آرین تشکر - ۴۰۰۲۳۴۹۴ تکلیف ششم درس شناسایی آماری الگو

6.1. (See the attached file "P6-1.png" for reference)

As we can see in the figure, the samples are not linearly classifiable. However, we can classify them using two linear decision boundaries and then AND-ing the decisions from each of them.

Taking a look at the samples we see that the lines:

$$x_2 = -x_1 + \frac{1}{2} \text{ and, } \quad (g_1(x) = x_2 + x_1 - \tfrac{1}{2} = 0)$$

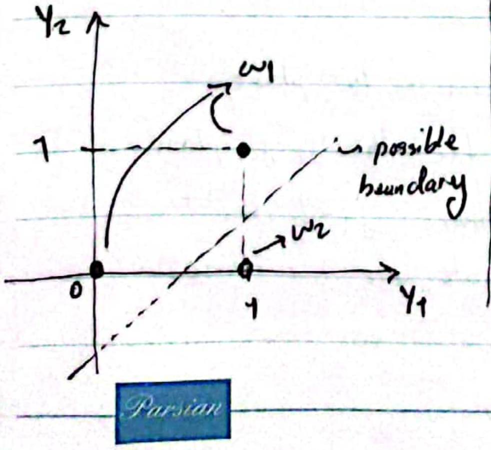
$$x_2 = -x_1 + \frac{3}{2}, \quad (g_2(x) = x_2 + x_1 - 3/2 = 0)$$

partition the (x_1, x_2) space into three subspaces, each of which houses exactly one class of the samples. Now, using the step function as the activation for our first layer neurons we can encode these subspaces into a new space (y_1, y_2):

$$y_1 = \begin{cases} 0, & g_1(x) < 0 \\ 1, & g_1(x) \geq 0 \end{cases}, \quad y_2 = \begin{cases} 0, & g_2(x) < 0 \\ 1, & g_2(x) \geq 0 \end{cases}$$

Obviously all possible combinations of (y_1, y_2) create 4 different points in the (y_1, y_2) space, however, because the chosen decision boundaries are parallel, only three points can exist. The points $(0,0)$ and $(1,1)$ encode ω_1 while $(1,0)$ encodes ω_2 (and $(0,1)$ simply does not exist).

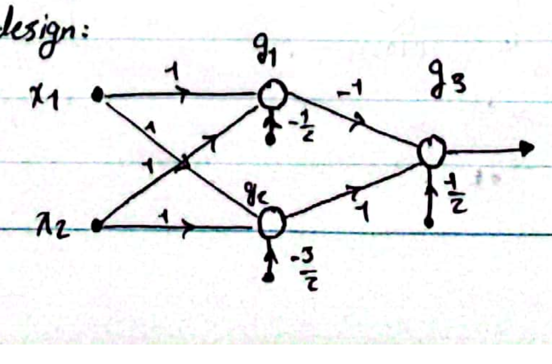
Now as the final step of our design, we are to linearly seperate the samples in this new (y_1, y_2) space. As shown in the figure below, the line



$$y_2 = y_1 - \frac{1}{2} \quad (g_3(y) = y_2 - y_1 + \tfrac{1}{2} = 0)$$

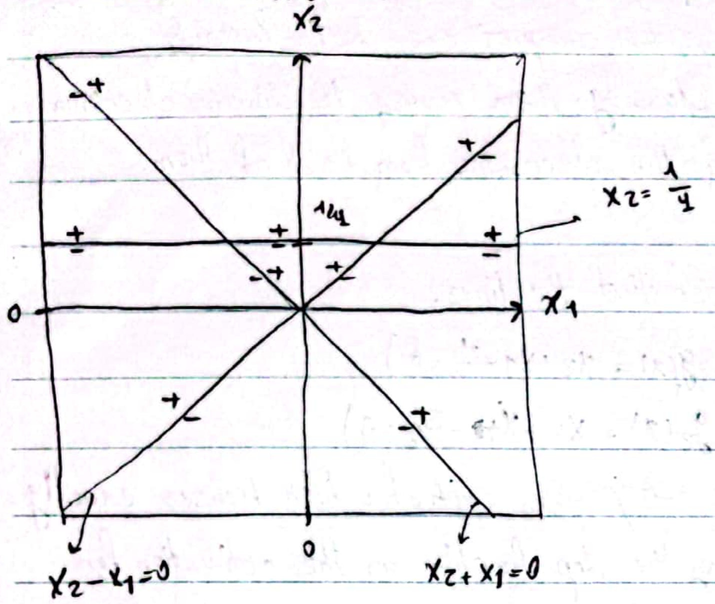
Can classify the samples in this new space.

