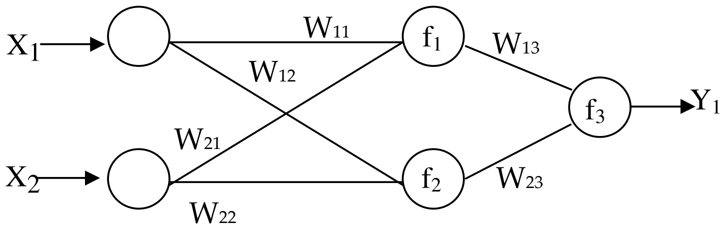
PERCEPTRON

1. In 1957, Rosenblatt and several other researchers developed perceptron, which used the similar network as proposed by McCulloch, and the learning rule for training network to solve pattern recognition problem.

   (*) But,this model was later criticized by Minsky who proved that it cannot solve the XOR problem.

2. The network structure:



3. The training process:

   ⑴ Choose the network layer,nodes,& connections

   ⑵ Randomly assign weights: W_{ij} & bias: θ_j

   ⑶ Input training sets X_i (preparing T_j for verification)

   Training computation:

$$net_j = \sum_i W_{ij} X_i - \theta_j$$

$$Y_j = \begin{cases} 1 & \text{if } net_j > 0 \\ 0 & net_j \le 0 \end{cases}$$

If $(T_j - Y_j) \neq 0$, than:

$$\Delta W_{ij} = \eta (T_j - Y_j) X_i$$

$$\Delta \theta_j = -\eta (T_j - Y_j)$$

⑷ Update weights and bias :

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

New $\theta_j = \theta_j + \Delta \theta_j$

⁀ repeat ‖ ~ ⌒ until every input pattern is satisfied

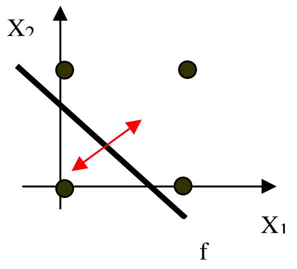
(\

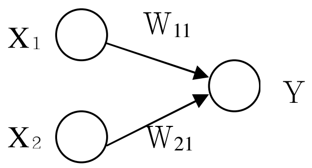
(7) Recall : after the network has trained as mentioned above, any input vector X can be send into the perceptron network. The trained weights, W_{ij}, and the bias, $\theta_j \rightarrow$, is used to derive $net_j \rightarrow$ and, therefore, the output Y_j can be obtained for pattern recognition.

Ex: Solving the OR problem

Let the training patterns are used as follow.

X_1	X_2	T
0	0	0
0	1	1
1	0	1
1	1	1





Let W_{11}=1, W_{21}=0.5, Θ=0.5

The initial net function is:

$$net = W_{11} \cdot X_1 + W_{21} \cdot X_2 - \Theta \qquad \eta = 0.1$$

$$net = 1 \cdot X_{11} + 0.5 \cdot X_{21} - 0.5$$

Feed the input pattern into network one by one

(0,0),	net= -0.5,	Y=0,	$\delta = 0$	O.K.
(0,1),	net= 0,	Y= 0,	δ =1- Ø=1	(need to update weight)
(1,0)	net= 0.5,	Y=1,	$\delta = 0$	O.K.
(1,1)	net= 1,	Y=1,	$\delta = 0$	O.K.

update weights for pattern (0,1) which is not satisfying the expected output:

ΔW_{11}=(0.1)(1)(0)= 0, ΔW_{12}=(0,1)(1)(1)= 0.1, $\Delta\Theta$=-(0.1)(1)=-0.1

W_{11}= $W_{11+}\Delta W_{11=}$1, W_{21}=0.5+0.1=0.6, Θ=0.5-0.1=0.4

Applying new weights to the net function:

$$net = 1 \cdot X_1 + 0.6 \cdot X_2 - 0.4$$

Verify the pattern (0,1) to see if it satisfies the expected output.

(0,1), net= 0.2, Y= 1, δ =Ø

Feed the next input pattern, again, one by one

(1,0), net= 0.6, Y=1, $\delta = \emptyset$

(1,1) , net= 1.2, Y=1, $\delta = \emptyset$

Since the first pattern(0,0) has not been testified with the new weights, feed again.

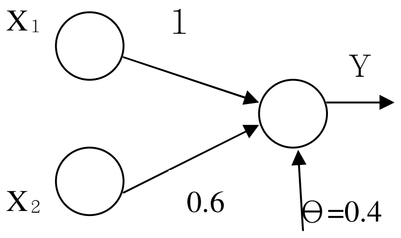
(0,0), net=-0.4, Y=∅, $\delta = \emptyset$

Now, all the patterns are satisfied the expected output. Hence, the network is successfully trained for understanding the OR problem(pattern).

We can generate the pattern recognition function for OR pattern is:

$\boxed{\textbf{net} = X_1 + 0.6X_2 - 0.4}$ (This is not the only solution, other solutions are possible.)

The trained network is formed as follow:



Recall process:

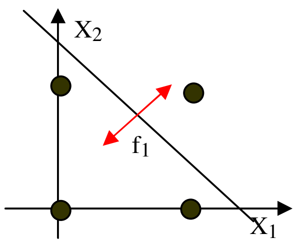
Once the network is trained, we can apply any two element vectors as a pattern and feed the pattern into the network for recognition. For example, we can feed (1,0) into to the network

(1,0), net= 0.6, Y=1 Therefore, this pattern is recognized as <u>1</u>.

Ex: Solving the AND problem (i.e., recognize the AND pattern)

Let the training patterns are used as follow.

