TÉCNICO LISBOA

Deep Learning

2021/2022

Homework No. 1

1. N.º 90007 Nome: Alice Rosa
2. N.º 90029 Nome: Beatriz Pereira
3. N.º 89916 Nome: José Marques

Group No. 55

Professors: André Martins, Francisco Melo, Ben Peters

1 Question 1

In this question is shown that the binary and multinomial logistic losses are convex.

1.1 Demonstration of the derivative of $\sigma(z) = \frac{1}{1+e^{-z}}$, $z \in \mathbb{R}^K$

The derivative of $\sigma(z)$ is given by

$$\frac{d}{dz}(\sigma(z)) = \frac{1}{1+e^{-z}}. \tag{1}$$

By the Quotient Rule:

$$\left(\frac{f}{g}\right)' = \frac{f' \cdot g - f \cdot g'}{g^2}. \tag{2}$$

Therefore, it is possible to write

$$\frac{d}{dz}(\sigma(z)) = \frac{(1)' \cdot (1+e^{-z}) - (1) \cdot (1+e^{-z})'}{(1+e^{-z})^2}. \tag{3}$$

Since $\frac{d}{dz}(1) = 0$ and $\frac{d}{dz}(1+e^{-z}) = -e^{-z}$, we get:

$$\frac{d}{dz}(\sigma(z)) = \frac{e^{-z}}{(1+e^{-z})^2}. \tag{4}$$

With some simplifications we arrive at:

$$\frac{d}{dz}(\sigma(z)) = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot \frac{1+e^{-z}-1}{1+e^{-z}}, \tag{5}$$

$$\sigma(z) \cdot \left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right), \tag{6}$$

$$\sigma(z)' = \sigma(z) \cdot (1-\sigma(z))_{C.E.D.}. \tag{7}$$

1.2 Computation of the first and second derivatives of a binary classification problem with $y \in \{\pm 1\}$ and prove that binary logistic loss is convex as a function of z

Since $y = +1$:

$$L(z;y) = -log[\sigma(z)]. \tag{8}$$

The first derivative can be computed as

$$L'(z;y=+1) = \frac{d}{dz}(-log[\sigma(z)]) = -\left(\frac{1}{\sigma(z} \cdot \sigma(z)'\right). \tag{9}$$

Therefore,

$$L'(z;y=+1) = -\frac{e^{-z}}{1+e^{-z}}, \tag{10}$$

Now, for the second derivative

$$L''(z;y=+1) = \frac{d}{dz}\left(-\frac{e^{-z}}{1+e^{-z}}\right). \tag{11}$$

Applying once again the Quotient Rule, we get:

$$L''(z; y=+1) = -\left[\frac{(e^{-z})' \cdot (1+e^{-z}) - (e^{-z}) \cdot (1+e^{-z})'}{(1+e^{-z})^2}\right]. \tag{12}$$

After some simplifications we arrive at:

$$L''(z; y=+1) = \frac{e^{-z}}{(1+e^{-z})^2}, \tag{13}$$

In order to prove that the logistic loss is convex with respect to z, we need to prove that the second derivative in order of z is greater or equal to zero in all its domain. Which is simple since both the numerator and the denominator of $L''(z; y=+1)$ are positive in $\mathbb{R}$.

1.3 Computation of the Jacobian matrix of the softmax transformation on point z

Let $z \in \mathbb{R}^K$. The softmax transformation is a a function from $\mathbb{R}^K$ to $\mathbb{R}^K$ that can be written as

$$[softmax(z)]_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{14}$$

for j=1,...,K. Note that all values of the function are non negative and all values obtained sum to 1, because the function is normalized.

