Deep Learning

MEEC21/MECD21

Homework Assignment #1

Authors:

Afonso Araújo (96138) afonso.d.araujo@tecnico.ulisboa.pt
Emma Dennis-Knieriem (105545) emma.dennis-knieriem@tecnico.ulisboa.pt

Group 18

2022/2023 – 1st Semester, P1

Contents

Individual Contribution

For this homework assignment, even though each group member was aware of the whole project, to save time, they divided tasks. While one group member tried to solve question 3 and the theoretical portion of question 1 (Emma), the other group member wrote down the code for questions 1 and 2 (Afonso).

1 Question 1

1.1 1

1.1.1 a)

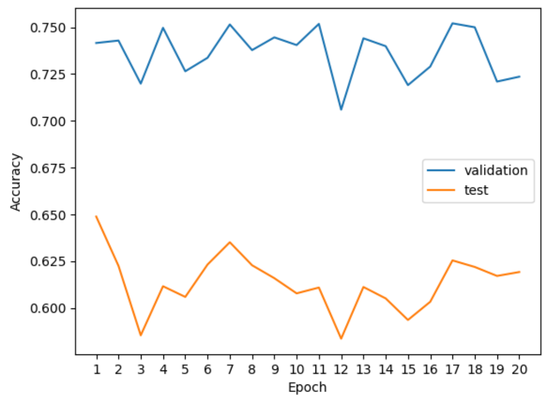

Figure 1: Accuracy Evolution for Validation and Test sets for Perceptron Learning Algorithm

1.1.2 b)

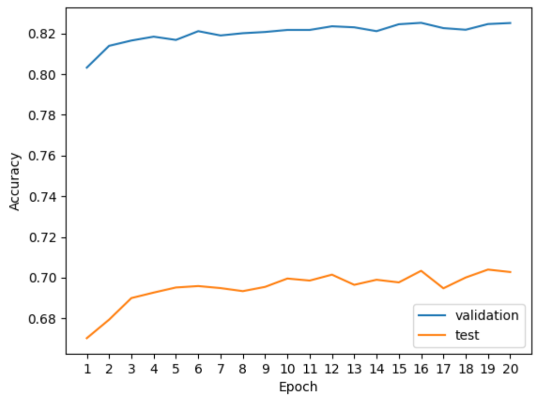

Figure 2: Accuracy Evolution for Validation and Test sets for Logistic Regression Algorithm

1.2 2

1.2.1 a)

If we were to use only linear activation functions, then our MLP (multi-layer perceptron) would behave like a simple Perceptron algorithm. Now, if we are to analyze a single Perceptron algorithm we can see that it needs the data to be linearly separable for it to create hyperplanes and successfuly separate the data. If it cannot, then we need to apply feature engineering to separate the data or use a more expressive algorithm, such as a MLP.

An MLP adds hidden layers in between the input and output layers. In between every layer, except for the first, the MLP applies a non-linear activation function such that it allows the MLP to process more complex problems. As stated above, these successive non-linear activations help to separate and flatten the data, just as we did to the Kuzushiji-MNIST dataset, which a simple perceptron would have a hard time with.

The Kuzushiji dataset is a collection of images of handwritten Japanese characters. In this case, an MLP with non-linear activations is able to learn more complex relationships between the pixel values in the images and the corresponding characters, therefore achieving better performance on the character recognition task.

