Question 1

1. $$\sigma'(z) = \left(\frac{1}{1+e^{-z}}\right)' = \frac{(1+e^{-z})\times(1)' - 1\times(1+e^{-z})'}{(1+e^{-z})^2} = \frac{(1+e^{-z})\times 0 - (-e^{-z})}{(1+e^{-z})^2} =$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$\sigma(z)(1-\sigma(z)) = \left(\frac{1}{1+e^{-z}}\right)\left(1 - \frac{1}{1+e^{-z}}\right) = \left(\frac{1}{1+e^{-z}}\right) - \left(\frac{1}{1+e^{-z}}\right)^2 =$$

$$= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} = \frac{(1+e^{-z})}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2} =$$

$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma'(z)$$

2. $$L'(z; y=+1) = (-\log\sigma(z))' = \left(-\log\left(\frac{1}{1+e^{-z}}\right)\right)' = \left(-\log\left((1+e^{-z})^{-1}\right)\right) =$$

$$= \left(\log(1+e^{-z})\right)' = \frac{(1+e^{-z})'}{1+e^{-z}} = -\frac{e^{-z}}{1+e^{-z}} = \frac{\frac{1}{e^z}}{1+\frac{1}{e^z}} =$$

$$= -\left(\frac{1}{e^z} \times \frac{1}{1+\frac{1}{e^z}}\right) = -\frac{1}{e^z+1}$$

$$L''(z; y=+1) = \left(-\frac{1}{e^z+1}\right)' = -\frac{(e^z+1)(1)' - 1\times(e^z+1)}{(e^z+1)^2} =$$

$$= \frac{e^z}{(e^z+1)^2}$$

$e^z > 0$ and $(e^z+1)^2 > 0$, therefore $\frac{e^z}{(e^z+1)^2} > 0$.

Since the second derivative is greater than 0, the function is a convex function.

3. Softmax Jacobian $\rightarrow$ $\left[J_{i,j} = \frac{\partial s(x)_i}{\partial x_j} \right]$

Diagonal principal

$i = j$

$$J_{i,i} = \frac{\partial s(x)_i}{\partial x_i}$$

$$= \frac{\sum_j e^{x_j} e^{x_i} - e^{x_i} e^{x_i}}{\left(\sum_j e^{x_j}\right)^2}$$

$$= s(x)_i - s(x)^2_i$$

Resto da matriz

$i \neq j$

$$J_{i,j} = \frac{\partial s(x)_i}{\partial x_j}$$

$$= \frac{\sum_j e^{x_j} \cdot 0 - e^{x_i} e^{x_j}}{\left(\sum_j e^{x_j}\right)^2}$$

$$= - s(x)_i \; s(x)_j$$

$$J = \begin{bmatrix} s(x)_0 - s(x)_0^2 & -s(x)_0 \, s(x)_1 & \cdots & -s(x)_0 \, s(x)_n \\ -s(x)_1 \, s(x)_0 & s(x)_1 - s(x)_1^2 & \cdots & -s(x)_1 \, s(x)_n \\ \cdots & \cdots & \cdots & \cdots \\ -s(x)_n \, s(x)_0 & -s(x)_n \cdot s(x)_1 & \cdots & s(x)_n - s(x)_n^2 \end{bmatrix}$$

4. According to the previous exercise the Jacobian of softmas(z) is:

$$J = \begin{bmatrix} s_0 - s_0^2 & -s_0 s_1 & \dots & -s_0 s_n \\ -s_1 s_0 & s_1 - s_1^2 & \dots & -s_1 s_n \\ \dots & \dots & \dots & \dots \\ -s_n s_0 & -s_n s_1 & \dots & s_n - s_n^2 \end{bmatrix}$$