Consider the Jacobian to be a K-by-K matrix and the (j,k)-th is given by

$$\frac{\partial}{\partial z_k}[softmax(z)]_j.$$

Due to all softmax values being strictly positive, it is possible to take logarithmic derivative, *i.e.* the partial derivative of the *log* of the output, instead of the partial derivative of the output.

$$\frac{\partial}{z_k} log[softmax(z)]_j = \frac{1}{[softmax(z)]_j}\frac{\partial}{\partial z_k}[softmax(z)]_j. \tag{15}$$

Now, we take the right-side of the result expression (15) follows the chain rule and can depicted as

$$\frac{\partial}{\partial z_k}[softmax(z)]_j = [softmax(z)]_j \frac{\partial}{\partial z_k}[softmax(z)]_j. \tag{16}$$

The left-side of the result expression (15) simplifies when multiplied by (16). Thus, it is possible to obtain the following expression

$$\frac{\partial}{\partial z_k}[softmax(z)]_j = [softmax(z)]_j \frac{\partial}{\partial z_k} log[softmax(z)]_j. \tag{17}$$

The *log* of the softmax function can be defined as

$$log[softmax(z)]_j = log\left(\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\right) = z_j - log\left(\sum_{k=1}^{K} e^{z_k}\right). \tag{18}$$

From (17) and (18) we get:

$$\frac{\partial}{\partial z_k} log[softmax(z)]_j = \frac{\partial z_j}{\partial z_k} - \frac{\partial}{\partial z_k} log\left(\sum_{k=1}^{K} e^{z_k}\right). \tag{19}$$

It is possible to conclude that $\frac{\partial z_j}{\partial z_k}$ will take the value 1 when $(j = k)$ and the value 0 otherwise. Thus, when $j = k$ it is possible to infer from (19) the following result

$$\frac{\partial}{\partial z_k} log[softmax(z)]_j = 1 - \frac{1}{\sum_{k=1}^{K} e^{z_k}} \frac{\partial}{\partial z_k} log\left(\sum_{k=1}^{K} e^{z_k}\right) = 1 - \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}}, \tag{20}$$

given that the partial derivative of a sum is known to be

$$\frac{\partial}{\partial z_k} log\left(\sum_{k=1}^{K} e^{z_k}\right) = \frac{\partial}{\partial z_k} e^{z_k} = e^{z_k}. \tag{21}$$

The result can be simplified as

$$\frac{\partial}{\partial z_k} log[softmax(z)]_j = 1 - [softmax(z)]_k \tag{22}$$

Now, when $(j = k)$, it is feasible to conclude

$$\frac{\partial}{\partial z_k} log[softmax(z)]_j = [softmax(z)]_j \cdot \frac{\partial}{\partial z_k} log[softmax(z)]_j = [softmax(z)]_j (1 - [softmax(z)]_k).$$

Let $[softmax(z)]_j = [s(z)]_j$ be taken as notation simplification.

Finally, is now possible to compute the K-by-K Jacobian matrix, J, of the softmax transformation on point z as

$$J = \begin{bmatrix} [s(z)]_1(1 - [s(z)]_1) & \cdots & -[s(z)]_1 \cdot [s(z)]_K \\ \vdots & \ddots & \vdots \\ -[s(z)]_K \cdot [s(z)]_1 & \cdots & [s(z)]_K(1 - [s(z)]_K) \end{bmatrix} \tag{23}$$

1.4 Compute the gradient and Hessian of the multinomial logistic loss with respect to z. Prove this loss is convex with respect to z.

The multinomial logistic loss is defined as

$$L(z; y = j) = -log[softmax(z)]_j \tag{24}$$

The gradient of this loss with respect to z, as shown in the lecture 4 slide 49 of the theoretical classes, can be depicted as

$$\frac{\partial L(z; y = j)}{\partial z_k} L(z; y = j) = [softmax(z)]_j - 1_{(y=k)} \tag{25}$$

Now, it is possible to compute the Hessian of the multinomial logistic loss with respect to z by computing the gradient of expression (25). Thus, each element of the Hessian is given by

$$\frac{\partial^2 L(z, y = j)}{\partial z_j \partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{\partial L(z, y = j)}{\partial z_j}\right) = \frac{\partial}{\partial z_i}([softmax(z)]_j - 1_{(y=k)}) = \frac{\partial}{\partial z_i}[softmax(z)]_j$$

The Hessian matrix, H, can be computed as follows:

$$H = \frac{\partial^2 L(z, y=j)}{\partial z_j \partial z_i} = \begin{bmatrix} \frac{\partial[softmax(z)]_1}{\partial z_1} & \cdots & \frac{\partial[softmax(z)]_1}{\partial z_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial[softmax(z)]_j}{\partial z_1} & \cdots & \frac{\partial[softmax(z)]_j}{\partial z_j} \end{bmatrix} \tag{26}$$

Is is possible to conclude the Hessian of the loss (26) is the same as the Jacobian matrix of the softmax transformation computed in (23).

Let's now prove the multinomial logistic loss with respect to z is convex. For that, is necessary to prove the Hessian, H, is a positive semidefinite matrix.

Consider the following expression

$$v^T \cdot H \cdot v = \sum_i \sum_j v_i \cdot v_j \cdot H_{i,j}.$$

Thus, to prove convexity is necessary to prove

$$v^T \cdot H \cdot v \geq 0, \tag{27}$$

for any v.

Let $[softmax(z)]_j = s_j$ be a notation simplification and $diag(\cdot)$ represent a diagonal matrix. From the results obtained in (23) the following steps were taken to prove (27):

$$\begin{aligned} v^T \cdot H \cdot v \geq 0 \\ \Leftrightarrow v^T \cdot (diag(s_j) - s_j \cdot s_i^T) \cdot v \geq 0 \\ \Leftrightarrow v^T \cdot diag(s_j) \cdot v \geq v^T \cdot ((s_j \cdot s_i^T)) \cdot v \\ \Leftrightarrow \sum_j v_j^2 \cdot s_j \geq (v^T \cdot s_j)^T \cdot (v^T \cdot s_j) \\ \Leftrightarrow \sum_j v_j^2 \cdot s_j \geq (v^T \cdot s_j)^2 \\ \Leftrightarrow \sum_j v_j^2 \cdot s_j \geq \left(\sum_j v_j \cdot s_j \right)^2 \\ \Leftrightarrow \sum_j v_j^2 \cdot \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \geq \left(\sum_j v_j \cdot \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \right)^2 \\ \Leftrightarrow \left(\sum_j v_j^2 \cdot e^{z_j} \right) \cdot \left(\frac{(\sum_{k=1}^K e^{z_k})^2}{\sum_{k=1}^K e^{z_k}} \right) \geq \left(\sum_j v_j \cdot e^{z_j} \right)^2 \\ \Leftrightarrow \left(\sum_j v_j^2 \cdot e^{z_j} \right) \cdot \left(\sum_j e^{z_j} \right) \geq \left(\sum_j v_j \cdot e^{z_j} \right)^2 \\ \Leftrightarrow \left(\sum_j v_j \cdot e^{z_j} \right)^2 \leq \left(\sum_j v_j^2 \cdot e^{z_j} \right) \cdot \left(\sum_j e^{z_j} \right) \end{aligned}$$

Now, it is useful to recall the Cauchy-Schwartz inequality that can be written as

$$\left(\sum_k a_k \cdot b_k \right)^2 \leq \left(\sum_k a_k^2 \right) \left(\sum_k b_k^2 \right).$$

In order to apply the above inequality to this case consider

$$a_k^2 = v_j^2 \cdot e^{z_j}, \quad b_k^2 = e^{z_j},$$

and compute

$$a_k \cdot b_k = (\sqrt{v_j^2 \cdot e^{z_j}}) \cdot (\sqrt{e^{z_j}}) = v_j \cdot e^{z_j}.$$

Therefore, (27) is proven and is now possible to affirm the multinomial logistic loss with respect to z is convex.

1.5 Show that in a linear model where $z = \phi W(x) + b$, 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. Check if this is also true when z is not a linear function of the model parameters.

First, note that, if (W,b) fixed, z is linear with respect to ϕ and also linear with respect to (W,b), if ϕ is fixed. Thus, function $g(W,b) = W\phi(x) + b$ can be considered an affine map.

It is known that the composition of two functions, $(f \circ g)$, keeps the convexity if: g is an affine map and f is a convex function; and if g is a convex function and f is a non-decreasing convex function.

Since $g(W,b) = W\phi + b$ is an affine map and $f(z, y = j) = -log[softmax(z)]_j$ was proven to be a convex function with respect to z, it is possible to conclude that the multinomial logistic loss is also convex with respect to the model parameters (W; b), and therefore a local minimum is also a global minimum.

