Deep Learning

Homework 1

Group 71 members:

- Luis Jose De Macedo Guevara (95621)
- Vincent Jakl (108529)

Contributions of each member

- Luis Jose De Macedo Guevara (95621): partially Q1.2a, Q1.2b, corrections on Q2.1b, partially Q2.2, Q3.
- Vincent Jakl (108529): Q1.1, partially Q1.2a, corrections on Q1.2b, partially Q2.2, corrections on Q3.

Question 1

1. a)

The single perceptron was not the best choice for this task. As can be seen in Figure 1 below, the perceptron was not able to fit to the high dimensional data also with no sign of improvement. With the 20 epochs, it ended up with a 0.3422 test accuracy which is slightly higher than 0.25 that would happen with random selection for 4 classes. Also the learning rate was left at the value of 1 so this might cause the problem as well.

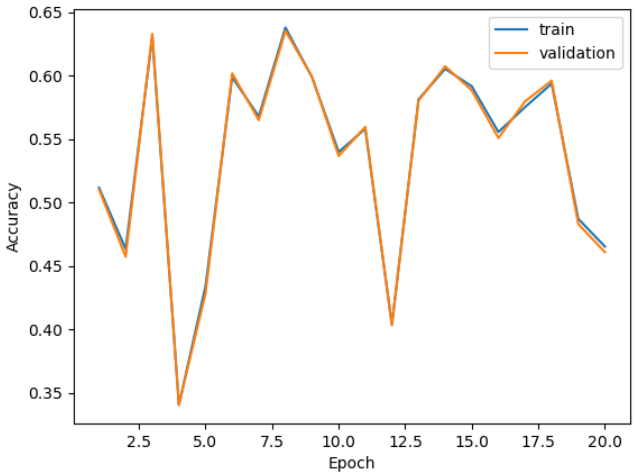
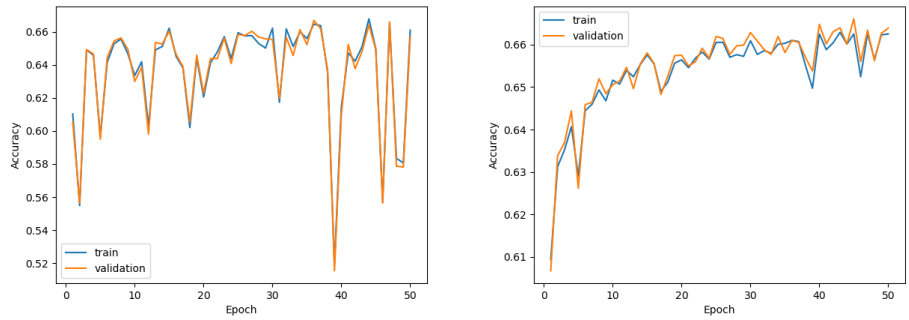


Figure 1: Single perceptron

1. b)

The logistic regression performed better than the single perceptron. As can be seen in Figure 2, the logistic regression was able to fit to the data better. There was however a big difference between the two learning rates. Where the model with the learning rate of 0.01 jumped faster to the higher accuracy, it did not improve on a regular basis. The 0.001 lr model was a bit slower to get up to higher accuracy, but it was able to consistently improve. In the end the 0.01 lr model ended up with 0.5784 test accuracy and the 0.001 lr model ended up with 0.5936 test accuracy. Running these models for more epochs would possibly improve the accuracy since the training was still showing

signs of improvement even with the evaluation metric.



(a) lr 0.01 (b) lr 0.001

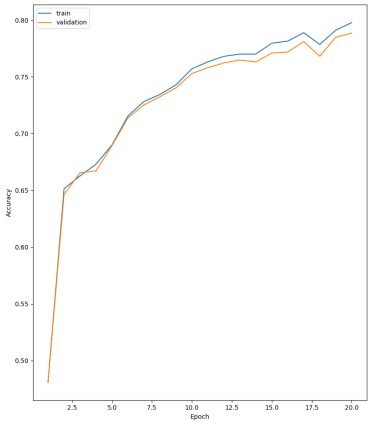
Figure 2: Logistic regression

2. a)

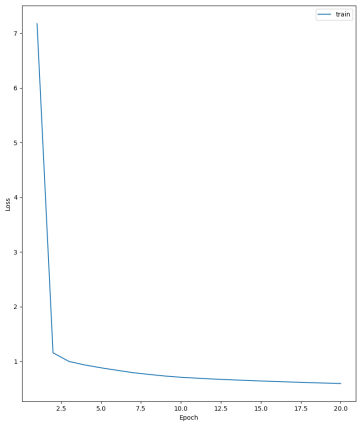
We do believe that the statement is correct. The multilayer perceptron can be more expressive than logistic regression. The logistic regression is a linear model and therefore it can only learn linear decision boundaries. The multilayer perceptron can learn non-linear decision boundaries due to the ReLU activation. However, the logistic regression is a convex optimization problem whereas an MLP generally is not, making it easier to train and find the global minima in logistic regression.

2. b)

In Figure 3 we can see that the multilayer perceptron, as specified, is a better fit for the data, not only its accuracy on both training and validation data largely improves with each epoch, so does its training loss. It also shows a good test accuracy of 0.7580 after 20 epochs. Looking at the trend in both plots, we can expect a better accuracy on the training/validation data with more epochs and possibly, taking care to avoid overfitting, a better testing accuracy.



(a) Training/Validation accuracy per epoch



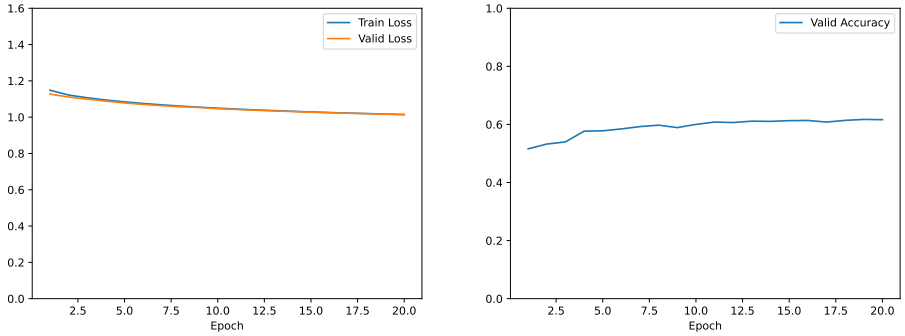
(b) Loss as a function of epoch number

Figure 3: MLP w/o Pytorch

Question 2

1.

After implementing the logistic regression with Pytorch, the best performing learning rate was 0.001 with the final test accuracy of 0.6503. All the test accuracies are as follows 0.5577, 0.6200, 0.6503 in the order of learning rate 0.1, 0.01, 0.001. Again here it looks like 20 training epochs were not enough to get the best performance out of the model.

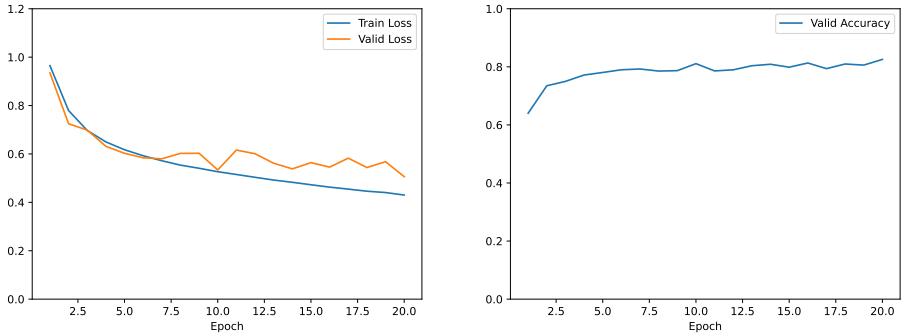


(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 4: Logistic regression lr 0.001

