like 40%, but increase soon. However, the final rate is very low, like 50%. When I choose to use L2 regularization, it start from 70% right away, and reach 85% eventually. It's not stable in general.

For ReLU, when one element is set to 0, it will be 0 forever and will never be activated again. So number will become [nan] for some reason. When the learning rate is huge, it only have around 40% of training and testing accuracy eventually.
LeakLU, however, encounter the same problem for becoming [nan].

For arctan, it start at around 70 percent, and the rate of increasing is very slow, and become stable at around 80%, so I didn't choose it eventually.

For my early stopping, I set up a timer, when the accuracy rate drop after each training for 10 times, early stopping activate. And it keep the accuracy from dropping.

```
JiadeMacBook-Air:Project3 elvishelvis$ python3 experiment.py
0.7597
0.7871
0.8109
restart
0.8537
restart
0.893
0.8989
restart
0.9328
Great!
The training accuracy is :
0.9356
The testing accuracy is :
0.9156
```