Yapay Sinir Ağları

Abdulkadir Şengür

Giriş

- Canlıların davranışlarını inceleyip, matematiksel olarak modelleyip, benzer yapay modellerin üretilmesine ***sibernetik*** denir.

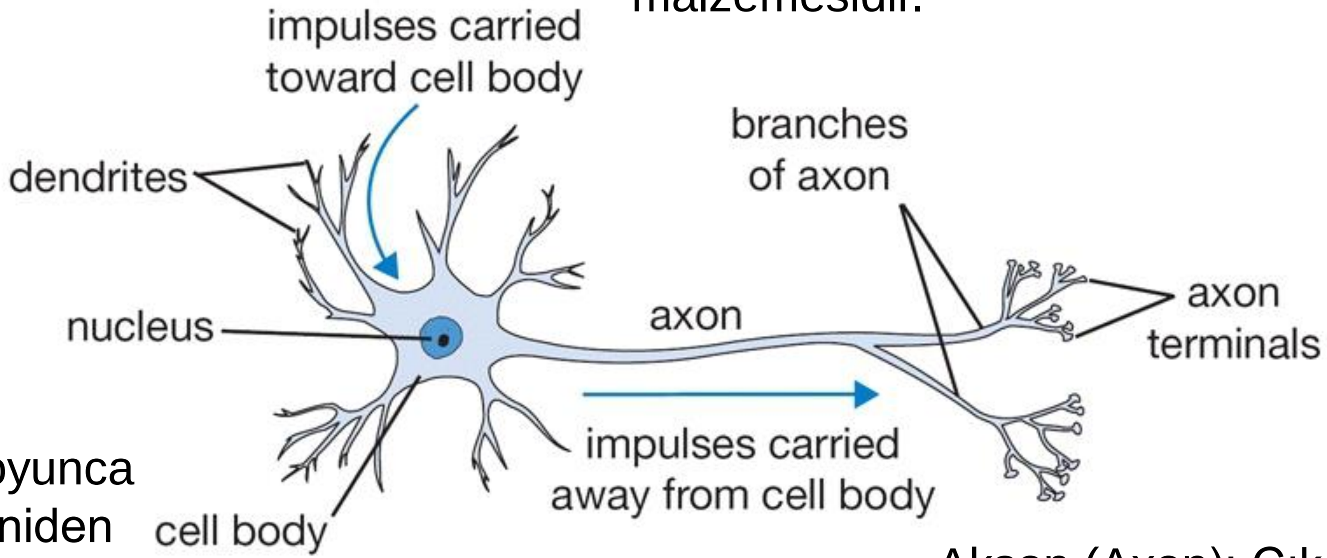
- Eğitilebilir, adaptif ve kendi kendine organize olup öğrenebilen ve değerlendirme yapabilen yapay sinir ağları ile insan beyninin öğrenme yapısı modellenmeye çalışılmaktadır.

- Aynı insanda olduğu gibi yapay sinir ağları vasıtasıyla makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır.

İnsandaki bir sinir hücresinin (nöron) yapısı

Dentritler (Dendrites): Diğer hücrelerden gelen işaretleri toplayan elektriksel anlamda pasif kollardır. Sistem girişidir.

Miyelin Tabaka (Myelin Sheath): Yayılma hızına etki eden yalıtım malzemesidir.

