İlk Yapay Sinir Ağları

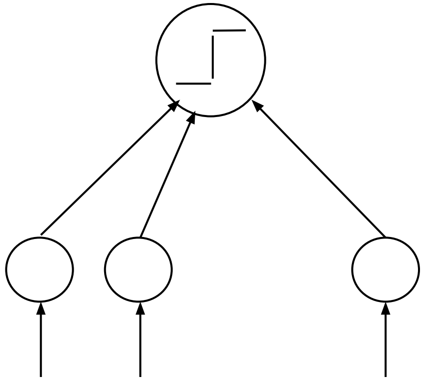
Pitts & McCulloch (1943)

- Biyolojik nöronların ilk matematiksel modeli
- Bütün boolean işlemleri nöron benzeri düğümler ile gerçeklenebilir (farklı eşik ve tahrik edici/dizginleyici bağlantılar ile)
- Genel amaçlı hesaplama cihazı için Von Neumann modeline rakip
- Otomata teorisinin orijini

Hebb (1949)

- Öğrenim için Hebbian kuralı : her ne zaman i ve j aktive edilirse; i ve j nöronları arasındaki bağlantı gücü artar.
- Veya her ne zaman eşzamanlı olarak i ve j nöronlarının her ikisi birden ON veya OFF yapılırsa i ve j düğümleri arasındaki bağlantı gücü artar.

İlk yükseliş (50’s – early 60’s)



Rosenblatt (1958)

- <u>Perceptron</u>: örüntü sınıflandırma için eşik düğümlerinin ağı

  <u>Perceptron öğrenim kuralı</u>

- <u>Perceptron convergence teoremi</u>: perceptron ile sunulabilen her şey öğrenebilir

Widrow and Hoff (1960, 1962)

- gradient descent (eğim düşümü) tabanlı öğrenim kuralı

Minsky, Pitts/McCulloch birimleri ile genel amaçlı bir makine icat etmeye girişmiştir.

Gerileme (60 ortaları – 70 sonları)

- Perceptron model ile ilgili ciddi problemler ortaya çıkmıştır (Minsky’s book 1969)
  - Single layer perceptrons; XOR gibi basit fonksiyonları sunamaz (öğrenemez)
  - Çok katmanlı doğrusal olmayan birimler daha büyük güce sahip olabilir fakat böylesi ağlar için öğrenim kuralı yoktur
  - Ölçekleme problemi: bağlantı ağırlıkları sonsuz olarak büyüyebilir
- İlk iki problemin 80’li yıllardaki çabalarla üstesinden gelinmiştir, fakat ölçekleme problemi hala devam etmektedir.
- Rosenblatt’ın ölümü (1964)
- Von Neumann machine ve AI çekişiyor

Yeniden büyük ilgi (80li yıllar ve sonrası)

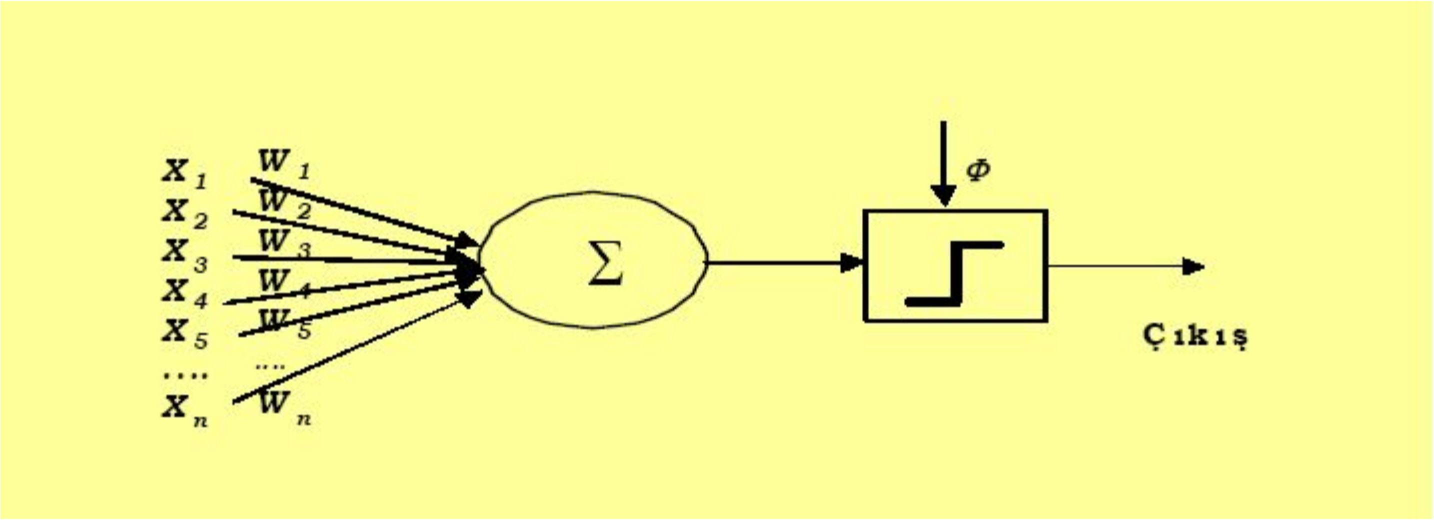
- Yeni teknikler
  - Çok katmanlı feed forward ağlar için Back-propagation öğrenim (with non-linear, differentiable node functions)
  - Termodinamik modeller (Hopfield net, Boltzmann machine, etc.)
  - Denetimsiz öğrenim
- Etkileyici uygulamalar (karakter tanıma, ses tanıma, metinden sese dönüşüm, proses kontrol, ortak bellek, vs)
- Geleneksel yaklaşımlar zor işlerle yeniden mücadele etmeye başlamıştır
- Uyarı:
  - Zorluklar ve limitler tahmin edilemeyebilir
  - Çözümlerden daha çok problemler ortaya çıkmaktadır

Tek Katmanlı Algılayıcılar

- Tek katmanlı algılayıcılar, sadece girdi ve çıktı katmanlarından oluşur.

- Ağın çıktısı, ağırlıklandırılmış girdi değerlerinin eşik değeri ile toplanması sonucu bulunur.

- Bu girdi değeri bir aktivasyon fonksiyonundan geçirilerek ağın çıktısı hesaplanır.



Basit bir perceptron

- Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusaldır.

$$Çıkış = f(\Sigma\, w_i x_i + \Phi)$$

- Ağa gösterilen örnekler iki sınıf arasında paylaştırılarak iki sınıfı birbirinden ayıran doğru bulunmaya çalışılır.

$$f(g) = \begin{cases} 1 & \text{Eğer} \quad Ç>0 \text{ ise} \\ -1 & \text{Eğer} \quad Ç\leq 0 \text{ ise} \end{cases}$$

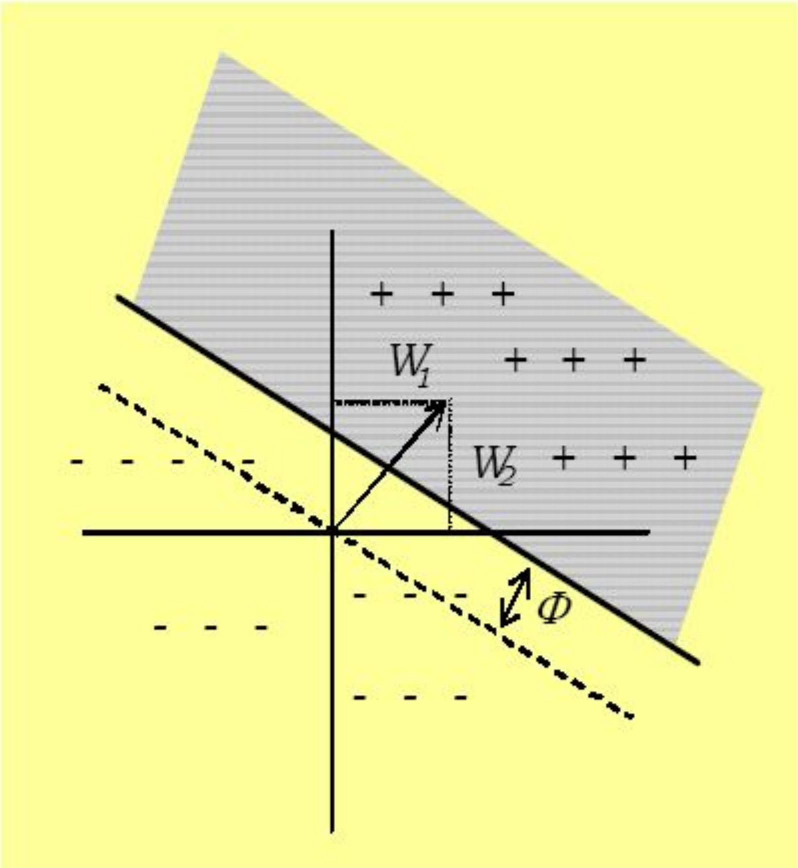
- Burada ağın çıktısı 1 ise birinci sınıfta , -1 ise ikinci sınıfta kabul edilmektedir.

Sınıf ayracı doğrusu şu şekilde tanımlanır:

$$W_1.X_1 + W_2.X_2 + \Phi = 0$$

$$X_1 = -(W_2/W_1)X_2 - \Phi/W_2$$
$$X_2 = -(W_1/W_2)X_1 - \Phi/W_1$$

Bu ağlardaki öğrenme işlemi ağırlık değerlerinin değiştirilmesi ile yapılır.



Ağırlıkların ve Sınıf Ayracı Olan Doğrunun Gösterimi

- t zaman biriminde ağırlık değerleri ΔW kadar değiştirilirse ağırlık değeri:

$$W_i\ (t+1) = W_i(t) + \Delta W_i(t)$$

olacaktır

.

- Ağırlıkların değiştirilmesi doğrunun eğiminin değişmesi anlamına gelir. Bu durumda eşik değerinin de değiştirilmesi gerekir.

- t zaman biriminde eşik değeri Δ kadar değiştirilirse eşik değeri:

$$\Phi(t+1) = \Phi(t) + \Delta\Phi(t)$$

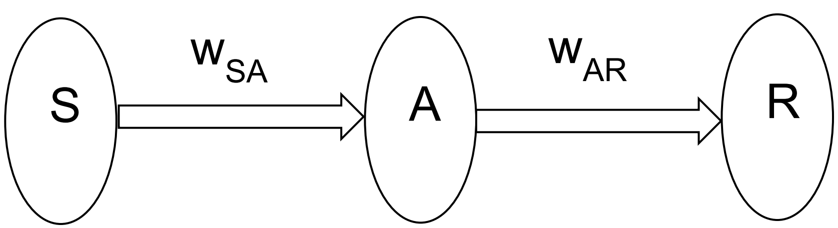
olacaktır

.

Perceptron

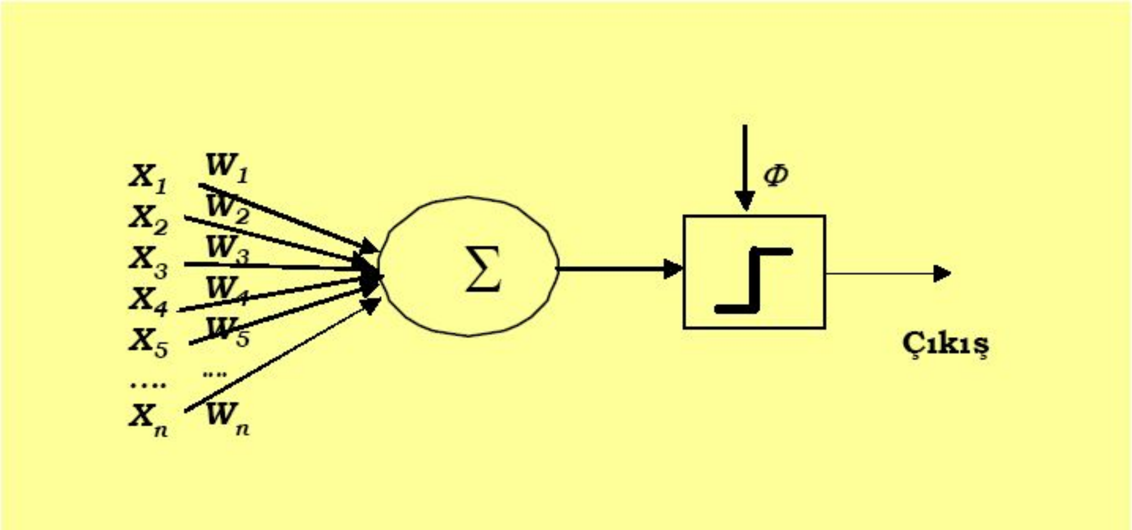
- Rosenblatt (1962)
  - İlk olarak görsel algıyı modellemede (retina) kullanıldı
  - Bir ileri beslemeli ağın üç parçası vardır :

    duyular(sensory), eşleştirme(association), cevap(response)
  - Öğrenim sadece A birimlerinden R birimlerine ağırlıklar üzerinde meydana gelir (S birimlerinden A birimlerine olan ağırlıklar sabittir).
  - Her bir ***R*** birimi n tane A biriminden giriş alır
  - Verilen bir eğitim örneği için (s:t), hedef çıktıdan (t) farklı bir y çıktısı hesap edilirse A ve R arasında ağırlıklar değiştirilir



Perceptron

- Perceptronlar, son derece sınırlı olmalarına karşın en eski sinir ağlarından biridir.
- Perceptron, bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanır.



Basit bir perceptron yapısı

Basit Algılayıcı Öğrenme Yapısı (Kurallı)

Adım1: Ağa, girdi seti ve ona karşılık beklenen çıktı gösterilir(X,B)

Girdiler $X = (x_1, x_2, \ldots, x_n)$

Çıktı $B = (1 \text{ veya } 0)$

Adım2: Perceptrona gelen Net girdi hesaplanır.

$$Net = \sum w_i x_i$$

Adım3: Perceptron çıkışı hesaplanır.

$$Çıkış = \begin{cases} 1 & \text{Eğer} \quad Net > \Phi \text{ ise} \\ 0 & \text{Eğer} \quad Net \leq \Phi \text{ ise} \end{cases}$$

- Ağın, beklenen çıktısı 0 iken Net girdi eşik değerinin üzerinde ise ağırlık değerleri azaltılmaktadır.

$$W_n = W_0 - \lambda X$$

- Ağın, beklenen çıktısı 1 iken Net girdi eşik değerinin altında ise ağırlık değerleri arttırılmaktadır.

$$W_n = W_0 + \lambda X$$

Adım4: Bütün girdi setindeki örnekler için doğru sınıflandırma yapılıncaya kadar ilk üç adımdaki işlemler tekrarlanır.

Örnek

İki girdi, bir çıktı ve öğrenmesi gereken 2 adet örnek veriliyor

1. örnek: $X_1=(x_1,x_2)=(1,0)$, $B_1=1$

2. örnek: $X_2=(x_1,x_2)=(0,1)$, $B_2=0$

Ağırlıklar $W=(w1,w2)=(1,2)$

Eşik değeri $\Phi = -1$

Öğrenme katsayısı $\lambda=0.5$

- 1.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1+ W_2.X_2 = 1*1+ 2*0 = 1$
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir.

- 2.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1+ W_2.X_2 = 1*0+ 2*1 = 2$
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n=W_0 - \lambda X$ dan
  $W_1=W_1 - \lambda X_1= 1-0.5*0=1$
  $W_2=W_2 - \lambda X_2= 2-0.5*1=1.5$

- 3.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*1+ 1,5*0 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir.

- 4.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*0+ 1.5*1 = 1.5
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n = W_0 - \lambda X$ dan
  $W_1 = W_1 - \lambda X_1$ = 1-0.5*0=1
  $W_2 = W_2 - \lambda X_2$ = 1.5-0.5*1=1

- 5.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*1+ 1*0 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir.

- 6.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*0+ 1*1 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n = W_0 - \lambda X$ dan
  $W_1 = W_1 - \lambda X_1$= 1-0.5*0=1
  $W_2 = W_2 - \lambda X_2$= 1-0.5*1=0.5

- 7.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*1+ 0.5*0 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir.

- 8.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*0+ 0.5*1 = 0.5
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n = W_o - \lambda X$ dan
  $W_1 = W_1 - \lambda X_1$= 1-0.5*0=1
  $W_2 = W_2 - \lambda X_2$= 0.5-0.5*1=0

- 9.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*1+ 0*0 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir.

- 10.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*0+ 0*1 = 0
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n = W_0 - \lambda X$ dan
  $W_1 = W_1 - \lambda X_1$ = 1-0.5*0=1
  $W_2 = W_2 - \lambda X_2$ = 0-0.5*1=-0.5

11.iterasyonda 1.örnek ağa gösterilir
Net = $W_1.X_1 + W_2.X_2$ = 1*1+ (-0.5)*0 = 1
Net > Φ olduğundan Çıkış=1 olacaktır,
Çıkış=B1 olduğundan ağırlıklar değişmeyecektir

12.iterasyonda 2.örnek ağa gösterilir
Net = $W_1.X_1 + W_2.X_2$ = 1*0+ (-0.5)*1 = -0.5
Net > Φ olduğundan Çıkış=1 olacaktır,
Çıkış≠B2 olduğundan ağırlıklar değişecektir. $W_n = W_o - \lambda X$ dan
$W_1 = W_1 - \lambda X_1 = 1-0.5*0=1$
$W_2 = W_2 - \lambda X_2 = -0.5-0.5*1=-1$

- 13.iterasyonda 1.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*1+ (-0.5)*0 = 1
  Net > Φ olduğundan Çıkış=1 olacaktır,
  Çıkış=B1 olduğundan ağırlıklar değişmeyecektir

- 14.iterasyonda 2.örnek ağa gösterilir
  Net = $W_1.X_1 + W_2.X_2$ = 1*0+ (-1)*1 = -1
  Net > Φ olduğundan Çıkış=0 olacaktır,
  Çıkış=B2 olduğundan ağırlıklar değişmeyecektir.

  Bundan sonra her iki örnek de doğru olarak sınıflandırılır. Öğrenme sonunda

  $\boldsymbol{W_1=1}$ $\boldsymbol{W_2=-1}$

Doğrusal Ayrılabilirlik

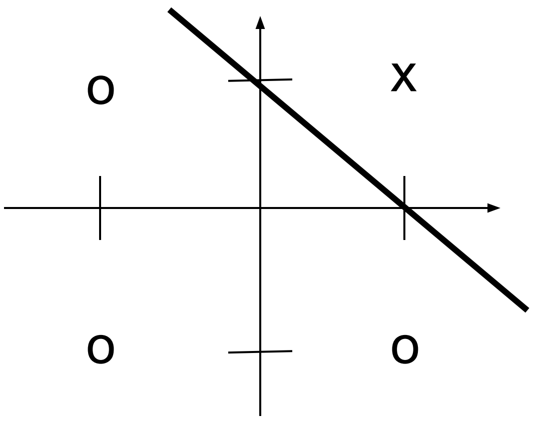
- Eğer (x_1, x_2) düzleminde bir hat varsa iki sınıfa ait iki boyutlu örüntülerin (x_1, x_2) bir kümesi doğrusal olarak ayrılabilir.
  - $w_0 + w_1 x_1 + w_2 x_2 = 0$
  - Bir sınıftan diğerine bütün örüntüleri ayır
- Bir perceptron;
  - Ağırlıkları (w_0, w_1, w_2) ve girişleri ($x_0 = 1$, x_1, x_2) olan üç girişi ile inşa edilebilir.
- n boyutlu örüntüler (x_1,…, x_n)
  - Düzlem $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$ uzayı iki parçaya bölüyor
- Örnek örüntülerin bir kümesinden ağırlıkları elde edebilir miyiz?
  - Eğer problem doğrusal olarak ayrılabilir ise o zaman EVET (perceptron öğrenimi ile)

Örnekler

- Mantıksal **AND fonksiyonu**

örüntüler (bipolar) karar sınırı

x1	x2	output	
			w1 = 1
-1	-1	-1	w2 = 1
-1	+1	-1	w0 =-1
+1	-1	-1	
+1	+1	+1	-1 + x1 + x2 = 0

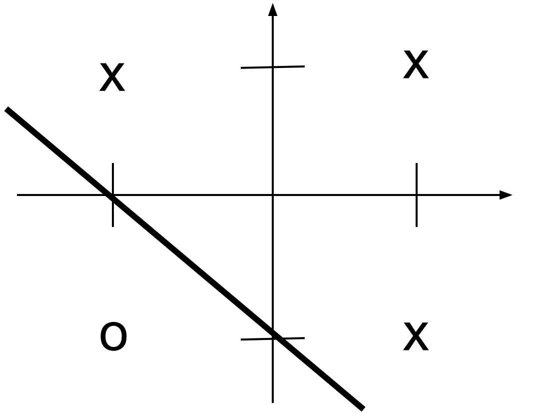


x:class I (output =1)
o:class II (output=-1)

- Mantıksal **OR** fonksiyonu

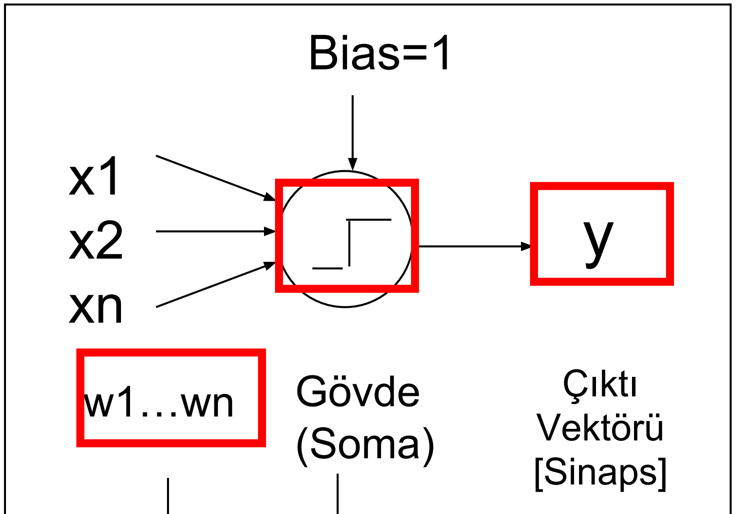
örüntüler (bipolar) karar sınırı

x1	x2	output	
			w1 = 1
-1	-1	-1	w2 = 1
-1	+1	+1	w0 = 1
+1	-1	+1	
+1	+1	+1	1 + x1 + x2 = 0



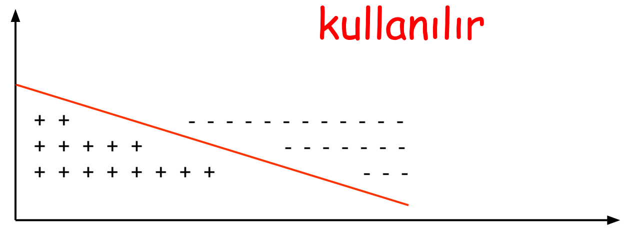
x: class I (output = 1)
o: class II (output = -1)

Perceptron Yapısı



Wn =Wn + LR*Xn*E

Tek katmanlı perceptronlar sadece doğrusal problemlerde kullanılır



Ağırlık Vektörü = W
Her girişin ağırlıkları atanıyor

Hard-Lim aktivasyon foksiyonu

g()=İf Σ(wi.xi)>0 then 1 else 0

Çıktı Vektörü başka katmanları besleyebilir

Perceptron Nerelerde Kullanılır ?

- Perceptron doğrusal bir fonksiyonla iki parçaya bölünebilen problemlerde kullanılabilir.

- Bu problemlere AND,OR,NOT durumları örnek olarak verilebilir.

Terimler

Epoch : Olası girdiler için ağırlıkların güncellenme sayısına denir.

Error: Çıktı değeriyle bizim fonksiyondan beklediğimiz değer arasındaki farktır. Örneğin, eğer biz çıktı olarak 0 bekleyip de 1 aldığımızda hata (error) değeri -1‘ dir.

Target Value, T : Perceptrondan öğrenmesini beklediğimiz değerdir. Örneğin, eğer AND fonksiyonuna [1,1] girdisini verirsek bekleyeceğimiz sonuç 1 ‘dir.

Output , O : Perceptron’un verdiği çıktıdır.

Xi : Neuron’ a verilen girdi

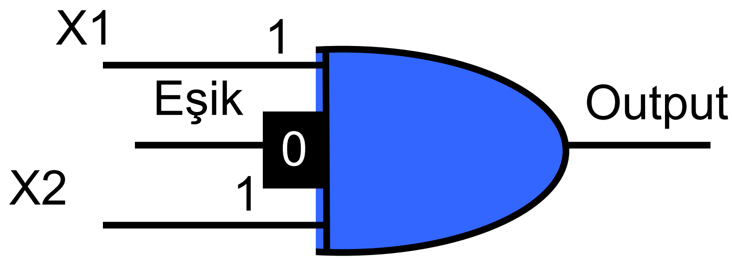
Wi : Xi inci girdinin ağırlık değeri

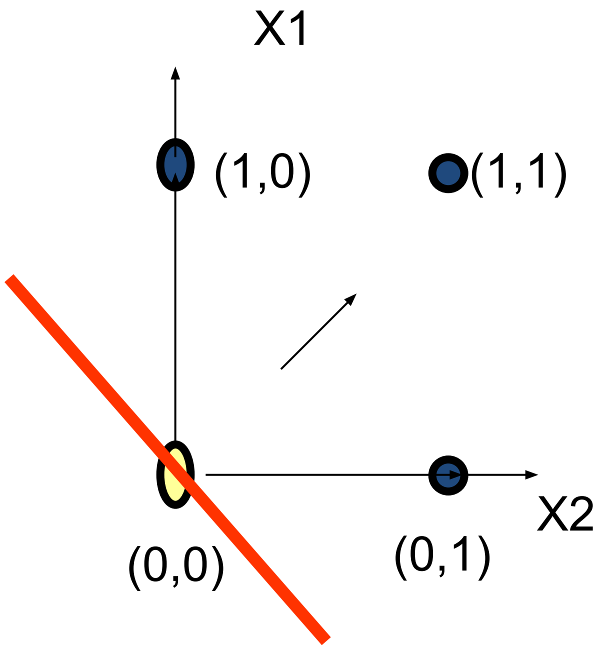
LR : Learning rate. Bir perceptron’un hedefe varması için gereken adım büyüklüğü.

Öğrenme Algoritması

```
Girdilere göre çıktıyı hesapla
Error = T – O
If Error <> 0 then
    Wi = Wi + LR * Xi * Error
End If
```

Örnekler





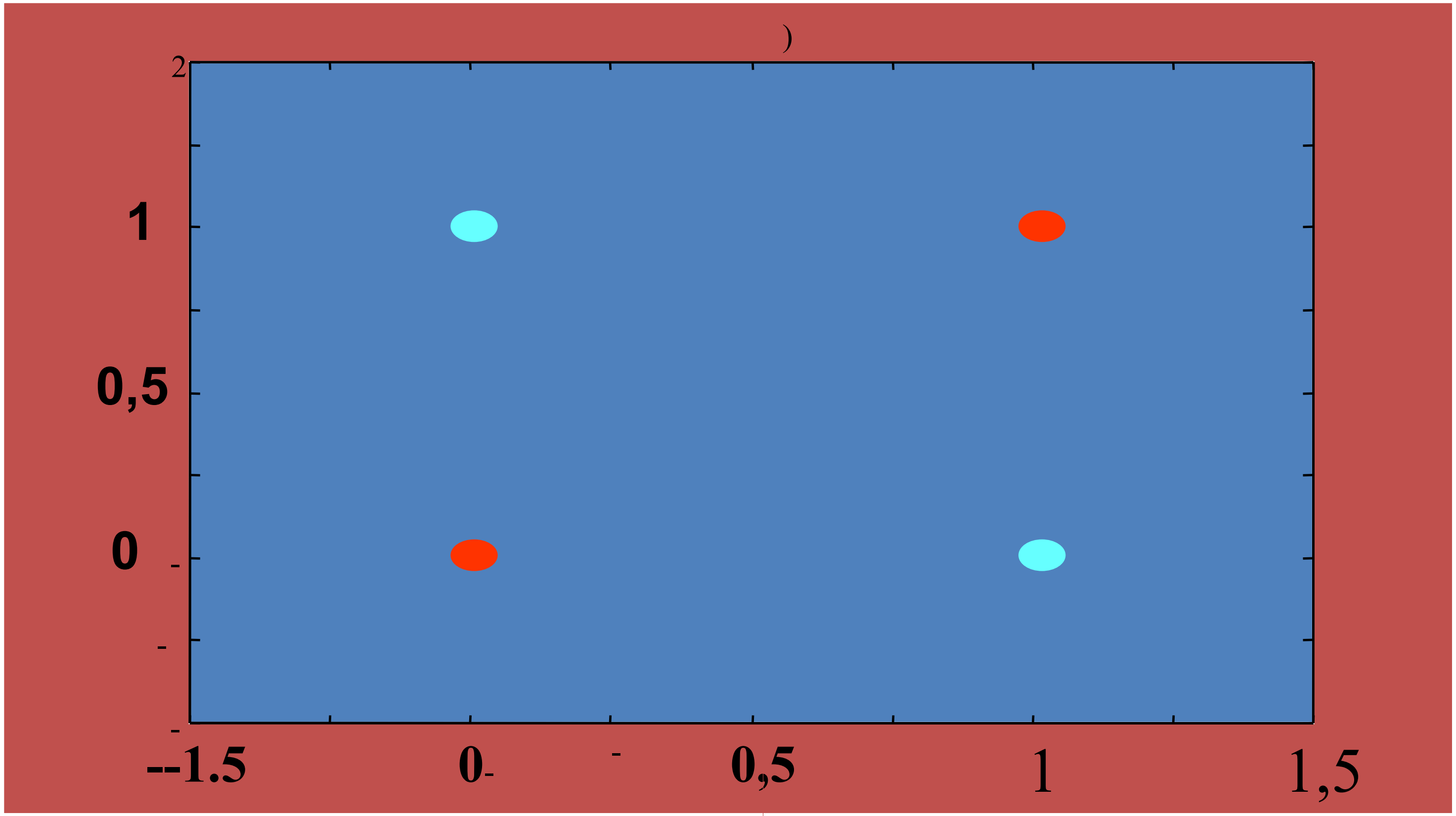
- Net parametre değerleri (bias, Wx1, Wx2) = 0, 1, 1)

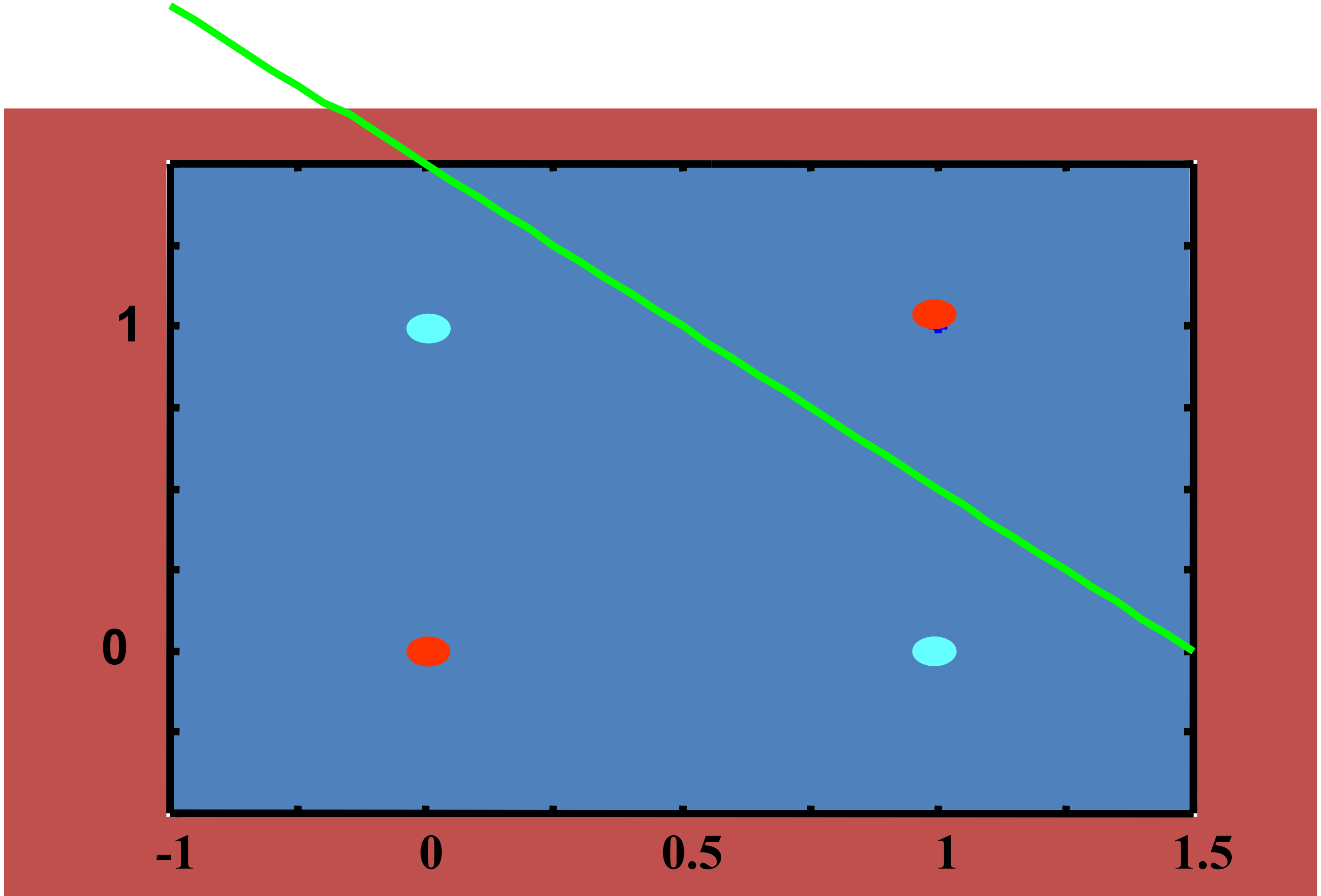
- If X1 = 1 and X2 =1 then Output Hard-Lim (Bias + Wx1 * X1 + Wx2 * X2)