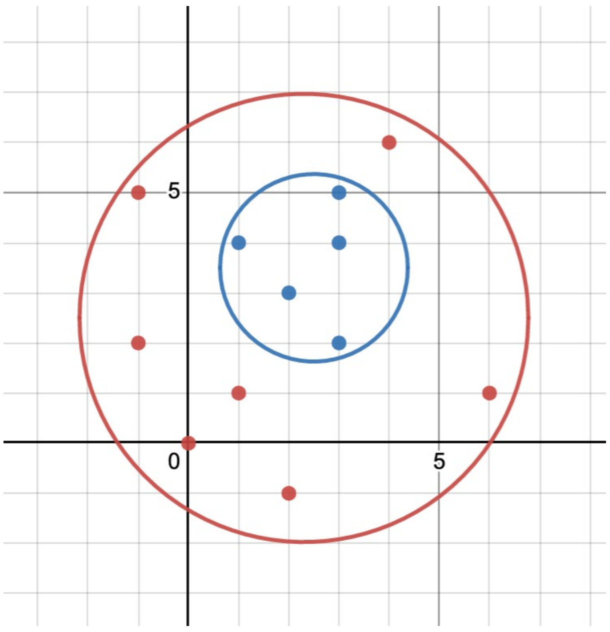

Figure 3: An example of data that are separable, but not linearly by a hyperplane. Here, an MLP could be implemented to separate the data instead.

1.2.2 b)

Below we have our training loss and the accuracy evolution throughout 20 epochs for a single layered MLP. As for the code it was an adaption of Practical_04 as given in class taking into account the existence of biases and weights, where the latter follows a normal distribution.

Total Loss	5507.1997
Final Test Accuracy	0.8537
Validation Accuracy	0.9325

Table 1: Total Loss for 1 Hidden Layer

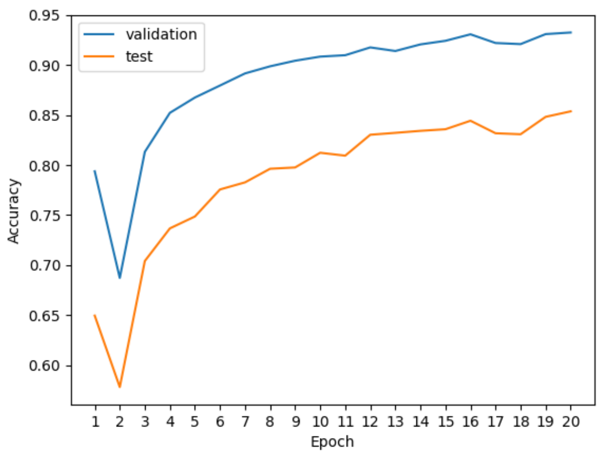

Figure 4: Accuracy Evolution for Validation and Test sets for MLP with 1 Hidden Layer

2 Question 2

2.1 1

	LR = 0.001	LR = 0.01	LR = 0.1
Training Loss	0.5874	0.6164	2.7026
Valid Accuracy	0.8256	0.8033	0.7411
Final Test Accuracy	0.7019	0.6806	0.6133

Table 2: Parameter Evolution for Increasing Learning Rates

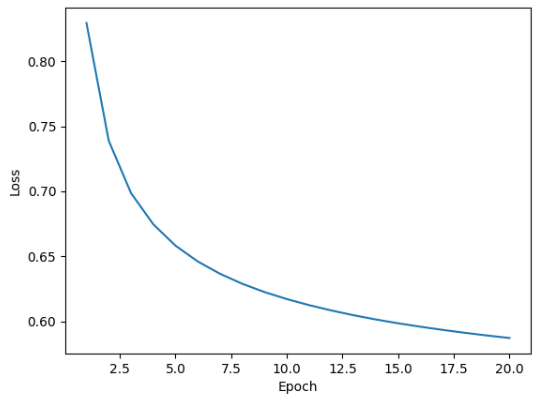

Figure 5: Loss Evolution for Logistic Regression Algorithm, using PyTorch

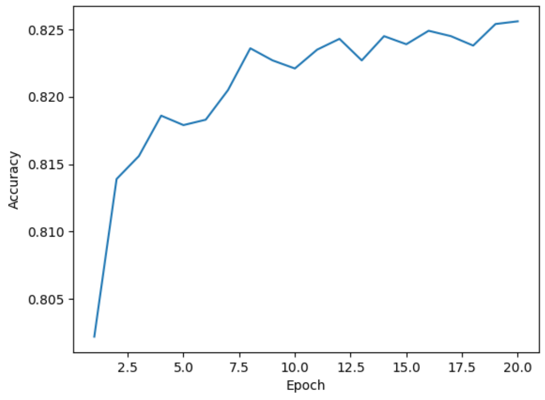

Figure 6: Accuracy Evolution for Logistic Regression Algorithm, using PyTorch

2.2 2

The following is a table that compares the DEFAULT VALUES of Table 1 as given on the problem statement with an individual tuning of each parameter. This means that if we state "Learning Rate = 0.1" then, we are using every default value except for such learning rate, or "Tanh as Activation Function" we are using every default value except for the activation function being Tanh and not ReLU.

