Instituto Superior Técnico

Deep Learning

Homework 1

Group 28:
Luís Freire D'Andrade *(ist194179)*
Rita Duarte *(ist195531)*

December 23, 2022

Work Distribution

The questions 1.2.a), 1.2.b), 2.2 and 2.3 were assigned to Luís D'Andrade.
The questions 1.1.a), 1.1.b) and 2.1 were assigned to Rita Duarte.
Both the colleagues worked together to solve the 3.1, 3.2 and 3.4 questions.

References

- Theorical slides
- Jupyter notebooks from practical 2, 3 and 5. And also annotations taken from the practical classes
- Piazza forum of the Deep Learning subject
- Apontamentos Álgebra Linear IST

Question 1

1.1.a)

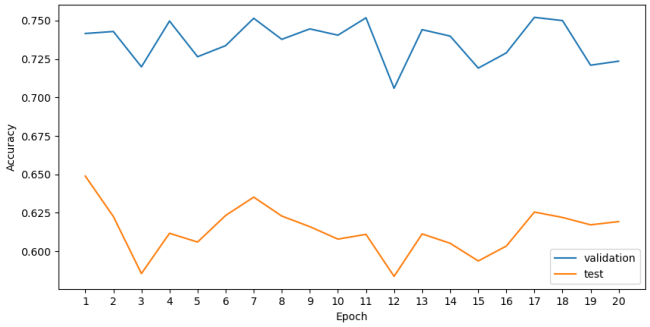

Figure 1: Plotting of the accuracy for each of the 20 trained epochs by the implemented perceptron

1.1.b)

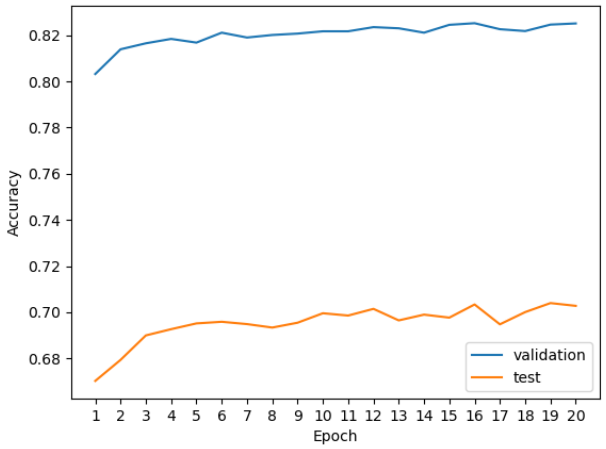

Figure 2: Plotting of the accuracy for each of the 20 trained epochs by the implemented logistic regression with learning rate of 0.001

1.2.a)

Unlike the the Simple Single Layer Perceptron (SLP) implemented above, a Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While in a SLP the neuron must have a linear activation function that imposes a threshold, neurons in a MLP can use any arbitrary (including non-linear) activation function.
This conveys that a MLP can learn non-linear target functions, i.e. learn a non-linear relationship between the inputs and outputs. If a neural network consists of only linear activation functions (like a SLP), it can only model a linear relationship between the inputs and outputs, which makes it not as expressive or useful in a lot of applications.

1.2.b)

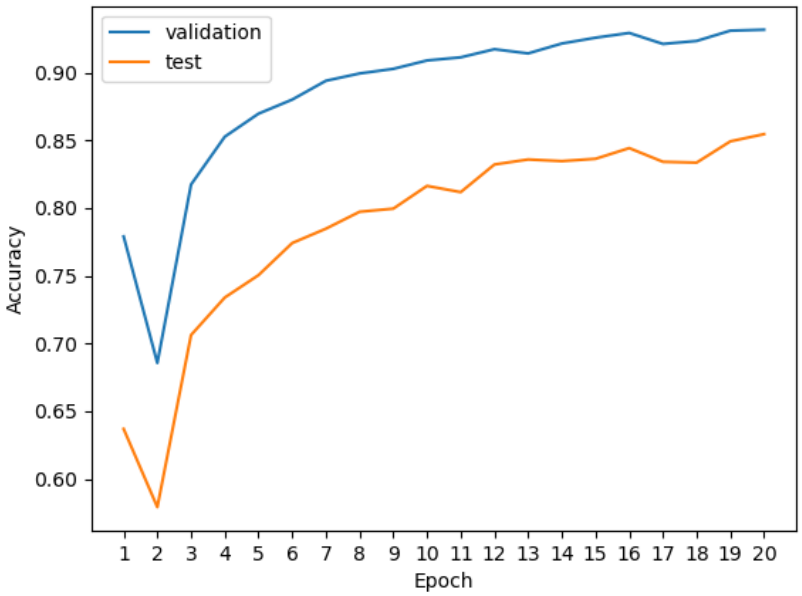

Figure 3: Plotting of the accuracy for each of the 20 trained epochs by a multi-layer perceptron with a single hidden layer

Number of epochs	20
Hidden Units	200
Hidden Layers	Activation function **R**eLU
Output layer	Multinomial logistic loss
Training algorithm	Stochastic gradient descent
Learning rate	0.001
Bias	Zero vectors
Weights	$w_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = 0.1\, \sigma^2 = 0.1^2$

Question 2

2.1.