The gradient of $L(z; y=j) = -\log[\text{softmax}(z)]_j$ can be computed as $(-\log[J_{ij}])$ and is equal to:

$$\begin{bmatrix} \frac{s_0 - s_0^2}{s_0} & \frac{-s_0 s_1}{s_0} & \dots & \frac{-s_0 s_n}{s_0} \\ \frac{-s_1 s_0}{s_1} & \frac{s_1 - s_1^2}{s_1} & \dots & \frac{-s_1 s_n}{s_1} \\ \dots & \dots & \dots & \dots \\ \frac{-s_n s_0}{s_n} & \frac{-s_n s_1}{s_n} & \dots & \frac{s_n - s_n^2}{s_n} \end{bmatrix}$$

To compute the Hessian of this loss we compute:

$$\frac{d}{ds_0}\left(-\left(\frac{s_0 - s_0^2}{s_0}\right)\right) = -\frac{d}{ds_0}\left(\frac{s_0 - s_0^2}{s_0}\right)$$

$$= -\left(\frac{((s_0 - s_0^2)' \times s_0) - ((s_0 - s_0^2) \times s_0')}{s_0^2}\right)$$

$$= -\frac{((s_0' - 2s_0) \times s_0) - ((s_0 - s_0^2) \times s_0')}{s_0^2}$$

$$= -\frac{(\cancel{s_0' s_0} - 2s_0^2 - \cancel{s_0' s_0} + s_0^2 s_0')}{s_0^2}$$

$$= \frac{2s_0^2 - s_0^2 s_0'}{s_0^2}$$

$$= \frac{\cancel{s_0^2}(2 - s_0')}{\cancel{s_0^2}}$$

$$= 2 - s_0'$$

$$\frac{\partial}{ds_0}\left(\frac{s_0 s_1}{s_0}\right) = \frac{(s_0 s_1)' s_0 - (s_0 s_1) s_0'}{s_0^2}$$

$$= \frac{(s_0' s_1 + s_0 s_1') \times s_0 - (s_0 s_1) s_0'}{s_0^2}$$

$$= \frac{\cancel{s_0' s_1 s_0} + s_0^2 s_1' - \cancel{s_0' s_1 s_0}}{s_0^2}$$

$$= \frac{\cancel{s_0^2} s_1'}{\cancel{s_0^2}} = s_1'$$

With these results we can conclude that the Hessian is:

$$H_{ij} = \begin{bmatrix} 2 - s_0' & s_1' & \cdots & s_n' \\ s_0' & 2 - s_1' & \cdots & s_n' \\ \cdots & \cdots & \cdots & \cdots \\ s_0' & s_1' & \cdots & 2 - s_n' \end{bmatrix}$$

In order to prove that this loss is convex we need to prove that the diagonal of the Hession is always positive.
The values of the diagonal are $2 - s_i'$ with i as the line in the Hessian.

$2 - s_i' = 2 - (s_i - s_i^2)$, with s_i' equal to J_{ii}

Given that the values of softmax are in the interval $[0,1]$, and so is the square of softmax, we can know than all the values in the diagonal are positive.

Question 2

1. In order to show that L is convex, we need to prove that $\frac{d^2L}{dz}$ is always positive.

Given that $\frac{d}{dz}(x^Tx) = 2x$ we can compute:

$$\frac{d}{dz}\left(\frac{1}{2}(z-y)^T(z-y)\right) =$$

$$= \cancel{\frac{d}{dz}\left(\frac{1}{2}\right)}_{0} \times \left((z-y)^T(z-y)\right) + \frac{1}{2}\frac{d}{dz}\left((z-y)^T(z-y)\right)$$

$$= \frac{1}{\cancel{2}} \times \cancel{2}(z-y)$$

$$= z-y$$

$$\frac{d}{dz}(z-y) = 1 > 0$$

Given that $\frac{d^2L}{dz} = 1$ which is always positive, we can conclude that L is convex with respect to (w,b), with $z = w^T\phi(x)+b$