When z is not a linear function of the model parameters, then the function g is not an affine map. Thus, in order for the composition to be convex, f must be a non-decreasing convex function.

When analysing the first derivative of the loss (25), it is possible to conclude that it is either negative or 0, when $(y = k)$. Therefore, the loss function is not a non-decreasing convex function.

So, for the multinomial logistic loss to be convex with respect to the model parameters (W; b), z must be a linear function of the model parameters.

2 Question 2

In this question we studied the regression of house prices with linear models and neural networks.

2.1 Prove that the squared error loss, L is convex with respect to (W, b).

In order to prove L is convex we need to prove the Hessian Matrix is positive semidefinite, that is, if all the values of the main diagonal are positive.

$$L(z; y) = \frac{1}{2}\|z - y\|_2^2 \quad being \quad z = W^T\phi(x) + b, \tag{28}$$

$$\frac{d}{dw}\left(\frac{1}{2}\|W^T\phi(x)+b-y\|_2^2\right), \tag{29}$$

Applying the chain rule, $h'(x) = f'(g(x)) \cdot g'(x)$, we get:

$$\frac{dL}{dw} = \phi(x)(W^T\phi(x)+b-y), \tag{30}$$

Also, as seen before, it possible to write

$$\frac{dL}{db} = (W^T\phi(x)+b-y), \tag{31}$$

Now, to build the Hessian Matrix it is necessary to compute the following second derivatives:

$$\frac{d^2L}{dw^2} = \phi(x)^2, \tag{32}$$

$$\frac{d^2L}{dwdb} = \frac{d^2L}{dbdw} = \phi(x), \tag{33}$$

$$\frac{d^2L}{db^2} = 1, \tag{34}$$

Since $\frac{d^2L}{dw^2} \geq 0$, $\frac{d^2L}{db^2} \geq 0$, we can say that $L(z;y)$ is convex with respect to (W,b).

2.2 (a) Linear Regression

After training for 150 epochs the Loss of the training set was 0.111 and the Loss of the test set was 0.238, having its lowest value around 60 epochs at 0.196.

It is possible to see in the plot shown in Figure 1, the difference between the mean squared error on the training and test sets as the number of epochs increases tends to increase, since the MSE value of the training set goes to zero and with the test set the value decreases at a lower rate and all of a sudden starts to go up again.

This is phenomenon is commonly called overfitting and it usually happens when the model trains for too long on the sampled data.

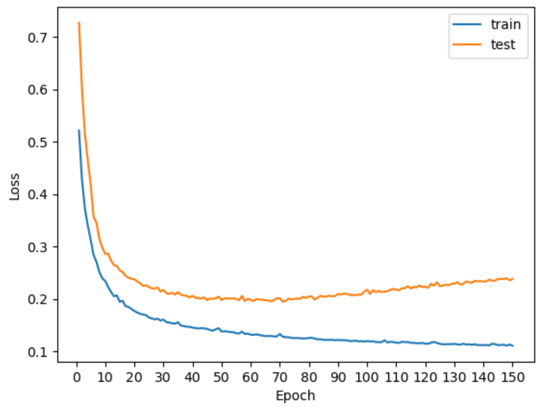


Figure 1: Performance plot for the linear regression

As we can see from the distance between the weight vector and the weight vector computed analytically as the number of epochs increases illustrated in Figure 2, the distance decreases non linearly with epochs iterations which is good. Since the distance is decreasing we know the train is working, and because it decreases faster in the beginning and slower by the end we know the method is converging and getting ever so slightly closer to the minimum.

