Shai kikozashvily
206202384

Theoretical part:

$$\hat{y}_i = \text{Softmax}(x)_i = \frac{\exp^{f(x;w)_i}}{\sum_{j=0}^{n} \exp^{f(x;w)_j}}.$$

For simplicity set $SM(x)_i = \frac{e^{f(w,x)_i}}{\sum_{j=0}^{n} e^{f(w,x)_j}}$

Find $\frac{\partial \hat{y}}{W}$*:*

$$\frac{\partial \hat{y}_i}{W_i} = \frac{e^{f(w,x)_i} \cdot f'(w,x)_i \cdot \sum_{j=0}^{n} e^{f(w,x)_j} - e^{f(w,x)_i} \cdot e^{f(w,x)_i} \cdot f'(w,x)_i}{\left(\sum_{j=0}^{n} e^{f(w,x)_j}\right)^2} =$$

$$\frac{f'(w,x)_i \cdot e^{f(w,x)_i}}{\sum_{j=0}^{n} e^{f(w,x)_j}} - \frac{\left(e^{f(w,x)_i}\right)^2}{\sum_{j=0}^{n} e^{f(w,x)_j}} = SM(x)_i(f'(w,x)_i - SM(x)_i)$$

Let $k \neq i$:

$$\frac{\partial \hat{y}_i}{W_k} = \frac{0 - e^{f(w,x)_i} \cdot e^{f(w,x)_k} \cdot f'(w,x)_k}{\left(\sum_{j=0}^{n} e^{f(w,x)_j}\right)^2} = \frac{e^{f(w,x)_i}}{\sum_{j=0}^{n} e^{f(w,x)_j}} \cdot \frac{f'(w,x)_k \cdot e^{f(w,x)_k}}{\sum_{j=0}^{n} e^{f(w,x)_j}}$$

$$= SM(x)_i \cdot SM(x)_k \cdot f'(w,x)_k$$

$$\Rightarrow \frac{\partial \hat{y}}{W} = \begin{cases} SM(x)_i(f'(w,x)_i - SM(x)_i) & , k = i \\ SM(x)_i \cdot SM(x)_k \cdot f'(w,x)_k & , k \neq i \end{cases}$$

$$\frac{\partial L(\hat{y}, y)}{W} = \frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{W}$$

■

Practical part:

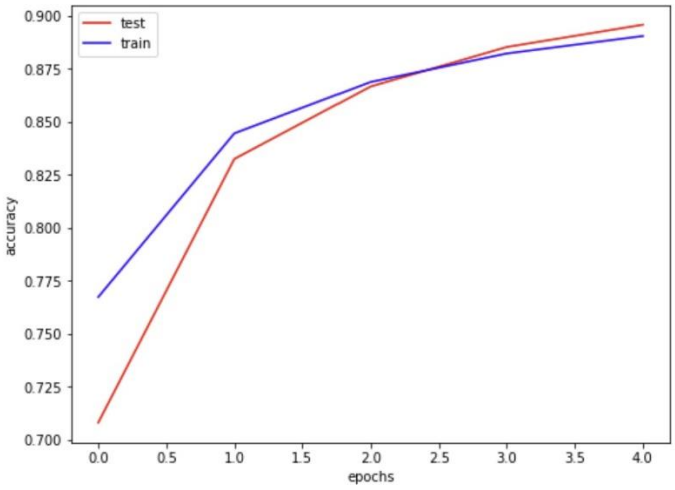
Q1:

Network architecture:
Input layer: size 784 (flatten a 28*28 pixels picture). activation function: ReLu.
First hidden layer: size 128, activation function: ReLu.
Second hidden layer: size 64, activation function: SoftMax..
Output layer: size 10 since there are 10 classes.

The loss function that is used for the learning is the MSE which yielded nice results.

First, I tried to train the network with the sigmoid function as the activation instead the ReLu but it took a lot more time (about 4 times more epochs) to achieve a good accuracy on the test set.

Moreover, a 0.01 learning rate caused a faster converging than 0.05 and 0.1.
To sum it up, here is the plot of test accuracy and train accuracy as function of number of epochs: (code added to the train script as a comment at the end)



Test accuracy is ~ 89%.

Q2:

In this part I used the same architecture as Q1 net, but with torch built in neural network and it's functions. The output size is 2 since there are 2 labels (0 and 1). In addition, the loss function is the cross entropy loss.
The leaning rate also stayed the same (0.01), but I had to increase the number of epochs to 20 so I would be able to achieve a loss of ~0 on the train set.
After 20 epochs it can be clearly seen the overfitting to the random variables. The network learned these 2 classes very well (too well), and when checked the loss on the test set the gap representing overfitting can be seen. Given the test set, the network could not perform well since it did not really learn something in the training phase.
Here is the graph describing the loss as function of number of epochs: