1 Single Perceptron

Using the perceptron model illustrated in Fig **??**. Choose a weight vector which will make the perceptron capable to replicate AND, OR, NAND and NOR gates respectively. The inputs x_1, x_2 and the output y are binary.

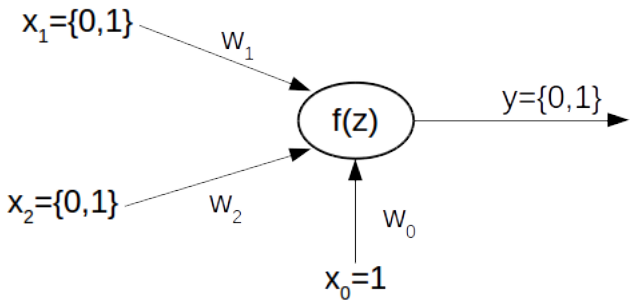

Figure 1: Single perceptron with two inputs and bias

$$z = \sum_{i=0}^{2} w_i x_i \qquad f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

AND

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

OR

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

NAND

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	0

NOR

2 Single perceptron without bias

Consider a single perceptron without bias. Given a set of $n = 4$ samples $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ in 3-dimensional space, where $\mathbf{x}_1 = [1, 0, 0]^T$, $\mathbf{x}_2 = [0, 1, 0]^T$, $\mathbf{x}_3 = [0, 0, 1]^T$, $\mathbf{x}_4 = [1, 1, 1]^T$, and the corresponding desired outputs y_1, y_2, y_3, y_4. Find any parameter vector $\mathbf{w}$ such as:

1. $y_1 = y_2 = y_3 = y_4 = 1$
2. $y_1 = y_2 = 1$ and $y_3 = y_4 = 0$
3. Find a set of values y_1, y_2, y_3, y_4 which the perceptron cannot realize for the given inputs $\mathbf{x}_n$

Use the same activation function $f(\cdot)$ as in Exercise 1.
Hint: Consider the problem as classification where inputs are $\mathbf{x_n}$ and labels are y_n

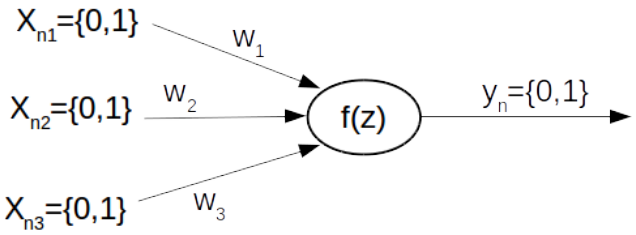

Figure 2: Single perceptron taking three-dimensional inputs, without bias

3 Computational Cost of Neural Networks

1. **Open Question** [The solution to this question is not needed for the rest of the exercise.]
   You are employed by a big software company and are asked to create a neural network that is able to play the game of Go (see Fig. **??**). The go board consists of a board with a grid of 19x19, and is played by two opponents (black and white). Each turn a player can either place a stone or pass. The goal of the game is to *encircle* the opponent.
   How could you design an artificial neural network that can be used to play the game of GO?

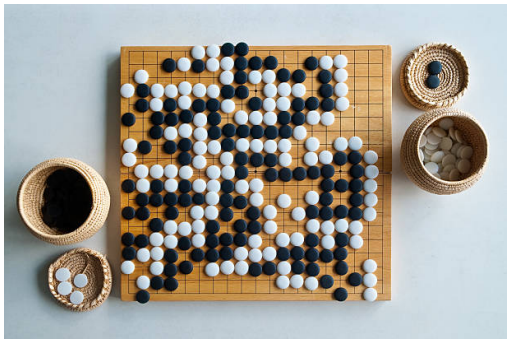

Figure 3: Go Board

2. You ended up choosing a neural network with input and output sizes of 361 each, additionally, it has 12 hidden layers each of size 722.
   How many parameters does your model have? How much memory is needed to store all of them? (Assume all parameters to be double values, i.e., 8 bytes on a 64-bit system.)

3. You want to take expert games to train your network. Let us assume that a game of GO has on average 200 moves. You use each move of the winner as one data point. You want to train your model with a train-test ratio of 70:30, and you want to have at least $500N$ samples to N parameters. How many games will you need?

