Assignment 1

Gal Meirom and Guy Green

In the assignment, we were asked to code and calculate some of the core features of fully connected neural networks and use the code we have written for classification tasks.

We will demonstrate our code, and our findings in the following pages, marked by the questions and clauses from the paper sheet.

Question 1:

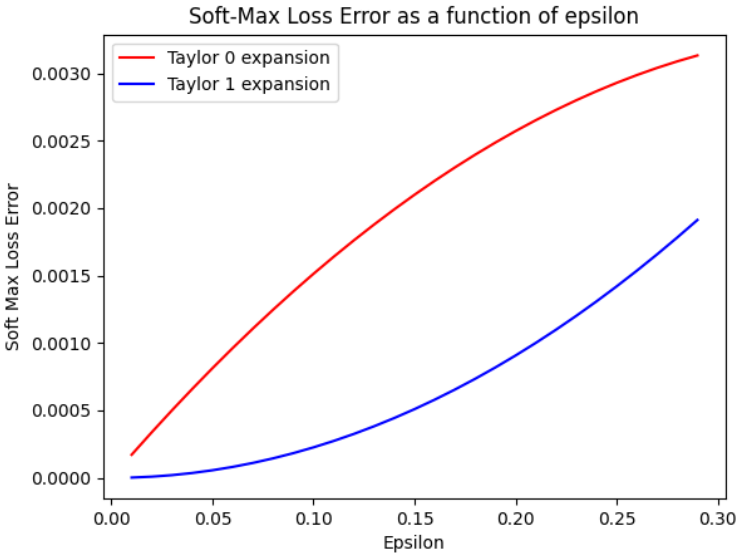
First clause

In the first clause, we were asked to write a code computing the loss function "Soft-max Regression", and its derivatives (Gradient) with respect to its weights matrix, W, and the bias vector, b.

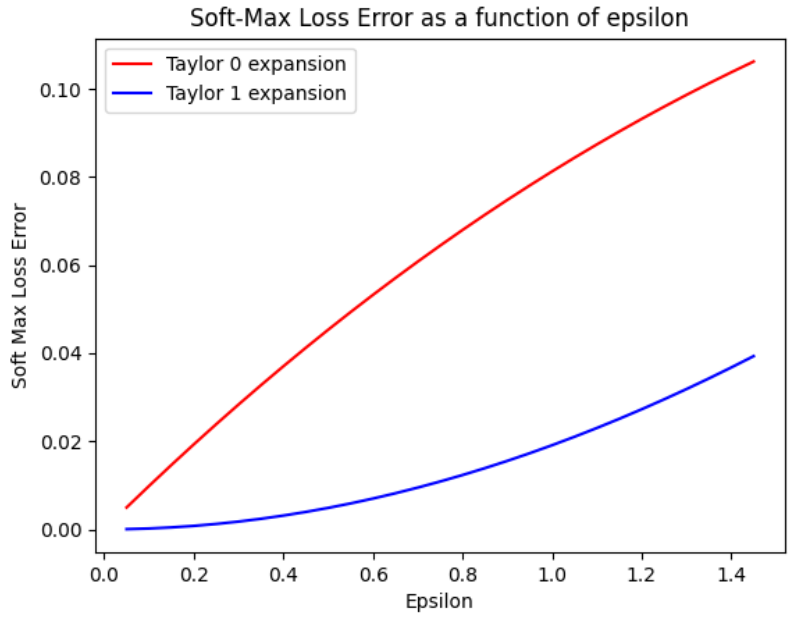
Shown below, our Gradient Test scores for the soft-max regression function, with respect to each of the data sets given to us.

Let it be known that the graphs are in corollary to the equations $|f(w+\epsilon d)-f(w)| \approx O(\epsilon)$ which is the Taylor approximation of order 0 of the function, and $|f(w+\epsilon d)-f(w)\ -\epsilon d^T \nabla_w f| \approx O(\epsilon^2)$ which is the Taylor approximation of order 1 if the function.

Data set "Swiss Roll":



Data set “Peaks”:

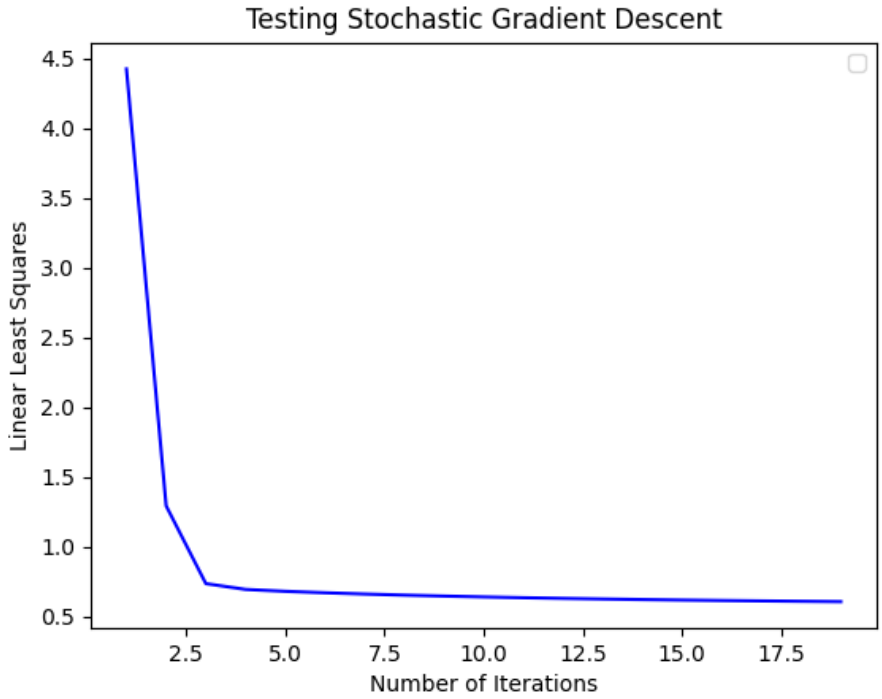


Data set “GMM”:

As you can observe, it is obvious that while the Taylor approximation of order 0 behaved linear with respect to epsilon, the Taylor approximation of order 1 behaved as a second-degree polynomial.

Second Clause

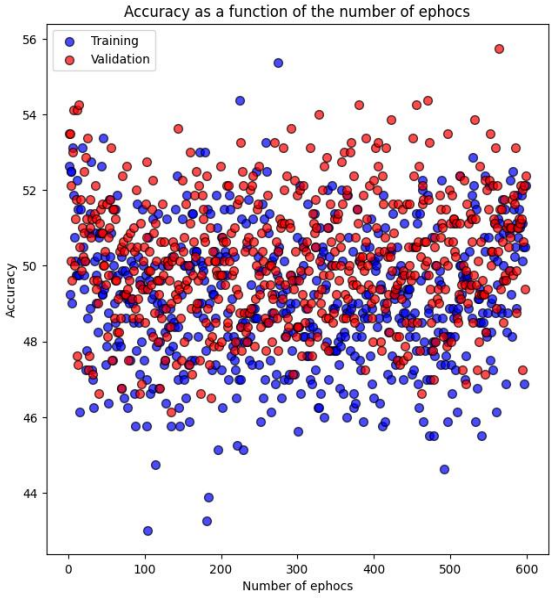
In this clause, we were asked to write a code minimizing an objective function using “Stochastic Gradient Decent” method we have seen in class, or SGD. As the code will be added as a zip file to the paper itself, we will demonstrate our optimizer with pictures it is scores on the different data sets we were given.

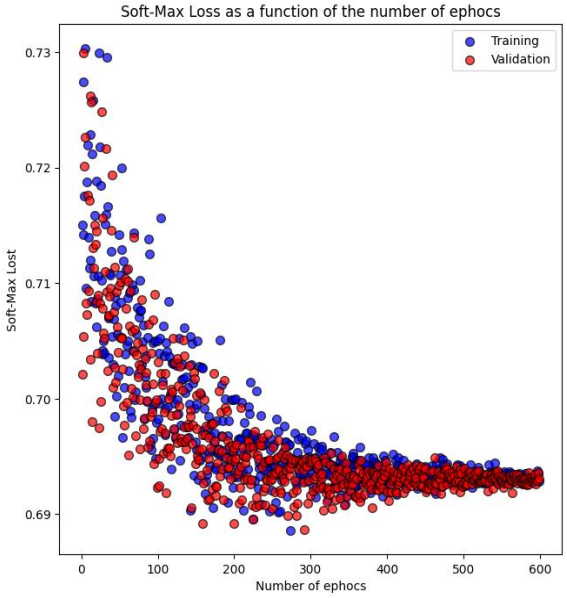


Third Clause:

In this clause, we were asked to use our SGD optimizer using soft-max regression as the target function. We have tried several options for the size of the randomly selected batches and found that sizes above 700 were not improving at a fast enough rate and that batches of sizes below 200 were too erratic and were not a good representation. Thus, we have landed on batches of size 500. Another parameter we have tested was the number of epochs, and found out that after 600 epochs, our model's results were not improving.

