Olá!

Catarina Sousa | 93695
Gonçalo Falcão | 102219
Miguel Ferreira | 90899

Question 1

Exercise 1

$$\begin{aligned}
&\frac{\delta\sigma}{\delta z} \\
&= \frac{\delta}{\delta z}\frac{1}{(1+e^{-z})} \\
&= \frac{\delta}{\delta z}(1+e^{-z})^{-1} \\
&= (-1)*(1+e^{-z})^{-2}*\frac{\delta}{\delta z}(1+e^{-z}) \\
&= \frac{1}{(1+e^{-z})^2}*(e^{-z}) \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} \\
&= \sigma(z) - \sigma^2(z) \\
&= \sigma(z)(1-\sigma(z))
\end{aligned}$$

, as we want to show.

Exercise 2

$$\begin{aligned}
L^{'}(z; y=+1) &= \frac{d}{dz} - log[\sigma(z)] \\
&= (-1)*\frac{\sigma^{'}(z)}{\sigma(z)} \\
&= (-1)*\frac{\sigma(z)*(1-\sigma(z))}{\sigma(z)} \\
&= (-1)*(1-\sigma(z)) \\
&= \sigma(z) - 1
\end{aligned}$$

$$\begin{aligned}
L^{''}(z; y=+1) &= \frac{d}{dz}(\sigma(z)-1) \\
&= \sigma(z)*(1-\sigma(z)) > 0, \forall z.
\end{aligned}$$

Now, let's derivate the binary logistic loss with respect to z to prove that it is convex as a function of z:

$$\begin{aligned}
&L^{'}(z; y) = \\
&\frac{d}{dz}[-\frac{1+y}{2}log(\sigma(z)) - \frac{1-y}{2}log(1-\sigma(z))] \\
&\frac{1+y}{2}(\sigma(z)-1) + \frac{1-y}{2}*\frac{\sigma(z)(1-\sigma(z))}{1-\sigma(z)} \\
&\frac{1+y}{2}(\sigma(z)-1) + \frac{1-y}{2}\sigma(z)
\end{aligned}$$

$$\begin{aligned}
&L^{''}(z; y) = \\
&= \frac{1+y}{2}\sigma(z)(1-\sigma(z)) + \frac{1-y}{2}\sigma(z)(1-\sigma(z)) \\
&= \sigma(z)(1-\sigma(z)) * \frac{1+y+1-y}{2} \\
&= \sigma(z)(1-\sigma(z)) \geq 0, \forall z.
\end{aligned}$$

Since the second order derivative of the loss with respect to z is $\geq 0, \forall z$, then the loss is convex as a function of z, as we want to show.

Exercise 3

So, what we need to calculate, to compute the Jacobian matrix is $\frac{\partial[softmax(z)]_i}{\partial z_j}$.

If $j = i$:

$$\begin{aligned}
&\frac{\partial[softmax(z)]_i}{\partial z_j} = \frac{\partial[softmax(z)]_j}{\partial z_j} = \\
&\frac{\partial}{\partial z_j}\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} = \frac{e^{z_j}(\sum_{k=1}^K e^{z_k}) - e^{z_j}e^{z_j}}{(\sum_{k=1}^K e^{z_k})^2} = \\
&\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}\frac{(\sum_{k=1}^K e^{z_k}) - e^{z_j}}{\sum_{k=1}^K e^{z_k}} = \\
&s_j(1-s_j)
\end{aligned}$$

If $j \neq i$:

$$\begin{aligned}
&\frac{\partial[softmax(z)]_i}{\partial z_j} \\
&= \frac{\partial}{\partial z_j}\frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}} = \frac{0 - e^{z_i}e^{z_j}}{(\sum_{k=1}^K e^{z_k})^2} = \\
&- \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} = -s_i * s_j
\end{aligned}$$

Where $s_k = softmax(z)_k$.

$$\begin{bmatrix}
s_0(1-s_0) & -s_1 s_0 & \dots & -s_j s_0 \\
-s_0 s_1 & \ddots & -s_j s_i & \vdots \\
\vdots & -s_j s_i & s_j(1-s_j) & \vdots \\
-s_0 s_i & \vdots & \ddots & \ddots
\end{bmatrix}_{(K,K)}$$

Exercise 4

In the last exercise we calculated the Jacobian matrix for the softmax transformation on point z. Computing the gradient for the logistic loss, $L(z; y = a) = -log[softmax(z)]_a$ is straightforward.