Deep Learning - Homework 1 - Group 52

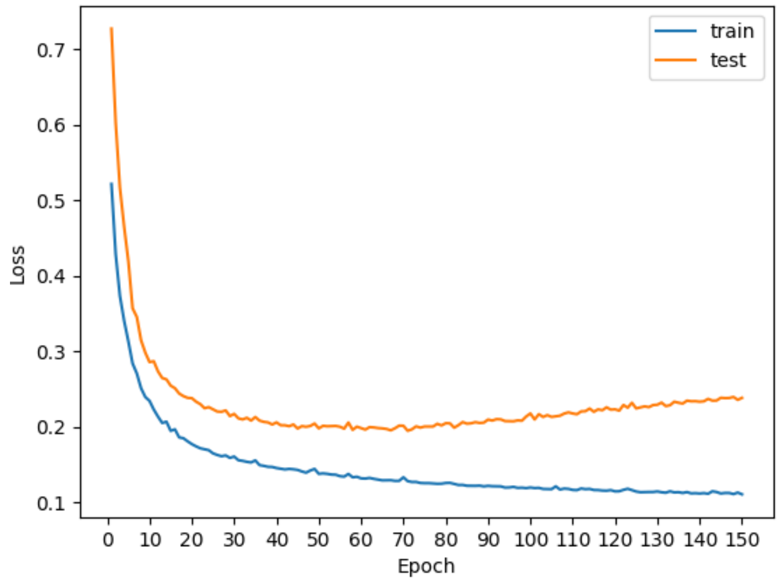
André Oliveira
ist193686

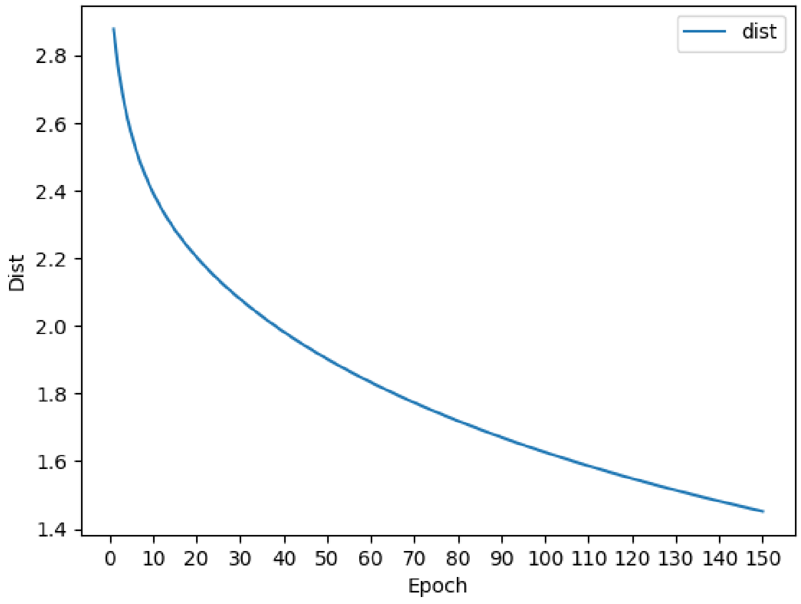
Beatriz Venceslau
ist193734

Rúben Gualdino
ist190632

Question 2.2 a)
The performance on the validation set is 0.111; the performance on the test set is 0.238.

In the graph below, we can observe the difference between the performance on the training and test sets as the number of epochs increases, this is due to the fact that the algorithm learns on the training set and not on the test set. The loss in the training set decreases faster because we can adapt the algorithm to the training set and not to the test set. In this graph we can see that we go from underfitting to overfitting as the epochs increase.

