Homework # 4.

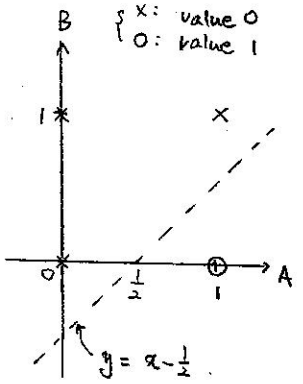
1. Design a perceptron that implements the boolean function A AND (NOT B). Design also a two-layer network that implements A XOR B (A exclusive OR B).

sol>

[A AND (NOT B)]

B \ A	0	1
0	0	1
1	0	0

$\Rightarrow$



Firstly, let's implement a boolean classifier which classifies a given point (x, y) into two categoies

— $C_0 = \{(x,y) \mid y > x - \frac{1}{2}\}$ and

$C_1 = \{(x,y) \mid y < x - \frac{1}{2}\}$

Let $\vec{x} = [1, x, y]^T$: a given point.

Find a linear discriminant function $g(\vec{x})$

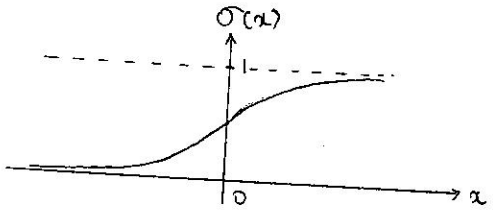
s.t. $\begin{cases} g(\vec{x}) < 0 & \text{if } \vec{x} \in C_0 \\ g(\vec{x}) > 0 & \text{if } \vec{x} \in C_1 \end{cases}$

Trivially,

$$g(\vec{x}) = x - y - \frac{1}{2}$$

$$= \left[-\frac{1}{2}, 1, -1\right] \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}$$

Now, consider a sigmoid unit functic



$$\sigma(x) = \frac{1}{1+e^{-x}} = 0.99$$

$$\therefore x = \ln 99 = 4.5951$$

$\Rightarrow$ Let's adjust $g(\vec{x})|_{A=1,B=0} > \ln 99 = 4.59$

$\vec{x} = [1, 1, 0]^T$.

$$k\,g(\vec{x}) = k\left[-\frac{1}{2}, 1, -1\right] \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= k\left(-\frac{1}{2} + 1\right) = \frac{k}{2} > 4.5951$$

Take $k = 10$

$\therefore$ Our perceptron has the output

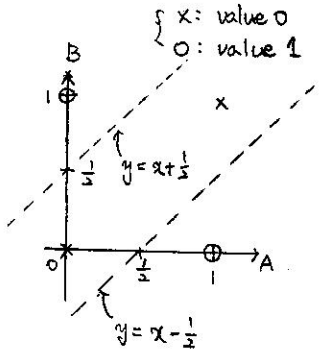
$$\sigma(\vec{x}) = \frac{1}{1 + e^{-(10[-\frac{1}{2}, 1, -1] \cdot \vec{x})}}$$

i.e., $$\sigma(A,B) = \frac{1}{1 + \exp(-10A + 10B + 5)}$$

[A XOR B]

B \ A	0	1
0	0	1
1	1	0

⇒



Firstly, implement a boolean classifier which classifies a given point (x, y) into two categories

— $\begin{cases} C_0 = \{ (x,y) \mid y > x + \frac{1}{2} \} \\ C_1 = \{ (x,y) \mid y < x + \frac{1}{2} \} \end{cases}$

Let $\vec{x} = [1, x, y]^T$: a given point.

⇒ linear discriminant function

$$g_1(\vec{x}) = -x + y - \frac{1}{2}$$

$$\left(\begin{array}{l} \text{if } g_1(\vec{x}) > 0 \rightarrow \vec{x} \in C_0 \\ \text{if } g_1(\vec{x}) < 0 \rightarrow \vec{x} \in C_1 \end{array} \right)$$

Secondly, implement a boolean classifier which classifies a given point (x, y) into two categories

— $\begin{cases} C_2 = \{ (x,y) \mid y > x - \frac{1}{2} \} \\ C_3 = \{ (x,y) \mid y < x - \frac{1}{2} \} \end{cases}$

⇒ linear discriminant function

$$g_2(\vec{x}) = x - y - \frac{1}{2}$$

$$\left(\begin{array}{l} \text{if } g_2(\vec{x}) < 0 \rightarrow \vec{x} \in C_2 \\ \text{if } g_2(\vec{x}) > 0 \rightarrow \vec{x} \in C_3 \end{array} \right)$$

