TÉCNICO LISBOA

Deep Learning

Homework 1

Authors:	**Student number:**
Vito Rozman	108734
Tomer Klein	108547

Group 91

1st Semester 2023/2024

Contents

1 Question

1.1 Solution Q1.1

We trained the perceptron for 20 epochs, achieving a final accuracy of 0.3422. The training and validation accuracies are illustrated in Figure 1.

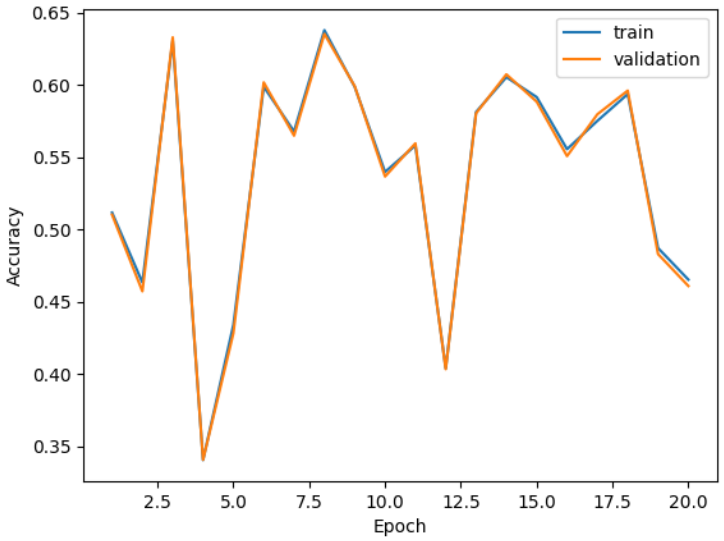


Figure 1: *Train and validation accuracies*

Logistic regression was employed without regularization, utilizing stochastic gradient descent as the training algorithm. The experiment involved assessing the final test accuracies for two distinct learning rates: $\eta = 0.01$, yielding a final test accuracy of 0.5784, and $\eta = 0.001$, resulting in a final test accuracy of 0.5936. In Figure 2 and Figure 3, a comparison of training and validation accuracies is presented. It can be inferred that employing $\eta = 0.001$ results in a more stable training process.

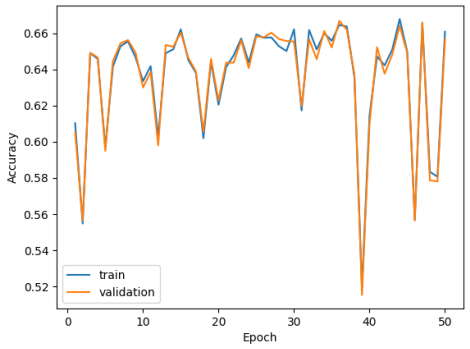


Figure 2: *Train and validation accuracies ($\eta = 0.01$)*

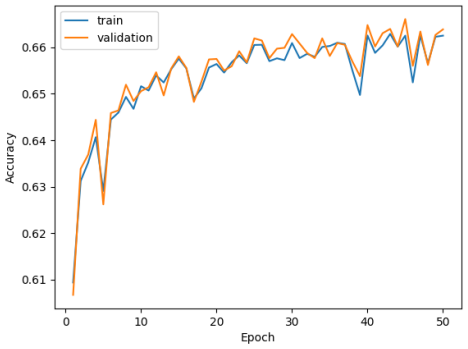


Figure 3: *Train and validation accuracies ($\eta = 0.001$)*

1.2 Solution Q1.2

a) Solution Q1.2 a

The claim: "A logistic regression model using pixel values as features is not as expressive as a multi-layer perceptron using relu activations. However, training a logistic regression model is easier because it is a convex optimization problem." is generally true.

Logistic regression is less expressive but easier to train, while an MLP with ReLU activations is more expressive but can be more challenging to train due to non-convex optimization. Logistic regression is simpler to train, its simplicity comes at the cost of reduced capacity to model complex relationships in the data. It might be suitable for linearly separable problems or situations where a simpler model suffices.

MLPs with ReLU activations are more powerful and can learn complex representations, but they may require more data and careful tuning of hyperparameters to avoid overfitting and convergence issues. The choice between the two depends on the complexity of the problem, the amount of available data, and computational resources.

b) Solution Q1.2 b

We implemented a multi-layer perceptron (MLP) with a single hidden layer, incorporating 200 hidden units, a rectified linear unit (ReLU) activation function for the hidden layers, and a multinomial logistic loss. The model was trained for 20 epochs using stochastic gradient descent with a learning rate of 0.001. In Figure 4 and Figure 5, you can see the training loss and accuracy, which contributed to achieving the final test accuracy of 0.7637.