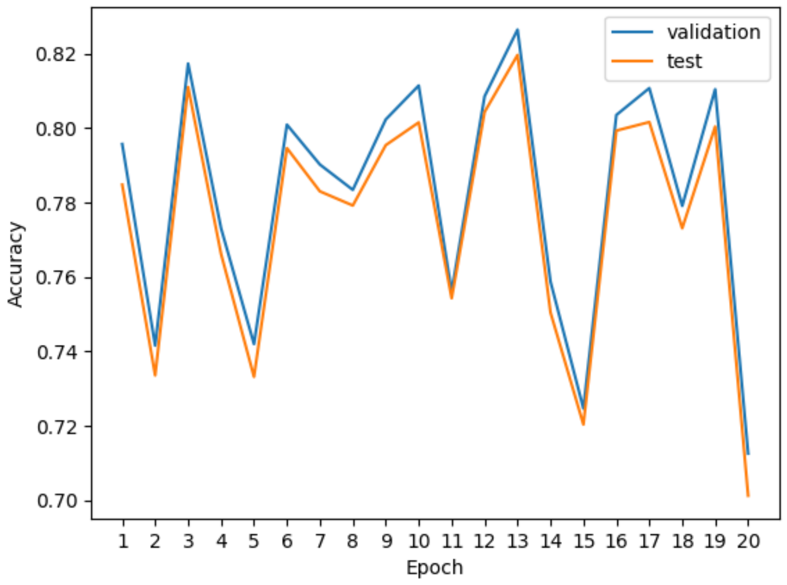




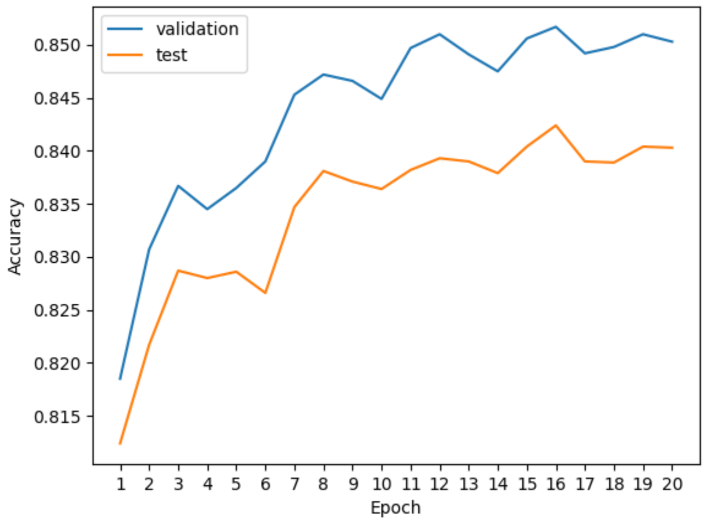
The distance between our weight vector and the weight vector computed analytically is 1.45.

Question 2.2 b)
Wrong results. We were unable to find and correct the bug in our implementation.

Question 3.1 a)
The performance on the validation set is 0.7126; the performance on the test set is 0.7013.



Question 3.1 b)
The performance on the validation set is 0.8503; the performance on the test set is 0.8403.



Question 3.2 a)
In case of a multi-layer perceptron, each of the hidden units in each of the layers computes some representation of the input and propagates forward that representation, as opposed to a simple perceptron, which has only one layer. With the multi-layer perceptron, this representation can be non-linear, allowing us to represent non-linear functions, which is not possible with the simple perceptron implemented above. This way, we can have better accuracy in a model without the limitation of a linear representation of the data.

