Homework 1 - DL 2022/23

Group 43: César Reis (96849), Francisco Silva (97433)

Q1 and Q3 - Francisco Silva and César Reis

Q2 - César Reis

Question 1

1.a)

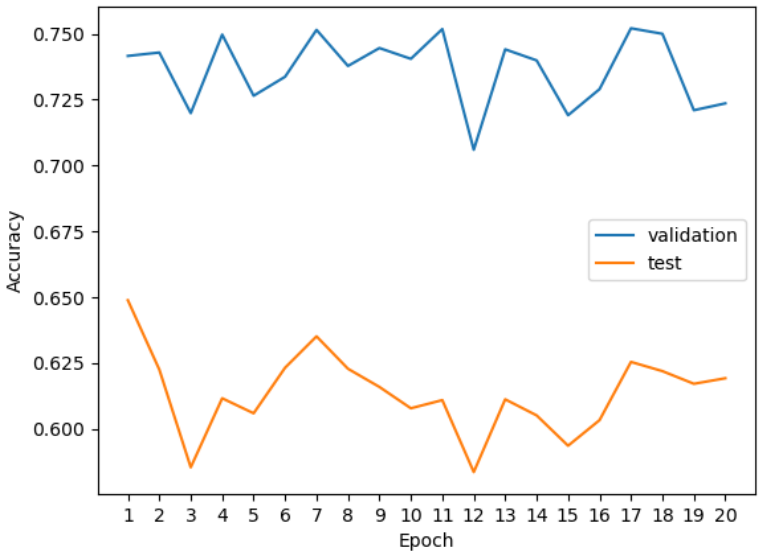


Figure 1. Perceptron performance of the validation and test set trained in 20 epochs.

1.b)

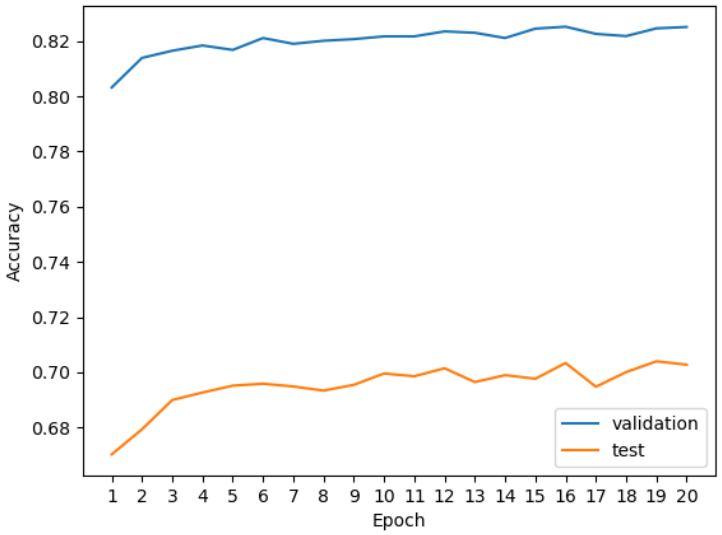


Figure 2. Logistic regression with Stochastic Gradient Descendent learning algorithm performance of the validation and test set trained in 20 epochs.

2.a)

Multilayer perceptrons (MLP) with non-linear activations act as universal approximators, able to capture and learn sophisticated patterns of the input brought by the non-linearity of the hidden units' activation functions regarding a particular task, and therefore, are very expressive compared to the simple perceptron (with only the linear output layer) of the question 1.a).

More precisely, the MLP expressive power, despite making it possible to separate (non-linearly) the input space, brings greater freedom to the model as well. That is, there are more parameters to be trained relative to the additional hidden units, which can lead the model to be based only on details, limiting its generalization (overfitting).

Considering only linear activation functions in hidden layers, the model becomes a combination of several simple linear regression models, and with the same number of weights to be updated in training this limitation remains without enough expressiveness for data non-linearly separated problems.

