TÉCNICO LISBOA

Deep Learning

Instituto Superior Técnico

1st Semester - 2021/22

Master in Electrical and Computer Engineering

Homework 1

Group 44:

Diogo Matos	86981
Diogo Moura	86976
Felicia Ekenstam	101636

January 12th, 2022

Question 1

In this exercise, you will show that the binary and multinomial logistic losses are convex.

1.1

Show that the derivative of the sigmoid activation function can be expressed as $\sigma(z) = \sigma(z)(1 - \sigma(z))$:

$$\begin{aligned}
\sigma(z) &= \frac{1}{1+e^{-z}} \\
\sigma'(z) &= \frac{-1 * -e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1}{1+e^{-z}} * \frac{e^{-z}}{1+e^{-z}} \\
&= \sigma(z) * \frac{(1+e^{-z})-1}{1+e^{-z}} \\
&= \sigma(z) * \left(1 - \frac{1}{1+e^{-z}}\right) \\
&= \sigma(z) * (1 - \sigma(z))
\end{aligned}$$

1.2

Show that the binary logistic loss below is convex as a function of z:

$$L(z; y) = -\frac{1+y}{2} * log(\sigma(z)) - \frac{1-y}{2} * log(1 - \sigma(z))$$

For y = +1 we have the simplified version:

$$L(z; y = +1) = -log(\sigma(z))$$

Which gives the first derivative of L with respect to z, by using the answer in 1.1 as the inner derivative:

$$\begin{aligned}
L'(z; y = +1) &= \frac{d}{dz}(-log(\sigma(z))) \\
&= -\frac{1}{\sigma(z)} * \sigma(z) * (1 - \sigma(z)) \\
&= \sigma(z) - 1 \\
&= \frac{-1}{1+e^{z}}
\end{aligned}$$

Then we get the second derivative of L with respect to z as:

$$\begin{aligned} L''(z; y = +1) &= \frac{d}{dz}(\sigma(z) - 1) \\ &= \sigma(z) * (1 - \sigma(z)) \\ &= \frac{1}{1 + e^{-z}} * \left(\frac{1 + e^{-z} - 1}{1 + e^{-z}}\right) \\ &= \frac{e^{-z}}{(1 + e^{-z})^2} \end{aligned}$$

Finally, to show that the binary logistic loss is convex we look at the second derivative (the Hessian) to see if it is positive-semi-definite, meaning that it should be bigger or equal to zero for all z. Here, since $e^{-z} > 0$ for all z, it means that the loss is convex.

1.3

Compute the Jacobian matrix of the softmax transformation below, on point z:

$$[softmax(z)]_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

for j = 1,...,K. The Jacobian is the K-K matrix whose (j,k)-th element is the derivative: $\frac{\partial [softmax(z)]_j}{\partial z_k}$. To simplify the calculation of the jacobian elements we can let M(z) = softmax(z) and S(z) = $\sum_{k=1}^{K} e^{z_k}$.

$$\begin{aligned} \frac{\partial [softmax(z)]_j}{\partial z_i} &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\right] \\ &= \frac{S(z) * (\frac{\partial e^{z_j}}{\partial z_i}) - e^{z_j}(\sum \frac{\partial e^{z_k}}{\partial z_i})}{S(z)^2} \end{aligned}$$

If i $\neq$ j we have that:

$$\begin{aligned} \frac{\partial [softmax(z)]_j}{\partial z_i} &= \frac{-e^{z_j}(\sum \frac{\partial e^{z_k}}{\partial z_i})}{S(z)^2} \\ &= \frac{-e^{z_j} * e^{z_i}}{S(z)^2} \\ &= -M(z)_j * M(z)_i \end{aligned}$$

And if i = j we have that:

$$\begin{aligned} \frac{\partial [softmax(z)]_j}{\partial z_i} &= \frac{S(z) * e^{z_j} - e^{z_j} * e^{z_i}}{S(z)^2} \\ &= \frac{e^{z_j}}{S(z)} * \frac{S(z) - e^{z_i}}{S(z)} \\ &= M(z)_j * (1 - M(z)_i) \end{aligned}$$