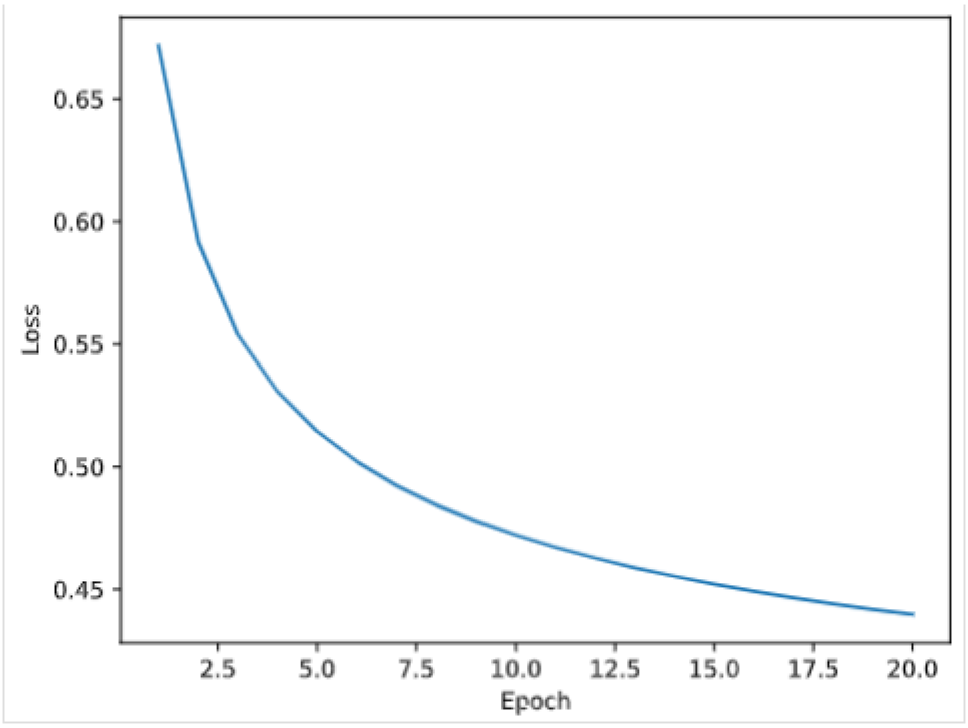
With a linear activation, the multi-layer perceptron will behave as linear units. Composing layers of linear units are equivalent to a single linear layer, so it is not more expressive than a simple perceptron.

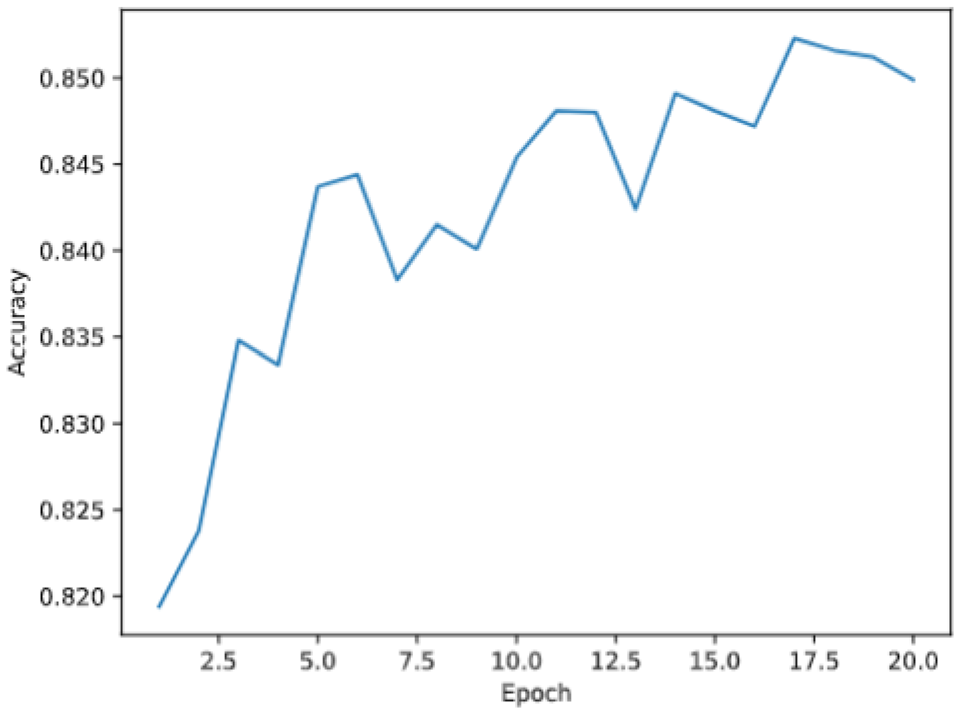
Question 3.2 b)
Wrong results. We were unable to find and correct the bug in our implementation.