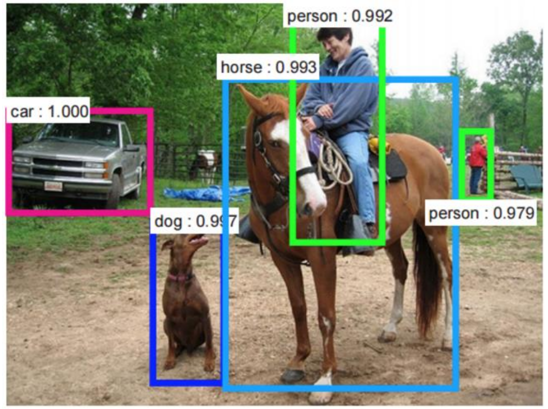


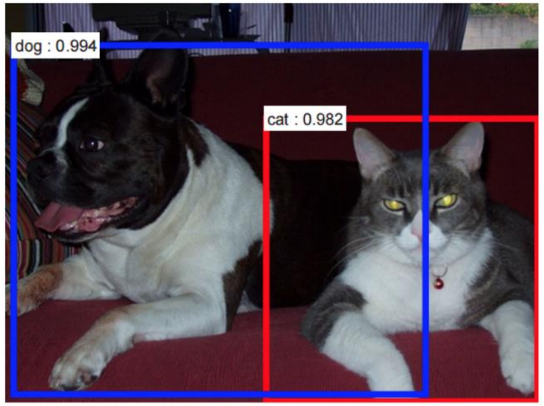
Çekirdek (Nucleus): Akson boyunca işaretlerin periyodik olarak yeniden üretilmesini sağlar.

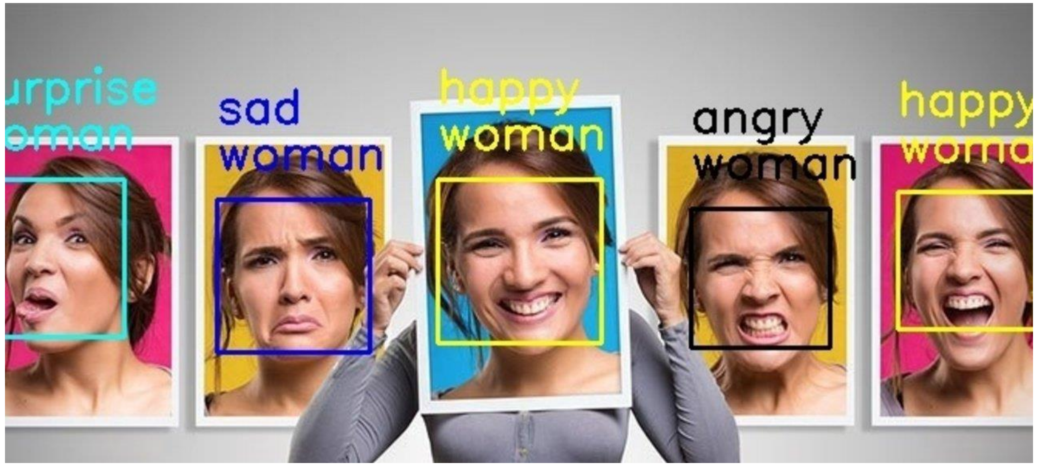
Akson (Axon): Çıkış darbelerinin üretildiği elektriksel aktif gövdedir ve gövde üzerinde iletim tek yönlüdür. Sistem çıkışıdır.

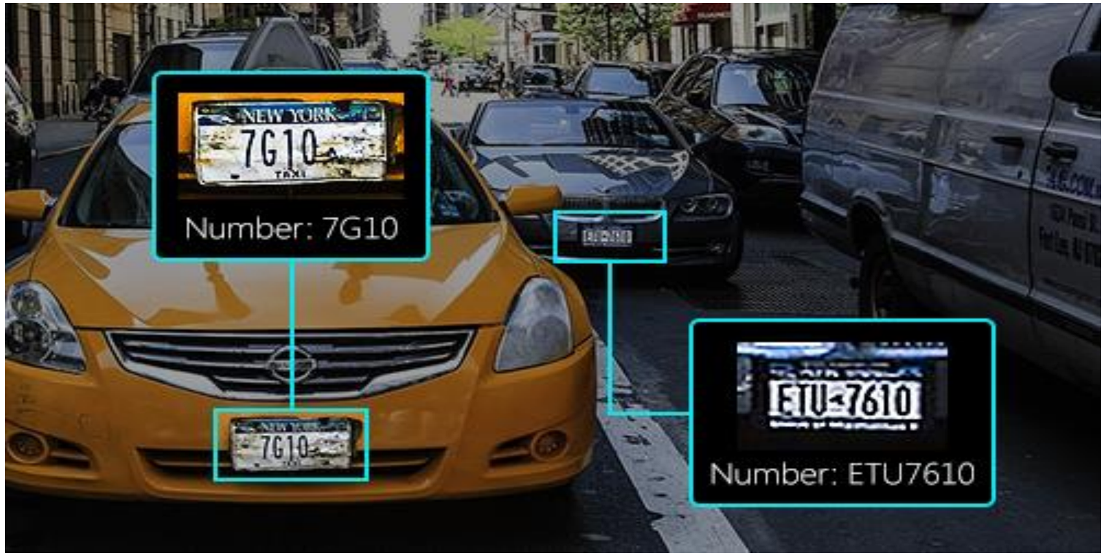
Sinaps (Synapse): Hücrelerin aksonlarının diğer dentritlerle olan bağlantısını sağlar.