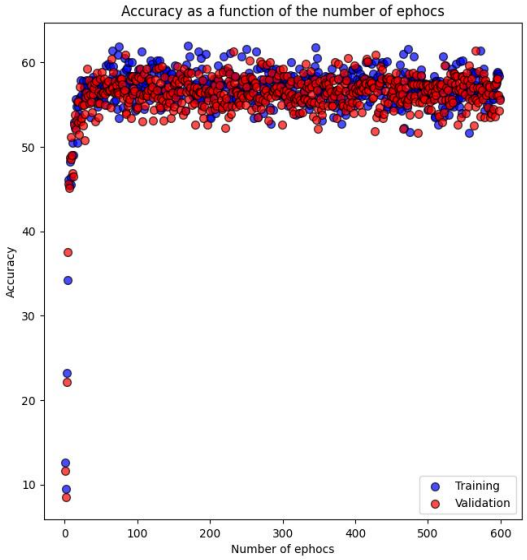
Data Set "Swiss Roll":

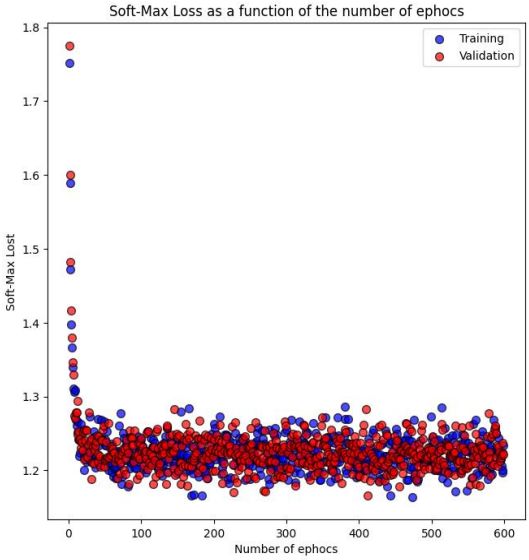




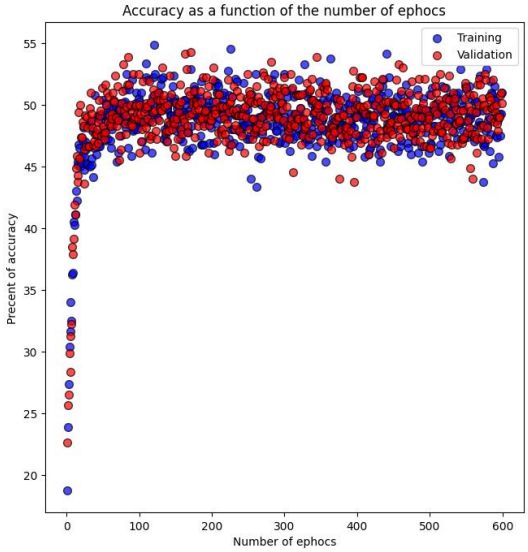
Observation – while our model managed to optimize and improve on the second and the third data sets, it hit a ceiling regarding the target function score on the first data set and did not manage to improve on its accuracy.

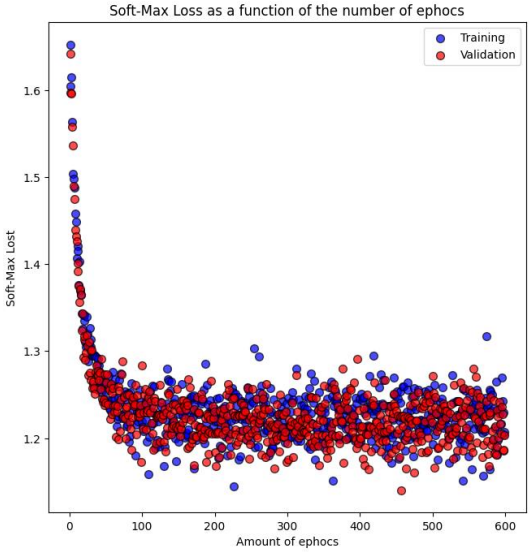
Data set "Peaks":





Data set “GMM”:


Accuracy as a function of the number of ephocs
Training
Validation
Precent of accuracy
Number of ephocs
55
50
45
40
35
30
25
20
0
100
200
300
400
500
600