Question 4.1

Learning Rate	Loss	Accuracy	Better
0.001	0.4397	0.8499 / 0.8388	x
0.01	0.4866	0.8405 / 0.8333	
0.1	3.1164	0.8053 / 0.7959	

The best configuration is the one with 0.001 learning rate. It finished with a loss of 0.4397 and the validation accuracy was 0.8499. The final accuracy was 0.8388.

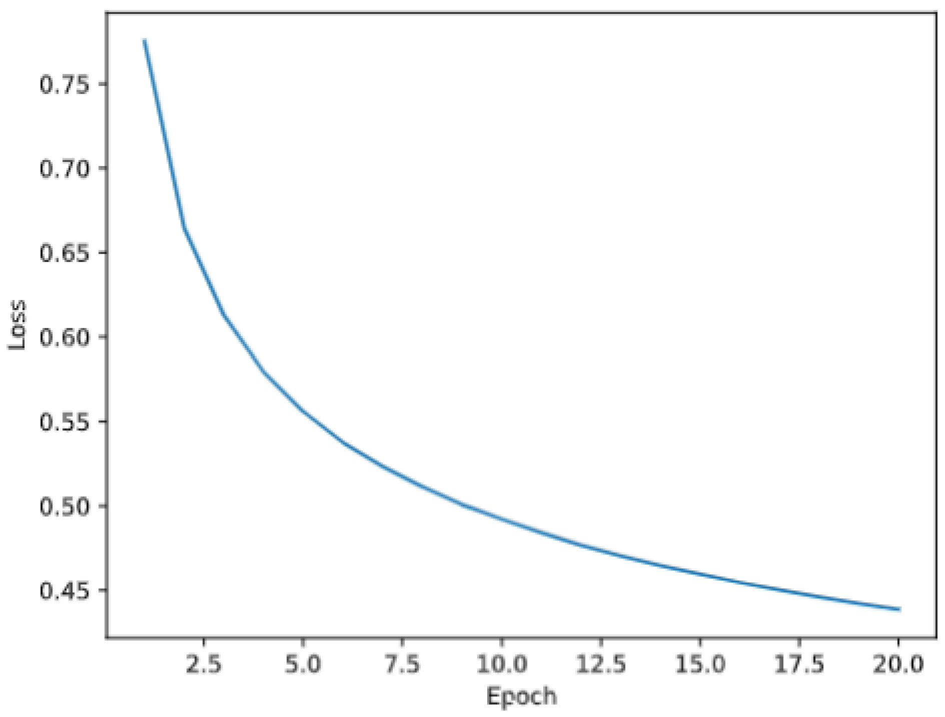


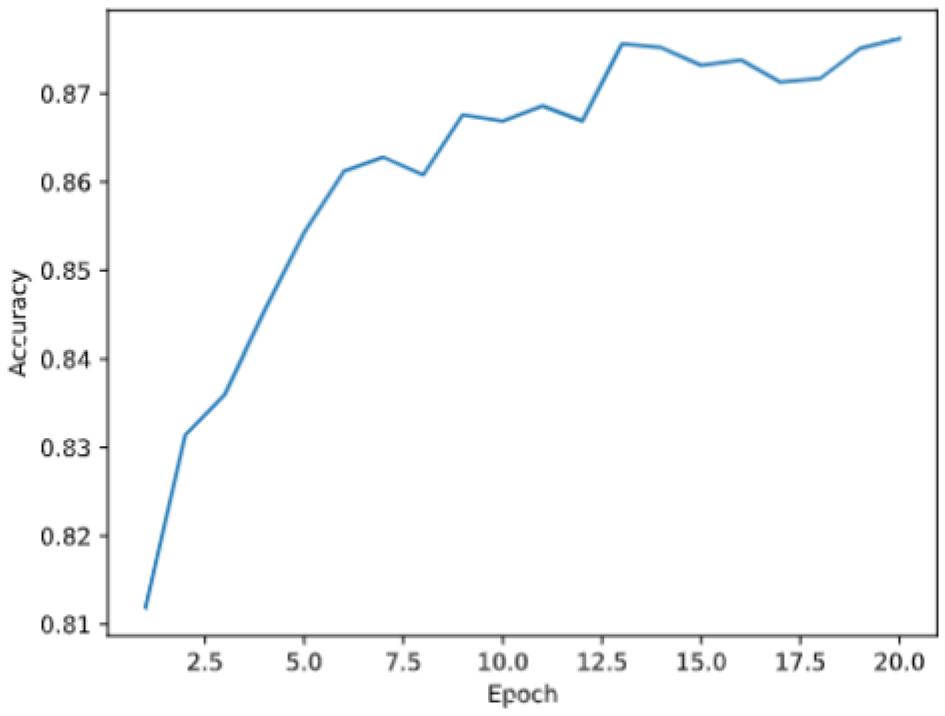


Question 4.2

Layers	LR	Hidden Size	Dropout	Activation	Optimizer	Loss	Accuracy	Better
1	0.1	200	0.3	ReLU	SGD	2.2521	0.1547 / 0.1570	
1	0.01	200	0.3	ReLU	SGD	0.4776	0.8619 / 0.8540	
1	0.001	200	0.3	ReLU	SGD	0.4386	0.8762 / 0.8731	x
1	0.01	100	0.3	ReLU	SGD	0.5080	0.8527 / 0.8420	
1	0.01	200	0.5	ReLU	SGD	0.6914	0.8200 / 0.8201	
1	0.01	200	0.3	Tanh	SGD	0.5944	0.8046 / 0.7988	
1	0.01	200	0.3	ReLU	Adam	2.0696	0.3101 / 0.3098	

The best configuration is the one with 0.001 learning rate. It finished with a loss of 0.4386 and the validation accuracy was 0.8762. The final accuracy was 0.8731.