Remember that:

$$-\frac{dlog(u)}{du} = -\frac{u'}{u}$$

This way, we now have that:

$$\text{With } L : \mathbb{R}^N \to \mathbb{R}, \nabla L(z; y = a) = \begin{bmatrix} \frac{\partial L_{(y=a)}}{\partial z_0} \\ \dots \\ \frac{\partial L_{(y=a)}}{\partial z_i} \\ \dots \\ \frac{\partial L_{(y=a)}}{\partial z_N} \end{bmatrix}_{(N,1)}, \text{where } \frac{\partial L_{(y=a)}}{\partial z_i} = softmax_i(z(x)) - 1_{(y=i)}$$

The basis to construct the Hessian matrix is this derivative: $\frac{\partial^2 L}{\partial z_i \partial z_j}$. And we also know that $\nabla^2 L(z; y = a)$ is a matrix of dimensions (N, N), where $y \in \{1, ..., N\}$ and $z \in \mathbb{R}^N$. This way, we need to calculate the *basis* derivative for the following *cases* (covering the entire Hessian matrix positions):

1) $i \neq j = a$; **2)** $j \neq i = a$; **3)** $i \neq j \neq a$; **4)** $i = j = a$; **5)** $i = j \neq a$;

For space and display motives, we are going to replace $softmax_i(z(x)) \to s_i$.

- For **1** we have $\to \frac{\partial}{\partial z_i \partial z_{j=a}} L(z; y = a) = \frac{\partial}{\partial z_a} - log(s_a) = \frac{\partial}{\partial z_i}(s_a - 1) = -s_i s_a \leq 0, \forall z$.
- For **2** we have almost the same as in **1** $\to -s_j s_a \leq 0, \forall z$.
- For **3** we have $\to \frac{\partial}{\partial z_i \partial z_j} - log(s_a) = \frac{\partial}{\partial z_i} s_j = -s_i s_j \leq 0, \forall z$.
- For **4** we have $\to \frac{\partial}{\partial z_i \partial z_j} - log(s_a) = \frac{\partial}{\partial z_a^2} - log(s_a) = \frac{\partial}{\partial z_a}(s_a - 1) = s_a(1 - s_a) \geq 0 \ ,\forall z$.
- For **5** we have $\to \frac{\partial}{\partial z_i^2} - log(s_a) = \frac{\partial}{\partial z_i} s_i = s_i(1 - s_i), \geq 0 \ \forall z$.

In fact, cases 1 until 5 can be grouped into two $\to H_{ij} = -s_j s_i$ if $j \neq i$ and $H_{ij} = s_j(1 - s_j)$ if $j = i$, where H is the hessian matrix.

(Note that since $s_i \in [0, 1]$ it is trivial to show that $-s_i s_j \leq 0, \forall (i, j)$ and that $s_i(1 - s_i) \geq 0, \forall i$)

With these results there's something we already know: we have a positive (non negative) diagonal, since the diagonal elements are always $\geq$0 (given by cases **4** and **5**). We also know that a symmetric matrix is the one where $a_{ij} = a_{ji}$ [2]. Such characteristic can be observed by using cases **1**, **2** and **3**.

With these two characteristic, we have a very curious matrix: one with a diagonal $\geq$0, and all other elements $\leq$0. Furthermore, we have that our matrix is **1)** a square one, and also that **2)** $|H_{ii}| \geq \sum_{j \neq i} |H_{ij}|$ ∀i. A matrix that holds these properties is a **diagonally dominant matrix** [3], this way there is a definition [3] we can use to prove that the Hessian is positive semidefinite and, therefore, that the loss $L(z; y = a)$ is convex with respect to z.