- Output =Hard-Lim(0 + 1 * 1 + 1 * 1) = 1

X1	X2	Bias + Wx1*X1 + Wx2 *X2	Output
1	1	0+1*1 + 1*1 = 2	1
1	0	0 + 1*1 + 1*0 = 1	1
0	1	0 + 1*0 + 1*1 = 1	1
0	0	0 + 1* 0 + 1* 0 = 0	0

XOR Problemi





İkinci Girdi kümesi için hatalı

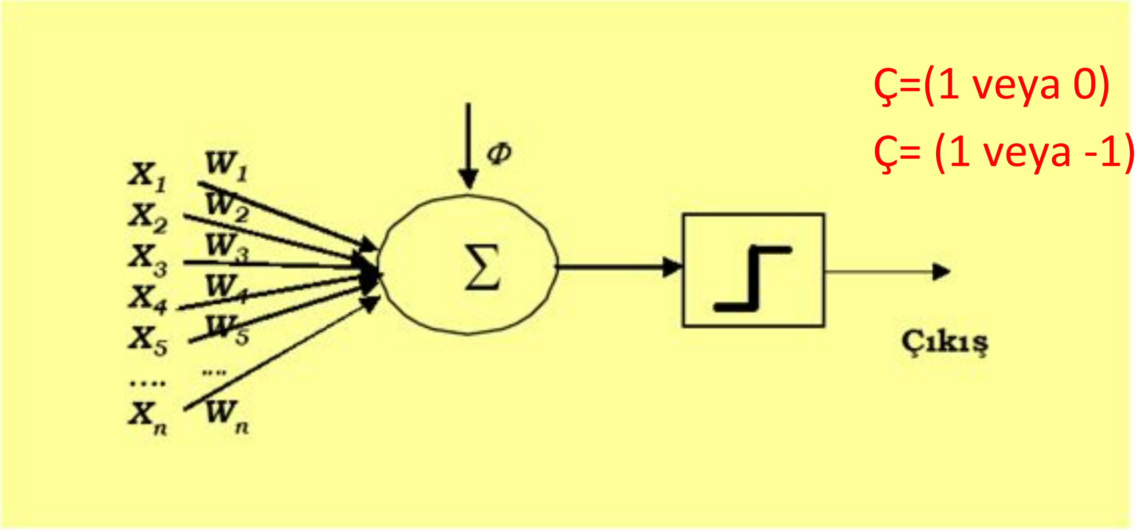
- Perceptronlar XOR problemi gibi doğrusal olarak sınıflandırılamayan problemleri çözümünde başarısızdır.

ADALINE/MADALINE Modeli

- ADALINE Widrow ve Hoff tarafından 1959 yılında geliştirilmiş olup adaptif doğrusal eleman (Adaptif Linear Element) ağının kısaltılmış şeklidir. Genel olarak ADALINE bir proses elemanından oluşan bir ağdır.

- Bu ağ en küçük ortalamaların karesi (least mean square) yöntemine dayanmaktadır. Öğrenme kuralına "Delta Kuralı" da denmektedir.

Öğrenme kuralı, ağın çıktısının beklenen çıktı değerine göre hatasını enazlayacak şekilde ağın ağırlıklarının değiştirilmesi prensibine dayanmaktadır.



ADALINE Ünitesi

ADALINE Ünitesinin Öğrenme Kuralı

- ADALINE ünitesi en küçük kareler yöntemini kullanarak öğrenme gerçekleştirir. Perceptron algoritmasına çok benzemektedir.

Adım1: Ağın net girdisi hesaplanır.

$$NET=\sum_{i=1}^{m} w_i x_i + \Phi$$

$$NET=\Phi+x_1w_1+x_2w_2+x_3w_3+.........+x_nw_n$$

Adım2: Ağın çıktısı belirlenir.

Çıktı(Ç)=1 Eğer NET >=0 ise

Çıktı(Ç)=-1 Eğer NET <0 ise

- Ağın çıktısını üreten fonksiyon bilinen bir step fonksiyondur. Beklenen değerin B olduğu varsayılırsa Adaline ünitesinin çıktısını ürettikten sonraki hatası :

$$\boxed{E = B - Ç}$$

olacaktır

.

Amaç bu hatayı en aza indirecek ağırlıkları bulmaktır. Bunun için her seferinde ağa farklı örnekler gösterilerek hatalar hesaplanmakta ve ağırlıklar hatayı azaltacak şekilde değiştirilmektedir.

Zaman içinde hata, olması gereken en küçük değere düşmektedir. Bu hatayı azaltmak için izlenen kural aşağıdadır:

$$W_{yeni} = W_{eski} + \alpha(B-Ç)\ X$$

Diğer bir deyişle her hangi bir t anında ;

$$W_i(t) = W_i(t-1) + \alpha * E * X_i$$

olacaktır

.