Putting it all together we end up with the following design:



1.2. Refer to the MATLAB script.

1.3. We will first take a look at these lines in the two-dimensional space:

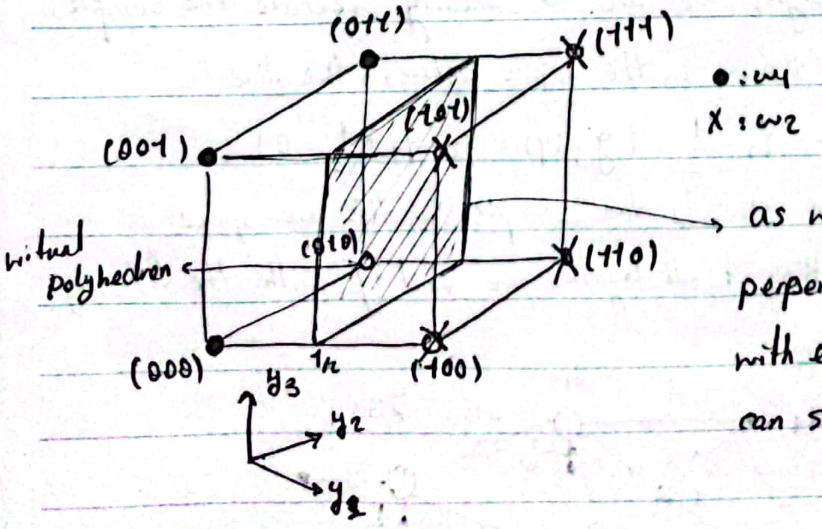


if we encode the positive side with (1) and the negative side with (0), these lines lie in a general position in the 2D space and create 7 real and 1 virtual polyhedra corresponding to vertices on a cube.

a) In order for a two-layer network to be sufficient for this problem the union of the regions for all classes must be connected. I.e. no union of the regions of the same class must be "disjointed" in the 2D space.

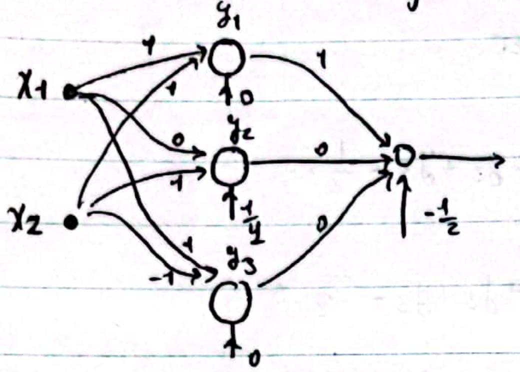
To give an easy example: $\begin{cases} \omega_1: (011) \cup (001) \cup (000) \\ \omega_2: (111) \cup (101) \cup (110) \cup (100) \end{cases}$

looking at the corresponding (hyper) cube we have:



as we can see this plane perpendicular to the y_1-y_2 plane with equation $y_1 - \frac{1}{2} = 0$ can separate the two classes.