But, a question arises. How can we say that **2)** holds? What we want to prove is that for each row of the matrix, the absolute value of the diagonal element is greater than or equal to the sum over the absolute value of all elements of the row excluding the diagonal one:

For a row i, we want: $|s_i(1-s_i)| \geq \sum_{j \neq i} |-s_i s_j| \Leftrightarrow$

(since $s_i \geq 0$) $\Leftrightarrow s_i(1-s_i) \geq \sum_{j \neq i} s_i s_j \Leftrightarrow$

$\Leftrightarrow 1 - s_i \geq \sum_{j \neq i} s_j \Leftrightarrow$

$\Leftrightarrow 1 \geq s_i + \sum_{j \neq i} s_j,$ and, indeed, we know that $\sum_k s_k = 1,$ as such,

$1 \geq s_i + \sum_{j \neq i} s_j,$ since $\sum_{k=\{j \cup i\}} s_k = 1.$

This way condition **2)** holds, and, by definition, since a **symmetric diagonally dominant real** matrix with **nonegative diagonal** entries is **positive semidefinite**, we've proven that the loss function is convex with respect to z.

Exercise 5

On the last exercise we've proven that the **multinomial logistic loss** is convex with respect to **z**. As such, if we now have that $z = W\phi(x) + b$, we know - using the **hint** - that the composition of an affine map (in this case $z = W\phi(x) + b$), with a convex function (**the multinomial logistic loss, as we have shown**):

$((L(z; y = j) \circ (W\phi(x) + b))(x),$ **is also convex**.

What this says is that, since $z = W\phi(x) + b$ is a linear transformation of both $W\phi(x)$ and b, then the function is convex to both parameters on their own, **just as we want to show**.

In case z is not a linear function, then the multinomial logistic loss may not be convex with respect to its parameters. Since we prove the convex property to z, we can only generalize it to linear transformations of itself (as we have just did). If the transformation is not linear (or a composition of an affine map), then we need to test each case on its own to see if it may be convex (in the case we can even compute its properties).

A pragmatic explanation is that if this was true, then the gold objective of Artificial Intelligence (our Master-Algorithm) would be within our reach, since we could compute any problem where we have a defined loss, and get a viable model. Obviously, we aren't there yet.

Question 2

Exercise 1

We have that our function $L(z; y) = \frac{1}{2}||z - y||_2^2 = \frac{1}{2}\sum_i (z_i - y_i)^2 = \frac{1}{2}[(z_0 - y_0)^2 + (z_1 - y_1)^2 + \cdots + (z_i - y_i)^2]$.

This way, and since we want to show that L is convex with respect to (W, b), let's first show that it is convex with respect to z, as we can after use the hint from the previous question to show it is also with respect to (W, b).

We know that the hessian is calculated with a second-order derivative with the following structure: $\frac{\partial L}{\partial z_i \partial z_j}$. Therefore:

- If $i \neq j \rightarrow \frac{\partial L}{\partial z_i \partial z_j} = \frac{\partial}{\partial z_i \partial z_j}\frac{1}{2}[(z_i - y_i)^2 + (z_j - y_j)^2] = \frac{\partial}{\partial z_i}(z_j - y_j) = 0$.
- If $i = j \rightarrow \frac{\partial L}{\partial z_i^2} = \frac{\partial}{\partial z_i^2}\frac{1}{2}[(z_i - y_i)^2] = \frac{\partial}{\partial z_i}(z_i - y_i) = 1$.

This means that the hessian of L, $\nabla^2 L = I$, with I being the Identity matrix. And, we know, by definition, that the Identity matrix is positive-definite.

With this, if we now apply our knowledge about the question 1, exercise 5, the composition of the affine map $g : W^\intercal \phi(x) + b$, with the convex function L, is also convex with respect to (W, b)

Exercise 2

Exercise 2a

We know that we are in an over-fitting situation when we are losing model generalization. What we see here is that until around epoch 70, the test loss decreases, as well as the training loss. Then, roughly after epoch 70, while the training loss continues to decrease, the test loss doesn't, it even increases! What this shows us, is that the model is not able anymore to correctly generalize the train data to the test data.

If the test data distributions aren't similar to the training data ones, this may be expected, however, if we assume they are - and we assume it -, this clearly shows that the model is over-fitting to the training data at the expense of losing generalization and performing worse in the test data.