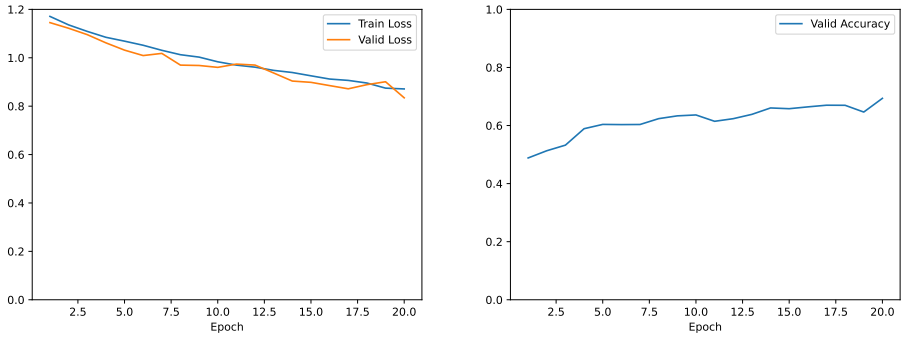
2. a)

When training the MLP with Pytorch, the batch size of 16 trained slower than the batch size of 1024. The training with batch size 16 finished in 71 seconds whereas with batch size 1024 it finished in 17 seconds. However, since the batch size 16 trained on overally more batches, it was able to perform better. The final testing accuracy with batch size 16 was 0.7656 whereas with batch size 1024 was 0.7278.



(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 5: MLP batch size 16

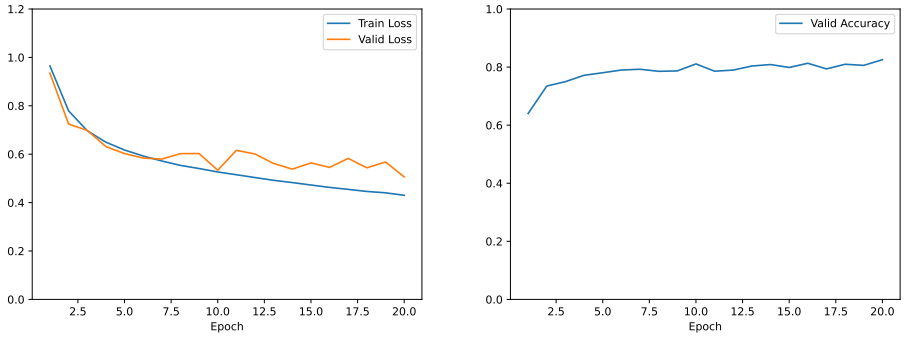


(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 6: MLP batch size 1024

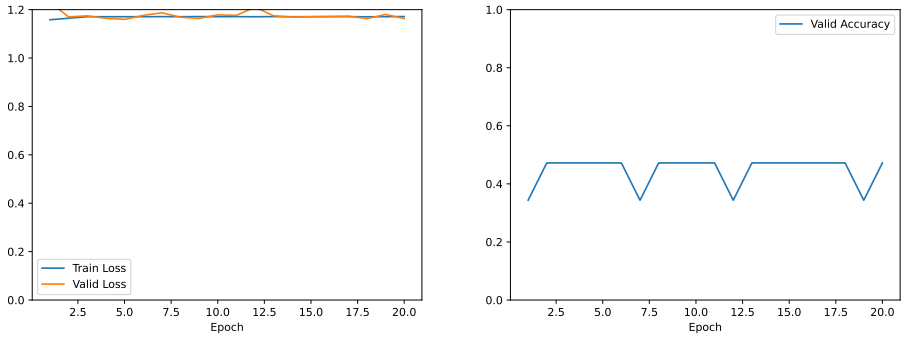
2. b)

Out of the four learning rates, the worst performing one was lr 1 (Figure 8). This makes sense, since the gradient descent with this high a learning rate probably overshoots the minimum. The best performing learning rate, in terms of validation accuracy, was 0.1 (Figure 7) though 0.01 was not far behind, with a learning rate of 0.1 we got a final validation accuracy of 0.8255, whereas the learning rate of 1 had the final validation accuracy of 0.4721. The best test accuracy, 0.7656 also comes from setting the learning rate to 0.1.



