# Project #1 : MLP implementation

This report outlines the entire project, network architecture, training, evaluation, and the challenges encountered. The following diagram summarizes the network’s architecture and defines the notations that I am going to use later.

A black background with white text

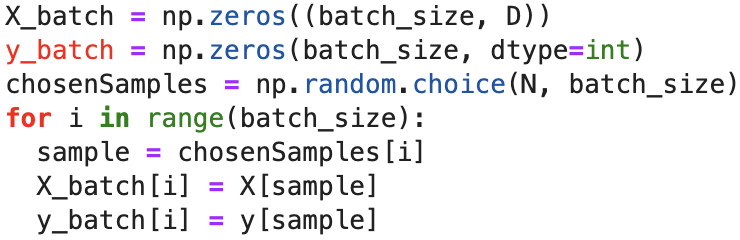
Description automatically generated

First, for the forward pass, I utilized Numpy’s functions to perform matrix multiplications as expected. After the forward pass, I had to implement the loss computation which includes log likelihood for the data loss and L2 regularization. Here is the expression of the “total” loss, which is going to be used later of the gradient descent (λ represents the regularization strength):

For the back propagation, I had to find the analytical expressions of the gradients of the loss with respect to the weights of the networks.

Here are the gradients of the loss with the respect of the intermediate vectors:

Using these we can easily find the gradient of the loss relative to the weights and the biases:

Next, I had to implement the batch initialization used later for the stochastic gradient descent. For this purpose, we used the Numpy function *random.choice(a, b)* which returns a size b array of integers between 0 and (a – 1). That array is going to pick randomly the indexes of the samples that are going to compose the batch.

Finally, I implemented the stochastic gradient descent using the formula of the gradient descent and the predict function which returns a Y vector indicating the class that has the biggest score.

A graph of loss and loss history

Description automatically generatedWith the default hyperparameters we notice that the validation accuracy is around 24%, which is not ideal. Looking at the loss curve, we notice that it decreases linearly, to solve that problem I changed the learning rate. The best I found was 1e-3. Then we can also see that the gap between the validation and training accuracy is not that big, to overcome that issue I tried to increase the hidden layers’ size. The best I found was 1000. Moreover, I increased the number of epochs up to 700 to decrease underfitting. At the same time to prevent overfitting I increased the regularization strength to 0.5.

With those hyperparameters we have a **validation accuracy of 44% and a test accuracy of 46%.** In addition, we see that the gap between validation and training set got bigger.