Bazı uygulama örnekleri

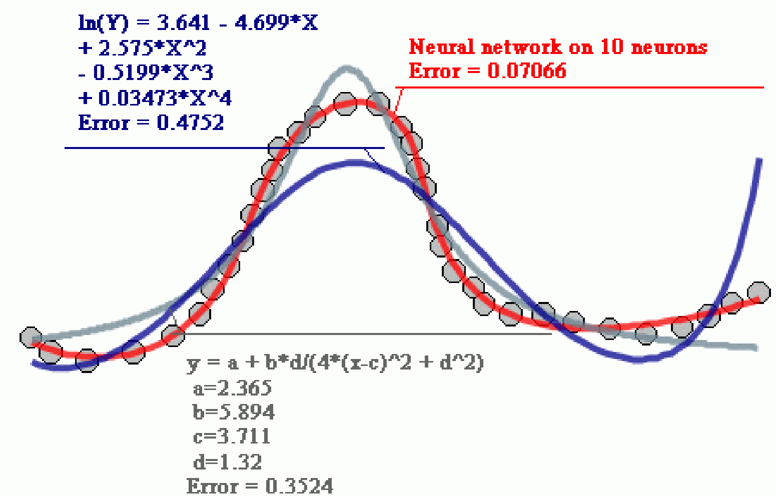


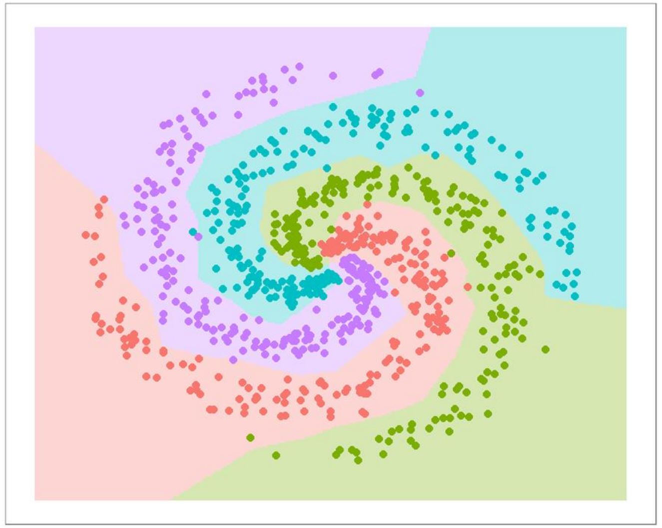






