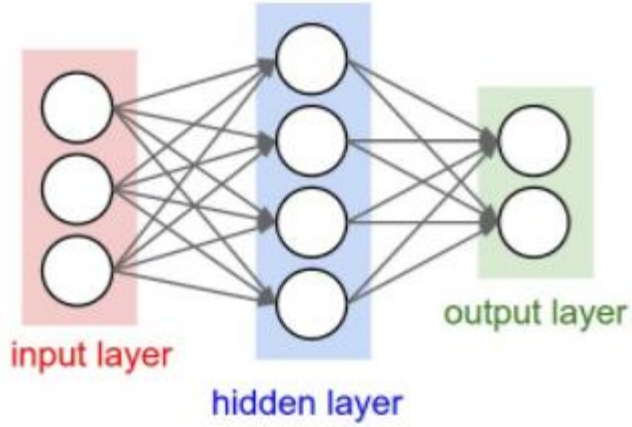
YBS519
Yapay Zeka ve Uygulamaları

Bölüm 3

Yapay Sinir Ağları



Gaziantep Üniversitesi

Yönetim Bilişim Sistemleri, Tezsiz Yüksek Lisans Programı

http://www1.gantep.edu.tr/~bingul/ai

Mart 2022

İçerik

1. Yapay Zeka ve Makine Öğrenmesi
2. Yapay Sinir Ağlarına Giriş
3. Basit Desen Sınıflandırma (Pattern Classification)
4. Hebb Kuralı ile Desen Eşleştirme (Pattern Association)
5. Matlab ile İleri Uygulamalar
   - Fitting app
   - Pattern Recognition app
   - Clustering app
   - Time Series app

1. Kısım

Yapay Zeka ve Makine Öğrenme

YZ ve MÖ

Soru: Yapay Zeka ve Makine Öğrenmesi arasındaki fark nedir?

- YZ ve MÖ birbiri le sıkı bağlı ama farklı kavramlardır. **YZ ≠ MÖ**
- YZ, bilgisayar sistemlerinin insan benzeri düşünme ve davranış sergilemesi amacıyla tasarlanan bir disiplindir. YZ, bilgisayar sistemlerinin **problem çözme**, **karar verme**, **öğrenme**, **dil işleme**, **algılama**, **bilgi temsili**, **planlama** ve **kontrol** gibi insan benzeri yeteneklerini modellenmesi ile ilgilenir.
- MÖ, bilgisayar sistemlerinin **veri analizi** yaparak örüntüler keşfedebilmesi ve bu örüntülerden öğrenerek **tahminler** yapabilmesi için kullanılan bir **teknikler ve algoritmalar** topluluğudur.
- Özetle:
  YZ: insan benzeri zekaya sahip bilgisayar sistemleri oluşturma hedefi taşır
  MÖ: bu hedefe ulaşmak için kullanılan bir araçtır.
  MÖ, YZ'nın alt kümesidir.

Yapay Zeka

- Dil işleme
- Görsel analiz
- İşitsel analiz
- Hareket algılama
- Planlama
- Karar verme
- Öğrenme
  * Yönlendirmeli (supervised)
  * Yönlendirmesiz (unsupervised)
  * Derin Öğrenme

2. Kısım

Yapay Sinir Ağlarına Giriş

YSA

- Yapay sinir ağları (YSA), insan beyninin bilgi işleme tekniğinden esinlenerek geliştirilmiş bir bilgi-işlem teknolojisidir.

- YSA ile basit biyolojik sinir sisteminin çalışma şekli taklit edilir. Taklit edilen sinir hücreleri nöronlar içerir. Bu nöronlar çeşitli şekillerde birbirlerine bağlanarak bir ağ oluşturur.

- Bu ağlar öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahiptirler.

YSA: Nerede Kullanılıyor?

- Sinyal İşleme
- Elektronik Kontrol
- Örüntü Algılama (Ses ve görüntü tanıma)
- Tıp (Tanı ve tedavi)
- Seslendirme (1000 kelimeyi öğrenip yeni kelimeleri telafuz)
- Veri Analizi (bağımsız girdiler ve çıktılar arasındaki ilişki)
- ...

YSA: Kim Geliştirdi?

- **1943: İlk fikir**

McCulloch, W. S., & W. Pitts. (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics*, 5:115–133. Reprinted in Anderson & Rosenfeld [1988], pp. 18–28.

- **1949: İlk öğrenme kuralı**

Hebb, D. O. (1949). *The Organization of Behavior*. New York: John Wiley & Sons. Introduction and Chapter 4 reprinted in Anderson & Rosenfeld [1988], pp. 45–56.

- **1958: Beynin bilgisayar modeli**

Von Neumann, J. (1958). *The Computer and the Brain*. New Haven: Yale University Press. Pages 66–82 are reprinted in Anderson & Rosenfeld [1988], pp. 83–89.

- **1982: Donanımsal YSA**

Hopfield, J. J. (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceeding of the National Academy of Scientists*, 79:2554–2558. Reprinted in Anderson & Rosenfeld [1988], pp. 460–464.

Farhat, N. H., D. Psaltis, A. Prata, & E. Paek. (1985). "Optical Implementation of the Hopfield Model." *Applied Optics*, 24:1469–1475. Reprinted in Anderson & Rosenfeld [1988], pp. 653–660.

Nöron = Sinir Hücresi

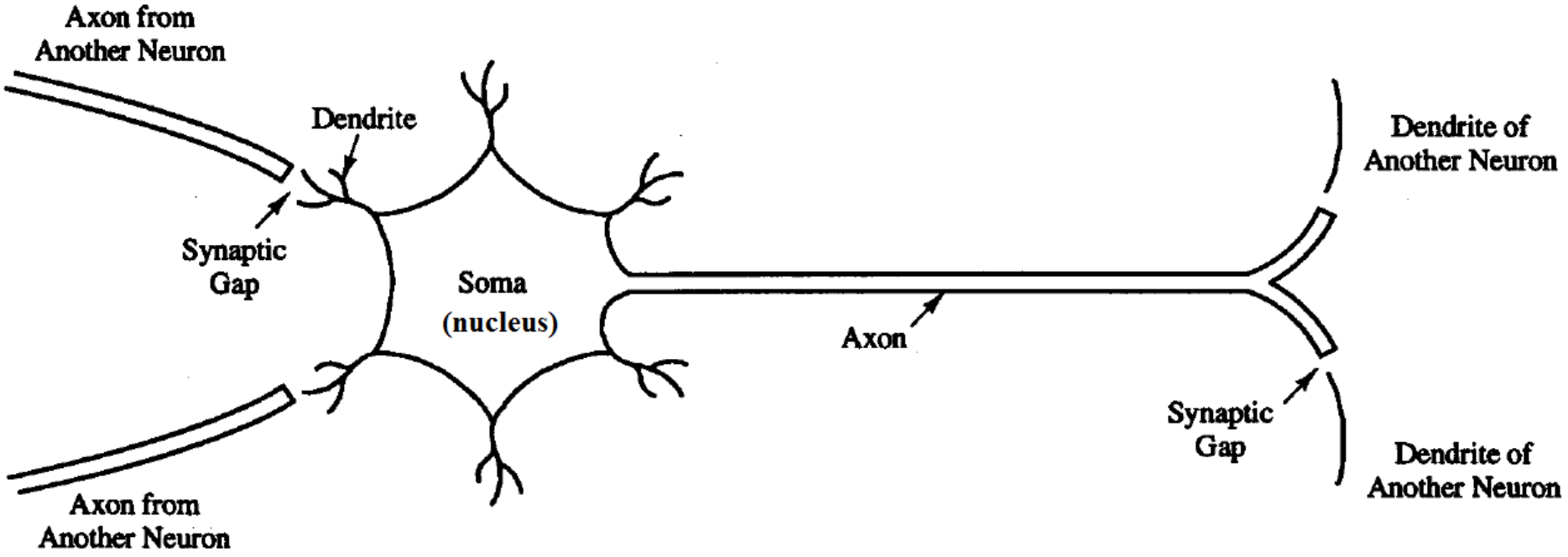
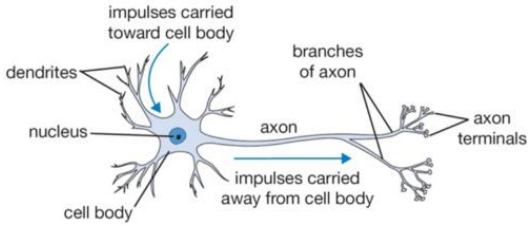


Figure 1.3 Biological neuron.

Nöron



Dendrite:

Diğer hücrelerden gelen işaretleri toplayan elektriksel anlamda pasif kollardır. Sistem girişidir.

Synapse:

Hücrelerin aksonlarının diğer dentritlerle olan bağlantısını sağlar.

Nucleus:

Akson boyunca işaretlerin periyodik olarak yeniden üretilmesini sağlar.

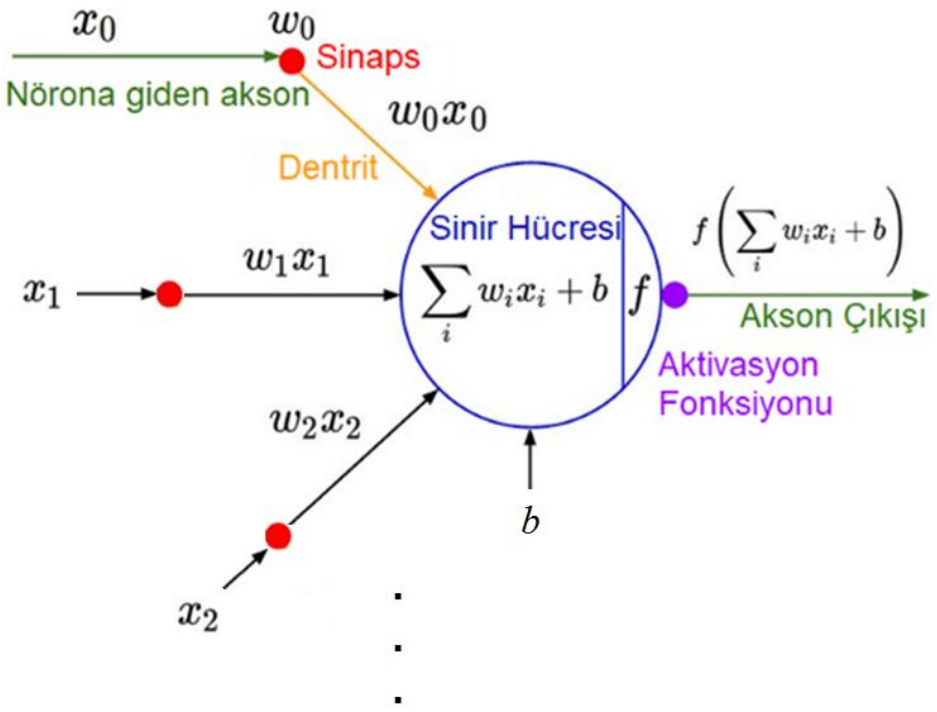
Axon:

Çıkış darbelerinin üretildiği elektriksel aktif gövdedir ve gövde üzerinde iletim tek yönlüdür. Sistem çıkışıdır.