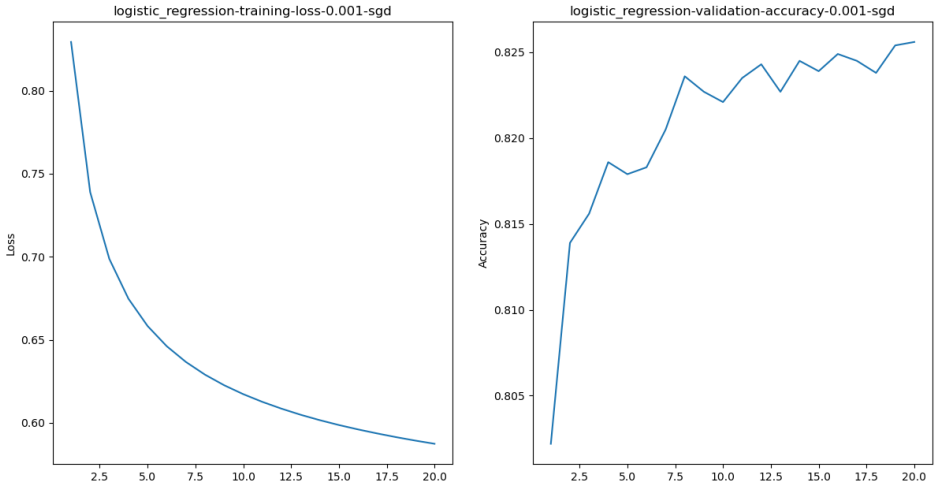

Figure 4: Plotting of the accuracy and training loss for each of the 20 trained epochs by a linear model with logistic regression
Final test accuracy: 0.7019

Number of epochs	20
Hidden Units	200
Batch size	1
Optimization algorithm	Stochastic gradient descent
Learning rate	0.001

2.2.

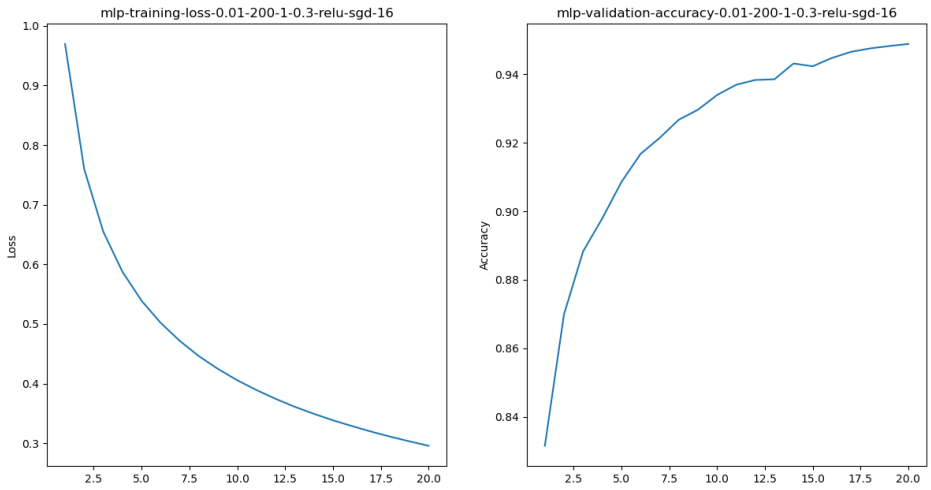

Figure 5: Plotting of the accuracy and training loss for each of the 20 trained epochs by feed-forward neural network using a dropout regularization
Final test accuracy:0.8818

Number of epochs	20
Number of Layers	1
Learning Rate	0.01
Hidden Size	200
Dropout	0.3
Batch Size	16
Activation	ReLU
Optimizer	SGD

2.3.