X_1	X_2	T
0	0	0
0	1	0
1	0	0
1	1	1



Let $W_{11}=0.5$, $W_{21}=0.5$, $\Theta=1$, Let $\eta=0.1$

The initial net function is:

net $=0.5X_{11}+0.5X_{21}-1$

Feed the input pattern into network one by one

(0,0),	net=-1	Y=Ø,	δ = Ø
(0,1),	net=-0.5	Y= Ø,	δ =Ø
(1,0)	net=- 0.5,	Y= Ø,	δ = Ø
(1,1)	net= Ø,	Y= Ø,	δ = 1

update weights for pattern (1,1) which does not satisfying the expected output:

$\Delta W_{11}=(0,1)(1)(1)=0.1$, $\Delta W_{21}=(0,1)(1)(1)=0.1$,

$\Delta\Theta=-(0.1)(1)=-0.1$

$W_{11}=0.6$, $W_{21}=0.5+0.1=0.6$, $\Theta=1-0.1=0.9$

Applying new weights to the net function:

net$=0.6X_1+0.6X_2-0.9$

Verify the pattern (1,1) to see if it satisfies the expected output.

(1,1)	net= 0.3,	Y= 1,	δ = Ø

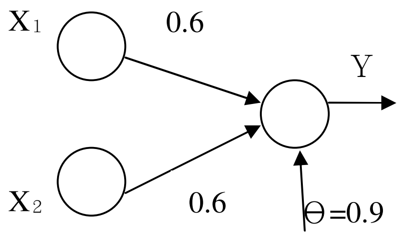
Since the previous patterns are not testified with the new weights, feed them again.

(0,0),	net=-0.9	Y=Ø,	δ = Ø
(0,1),	net=-0.3	Y= Ø,	δ =Ø
(1,0)	net=- 0.3,	Y= Ø,	δ = Ø

We can generate the pattern recognition function for OR pattern is:

$\boxed{\textbf{net}=0.6X_1+0.6X_2-0.9}$ (This is not the only solution, other solutions are possible.)

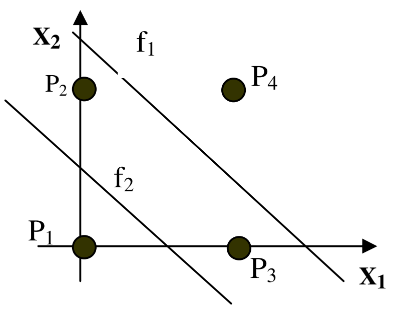
The trained network is formed as follow:

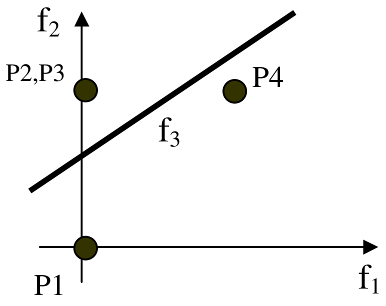


Ex: Solving the XOR problem

Let the training patterns are used as follow.

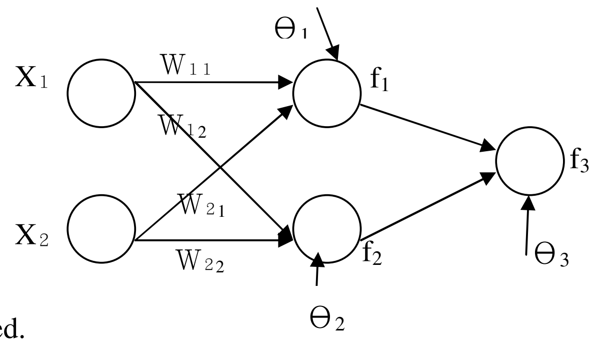
X_1	X_2	T
0	0	0
0	1	1
1	0	1
1	1	0





Let W_{11}=1.0, W_{21}= -1.0, Θ=0,

If we choose one layer network, it will be proved that the network cannot be converged. This is because the XOR problem is a non-linear problem, i.e., one single linear function is not enough to recognize the pattern. Therefore, the solution is to add one hidden layer for extra functions. The following pattern is formed.



X_1	X_2	T_1	T_2	T_3
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Let W_{11}=0.3, W_{21}=0.3, W_{12}= 1, W_{22}= 1

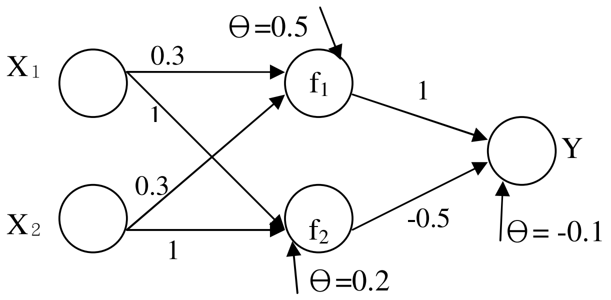
The initial net function for node f1 and node f2 are:

$f_1 = 0.3X_{11} + 0.3X_{21} - 0.5$

$f_2 = 1 \cdot X_{12} + 1 \cdot X_{22} - 0.2$

Now we need to feed the input one by one for training the network.for f1 and f2 seprearately. This is to satisfiying the expected output for f1 using T1 and for f2 using T2. Finally, we use f1 and f2 as input pattern to train the node $f_{3,}$ the result is

$f_3 = 1 \cdot X_{13} - 0.5X_{23} + 0.1$



Θ=0.5
X_1
0.3
f_1
1
1
Y
0.3
X_2
f_2
-0.5
Θ= -0.1
1
Θ=0.2