A generalized expression for this Jacobian could be the following in matrix notation:

$$\frac{\partial softmax(z)}{\partial z} = Diagonal(M(z)) - M(z) * M(z)^T$$

1.4

Compute the gradient and Hessian of the multinomial logistic loss below, with respect to z and show that the loss is convex:

$$\begin{aligned} L(z; y = j) &= -log[softmax(z)]_j \\ &= -log(M(z)_j) \end{aligned}$$

The gradient of L(z;y=j) with respect to z is as follows, with the use of the previous answer in 1.3 and the lecture slides for Neural Networks (Lecture 5):

$$\begin{aligned} \nabla L &= -\nabla_z log(softmax(z))_j \\ &= -\frac{1}{M(z)_j} * \frac{\partial M(z)}{\partial z} \\ &= softmax(z) - 1_{y=j} \end{aligned}$$

where $1_{y=j}$ means that it is only equal to 1 for y equal to the j-th label. From this, the Hessian is given as:

$$\begin{aligned} H(z) &= \frac{\partial \nabla L}{\partial z} \\ &= \frac{\partial M(z)}{\partial z} - \frac{\partial 1_{y=j}}{\partial z} \\ &= Diagonal(M(z)) - M(z) * M(z)^T \end{aligned}$$

To show that the loss is convex we must show that the Hessian H(z) is positive semi-definite for any z, meaning that $v^T * H(z) * v \geq 0$ for any vector v $\in R^K$. Here we use that $v^T * H(z) * v$ can be written as $\sum_i \sum_j v_i * v_j * M(z)_{i,j}$, and that $\sum_j M(z)_j = 1$.

$$\begin{aligned} v^T * H(z) * v &= v^T * (Diagonal(M(z)) - M(z) * M(z)^T) * v \\ &= v^T * Diagonal(M(z)) * v - (v^T * M(z))^2 \\ &= \sum_j v_j^2 * M(z)_j - \left(\sum_j v_j * M(z)_j \right)^2 \\ &= \left(\sum_j M(z)_j \right) \left(\sum_j v_j^2 * M(z)_j \right) - \left(\sum_j v_j * M(z)_j \right)^2 \end{aligned}$$

Then, based on the Cauchy Schwarz inequality where you have $a = \sqrt{M(z)_j}$ and $b = \sqrt{M(z)_j} v_j$, we would have that:

$$|||a|||_2^2 * |||b|||_2^2 - |||ab|||_2^2 = \left(\sum_j M(z)_j \right) \left(\sum_j M(z)_j * v_j^2 \right) - \left(\sum_j M(z)_j * v_j \right)^2 \geq 0$$

So this means that the hessian is ≥ 0 and therefore is the loss L(z;y) convex.

1.5

Show that in a linear model, where z = W*ϕ(x)+b, the multinomial logistic loss is also convex with respect to the model parameters (W,b), and therefore a local minimum is a also global minimum. Is this also true in general when z is not a linear function of the model parameters?

Since the composition of an affine map with a convex function is convex, the same idea can apply to our case. The multinomial logistic loss as a function of (W,b) is a composition of the map z=W*ϕ(x)+b with the function L(z,y). So, since we have proven that the multinomial logistic loss is convex in earlier questions this means that we have a composition of a convex function and an affine map. This also means that the composition is convex and a local minimum is also a global minimum.

But in the general case. when z is not a linear function of (W,b) this would mean that the composition is not convex, at least not in every case, which means that it cannot be implied that the composition is convex. Only if the mapping is affine, meaning linear with some constant it can be implied that the composition is convex.