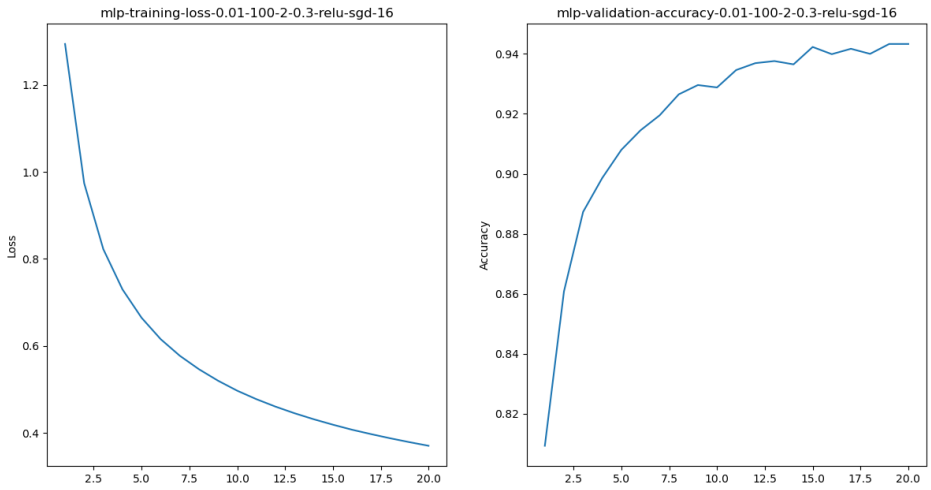

Figure 6: Plotting of the accuracy and training loss for each of the 20 trained epochs by feed-forward neural network using a dropout regularization
Final test accuracy: 0.8724

Number of epochs	20
Number of Layers	2
Learning Rate	0.01
Hidden Size	100
Dropout	0.3
Batch Size	16
Activation	ReLU
Optimizer	SGD

Question 3

3.1.

For h to be a linear transformation of $\Phi(x)$ we need to find a feature transformation $\Phi(x)$ and a matrix A_Θ such that $h = A_\Theta \Phi(x)$.
Since $h = g(Wx), x \in \mathbb{R}^D$ and the activation function is $g(z) = z^2$ then:

$$h = g(Wx) \Leftrightarrow h = (Wx)^2$$

Also $\Theta = (W, v) \in \mathbb{R}^{k \times D} \times \mathbb{R}^k$ hence:

$$W = \begin{bmatrix} W_{11} & ... & W_{1D} \\ . & ... & . \\ W_{k1} & ... & W_{kD} \end{bmatrix}$$

$$\text{and } x = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_D \end{bmatrix}$$

Now,

$$h = (Wx)^2 \Leftrightarrow h = \left(\begin{bmatrix} W_{11} & ... & W_{1D} \\ . & ... & . \\ W_{k1} & ... & W_{kD} \end{bmatrix} \times \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_D \end{bmatrix} \right)^2 \Leftrightarrow \left(\begin{bmatrix} \sum_{i=1}^{D} W_{1i} \times x_i \\ . \\ . \\ . \\ \sum_{i=1}^{D} W_{ki} \times x_i \end{bmatrix} \right)^2$$

From the Square of the Sum:

$$(\sum_{i=1}^{N} a_i)^2 = \sum_{i=1}^{N} {a_i}^2 + 2 \times \sum_{j=1}^{N} \sum_{i=1}^{j-1} a_i a_j$$

We can expand the last matrix into,

$$\Leftrightarrow \begin{bmatrix} \sum_{i=1}^{D} (W_{1i}x_i)^2 + 2 \times \sum_{j=1}^{D} \sum_{i=1}^{j-1} (W_{1i}x_i)(W_{1j}x_j) \\ . \\ . \\ . \\ \sum_{i=1}^{D} (W_{ki}x_i)^2 + 2 \times \sum_{j=1}^{D} \sum_{i=1}^{j-1} (W_{ki}x_i)(W_{kj}x_j) \end{bmatrix}$$

In the description of the exercise 3.1), $\Phi : \mathbb{R}^D \to \mathbb{R}^{\frac{D(D+1)}{2}}$ and $A \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$.
As stated before $h = A_\Theta \Phi(x)$ and the product of the matrix A_Θ with size $k \times \frac{D(D+1)}{2}$ by the vector $\Phi(x)$ with size $\frac{D(D+1)}{2} \times 1$ will result in a vector of size $k \times 1$:

$$h = A_\Theta \Phi(x) = \begin{bmatrix} a_{11} & ... & a_{1\frac{D(D+1)}{2}} \\ . & . & . \\ a_{k1} & ... & a_{k\frac{D(D+1)}{2}} \end{bmatrix} \begin{bmatrix} b_{11} \\ . \\ b_{k1} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{\frac{D(D+1)}{2}} a_{1i} b_{1i} \\ . \\ \sum_{i=1}^{\frac{D(D+1)}{2}} a_{ki} b_{ki} \end{bmatrix}$$

