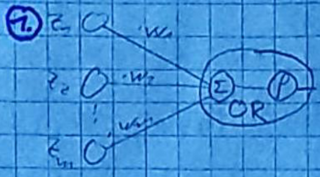
2 Classification Capacity

comment

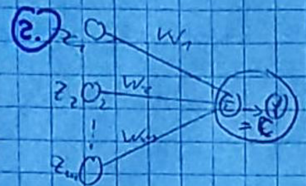
2.1 Simple Networks



$w_i = 1 \ \forall i \in \{1,..,m\}$, $f(\vec z) = \begin{cases} 0 & \text{if } \vec z \le 0 \\ 1 & \text{else} \end{cases}$

If one of the z_i is >0 the sum of all z_i is ~~bigger~~ >0.

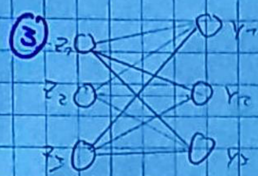
same as in sample solution



$w_i = 10^{i-1} \ \forall i \in \{1,..,n\}$, $f(\vec z) = \begin{cases} 0 & \text{if } \vec z - c \cdot w \neq 0 \\ 1 & \text{else} \end{cases}$

encode z as a decimal number and compute with c in same representation

different approaches but same result

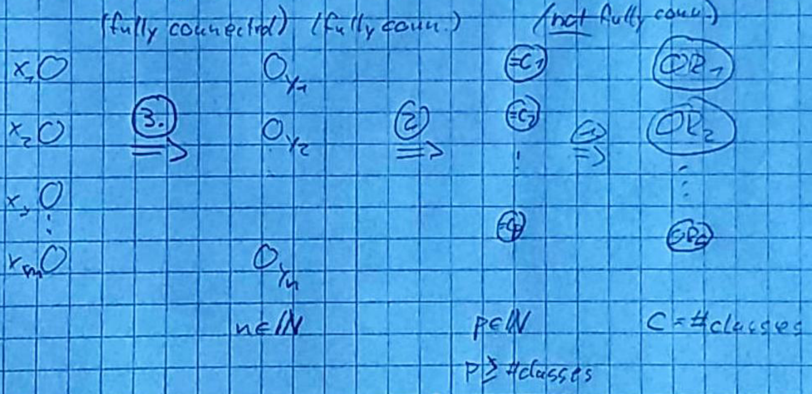


each y-neuron has weight vector $y_i \sim B_i$

$y_i = \text{step}(B_i z) = \begin{cases} 1 & B_i x \ge 1 \\ 0 & \text{else} \end{cases}$

used definition of the lecture

2.2 3-layer Universal Classifier



First the data is split, such that each input vector is projected on a corner of a hyper-cube (dim n), where in one corner ~~each~~ are only members of one class. the c_i ~~encode~~ ~~the~~ layer reads of the bit representation such that on c_i is always one and all c_j $j \neq i$ are 0. Since each corner only contains one class, but the members of one class can be projected onto different corners, we need the OR-layer, which will show the classification of the instances. Each OR neuron represents one class, and is only connected to the neurons of the previous layer, which represents a corner (of the hypercube) of that class. This way if the First layer is trained there will be perfect classification of a training set

Less explicit but same idea and order of gates

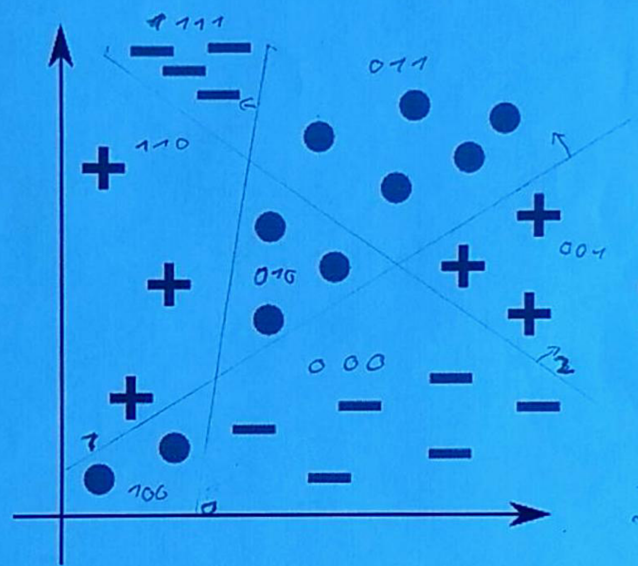
1. logical OR operation on a binary input vector $z \in \{0,1\}^m$

$$\text{i.e. } z \to f(z) = \begin{cases} 0 & \forall i:\ z_i = 0 \\ 1 & \text{otherwise} \end{cases}$$

2. for an arbitrary but fixed binary vector $c \in \{0,1\}^m$ map the input vector $z \in \{0,1\}^m$ to

$$z \to f(z) = \begin{cases} 1 & z = c \\ 0 & \text{otherwise} \end{cases}$$

3. for the dataset X, Y displayed in Figure 1 (with feature vectors X_i and associated classes $Y_i \in$ { red minus, blue plus, green circle }) map every X_i onto the corners of a hypercube $\{0,1\}^m$ such that each corner contains only one class (you can map one class to multiple corners, but not multiple classes onto one corner). The dimension m of the hypercube is not a priori fixed and may be adjusted to fit the dataset. Draw the decision boundaries of your network in Figure 1 (you do not need to specify the precise equations of these boundaries) and indicate for each region to which hypercube corner it will be mapped. How can this be generalized to arbitrary many input dimensions and arbitrary (non degenerate) class distributions?



hyper plains as boundaries

different decision boundaries but ~~same~~ still completely separated

index

Abbildung 1: Example dataset with three classes (red minus, blue plus, green circle). Sketch the decision boundaries of your network here and draw the matching labeled hypercube.

2.2 3-layer Universal Classifier (5 Points)

Combine the networks from above into a universal classifier which classifies an arbitrary training set with zero training error. Draw a sketch of the network and explain in a few sentences how it works. Do you see any problems with this zero training loss classifier?

3. Linear Activation Function

$$Z_L = \phi_L\left(B_L \cdot \tilde{Z}_{L-1}\right)$$
$$Z_L = \phi_L\left(B_L \cdot \phi_{L-1}\left(B_{L-1} \cdot \tilde{Z}_{L-2}\right)\right)$$

Let $f_l : \mathbb{R}^{H_{l-1}} \to \mathbb{R}^{H_l}$ be the function that calculates the pre-activations: $Z_{l-1} \mapsto \tilde{Z}_l = B_l Z_{l-1}$ which is obviously a linear function. So the total function reads

$$Z_L = \underbrace{\phi_L \circ f_L \circ \phi_{L-1} \circ f_{L-1} \circ \ldots \circ \phi_1 \circ f_1}_{=:\mathbf{F}}(Z_0)$$

As the composition between two linear functions is still a linear function, $\mathbf{F}$ is also a linear function which can be written as a composition of two arbitrary linear functions $\tilde{f}, \tilde{\phi}$ so that $\mathbf{F} = \tilde{\phi} \circ \tilde{f}$.

$$\Rightarrow Z_L = \tilde{\phi} \circ \tilde{f}(Z_0)$$

That means that a neural network with a linear activation function is equivalent to a 1-layer neural network.

The proof is essentially the same