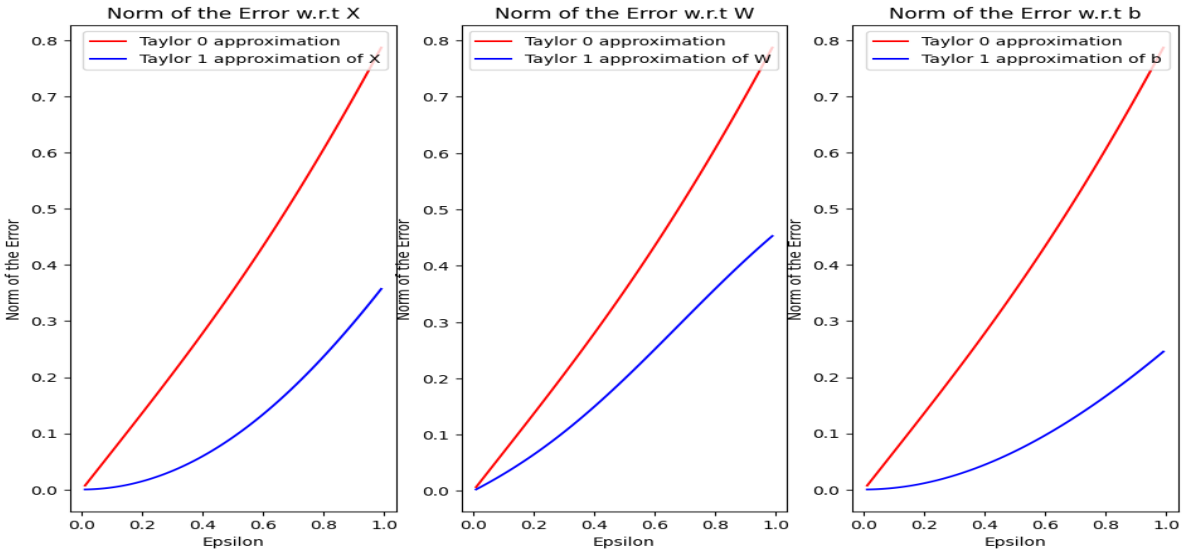

Soft-Max Loss as a function of the number of ephocs
Training
Validation
Soft-Max Lost
Amount of ephocs
1.6
1.5
1.4
1.3
1.2
0
100
200
300
400
500
600

Question 2

First Clause

In this clause, we were asked to write a layer function of a standard fully connected neural network, both its forward and backward pass. The Jacobian test for each of the layer's parameters, and the Transpose test for each of the variables are shown below:

The Jacobian test:



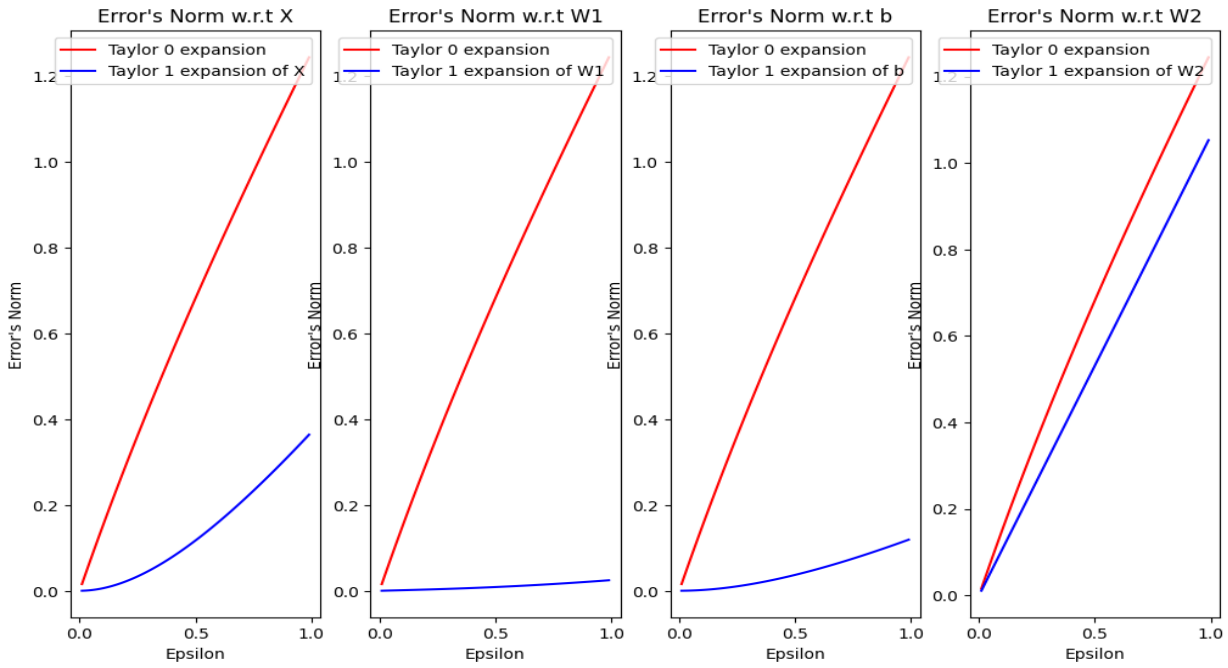
The Transpose test results are as follows:

```
The Jacobian Transpose test w.r.t X is 0.0
The Jacobian Transpose test w.r.t W is 0.003994913416483538
The Jacobian Transpose test w.r.t b is 1.1102230246251565e-16
```

Second Clause

In this clause we were asked to do the same as the last clause, but with a Residual neural network, the results are as follows:

The Jacobian Test:



Observation – while the approximation of the first order with respect to X and b noticeably behave as second-degree polynomials, W1 has a slight trend of that sort and W2 is almost linear. We believe that the explanation for this behavior is that for calculation for both W's, we needed to vectorize their shifts, which affected their scores.