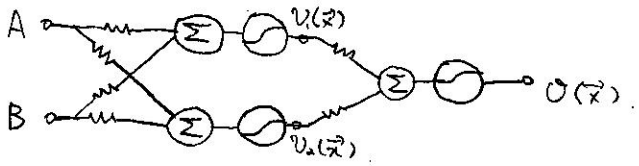
B \ A	0	1
0	$g_1 < 0$, $g_2 < 0$	$g_1 < 0$, $g_2 > 0$
1	$g_1 > 0$, $g_2 < 0$	$g_1 < 0$, $g_2 < 0$

Let $\text{step}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$.

Then,

A XOR B = $\text{step}(g_1) + \text{step}(g_2)$

⇒



$$\begin{cases} v_1(\vec{x}) = \dfrac{1}{1 + \exp(-k_1 g_1(\vec{x}))} & (k_1 > 0) \\ v_2(\vec{x}) = \dfrac{1}{1 + \exp(-k_2 g_2(\vec{x}))} & (k_2 > 0) \\ o(\vec{x}) = \dfrac{1}{1 + \exp\left(-k_3 \left(v_1(\vec{x}) + v_2(\vec{x}) - \frac{1}{2}\right)\right)} & (k_3 > 0) \end{cases}$$

Determine k_1

: for $\vec{x} = [1, 0, 1]^T$, $v_1(\vec{x}) > 0.99$

$$k_1 > 2 \ln 99 = 9.1902$$

Take $k_1 = 10$.

Determine k_2

: for $\vec{x} = [1, 1, 0]^T$, $v_2(\vec{x}) > 0.99$

$$k_2 > 2 \ln 99 = 9.1902$$

Take $k_2 = 10$.

Determine k_3

: for $\vec{x} = [1, 0, 1]^T$, $o(\vec{x}) > 0.99$

$$\Theta(\vec{x}) = \frac{1}{1+\exp(-k_3(v_1(\vec{x})+v_2(\vec{x})-\frac{1}{2}))} > 0.99$$

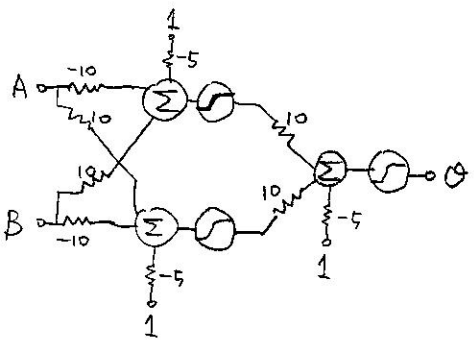
$$-k_3(v_1(\vec{x})+v_2(\vec{x})-\tfrac{1}{2}) < \ln\tfrac{1}{99}$$

$$k_3 > \frac{\ln 99}{v_1(\vec{x})+v_2(\vec{x})-\frac{1}{2}} > \frac{\ln 99}{1+0.01-\frac{1}{2}}$$

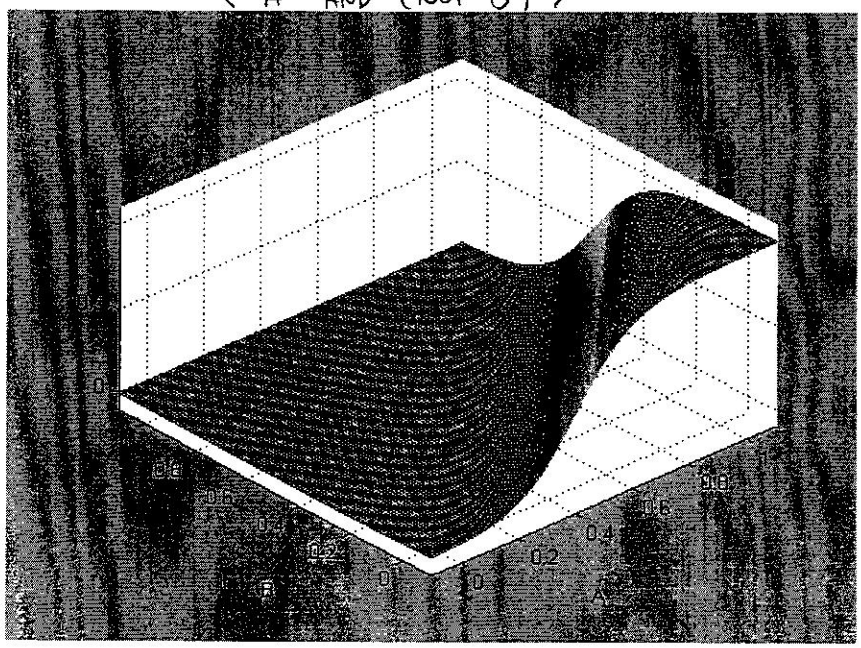
$$= 9.0100$$

Take $k_3 = 10$.