2.b)

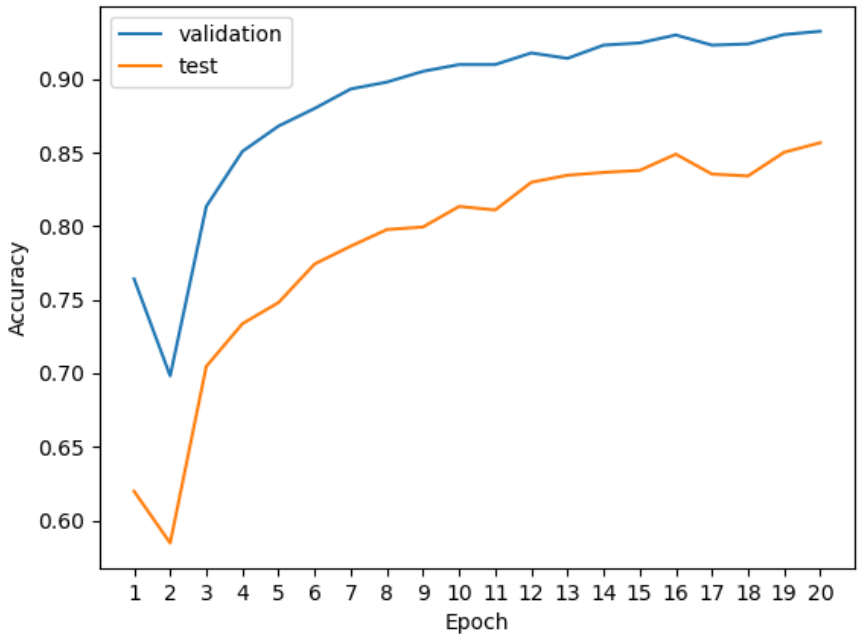


Figure 3. MLP with Stochastic Gradient Descendent learning algorithm performance of the validation and test set trained in 20 epochs.

Question 2

1.

Following 20 epochs of training logistic regression models with different learning rates, we found that the best configuration is the one using a *learning_rate* of **0.001**, with the highest validation accuracy in the last epoch of **82,56%** and **0.5874** of training loss.

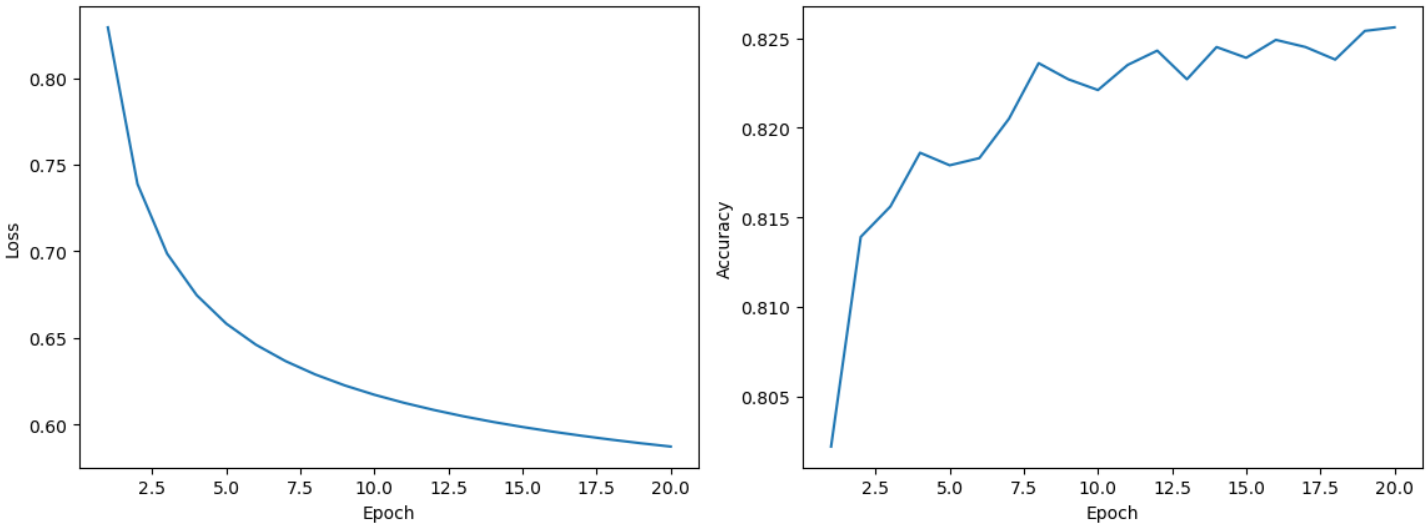


Figure 4. Performance of the best logistic regression configuration regarding the *leaning_rate* tuning, during 20 epochs of training, according to the training loss (at left) and validation accuracy (at right) per epoch.