Yapay Nöron

Perceptron:

Yapay sinir ağının en küçük parçasıdır. Doğrusal (lineer) bir fonksiyonla ifade edilmektedir.



$$y = \sum_{i=0}^{n} w_i x_i + b = w_0 x_0 + w_1 x_1 + \cdots w_n x_n + b$$

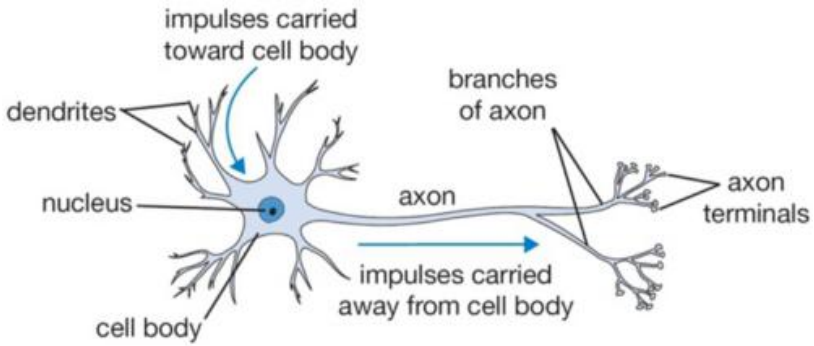
x: Girdi değeri, bağımsız değişkenler.
w: Ağırlık (weight) parametresi
b: Kayma (bias) değeri
y: Çıkış değeri, girdiye ait skoru verir.

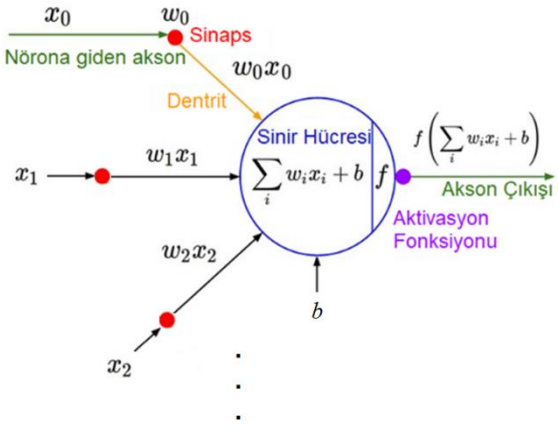
Aktivasyon fonksiyonu:

$$f(y) = f\left(\sum w_i x_i + b\right)$$

YSA ve Sinir sistemi elemanlarının karşılaştırılması

Sinir Sistemi Elemanı	Yapay Sinir Ağı Elemanı
Sinir	Yapay Sinir Hücresi
Sinaps	Ağırlıklar
Dendrit	Toplama Fonksiyonu
Hücre Gövdesi	Aktivasyon Fonksiyonu
Aksonlar	Çıkış





Yapay Sinir Ağı

- Birden çok sayıda perceptron bir araya gelip ağ oluşturulur:
- YSA (veya Derin Öğrenme) modelinde yapılan temel işlem; modelin en iyi skoru vereceği **w** ve **b** parametrelerini hesaplamaktır.

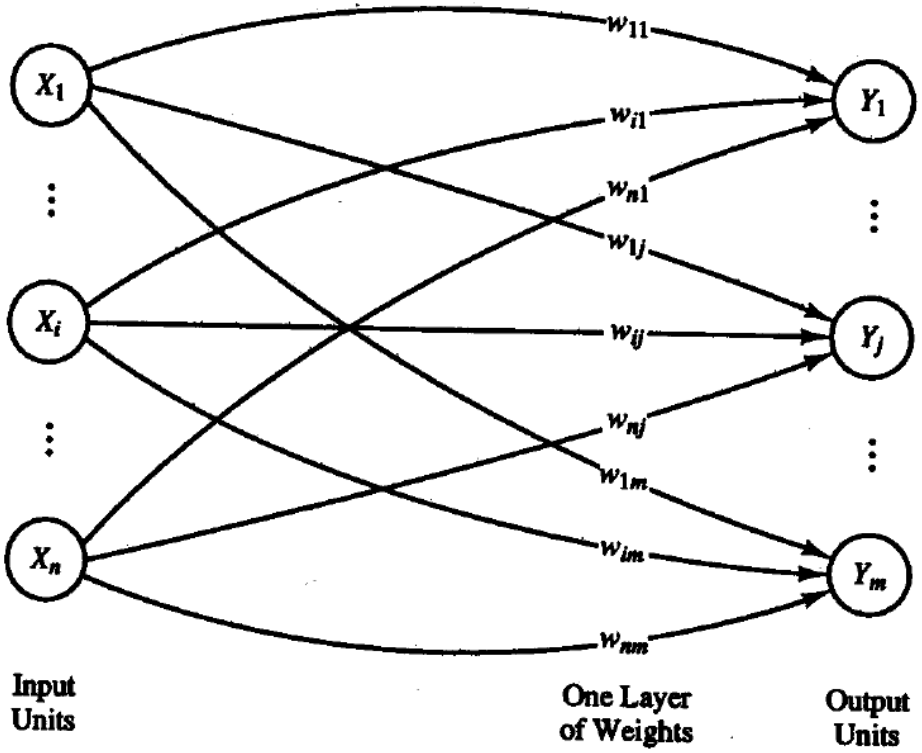


Figure 1.4 A single-layer neural net.

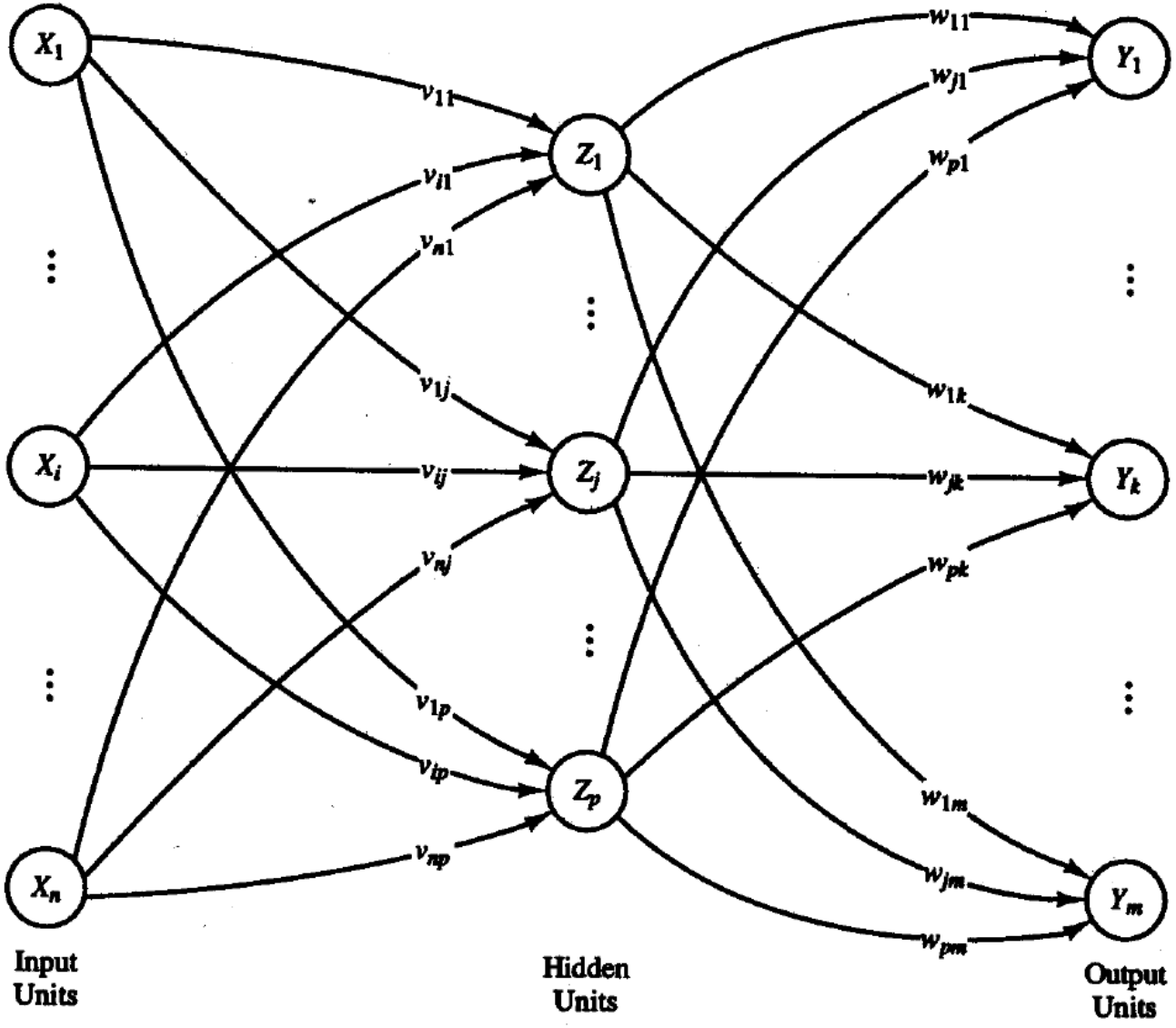
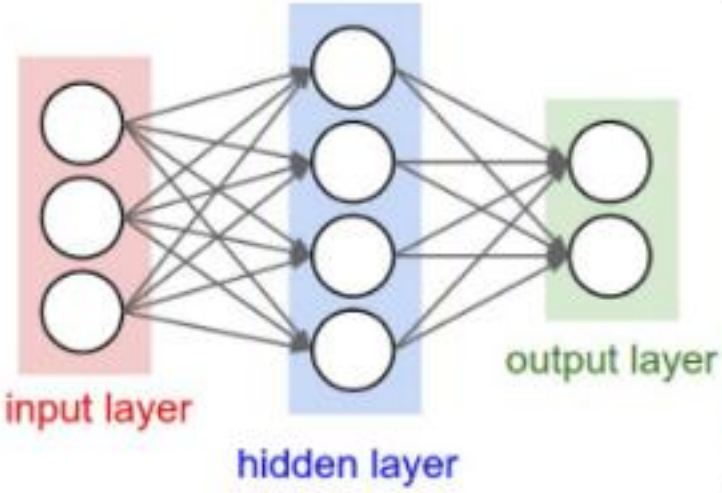
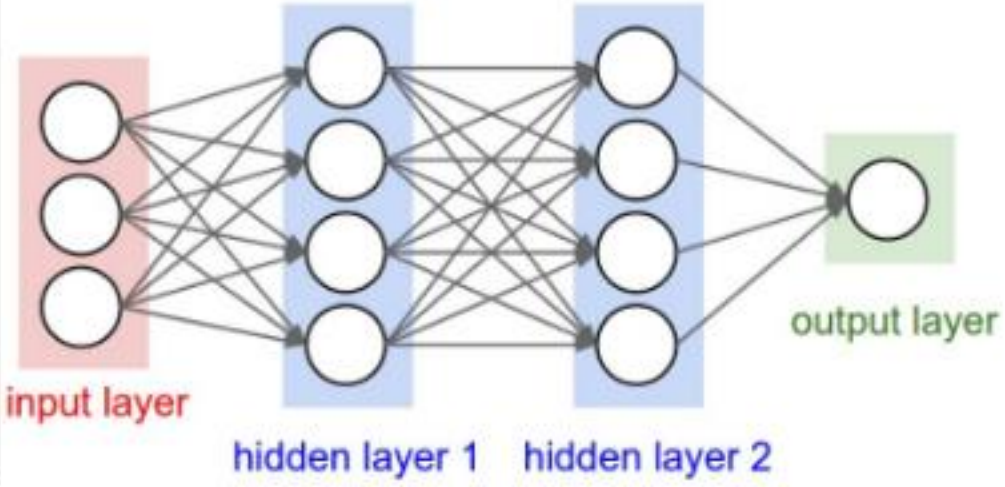


Figure 1.5 A multilayer neural net.