Obtained loss for train set: 0.111 | for test set: 0.238

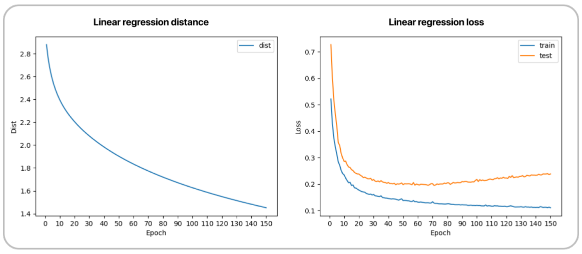

Figure 1: Left: the distance for linear regression between the analytical and non-analytical methods. Right: the loss for the linear regression for the train and test sets.

Exercise 2b

Obtained loss for train set: 0.099 | for test set: 0.165

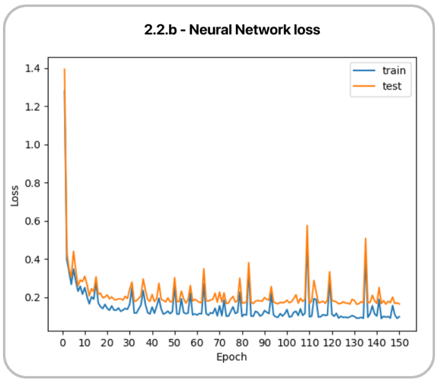

Figure 2: Loss for train and test sets for neural network.

We also observe that even though one might intuitively think that with a neural network the loss would be lower (even greatly lower) than with linear regression, what we observe is quite tricky! With the neural network, the train set loss is 12% better (lower) than with the linear regression and 45% better for the test set loss.

In absolute terms, the difference between test and train set losses for both models, is very small, however, relatively speaking, the difference is quite striking.
What we need to evaluate is if the tradeoff between complexity and performance we gain and lose (respectively) is worth against linear regression.

In the end, since the difference between the values is so small, we would argue that the data is highly linearly separable, and that's why the linear regression works so well. In this specific case, where the two models perform almost as well, the linear regression, since it is simpler, probably would be the best fit for real-world deployment circumstances.

Question 3

Exercise 1

Exercise 1a

Obtained accuracy for validation set: 0.7126 | for test set: 0.7013

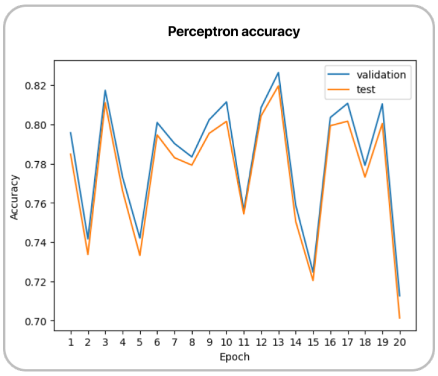

Figure 3: Accuracy for train and test sets for perceptron.

Exercise 1b

Obtained accuracy for validation set: 0.8503 | for test set: 0.8403

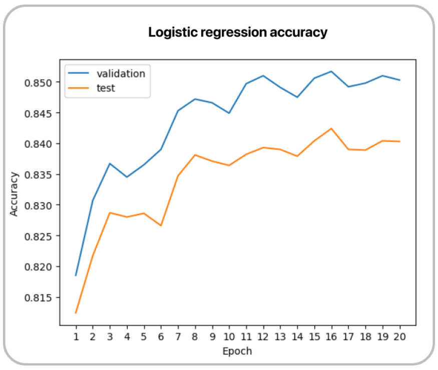

Figure 4: Accuracy for train and test sets for logistic regression.

Exercise 2

Exercise 2a

Intuitively speaking, just the fact that we have multi-layer perceptrons (meaning we have multiple parameters and 'knobs' to manipulate) gives us an explosion of combinatory possibilities and granularity to predict whatever we want. As such, we can gain much more predictive power by combining multiple perceptrons.

With one and one only perceptron we have a linear relationship between inputs and outputs, this means that if we want to solve more complex problems, where there are non-linear relationships between inputs and outputs, we need more complex models, something solved by the multi-layer perceptrons.

Moreover, non-linear activation functions imply that the model's output can't be reproduced from a linear combination of the inputs. This means that a multilayer perceptron with linear activation functions in all neurons, no matter how many layers it has, can be reduced to a two-layer input-output model [1] since adding those layers would just give another linear function.

Ultimately, most of the real-world data has non-linear decision boundaries [4], and non-linearity is needed in activation functions for the neural networks to be able to produce exactly these non-linear decision boundaries via non-linear combinations of weights and inputs.