In terms of generalization, we obtain a final accuracy of **70,19%** on the test set for this configuration.

2.

Following 20 epochs of training Feed-forward neural networks by tuning the hyperparameters considered in the question, we found that the best configuration corresponds to the tuning of the *hidden_size* hyperparameter when we have **200** hidden units in the single hidden layer, achieving the highest validation accuracy in the last epoch of **94,93%** and **0.2951** of training loss.

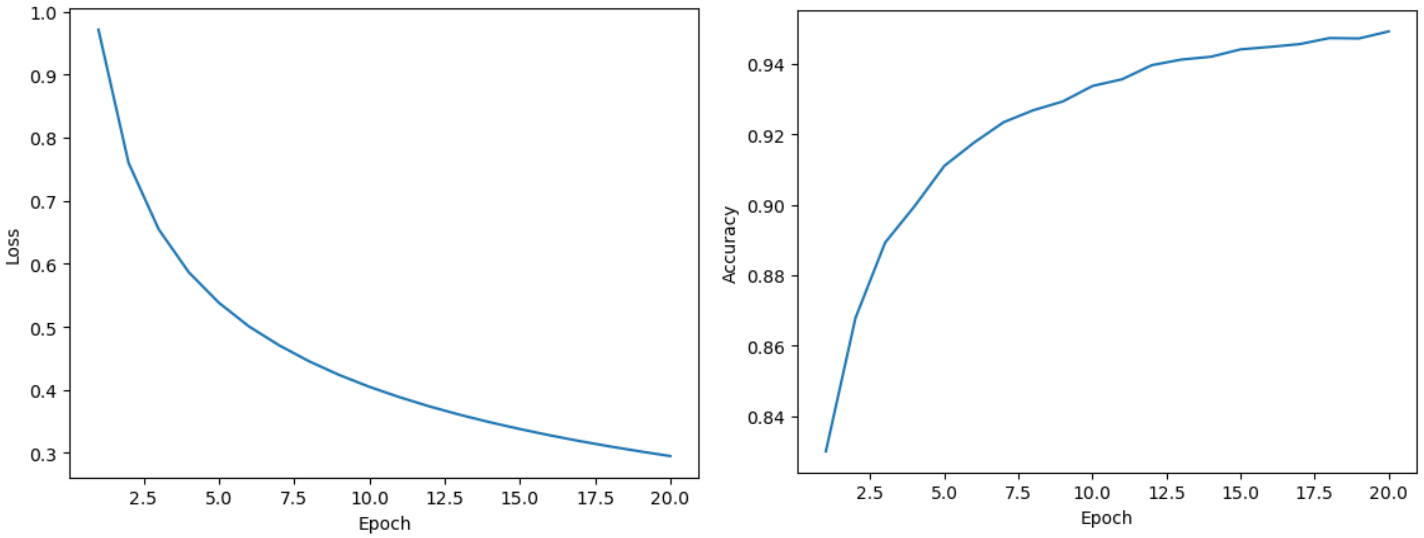


Figure 5. Performance of the best MLP configuration, during 20 epochs of training (batch size of 16), according to the training loss (at left) and validation accuracy (at right) per epoch.

In terms of generalization, we obtain a final accuracy of **87,99%** on the test set for this configuration.

3.

Keeping all the previous best configuration hyperparameters to tuning only the number of hidden layers during 20 epochs, lead us to identify that the model with **2** hidden layers achieves the highest validation accuracy in the last epoch of **95.72%** and **0.2820** of training loss.