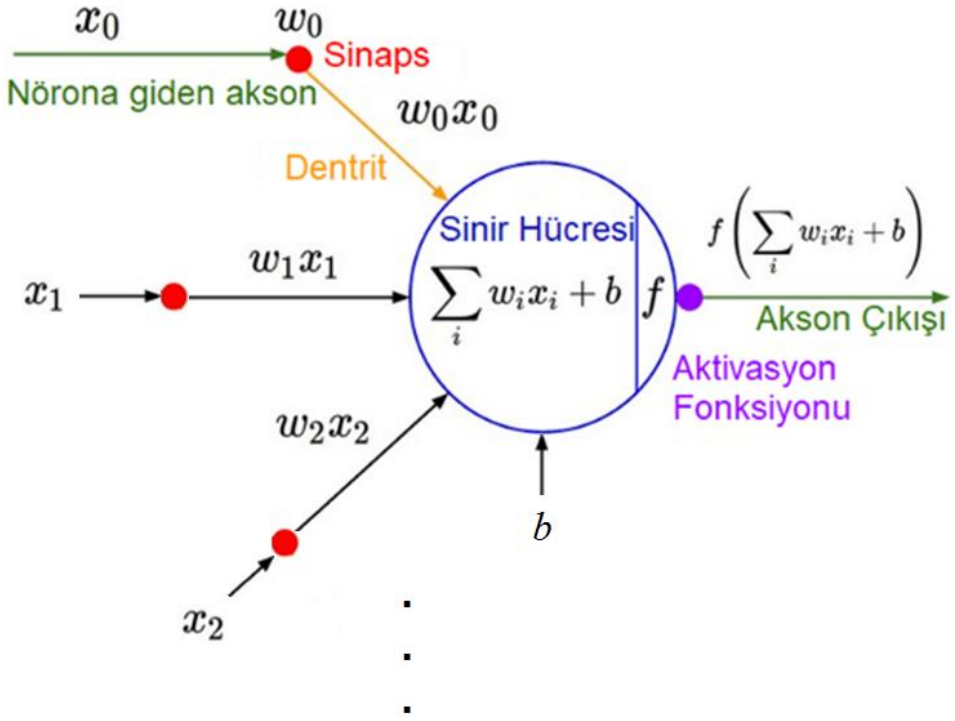
Tek gizli katmanlı ağ
(3 giriş -> 2 çıkış)

```
4+2       = 6 nöron
3x4+4x2   = 20 ağırlık
4+2       = 6  bias değeri
Toplam    = 26 adet
            öğrenilmesi gereken
            parametre var.
```

İki gizli katmanlı ağ
(3 giriş -> 1 çıkış)

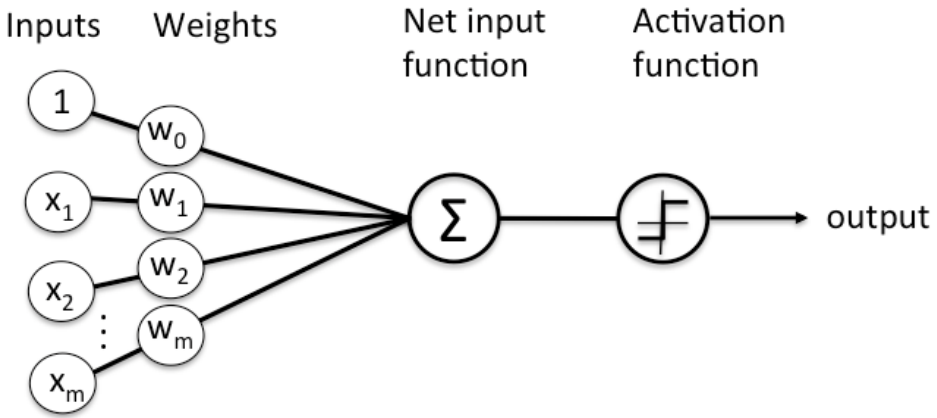
```
4+4+1         =  9 nöron,
3x4+4x4+4x1   = 32 ağırlık
4+4+1         =  9 bias değeri
Toplam        = 41 adet
                öğrenilmesi gereken
                parametre var.
```

Perceptron Aktivasyonu



Yapay sinir ağlarına doğrusal olmayan gerçek dünya özelliklerini tanıtmak için *aktivasyon fonksiyonuna* ihtiyaç duyarız.

Aktivaston



Nöron çıktısı:

$$y = w_0x_0 + w_1x_1 + \cdots + w_nx_n + b$$

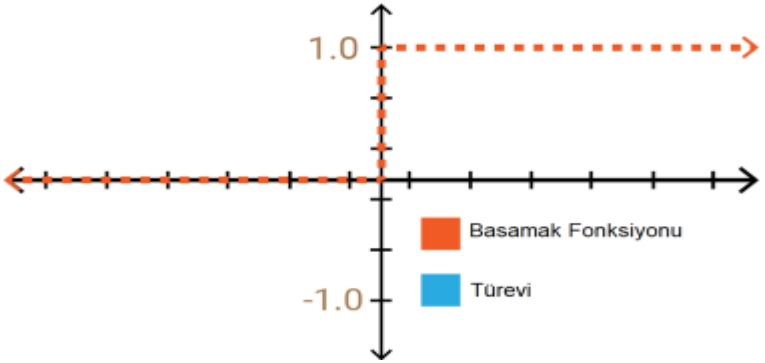
Aktivasyon fonksiyonu: $f(y)$

Ağdaki herbir nöron için her zaman:

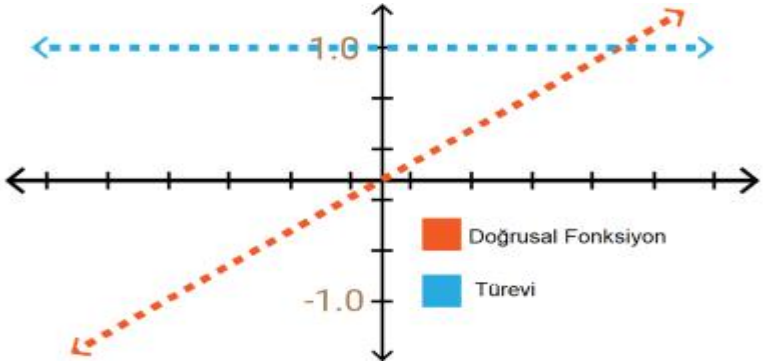
1. Girişlerle ağırlıkları çarp,
2. Bias ile topla
3. Aktivasyon uygula

Aktivaston Fonksiyonu Örnekleri

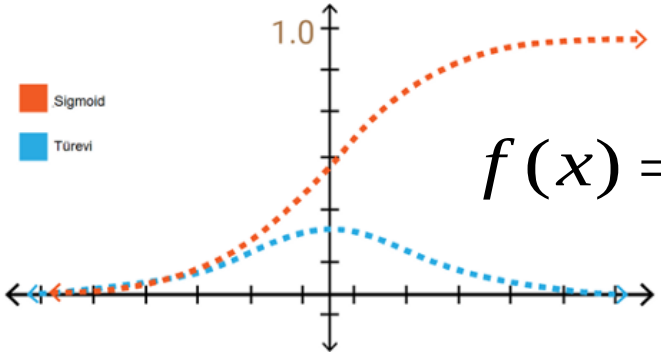
Basamak Fonksiyonu:



Doğrusal fonksiyonu:

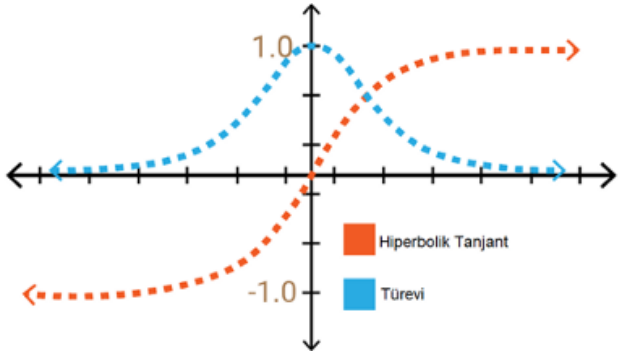


Sigmoid Fonksiyonu:



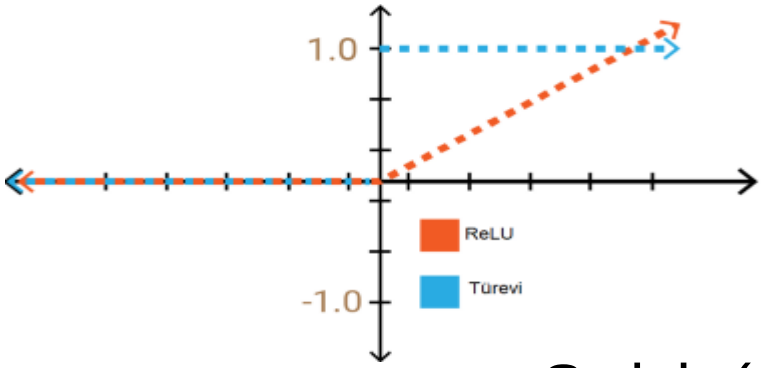
$$f(x) = \frac{1}{1+e^{-x}}$$

tanh(x) fonksiyonu:

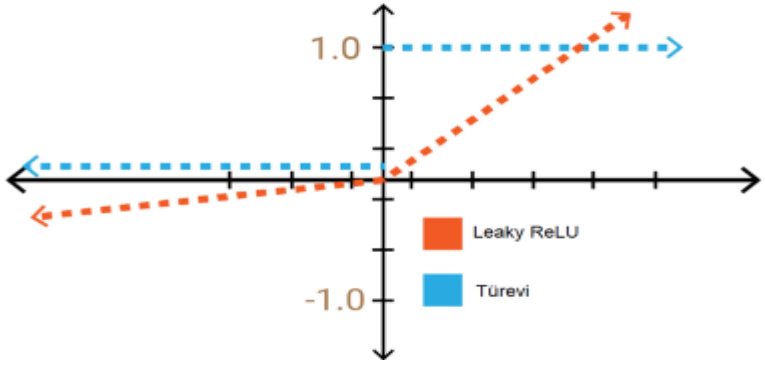


Aktivaston Fonksiyonu Örnekleri

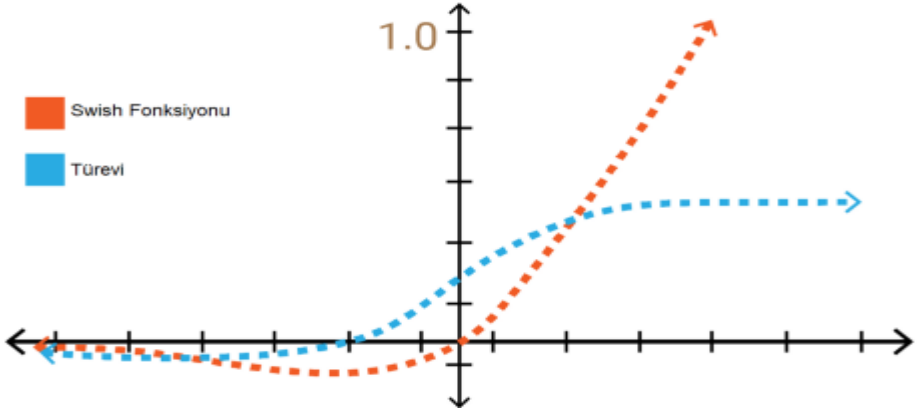
ReLU (Rectified Linear Unit)



Sızıntı (Leaky) ReLU



Swish (A Self-Gated):



Aktivaston Fonksiyonu Örnekleri

AKTİVASYON FONKSİYON	DENKLEM	ARALIK
Doğrusal Fonksiyon	$f(x) = x$	$(-\infty, \infty)$
Basamak Fonksiyonu	$f(x) = \begin{cases} 0 \text{ için } x < 0 \\ 1 \text{ için } x \geq 0 \end{cases}$	$\{0, 1\}$
Sigmoid Fonksiyon	$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Hiperbolik Tanjant Fonksiyonu	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0 \text{ için } x < 0 \\ x \text{ için } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky (Sızıntı) ReLU	$f(x) = \begin{cases} 0.01 \text{ için } x < 0 \\ x \text{ için } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Swish Fonksiyonu	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 \text{ için } f(x) = x \\ \beta \to \infty \text{ için } f(x) = 2\max(0, x) \end{cases}$	$(-\infty, \infty)$

Püf Noktalar

$$f(y) = f\left(\sum w_i x_i + b\right)$$

Çıkışa Aktarılacak bir Sinyali (y değerini) Neden Aktive Ediyoruz?