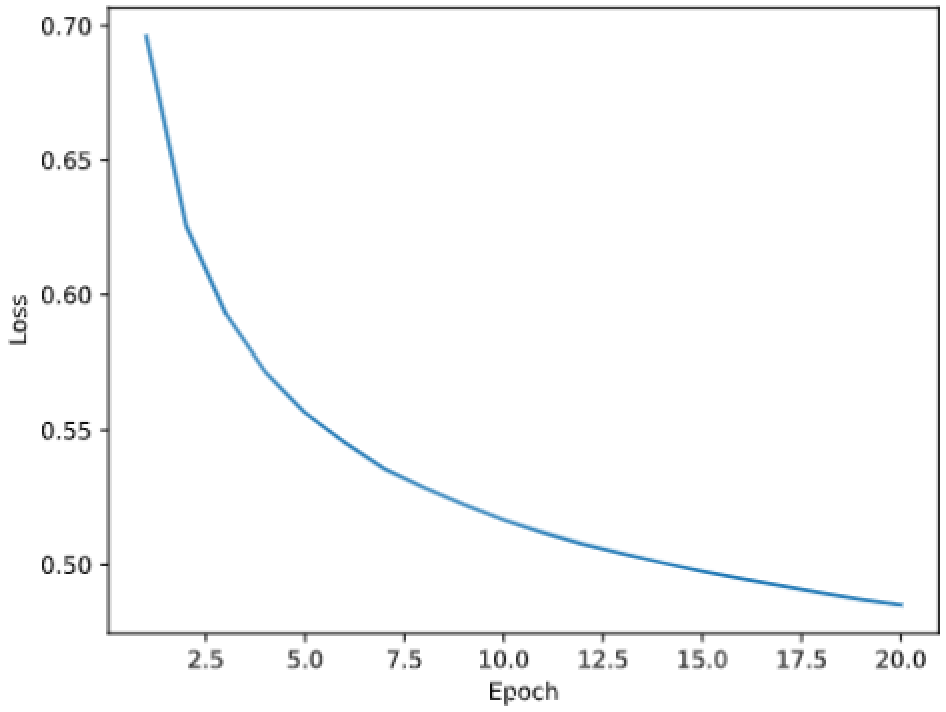

Loss
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

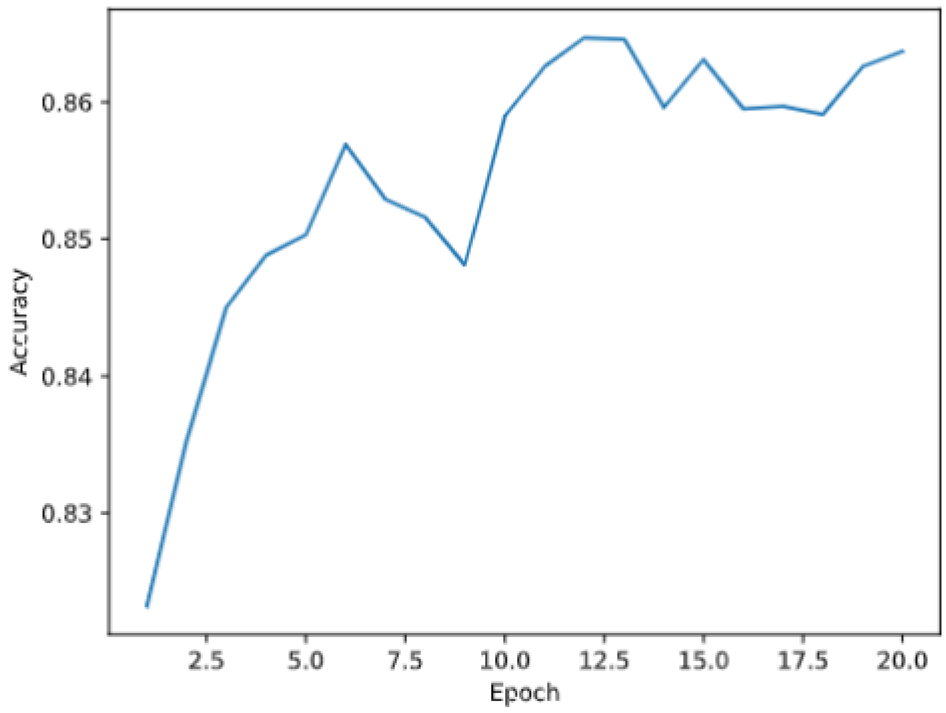

Accuracy
0.87
0.86
0.85
0.84
0.83
0.82
0.81
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

Question 4.3

Layers	LR	Hidden Size	Dropout	Activation	Optimizer	Loss	Accuracy	Better
2	0.01	200	0.3	ReLU	SGD	0.4850	0.8637 / 0.8550	X
3	0.01	200	0.3	ReLU	SGD	0.5409	0.8537 / 0.8510	

The best configuration is the one that uses 2 hidden layers. It finished with a loss of 0.4850 and the validation accuracy was 0.8637. The final accuracy was 0.8550.





References:
2.2b)https://towardsdatascience.com/building-a-neural-network-with-a-single-hidden-layer-using-numpy-923be1180dbf

3.2b) Solutions provided for the exercices of the practical lecture “Neural Networks and Backpropagation” - (neural_nets.ipynb)

We also used the solutions provided for all practical lectures and the slides of the theoretical lectures as references when solving the questions given.