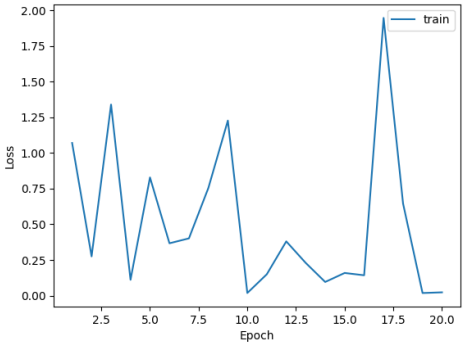


Figure 4: *Training loss MLP*

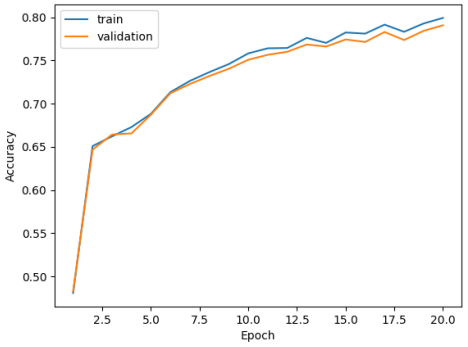


Figure 5: *Training accuracy MLP*

2 Question

2.1 Solution Q2.1

We implemented a linear model with logistic regression, employing stochastic gradient descent with a batch size of 16. The model was trained for 20 epochs, and we fine-tuned the learning rate on the validation data, exploring the values $\eta \in \{0.001, 0.01, 0.1\}$. The best learning rate $\eta = 0.1$, was selected based on test accuracy. The results are as follows: 0.4802 ($\eta = 0.1$), 0.4745 ($\eta = 0.01$) and

0.4726 ($\eta = 0.001$). Figures 6 and 7 display the training loss and validation accuracy as functions of epochs.

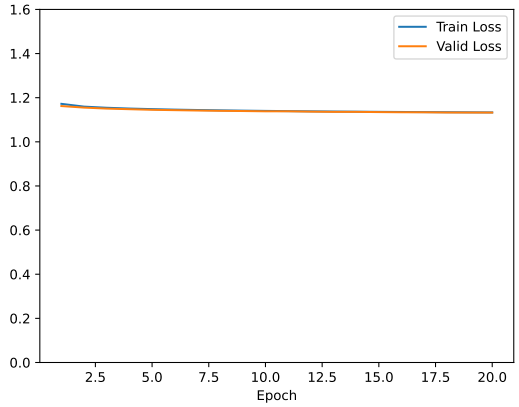


Figure 6: *Training loss ($\eta = 0.1$)*

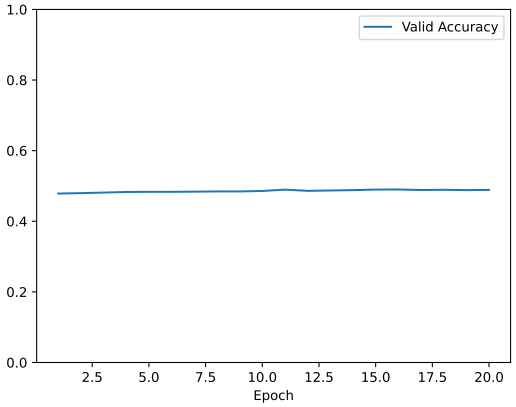


Figure 7: *Training accuracy ($\eta = 0.1$)*

2.2 Solution Q2.2

a) Solution Q2.2 a

We implemented a feed-forward neural network with dropout regularization, using default parameters specified in the exercise. Our objective was to determine the optimal batch size between 16 and 1024. The training loss and accuracy are presented in Figures 8, 9, 10 and 11. The decision was based on both time considerations and final test accuracy. For a batch size of 16, the model required 67.33 seconds and achieved a test accuracy of 0.7618. Conversely, with a batch size of 1024, the training time reduced to 16.108 seconds, but the test accuracy slightly decreased to 0.7183. After evaluation, we opted for a batch size of 16 as it yielded better overall results.