Eğer aktivasyon fonksiyonu uygulanmazsa çıkış sinyali basit bir doğrusal fonksiyon olur. Doğrusal fonksiyonlar yalnızca tek dereceli polinomlardır. Öğrenme düzeyi düşüktür. Biz sinir ağımızın doğrusal olmayan durumları da öğrenmesini istiyoruz.

Hangi Aktivasyon Fonksiyonu Tercih edilmelidir?

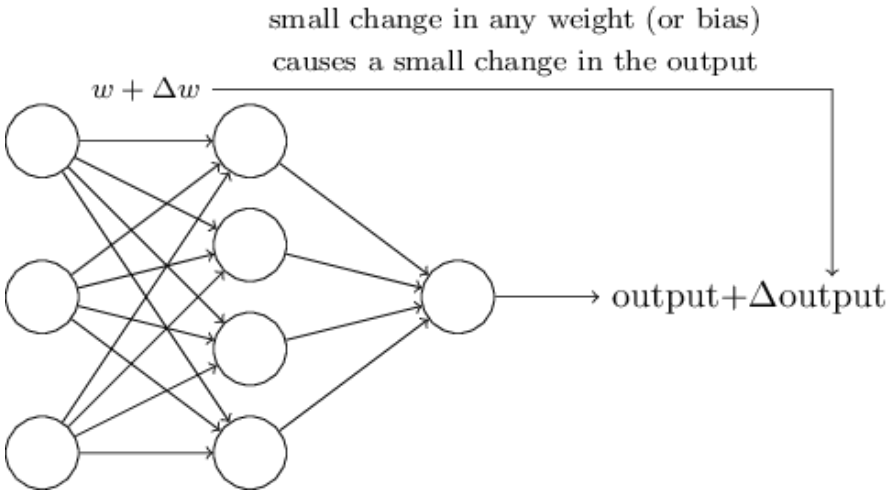
Her aktivasyon fonksiyonunun avantajı/dezavantajı vardır.

* Geniş aralıkta aktivasyon için → tanh(x)
* Modelim yavaş öğrensin → Sigmoid
* Derin ağlarda çalışmak için → ReLU

Çok katmanlı Ağlar için Geri Yayılım (Back Propagation)

1. Rastgele ağırlıkla başla
2. Son çıkış değerlerini hesapla
3. Çıkıştaki hatayı hesapla
   [Error = (target – calculated)^2]

$$E = (t - y)^2$$



4. Ağırlıkları güncelle

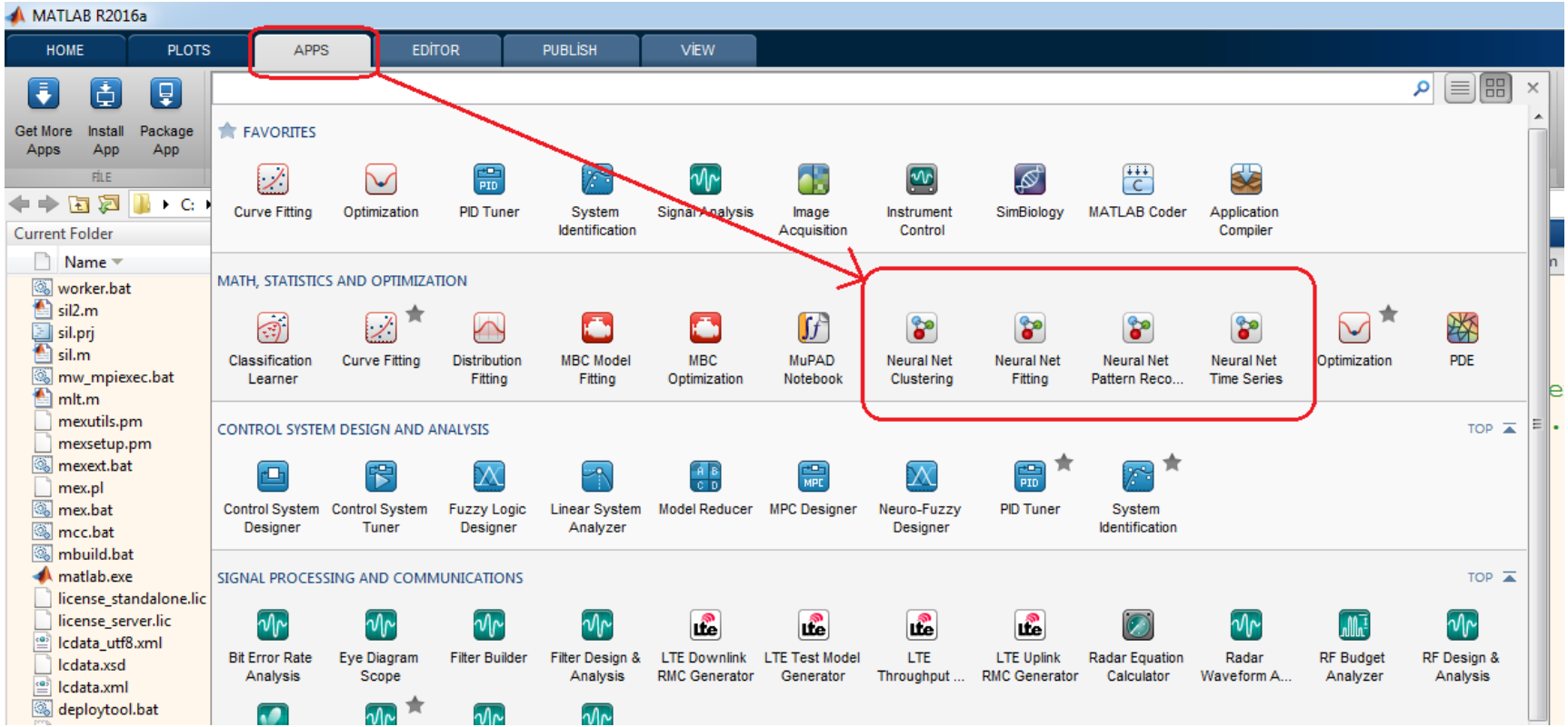
$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

eta = öğrenme hızı (learning rate, genellikle η = [0,1])

5. Hata en küçük oluncaya kadar 2., 3. ve 4. adımları tekrarla.

MATLAB ile Yapay Sinir Ağları

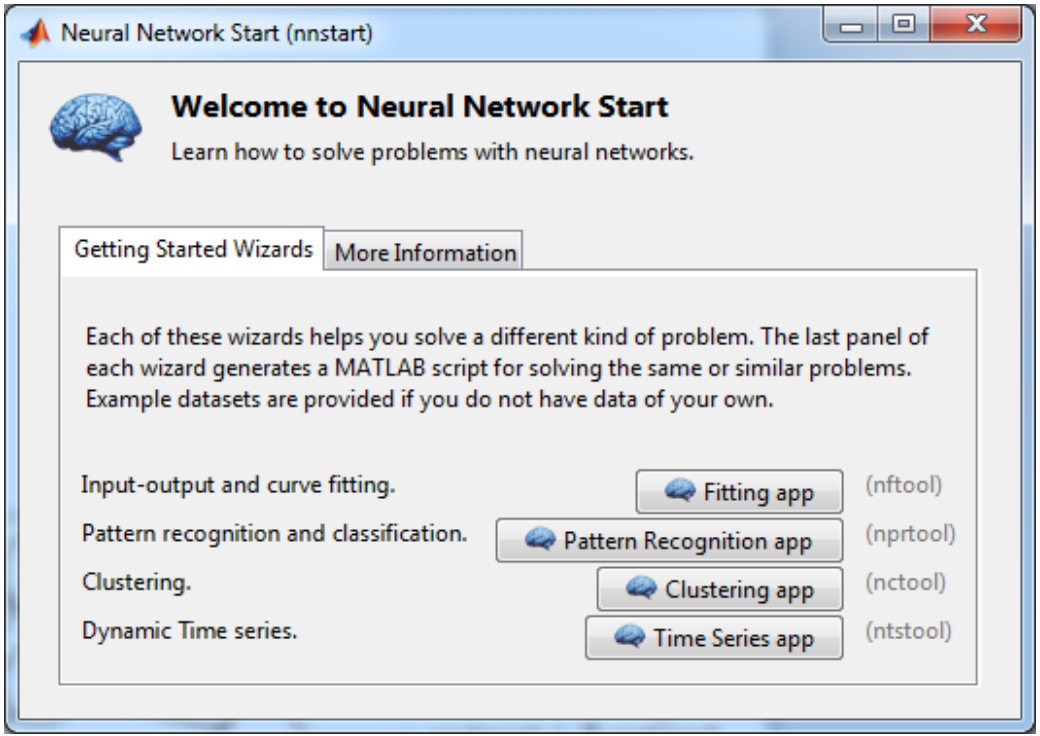
1. yol: Matlab APPS menüsünü kullan



MATLAB ile Yapay Sinir Ağları

2. yol: Matlab Komut penceresinden başlat

```
>> nnstart
```



MATLAB ile Yapay Sinir Ağları

3. yol: Kod yaz

```
% Tek gizli katmanlı ağ.
% YSA öğrenme
[x,t] = veri_kümesi;
% katmandaki nörön sayısı
gk1 = 10;
% ağı tanımla
net = fitnet(gk1);
% eğitim
net = train(net,x,t);
view(net)
% kullan
y = net(x);
plot(x,y)
```

```
% İki gizli katmanlı ağ.
% YSA derin öğrenme
[x,t] = veri_kümesi;
% katmandaki nörön sayıları
gk1 = 10;
gk2 = 7;
% ağı tanımla
net = fitnet([gk1 gk2]);
% eğitim
net = train(net,x,t);
% kullan
view(net)
y = net(x);
plot(x,y)
```

Örnek

$y = x^2$ fonksiyonunu öğrenme!