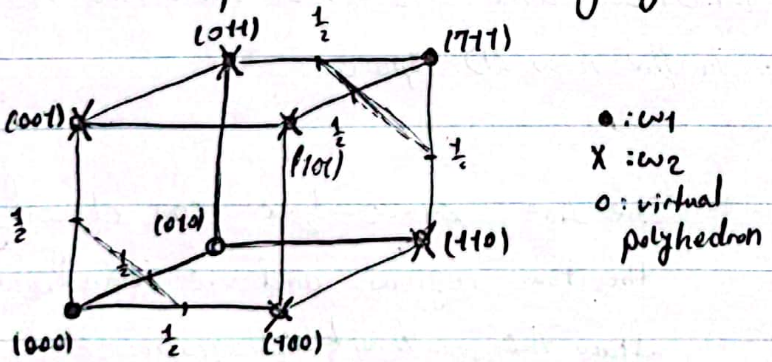
Therefore we have a two layer network where the first layer maps the points on the 2D-space to the 3D (y_1, y_2, y_3) space using the three lines given. And afterthat, a hyperplane that is capable of classifying the patterns in that space to return a final output:



b) This time in order for three layers to be "needed", we have to make it so that the union of regions for one of the classes is disjointed. To that end consider the partitioning:

$$\begin{cases} \omega_1: (000) \cup (111) \\ \omega_2: (011) \cup (001) \cup (101) \cup (110) \cup (100) \end{cases}$$

which corresponds to the following (hyper) cube:



If we recall from the solution the XOR problem using a two-layer network, here we need to employ a similar strategy. we will create three subspace inside the hypercube so that the entirety of the second class is mapped to a single point in the resulting 2D space from the two hyperplanes that partition the cube. Then we will perform the classification in the new 2D space and we are done. There are many hyperplanes capable of such partitioning but we will choose the planes described

, with the two sets of points below:

$$\Gamma_1 : \begin{cases} (\frac{1}{2},0,0) \\ (0,\frac{1}{2},0) \\ (0,0,\frac{1}{2}) \end{cases} , \quad \Gamma_2 : \begin{cases} (\frac{1}{2},1,1) \\ (1,\frac{1}{2},1) \\ (1,1,\frac{1}{2}) \end{cases}$$

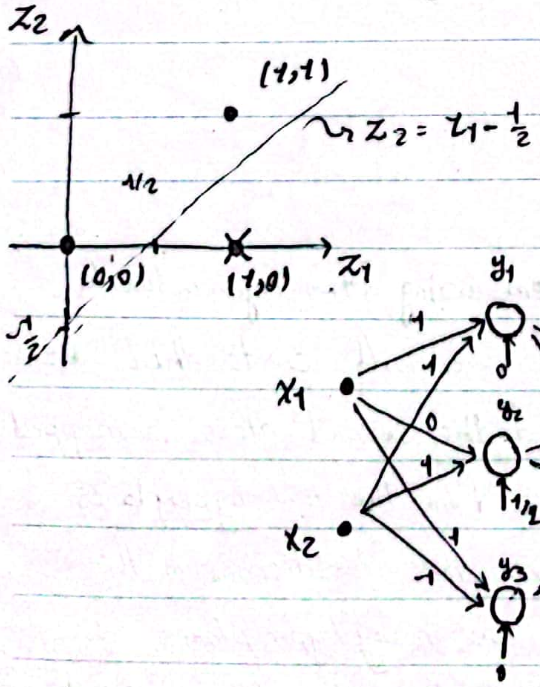
the equations for which are derived to be:

$$\Gamma_1 : \frac{1}{4}y_1 + \frac{1}{4}y_2 + \frac{1}{4}y_3 - \frac{1}{8} = 0 \quad \text{or} \quad y_1 + y_2 + y_3 - \frac{1}{2} = 0$$