By only looking at line k of the matrix h,

$\sum_{i=1}^{D} (W_{ki}x_i)^2 + 2 \times \sum_{j=1}^{D} \sum_{i=1}^{j-1} (W_{ki}x_i)(W_{kj}x_j)$ needs to be written as $\sum_{i=1}^{\frac{D(D+1)}{2}} a_{ki} b_{ki}$

$$\sum_{i=1}^{D}(W_{ki}x_i)^2 + 2 \times \sum_{j=1}^{D}\sum_{i=1}^{j-1}(W_{ki}x_i)(W_{kj}x_j) \Leftrightarrow$$

$$\Leftrightarrow \sum_{i=1}^{D} W_{ki}^2 x_i^2 + 2 \times \sum_{j=1}^{D}\sum_{i=1}^{j-1}(W_{ki}x_i)(W_{kj}x_j) \Leftrightarrow$$

$$\Leftrightarrow (W_{k1}^2x_1^2 + W_{k2}^2x_2^2 + ... + W_{kD}^2x_D^2) + (2W_{k1}W_{k2}x_1x_2 + 2W_{k1}W_{k3}x_1x_3 + ... + 2W_{k(j-1)}W_{kD}x_{(j-1)}x_D)$$

For each $n \in [1, D]$, by putting

x_n in evidence

W_{kn} in evidence

$$x_n^2 \times (x_nx_{n+1} \times x_nx_{n+2} \times x_nx_{n+3} \times ... \times x_nx_D)$$
and
$$W_{in}^2 \times 2W_{in}W_{i(n+1)} \times 2W_{in}W_{i(n+2)} \times 2W_{in}W_{i(n+3)} \times ... \times 2W_{in}W_{iD}$$

By analysing the behaviour of h in line $= k$ we can detect a pattern and isolate the x from the W in order to write h as the product of matrices A_Θ and $\Phi(x)$,

$$h = \left[A_\Theta\right]\ \left[\Phi(x)\ \right] \Leftrightarrow$$

$$\Leftrightarrow \text{h}= \begin{bmatrix} W_{11}^2 & ... & W_{1D}^2 & 2W_{11}W12 & ... & 2W_{1(D-1)}W_{1D} \\ ... & ... & ... & ... & ... & ... \\ W_{k1}^2 & ... & W_{kD}^2 & 2W_{k1}Wk2 & ... & 2W_{k(D-1)}W_{kD} \end{bmatrix} \begin{bmatrix} x_1^2 \\ . \\ . \\ . \\ x_D^2 \\ x_1x_2 \\ . \\ . \\ . \\ x_{D-1}x_D \end{bmatrix}$$

Hence the mapping Φ and the matrix A_Θ are given by:

$$\Phi(x) = \begin{bmatrix} x_1^2 x_1x_2...x_1x_D & x_2^2x_2x_3...x_2x_D & ..... & x_D^2 \end{bmatrix}^T$$

$$A_\Theta = \begin{bmatrix} W_{i1}^2 2W_{i1}W_{i2}...2W_{i1}W_{iD} & W_{i2}^2 2W_{i2}W_{i3}...2W_{i2}W_{iD} & ... & W_{iD}^2 \end{bmatrix}$$

3.2.

In the description of exercise 3.2) there is information relevant for the solving of the problem such as:

1. $\hat{y} = v^T h \in \mathbb{R}$ hence $\hat{y} \in \mathbb{R}$
2. $\Theta = (W, v) \in \mathbb{R}^{K \times D} \times \mathbb{R}^K$ hence $v \in \mathbb{R}^k$.
3. $h \in \mathbb{R}^K$
4. $A_\Theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$
5. $\Phi(x) \in \mathbb{R}^{\frac{D(D+1)}{2}}$

Based on the previous claim $h = A_\Theta \Phi(x)$,

$$\hat{y} = v^T h = v^T A_\Theta \Phi(x)$$

The hypothesis is we can write $\hat{y}(x; c_\Theta) = c_\Theta^T \Phi(x)$,

$$\hat{y} = v^T A_\Theta \Phi(x) = c_\Theta^T \Phi(x),$$

where v^T is a vector size $1 \times k$, A_Θ is a matrix size $k \times \frac{D(D+1)}{2}$:
c_Θ^T is size $1 \times \frac{D(D+1)}{2}$.

$$c_\Theta^T = v^T A_\Theta \Leftrightarrow c_\Theta^T = \begin{bmatrix} v_1 & ... & v_k \end{bmatrix} \begin{bmatrix} - & A_{\Theta 1} & - \\ - & ... & - \\ - & A_{\Theta k} & - \end{bmatrix} \Leftrightarrow$$

Having into consideration the following transpose properties for matrices:

$(A^T)^T = A$, let A be a matrix.