Bazı uygulama örnekleri





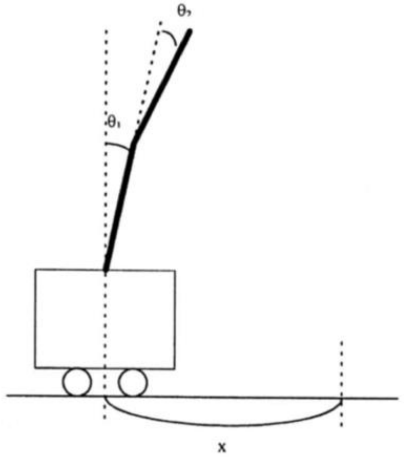
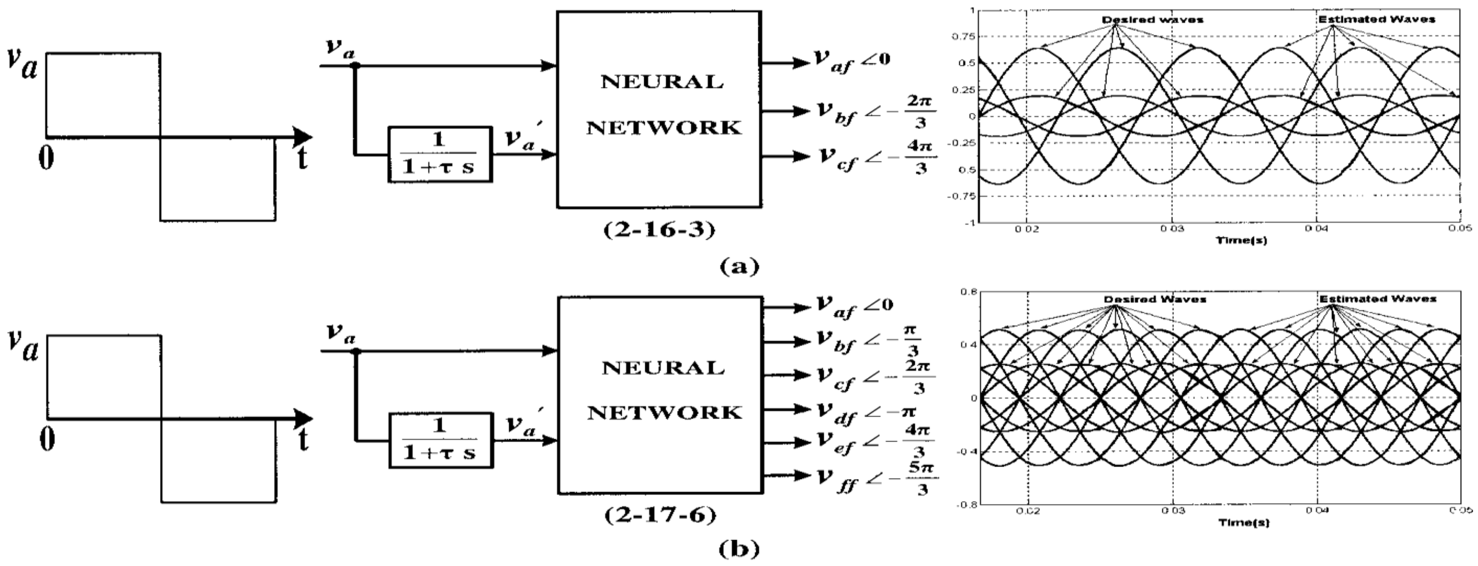