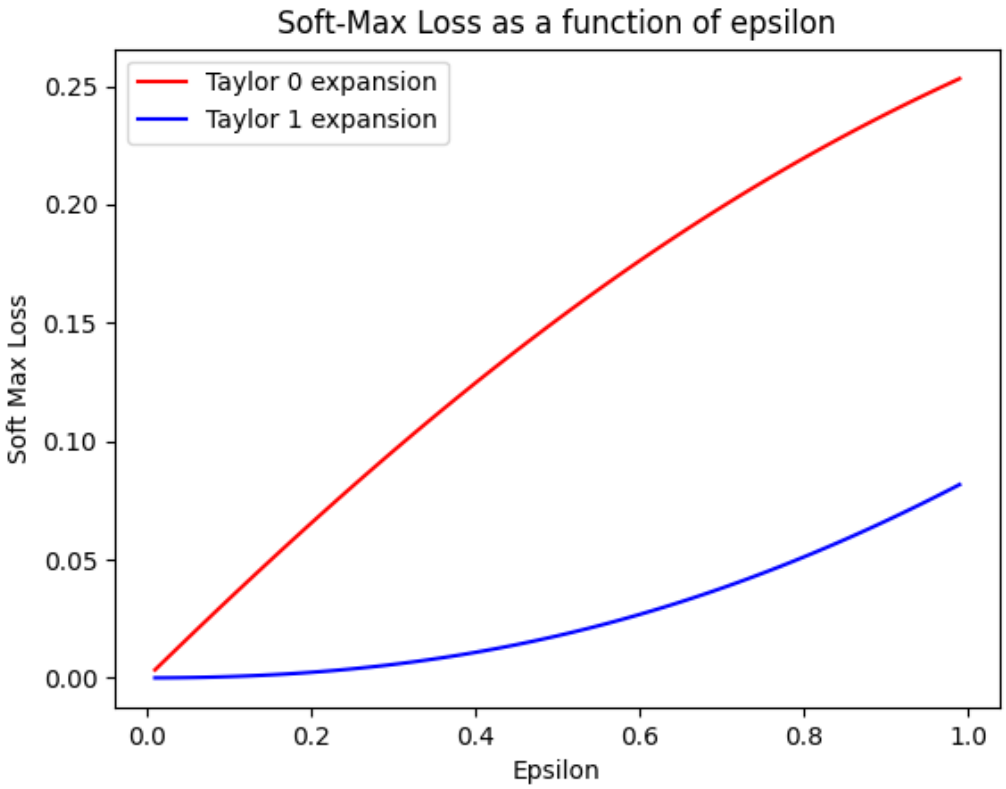
The Transpose test results are as follows:

```
The Jacobian Transpose test w.r.t X is 4.440892098500626e-16
The Jacobian Transpose test w.r.t W1 is 0.008442272817497987
The Jacobian Transpose test w.r.t b is 1.1102230246251565e-16
The Jacobian Transpose test w.r.t W2 is 0.023192108198600847
```

Third Clause

In this clause, we were asked to combine the various parts of the neural network we have built this far, and make sure our network passes the Gradient test.

The results follow below:



Fourth Clause

In this clause we were asked to activate our fully connected network, and train it to optimize a classification problem using soft-max loss as the target function. The hyperparameters of our system were the learning rate of the SGD operation, the depth of the network itself and the width of each layer.

We found that different learning rates are better for the different data sets, and that some depths are also better fitted for different data sets. One surprising conclusion was that increasing the width of the network did not always mean improving our network's score, which, in our opinion, was counter intuitive.