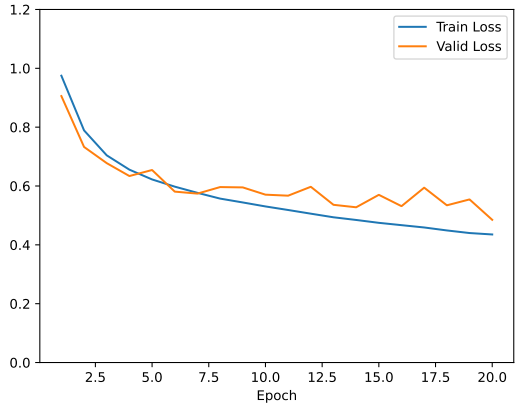


Figure 8: *Training loss (Batch size = 16)*

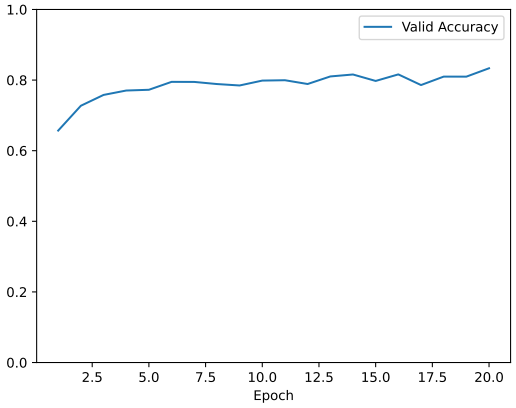


Figure 9: *Training accuracy (Batch size = 16)*

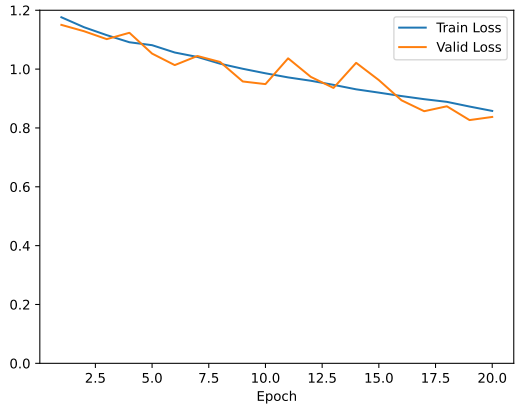


Figure 10: *Training loss (Batch size* = 1024)

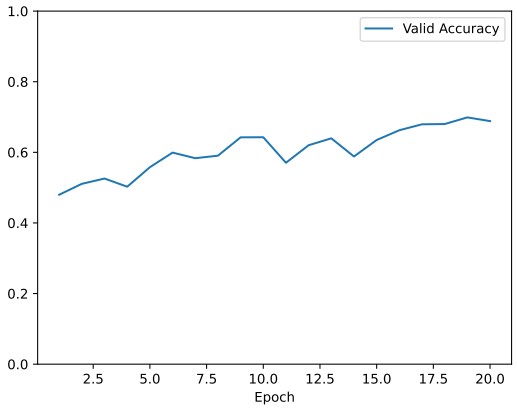


Figure 11: *Training accuracy (Batch size* = 1024*)*

b) Solution Q2.2 b

To address the challenge of tuning the learning rate, we experimented with values of 1, 0.1, 0.01, and 0.001 while keeping the remaining hyperparameters at their default values. The train and validation losses were plotted for both the best and worst configurations in terms of validation accuracy (see Figures 12, 13, 14, and 15). It is evident that using $\eta = 0.01$ yielded the best results, as it avoided excessively large or small steps during weight updates. The choice of this parameter resulted in a smooth increase in accuracy and a decrease in loss during training, ultimately leading to a test accuracy of 0.7637, where whores performance is 0.4726.

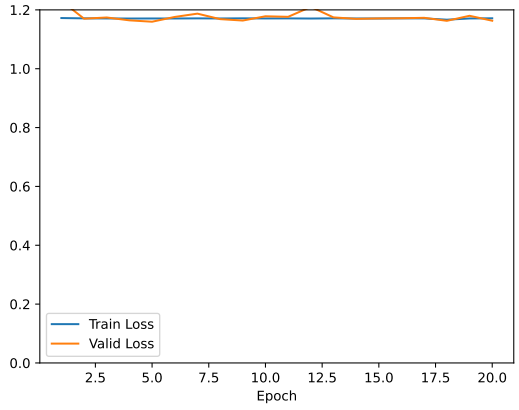


Figure 12: *Training loss* ($\eta = 1$)