Question 2

1

Show that the squared error loss L(z;y) is convex with respect to (W,b), where $z = W^T * \phi(x) + b$**.** Here we will add the constant feature $\phi_0(x) = 1$ to capture the bias term $b = w_0$ so that we can prove the complexity with respect to only W.

$$\begin{aligned} L(z;y) &= \frac{1}{2}\|\|z-y\|\|_2^2 = \frac{1}{2}(z-y)^T(z-y) \\ &= \frac{1}{2}(W^T\phi(x)-y)^T(W^T\phi(x)-y) \\ \nabla L(W;y) &= \phi(X)^T(\phi(X)W-y) \\ H(W;y) &= \frac{\partial \nabla L(W;Y)}{\partial W} \\ &= \phi(X)^T\phi(X) \end{aligned}$$

Then, to be able to show that the squared error loss L(z;y) is convex we must show that the hessian is positive semi-definite. Meaning that $H(W;y) \geq 0$, or in other words that $y^T * H(W;y) * y \geq 0$.

$$\begin{aligned} y^T * H(W;y) * y &= y^T * \phi(X)^T * \phi(X) * y \\ &= (\phi(X) * y)^T * (\phi(X) * y) \\ &= \|\|\phi(X) * y\|\|^2 \geq 0 \end{aligned}$$

So, since the hessian is greater or equal to zero this means that the squared error loss is convex.

2

a)

The final loss on the train set is 0.111 and on the test set is 0.238. It is expected that the loss on the test set is greater than on the train set, since the test set is composed by data "unseen" by the model, which is not used to update the model weights. As can be seen on figure 1, the loss of the train set is always decreasing, but the loss on the test set starts increasing at the last epochs. This

phenomena is known as "overfitting". The model is fitting the data on the train set too much and loses its ability to generalize.

It can also be seen on figure 1, that the distance between the weight vector of the linear regression and the weight vector computed analytically decreases as the number of the epoch increases, which is expected.

To compute the analytical solution of the weight vector, the pseudo-inverse was used, because it provides the optimal solution to the least squares problem.

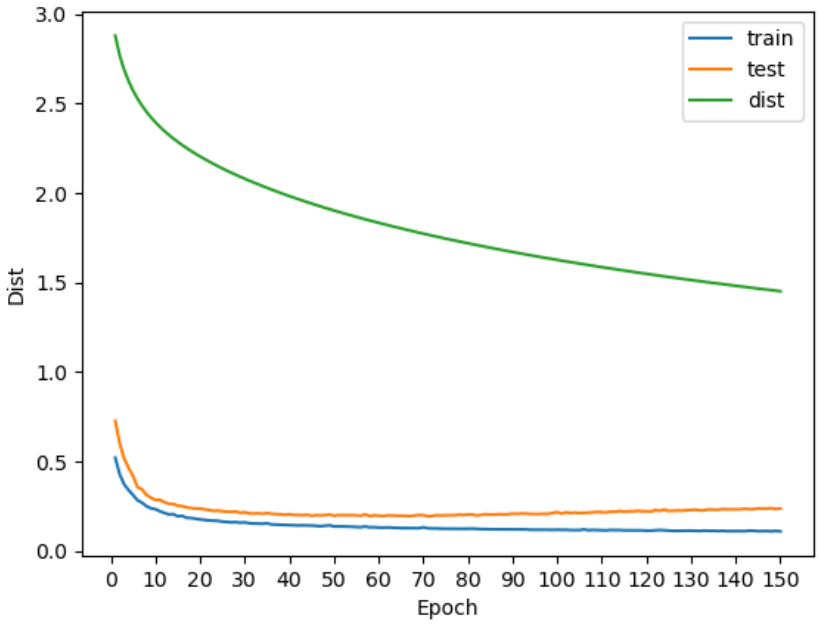

Figure 1: Plot of the losses of the linear regression on the train and test sets and distance to weight vector computed analytically

b)

The final loss on the train set is 0.099 and on the test set is 0.165. The loss is smaller than in the linear regression, which is expected since a neural network has a greater ability to learn and fit functions than a simple linear regression, which is composed by a single unit.

