# Exercise 2

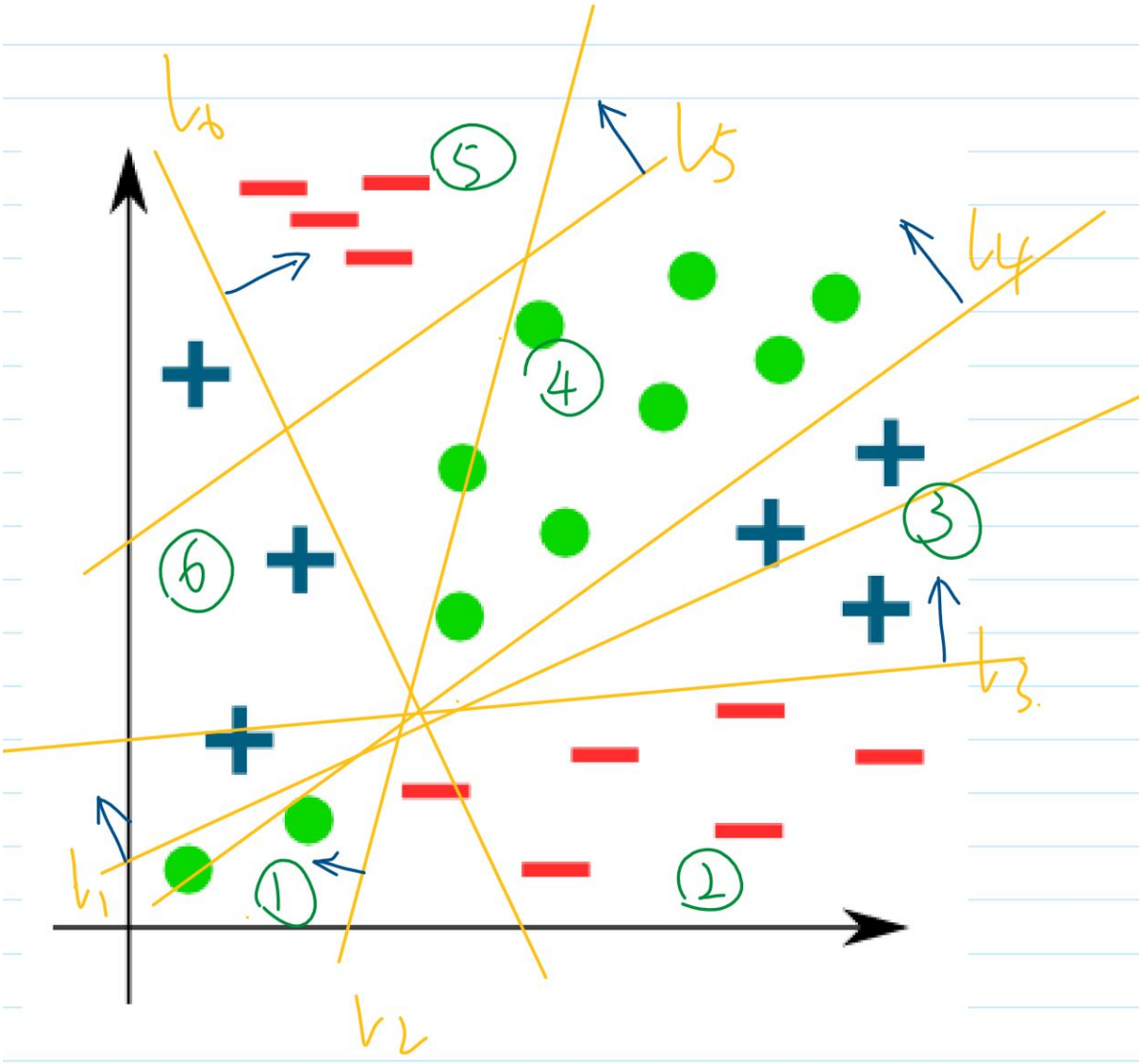
Authors:

- Tuoxing Liu
- Sima Esmaeili
- Shruti Ghargi

## Exercise 2.1

First layer's decision boundaries and normal vectors:

- The decision boundaries are drawn as yellow straight lines
- The normal vectors are drawn in blue arrows




### Design single neurons

Design single neurons to perform following tasks.

### logical OR

- weights: all 1 vector
- bias: -0.5 (to deal with the problem of comparing float point numbers)
- activation function:

$$f:\mathbb{R}\to\{0,1\}\quad f(x)=\begin{cases}1 & \text{for} \quad x>0\\ 0 & \text{for} \quad x\leq 0\end{cases}$$

### masked logical OR

- weights: the fixed binary vector $c\in\{0,1\}^D$
- bias : -0.5
- activation function:

$$f:\mathbb{R}\to\{0,1\}\quad f(x)=\begin{cases}1 & \text{for} \quad x>0\\ 0 & \text{for} \quad x\leq 0\end{cases}$$

### perfect match

for the fixed binary vector c

- weights: $v_{pow}:=[1,2,4,\dots,2^D]^T$
- bias : 0
- activation function:

$$f:\mathbb{R}\to\{0,1\}\quad f(x)=\begin{cases}1 & \text{for} \quad x=c\cdot v_{pow}\\ 0 & \text{for} \quad \text{otherwise}\end{cases}$$