2. Yol

```
>> x = 0:0.5:20;
>> y = x.^2;
>> nnstart
>> Y = myNeuralNetworkFunction(x)
>> plot(x,y,'*b')
>> hold on
>> plot(x,Y,'-r')
```

3. Yol

```
clear; clc;
% veri_kümesi;
x = 0:0.5:20;
y = x.^2;
% katmandaki nörön sayısı
gk1 = 10;
% ağı tanımla
net = fitnet(gk1);
% eğitim
net = train(net,x,y);
view(net)
% kullan
y2 = net(x);
plot(x,y,'*b')
hold on
plot(x,y2,'-r')
```

Örnek

İki değişkenli XOR fonksiyonunu öğrenme

s_1	s_2	t (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

```
>> s = [0 0; 0 1; 1 0; 1 1];
>> y = [0 1 1 0]';
>> nnstart
```

Alıştırma

Üç değişkenli XOR fonksiyonunu öğrenme

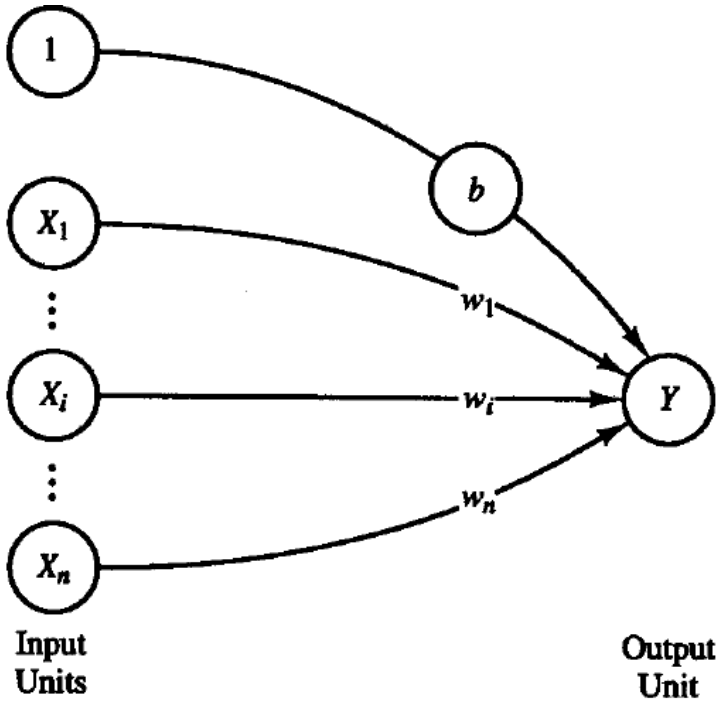
s_1	s_2	s_3	t (XOR)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

3. Kısım

Hebb Kuralı ile Desen Sınıflandırma (Pattern Classification)

Basit Mimari

Desen sınıflandırma için kullanılan en basit mimari yandaki gibidir. Bilinen en eski öğrenme yöntemi Hebb Kuralı'dır.



Hesaplamalar için aşağıdaki formüller kullanılır:

$$\text{net} = b + \sum_i x_i w_i.$$

$$Y = f(\text{net}) = \begin{cases} 1 & \text{if net} \geq 0; \\ -1 & \text{if net} < 0; \end{cases}$$

Hebb Net Algoritması [Girdiler s, Çıktı t]

Step 0. Initialize all weights:

$$w_i = 0 \qquad (i = 1 \text{ to } n).$$

Step 1. For each input training vector and target output pair, s : t, do steps 2–4.

Step 2. Set activations for input units:

$$x_i = s_i \qquad (i = 1 \text{ to } n).$$

Step 3. Set activation for output unit:

$$y = t$$

Step 4. Adjust the weights for

$$w_i(\text{new}) = w_i(\text{old}) + x_i y \qquad (i = 1 \text{ to } n).$$

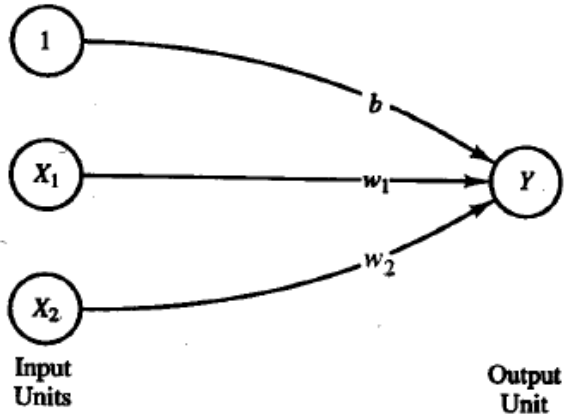
Adjust the bias:

$$b(\text{new}) = b(\text{old}) + y$$

Örnek:

2 değişkenli Bipolar AND Fonksiyonu

s_1	s_2	t (AND)
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

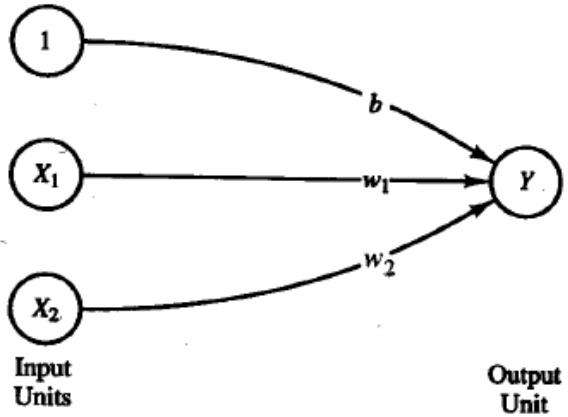



```
% Heb net for bipolar AND data
clear; clc;
% Data
s1 = [1  1 -1 -1];
s2 = [1 -1  1 -1];
t  = [1 -1 -1 -1];
% Agirliklari sifirla
w1 = 0; w2 = 0; b = 0;
% Egitim: agirliklari ayarla
for i = 1:4
  x1 = s1(i);
  x2 = s2(i);
  y  = t(i);
  w1 = w1 + x1*y;
  w2 = w2 + x2*y;
  b  = b  + y;
end
% Sinama: Test
A = 1; B = -1;          % Girdiler
net = w1*A + w2*B + b; % Cikti
if net >= 1
    fnet = 1
else
    fnet = -1
end
```

Örnek:

2 değişkenli Bipolar OR Fonksiyonu

s_1	s_2	t (AND)
1	1	1
1	-1	1
-1	1	1
-1	-1	-1




```
% Heb net for bipolar OR data
clear; clc;
% Data
s1 = [1  1 -1 -1];
s2 = [1 -1  1 -1];
t  = [1  1  1 -1];
% Agirliklari sifirla
w1 = 0; w2 = 0; b = 0;
% Egitim: agirliklari ayarla
for i = 1:4
  x1 = s1(i);
  x2 = s2(i);
  y  = t(i);
  w1 = w1 + x1*y;
  w2 = w2 + x2*y;
  b  = b  + y;
end
% Sinama: Test
A = 1; B = -1;          % Girdiler
net = w1*A + w2*B + b; % Cikti
if net >= 1
    fnet = 1
else
    fnet = -1
end
```

3. Kısım

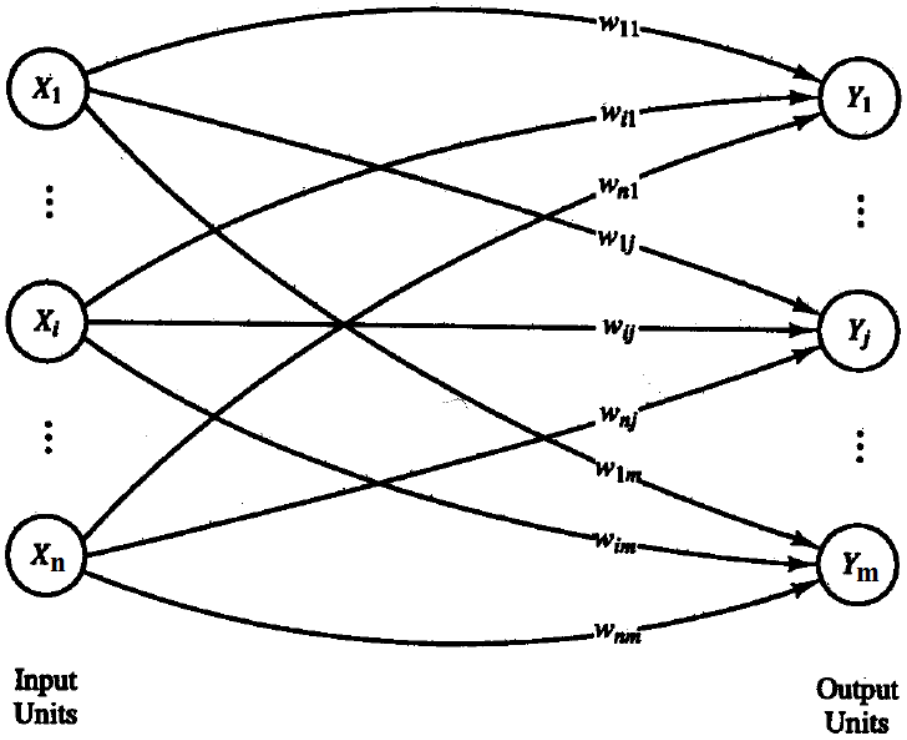
Hebb Kuralı ile Desen Eşleştirme (Pattern Association)

Basit Mimari

Hebb Kuralı Desen eşleştirme için de kullanılır.

Binary veya Bipolar veri kümeleri için uygundur.

Hesaplamalar için aşağıdaki formüller kullanılır:



Hesaplama

Girdi vektörü:

$$\mathbf{s} = (s_1, \ldots, s_i, \ldots, s_n)$$

Hedef vektörü:

$$\mathbf{t} = (t_1, \ldots, t_j, \ldots, t_m)$$

Ağırlıklar (eğitim):

$$w_{ij} = \sum_{p=1}^{P} s_i(p) t_j(p)$$

ÖRNEK:

s_1	s_2	s_3	s_4	t_1	t_2
1	-1	-1	-1	1	-1
1	1	-1	-1	1	-1
-1	-1	-1	1	-1	1
-1	-1	1	1	-1	1

Algoritma [Girdiler s, Çıktılar t]

Step 0. Initialize all weights ($i = 1, \ldots, n; j = 1, \ldots, m$):

$$w_{ij} = 0.$$

Step 1. For each input training–target output vector pair **s**:**t**, do Steps 2–4.