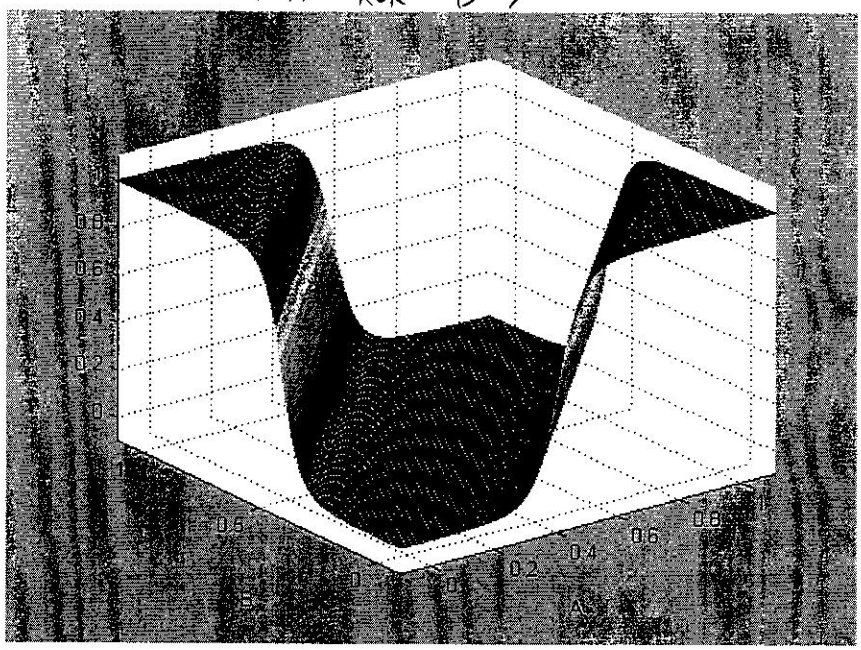
∴ Our network is



< A AND (NOT B) >



< A XOR B >



2. Let us consider multilayer perceptrons (MLPs) with one hidden layer in which the number of units of the input, hidden and output layers are given by N, M, and L respectively.

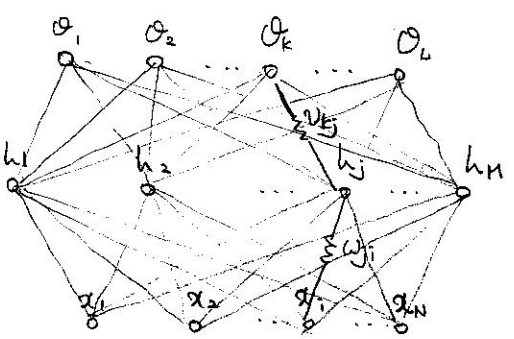
For the training of MLPs, we consider the conjugate gradient method with the weight elimination method to avoid the over-fitting problem.

So the learning algorithm should get the inputs from the structure parameters of MLPs (L, M, and N), the parameters of weight elimination method (λ and ω_0) and also the training patterns.

The output of the learning algorithm will be the weight parameters of MLPs.

Describe the learning algorithm of the suggested MLPs in detail: for each step, you need the describe the equations in exact form.

Sol> [Network Structure]



Let

$$R(\vec{v},\vec{w}) = E_d(\vec{v},\vec{w}) + \lambda E_c(\vec{v},\vec{w})$$

where $E_d(\vec{v},\vec{w}) = \frac{1}{2}\sum_{k=1}^{L}(t_k-O_k)^2$,

$$E_c(\vec{v},\vec{w}) = \sum_{k=1}^{L}\sum_{j=0}^{M}\frac{(v_{kj}/\omega_0)^2}{1+(v_{kj}/\omega_0)^2} + \sum_{j=1}^{M}\sum_{i=0}^{N}\frac{(\omega_{ji}/\omega_0)^2}{1+(\omega_{ji}/\omega_0)^2}$$