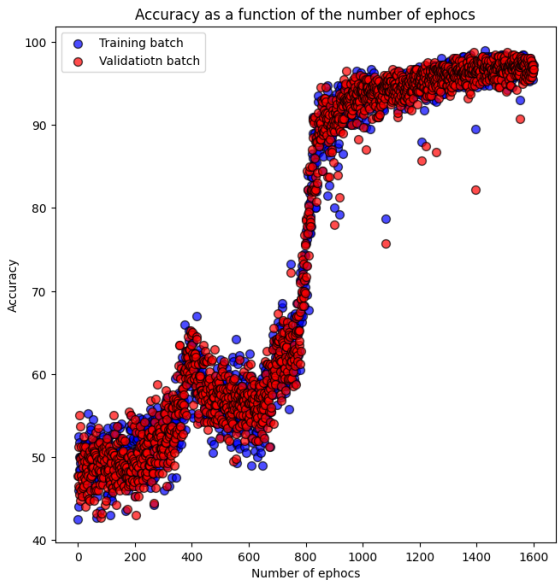
It is also worth noting that we decided to set some of the hyper parameters as constants, for example the number of epochs and the size of the random minibatch to test the score of the network. We did that to focus on the hyperparameters that we thought were more influential to the system itself.

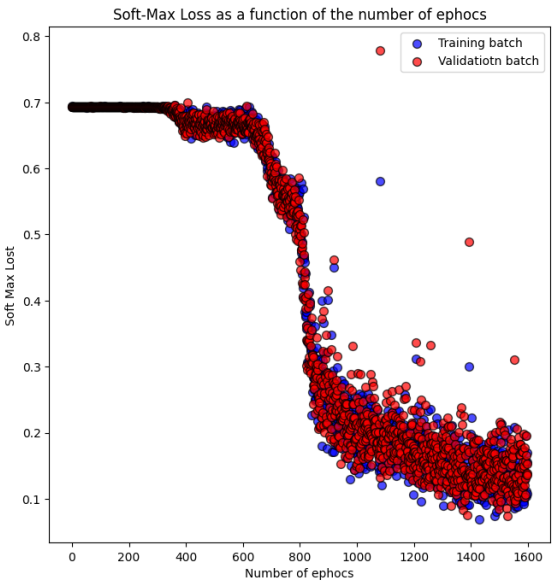
In the following results, we will demonstrate our system performance on the different data sets, noting the hyper parameters that were used for the specific run (for each data set we chose the best suited hyperparameters)

Data set "Swiss Roll":

For this data set, we used a learning rate of 0.01 for the SGD operation, with a network with 3 layers and the soft-max layer, the two first layers had width of 32 and the last one had width of 5.

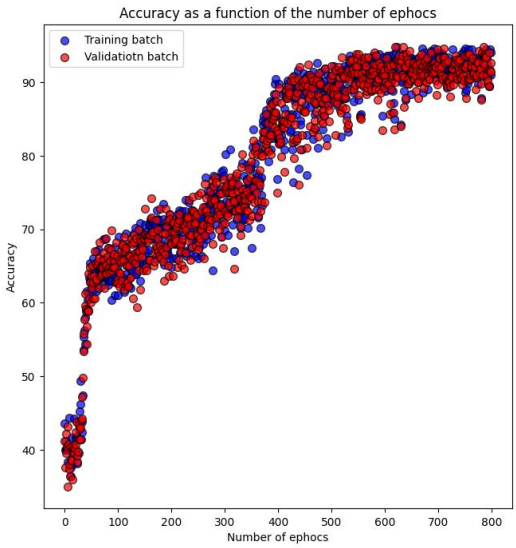
Note that for this specific data set, we saw the network took longer to optimize its wights thus we run it for 1600 epochs, twice as much as we did in the other data sets.

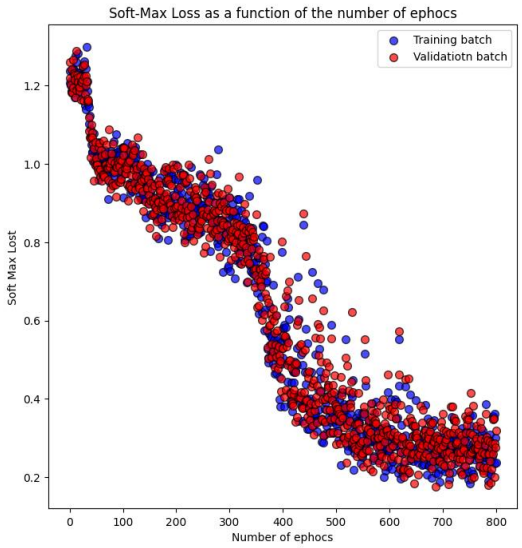




Data set “Peaks”:

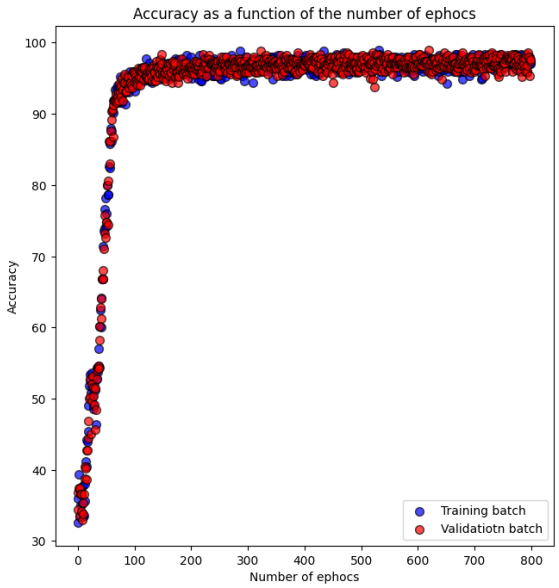
For this data set, we used a learning rate of 0.1 for the SGD operation, with a network with 3 layers and the soft-max layer, the two first layers had width of 32 and the last one had width of 5.

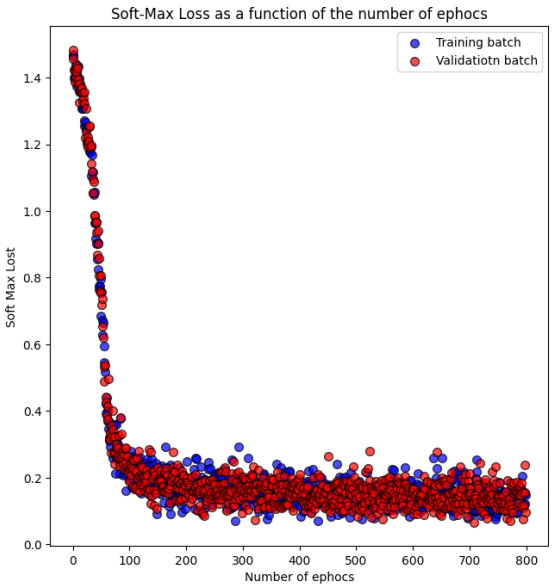




Data set “GMM”:

For this data set, we used a learning rate of 0.1 for the SGD operation, with a network with 2 layers and the soft-max layer, the first layers had width of 32 and the last one had width of 5.