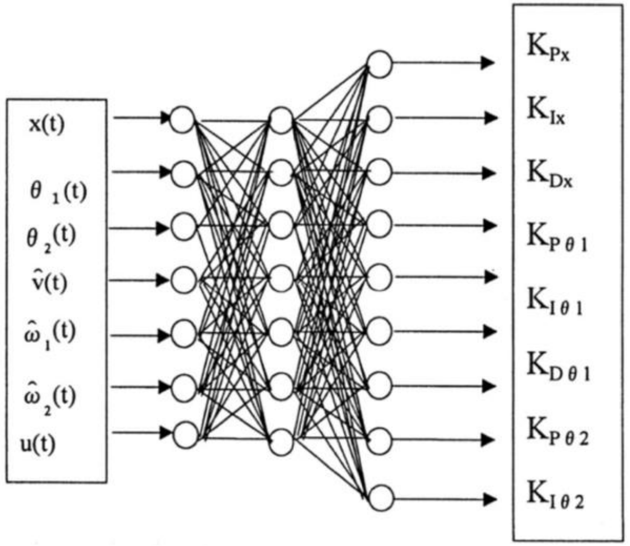
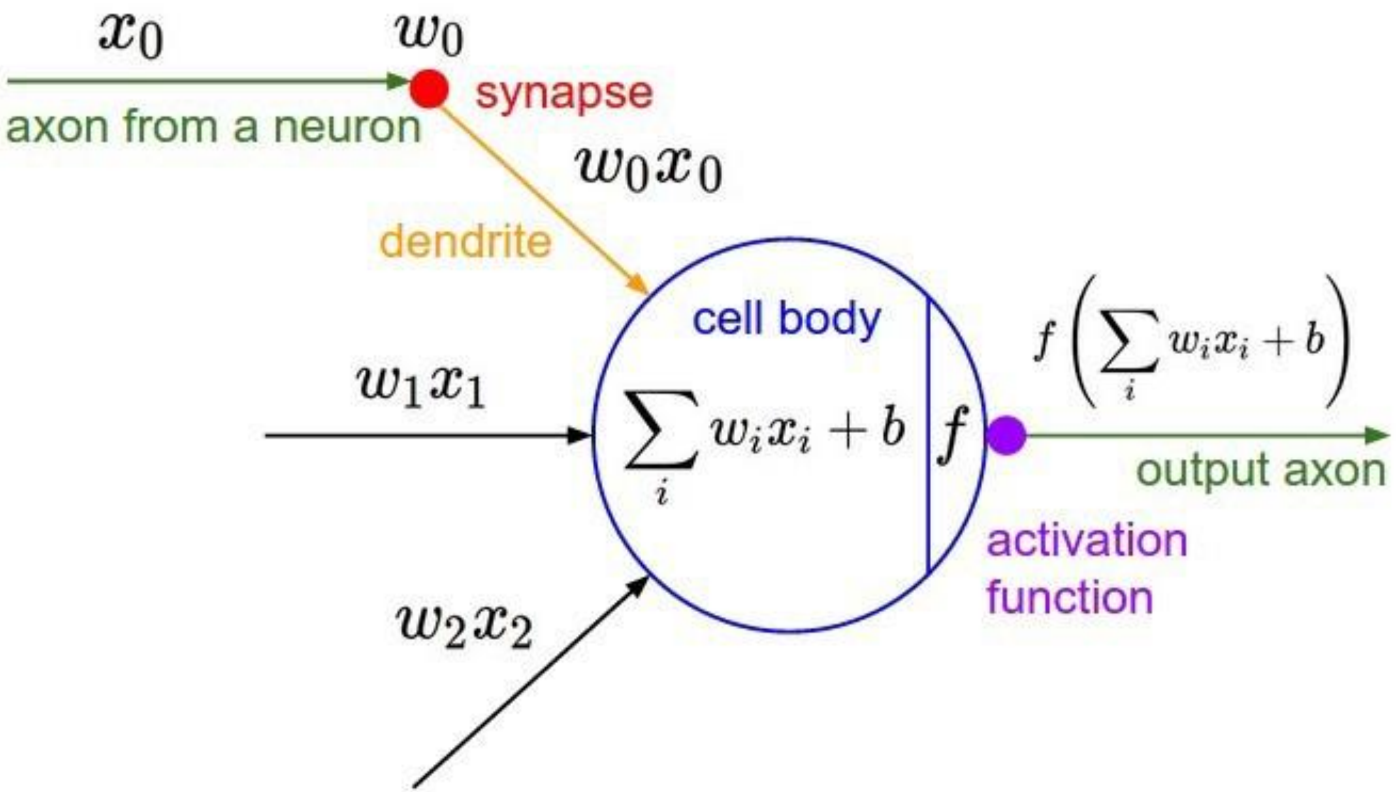