Step 2. Set activations for input units to current training input ($i = 1, \ldots, n$):

$$x_i = s_i$$

Step 3. Set activations for output units to current target output ($j = 1, \ldots, m$):

$$y_j = t_j.$$

Step 4. Adjust the weights ($i = 1, \ldots, n; j = 1, \ldots, m$):

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j.$$

Örnek:

Verilen data

Girdiler: [-1 veya 1]
Çıktılar: [-1 veya 1]

s_1	s_2	s_3	s_4	t_1	t_2
1	-1	-1	-1	1	-1
1	1	-1	-1	1	-1
-1	-1	-1	1	-1	1
-1	-1	1	1	-1	1

```
% Pattern association for bipolar data
clear; clc;

% Data
s = [1 -1 -1 -1; ...
     1  1 -1 -1; ...
    -1 -1 -1  1; ...
    -1 -1  1  1];
t = [1 -1; 1 -1; -1 1; -1 1];

% train
W = s'*t;

% test
x = [1 1 -1 -1];
y = x*W;

a = find(y>1); y(a) =  1;
b = find(y<1); y(b) = -1;
y
```

Örnek:

Verilen data

Girdiler: [-1,1] arası gerçel sayılar

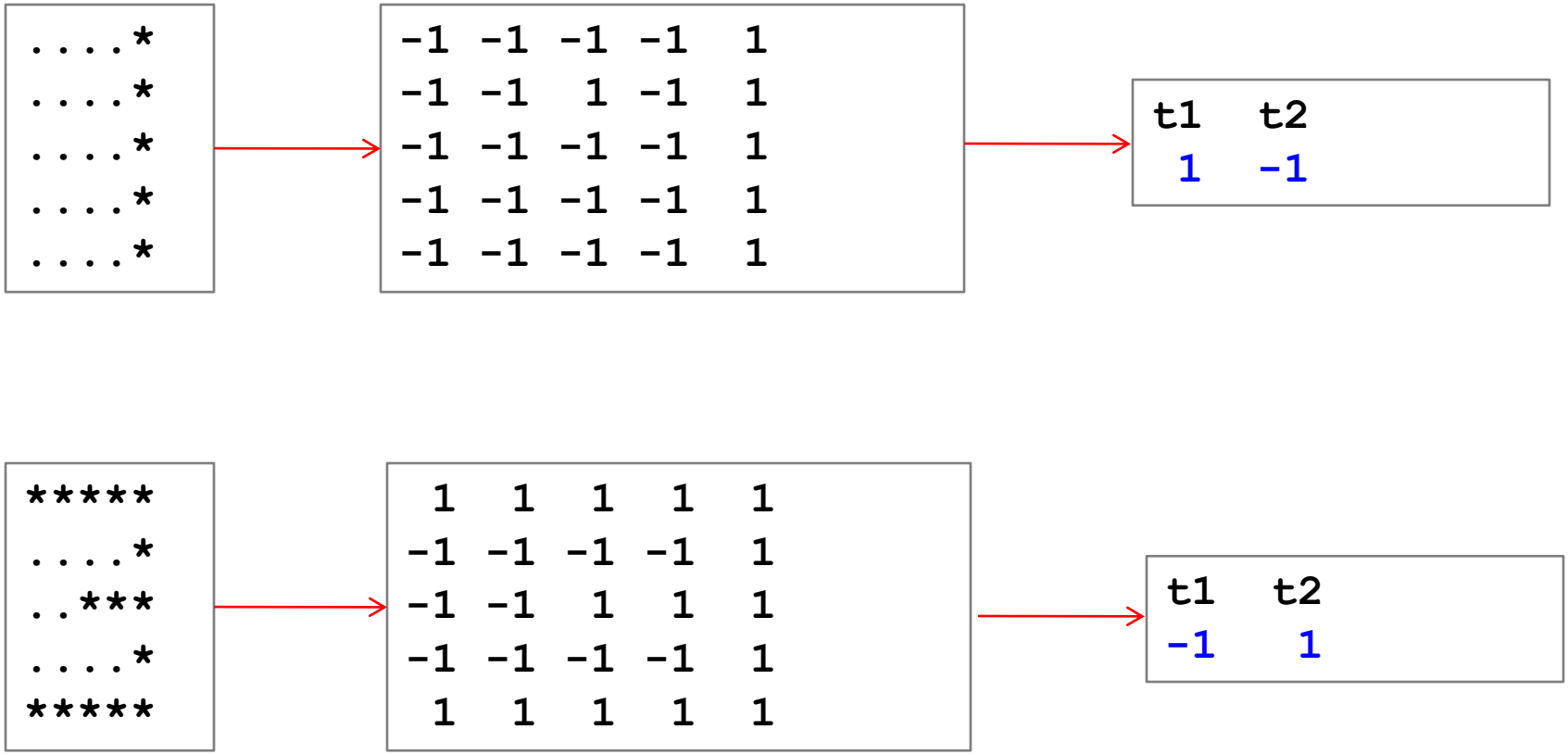
Çıktılar: [-1 veya 1]

s_1	s_2	s_3	s_4	s_5	t_1	t_2	t_3	t_4
1.0	-0.5	0.5	-1.0	-0.4	1	-1	-1	-1
1.0	-0.2	-1.0	-0.5	0.4	-1	1	-1	-1
-0.1	-0.3	0.9	1.0	0.3	-1	-1	1	-1
0.2	0.5	0.6	0.9	-1.0	-1	-1	-1	1
0.3	0.4	0.7	1.0	-0.9	-1	-1	-1	1

```
% Pattern association for gray data
clear; clc;
% Data
s = [1.0 -0.5  0.5 -1.0 -0.4; ...
     1.0 -0.2 -1.0 -0.5  0.4; ...
    -0.1 -0.3  0.9  1.0  0.3; ...
     0.2  0.5  0.6  0.9 -1.0; ...
     0.3  0.4  0.7  1.0 -0.9];
t = [ 1 -1 -1 -1; ...
     -1  1 -1 -1; ...
     -1 -1  1 -1; ...
     -1 -1 -1  1; ...
     -1 -1 -1  1];
% train
W = s'*t;

% test
x = [0.3  0.4  0.7  1.0 -0.9];
y = x*W;
y
a = find(y>1); y(a) =  1;
b = find(y<1); y(b) = -1;
y
```

Örnek: Basit karakter tanıma 1

İki karakter için 5x5 matris oluştur:



Örnek: Basit karakter tanıma 2

Matris elemanlarını yanyana yazıp, bir boyutlu dizi oluştur

Girdiler																									Çıktılar	
s1	s2			.	.	.																		s25	t1	t2
--	--	--																						---	---	---
-1	-1	-1	-1	1	-1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	-1	-1	1	1	-1
1	1	1	1	1	-1	-1	-1	-1	1	-1	-1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	-1	1

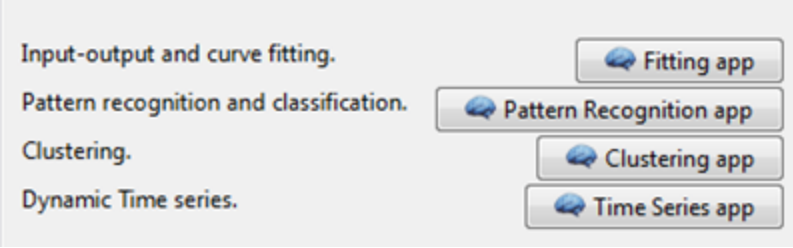
Örnek: Basit karakter tanıma 3

Hebb kuralını uygula

```
% Pattern association for bipolar data
clear; clc;

% Data
s = [-1 -1 -1 -1  1  ...
     -1 -1  1 -1  1  ...
     -1 -1 -1 -1  1  ...
     -1 -1 -1 -1  1  ...
     -1 -1 -1 -1  1;

      1  1  1  1  1  ...
     -1 -1 -1 -1  1  ...
     -1 -1  1  1  1  ...
     -1 -1 -1 -1  1  ...
      1  1  1  1  1];
t = [1 -1; -1  1];

% train
W = s'*t;

% test
x = [-1 -1 -1 -1  1 -1 -1  1 -1  1  ...
     -1 -1 -1 -1  1 -1 -1 -1 -1  1  -1 -1 -1 -1  1];
% hesapla
y = x*W;
a = find(y>1); y(a) =  1;
b = find(y<1); y(b) = -1;
y
```

4. Kısım

Uygulamalar



Atış Problemi: Fitting

Bir silahtan çıkan 9 mm çapındaki merminin ilk hızı ve yatayla yaptığı açı bilgisinden, hava direnci ve rüzgar etkileri dikkate alınarak, merminin menzili hesaplanmıştır.

Veri dosyaları:
http://www1.gantep.edu.tr/~bingul/ai/data/data_projectile.txt
http://www1.gantep.edu.tr/~bingul/ai/data/data_projectile.xlsx

Nişastadan Şurup Eldesi: Fitting

Bir kabın içine bir miktar nişata ve iki tip enzim konmuştur. Enzimler ile tepkimeye giren nişasta belli bir süre sonra

Dextrose, Maltose, D3 ve DP

gibi şurupların ortaya çıkmasına sebep olur.

Veri dosyaları:
http://www1.gantep.edu.tr/~bingul/ai/data/data_nisasta.txt
http://www1.gantep.edu.tr/~bingul/ai/data/data_nisasta.xlsx

Iris Flower: Pattern Recognition

Süsen Çiçeği sınıflandırması
(Çanak yaprak uzunluğu ve genişliği, Taç yaprak uzunluğu ve genişliği)

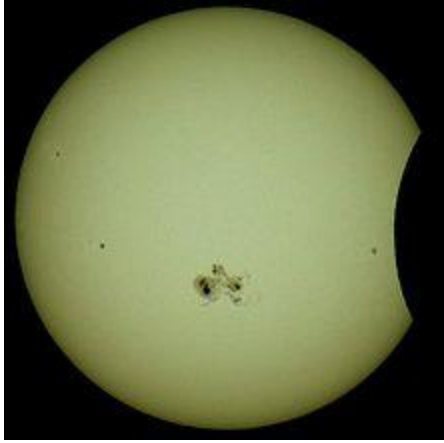
Veri dosyaları:
http://www1.gantep.edu.tr/~bingul/ai/data/data_iris.txt
http://www1.gantep.edu.tr/~bingul/ai/data/data_iris.xlsx

Tohumların Sınıflandırılması: Pattern Rec.

Tohumların geometrisine göre sınıflandırılması.
(alan, çevre, yoğunluk, en, boy, asimetri, çentik uzunuluğu)

Veri dosyaları:
http://www1.gantep.edu.tr/~bingul/ai/data/data_seeds.txt
http://www1.gantep.edu.tr/~bingul/ai/data/data_seeds.xlsx

