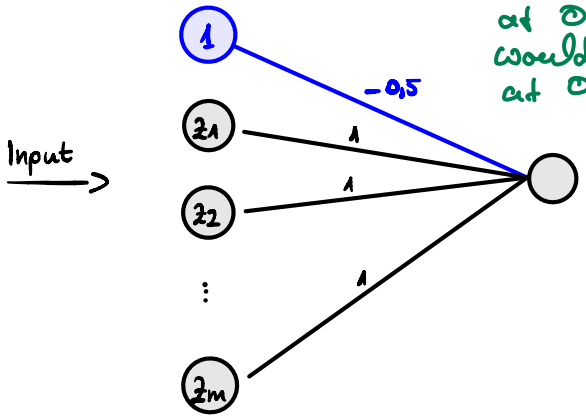
# 1 Classification Capacity

## 1.1 Simple Networks

1.




$M_0$ $\quad B_1 = \begin{pmatrix} -0{,}5 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ $\quad M_1$

Comment: A sigmoid as activation function wouldn't return a binary number. You would have to use a step-function, e.g. with a threshold at 0. Since you added a bias of -0.5, you wouldn't have to worry about the value at 0.

Comment: The weights and bias are the same as in the sample solution. But again, if $\sigma$ is the sigmoid function, then the network won't return 0 or 1. You would need a second activation function $x \mapsto \begin{cases} 1 & x \geq 0.5 \\ 0 & x < 0.5 \end{cases}$ which is unnecessary. You can use a step function straight away.

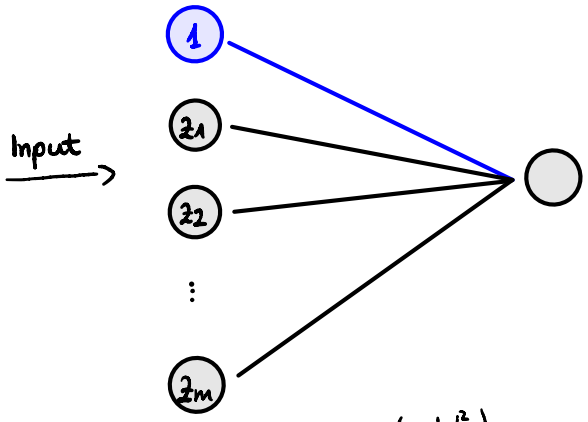
2.

Construct: vector $b = 2c - 1 = \begin{cases} 1 & \text{if } c = 1 \\ -1 & \text{else} \end{cases}$

If you defined $\sigma$ to be $\sigma(x) = \begin{cases} 1 & x \geq 0.5 \\ 0 & x < 0.5 \end{cases}$ then the network would return a wrong result: For example $c = 1 \Rightarrow B = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