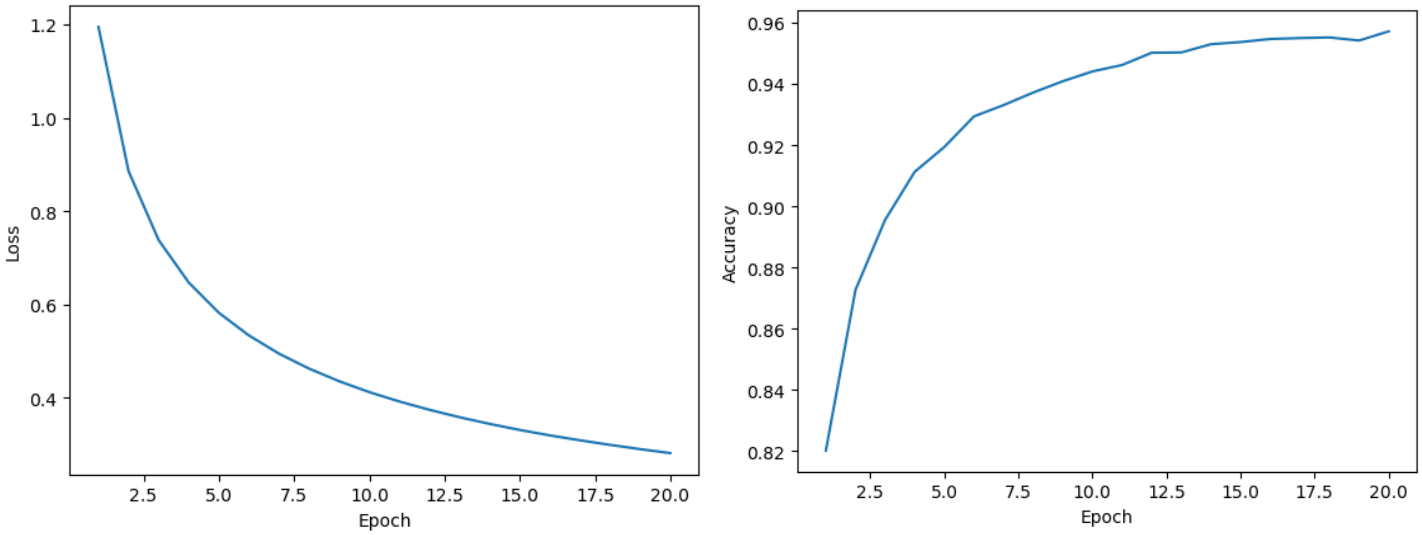


Figure 6. Performance of the best MLP configuration regarding the *hidden_layers* tuning, during 20 epochs of training (batch size of 16), according to the training loss (at left) and validation accuracy (at right) per epoch.

In terms of generalization, we obtain a final accuracy of **89,48%** on the test set for this configuration.

Question 3

1.

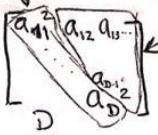
(3)

3.1

$$h = g(Wx) = g\left(\sum_{i=1}^{D} w_i x_i\right) \quad \text{with } w_i \text{ (column vector) of } W$$

$$= \left(\sum_{i=1}^{D} w_i x_i\right)^2 = \sum_{i=1}^{D} (w_i x_i)^2 + 2 \sum_{i<j}^{D} w_i x_i \odot w_j x_j$$

x_i is escalar



$$= \underbrace{\sum_{i=1}^{D} (w_i \odot w_i) x_i^2}_{\text{Diagonal}} + \underbrace{\sum_{i<j}^{D} (2 w_i \odot w_j) x_i x_j}_{\text{Upper triangular}}$$

$$\rightarrow D + \frac{D(D-1)}{2} = \frac{D(D+1)}{2}$$

Considering, for example, a matrix A_θ, with the first D colums, the vectors $(w_i \odot w_i)$ of the diagonal and the following $\frac{D(D-1)}{2}$ colums with respect to the vectors $(w_i \odot w_j)$ of upper triangular, we have: $h = A_\theta \phi(x)$, com $A_\theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$ e $\phi \in \mathbb{R}^{\frac{D(D+1)}{2}}$ c.q.d.

$$A_\theta = \underbrace{\left[(w_1 \odot w_1) \dots (w_D \odot w_D)\right.}_{D \text{ columns}} \underbrace{\left. 2(w_1 \odot w_2) \dots 2(w_1 \odot w_D) \dots 2(w_{D-1} \odot w_D)\right]}_{\frac{D(D-1)}{2} \text{ columns}}$$

and $\phi = \begin{bmatrix} x_1^2 \\ \vdots \\ x_D^2 \\ x_1 x_2 \\ x_1 x_D \\ x_{D-1} x_D \end{bmatrix}$ $\begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix}$ $\begin{matrix} D \text{ lines} \\ \\ \\ \\ \\ \frac{D(D-1)}{2} \text{ lines} \end{matrix}$

2.

3.2.