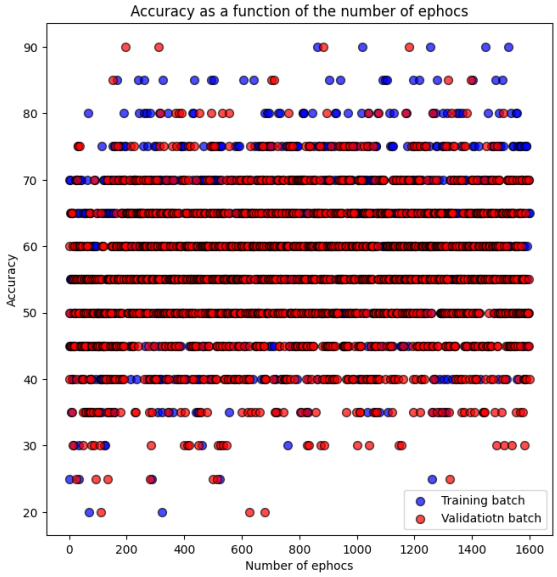


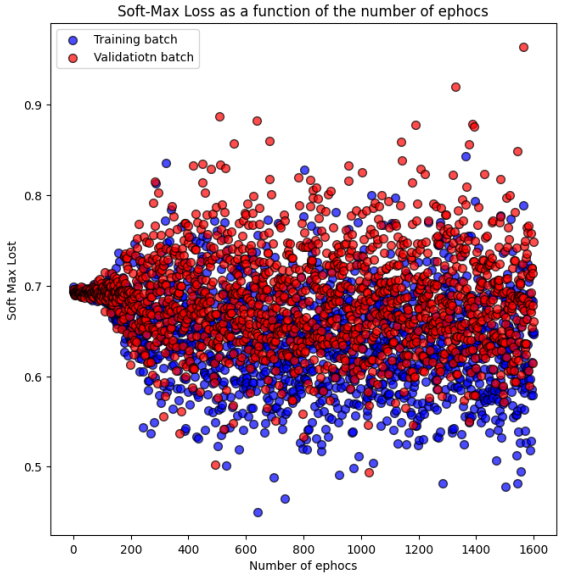


Fifth Clause

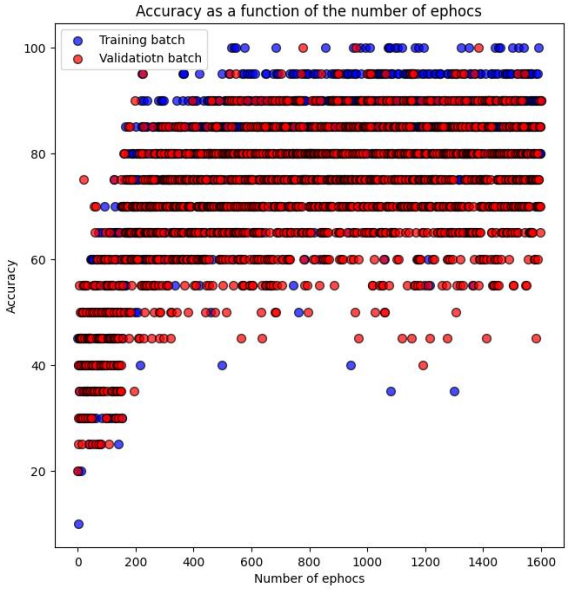
In this clause, we were asked to repeat the last clause, but only using 200 data points for training, we kept the hyperparameters as in the previous clause to isolate the difference:

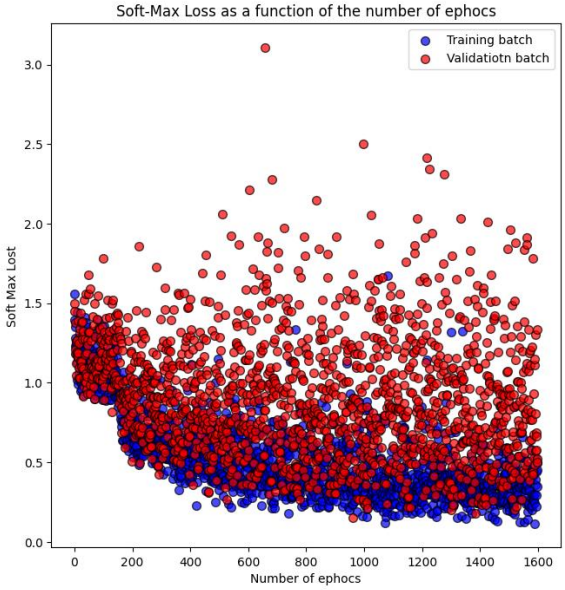
Data set “Swiss Roll”:



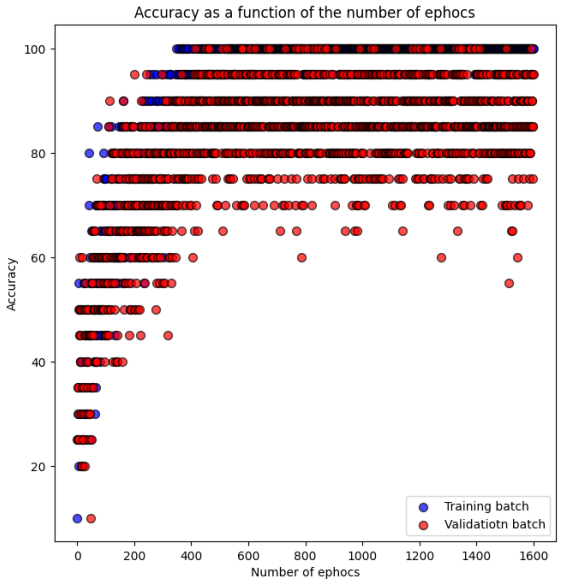


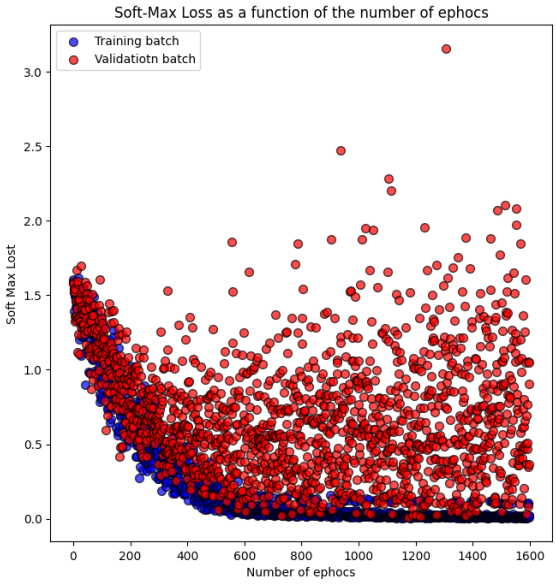
Data set “Peaks”:





Data set “GMM”:





Conclusions – it is easy to see that the results with the much smaller training data set are worse than the larger training data sets, especially for the validation batches. This fact makes sense because while we train on a smaller data set, the system optimize itself better to the mini batches selected from the training data itself, thus the scores for the Training batches “suffer” from overfitting but does not optimize the network good enough for the validation batches.