$\Rightarrow \sigma((1,1) \cdot B) = \sigma(0) = 0$ ↯ $\quad \Sigma = 2$

You would either have to add a bias of 0.5 as above, or set the step at $x = 0$



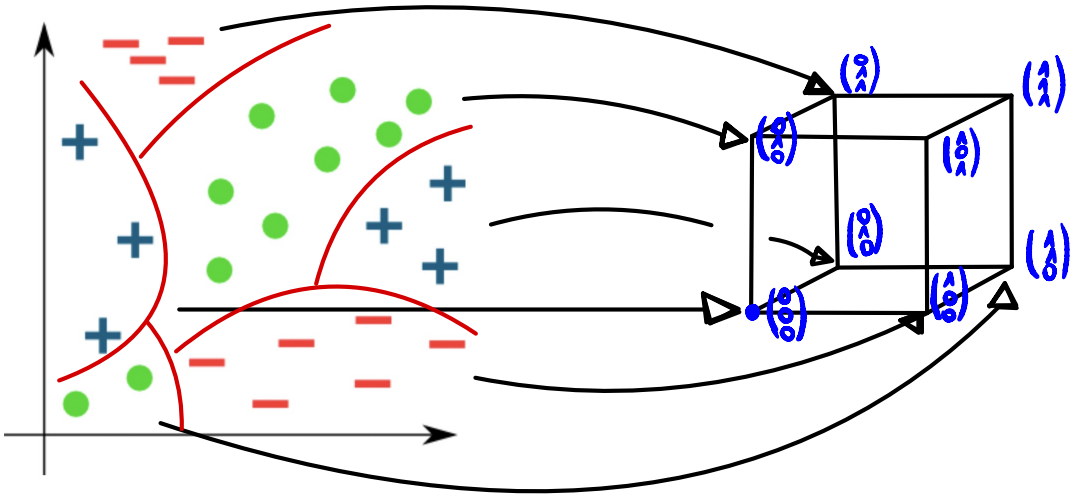

$M_0$ $\quad B_1 = \begin{pmatrix} -|c|^2 \\ b_1 \\ \vdots \\ b_m \end{pmatrix}$ $\quad M_1$

(classify ≥0,5 as 1 and <0,5 as 0)

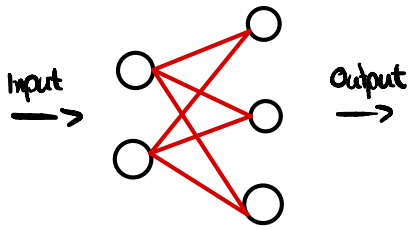
Explanation $\quad z \cdot b = |z| \cdot |b| \cos(\sphericalangle z, b)$

$$\begin{cases} = |c|^2 & \text{if } z = c \\ < |c|^2 & \text{else} \end{cases}$$

3.



We map each cluster on a corner of the cube. Therefor in this case we need three output neurons. The activation functions and weights must be designed to mapp onto one of the vectors of the cube.