Güneş Lekeleri: Time Series



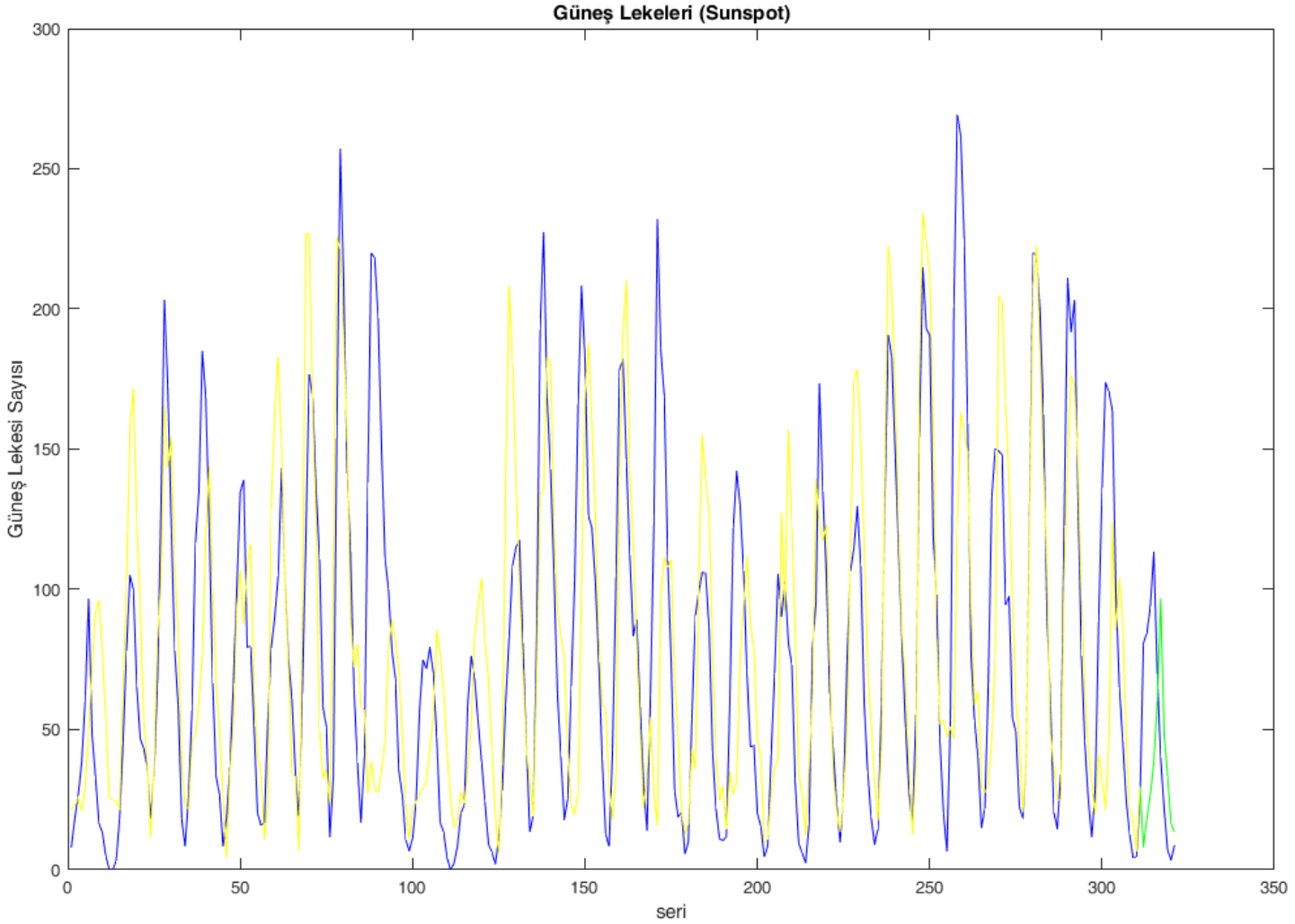
Güneş lekeleri güneş yüzeyinde gözlenen Koyu renkli bölgelerdir.
Manyetik alanın belli bölgelerde yoğunlaşması, ısının eşit bir şekilde yayılımını engeller. Sonuç olarak çevresindeki ışık küreye göre daha düşük yüzey sıcaklığına sahip Güneş Lekeleri dediğimiz bölgeler oluşur. Bunlar genellikle çiftler halinde görünür. Her ikisi de birbirlerinin zıt manyetik kutuplarıdır. Bu lekeler 11 yılda bir sayılarının arttığı gözlenmiştir.

Veri dosyaları:
http://www1.gantep.edu.tr/~bingul/ai/data/data_sunspot.txt
http://www1.gantep.edu.tr/~bingul/ai/data/data_sunspot.xlsx

Güneş Lekeleri: Time Series

```
% zaman serisi analizi, Yıllık güneş lekesi verileri
clear; clc;
T = load('data.txt');
d = 10; % gecikme
T2= num2cell(T(1:end)'); % diziden -> hücreye
% ağı hazırla
net = narnet(1:d, [10 5]);
[Xs,Xi,Ai,Ts] = preparets(net,{},{},T2);
% eğitim
net = train(net,Xs,Ts,Xi,Ai);
% eğitilmiş ağın sonuçları
Y = net(Xs,Xi,Ai);
% ağı göster
view(net)
Y = cell2mat(Y);
Y2= [Y cell2mat(Xi)]
plot(T, 'b-')
hold on
plot(Y2,'g-')
plot(Y,'y-')
```

Güneş Lekeleri: Time Series



Diğer Uygulamalar

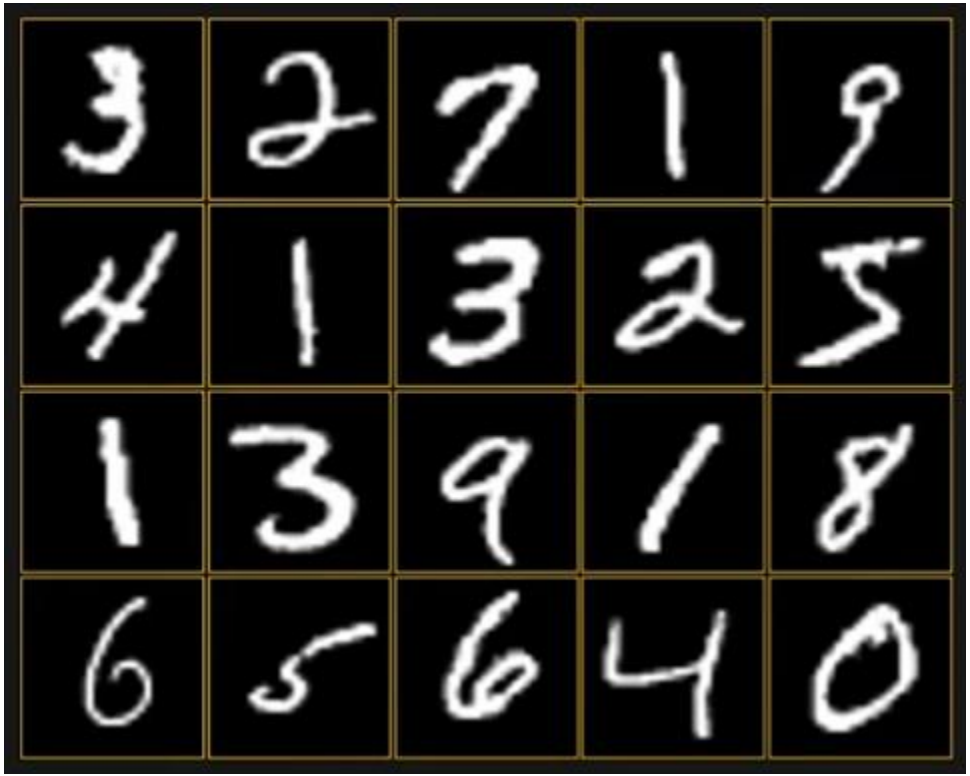
Ses Tanıma

Ses dosyaları web sayfasından indir:
http://www1.gantep.edu.tr/~bingul/ai/ses

Karakter Tanıma

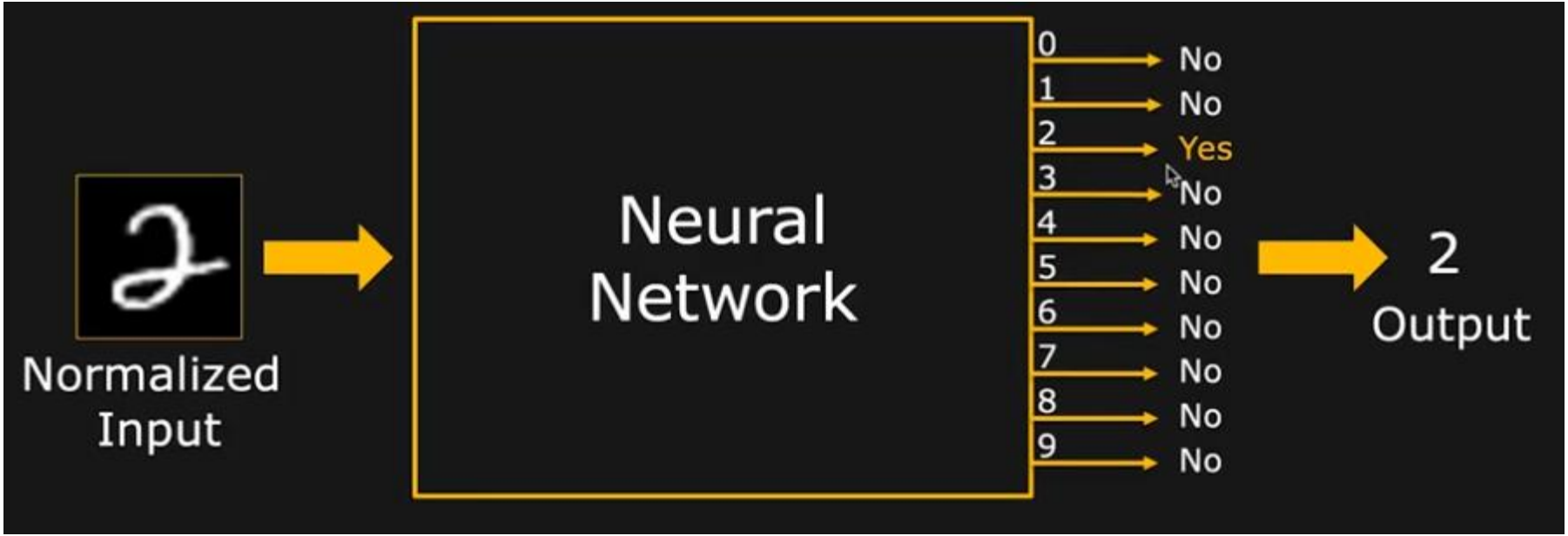
El yazısı tanıma eğitimi için kullanılan MNIST Dataset
http://yann.lecun.com/exdb/mnist/

Her bir görüntü
28x28
boyutlarında saklanmış.