	Final Test Accuracy	Validation Accuracy	Training Loss
DEFAULT VALUES	0.8618	0.9396	0.3489
Learning Rate = 0.1	0.8679	0.9432	0.2169
Learning Rate = 0.001	0.7440	0.8645	0.7873
Hidden Size = 200	0.8768	0.9452	0.2983
Tanh as Act Function	0.8201	0.9146	0.4513
Dropout = 0.5	0.8417	0.9310	0.4376

Table 3: Final Accuracy and Validation for different configurations

As we can see the best learning rate is 0.1,

$$0.8679 > 0.8618 > 0.7440 \tag{1}$$

the best hidden size is 200, since we should always compare with the DEFAULT row,

$$0.8768 > 0.8618 \tag{2}$$

the best activation function is the ReLU activation function, as seen on the DEFAULT row,

$$0.8618 > 0.8201 \tag{3}$$

and the best dropout value is 0.3, again referencing to the DEFAULT row,

$$0.8618 > 0.8417 \tag{4}$$

Therefore if we are to "cherry-pick" each of the best parameters, taking DEFAULT VALUES as Table 1 presented on the problem we have the following Final Test Accuracy for a Hidden Size of 200, ReLU activation function, dropout factor of 0.3 and learning rate of 0.1:

Final Test Accuracy	0.8939
Validation Accuracy	0.9587
Training Loss	0.1332

Table 4: Final Test and Validation Accuracy for best configuration

And its Loss and Accuracy evolution throughout 20 epochs:

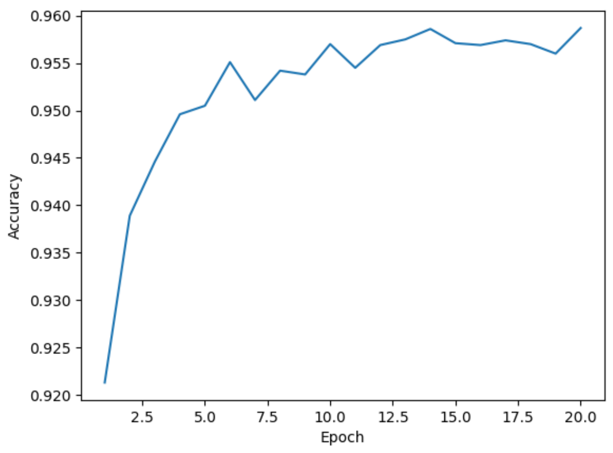

Figure 7: Accuracy Evolution for best configuration, using PyTorch

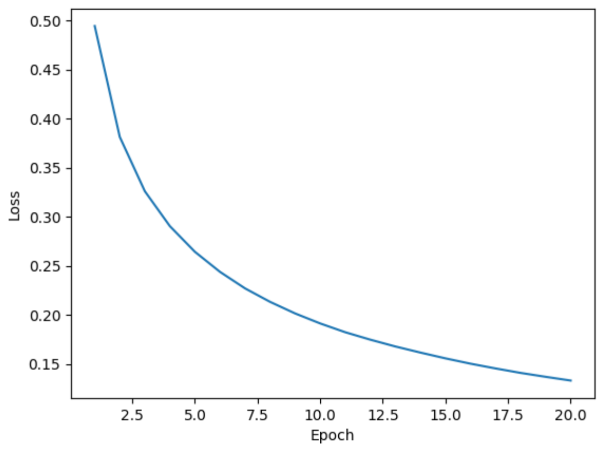

Figure 8: Loss Evolution for best configuration, using PyTorch

2.3 3

Below is the parameter statement for 2 and 3 hidden layers, using the Table 1 of the problem statement as default.