$(A \times B)^T = B^T \times A^T$, let A and B be matrices.

Then:

$$\Leftrightarrow (c_\Theta^T)^T = \left(\begin{bmatrix} v_1 & ... & v_k \end{bmatrix} \begin{bmatrix} - & A_{\Theta 1} & - \\ - & ... & - \\ - & A_{\Theta k} & - \end{bmatrix} \right)^T \Leftrightarrow$$

$$\Leftrightarrow c_\Theta = \begin{bmatrix} v_1 & ... & v_k \end{bmatrix}^T \begin{bmatrix} - & A_{\Theta 1} & - \\ - & ... & - \\ - & A_{\Theta k} & - \end{bmatrix}^T \Leftrightarrow$$

$$\Leftrightarrow c_\Theta = \begin{bmatrix} | & ... & | \\ A_{\Theta 1} & ... & A_{\Theta k} \\ | & ... & | \end{bmatrix} \begin{bmatrix} v_1 \\ ... \\ v_k \end{bmatrix},$$

c_Θ corresponds to the product of a matrix A_Θ^T of size $\frac{D(D+1}{2} \times k$ and a vector v of size $k \times 1$.
Therefore c_Θ is size $\frac{D(D+1)}{2} \times 1$ and we can write the matrix as:

$$c_{\Theta i} = \sum_{j=1}^{k} A_{\Theta ij} v_j, \; i \text{ of size } \frac{D(D+1)}{2}$$

Therefore it is proven that,

$$\hat{y} = (v^T A_\Theta)\Phi(x) \Leftrightarrow \hat{y} = c_\Theta^T \Phi(x), \text{ for some } c_\Theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$$

Linear regression, as confirmed, can be used to learn this model.
Nevertheless, this is still a non-linear model in terms of the original parameters Θ because its activation function is non-linear: $g(z) = z^2$.

3.4.

Since $\hat{y} = {c_\Theta}^T \Phi(x)$ is linear, it's possible to minimize the squared loss, $L(c_\Theta; \mathcal{D})$, and find a closed form solution for $\hat{c_\Theta}$. To do so we need to solve,

$$\hat{c_\Theta} = arg\,min_{c_\Theta} \tfrac{1}{2} \sum_{n=1}^{N} \left(\hat{y}(x_n; c_\Theta) - y_n\right)^2$$

The factor $\frac{1}{2}$ is irrelevant and the subtraction can be commuted, as it's squared,

$$\hat{c_\Theta} = arg\,min_{c_\Theta} \sum_{n=1}^{N} \left(y_n - \hat{y}(x_n; c_\Theta)\right)^2 = arg\,min_{c_\Theta} \sum_{n=1}^{N} \left(y_n - {c_\Theta}^T \Phi(x_n)\right)^2$$

Recalling the Euclidean/Frobenius norm formula,

$$\sum_{n=1}^{N} \left(y_n - {c_\Theta}^T \Phi(x_n)\right)^2 = \|y - X c_\Theta\|^2,$$

$$\text{with } X = \begin{bmatrix} \Phi(x_1)^T \\ ... \\ \Phi(x_N)^T \end{bmatrix} \in \mathbb{R}^{N \times \frac{D(D+1)}{2}},\ y = \begin{bmatrix} y_1 \\ ... \\ y_N \end{bmatrix} \in \mathbb{R}^N$$

Writing the optimality condition, when the gradient equals zero,

$$\begin{aligned} 0 &= \nabla_{c_\Theta} \|y - X c_\Theta\|^2 \\ &= \nabla_{c_\Theta} ({c_\Theta}^T X^T X c_\Theta - 2 {c_\Theta}^T X^T y + \|y\|^2) \\ &= 2X^T X c_\Theta - 2X^T y \end{aligned}$$

Which results in,

$$\hat{c_\Theta} = (X^T X)^{-1} X^T y$$

Feedforward neural network training is a special case of function minimisation, and many of its optimization algorithms can easily get stuck in non-global minima, which makes global minimization usually intractable. However, in our case, we're dealing with a squared loss (a convex optimization problem) and a polynomial activation function. Local minimizers of a convex optimization problem are also global minimizers, which makes convex formulations a preferable special case.