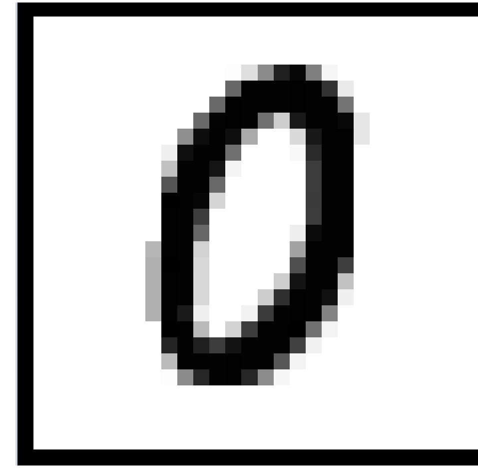


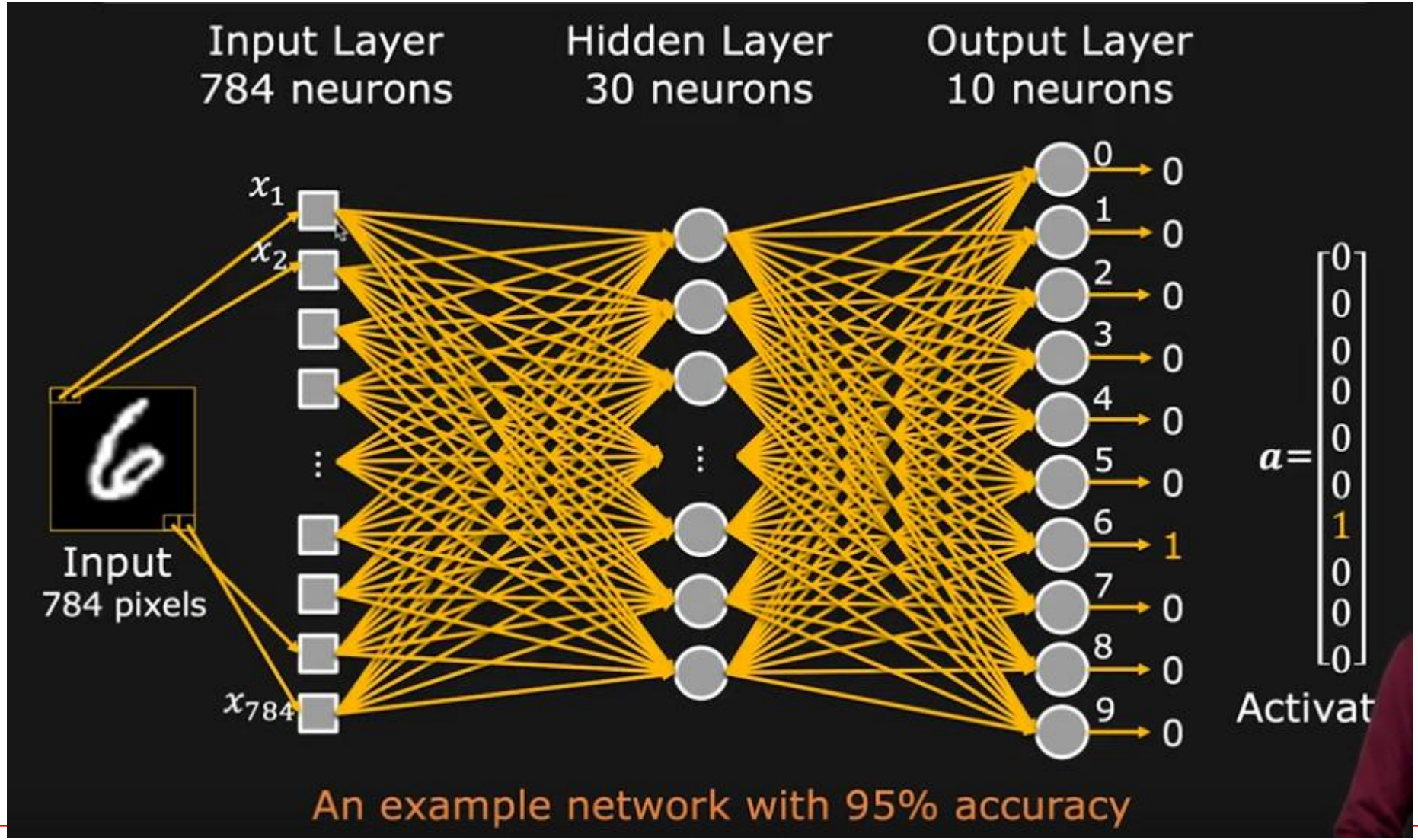
784 giriş ve 10 çıkış olacak şekilde ağ tanımlanır.

Örneğin “2” görüntüsü sisteme verilince sadece “2” nolu çıkış sinyal vermeli.



255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	182	176	176	176	176	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	239	193	85	0	1	1	1	0	1	1	1	0	36	187	253	255	255	255	255	255
255	255	255	255	255	255	255	143	15	2	2	1	2	2	2	1	2	2	26	1	2	2	139	255	255	255	255	255
255	255	255	255	255	255	105	13	2	2	2	6	60	103	210	216	216	216	240	186	32	2	38	255	255	255	255	255
255	255	255	255	255	105	3	1	2	20	104	213	255	255	255	255	255	255	255	255	60	2	2	255	255	255	255	255
255	255	255	253	103	1	1	14	111	243	255	255	255	255	255	255	255	255	255	211	19	1	1	255	255	255	255	255
255	255	255	223	2	2	2	170	255	255	255	255	255	255	255	255	255	255	242	60	2	2	38	255	255	255	255	255
255	255	255	139	2	2	111	255	255	255	255	255	255	255	255	255	255	205	59	1	2	2	140	255	255	255	255	255
255	255	255	31	2	2	206	255	255	255	255	255	255	255	255	255	206	47	2	1	2	87	247	255	255	255	255	255
255	255	255	7	1	1	25	218	248	255	255	255	255	238	163	79	19	1	1	0	65	243	255	255	255	255	255	255
255	255	255	127	2	2	2	24	42	60	60	59	60	11	2	1	2	2	2	113	241	255	255	255	255	255	255	255
255	255	255	246	51	2	2	1	2	2	2	1	2	2	2	1	2	16	131	243	255	255	255	255	255	255	255	255
255	255	255	255	234	116	8	1	2	2	2	1	2	2	2	62	194	240	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	230	230	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255




Input Layer
784 neurons
Hidden Layer
30 neurons
Output Layer
10 neurons
x_1
x_2
x_{784}
Input
784 pixels
0
1
2
3
4
5
6
7
8
9
0
0
0
0
0
1
0
0
$a=$
Activat
An example network with 95% accuracy

Sample Training Data - MNIST Dataset (60,000 images)

Training Image x										
Label	5	0	4	1	9	2	1	3	1	4
Desired Activation $\hat{a}(x)$	$\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$

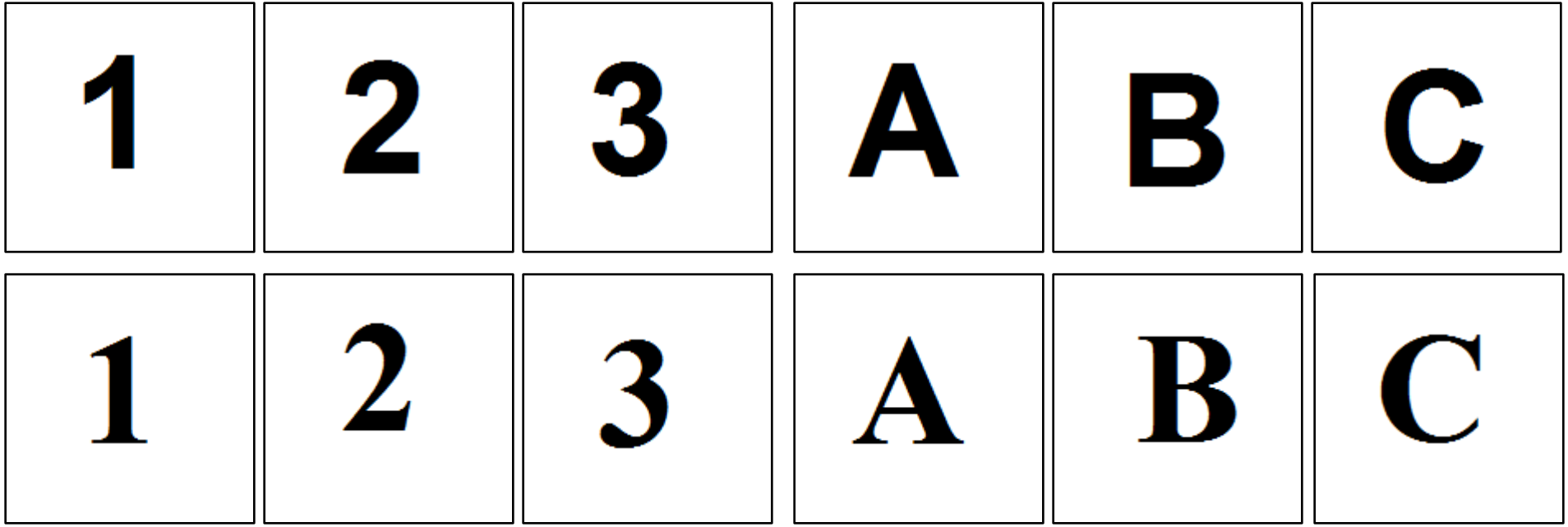
Yedek Yansılar

Hebb Rule ile Görüntü dosyasından karakter tanıma

Buradaki çalışma dosyaları web sayfasından indirilebilir.

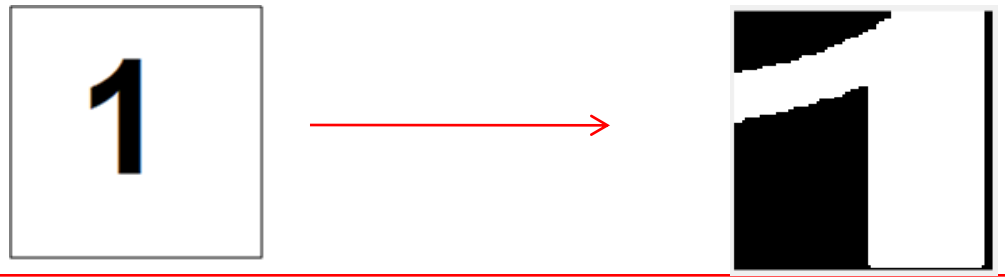
Girdiler: Aşağıda verilen siyah-beyaz görüntü dosyaları

Çıktılar: [1,2,3, A,B,C] semboleri



Görüntü dosyasından karakter tanıma 1

```
i1 = imread('1.png');       % görüntüyü al
i2 = im2bw(i1);             % siyah-beyaz yap
i3 = imcomplement(i2);      % negatifini al
bolge = regionprops(i3); % bölge belirle

i4 = imcrop(i3,bolge(1).BoundingBox); % kırp
i5 = imresize(i4,[100 100]);          % yeniden boyutlandır
imshow(i5);                           % göster

i5 = double(i5);                % gerçel sayı matrisine çevir
i5 = changem(i5,-1,0);          % görüntü içindeki 0 -> -1 yap
dizi= reshape(i5',1,[]);        % i1 matrisi dizi'ye çevir
```



Görüntü dosyasından karakter tanıma 2

```
% eğitim için bir önceki sayfadaki işlemleri tekrarla
% *** Girişler ***
a1 = sifir_1('t/a1.png'); % arial font
a2 = sifir_1('t/a2.png');
a3 = sifir_1('t/a3.png');
aA = sifir_1('t/aA.png');
aB = sifir_1('t/aB.png');
aC = sifir_1('t/aC.png');
t1 = sifir_1('t/t1.png'); % times new roman
t2 = sifir_1('t/t2.png');
t3 = sifir_1('t/t3.png');
tA = sifir_1('t/tA.png');
tB = sifir_1('t/tB.png');
tC = sifir_1('t/tC.png');
% girdi vektörünü oluştur
s = [a1; a2; a3; aA; aB; aC; ...
     t1; t2; t3; tA; tB; tC]';
```

Görüntü dosyasından karakter tanıma 3

```
% *** Çıkışlar ***
t = [+1 -1 -1 -1 -1 -1; ...   % 1
     -1 +1 -1 -1 -1 -1; ...   % 2
     -1 -1 +1 -1 -1 -1; ...   % 3
     -1 -1 -1 +1 -1 -1; ...   % A
     -1 -1 -1 -1 +1 -1; ...   % B
     -1 -1 -1 -1 -1 +1; ...   % C

     +1 -1 -1 -1 -1 -1; ...   % 1
     -1 +1 -1 -1 -1 -1; ...   % 2
     -1 -1 +1 -1 -1 -1; ...   % 3
     -1 -1 -1 +1 -1 -1; ...   % A
     -1 -1 -1 -1 +1 -1; ...   % B
     -1 -1 -1 -1 -1 +1];      % C
```

Görüntü dosyasından karakter tanıma 4

```
% *** Ağırlıkları hesapla ***
W = s * t;

%-----------------------------------------
% test dosyasını oku (el yazısı ile yazılmış)
x = sifir_1('t/e1.png');

% çıkışı hesapla (karakteri belirle)
y = x*W
```