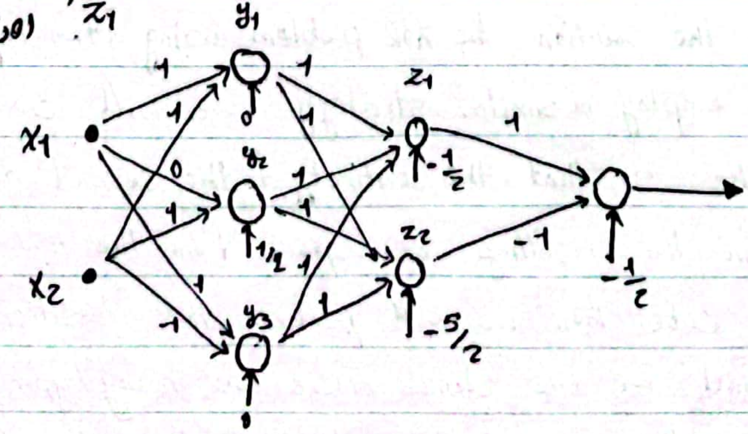
$$\Gamma_2 : \frac{1}{4}y_1 + \frac{1}{4}y_2 + \frac{1}{4}y_3 - \frac{5}{8} = 0 \quad \text{or} \quad y_1 + y_2 + y_3 - \frac{5}{2} = 0$$

Now if we partition the space based on the sign of points w.r.t. either of these planes we end up with three regions in a 2D space:

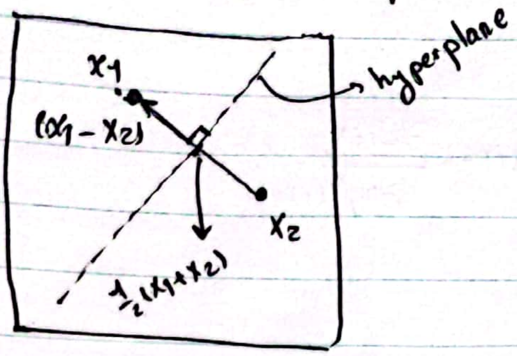
R_1: points below $y_1 + y_2 + y_3 - \frac{1}{2} = 0$ are mapped to $(0,0)$
R_2: points between the two planes are mapped to $(1,0)$
R_3: points above $y_1 + y_2 + y_3 - 5/2 = 0$ are mapped to $(1,1)$
we also know that $\omega_1 \in R_1 \cup R_3$ and $\omega_2 \in R_2$ therefore we have the following configuration in the new 2D space:



The line $z_1 - z_2 - \frac{1}{2} = 0$ can classify the two regions and we can finally draw the network diagram:



4.4. For the sake of simplicity and w.l.o.g. we will imagine the problem in a 2D setting as depicted below:



Since the hyperplane bisects the distance between x_1 and x_2, it must pass through $\frac{1}{2}(x_1+x_2)$. Moreover, the vector x_1-x_2 is normal to the hyperplane (positive side pointing to x_1). This means that for any point x on the hyperplane, the vector $(x-\frac{1}{2}(x_1+x_2))$ is normal to x_1-x_2 and hence results in a zero dot product:

$$(x_1-x_2)^T(x-\tfrac{1}{2}(x_1+x_2))=0$$

$$\Rightarrow (x_1-x_2)^T x-\tfrac{1}{2}(x_1-x_2)^T(x_1+x_2)=0$$

$$\Rightarrow (x_1-x_2)^T x-\tfrac{1}{2}\left[x_1^T x_1+\underbrace{x_1^T x_2-x_2^T x_1}-x_2^T x_2\right]=0$$

This is zero because x_1 and x_2 are vectors so $x_1^T x_2$ and $x_2^T x_1$ are the same scalar value

$$\Rightarrow \text{hyperplane}: (x_1-x_2)^T - \tfrac{1}{2}\|x_1\|^2+\tfrac{1}{2}\|x_2\|^2$$

4.5. a) $J=-\sum_{i=1}^{N}\sum_{k=1}^{K} y_k(i)\ln\frac{\hat{y}_k(i)}{y_k(i)}$ → takes its minimum when $\hat{y}_k(i)=y_k(i)$ (According to Gibbs' Inequality)

at that point, the cost function becomes: $J=-\sum_i\sum_k y_k(i)\underbrace{\ln 1}_{0}=0$

b) let's assume that $\hat{y}_k(i)$ is off from the desired response by $e_k(i)$. I.e.:

$\hat{y}_k(i)-y_k(i)=e_k(i)$ [$e_k(i)$ is the "error"]

Then we have:

$$J=-\sum_i\sum_k y_k(i)\ln\frac{y_k(i)+e_k(i)}{y_k(i)}=-\sum_i\sum_k y_k(i)\ln\left[1+\frac{e_k(i)}{y_k(i)}\right]$$

where the term $e_k(i)/y_k(i)$ is the error relative to the desired output.

