**Deep Learning – Assignment 1**

**Part 1:**

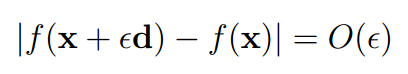
1. We created functions to compute the loss “soft-max regression” and its gradient with respect to and the biases ().

We then tested our implementation using a gradient test:

A comparison of a graph

Description automatically generated with medium confidence

The blue plot implements the formula:



While the orange plot implements the formula:

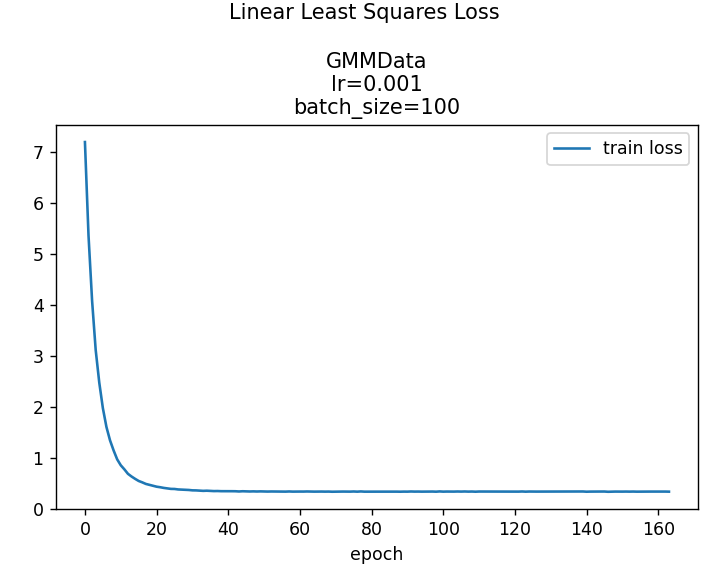


We can see that the test was successful since the blue plot behaves like while the orange plot behaves like .

For example, the blue plot starts at about while the orange plot starts at about .

1. We created functions for minimizing an objective function using SGD.

We then verified that our optimizer works on a small least squares example:



As can be seen, the loss function decreases as the number of epochs increases, just as we expected. In addition, as the number of epochs goes to infinity, the loss function tends to roughly 0.35.

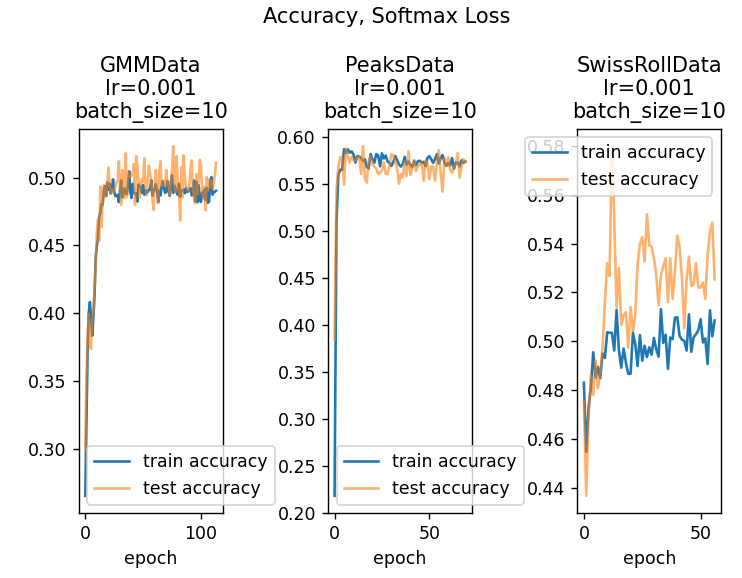
1. We ran our SGD minimization algorithm using the softmax function.

During the run, after each epoch, we checked the percentages of data classification – for both the training data and the test data.

We made sure to compute that metric only on random subsamples of the datasets.

We then tried multiple combinations of learning rates (0.01, 0.001, 0.0001) and batch sizes (10, 100, 1000).

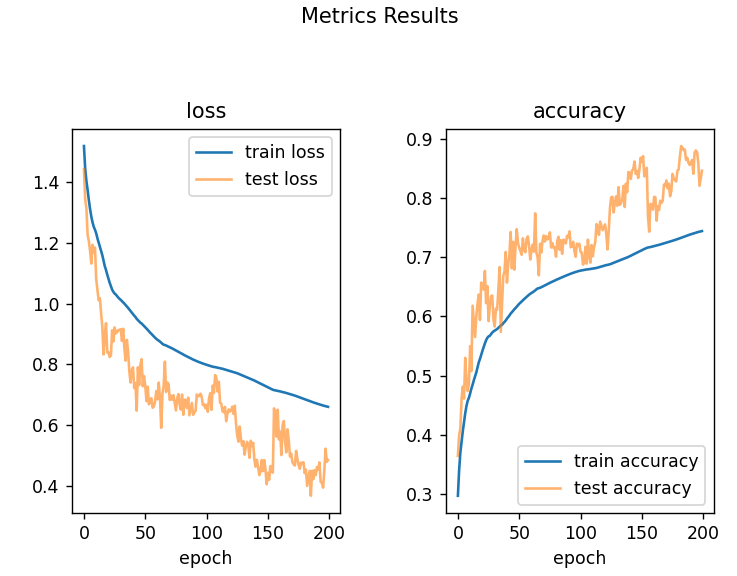
Here are the graphs of the best hyper-parameters that we’ve got:



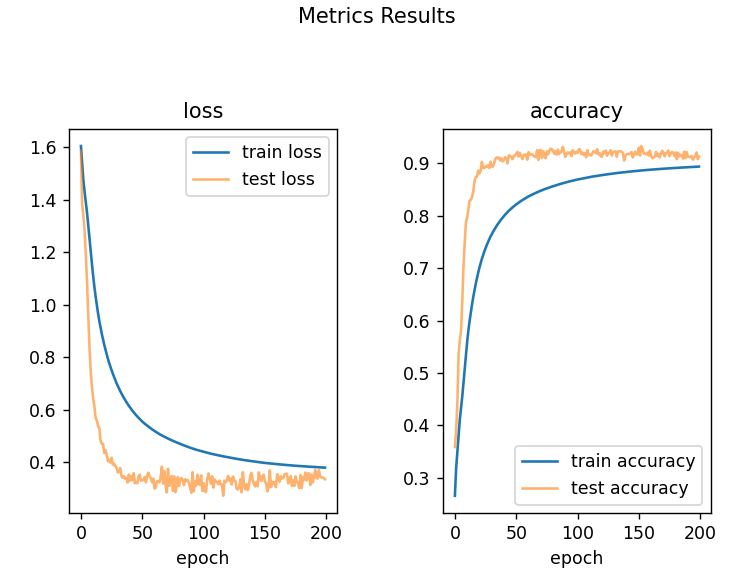
**Part 2:**

1. We wrote the code for the standard neural network, using tanh() as the activation function.

Here you can see the results of training a network with two inner layers:



Here is with one inner layer:



We then made sure that our implementation is correct using a gradient test:

1. We then extended our network and added residual layers.

Here are the results of training a network with regular layer (tanh), residual layer (tanh) and softmax layer:

