SAKARYA
ÜNİVERSİTESİ

BSM 310
YAPAY ZEKA

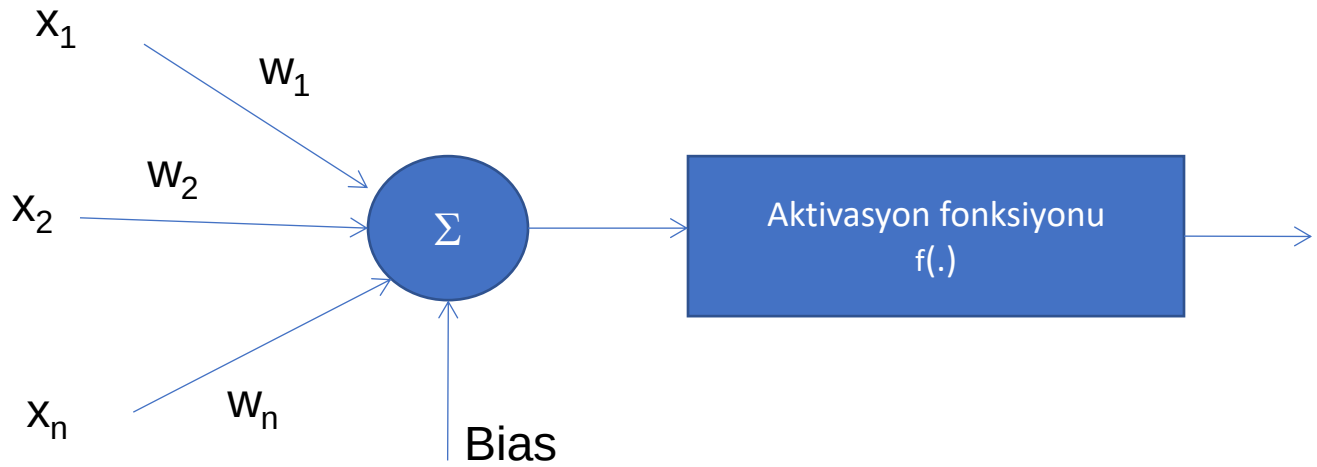
CEMİL ÖZ, İSMAİL ÖZTEL

~ YAPAY SİNİR AĞLARI ~

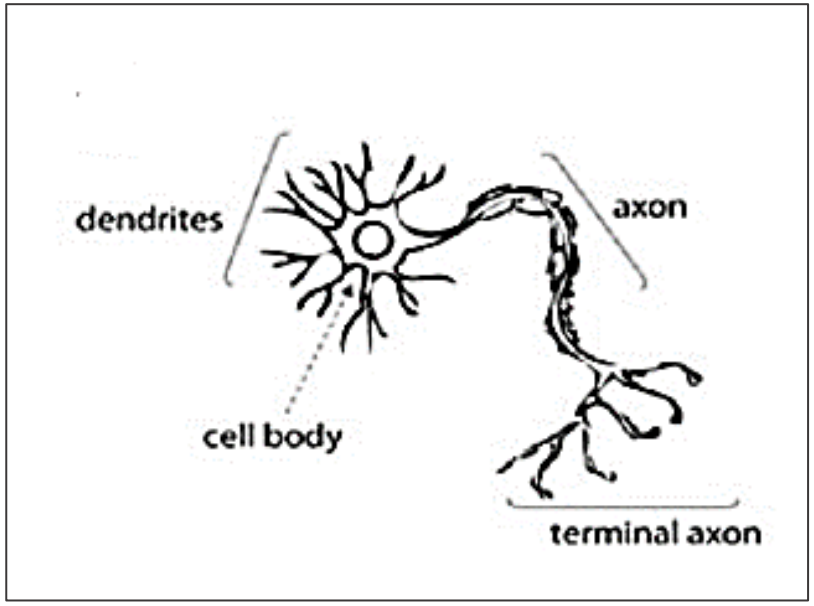
KONULAR

- Yapay sinir ağlarına giriş
- Yapay sinir ağlarının kullanım alanları
- Yapay sinir ağlarında öğrenme
- Yapay sinir hücresi
- Perceptron (algılayıcı)
- Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları
- Geri Yayılım Algoritması

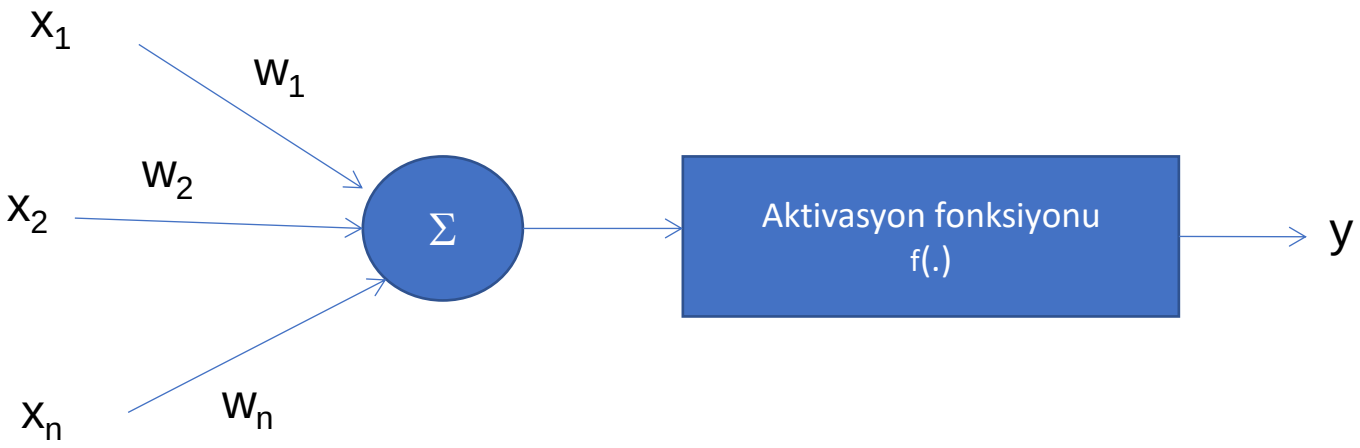
Yapay sinir hücresi



- Yapay sinir; biyolojik sinirin girdi, işlem ve çıktı karakteristiğini taklit etmek için tasarlanmıştır.
- Burada her bir girdi kendi ağırlığı ile çarpılmakta ve bu çarpımların hepsi toplanmaktadır.
- Bu toplam sinaptik kuvvete benzetilebilir ve nöronun aktivitesini belirlemek için kullanılır



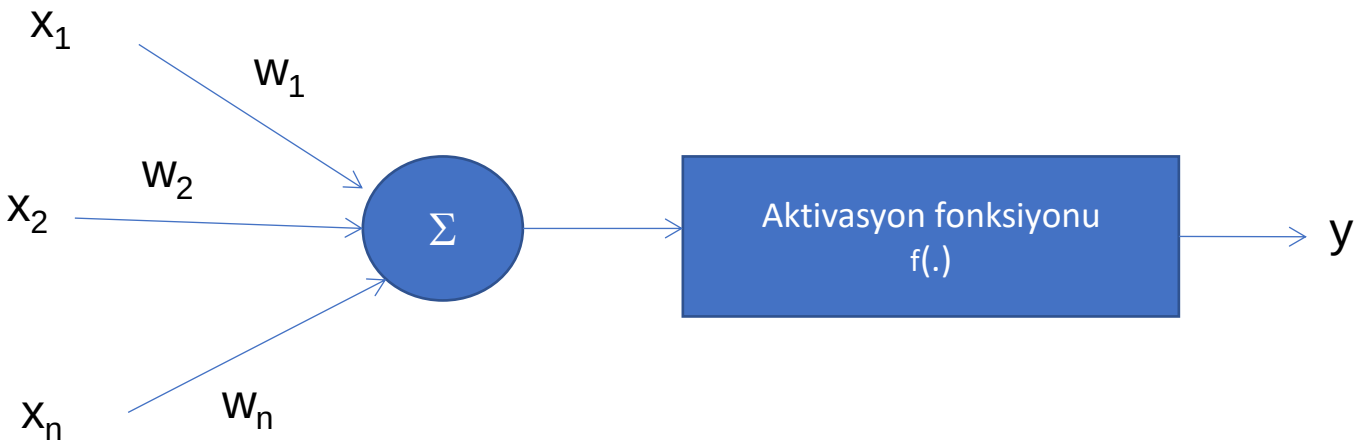
Yapay sinir hücresi



$$\boldsymbol{net} = \boldsymbol{x_1 w_1} + \boldsymbol{x_2 w_2} + \ldots + \boldsymbol{x_n w_n}$$

$$= \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_i}$$

Yapay sinir hücresi



$$\boldsymbol{y = f(net)}$$

$$= \begin{cases} \mathbf{1}, & \boldsymbol{net \geq t} \\ \mathbf{0}, & \boldsymbol{net < t} \end{cases}$$

(örnek fonksiyon)

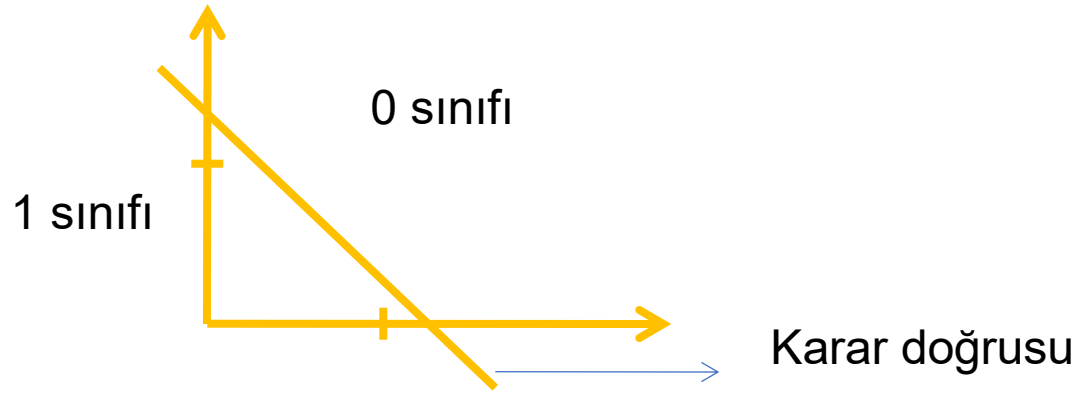
Yapay sinir hücresi

- YSA'da *net* sinyali genellikle aktivasyon fonksiyonu *f* ile işlem görerek nöronun çıkış sinyalini oluşturur.
- YSA'da kullanılacak olan aktivasyon fonksiyonuna problemin yapısına göre karar verilir.
- Daha önceki deneyimlerden yararlanılarak uygun aktivasyon fonksiyonu seçilebilir.
- Örnek fonksiyonlar: $\boldsymbol{f(x) = x}$ .....lineer fonksiyon

$$\boldsymbol{f(x) = \frac{1}{1+e^{-x}}}$$ .....sigmoidal fonksiyon

Perceptron (algılayıcı)

- Tek katmanlı algılayıcılar problem uzayını bir doğru, düzlem veya hiper düzlem ile doğrusal olarak sınıflandırılabilir.
- Doğrusal olmayan sınıflama işlemlerini yapamazlar. Noonlineer sınıflandırma için çok katmanlı algılayıcılar geliştirilmiştir.
- Örnek ayrıştırma problemi: NAND problemi

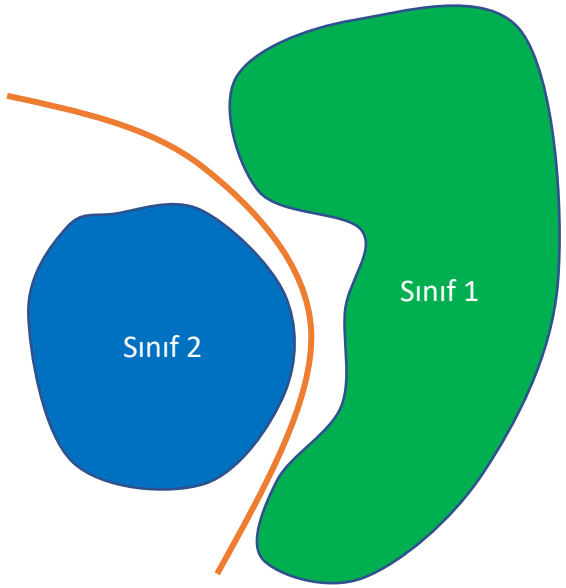


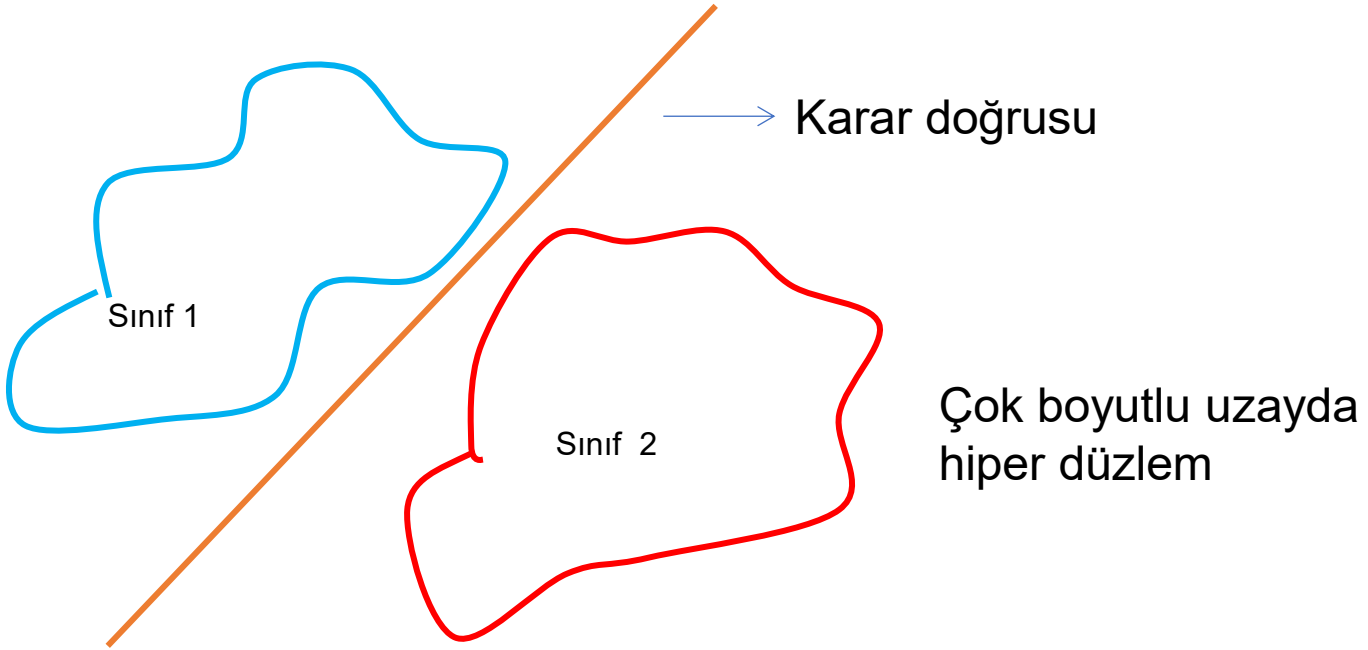
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

- Doğrusal ayrıştırılabilir. Tek perceptron yeterli

Perceptron (algılayıcı)

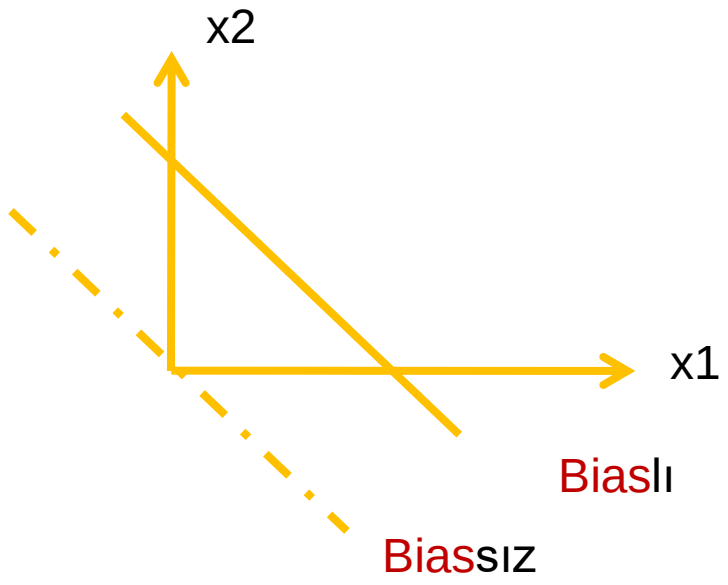
- Perceptronun uygun bir şekilde sınıflandırılabilmesi için, ayırt edilecek iki sınıfın doğrusal olarak ayırt edilebilmesi gereklidir.



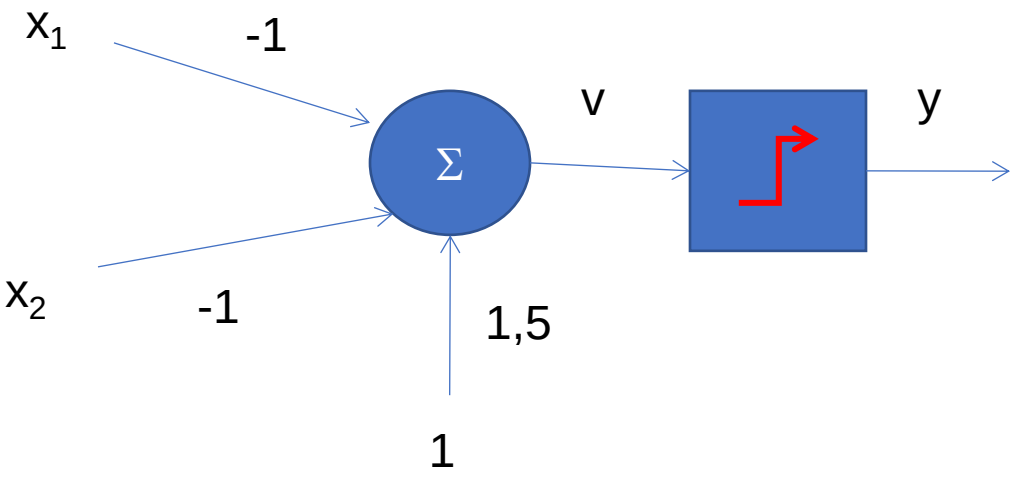


Perceptron (algılayıcı)

- Karar doğrusunda bias doğrunun orijinden kaydırılmasını sağlar.
- Ağırlıklar eğrinin eğimini belirler.

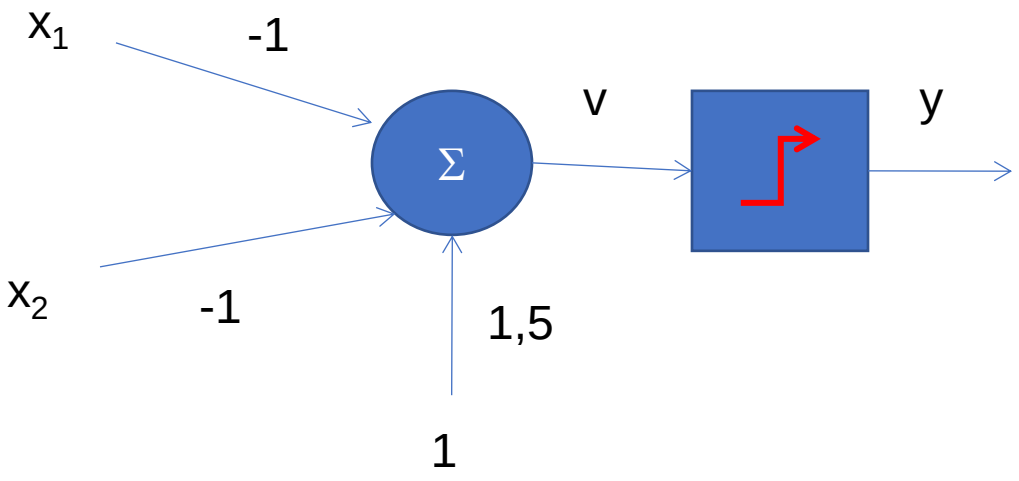


Lineer NAND problemi



$$Y = \begin{cases} 1 \text{ eğer } v>0 \\ 0 \text{ eğer } v<=0 \end{cases}$$

Lineer NAND problemi



$$Y = \begin{cases} 1 \text{ eğer } v>0 \\ 0 \text{ eğer } v<=0 \end{cases}$$

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

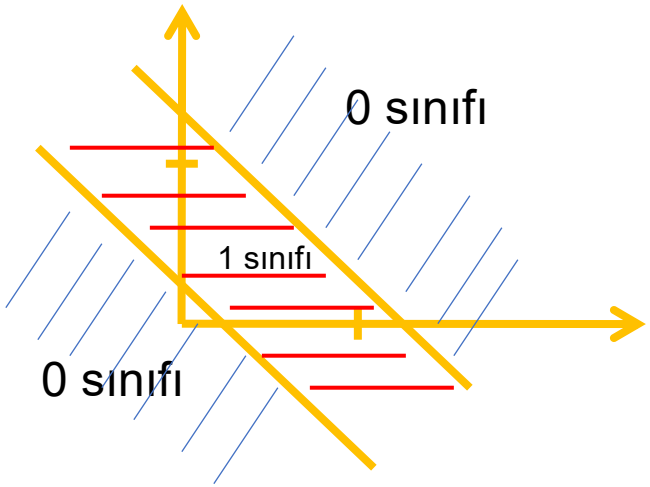
- $v = -x_1 - x_2 + 1,5$

- -1(0)-1(0)+1,5=1,5
  - ise y=1
- -1(0)-1(1)+1,5=0,5
  - ise y=1
- -1(1)-1(0)+1,5=0,5
  - ise y=1
- -1(1)-1(1)+1,5=-0,5
  - ise y=0

Doğrusal olmayan örnek problem: XOR

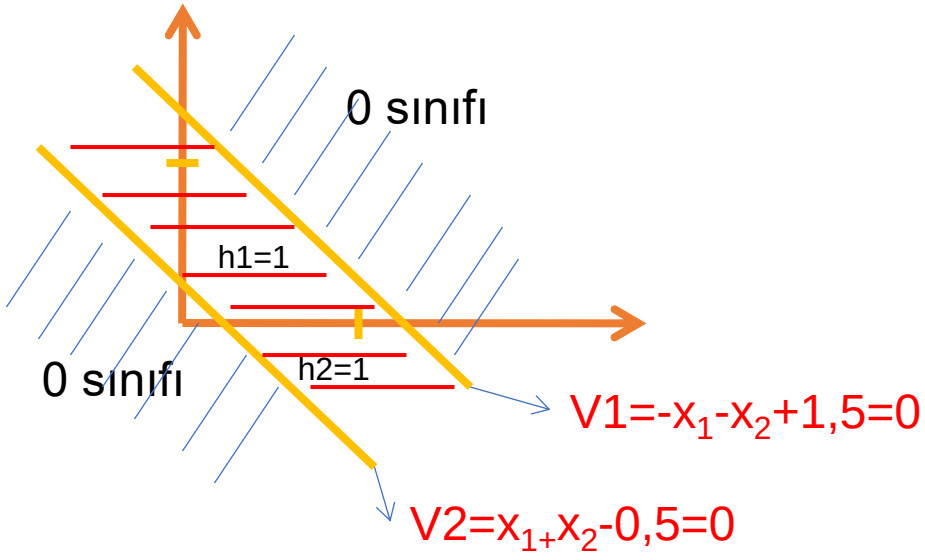
- XOR problemi; doğrusal ayrıştırılamayan problemdir, tek katmanlı perceptronlar çözemez.
- XOR probleminin önemi; tek katmanlı perceptronların bu problemi çözemeyeceği gösterilmesi ile YSA ile ilgili çalışmalar durma noktasına gelmiştir.
- Çok katmanlı perceptron ile bu problemin çözülmesi YSA'ya olan ilgiyi yeniden artırmıştır.
- XOR problemi doğrusal olmayan bir problemi/ilişkiyi ifade ettiği için doğrusal olmayan problemleri temsil etmektedir.

Doğrusal olmayan örnek problem: XOR



A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Doğrusal olmayan örnek problem: XOR

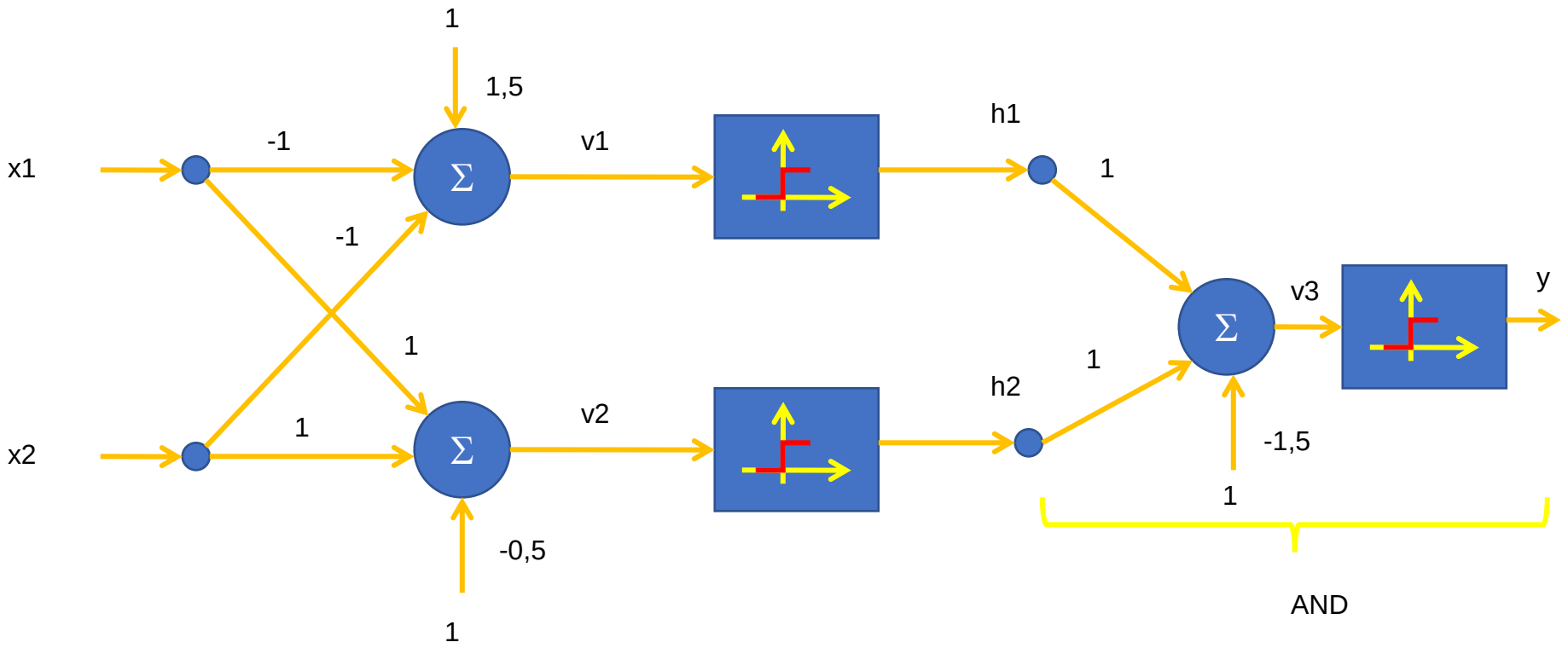


$V1=-x_1-x_2+1,5=0$

$V2=x_{1+}x_2-0,5=0$

$v3=h_1+h_2-1,5=0$

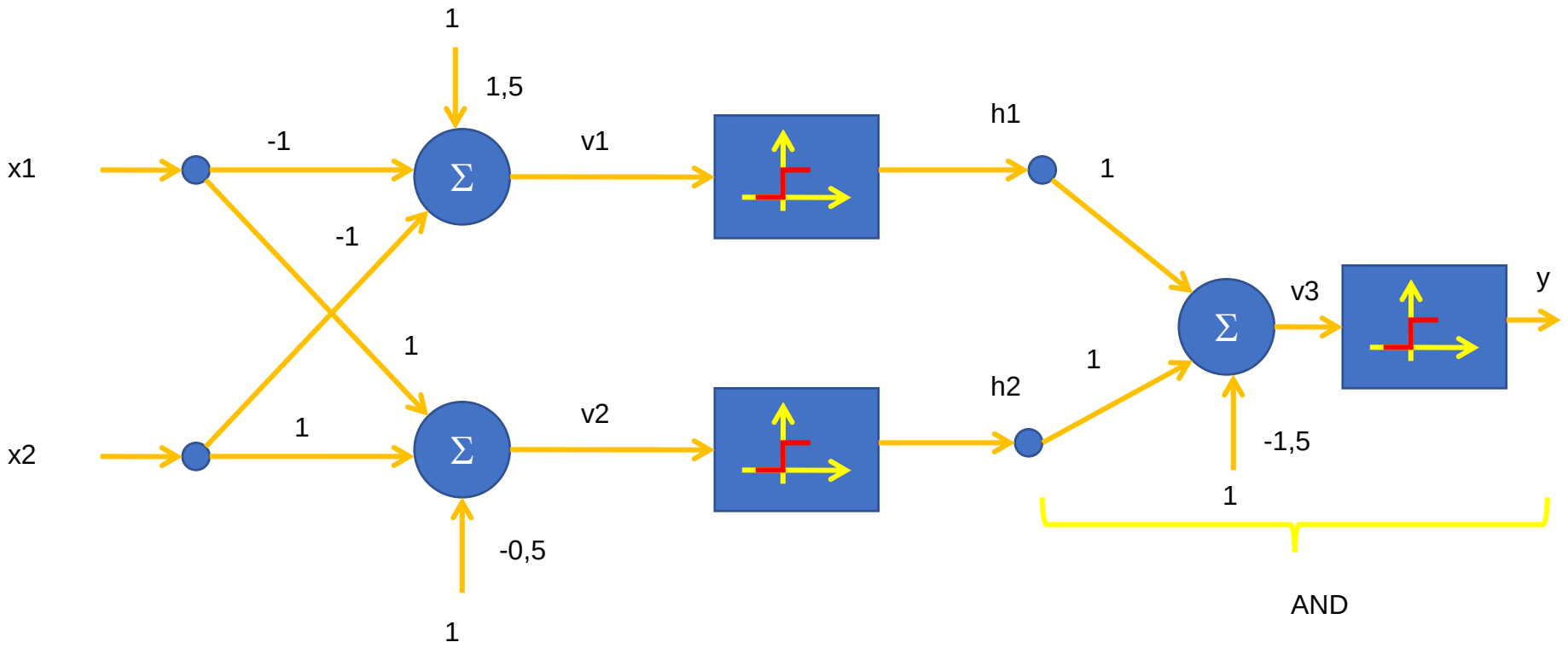
Doğrusal olmayan örnek problem: XOR



x1	x2	v1	v2	h1	h2	v3	y
0	0						
0	1						
1	0						
1	1						

Doğrusal olmayan örnek problem: XOR

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0



x1	x2	v1	v2	h1	h2	v3	y
0	0	1,5	-0,5	1	0	-0,5	0
0	1	0,5	0,5	1	1	0,5	1
1	0	0,5	0,5	1	1	0,5	1
1	1	-0,5	1,5	0	1	-0,5	0

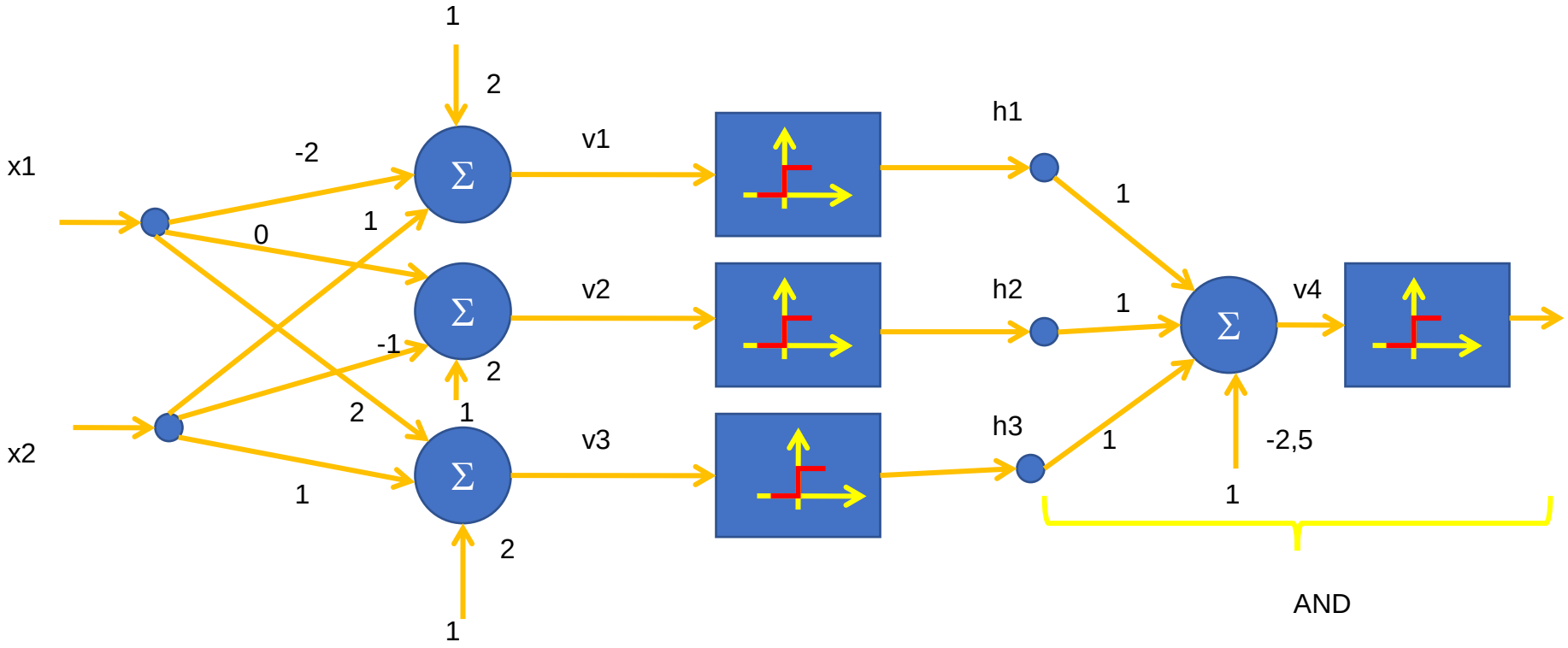
Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları

- Çok katmanlı ağlar denetlemeli öğrenme kullanılarak yapılan ağırlık güncellemesi ile doğrusal olmayan bir giriş çıkış ilişkisine (y=f(x)) yakınsama yapar.
- Çok katmanlı ileri beslemeli yapay sinir ağlarında genellikle sigmoid türü aktivasyon fonksiyonları kullanılır.
- Diğer aktivasyon fonksiyonları da yer yer bir alternatif olarak kullanılmaktadır.
- Dikkat edilmesi gereken durumlardan birisi türev bazlı öğrenme kurallarının uygulanabilmesi için aktivasyon fonksiyonunun türevinin alınabilir olması gerekmektedir.

Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları

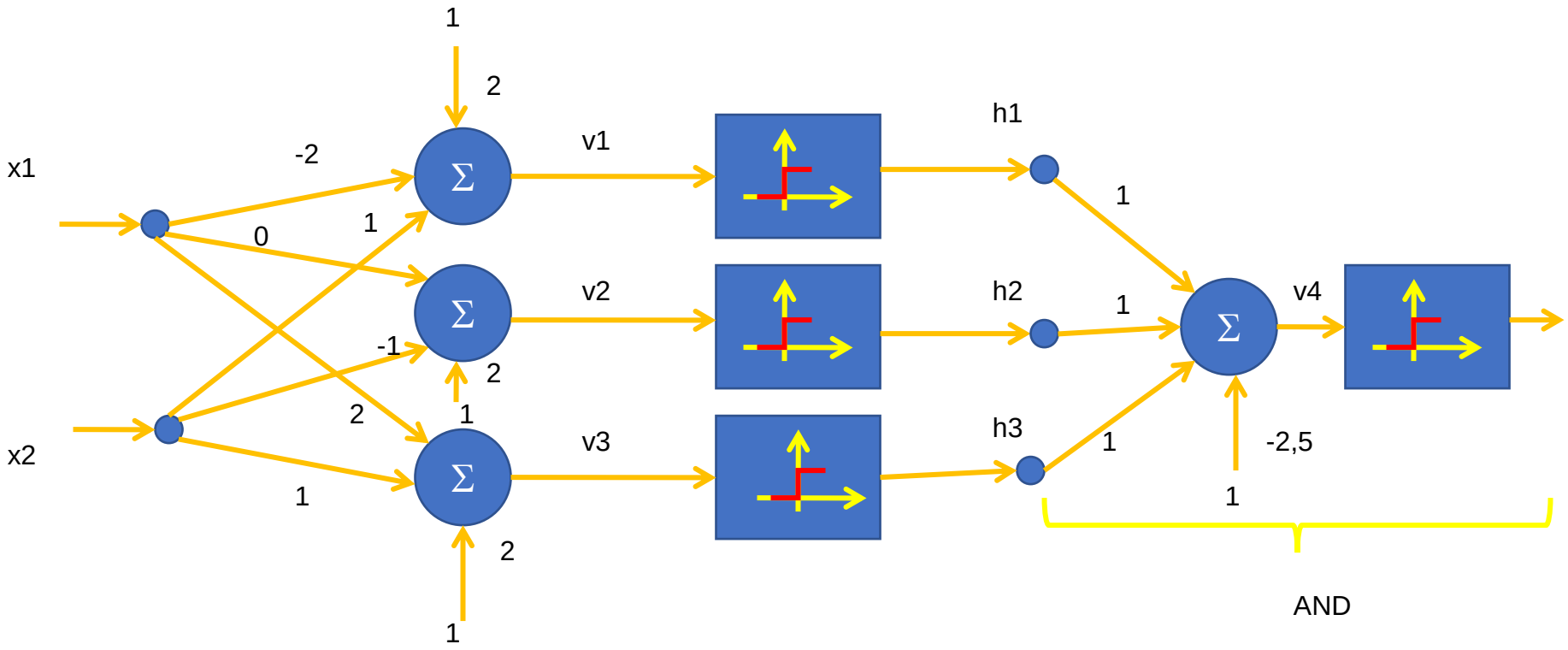
- Sigmoid türü fonksiyonlar, ileri beslemeli yapay sinir ağlarında kullanılan türevi alınabilir tipik doğrusal olmayan fonksiyonlardır.
- Sigmoid tipi aktivasyon fonksiyonlarının özellikleri;
  - Sigmoid fonksiyonlarının türevleri kolaylıkla basit aritmetik işlemler kullanılarak çıkış sinyallerinden elde edilebilirler.
  - Sigmoid tipi aktivasyon fonksiyonlarının türevleri hiç bir zaman negatif olmaz.
  - Türevler öğrenme kurallarında kolaylıkla kullanılmaktadır.

Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları



x1	x2	v1	v2	v3	h1	h2	h3	v4	y
0	0								
0	1								
1	0								
1	1								

Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları

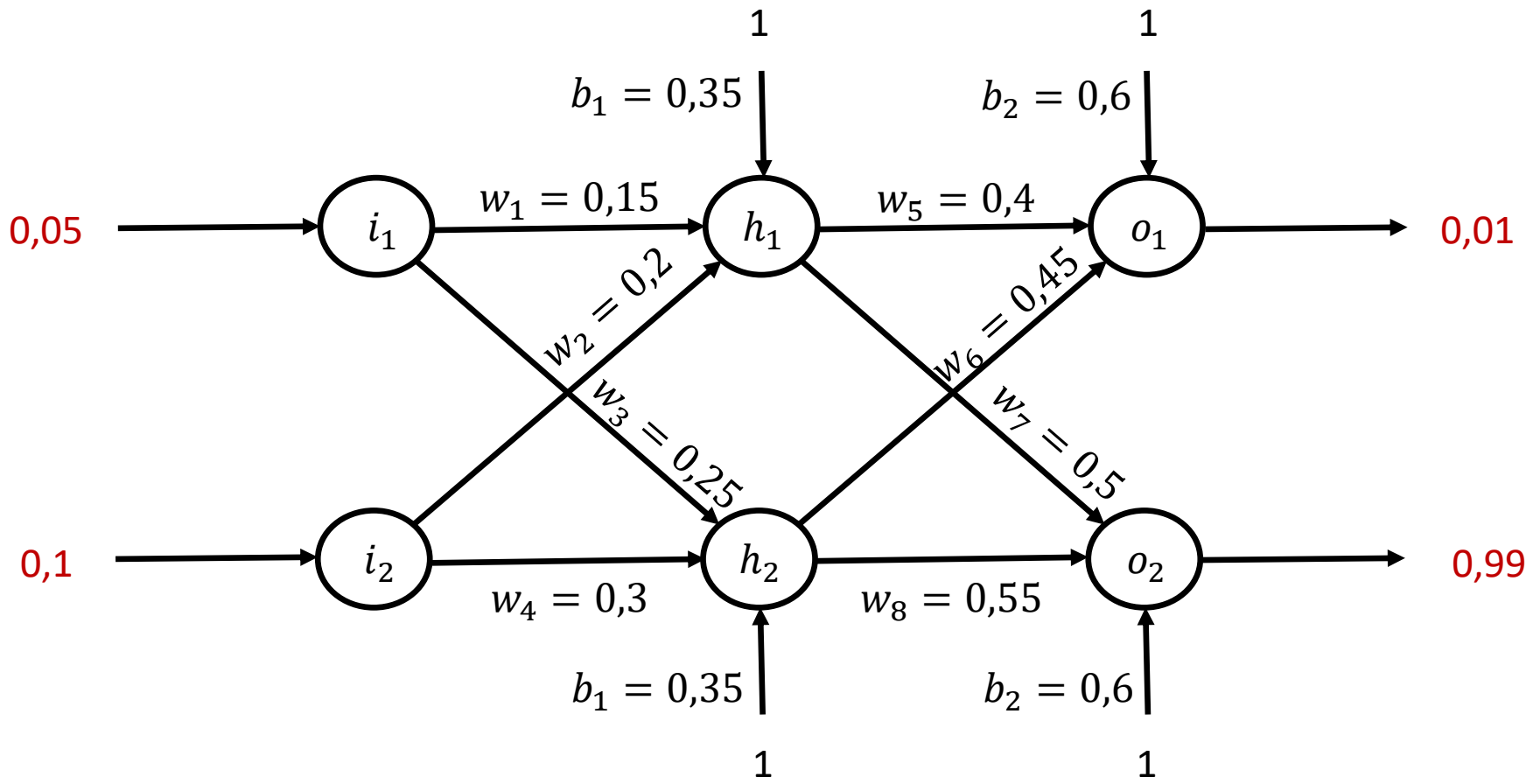


x1	x2	v1	v2	v3	h1	h2	h3	v4	y
0	0	2	2	2	1	1	1	0,5	1
0	1	3	1	3	1	1	1	0,5	1
1	0	0	2	4	0	1	1	-0,5	0
1	1	1	1	5	1	1	1	0,5	1

Geri Yayılım Algoritması

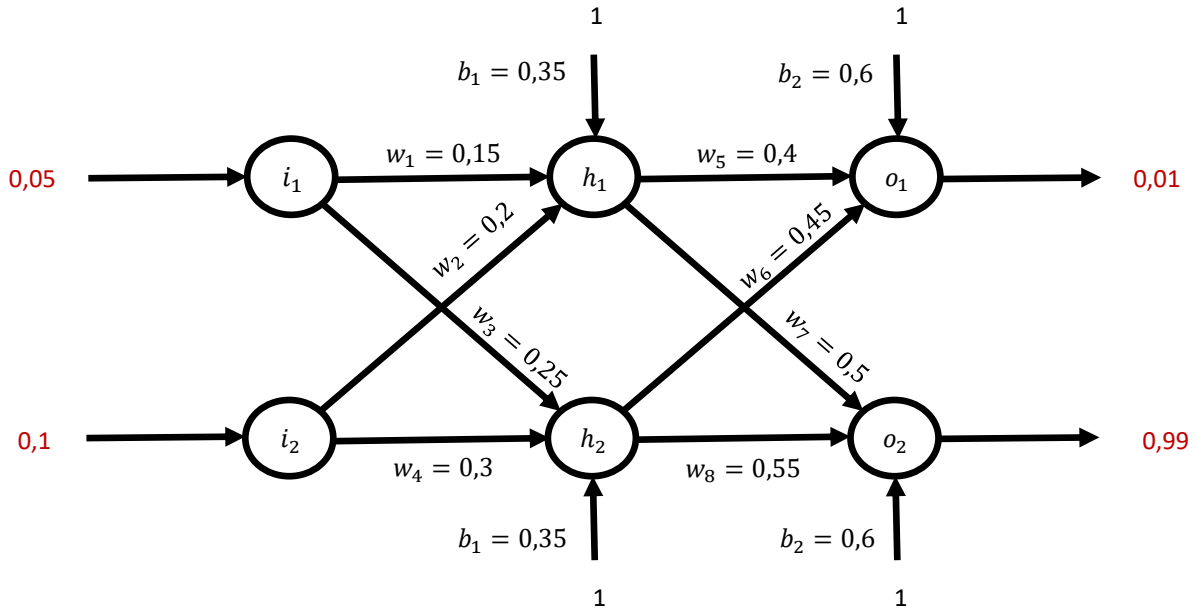
- Geri yayılım algoritması, ileri beslemeli yapay sinir ağları için çok popüler ve önemli bir algoritmadır.
- Bu popülariteden dolayı ileri beslemeli yapay sinir ağları sıklıkla geriye yayılım sinir ağları (back propagation neural networks) olarak da adlandırılırlar.
- Geriye yayılım algoritması, denetlemeli öğrenme algoritmalarındandır.
- İstenen çıkışlar ile gerçek çıkışlar arasındaki hata gizli katmanlara doğru geriye yayılır.

Geri Yayılım Algoritması



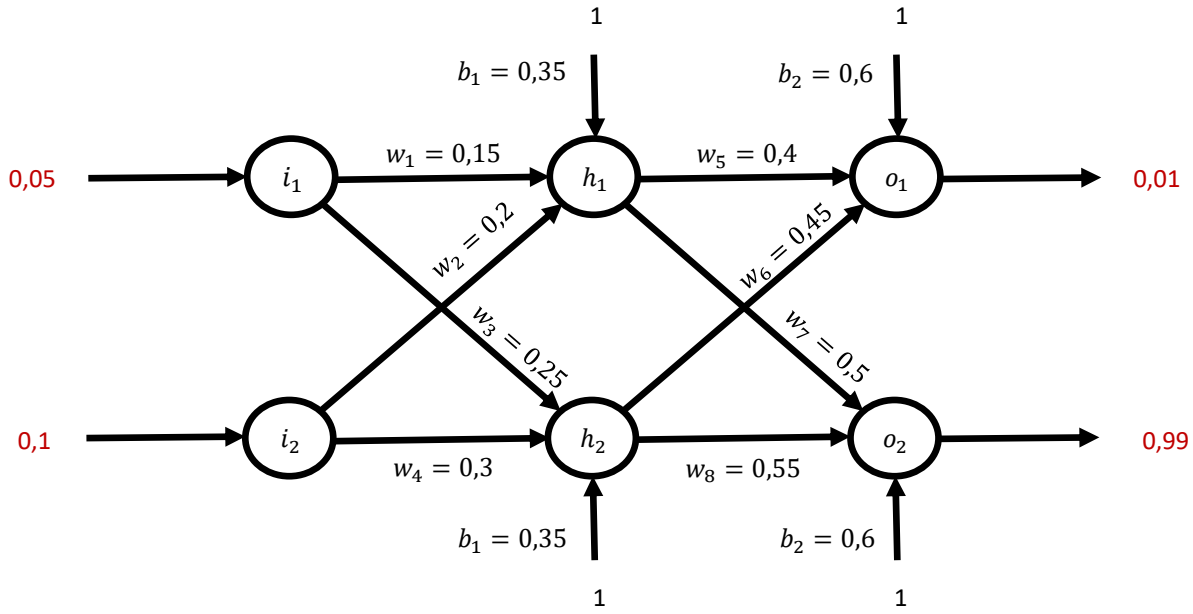
Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



- $net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$

  $= 0{,}15 * 0{,}05 + 0{,}2 * 0{,}1 + 0{,}35 * 1$

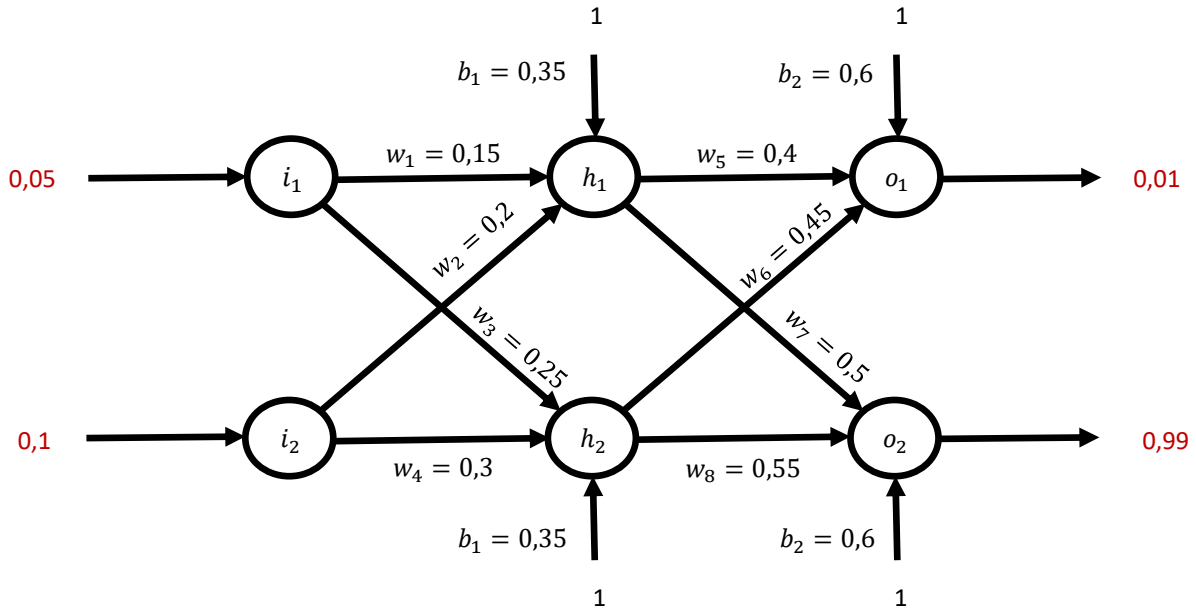
  $= 0{,}3775$

- $out_{h1} = \frac{1}{1+e^{-net_{h1}}}$

  $= \frac{1}{1+e^{-0{,}3775}} = 0{,}5932$

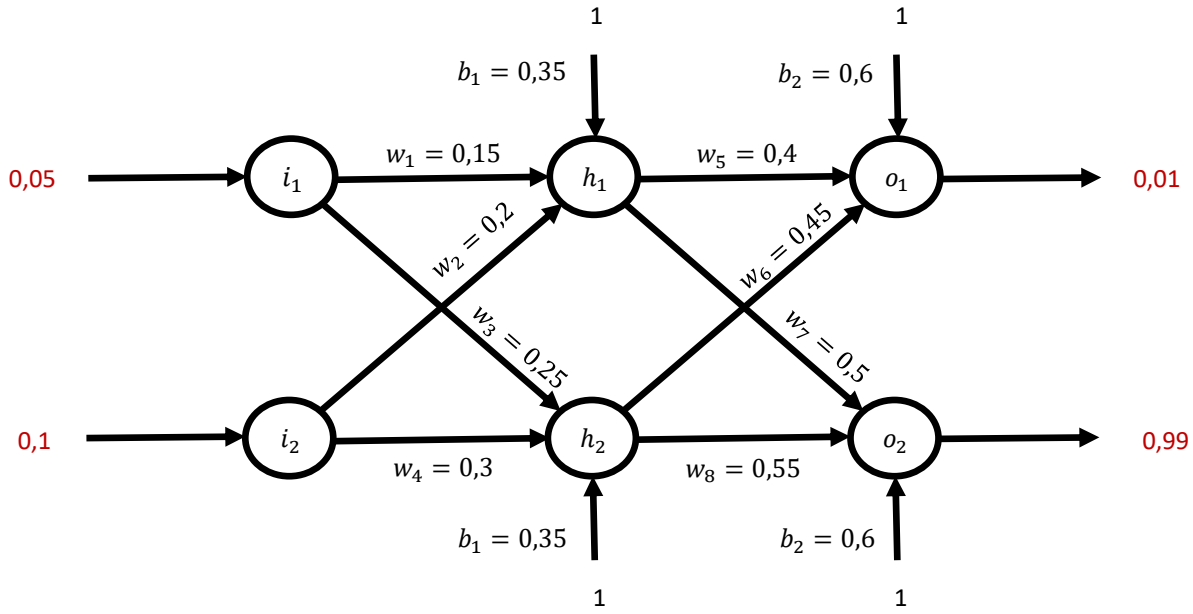
Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



- $net_{h2} = w_3 * i_1 + w_4 * i_2 + b_1 * 1$

  $= 0{,}25 * 0{,}05 + 0{,}3 * 0{,}1 + 0{,}35 * 1$

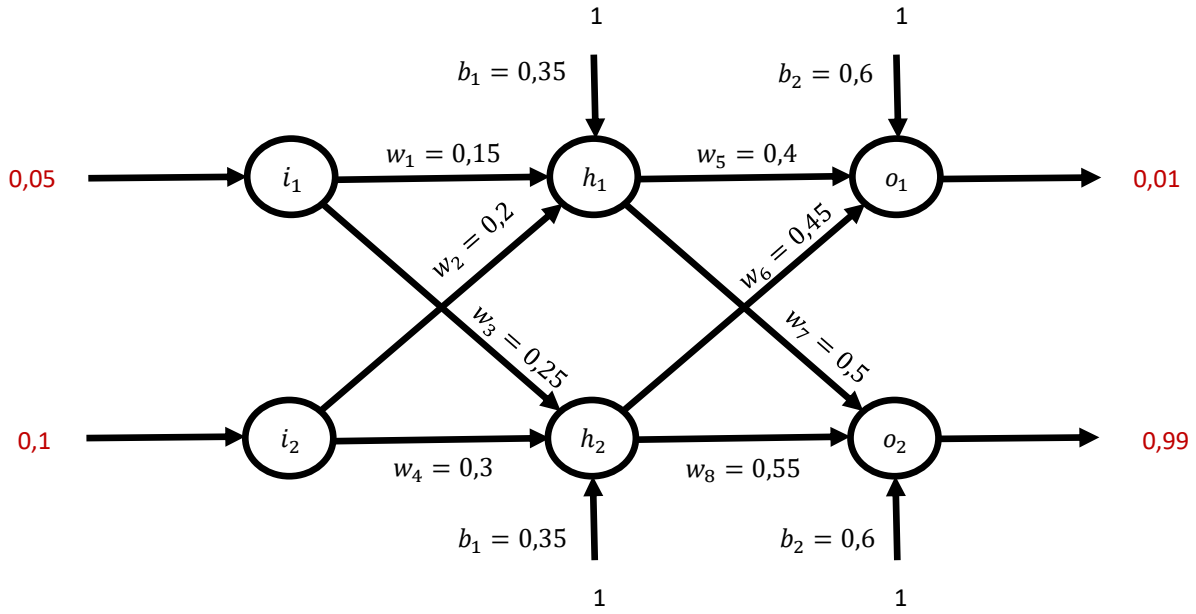
  $= 0{,}3925$

- $out_{h2} = \frac{1}{1+e^{-net_{h2}}}$

  $= \frac{1}{1+e^{-0{,}3925}} = 0{,}5968$

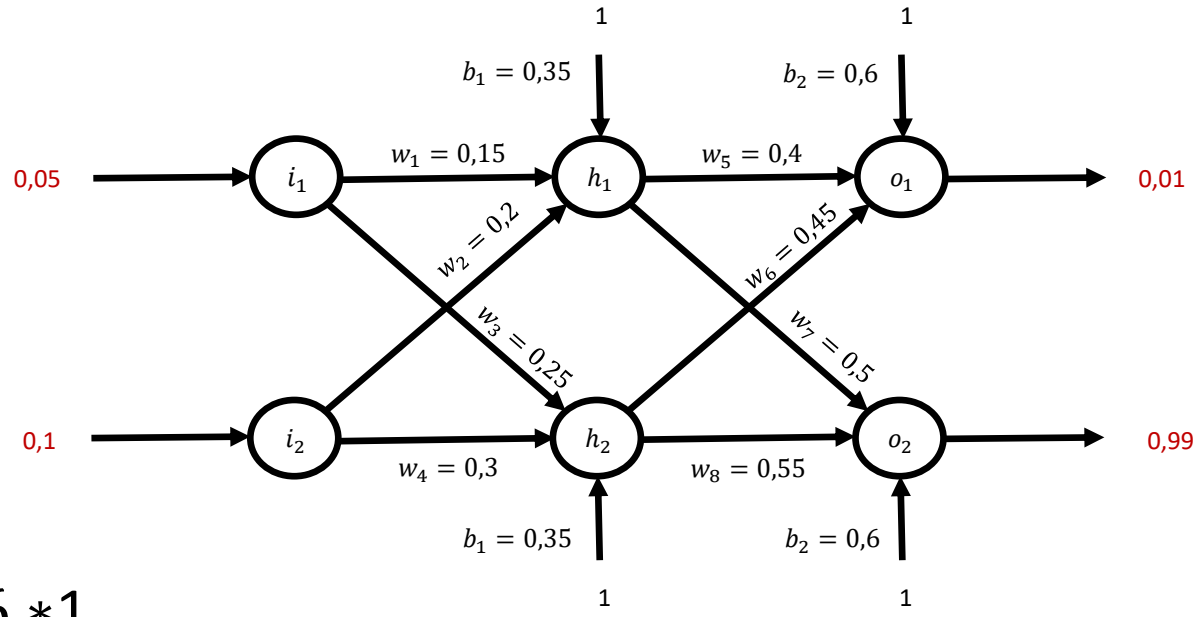
Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



- $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

  $= 0{,}4 * 0{,}5932 + 0{,}45 * 0{,}5968 + 0{,}6 * 1$

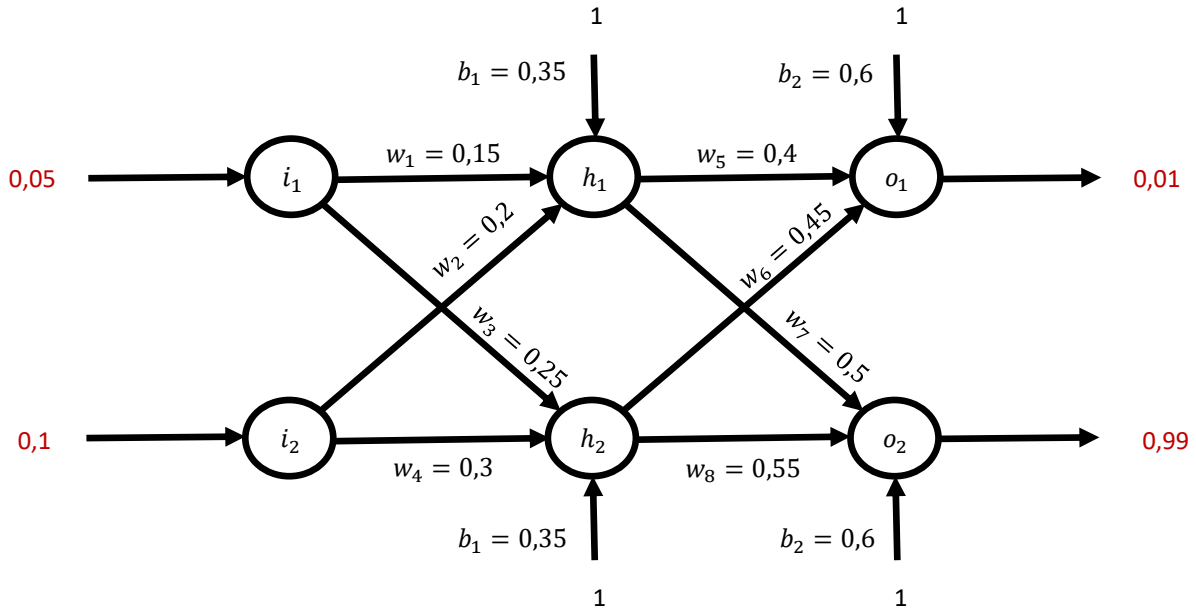
  $= 1{,}1059$

- $out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

  $= \frac{1}{1+e^{-1{,}1059}} = 0{,}7513$

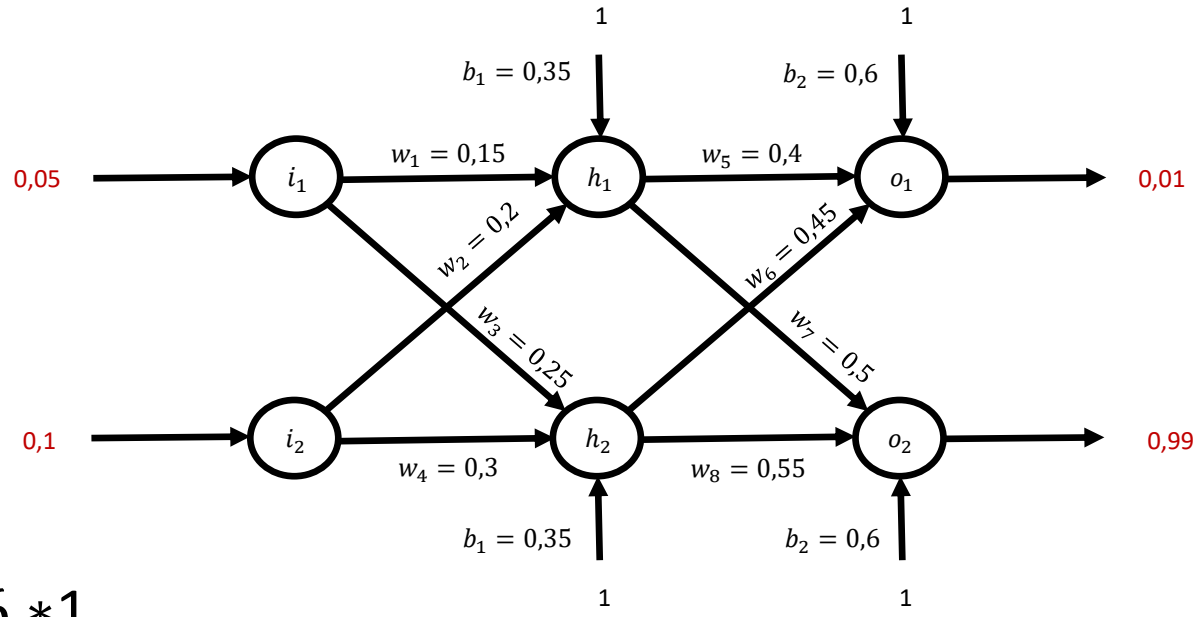
Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



Geri Yayılım Algoritması

(ileri yayılım hesaplamaları)



- $net_{o2} = w_7 * out_{h1} + w_8 * out_{h2} + b_2 * 1$

  $= 0{,}5 * 0{,}5932 + 0{,}55 * 0{,}5968 + 0{,}6 * 1$

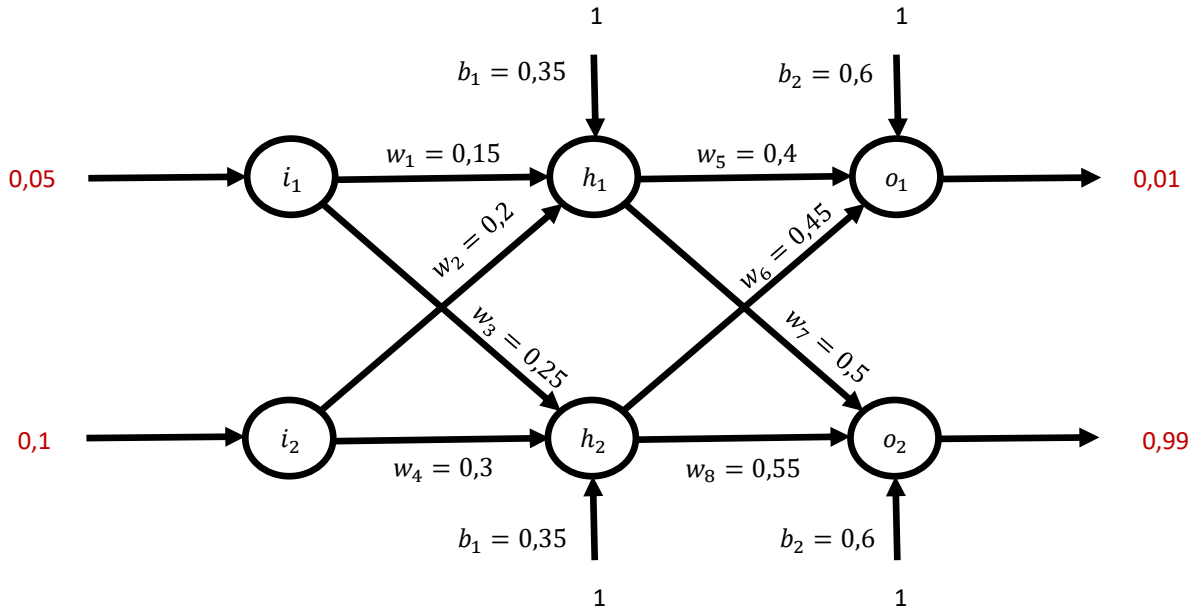
  $= 1{,}2248$

- $out_{o2} = \frac{1}{1+e^{-net_{o2}}}$

  $= \frac{1}{1+e^{-1{,}2248}} = 0{,}7729$

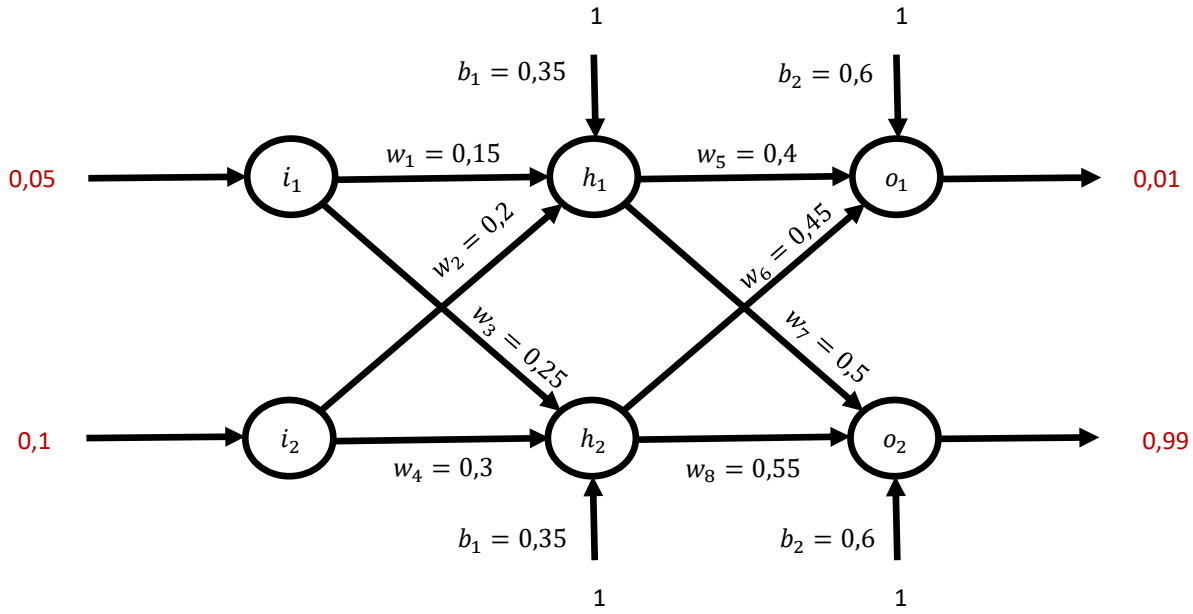
Geri Yayılım Algoritması

(toplam hata hesaplamaları)



Geri Yayılım Algoritması

(toplam hata hesaplamaları)



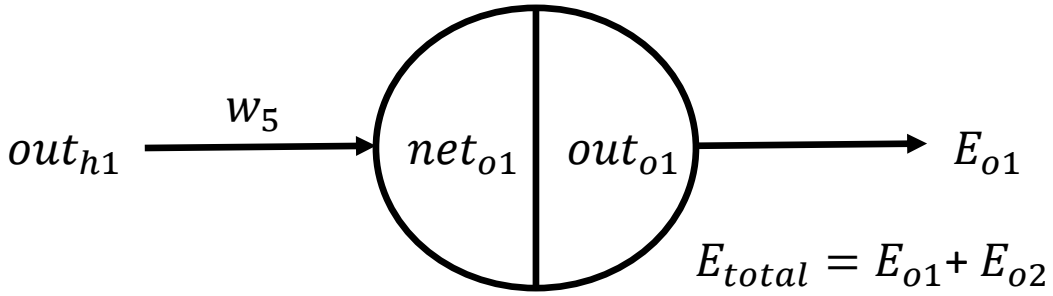
- $E_{total} = \Sigma \frac{1}{2}(target\ - output)^2$

- $E_{o1} = \frac{1}{2}(target_{o1}\ - out_{o1})^2 = \frac{1}{2}(0{,}01\ - 0{,}7513)^2 = 0{,}2748$
- $E_{o2} = \frac{1}{2}(target_{o2}\ - out_{o2})^2 = \frac{1}{2}(0{,}99\ - 0{,}7729)^2 = 0{,}0235$

- $E_{total} = E_{o1} + E_{o2} = 0{,}2748 + 0{,}0235 = 0{,}2983$

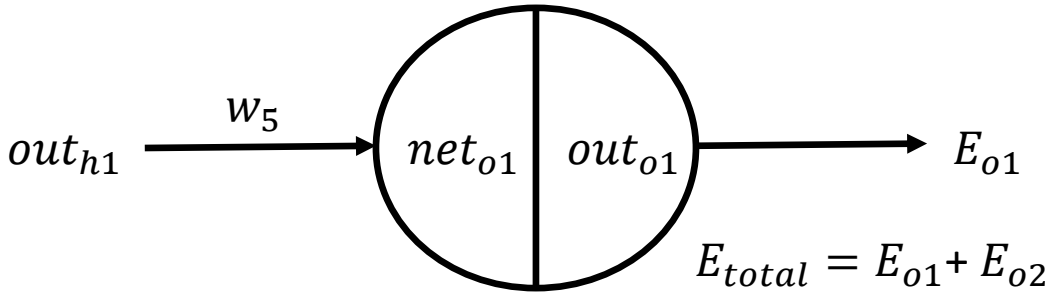
Geri Yayılım Algoritması

- w_5 ağırlığındaki bir değişiklik toplam hatayı ne kadar etkiler?



Geri Yayılım Algoritması

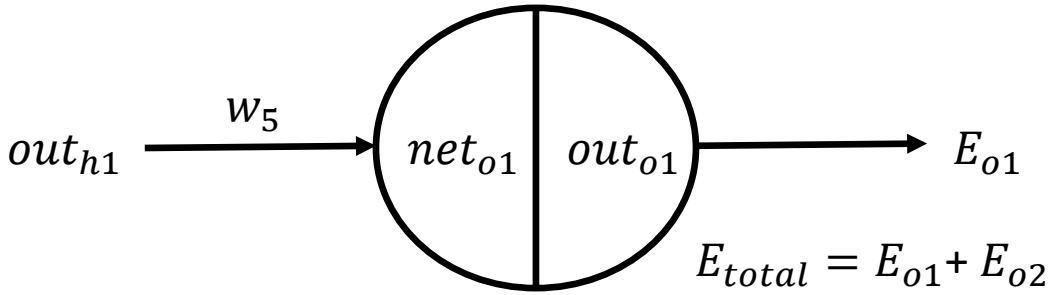
- w_5 ağırlığındaki bir değişiklik toplam hatayı ne kadar etkiler?



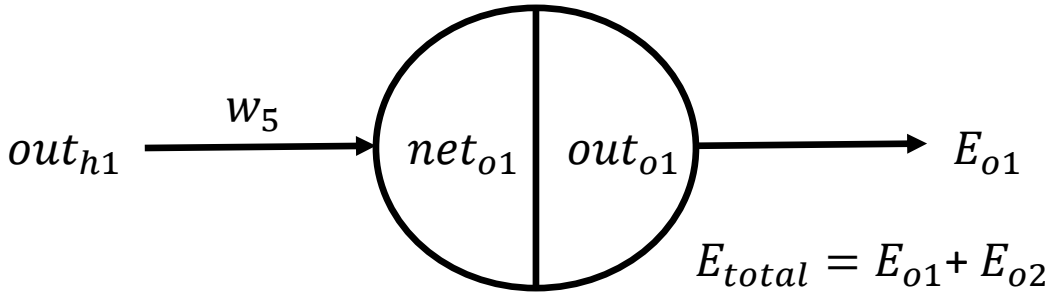
- E_{total} 'in w_5 'e göre kısmi türevi: $\frac{\partial E_{total}}{\partial w_5}$

- Zincir kuralı (chain rule): $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial net_{o1}}{\partial w_5} \frac{\partial out_{o1}}{\partial net_{o1}} \frac{\partial E_{total}}{\partial out_{o1}}$

Geri Yayılım Algoritması



Geri Yayılım Algoritması

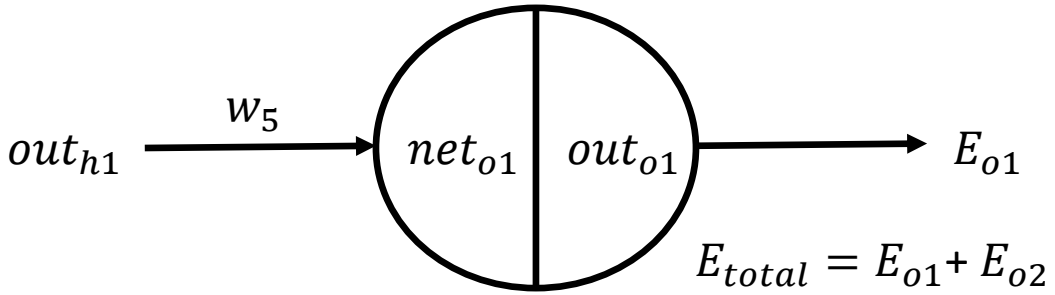


- $\frac{\partial E_{total}}{\partial out_{o1}} = (\frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2)'$

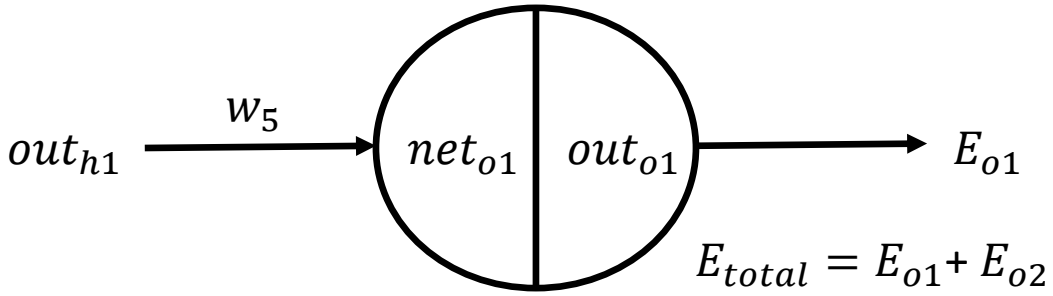
  $= 2\frac{1}{2}(target_{o1} - out_{o1})(-1) + 0 = out_{o1} - target_{o1}$

  $= 0{,}7513 - 0{,}01 = 0{,}7413$

Geri Yayılım Algoritması



Geri Yayılım Algoritması



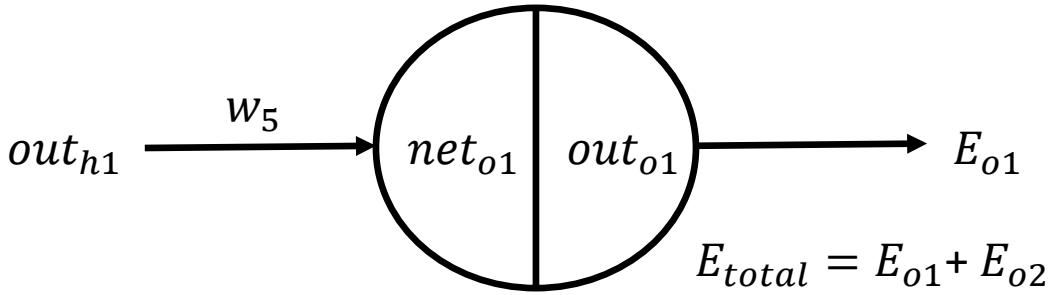
- $\frac{\partial out_{o1}}{\partial net_{o1}} = ?$

- $out_{o1} = \frac{1}{1+e^{-net_{o1}}} \rightarrow (out_{o1})' = \frac{e^{-net_{o1}}}{(1+e^{-net_{o1}})^2} = \frac{e^{-net_{o1}+1-1}}{(1+e^{-net_{o1}})^2} = \frac{1}{1+e^{-net_{o1}}} - \frac{1}{(1+e^{-net_{o1}})^2}$

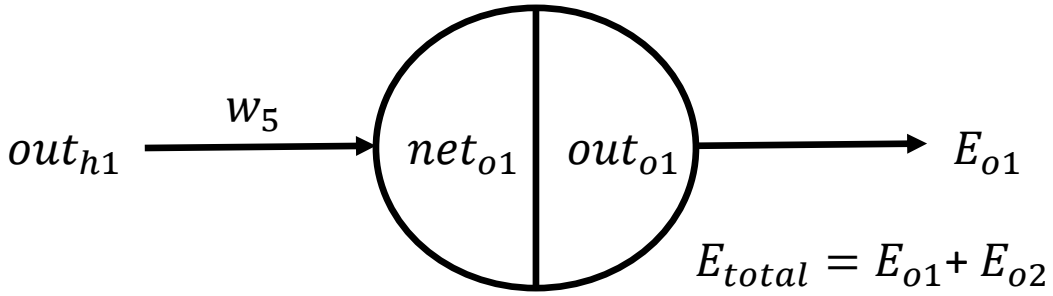
$$= out_{o1}(1- out_{o1})$$

- $\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1- out_{o1}) = 0{,}7513(1\text{-}0{,}7513) = 0{,}1865$

Geri Yayılım Algoritması

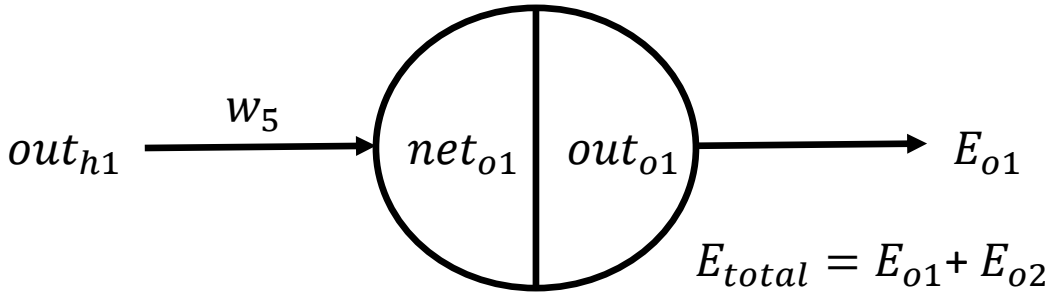


Geri Yayılım Algoritması



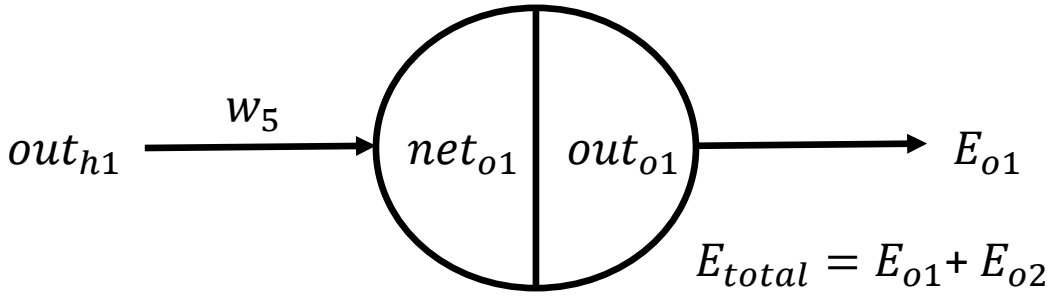
- $\frac{\partial net_{o1}}{\partial w_5} = ?$
- $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$
- $\frac{\partial net_{o1}}{\partial w_5} = out_{h1} = 0{,}5932$

Geri Yayılım Algoritması



Geri Yayılım Algoritması

- $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial net_{o1}}{\partial w_5} \frac{\partial out_{o1}}{\partial net_{o1}} \frac{\partial E_{total}}{\partial out_{o1}}$



= 0,5932 * 0,1865 * 0,7413 = 0,082

- $\frac{\partial E_{total}}{\partial w_5} = (out_{o1} - target_{o1})\, out_{o1}\, (1 - out_{o1})\, out_{h1}$

- $\frac{\partial E_{total}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial out_{o1}} \frac{\partial out_{o1}}{\partial net_{o1}} = \delta_{o1} \rightarrow \frac{\partial E_{total}}{\partial w_5} = \delta_{o1}\, out_{h1}$

- $w_5^{+} = w_5 - \eta \frac{\partial E_{total}}{\partial w_5} = 0,4 - 0,5*0,082 = 0,3589$

YSA’nın genel özellikleri

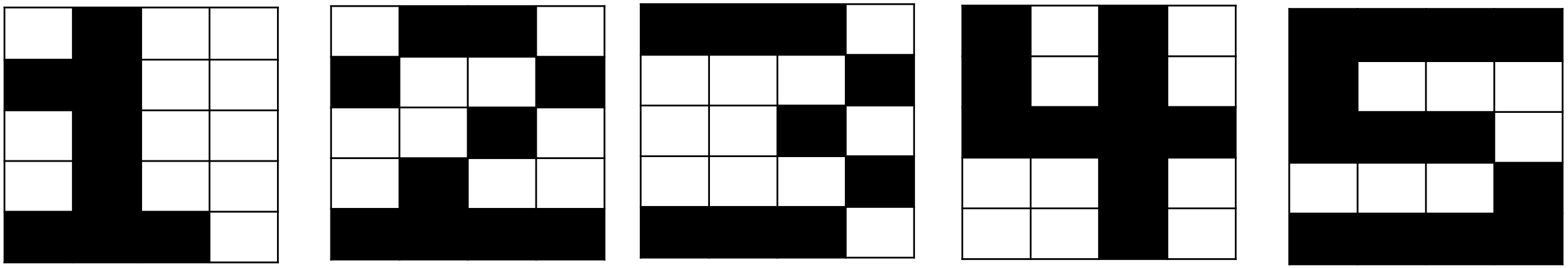
- YSA’lar öğrenmenin dijital ortamdaki bir modelidir.
- YSA’dan bir sonuç elde etmek için ağ, uygun örnekler ile eğitilmelidir.
- YSA eksik ve hatalı bilgilere karşı dayanıklıdır.
- Algoritmik bir çözüm olmayan durumlarda yaygın olarak kullanılır ve başarılı sonuçlar üretir.

YSA’nın olumsuz yönleri

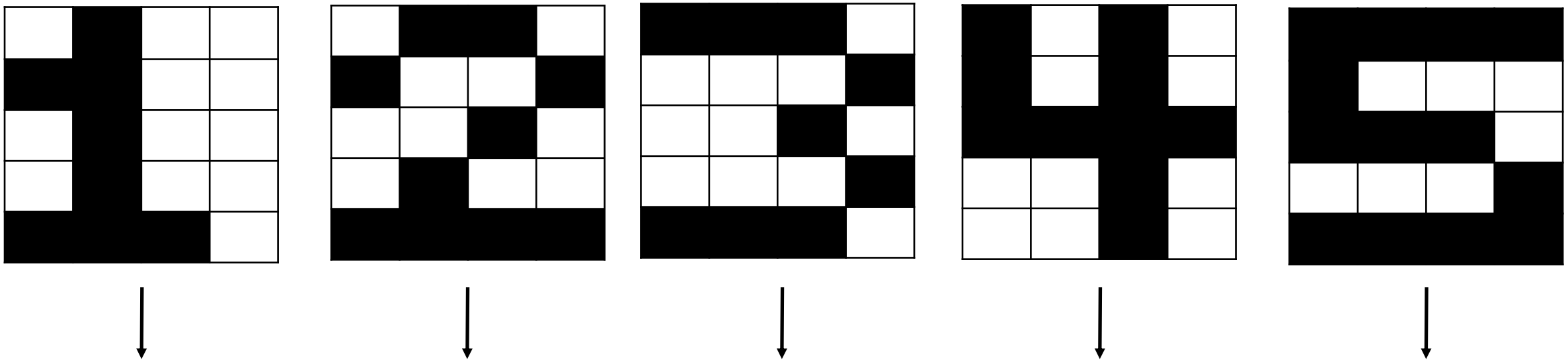
- YSA’lar yerel çözümlere takılabilir.
- Uygun ağ yapısı deneme yanılma yöntemi ile bulunur.
- Gizli katmandaki nöron sayısı uygulamanın hızını önemli derecede etkilemektedir: Örneğin karmaşık görüntülerin analizi için az sayıda nöron içeren ağ yeterli olmazken; çok fazla nöron içeren gizli katman ise işlem hızını büyük ölçüde azaltır.
- YSA eğitiminde eğitim uzayı büyük önem taşımaktadır. Uygun seçilmeyen örnekler ile başarım düşer.

YSA örnek problem

- Yapay sinir ağları ile karakter algılama örneği
  - Elle yazılmış karakterlerin otomatik olarak algılanması uzun zamandır üzerinde çalışılan bir konudur.
  - Karakter tanıma YSA ile yapılabilmektedir. Örneğimizde 0 ile 9 arasındaki karakterler ağa öğretilmektedir.
  - Öncelikle karakterler aşağıdaki gibi matrissel olarak temsil edilir.



YSA örnek problem



0	1	0	0
1	1	0	0
0	1	0	0
0	1	0	0
1	1	1	0

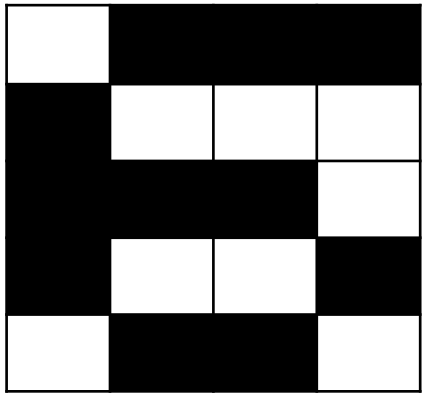
0	1	1	0
1	0	0	1
0	0	1	0
0	1	0	0
1	1	1	1

1	1	1	0
0	0	0	1
0	0	1	0
0	0	0	1
1	1	1	0

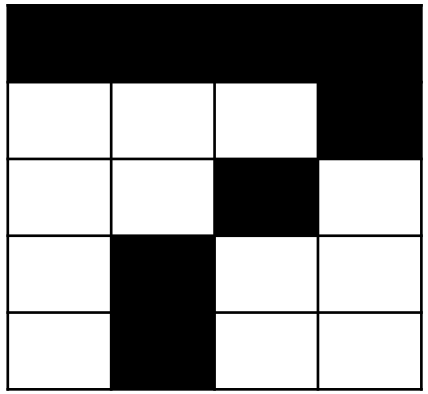
1	0	1	0
1	0	1	0
1	1	1	1
0	0	1	0
0	0	1	0

1	1	1	1
1	0	0	0
1	1	1	0
0	0	0	1
1	1	1	1

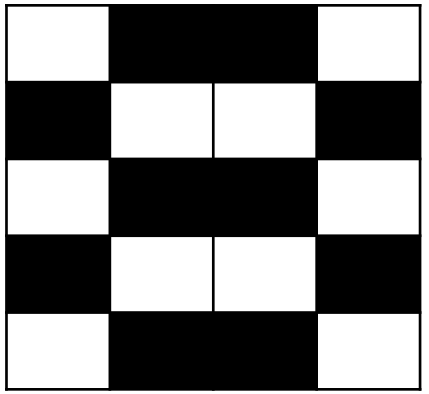
YSA örnek problem



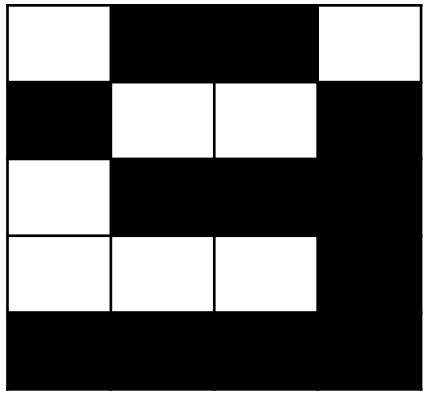
0	1	1	1
1	0	0	0
1	1	1	0
1	0	0	1
0	1	1	0



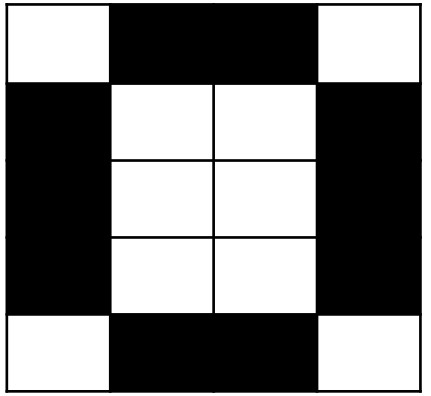
1	1	1	1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	0	0



0	1	1	0
1	0	0	1
0	1	1	0
1	0	0	1
0	1	1	0



0	1	1	0
1	0	0	1
0	1	1	1
0	0	0	1
1	1	1	1



0	1	1	0
1	0	0	1
1	0	0	1
1	0	0	1
0	1	1	0

YSA örnek problem

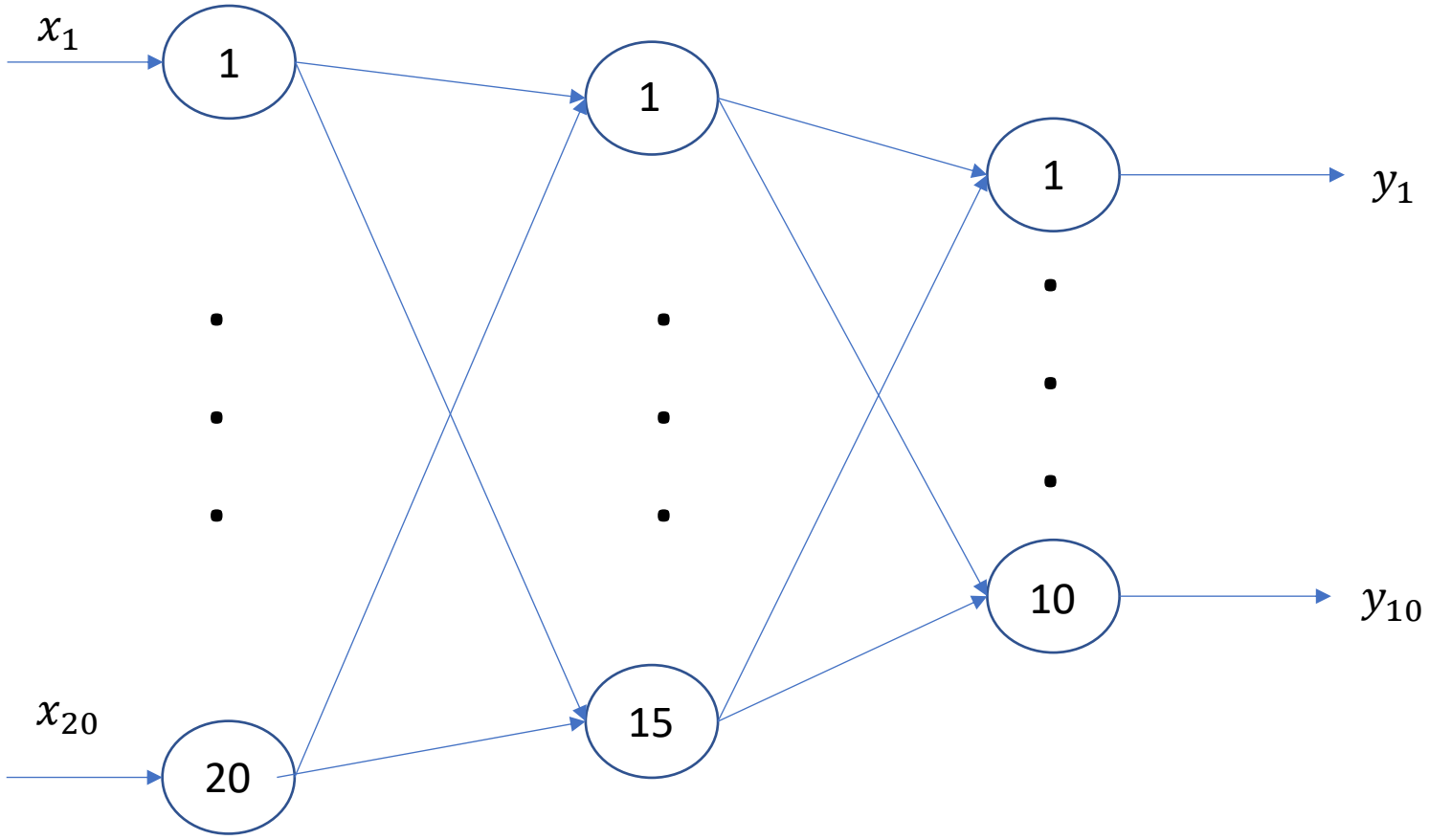
- Giriş karakterleri 4x5 lik matrislerle gösterildiği için ağın giriş katmanındaki sinir sayısı 20 olmalıdır.
- Matrislerdeki bilgiler 1. satırdan başlamak üzere her bir hücredeki değer giriş katmanının her bir sinirine gelecek şekilde ağa uygulanır.
- Çıkış temsili 10 bitlik ikili sayılarla olacak şekilde tasarım yapılacaktır. Bu da çıkış katmanındaki sinir sayısının 10 adet olması anlamına gelir.
- 10 bitlik ikili sayının temsili gösterimi aşağıdaki gibidir.
- Örneğin çıkışta sinirler sırasıyla 1000000000 ürettiyse bu çıkış için sonuç 1'dir.

1	2	3	4	5	6	7	8	9	10	çıkış
1	0	0	0	0	0	0	0	0	0	**1**
0	1	0	0	0	0	0	0	0	0	**2**
0	0	1	0	0	0	0	0	0	0	**3**
0	0	0	1	0	0	0	0	0	0	**4**
0	0	0	0	1	0	0	0	0	0	**5**
0	0	0	0	0	1	0	0	0	0	**6**
0	0	0	0	0	0	1	0	0	0	**7**
0	0	0	0	0	0	0	1	0	0	**8**
0	0	0	0	0	0	0	0	1	0	**9**
0	0	0	0	0	0	0	0	0	1	**0**

YSA örnek problem

- Çıkışlar bu şekilde tasarlandıktan sonra yapılması gereken gizli katmandaki sinir sayısını belirlemektir.
- Gizli katman sinir sayısı genellikle giriş sinir sayısı ile çıkış sinir sayısı arasında bir değer olarak belirlenir.
- Ayrıca öğrenme durumuna göre deneme yanılma yoluyla da bir değer verilebilir.
- Sinirlerin sayısı ne kadar fazla tutulursa ağın eğitimden sonra hatırlama yeteneği o kadar iyi olur, fakat sayının fazla olması eğitim süresini uzatır.
- Bu örnekte gizli katman sayısı 15 olarak belirlenmiştir.

YSA örnek problem



Yandaki ağda;

- Giriş sinir sayısı:20
- Giriş katmanı ile gizli katman arasında 300 bağlantı vardır
- Gizli katman ile çıkış katmanı arasında 150 bağlantı vardır.
- Çıkış sinir sayısı:10
- Örnekte 4x5 lik bir görüntü yerine 224x224 gibi daha büyük boyutlu bir görüntü kullanılsaydı giriş gizli katmanlar ve bunların arasındaki bağlantı sayısı çok büyük sayılara çıkacak ve bu da ağın eğitilmesini çok zorlaştıracaktı.

Ek kaynaklar:

- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/