Also the model does not seem to overfit like in the case of the simple linear regression, since it is more complex and composed by more units than the simple linear regression.

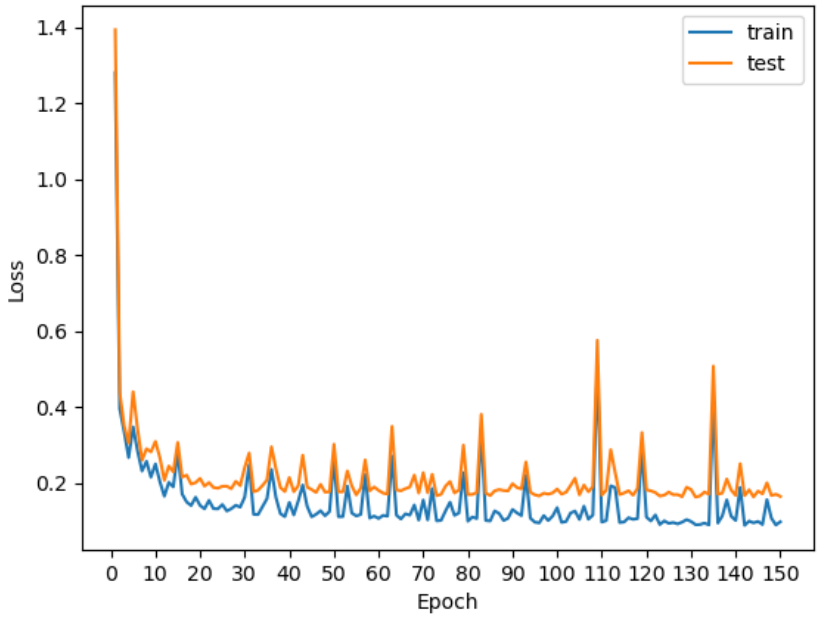

Figure 2: Plot of the losses of the neural network on the train and test sets

Question 3

1

a)

The accuracy on the test set was 0.7013 and on the validation set it was 0.7126.

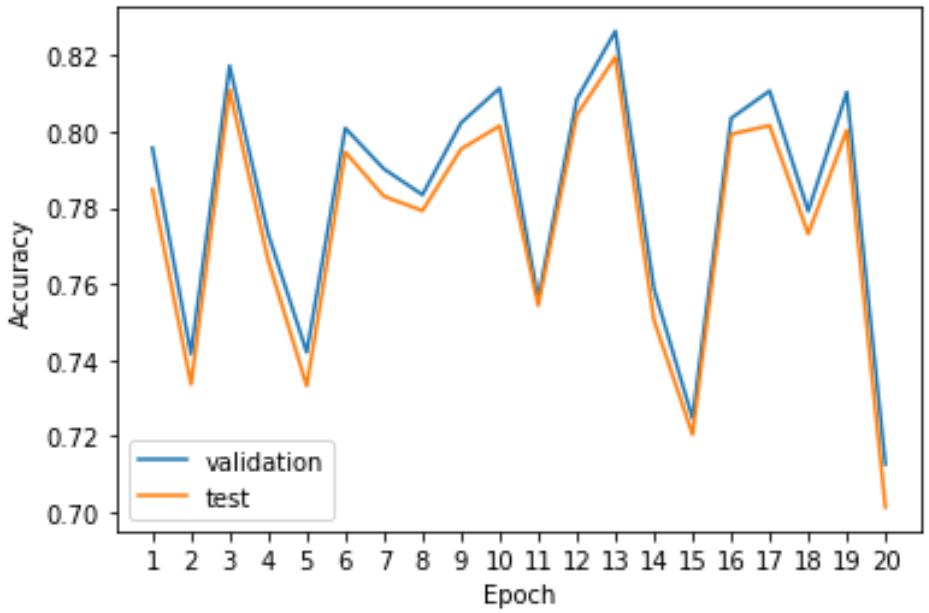

Figure 3: Plot of the accuracies for the Perceptron Implementation

b)

The accuracy on the test set was 0.8403 and on the validation set it was 0.8503.