(* v_{k0}'s and ω_{j0}'s are weights for bias)

Find $\frac{\partial E_d}{\partial v_{kj}}$, $\frac{\partial E_d}{\partial \omega_{ji}}$, $\frac{\partial E_c}{\partial v_{kj}}$, $\frac{\partial E_d}{\partial \omega_{ji}}$

(i) $\frac{\partial E_d}{\partial v_{kj}}$

$$\frac{\partial E_d}{\partial v_{kj}} = \frac{\partial}{\partial v_{kj}}\frac{1}{2}(t_k-O_k)^2 = (t_k-O_k)\left(-\frac{\partial O_k}{\partial v_{kj}}\right)$$

$$= -(t_k-O_k)O_k(1-O_k)\frac{\partial}{\partial v_{kj}}\left(\sum_{j=0}^{M} v_{kj}h_j\right)$$

$$= -(t_k-O_k)O_k(1-O_k)h_j$$

$$= -\delta_k h_j$$

where $\delta_k = (t_k-O_k)O_k(1-O_k)$

(ii) $\frac{\partial E_d}{\partial \omega_{ji}}$

$$\frac{\partial E_d}{\partial \omega_{ji}} = \frac{\partial}{\partial \omega_{ji}}\frac{1}{2}\sum_{k=1}^{L}(t_k-O_k)^2 = \sum_{k=1}^{L}\frac{\partial}{\partial \omega_{ji}}\frac{1}{2}(t_k-O_k)^2$$

$$= \sum_{k=1}^{L}\left(\frac{\partial}{\partial h_j}\frac{1}{2}(t_k-O_k)^2\right)\frac{\partial h_j}{\partial \omega_{ji}}$$

$$= \sum_{k=1}^{L} -(t_k-O_k)O_k(1-O_k)\frac{\partial}{\partial h_j}\left(\sum_{j=0}^{M} v_{kj}h_j\right)\frac{\partial h_j}{\partial \omega_{ji}}$$

$$= \sum_{k=1}^{L} -\delta_k \cdot v_{kj}\frac{\partial h_j}{\partial \omega_{ji}} = \sum_{k=1}^{L} -\delta_k \cdot v_{kj}\cdot h_j(1-h_j)\alpha_i$$

$$= -\delta_j'\alpha_i$$

where $\delta_j' = h_j(1-h_j)\sum_{k=1}^{L}\delta_k v_{kj}$

(iii) $\frac{\partial E_c}{\partial v_{kj}}$

$$\frac{\partial E_c}{\partial v_{kj}} = \frac{\partial}{\partial v_{kj}} \frac{(v_{kj}/\omega_0)^2}{1+(v_{kj}/\omega_0)^2}$$

$$= \frac{\left(\frac{2}{\omega_0^2}v_{kj}\right)\left(1+\left(\frac{v_{kj}}{\omega_0}\right)^2\right) - (v_{kj}/\omega_0)^2\left(\frac{2}{\omega_0^2}v_{kj}\right)}{\left[1+(v_{kj}/\omega_0)^2\right]^2}$$

$$= \frac{\frac{2v_{kj}}{\omega_0^2}}{\left[1+(v_{kj}/\omega_0)^2\right]^2}$$

(iv) $\frac{\partial E_c}{\partial \omega_{ji}}$

$$\frac{\partial E_c}{\partial \omega_{ji}} = \frac{\partial}{\partial \omega_{ji}} \frac{(\omega_{ji}/\omega_0)^2}{1+(\omega_{ji}/\omega_0)^2} = \frac{\frac{2\omega_{ji}}{\omega_0^2}}{\left[1+(\omega_{ji}/\omega_0)^2\right]^2}$$

$$\Rightarrow \begin{cases} \frac{\partial R}{\partial v_{kj}} = \frac{\partial E_d}{\partial v_{kj}} + \frac{\partial E_c}{\partial v_{kj}} \\ \frac{\partial R}{\partial \omega_{ji}} = \frac{\partial E_d}{\partial \omega_{ji}} + \frac{\partial E_c}{\partial \omega_{ji}} \end{cases}$$

Now apply $R(\vec{v},\vec{w})$ to Conjugate Gradient Algorithm for MLP.

[Algorithm]

Step 1. Determine L, M, N, λ and ω_0.