(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 7: MLP with lr 0.1

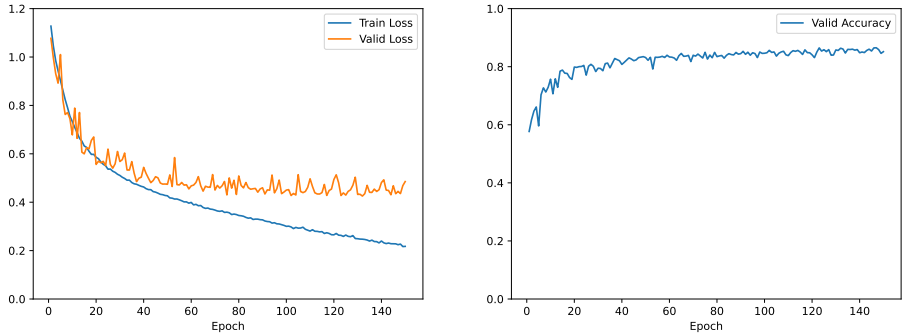


(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 8: MLP with lr 1

2. c)

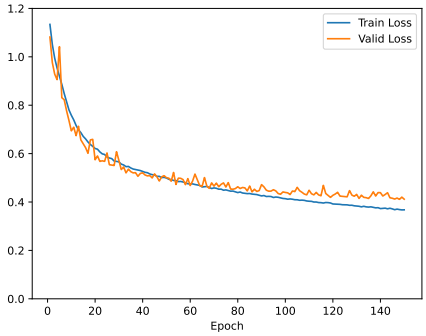
When training the network for 150 epochs and 256 batch size, the model gets to higher accuracies than the previous models. However in the training/validation loss graph (Figure 9) we can see, that the training and validation loss start to diverge after around 40 epochs. This is a sign of overfitting.



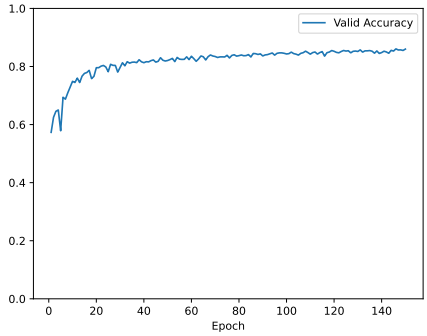
(a) Training/Validation loss per epoch (b) Validation accuracy per epoch

Figure 9: MLP overfitting

When comparing the L2 regularization (Figure 11) and dropout regularization (Figure 10), the better final test accuracy is the one with the dropout regularization, the L2 regularization, as specified, is not enough to avoid overfitting, maybe a higher value could improve the validation accuracy. The dropout regularization reduces significantly the overfitting, although in the plot it can be seen a little divergence between the training and validation, which may be a sign of overfitting. The final test accuracy for the L2 regularization was 0.7750 and for the dropout regularization it was 0.7902.