	2 Layers	3 Layers
Training Loss	0.3631	0.4154
Valid Accuracy	0.9444	0.9424
Final Test Accuracy	0.8672	0.8636

Table 5: Parameter Evolution for Increasing Hidden Layers with Default Values

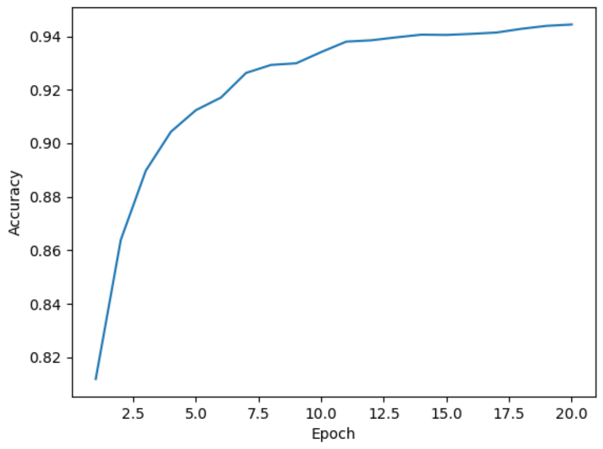

Figure 9: Accuracy Evolution for MLP with 2 Hidden Layers, using PyTorch

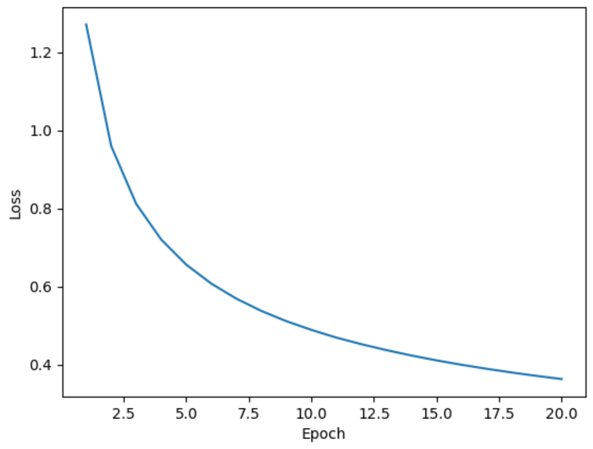

Figure 10: Loss Evolution for MLP with 2 Hidden Layers, using PyTorch

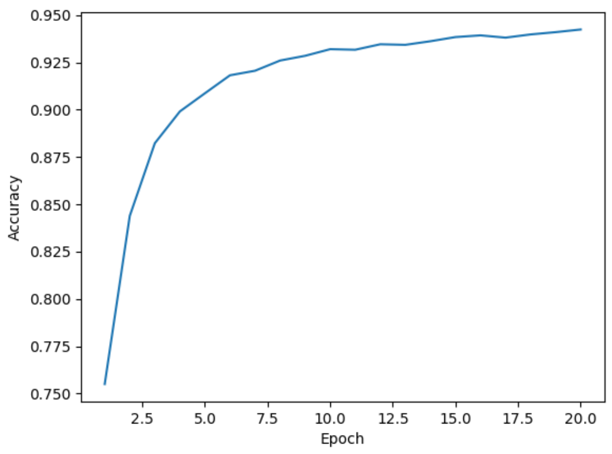

Figure 11: Accuracy Evolution for MLP with 3 Hidden Layers, using PyTorch

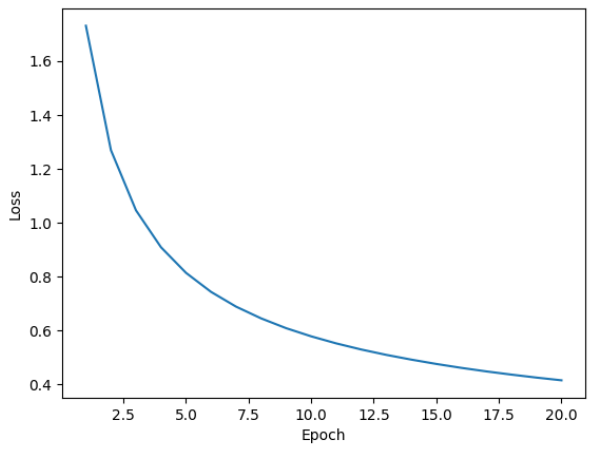

Figure 12: Loss Evolution for MLP with 3 Hidden Layers, using PyTorch

3 Question 3

3.1 1

$\hat{y} = v^T h$ where $h = g(Wx)$ and $g(z) = z^2$.