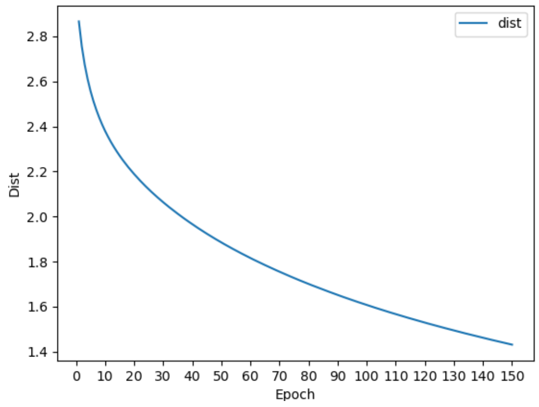


Figure 2: Distance between weight vector

2.2 (b) Neural Network

After training for 150 epochs the Neural Network presented a Loss of 0.099 in the training set, which is slightly better then the linear model, and 0.165 for the test set, that shows a big difference from the values reported in section 2.2(a). This difference is possible to understand, when we take a look at the MSE plot in Figure 3. Immediately, we notice that in this graph the model does not seem to be so prone to overfitting, making it able to follow the training MSE closer, resulting in a better performance.

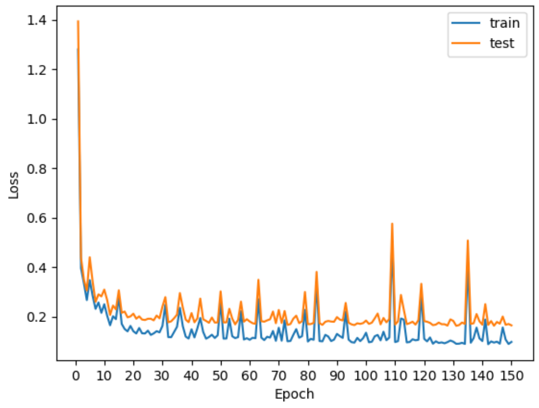


Figure 3: Performance plot for the Neural Network

3 Question 3

In this question we implemented different linear classifiers for a simple image classification problem, using the Fashion MNIST dataset. The results obtained are reported below.

3.1 (a) Perceptron

In this question we implemented a liner classifier, the perceptron, and trained with 20 epochs.

The plot of the accuracies as a function of the epoch number, for the test and validation sets, is represented in Figure 4. The performance on the validation set is 71.26%; the performance on the test set is 70.13%.

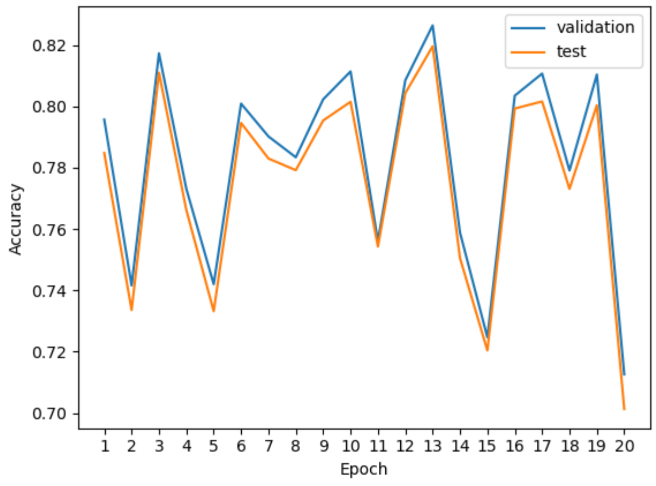


Figure 4: Performance plot for the perceptron

3.1 (b) Logistic Regression

Now, we repeated the previous exercise, for the logistic regression linear classifier.

The plot obtained is shown in Figure 5. The performance on the validation set is 85.03%; the performance on the test set is 84.03%.

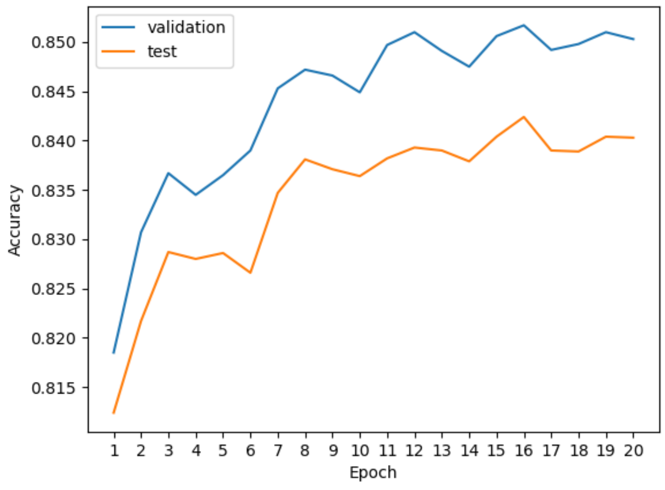


Figure 5: Performance plot for the logistic regression classifier

3.2 (a) MLP - Justification

The perceptron in a multi-class classification problem is able to find the weights and bias that split the feature space into regions delimited by hyperplanes. On the other hand, the multi-layer perceptron has the capability to learn non-linear models, which for this particular task is better because, in images, the number of features being considered is very high. Thus, we need a more complex model that can better separate them in the different classes.