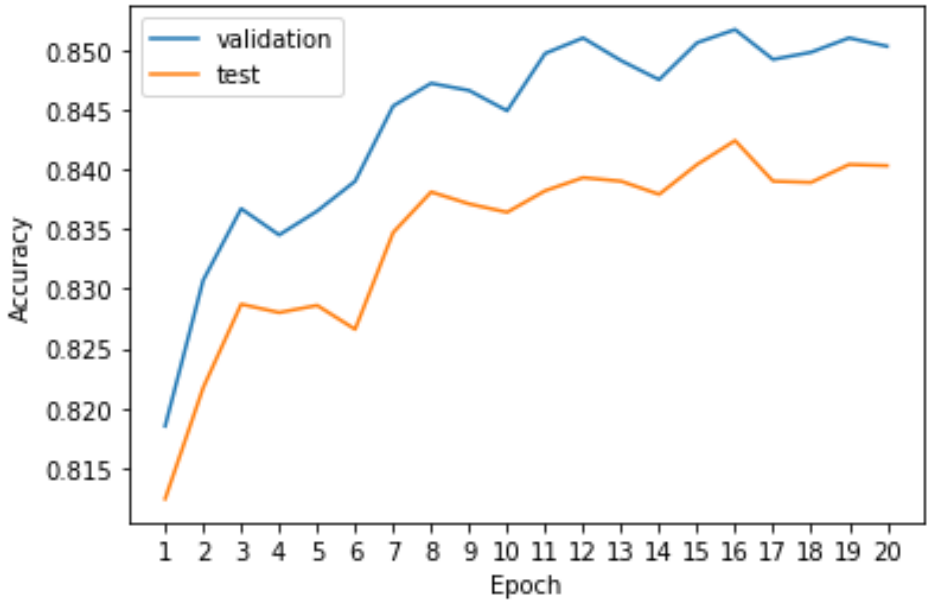

Figure 4: Plot of the accuracies for the Logistic Regression Implementation

2

a)

The multi-class perceptron implemented is a linear classifier, so it will separate the data classes using hyperplanes. Thus, it cannot solve non-linearly separable problems, as is often demonstrated by the classic example of the XOR problem.

MLP's on the other hand, are capable of solving non-linearly separable problems. This is because in an MLP by adding hidden units with non-linear activation functions, units which will compute a representation of their input and pass it forwards, we will be adding to the expressiveness of the network and making it capable of modelling more complex non-linear classifiers or functions. This is contrary to a linear classifier where feature engineering would have to be done in the data preprocessing stage in order to generate a dataset that would be linear separable, or at least very close to it.

However, the case changes for an MLP with a linear activation function, because no matter the number of layers or neurons, using linear activation functions means the output layer will be a linear function of the first layer, and so having an MLP with linear activation function becomes equivalent to a simple linear regression model.

b)

For a learning rate of 0.001 the accuracy on the test set was 0.8698 and on the validation set it was 0.8764.