To show that h is a linear transformation of $\phi(x)$, and therefore that h can be written as $h = A_\Theta \phi(x)$, it must be true that $(Wx)^2 = A_\Theta \phi(x)$.

$(Wx)^2 = Wx \odot Wx$, where $\odot$ denotes the Frobenius inner product. $W \in \mathbb{R}^{K\times D}$, $x \in \mathbb{R}^K$.

$$(Wx)^2 = \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & & \vdots \\ w_{K1} & \dots & w_{KD} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} \odot \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & & \vdots \\ w_{K1} & \dots & w_{KD} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix},$$

where w_{ij} is the element in the i^{th} row and j^{th} column of W.

$$(Wx)^2 = \begin{bmatrix} w_{11}x_1 + \ldots + w_{1D}x_D \\ \vdots \\ w_{K1}x_1 + \ldots + w_{KD}x_D \end{bmatrix} \odot \begin{bmatrix} w_{11}x_1 + \ldots + w_{1D}x_D \\ \vdots \\ w_{K1}x_1 + \ldots + w_{KD}x_D \end{bmatrix},$$

where each $(Wx) \in \mathbb{R}^{K\times 1}$.

$$(Wx)^2 = \begin{bmatrix} w_{11}^2x_1^2 + 2w_{11}x_1w_{1D}x_D \ldots + w_{1D}^2x_D^2 \\ \vdots \\ w_{K1}^2x_1^2 + 2w_{K1}x_1w_{KD}x_D \ldots + w_{KD}^2x_D^2 \end{bmatrix},$$

where $(Wx)^2 \in \mathbb{R}^{K\times 1}$ since the Frobenius inner product is element-wise multiplication of matrices with identical dimensions.

$$(Wx)^2 = \begin{bmatrix} w_{11}^2 & 2w_{11}w_{1D} & \ldots & w_{1D}^2 \\ \vdots & \vdots & & \vdots \\ w_{k1}^2 & 2w_{K1}w_{KD} & \ldots & w_{KD}^2 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1x_D \\ \vdots \\ x_D^2 \end{bmatrix}.$$

Let $A_\Theta = \begin{bmatrix} w_{11}^2 & 2w_{11}w_{1D} & \ldots & w_{1D}^2 \\ \vdots & \vdots & & \vdots \\ w_{k1}^2 & 2w_{K1}w_{KD} & \ldots & w_{KD}^2 \end{bmatrix}$. $A_\Theta \in \mathbb{R}^{K\times \frac{D(D+1)}{2}}$ because $(Wx) \in \mathbb{R}^{K\times 1}$ (see above), and the number of columns of A_Θ is the same number of elements in an upper triangular matrix (including the main diagonal) of dimensions $D \times D$: $\frac{D(D+1)}{2}$.

Let $\phi(x) = \begin{bmatrix} x_1^2 \\ x_1x_D \\ \vdots \\ x_D^2 \end{bmatrix}$. $\phi(x) : \mathbb{R} \to \mathbb{R}^{\frac{D(D+1)}{2}}$, where the dimension of rows in the column vector, $\frac{D(D+1)}{2}$, is for the same reason as above.

3.2 2

$$\hat{y} = v^T h, \text{ and } h = A_\Theta\phi(x).$$
$$\hat{y} = v^T A_\Theta \phi(x)$$
$$\text{Let } c_\Theta^T = v^T A_\Theta, \text{ so that } \hat{y} = c_\Theta^T\phi(x).$$
$$c_\Theta^{TT} = A_\Theta^T v^{TT}$$
$$c_\Theta^T = A_\Theta^T v$$
$$c_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}} \text{ because } c_\Theta^T = v^T A_\Theta, \text{ where } A_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}.$$

No, this is not a linear model in terms of c_Θ. It is nonlinear in Θ because A_Θ depends on W, and W is an element of $\Theta = (W, v)$.