If we only use linear activation functions in the multi-layer perceptron the expressive power of the MLP remains the same, does not increase.

3.2 (b) MLP

Here we implemented a multi-layer perceptron with a single hidden layer.

The plot obtained of the accuracies, for the validation and test set, as a function of the the epoch number is presented in Figure 6. The performance on the validation set is 87.75%; the performance on the test set is 86.78%.

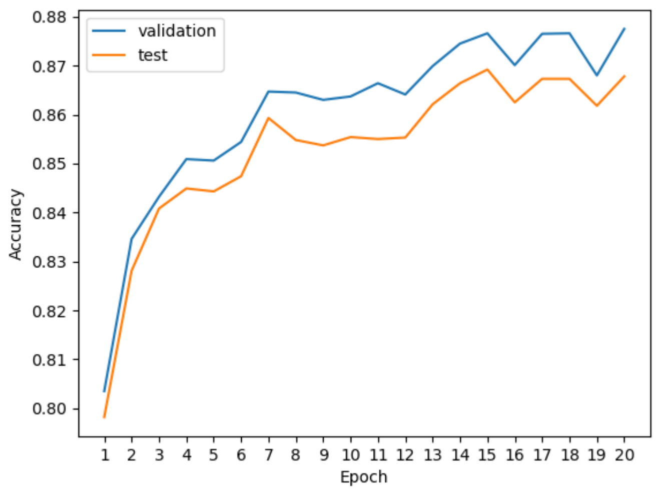


Figure 6: Plot of the multi-layer perceptron accuracy

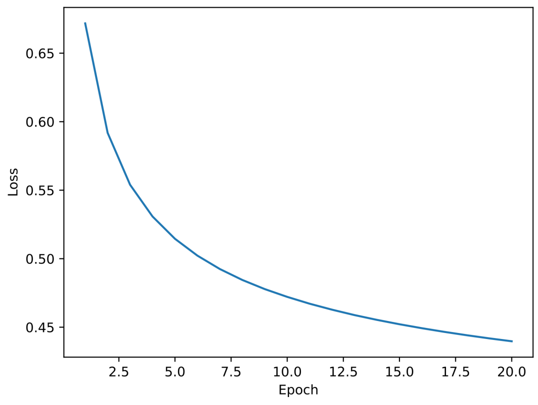
4 Question 4

In this question we implemented classifiers using a deep learning framework with automatic differentiation.

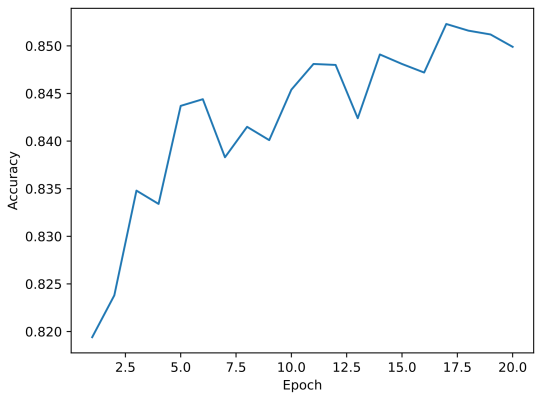
4.1 Logistic Regression

We now implemented a linear model with logistic regression.

The best configuration was obtained with a learning rate of 0.001. The plots obtained with this configuration for the training loss and the validation accuracy, both as a function of the epoch number, is reported in Figures 7(a) and 7(b). The final test accuracy obtained for this configuration is: 83.88%.