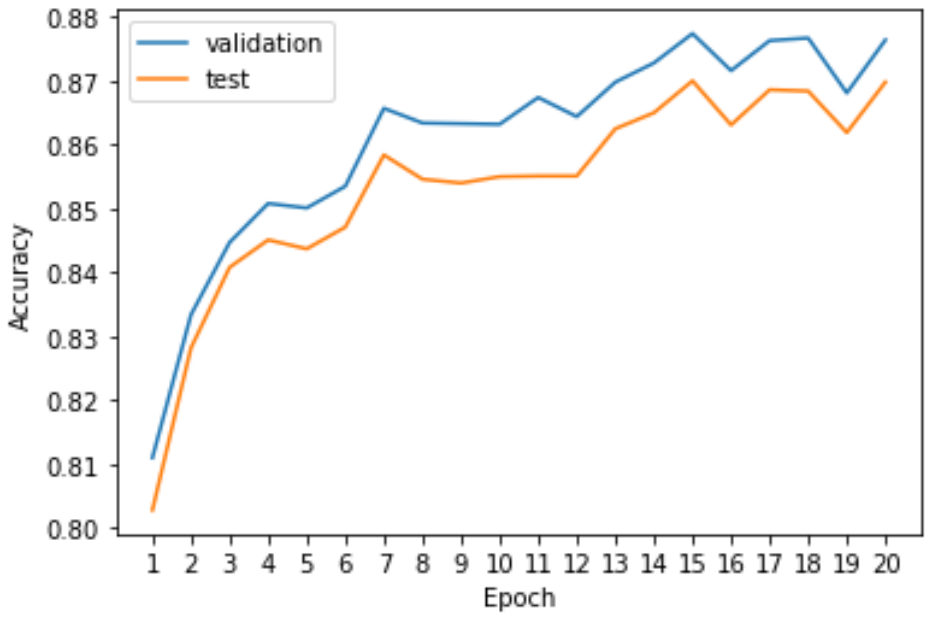

Figure 5: Plot of the accuracies for the MLP Implementation

Question 4

1

According to table 1, the best configuration is the one with learning rate equal to 0.001.

Table 1: Trainning Loss and Validation accuracy of Logistic Regression according to learning rate

Learning Rate	Training Loss	Final Test Acc.
Default (0.01)	0.4866	0.8333
0.001	0.4397	0.8388
0.1	3.1164	0.7959

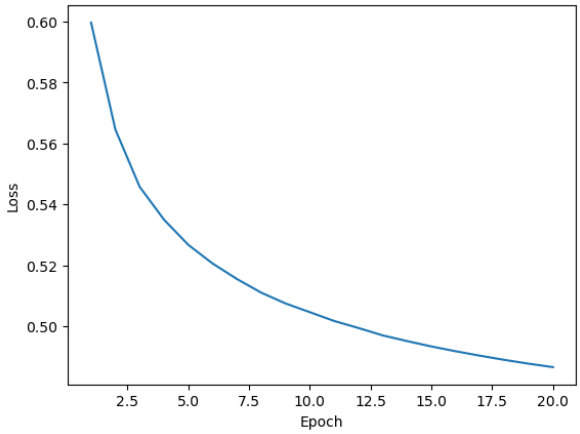

Figure 6: Default configuration training loss with learning rate equal to 0.01

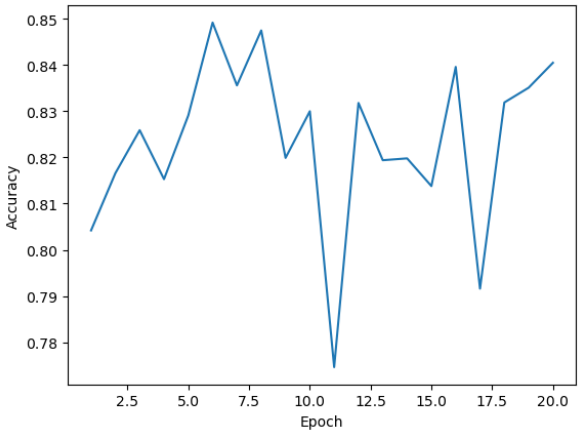

Figure 7: Default configuration validation accuracy with learning rate equal to 0.01