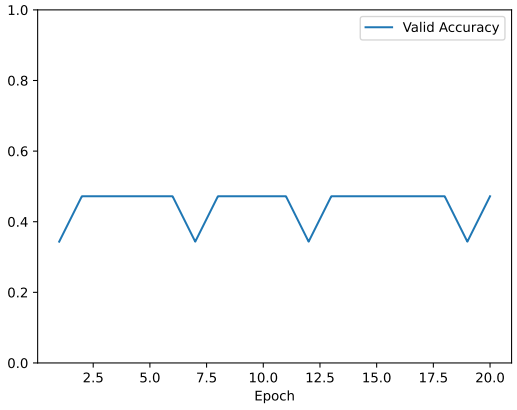


Figure 13: *Training accuracy* ($\eta = 1$)

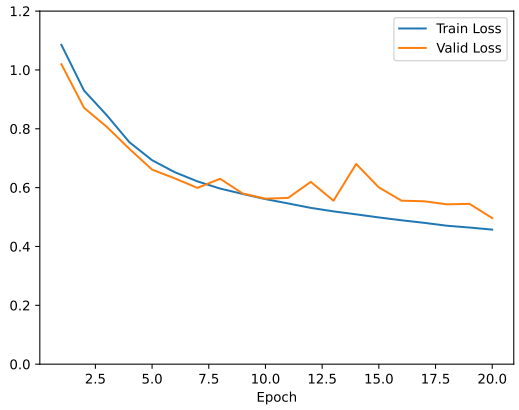


Figure 14: *Training loss ($\eta = 0.01$)*

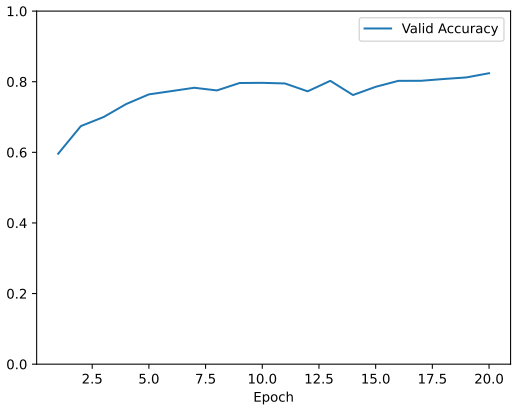


Figure 15: *Training accuracy ($\eta = 0.01$)*

c) Solution Q2.2 c

Using a batch size of 256, we ran the default model for 150 epochs with the objective of examining potential overfitting. Additionally, we explored two regularization approaches for these parameters. The first involves dropout with a probability of 0.2, and the results are presented in Figures 16 and 17. The second approach employs L2 regularization with a value of 0.0001, as shown in Figures 18 and 19. Despite the incorporation of regularization, a slight degree of overfitting is observed with increasing epochs, we noticed that the validation loss stopped improving while the training loss kept decreasing. Test accuracies for each setting are 0.7656 for dropout and 0.7580 for L2 regularization.