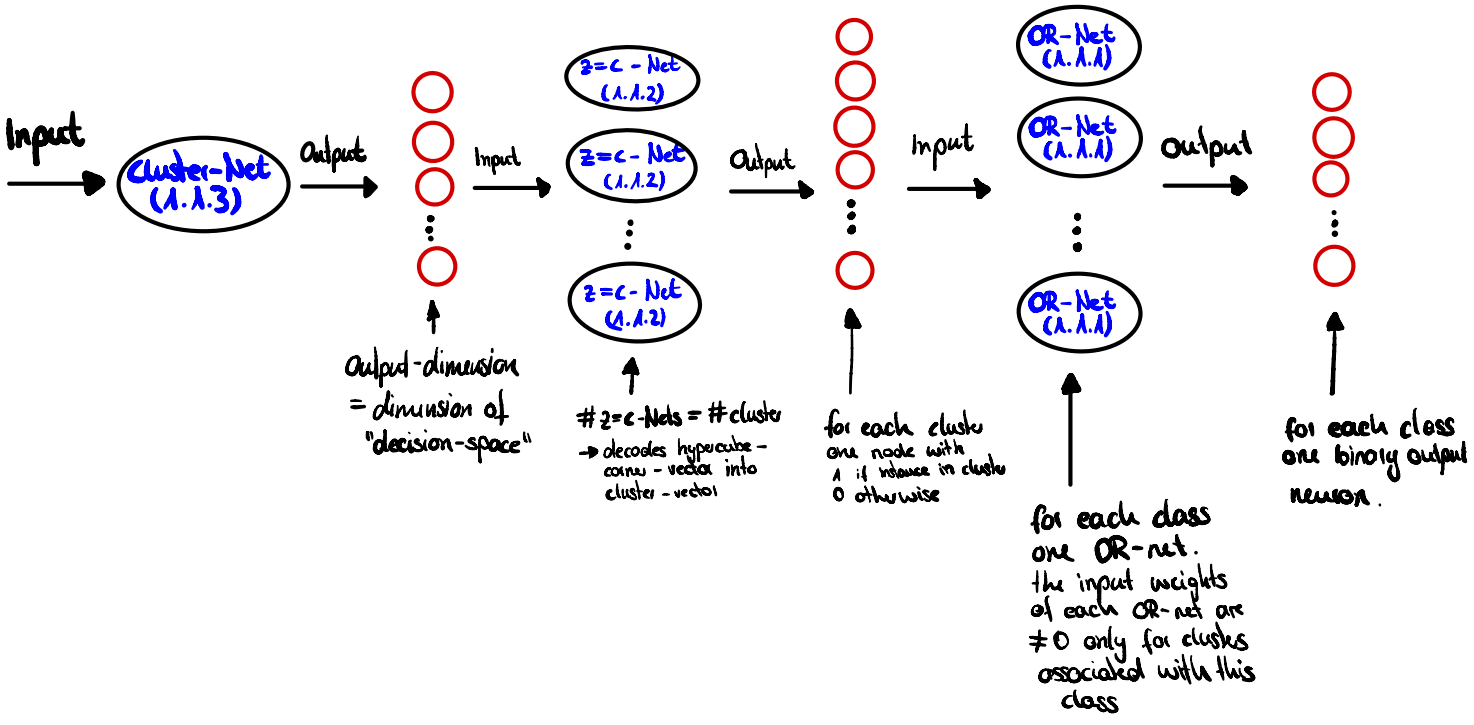



For an arbitrary data set the dimensionaly of the cube must be adapted to the number of clusters in the given input dimension.

Comment: The decision boundaries should be linear and the number of decision planes should match the dimension of the hypercube. The elementwise sign of the preactivations $\geq$ indicates on which side of each decision plane the input lies. Using a step function then maps the result to $\{0,1\}^m$

## 1.2 3-layer Universal Classifier

Comment: Same as sample solution, but the description of the problems of this 0-training-loss classifier is missing



## 2. Linear Activation Function

Comment: Essentially equal to the sample solution. The sample solution only used linear functions with scalar $M_\ell$, but this can be generalized to matrices, as you did.

$z_0 = X$, $\tilde{z}_\ell = z_{\ell-1} \cdot B_\ell + b_\ell$, $z_\ell = \Phi_\ell(\tilde{z}_\ell)$ with $\Phi_\ell$ a linear mapping.

Proof:

Let $\Phi_\ell$ be a linear mapping for every $\ell$.

First we absorbe the bias into the weight matrix:

$$\tilde{z}_\ell = z_{\ell-1} \cdot B_\ell + b_\ell = [1, z_{\ell-1}] \cdot [b_\ell, B_\ell] =: \overline{z_{\ell-1}} \cdot \overline{B_\ell}$$

Then we can replace $\Phi_\ell$ by its matrix representation:

$$\Phi_\ell(\tilde{z}_\ell) \doteq M_\ell \cdot \tilde{z}_\ell = M_\ell \, \overline{z_{\ell-1}} \, \overline{B_\ell}$$

Starting from the output layer we can write:

$$\Phi_L(\tilde{z}_L) \doteq M_L \, \overline{z_{L-1}} \, \overline{B_L} = M_L \cdot M_{L-1} \cdot \overline{z_{L-2}} \cdot \overline{B_{L-1}} \, \overline{B_L}$$

$$= \ldots = M_L \cdot \ldots \cdot M_1 \cdot \overline{z_0} \cdot \overline{B_1} \cdot \ldots \cdot \overline{B_L}$$

We now can define another linear mapping:

$$z_L = \Phi(\tilde{z}_0) = (\Phi_L \circ \Phi_{L-1} \circ \ldots \circ \Phi_1)(\tilde{z}_0)$$

$$\doteq M_L \cdot \ldots \cdot M_1 \, \tilde{z}_0$$

Which is a linear mapping because $\Phi_i$ $i \in \{1, \ldots, L\}$ are linear mappings. Furthermore we can define the weight matrix of the single layer to be:

$$\overline{B} = \overline{B_L} \cdot \ldots \cdot \overline{B_1}$$

Therefore one can reduce the Network to one single layer.