W(t)→ Ağırlıkların t zamanındaki yeni değerini W(t-1) → Ağırlıkların önceki değerini α→ Öğrenme katsayısını B→ Beklenen çıktıyı E→Beklenen değer ile çıktı arasındaki hatayı X→ Girdi değerini göstermektedir.

- Benzer şekilde eşik değeri de yine zaman içerisinde değiştirilerek olması gereken eşik değeri bulunur.

$$\Phi_{yeni} = \Phi_{eski} + \alpha\,(B\text{-}Ç)$$

Örnek

Bir donanım üreticisi elindeki CRT ve LCD monitörleri birbirlerinden ayırt edebilen bir sistem geliştirmek istemektedir. Bu amaçla bir yapay sinir ağı geliştirilebilir mi?

Yapılacak ilk iş monitörlerin birbirlerinden farklılıklarını ortaya koyabilen örnekler oluşturabilmektir. Bunun için ise monitörleri ve onun özelliklerini gösteren vektörleri belirlemek gerekmektedir. Monitörlerin ağırlığını, en-boy-yükseklik bilgisini temsil etmek üzere 4 boyutlu bir vektör ilk aşamada oluşturulabilir.

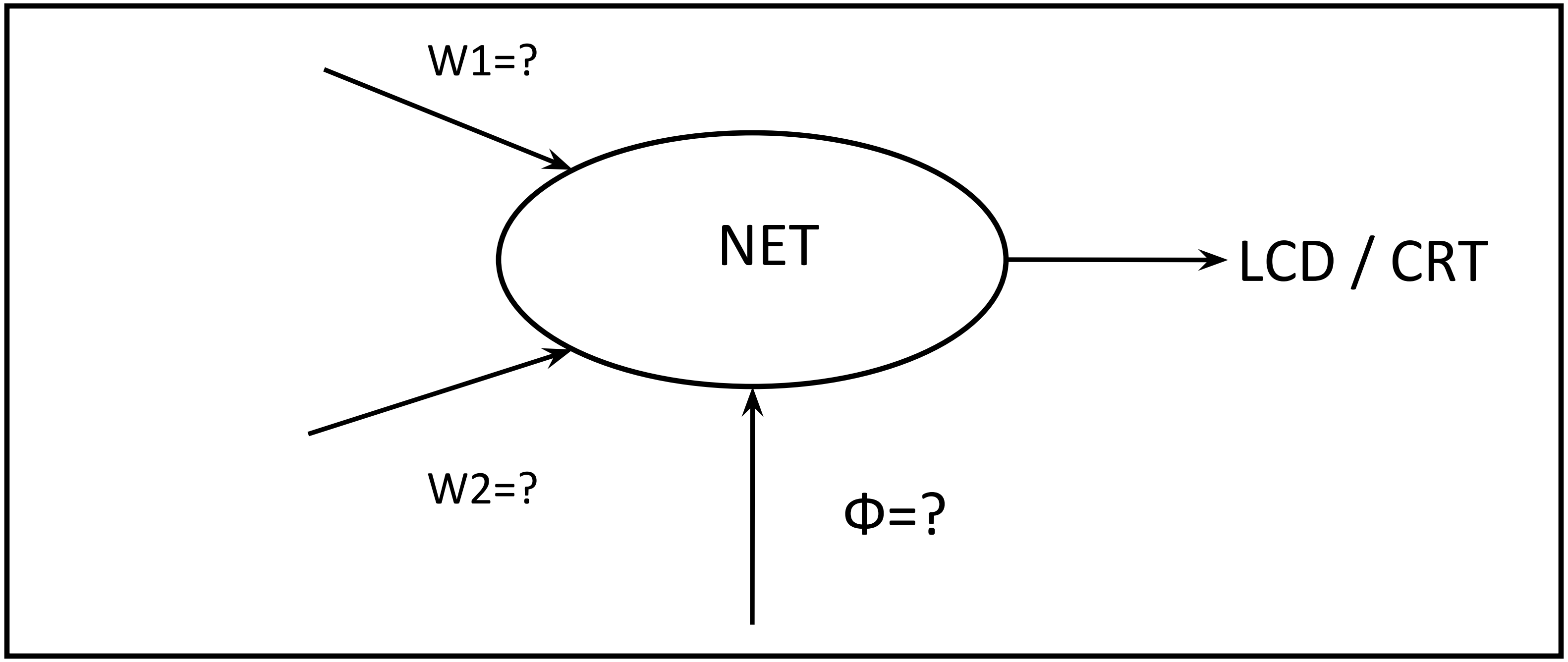
Çözüm

- CRT ve LCD monitörü gösteren prototiplerin aşağıdaki vektörler ile gösterildiği varsayılırsa örnek setinde 2 örnek olacaktır.

Örnek1: CRT X1=(1,0); Bu örneğin beklenen çıktısı Ç1=-1

Örnek2: LCD X2=(0,1); Bu örneğin beklenen çıktısı Ç2= 1

Bu problemi çözebilmek için 2 girdisi olan bir Adaline ünitesi tasarlanacaktır. Öğrenmenin amacı problem girdilerini doğru sınıflandıracak ağırlık değerleri ve eşik değerlerini bulmaktır.



Problemin çözümü için ağırlık değerleri (w1, w2 ve eşik değerleri) başlangıçta rasgele atanmaktadır. Bu değerler:

$w1=0.3$ $w2=0.2$

$\alpha=0.5$ $\Phi=0.1$

Birinci İterasyon

Bu iterasyonda öncelikle birinci girdi vektörünün ağa gösterilmesi sonucu ağın çıktısı hesaplanır.

$$NET=\sum_{i=1}^{m} w_i x_i+\Phi$$

$$NET=0.3*1+0.2*0+0.1=0.4$$

$$NET>0 \quad Ç=1$$

$$E=B-Ç=-1-1=-2$$

$$E=-2$$

Oysaki beklenen B=-1’dir. Bu durumda ağın ağırlıklarının değiştirilmesi gerekmektedir.

- Bu ağın yeni ağırlıkları aşağıdaki gibi hesaplanır:

$$W_{yeni} = W_{eski} + \alpha(B\text{-}Ç)\ X$$
$$W1_{yeni} = 0.3+0.5(-2)1=-0.7$$
$$W2_{yeni} = 0.2+0.5(-2)0=0.2$$

- Eşik değer ünitesinin ağırlığı da benzer şekilde hesaplanır.

$$\Phi_{yeni}= \Phi_{eski} + \alpha\ (B\text{-}Ç)$$

$$\Phi_{yeni}= 0.1+0.5(-2)=-0.9$$

- Bu şekilde birinci iterasyon tamamlanmış olur.