Note the output of first layer as $v_i$ in the following way:

$$\begin{aligned}&(1)v_1:=[0,1,0,1,0,0]\\&(2)v_{2,3}:=[0,0,0,0,0,0/1]\\&(3)v_{4,5}:=[0/1,0,1,0,0,1]\\&(4)v_{6,7}:=[1,0/1,1,1,0,1]\\&(5)v_8:=[1,1,1,1,1,1]\\&(6)v_{9-12}:=[1,1,0/1,1,0/1,0]\end{aligned}$$

## For the second layer:

- Contain 12 neurons $N_{2,i},i=1,\dots,12$
- The weight of neurons is $v_{pow}:=[1,2,4,\dots,2^D]^T$
- The bias of all neurons is 0
- The activation functions are:

$$\sigma_{2,i}:\mathbb{R}\to\{0,1\}\quad \sigma_{2,i}(x)=\begin{cases}1 & \text{for} \quad x=v_i\cdot v_{pow}\\ 0 & \text{for} \quad \text{otherwise}\end{cases}$$

Then neurons in second layer $N_{2,i},i=1,\dots,12$ are equivalent to $h(z;v_i),i=1,\dots,12$, $h$ is perfect match function.

Define $\delta_S(x),\quad S\subseteq 2^{\mathbb{Z}}$ is a set contains the indexes of $v_i$ .

$$\delta_S:\mathbb{R}\to\{0,1\}\quad \delta_S(x)=\begin{cases}1 & \text{for} \quad x\in S\\ 0 & \text{for} \quad x\notin S\end{cases}$$

## For the third layer:

- Contain 3 neurons $N_{3,1}, N_{3,2}, N_{3,3}$
- Weights:
  - $w_1 := (\delta_{\{1,6,7\}}(a_i))_{i=1,...,12}^T$, i.e. $[1,0,0,0,0,1,1,0,0,0,0,0]^T$
  - $w_2 := (\delta_{\{2,3,8\}}(a_i))_{i=1,...,12}^T$
  - $w_3 := (\delta_{\{4,5,9,10,11,12\}}(a_i))_{i=1,...,12}^T$
- The bias of all neurons is -0.5 (to deal with the problem of comparing float point numbers)
- The activation function of all neurons is

$$\sigma_3 : \mathbb{R} \to \{0,1\} \quad \sigma_3(x) = \begin{cases} 1 & \text{for} \quad x > 0 \\ 0 & \text{for} \quad x \leq 0 \end{cases}$$

## The equations for the network's second and third layer:

- second layer: $Z_2 = (h(Z_1; v_i))_{i=1,...,12}$ or $Z_2 = (\sigma_{2,i}(Z_1 \cdot v_{pow}))_{i=1,...,12}$, $h$ is the perfect match function, $v_i$ are the results of first layer, each $v_i$ represents a cluster or one corner of the hypercube
- third layer: $Z_3 = (g(Z_2; w_i))_{i=1,2,3}$ or $Z_3 = \sigma_3(Z_2 \cdot [w_1, w_2, w_3])$, $g$ is the masked logical OR function, $w_i$ are the masks for different clusters with same data label.

## How could it be generalized to arbitrary many input dimensions and arbitrary (non degenerate) label distributions?

The input dimension mainly changes the first layer of the network. The higher the input dimension, the higher the dimension of hyperplanes is needed to use to separate the data. The length of weights in the first layer can be very large if the input dimension is very large. However, this classifier still works.

The label distribution only changes the number of hyperplanes. The more dispersed the data label is, the more hyperplanes are needed to separate the data. As long as the data can still be separated by hyperplanes, it doesn't change the conclusion.

## The potential problems with this classifier are:

- Some data can NOT be classified: e.g. $v = [0,0,0,0,1,0]^T$
- The parameters of classifier can be very large, since it needs to store all unique results from the first layer.
- The number of neurons in the second layer can be very large, since every unique vector from the first layer needs an exclusive neuron.

# Exercise 2.2

$$\begin{aligned} &\text{For L} = 1: \\ &Z_0 = X \\ &\tilde{Z}_1 = Z_0 B_0 + b_0 \\ &Z_1 = \tilde{Z}_1 = X B_0 + b_0 \end{aligned}$$

For L = 2 :

$$\tilde{Z}_2 = Z_1 B_1 + b_1$$

$$\begin{aligned} Z_2 &= \tilde{Z}_2 \\ &= (Z_0 B_0 + b_0) B_1 + b_1 \quad \text{Let} \tilde{B}_1 = B_0 B_1, \tilde{b}_1 = b_0 B_1 + b_1, \\ &= X B_0 B_1 + b_0 B_1 + b_1 \end{aligned}$$

$$\Rightarrow Z_2 = X \tilde{B}_1 + \tilde{b}_1$$

$\therefore$ network with depth L = 2 is equivalent to 1-layer network (a)

For L = n and L = n+1 : $Z_n = Z_{n-1} B_n + b_n$

$$\begin{aligned} Z_{n+1} &= Z_n B_{n+1} + b_{n+1} \\ &= (Z_{n-1} B_n + b_n) B_{n+1} + b_{n+1} \quad \text{Let} \tilde{B}_{n+1} = B_n B_{n+1}, \tilde{b}_{n+1} = b_n B_{n+1} + b_{n+1}, \\ &= Z_{n-1} B_n B_{n+1} + b_n B_{n+1} + b_{n+1} \end{aligned}$$

$$\Rightarrow Z_{n+1} = Z_{n-1} \tilde{B}_{n+1} + \tilde{b}_{n+1}$$

$\therefore$ network with depth L = n is equivalant to network with depth L = n + 1 (b)

$\because$ (a) and (b)

$\therefore$ Any network with depth L larger than 1 is equivalent to a 1-layer network