(a) Training Loss



(b) Validation Accuracy

Figure 7: Resulting plots for the linear model with logistic regression

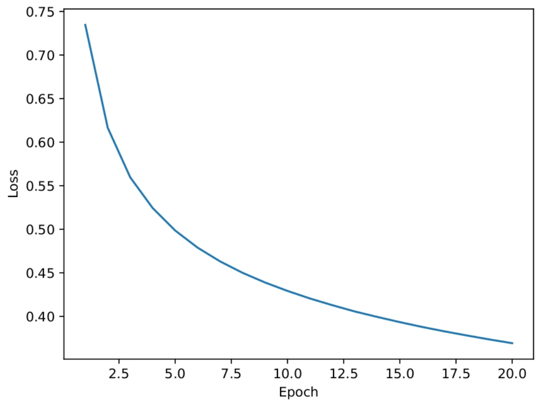
4.2 Feed-forward neural network

Here we implemented a feed-forward neural network with a single layer, using dropout regularization.

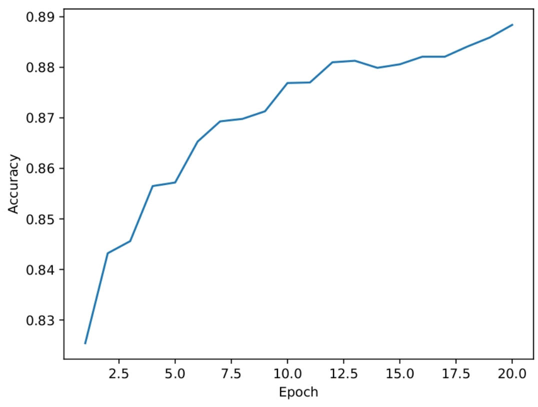
After tunning, the best configuration obtained for this model is reported in the table 1 and the plots obtained for this configuration for the training loss and the validation accuracy, as a function of the epoch number, are presented in Figures 8(a) and 8(b). The final test accuracy obtained for this configuration is: 87.94%.

Tabela 1: Hyperparameters corresponding to the best configuration

Learning rate	0.001
Hidden Size	200
Drouptout	0.3
Activation	ReLU
Optimizer	SGD



(a) Training Loss



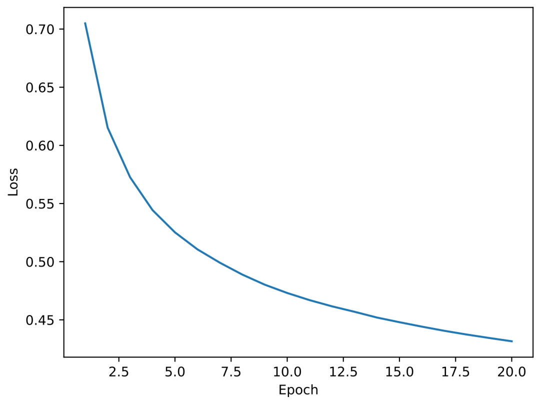
(b) Validation Accuracy

Figure 8: Resulting plots for the feed-forward neural network

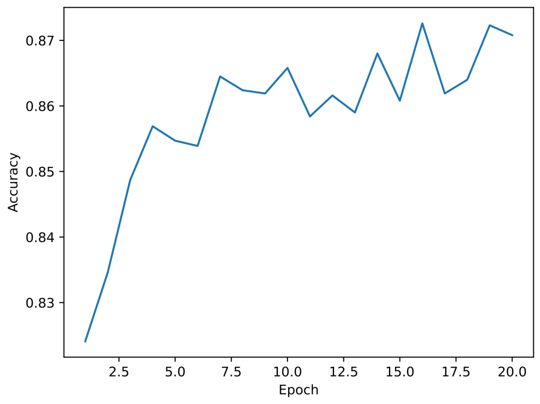
4.3 Increase the model layers

In this question, we increased the model to 2 and 3 layers.

The best configuration was obtained to 3 layers. Plots obtained for this configuration are presented in Figures 9(a) and 9(b). The final test accuracy obtained for this configuration is: 86.68%.



(a) Training Loss



(b) Validation Accuracy

Figure 9: Resulting plots for the feed-forward neural network with 3 layers