Model: $\hat{y} = v^T h$

We saw that h is a linear transformation of $\phi(x)$, so we can write:

$$h = A_\theta \phi(x), \text{ with } A_\theta \in \phi \text{ given.}$$

Thus, our model can be: $\hat{y} = v^T h = \underbrace{v^T A_\theta}_{C_\theta^T} \phi(x)$

Considering $C_\theta = (v^T A_\theta)^T \in \mathbb{R}^{\frac{D(D+1)}{2}}$ ($1 \times K$, $K \times \frac{D(D+1)}{2}$), we have $\hat{y}(x; C_\theta) = C_\theta^T \phi(x)$.

$\therefore$ $\hat{y}$ is a linear transformation of C_θ.

This model is not linear, in terms of the original parameters, because the relation between that parameters θ and C_θ is not linear.

3.

3.3

We can obtain the parameter C_θ via some non-linear function $\psi: \mathbb{R}^{K\times D} \times \mathbb{R}^K \to \mathbb{R}^{\frac{D(D+1)}{2}}$, such that $C_\theta = \psi(W,v)$ since we demonstrated that C_θ is a non-linear function of the original network parameters $\theta = (W,v)$.

Considering $\psi(W,v) = v^T A_w = A_w^T v = C$, as we demonstrated before:

$$A_w = \underbrace{[w_i \odot w_i \cdots w_i \odot w_j]}_{D \qquad \frac{D(D-1)}{2}}$$

The first D columns can be linearly independent, which does not happen for the following columns.

As a result, we only need the first D rows to reach all the row space ($\mathbb{R}^{\frac{D(D+1)}{2}}$) for the system $A_w^T v = C$. A_w^T columns are linearly independent.

For $K \geq D$:

We have enough independent colums to encompass all of $\mathbb{R}^{\frac{D(D+1)}{2}}$

So for any $c \in \mathbb{R}^{\frac{D(D+1)}{2}}$, there is a pair (W,v) such that $c = \psi(W,v)$.

For $K < D$:

As A_w^T does not contain at least a set of D linear independent columns, we cannot cover all of $\mathbb{R}^{\frac{D(D+1)}{2}}$.

4

3.4.

$$X = \begin{bmatrix} \phi(x^{(1)}) \\ \vdots \\ \phi(x^{(N)}) \end{bmatrix} = \begin{bmatrix} x_1^{(1)2} \cdots x_D^{(1)2} \; x_1^{(1)}x_2^{(1)} \cdots x_1^{(1)}x_D^{(1)} \cdots x_{D-1}^{(1)}x_D^{(1)} \\ \vdots \\ x_1^{(N)2} \cdots x_D^{(N)2} \; x_1^{(N)}x_2^{(N)} \cdots x_1^{(N)}x_D^{(N)} \cdots x_{D-1}^{(N)}x_D^{(N)} \end{bmatrix}$$

$$\mathcal{L}(c_\theta; \mathcal{D}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}_n(x_n; c_\theta) - y_n)^2 =$$

$$= \frac{1}{2}\sum_{n=1}^{N}(y_n - c_\theta^T \phi(x))^2 = \frac{(y_n - X\hat{c}_\theta)(y - X\hat{c}_\theta)^T}{2}$$

$$\frac{\partial \mathcal{L}}{\partial c_\theta} = 0 \Leftrightarrow (-X)^T(y - X\hat{c}_\theta) + (y - X\hat{c}_\theta)^T(-X) = 0$$

$$\Leftrightarrow (-X)^T(y - X\hat{c}_\theta) + \left((-X)^T(y - X\hat{c}_\theta)\right)^T = 0$$

So $(-X)^T(y - X\hat{c}_\theta) = 0 \Leftrightarrow$

$\Leftrightarrow -X^Ty + X^TX\hat{c}_\theta = 0 \Leftrightarrow$

$\Leftrightarrow I\hat{c}_\theta = \hat{c}_\theta = (X^TX)^{-1}X^Ty$

Yes, we can find a closed form solution $\hat{c}_\theta$.

The loss function is a quadratic functions, so it has a global minima in $\hat{c}_\theta$.

Design matrix X does not apply any transformation on the input ($\phi(x)$) of the model, and it still linear for $\phi(x)$ when we minimize the loss function.

References

https://cs231n.github.io/linear-classify/#softmax
https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
https://wandb.ai/authors/ayusht/reports/Implementing-Dropout-in-PyTorch-With-Example--VmlldzoxNTgwOTE