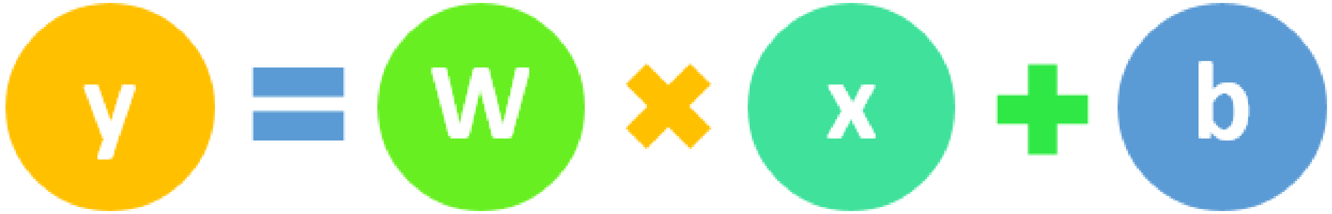
Fig. 11. Control system of double inverted pendulum





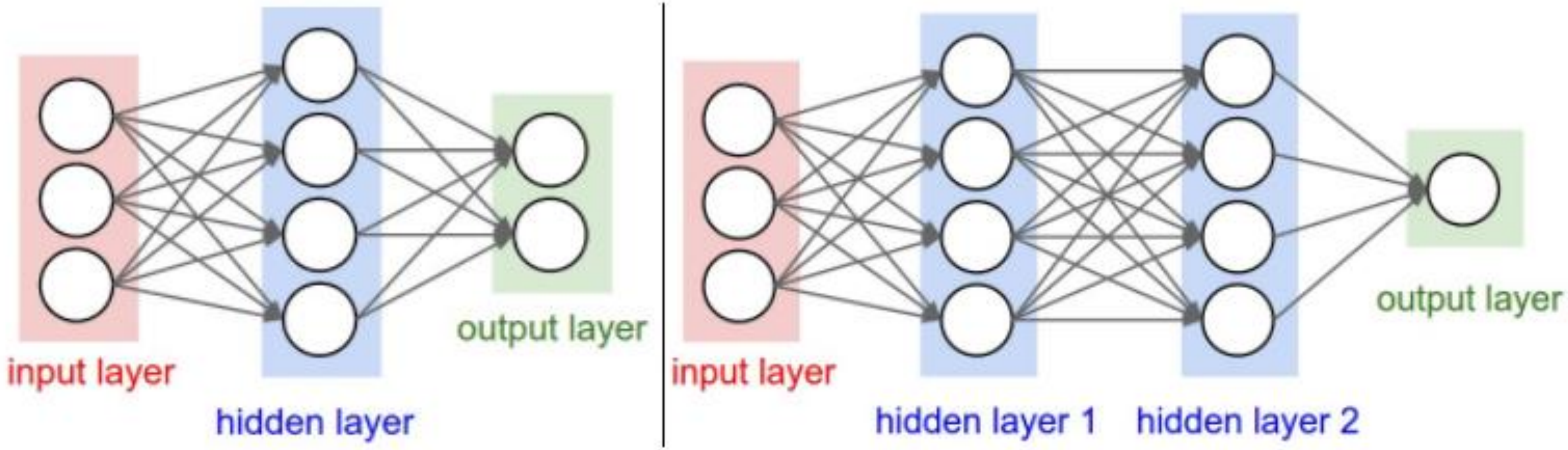
İnsandaki bir sinir hücresinin matematiksel modeli