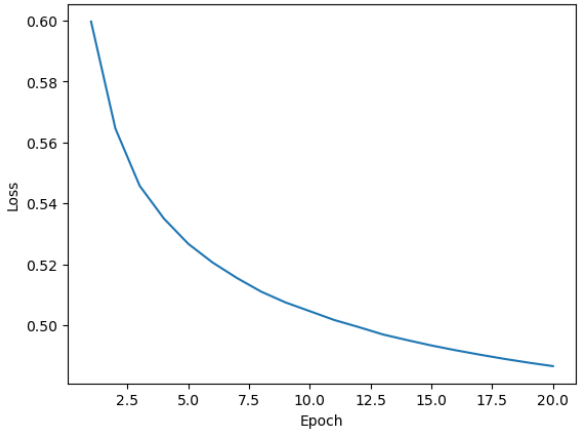

Figure 8: Best configuration training loss with learning rate equal to 0.001

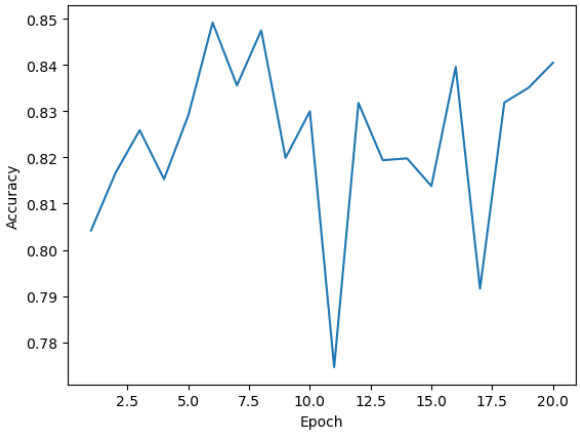

Figure 9: Best configuration validation accuracy with learning rate equal to 0.001

2

The default configuration is Learning rate = 0.01, Hidden Size = 200, Number of Layers = 1, dropout= 0.3, activation = $ReLU$, optimizer = SGD.

On table 2 is presented the results for each configuration used for hyperparameter tuning. For each hyperparemeter tuned, the remaining hyperparameters are left on their default, according to the default configuration.

Table 2: Trainning Loss and Validation accuracy according to hyperparmeters

Hyperparemeters	Training Loss	Final Test Acc.
Default	0.3764	0.8634
Learning Rate= 0.1	2.2389	0.1477
Learning Rate= 0.001	0.3693	0.8794
Hidden Size= 100	0.4066	0.8608
Dropout Probability= 0.5	0.4768	0.8507
Activation = $Tanh$	0.5155	0.8452
Optimizer = $Adam$	1.8053	0.5032

According to table 2, the best configuration is the one which has the learning rate equal to 0.001.