Exercise 2b

Obtained accuracy for validation set: 0.8761 | for test set: 0.8699

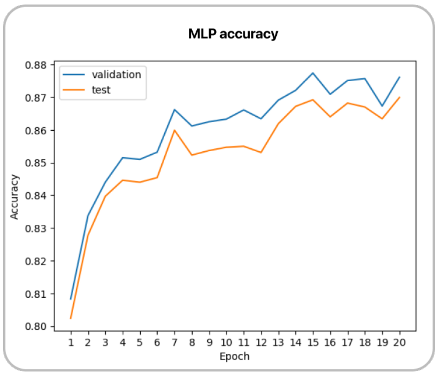

Figure 5: Accuracy for train and test sets for multi layer perceptron.

Question 4

Exercise 1

Regarding the best configuration (the one that performed the best on the validation set), we found it using as learning rate the value 0.001. For it, the training loss was 0.4398, and validation accuracy of 0.8513. For the final test accuracy the value obtained is 0.8402.

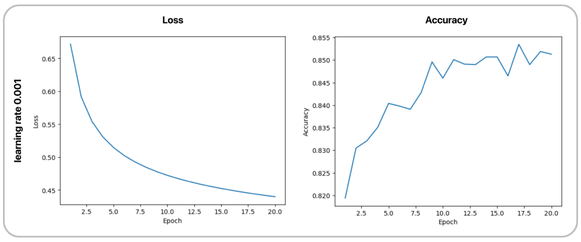

Figure 6: Plot for the best configuration, with 0.001 as the learning rate.

Exercise 2

Regarding the best configuration (the one that performed the best on the validation set), we found it also using as learning rate the value 0.001. For it, the training loss was 0.3696, and validation accuracy of 0.8871. For the final test accuracy the value obtained is 0.8774.

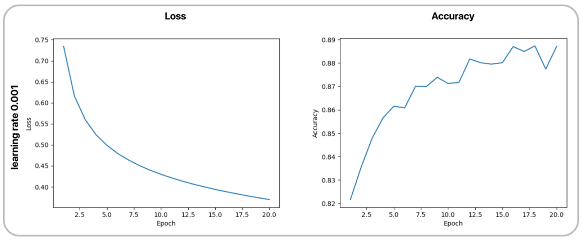

Figure 7: Plot for the best configuration, with 0.001 as the learning rate.

Exercise 3

The difference between 2 and 3 layers is more subtle! With **2** layers we got a training loss of: 0.4028, a validation accuracy of: 0.8677 and a final test accuracy of 0.8613. With **3** layers we got a training loss of: 0.4277, a validation accuracy of: 0.8688 and a final test accuracy of 0.8608.

With this, we choose the best model as the one with the highest accuracy on the validation set! As such, the best model is the one with 3 layers, even though the lowest training loss and the best final test accuracy were both obtained with 2 layers.

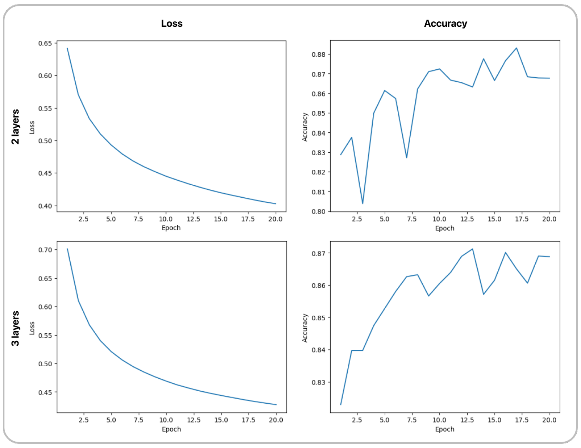

Figure 8: Plot for 2 and 3 layers.

References

[1] Multilayer perceptron. https://en.wikipedia.org/wiki/Multilayer_perceptron.

[2] Symmetric matrix. https://en.wikipedia.org/wiki/Symmetric_matrix.

[3] Diagonally dominant matrix. https://mathworld.wolfram.com/DiagonallyDominantMatrix.html.

[4] Why must a nonlinear activation function be used in a backpropagation neural network? https://stackoverflow.com/questions/9782071/.