(addendum to this in the last page)

4.6. From the definition $J = \sum_{i=1}^{N} \mathcal{E}(i)$ we can derive that

$$\mathcal{E}(i) = -\sum_{k=1}^{K_L} y_k(i) \ln(\hat{y}_k(i) / y_k(i))$$

again from the definition we have: $\delta_j^L(i) = \dfrac{\partial \mathcal{E}(i)}{\partial v_j^L(i)}$

and $v_j^L(i) = \sum_{k=1}^{K_{L-1}} w_{jk}^L y_k^{L-1}(i)$

$$\rightarrow \delta_j^L(i) = \partial / \partial v_j^L(i) \left[-\sum_{k=1}^{K_L} y_k(i) \ln \frac{f(v_k^L(i))}{y_k(i)} \right]$$

all of the terms are zeroed out w.r.t. $\partial v_j^L(i)$ except for $y_j(i) \ln \dfrac{f(v_j^L(i))}{y_j(i)}$

$$\rightarrow \delta_j^L(i) = -\partial / \partial v_j^L(i) \left[y_j(i) \ln \frac{f(v_j^L(i))}{y_j(i)} \right]$$

$$= -y_j(i) \frac{f'(v_j^L(i))}{f(v_j^L(i))}$$

If $f(\cdot)$ is the sigmoid function we know that $f'(\cdot) = a f(\cdot)(1 - f(\cdot))$

$$\Rightarrow \delta_j^L(i) = -y_j(i) \frac{a \hat{y}_j(i)(1 - \hat{y}_j(i))}{\hat{y}_j(i)} = a y_j(i)(1 - \hat{y}_j(i))$$

4.16. $\mu = \mu_0 \left[\dfrac{1}{1 + t/t_0} \right]$, if t_0 is large enough for small values of t, t/t_0 tends to zero and therefore $\mu = \mu_0$ is constant in the early stages, however as t grows and surpasses t_0, $\dfrac{1}{1 + t/t_0} = \dfrac{1}{t_0 + t / t_0} = t_0 / t_0 + t$ decreases in inverse proportion to "t" (note that t_0 is fixed). This is desirable because due to the nature of random initialization, it is likely that we are far away from the optimal answer in the early stages of optimization and therefore we would like to take bigger steps towards our goal. However as we get closer to the optimum, we should slowly decrease the size of our steps to be able fine-tune the response of the network and more accurately navigate the surface area of the cost function in search of the closest possible answer to the actual optimum. If unbound by precision errors, given a well-formed cost function (like BCE), this learning rate annealing scheme can likely achieve any degree of accuracy.

Addendum to 4.5 (a):

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_k(i) \ln \frac{\hat{y}_k(i)}{y_k(i)}$$

$$\rightarrow \frac{\partial J}{\partial \hat{y}_k} = \frac{\partial}{\partial \hat{y}_k}\left[-\sum_i \sum_k y_k(i)\left[\ln \hat{y}_k(i) - \ln y_k(i)\right]\right]$$

$$= \frac{\partial}{\partial \hat{y}_k}\left[-\sum_i y_k(i)\left[\ln \hat{y}_k(i) - \ln y_k(i)\right]\right]$$

$$= -\sum_i \frac{y_k(i)}{\hat{y}_k(i)} \rightarrow 0$$

$\rightarrow$ where $y_k(i) = 0$ the summand is 0 and where $y_k(i)=1$ the summand gets smaller the greater $\hat{y}_k(i)$ is. However since $\hat{y}_k(i)$ is interpreted as a probability value it cannot be greater than one so at $y_k(i) = 1 \rightarrow \hat{y}_k(i) = 1$.