İkinci İterasyon

- İkinci iterasyonda benzeri işlemler ikinci örnek için yapılır.

$$NET=\sum_{i=1}^{m} w_i x_i + \Phi$$

$$NET=-0.7*0+0.2*1-0.9=-0.7$$

$$NET<0 \quad Ç=-1$$

$$E=B-Ç=1-(-1)=2$$

$$E=2$$

Oysaki beklenen B=1’dir. Bu durumda ağın ağırlıklarının değiştirilmesi gerekmektedir.

- Bu ağın yeni ağırlıkları aşağıdaki gibi hesaplanır:

$$W_{yeni} = W_{eski} + \alpha(B-Ç)\ X$$

$$W1_{yeni} = -0.7+0.5(2)0=-0.7$$

$$W2_{yeni} = 0.2+0.5(2)1=1.2$$

- Eşik değer ünitesinin ağırlığı da benzer şekilde hesaplanır.

$$\Phi_{yeni}= \Phi_{eski} + \alpha\ (B-Ç)$$

$$\Phi_{yeni}= -0.9+0.5(2)=0.1$$

Üçüncü İterasyon

- Bu iterasyonda yine birinci girdi vektörünün ağa gösterilmesi sonucu ağın çıktısı hesaplanır.

$$NET=\sum_{i=1}^{m} w_i x_i+\Phi$$

$$NET=-0.7*1+1.2*0+0.1=-0.6$$

$$NET<0 \longrightarrow Ç=-1$$

- Bu iterasyonda B=-1 olması gerektiğinden ağın sınıflandırması doğrudur. Bu durumda ağırlıklarda bir değişiklik yapılması gerekmez.

Dördüncü İterasyon

- Bu iterasyonda ise ikinci girdi vektörünün ağa gösterilmesi sonucu ağın çıktısı hesaplanır.

$$NET=\sum_{i=1}^{m} w_i x_i + \Phi$$

$$NET=-0.7*0+1.2*1+0.1=1.3$$

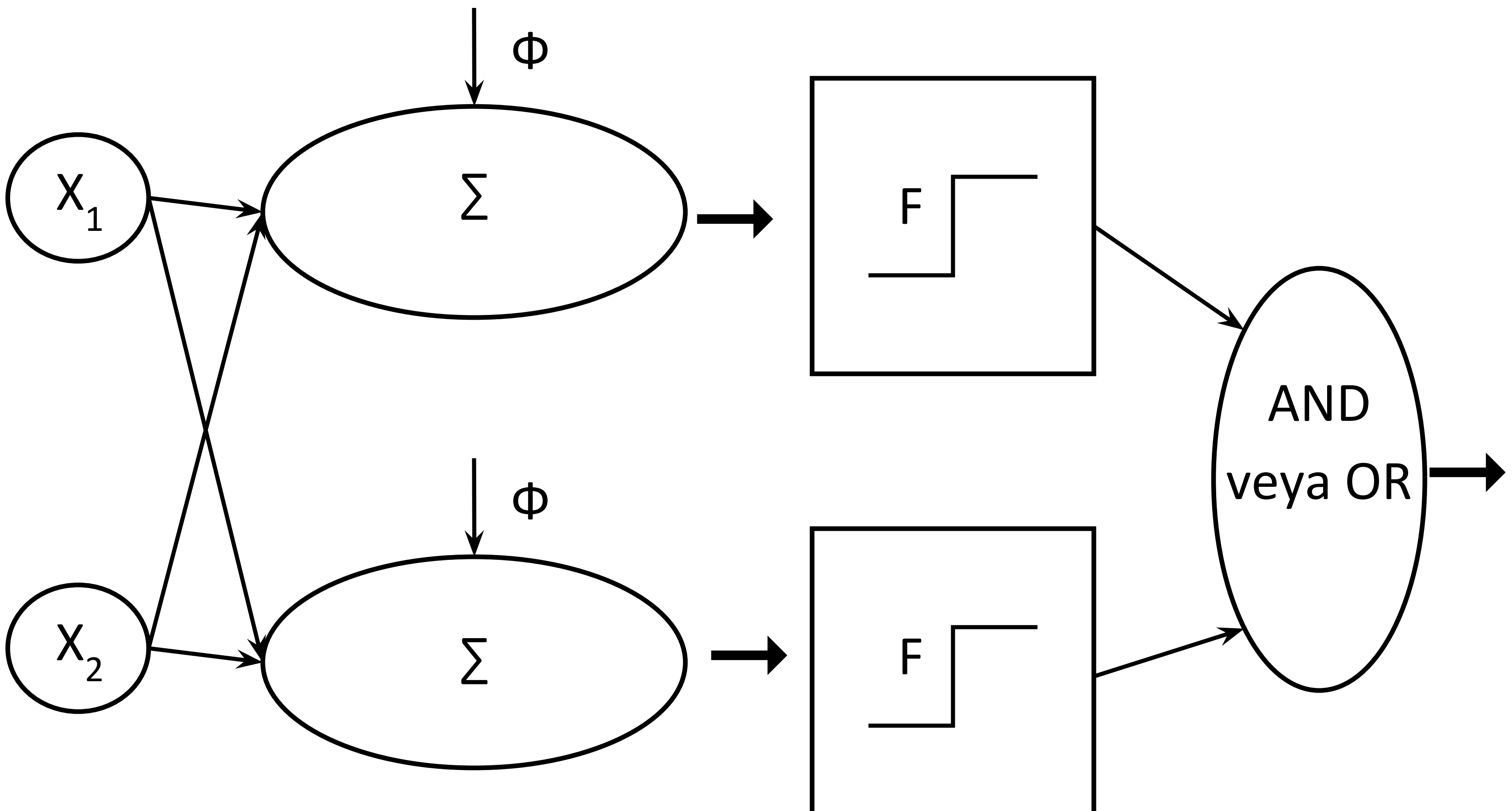
$$NET>0 \longrightarrow Ç=1$$

- Bu iterasyonda B=1 olması gerektiğinden ağın sınıflandırması doğrudur. Bu durumda ağırlıklarda bir değişiklik yapılması gerekmez.

MADALINE

- MADALINE ağları birden fazla ADALINE ünitesinin bir araya gelerek oluşturdukları ağa verilen isimdir. Bu ağ ile ilgili açıklamalar Widrow ve Lehr tarafından verilmiştir.

- MADALINE ağları genel olarak iki katmandan oluşmaktadır. Her katmanda değişik sayıda ADALINE ünitesi bulunmaktadır. Ağın çıktısı yine 1 ve -1 değerleri ile gösterilmektedir.



İki ADALINE Ünitesinden Meydana Gelen MADALINE Ünitesi