ML2 HW1

Theoretical Part

$$\hat{y}_i = Softmax(x)_i = \frac{e^{f(x;w)i}}{\Sigma e^{f(x;w)_j}}$$

$$Cross - entropy = C_i = -y_i \log(\hat{y})$$

Part 1 derivative of SoftMax:

First, we will find the logarithmic derivative of the SoftMax function:

By finding the log of the SoftMax function:

$$\log(\hat{y}_i) = \log\left(\frac{e^{f(x;w)i}}{\Sigma e^{f(x;w)_j}}\right) = f(x;w)_i - \log(\Sigma e^{f(x;w)_j})$$

Now we will find the derivative with and without using the log we've just found:

$\frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} = \frac{1}{\hat{y}_i} * \frac{\partial \hat{y}_i}{\partial f(x;w)_j}$ by rearranging the formula we get: $\frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} * \hat{y}_i = \frac{\partial \hat{y}_i}{\partial f(x;w)_j}$

$$\frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} = \frac{\partial f(x;w)_i - \log(\Sigma e^{f(x;w)_j})}{\partial f(x;w)_j} = \frac{\partial f(x;w)_i - \log(\Sigma e^{f(x;w)_j})}{\partial f(x;w)_j}$$

$$= \frac{\partial f(x;w)_i}{\partial f(x;w)_j} - \frac{\partial \log(\Sigma e^{f(x;w)_j})}{\partial f(x;w)_j}$$

We can see that :

$$\frac{\partial f(x;w)_i}{\partial f(x;w)_j} = \begin{cases} 1 & if\ i = j \\ 0 & otherwise \end{cases}$$

Thus:

$$\frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} = 1\{i = j\} - \frac{1}{\Sigma e^{f(x;w)_k}} * \left(\frac{\partial \Sigma e^{f(x;w)_k}}{\partial f(x;w)_j}\right) = 1\{i = j\} - \frac{1}{\Sigma e^{f(x;w)_k}} *$$

$$\frac{\partial\left(e^{f(x;w)_1} + e^{f(x;w)_2} + e^{f(x;w)_3} \ldots\ldots e^{f(x;w)_n}\right)}{\partial f(x;w)_j} = 1\{i = j\} - \frac{e^{f(x;w)_j}}{\Sigma e^{f(x;w)_k}}$$

By plugging the definition of SoftMax we get :

$$\frac{\partial \hat{y}_i}{\partial f(x;w)_j} = \hat{y}_i * \frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} = \hat{y}_i * (1\{i = j\} - \hat{y}_j) \ \ (*)$$

Part 2 derivative of Cross-Entropy:

$$Cross - entropy = C = -\Sigma y_i \log(\hat{y})$$

$$\frac{\partial C}{\partial f(x;w)_j} = -\frac{\partial \Sigma y_i \log(\hat{y}_i)}{\partial f(x;w)_j} = -\Sigma \frac{y_i}{\hat{y}_i} * \frac{\partial \log(\hat{y}_i)}{\partial f(x;w)_j} =_{(*)} - \Sigma y_i (1\{i = j\} - \hat{y}_j) = \Sigma y_i * \hat{y}_j - \mathrm{y_j}$$

$$= \hat{y}_{\mathrm{j}} \Sigma \mathrm{y_i} - y_j = \hat{y}_j - y_j$$

Or in general:

$$\frac{\partial C}{\partial f(x;w)} = \hat{y} - y$$

Practical Part

1. Using the example we were given in the third tutorial as a base I've added a few tweaks to the class itself and utility functions:

   The network is made of two layers with an activation function (**sigmoid**) between the first and the second layer and another one (**SoftMax**) after the second layer.

   I've set my loss function to be **Cross Entropy**

   The **hyper parameters** are size of batch, input size, number of classes, number of epochs in total and the learning rate.

   I've transformed the labels of the images to One Hot encoded vectors for the network.