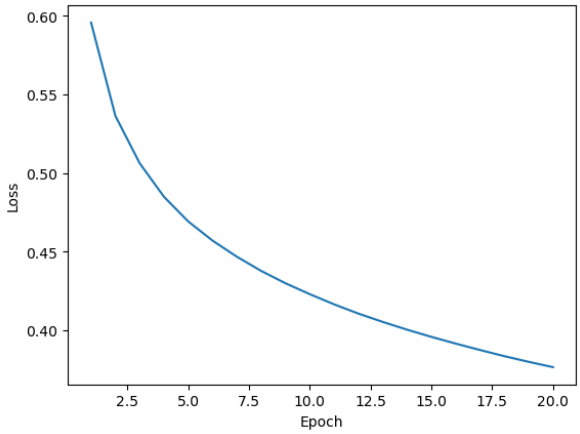

Figure 10: Default configuration training loss

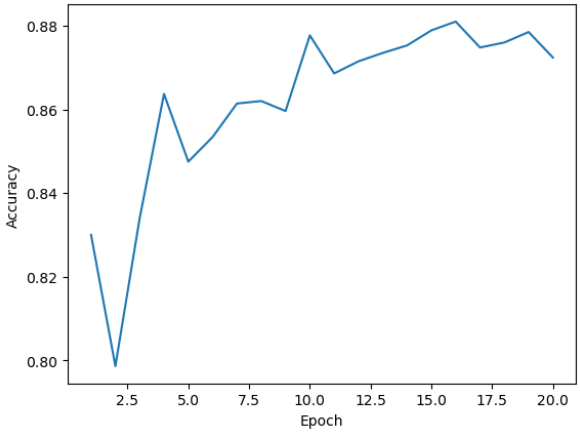

Figure 11: Default configuration validation accuracy

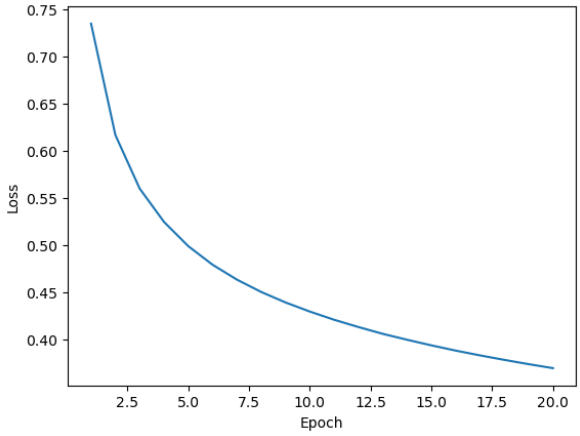

Figure 12: Best configuration training loss with Learning Rate = 0.001

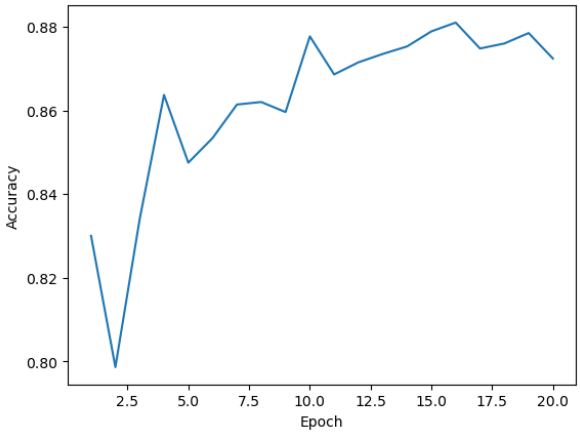

Figure 13: Best configuration validation accuracy with Learning Rate = 0.001

3

According to table 3, the best configuration is the one with 2 layers.

Table 3: Trainning Loss and Validation accuracy according to number of layers

Layers	Training Loss	Final Test Acc.
Default (1)	0.3764	0.8634
2	0.4025	0.8711
3	0.4328	0.8622

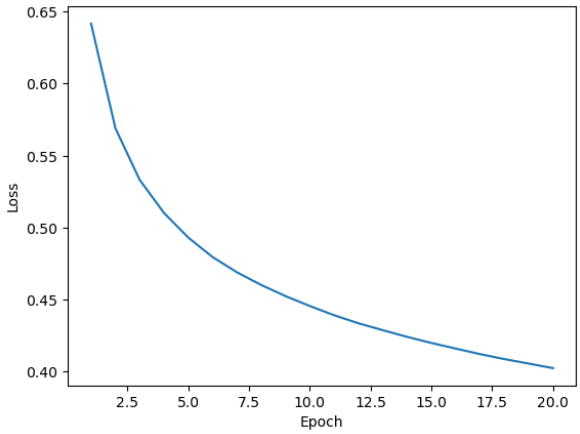

Figure 14: Best configuration training loss with 2 Layers

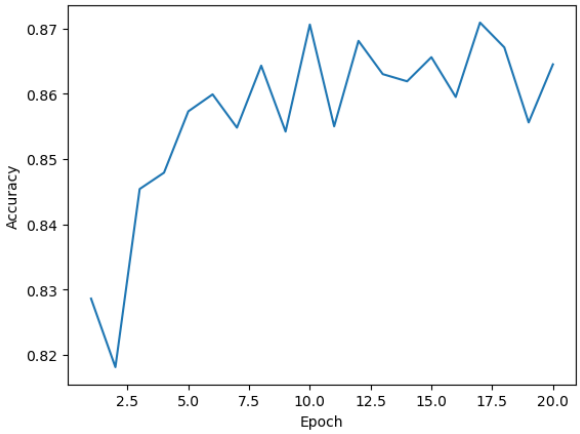

Figure 15: Best configuration validation accuracy with 2 Layers