4. How many operations will be needed for the inference[1] of the network? How fast could this be evaluated on a 4GHz CPU (we simplify that one mathematical operation can be executed in 5 clock cycles, including loading and storing).

5. The large computational cost and time stem from training these networks. The neural network-based Go-engine AlphaGo[2] used supposedly 50 GPUs for 3 weeks. Let us assume they used GPUs with an average power consumption of 200 W. How much energy did the training consume? Compare this to the average per capita electricity consumption of a Switzerland (6721 kWh / year).

6. In 2020, the natural language model GPT-3[3] used around 1000 GPUs for 34 days to train the network. How does this compare to the AlphaGo model? How do you explain the difference?

[1] Inference is the process of running data points into a machine learning model to calculate an output such as a numerical score.

[2] `https://en.wikipedia.org/wiki/AlphaGo`

[3] `https://en.wikipedia.org/wiki/GPT-3`

4 Case Study Robotics: Comparison of Algorithms

You want to learn a model to predict the self-collision of a robot. The input is the joint positions, and the model predicts collisions (value=1) or predicts when the robot is in a collision-free configuration (value=-1). The training data points are obtained from a simulator.
You ask one of your engineers to train different models. The engineer comes back with the results below:

Learning Method	SVM	NN
Model Size [# of doubles]	2'099'442	15'333
Total training time [s]	2106 (GPU)	11250 (GPU)
True Positive Rate (TPR) [ ]	0.995	0.940
True Negative Rate (TNR) [ ]	0.960	0.990
Inference time [ms]	1.38 (GPU)	0.11 (CPU)

Table 1: Comparison of performance and energy consumption of two classifiers on collision vs. no-collision dataset. Assume a power consumption of 125 W for the GPU and 50 W for the CPU.

1. Compare the energy usage between the two algorithms.
2. Compare the energy usage for inference of the algorithm.
3. Which algorithm would you choose? Justify your choice.

5 Two-layer Feed-Forward Neural Network

Find a set of weights for the two-layer feed-forward neural network (Fig. **??**) for which it can learn the XOR gate (see Table). Consider binary inputs ($x_i = \{0, 1\}$) and output ($y = \{0, 1\}$) also consider bias at the hidden and output nodes. Using the same activation function $f(\cdot)$ as in Exercise 1, the nodes are activated by the following equations:

$$h_n = f\left(\sum_{i=1}^{2} w_{in} x_i + w_n\right)$$

$$f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

$$y = f\left(\sum_{i=1}^{2} w_{iy} h_i + w_y\right)$$

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

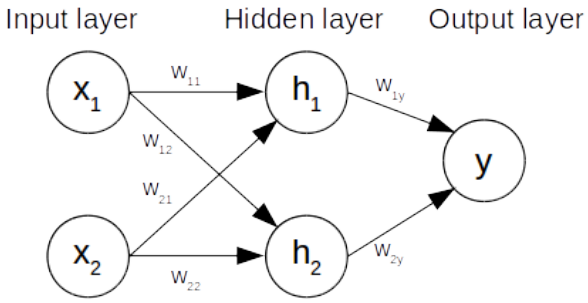

Figure 4: Feed-Forward ANN with a hidden layer

6 Recurrent Neural Network (RNN)

Find a set of weights for the RNN illustrated below such that it exhibits oscillating behavior (i.e. the values of the nodes alternate continuously between 0 and 1). Consider that the two nodes are activated synchronously as:

$$x_2^{t+1} = f(w_{12}x_1^t + w_{22}x_2^t) \qquad x_1^{t=0} = 1 \qquad f(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$$

$$x_1^{t+1} = f(w_{21}x_2^t + w_{11}x_1^t) \qquad x_2^{t=0} = 0$$

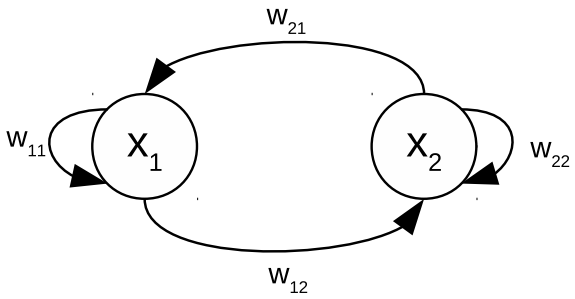

Figure 5: Recurrent Neural Network