And get training samples $\{(\vec{x}_d, \vec{t}_d)\}_{d=1}^{n}$

Step 2. Set $k=0$, initialize $\vec{v}_0, \vec{w}_0$

and compute $\begin{cases} \nabla_{v\_k} = \frac{\partial R}{\partial \vec{v}}\Big|_{\vec{v}=\vec{v}_k} \text{ at } (\vec{x}_d, \vec{t}_d) \\ \nabla_{w\_k} = \frac{\partial R}{\partial \vec{w}}\Big|_{\vec{w}=\vec{w}_k} \end{cases}$

(let $d = (k \bmod n)+1$)

Step 3. Set $\vec{g}_k = \nabla_{v\_k}$ and $\vec{d}_k = -\vec{g}_k$.

Set $\vec{f}_k = \nabla_{w\_k}$ and $\vec{e}_k = -\vec{f}_k$

Step 4. Find α_k that minimizes $R[\vec{v}_k + \alpha_k \vec{d}_k]$

and α_k that minimizes $R[\vec{w}_k + \alpha_k \vec{e}_k]$

Step 5. Update the weight parameters:

$\vec{v}_{k+1} = \vec{v}_k + \alpha_k \vec{d}_k$.

$\vec{w}_{k+1} = \vec{w}_k + \alpha_k \vec{e}_k$.

Step 6. Set $\vec{g}_{k+1} = \nabla_{v\_k+1}\Big|_{\text{at } (\vec{x}_{d+1}, \vec{t}_{d+1})}$ and $\beta_k = \frac{\vec{g}_{k+1}^{\,T} \cdot \vec{g}_{k+1}}{\vec{g}_k^{\,T} \cdot \vec{g}_k}$

Set $\vec{f}_{k+1} = \nabla_{w\_k+1}\Big|_{\text{at } (\vec{x}_{d+1}, \vec{t}_{d+1})}$ and $b_k = \frac{\vec{f}_{k+1}^{\,T} \cdot \vec{f}_{k+1}}{\vec{f}_k^{\,T} \cdot \vec{f}_k}$

Step 7. Set $\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k$.

Set $\vec{e}_{k+1} = -\vec{f}_{k+1} + b_k \vec{e}_k$.

Step 8. If satisfied stop.

Otherwise, $k \leftarrow k+1$ and go to Step 4.

▨

3. In the probability density estimation method, suggest a possible candidate of window functions other than the original window function given by Parzen. Justify your answer :

You have to show that the convergence conditions are satisfied with your window function.

sol>

Let

$$\psi(\vec{x}) = \begin{cases} \frac{1}{\sqrt{\pi}} e^{-x_i^2} & \text{if } |x_i| \le \frac{1}{2},\ i = 2, \cdots, d \\ 0 & \text{otherwise.} \end{cases}$$

For the convergence of $p_n(\vec{x})$,

$\psi(\vec{x})$ must satisfy

$$\max_{\vec{x}} \psi(\vec{x}) < \infty \quad \text{and} \quad \lim_{\|\vec{x}\| \to \infty} \psi(\vec{x}) \prod_{i=1}^{d} x_i = 0$$

(i) $\max_{\vec{x}} \psi(\vec{x}) = \frac{1}{\sqrt{\pi}} < \infty$

$\therefore$ satisfied

(ii)

$$\psi(\vec{x}) \prod_{i=1}^{d} x_i \longrightarrow 0 \quad \text{as} \quad x_i^2 \to \infty \quad i = 2, \cdots, d.$$

$$\lim_{x_i^2 \to \infty} \psi(\vec{x}) \prod_{i=1}^{d} x_i = \lim_{x_i^2 \to \infty} \frac{1}{\sqrt{\pi}} e^{-x_i^2} \cdot x_1 \cdot \prod_{i=2}^{d} x_i$$

$$= \frac{1}{\sqrt{\pi}} \prod_{i=2}^{d} x_i \lim_{x_i^2 \to \infty} \frac{1}{\sqrt{\pi}} e^{-x_i^2} \cdot x_1 = 0$$

$\therefore$ satisfied.

4. In the training of radial basis function networks (RBFNs), the centers of basis functions can greatly influence the performance of network. Describe the possible methods (at least two) to determin the centers of basis functions. Explain the advantages and disadvantag of the described methods also.

sol>

One approach is to allocate the centers of basis functions uniformly throughout the instance space X

This approach is very simple and efficient. But the centers are located non-optimal positions so we can't expect stable performances.

Another approach is using genetic algorithm for selecting the centers of basis functions.

This approach is very likely to obtain the optimal distribution of the centers. But it requires exhaustive search of the centers.