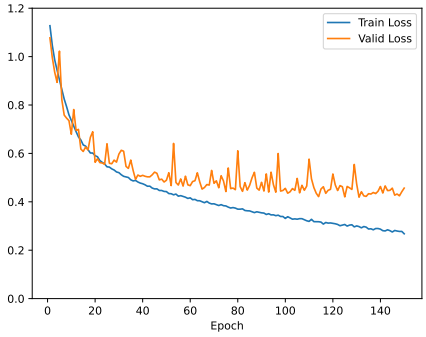


(a) Training/Validation loss per epoch

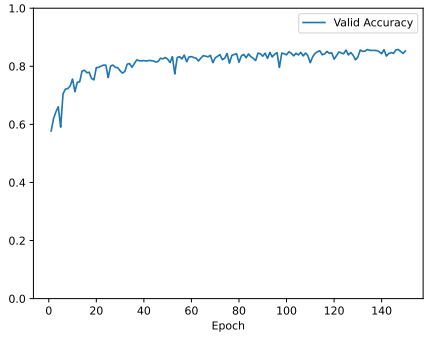


(b) Validation accuracy per epoch

Figure 10: MLP dropout



(a) Training/Validation loss per epoch



(b) Validation accuracy per epoch

Figure 11: MLP L2 regularization

Question 3

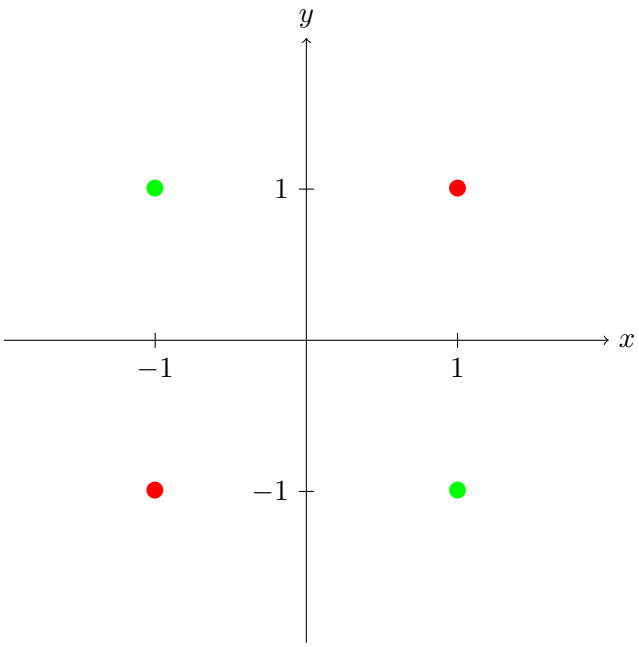
1.

a)

Let

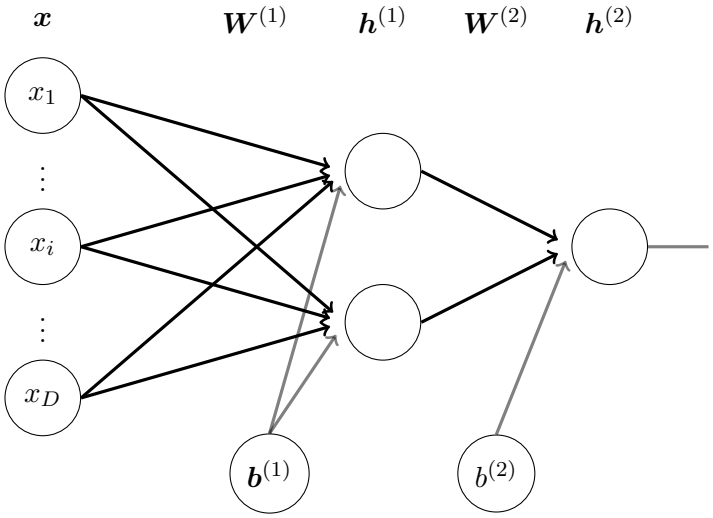
- $D = 2$
- $A = B = 0$
- $\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\boldsymbol{x}^{(2)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\boldsymbol{x}^{(3)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\boldsymbol{x}^{(4)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$,

We have $f(\boldsymbol{x}^{(1)}) = -1$, $f(\boldsymbol{x}^{(2)}) = +1$, $f(\boldsymbol{x}^{(3)}) = +1$ and $f(\boldsymbol{x}^{(4)}) = -1$



Which we can see is not linearly separable and therefore a perceptron cannot learn a separating hyperplane.

b)



$$\boldsymbol{W}^{(1)} = \underbrace{\begin{bmatrix} — & \boldsymbol{w}_1^{(1)} & — \\ — & \boldsymbol{w}_2^{(1)} & — \end{bmatrix}}_{2\times D}, \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} w_1^{(2)} & w_2^{(2)} \end{bmatrix}, \qquad b^{(2)}$$

We have that $A \leq \sum_{i=1}^{D} x_i \leq B$ iff

- $\sum_{i=1}^{D} x_i \leq A \iff \sum_{i=1}^{D} x_i - A \geq 0$ and
- $\sum_{i=1}^{D} x_i \geq B \iff B - \sum_{i=1}^{D} x_i \geq 0$.