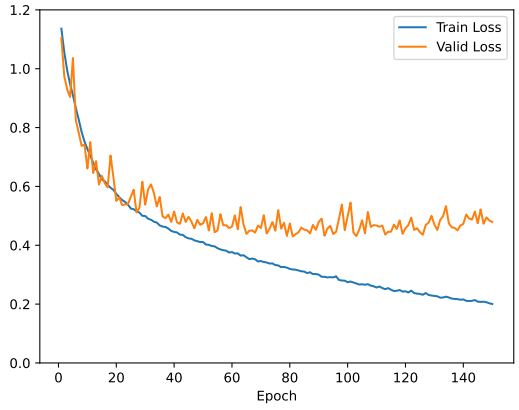


Figure 16: *Training loss (dropout = 0.2)*

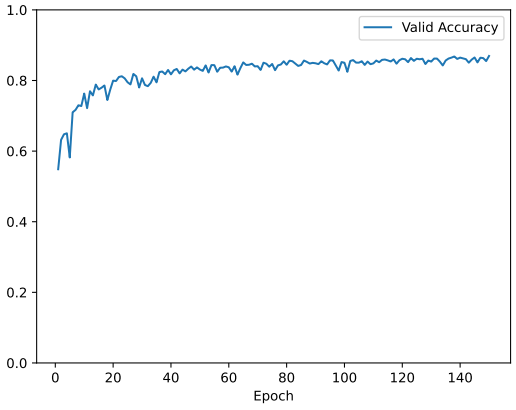


Figure 17: *Training accuracy (dropout = 0.2)*

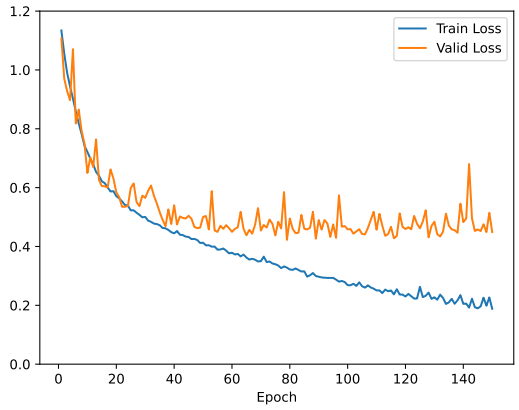


Figure 18: *Training loss (L2 = 0.0001)*

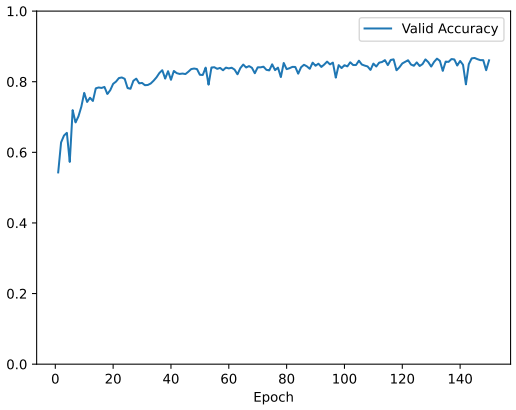


Figure 19: *Training accuracy (L2 = 0.0001)*

3 Question

In this exercise we design a multilayer perceptron to compute a Boolean function of D variables, $f : \{-1, +1\}^D \rightarrow \{-1, +1\}$, defined as:

$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{D} x_i \in [A, B], \\ -1 & \text{otherwise,} \end{cases}$$

where A and B are integers such that $-D \leq A \leq B \leq D$.

a) Solution Q3.1 a

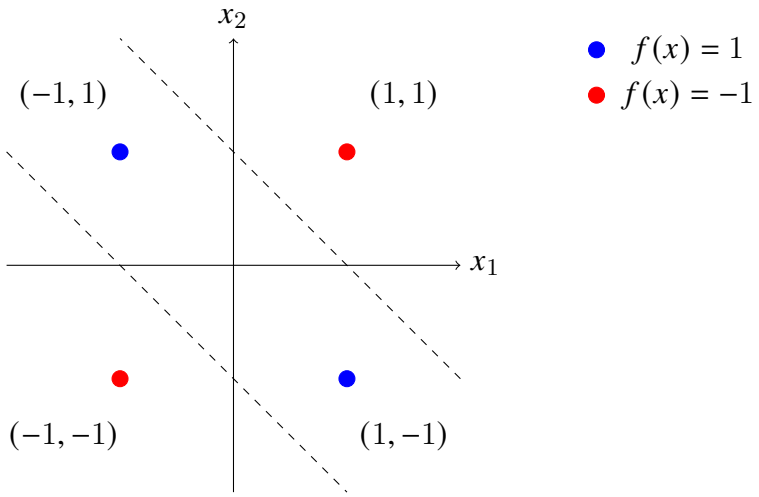
Let's show that the function above cannot generally be computed with a single perceptron. Consider the case where $D = 2$ and $A = 0, B = 1$. The function $f(x)$ then becomes:

$$f(x) = \begin{cases} 1, & \text{if } x_1 + x_2 \in [0, 1] \\ -1, & \text{otherwise} \end{cases}$$

Let's analyze the truth table for this function:

x_1	x_2	$x_1 + x_2$	$f(x)$
-1	-1	-2	-1
-1	1	0	1
1	-1	0	1
1	1	2	-1

Now, let's visualize this in the input space.



If we try to separate the points where $f(x) = 1$ from the points where $f(x) = -1$ using a single line (which is what a single perceptron can do), it is not possible. The region where $f(x) = 1$ is not linearly separable from the region where $f(x) = -1$ in this case.