3.3 3

If given a c_Θ, v and W can be found such that $c_\Theta = v^T A_\Theta$, where $A_\Theta = \begin{bmatrix} w_{11}^2 & 2w_{11}w_{1D} & \dots & w_{1D}^2 \\ \vdots & \vdots & & \vdots \\ w_{k1}^2 & 2w_{K1}w_{KD} & \dots & w_{KD}^2 \end{bmatrix}$. $W \in \mathbb{R}^{K\times D}$ and $v \in \mathbb{K}$, so c_Θ has $D \times K + K$ unknowns. A_Θ represents $\frac{D(D+1)}{2}$ equations since $A_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$. When the number of equations are fewer than or equal to the number of unknowns, this problem will always have a solution. This is such a case, since $D\times K+K \geq \frac{D(D+1)}{2}$ and $K \geq D$. When $K < D$, there will either be a unique solution or no solution, since the number of equations is greater than the number of unknowns, so finding v^T and A_Θ given c_Θ is more constrained and less flexible.

3.4 4

$$\hat{y_n}(x_n|c_\Theta) = c_\Theta^T \phi(x).$$

The objective here is to minimize L over c_Θ, where $L(c_\Theta|D) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y_n}(x_n|c_\Theta) - y_n)^2$.

To do this, $L(c_\Theta|D)$ will be set equal to $\frac{1}{2}(c_\Theta^T\phi(X) - Y)^2$. The derivative of both sides will be taken with respect to c_Θ, and the side denoting the derivative of L will be set to 0.

$$\begin{gathered}
L(c_\Theta|D) = \tfrac{1}{2}(c_\Theta^T\phi(X) - Y)^2 \\
L(c_\Theta|D) = \tfrac{1}{2}(c_\Theta^T\phi(X) - Y)^T(c_\Theta^T\phi(X) - Y) \\
\tfrac{\partial L}{\partial c_\Theta} = \tfrac{\partial}{\partial c_\Theta}[\tfrac{1}{2}(c_\Theta^T\phi(X) - Y)^T(c_\Theta^T\phi(X) - Y)] \\
0 = \tfrac{\partial}{\partial c_\Theta}[\tfrac{1}{2}c_\Theta^{TT}\phi(X)^T c_\Theta^T\phi(X) + c_\Theta^{TT}\phi(X)^T(-Y)^T + c_\Theta^T\phi(X)(-Y) + (-Y)^T(-Y)] \\
0 = \tfrac{\partial}{\partial c_\Theta}\tfrac{1}{2}[c_\Theta^T c_\Theta\phi(X)^T\phi(X) - 2c_\Theta^T\phi(X)^T(Y) + Y^TY] \\
0 = \tfrac{\partial}{\partial c_\Theta}\tfrac{1}{2}[c_\Theta^2\phi(X)^2 - 2c_\Theta\phi(X)^T(Y) + Y^2] \\
0 = \tfrac{1}{2}[2c_\Theta\phi(X)^T\phi(X) - 2\phi(X)^TY] \\
0 = c_\Theta\phi(X)^T\phi(X) - \phi(X)^TY \\
\phi(X)^TY = c_\Theta\phi(X)^T\phi(X) \\
\phi(X)^TY[\phi(X)^T\phi(X)]^{-1} = c_\Theta\phi(X)^T\phi(X)[\phi(X)^T\phi(X)]^{-1} \\
\phi(X)^TY[\phi(X)^T\phi(X)]^{-1} = c_\Theta
\end{gathered}$$

The estimate for c_Θ that minimizes L, $\hat{c_\Theta}$, is
$\hat{c_\Theta} = \phi(X)^TY[\phi(X)^T\phi(X)]^{-1}$, which is a closed form solution. This problem of global minimization is a convex optimization problem, which is unusual to see in feedforward neural networks.