Knowing that

$$\boldsymbol{h}^{(1)} = \text{sign}\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) = \begin{bmatrix} \text{sign}\left(\boldsymbol{w}_1^{(1)}\boldsymbol{x} + b_1^{(1)}\right) \\ \text{sign}\left(\boldsymbol{w}_2^{(1)}\boldsymbol{x} + b_2^{(1)}\right) \end{bmatrix}$$

We can set $\boldsymbol{W}^{(1)} = \left[\begin{smallmatrix} 1 & \cdots & 1 \\ -1 & \cdots & -1 \end{smallmatrix}\right]$ and $\boldsymbol{b}^{(1)} = \left[\begin{smallmatrix} -A \\ B \end{smallmatrix}\right]$ to get

$$\boldsymbol{h}^{(1)} = \begin{bmatrix} \text{sign}\left(\sum_{i=1}^{D} x_i - A\right) \\ \text{sign}\left(B - \sum_{i=1}^{D} x_i\right) \end{bmatrix}$$

which results in

$$\boldsymbol{h}^{(1)} = \begin{cases} \begin{bmatrix} +1 \\ +1 \end{bmatrix} & \text{if } A \leq \sum_{i=1}^{D} x_i \leq B \\ \begin{bmatrix} -1 \\ +1 \end{bmatrix} & \text{if } \sum_{i=1}^{D} x_i < A \leq B \\ \begin{bmatrix} +1 \\ -1 \end{bmatrix} & \text{if } A \leq B < \sum_{i=1}^{D} x_i \end{cases}$$

In order to learn the function f, we just need to choose $\boldsymbol{W}^{(2)}$ and $b^{(2)}$ such that

$$\boldsymbol{h}^{(2)} = \begin{cases} +1 & \text{if } \boldsymbol{h}^{(1)} = \left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right] \\ -1 & \text{if } \boldsymbol{h}^{(1)} \neq \left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right] \end{cases}$$

equivalently

- $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)} \geq 0$ when $\boldsymbol{h}^{(1)} = \left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right]$
- $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)} < 0$ when $\boldsymbol{h}^{(1)} \neq \left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right]$

Which we can achieve by

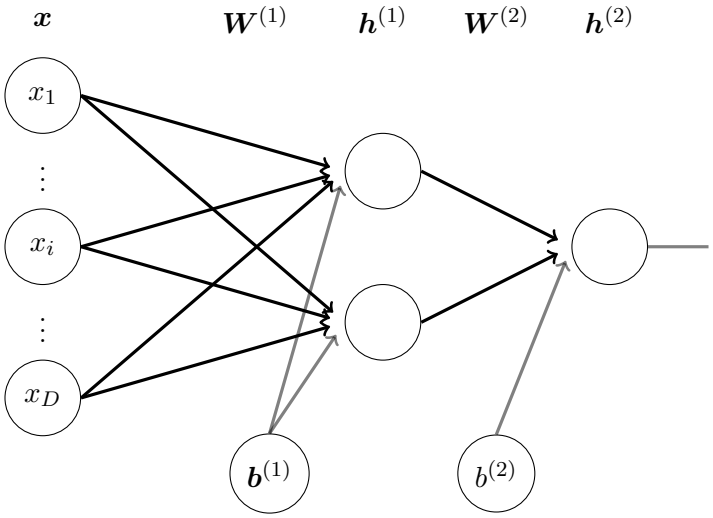
1. setting $\boldsymbol{W}^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}$, from which we get $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)}$ equal to 2, 0 and 0 for $\left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} -1 \\ +1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} +1 \\ -1 \end{smallmatrix}\right]$ respectively, and
2. setting $b^{(2)} = -1$, from which we get $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)}$ equal to 1, -1 and -1 for $\left[\begin{smallmatrix} +1 \\ +1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} -1 \\ +1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} +1 \\ -1 \end{smallmatrix}\right]$ respectively.