   The hyper parameters I used are as follows:

   Input size(MNIST) : 784

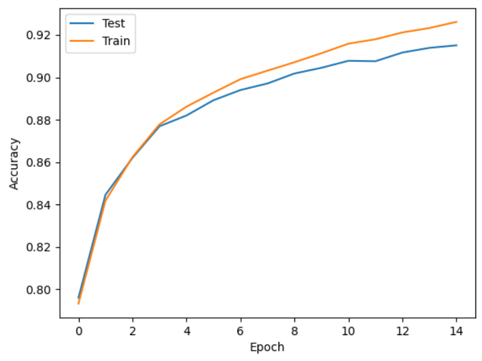
   Number of classes(MNIST):10

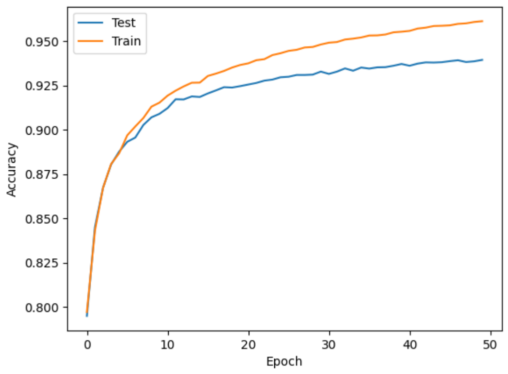
   Batch size: 128

   Learning rate: 0.22

   number of epochs in total:15

   I've "danced" around several learning rates between 0.1 and 0.3 but found a sweet spot in the rate above.





   As seen above I've reached an accuracy of above 75% quite early on.

   Furthermore running the algorithm for more than around 15 epochs won't conceive a good return of investment .

2. As requested in this segment I'll try to overfit data with randomly selected labels. we'd expect that overfitting would result in a high accuracy and low loss for the training data and the opposite for the test samples

   The hyper parameters I used are as follows:

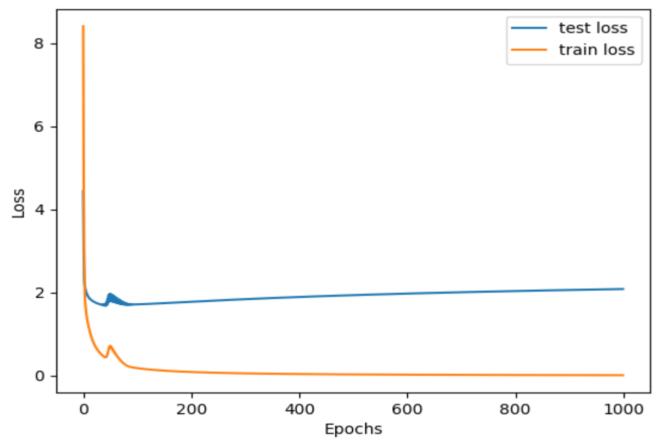
   Input size(MNIST) : 784

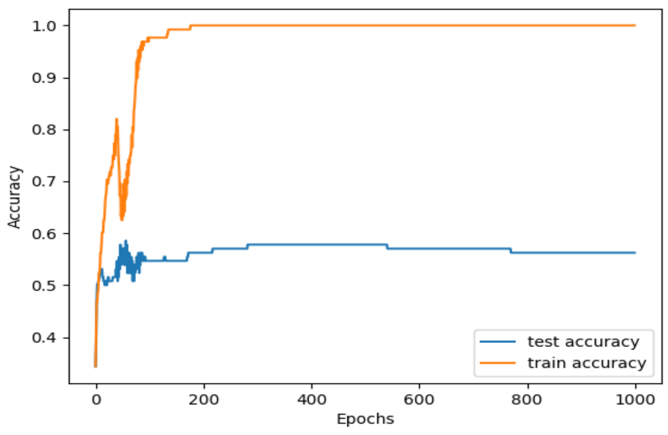
   Number of classes(MNIST):10

   Batch size: 128

   Learning rate: 0.2

   number of epochs in total:1000(to reach train loss of around 0)





Mean test loss: 1.9286660822629929 and we've reached an accuracy of 50% because there are only two labels.