Therefore, a single perceptron cannot accurately represent this function for all possible inputs, demonstrating that this function cannot generally be computed with a single perceptron.

b) Solution Q3.1 b

Parameters that are used in calculations:

$$
\begin{aligned}
&x \in \{-1, 1\}^D \ldots \text{input vector},\\
&W^1 \in \mathbb{R}^{D\times 2} \ldots \text{wights matrix of first hidden layer},\\
&b^1 \in \mathbb{R}^2 \ldots \text{bias vector of first hidden layer},\\
&z \in \{-1, 1\}^2 \ldots \text{output of first hidden layer},\\
&W^2 \in \mathbb{R}^2 \ldots \text{wights matrix of output layer},\\
&b^2 \in \mathbb{R} \ldots \text{bias of output layer}.
\end{aligned}
$$

Lets define $S := \sum_{i=1}^{D} x_i$, for more compact notation. In the first layer we want that each of the perceptrons provides two conditions. The first is that $S \geq A$, and the second is that $S \leq B$. If we look at formula for each component we get:

$$
\begin{aligned}
sign(\sum_{i=1}^{D} x_i w_{i1}^{(1)} + b_1^{(1)}) = 1 \Leftrightarrow S \geq A,\\
sign(\sum_{i=1}^{D} x_i w_{i2}^{(1)} + b_2^{(1)}) = 1 \Leftrightarrow S \leq B.
\end{aligned}
$$

One of the solution is

$$
\begin{aligned}
w_{i1}^{(1)} &= 1, \text{ for } i = 1, \ldots, D\\
w_{i2}^{(1)} &= -1, \text{ for } i = 1, \ldots, D\\
b_1^{(1)} &= -A\\
b_2^{(1)} &= B
\end{aligned}
$$

By setting these parameters, we can check the possible values of z_1 and z_2, which are the output components of the first hidden layer.

z_1	z_2	Condition
1	1	$A \leq S \leq B$
1	-1	$A \leq B < S$
-1	1	$S < A \leq B$
-1	-1	never

If we look at the second (output) layer, we get equation

$$sign(w_1^{(2)} z_1 + w_2^{(2)} z_2 + b^{(2)}) = 1 \Leftrightarrow z_1 = z_2 = 1,$$

which determines the parameters of the second layer as

$$w_1^{(2)} = 1,\ w_2^{(2)} = 1 \text{ and } b^{(2)} = -1.$$

The above calculation and setting of weights and biases also applies to that the resulting network is robust to infinitesimal perturbation of the inputs, we can check this by: Lets $v \in \mathbb{R}^D$ and $t > 0$. We can use above define network and calculate the limit $\lim_{t \to 0} h(x + tv) = h(x) = f(x)$, which result the same output.

c) Solution Q3.1 c

Lets now use ReLU activation function in hidden layer, which results similar idea as in previous example. In the first layer we want that each of the perceptrons provides two conditions. The first is that $S \geq A$, and the second is that $S \leq B$. If we look at formula for each component we get:

$$ReLU(\sum_{i=1}^{D} x_i w_{i1}^{(1)} + b_1^{(1)}) = 0 \Leftrightarrow S \geq A,$$
$$ReLU(\sum_{i=1}^{D} x_i w_{i2}^{(1)} + b_2^{(1)}) = 0 \Leftrightarrow S \leq B.$$

One of the solution is

$$w_{i1}^{(1)} = -1, \text{ for } i = 1, \ldots, D$$
$$w_{i2}^{(1)} = 1, \text{ for } i = 1, \ldots, D$$
$$b_1^{(1)} = A$$
$$b_2^{(1)} = -B$$

By setting these parameters, we can check the possible values of z_1 and z_2, which are the output components of the first hidden layer and $a, b > 0$.

z_1	z_2	Condition
0	0	$A \leq S \leq B$
0	b	$A \leq B < S$
a	0	$S < A \leq B$
a	b	never

If we look at the second (output) layer, we get equation

$$sign(w_1^{(2)} z_1 + w_2^{(2)} z_2 + b^{(2)}) = 1 \Leftrightarrow z_1 = z_2 = 0,$$

which determines the parameters of the second layer as

$$w_1^{(2)} = -1,\ w_2^{(2)} = -1 \text{ and } b^{(2)} = 0.$$

We can check the solution with example where $D = 2$, $A = B = 0$. Calculation for first layer are:

$$z_1 = ReLU(0 - x_1 - x_2) = 0 \Leftrightarrow x_1 + x_2 \geq 0 = A$$
$$z_2 = ReLU(x_1 + x_2 - 0) = 0 \Leftrightarrow x_1 + x_2 \leq 0 = B$$

If we use all possible combinations of input data in the above equations, we get zero values only if the condition $x_1 + x_2 = 0$ is satisfied. Since $z_1 = z_2 = 0$, we get $sign(0)$, which returns 1. This is how we checked that our network is working.

4 Contribution of each member

In our project, each team member contributed equally for the project. We maintained a collaborative spirit throughout, working closely together and meeting regularly to ensure shared decision-making. This balanced approach allowed us to leverage individual strengths, resulting in a successful project outcome.