and so we learn our desired function with the weights and biases:

$$\boldsymbol{W}^{(1)} = \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ -1 & \cdots & -1 \end{bmatrix}}_{2 \times D}, \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} -A \\ B \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \qquad b^{(2)} = -1$$

c)



$$\boldsymbol{W}^{(1)} = \underbrace{\begin{bmatrix} — & \boldsymbol{w}_1^{(1)} & — \\ — & \boldsymbol{w}_2^{(1)} & — \end{bmatrix}}_{2\times D}, \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} w_1^{(2)} & w_2^{(2)} \end{bmatrix}, \qquad b^{(2)}$$

We have that $A \leq \sum_{i=1}^{D} x_i \leq B$ iff

- $\sum_{i=1}^{D} x_i \leq A \iff A - \sum_{i=1}^{D} x_i \geq 0$ and
- $\sum_{i=1}^{D} x_i \geq B \iff \sum_{i=1}^{D} x_i - B \geq 0$.

Knowing that

$$\boldsymbol{h}^{(1)} = \text{ReLU}\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) = \begin{bmatrix} \text{ReLU}\left(\boldsymbol{w}_1^{(1)}\boldsymbol{x} + b_1^{(1)}\right) \\ \text{ReLU}\left(\boldsymbol{w}_2^{(1)}\boldsymbol{x} + b_2^{(1)}\right) \end{bmatrix}$$

We can set $\boldsymbol{W}^{(1)} = \left[\begin{smallmatrix} -1 & \cdots & -1 \\ 1 & \cdots & 1 \end{smallmatrix}\right]$ and $\boldsymbol{b}^{(1)} = \left[\begin{smallmatrix} A \\ -B \end{smallmatrix}\right]$ to get

$$\boldsymbol{h}^{(1)} = \begin{bmatrix} \text{ReLU}\left(A - \sum_{i=1}^{D} x_i\right) \\ \text{ReLU}\left(\sum_{i=1}^{D} x_i - B\right) \end{bmatrix}$$

which results in

$$\boldsymbol{h}^{(1)} = \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \end{bmatrix} \text{ where } \begin{cases} h_1^{(1)} = 0 & \text{if } \sum_{i=1}^{D} x_i \geq A \\ h_1^{(1)} > 0 & \text{if } \sum_{i=1}^{D} x_i < A \\ h_2^{(1)} = 0 & \text{if } \sum_{i=1}^{D} x_i \leq B \\ h_2^{(1)} > 0 & \text{if } \sum_{i=1}^{D} x_i > B \end{cases}$$

In order to learn the function, we just need to choose $\boldsymbol{W}^{(2)}$ and $b^{(2)}$ such that

$$\boldsymbol{h}^{(2)} = \begin{cases} +1 & \text{if } \boldsymbol{h}^{(1)} = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right] \\ -1 & \text{otherwise} \end{cases}$$

equivalently

- $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)} \geq 0$ when $\boldsymbol{h}^{(1)} = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$
- $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)} < 0$ when $\boldsymbol{h}^{(1)} \neq \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$

Knowing that $h_1^{(1)} \geq 0$ and $h_2^{(1)} \geq 0$ we just need to set $\boldsymbol{W}^{(2)} = \begin{bmatrix} -1 & -1 \end{bmatrix}$ and $b^{(2)} = 0$, from which we get $\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + b^{(2)}$ equal to 0 when $\boldsymbol{h}^{(1)} = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$, and negative otherwise.

and so we learn our desired function with the weights and biases:

$$\boldsymbol{W}^{(1)} = \underbrace{\begin{bmatrix} -1 & \cdots & -1 \\ 1 & \cdots & 1 \end{bmatrix}}_{2 \times D}, \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} A \\ -B \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} -1 & -1 \end{bmatrix}, \qquad b^{(2)} = 0$$