Çok katmanlı bir YSA

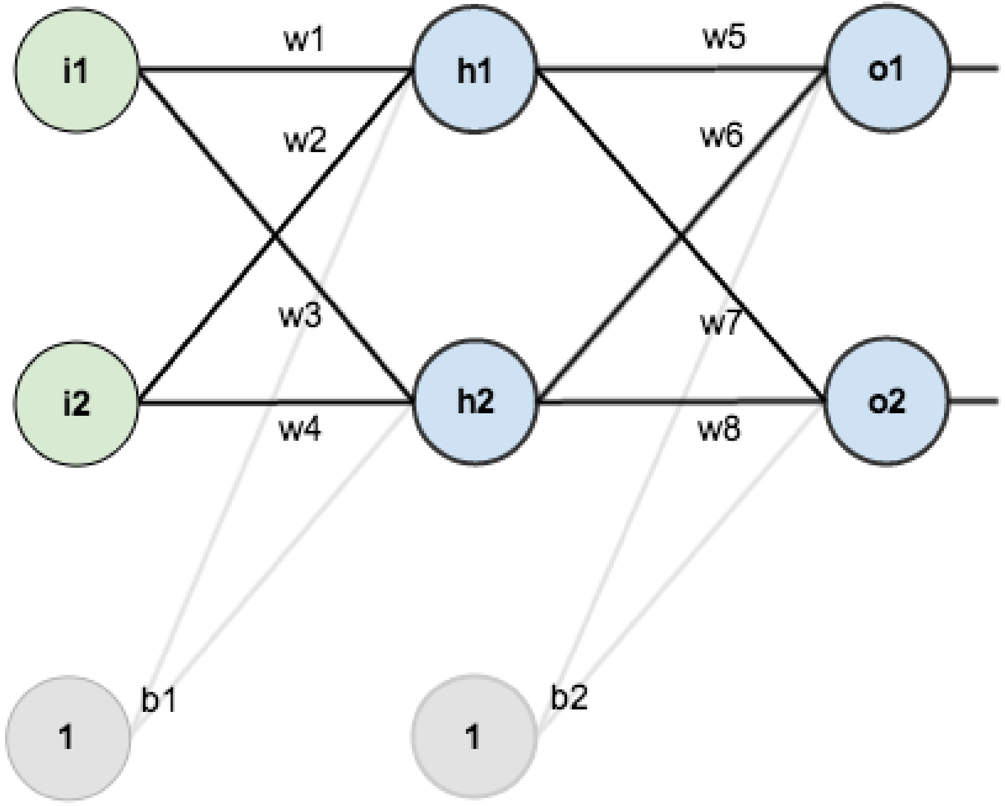
Tek ve Çok Katmanlı Sinir Ağı Yapısı



- Tek katmanlı YSA modelinde 4+2=6 nöron bulunmaktadır (giriş katmanları hariç), [3x4]+[4x2]=20 ağırlık ve 4+2=6 bias değeri olmak üzere toplamda 26 adet öğrenilmesi gereken parametre vardır.

- İki gizli katmanlı YSA ise modelinde 4+4+1=9 nöron, [3x4]+[4x4]+[4x1]=12+16+4=32 ağırlık ve 4+4+1=p bias değeri olmak üzere toplamda 41 adet öğrenilmesi gereken parametre vardır.

Örnek bir hesaplama



- İki giriş ve iki çıkışa sahip olan tek ara katmanlı bir YSA modelinin ileri hesaplamasını şöyledir;
- $X = [0.05\ 0.10]$
- $W^i = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} = \begin{bmatrix} 0.15 & 0.20 \\ 0.25 & 0.30 \end{bmatrix}$
- $B = [b_1\ b_2] = [0.35\ 0.60]$
- $W^h = \begin{bmatrix} w_5 & w_6 \\ w_7 & w_8 \end{bmatrix} = \begin{bmatrix} 0.40 & 0.45 \\ 0.50 & 0.55 \end{bmatrix}$
- $T = [t_1\ t_2] = [0.01\ 0.99]$
- $\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = f(W^i * X + B)$
- Burada *f* aktivasyon fonksiyonudur ve türevinin alınabiliyor olması gerekir. Aktivasyon fonksiyonu olarak lojistik fonksiyonu kullanılırsa;
- $f(x) = {}^{1}/_{1+e^{-x}}$

Örnek bir hesaplama (İleri yön hesaplama)

- $h_1 net = w_1 * i_1 + w_2 * i_2 + b_1 * 1$
- $= 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$
- $out_{h1} = {}^{1}/_{1+e^{-0.3775}} = 0.5933$
- $h_2 net = w_3 * i_1 + w_4 * i_2 + b_1 * 1$
- $= 0.05 * 0.25 + 0.1 * 0.30 + 0.35 * 1 = 0.3925$
- $out_{h2} = {}^{1}/_{1+e^{-0.3925}} = 0.5969$
- $o_1 net = out_{h1} * w_5 + out_{h2} * w_6 + b_2 * 1$
- $= 0.5933 * 0.4 + 0.5969 * 0.45 + 0.6 * 1 = 1.1059$
- $out_{o1} = {}^{1}/_{1+e^{-1.1059}} = 0.7514$

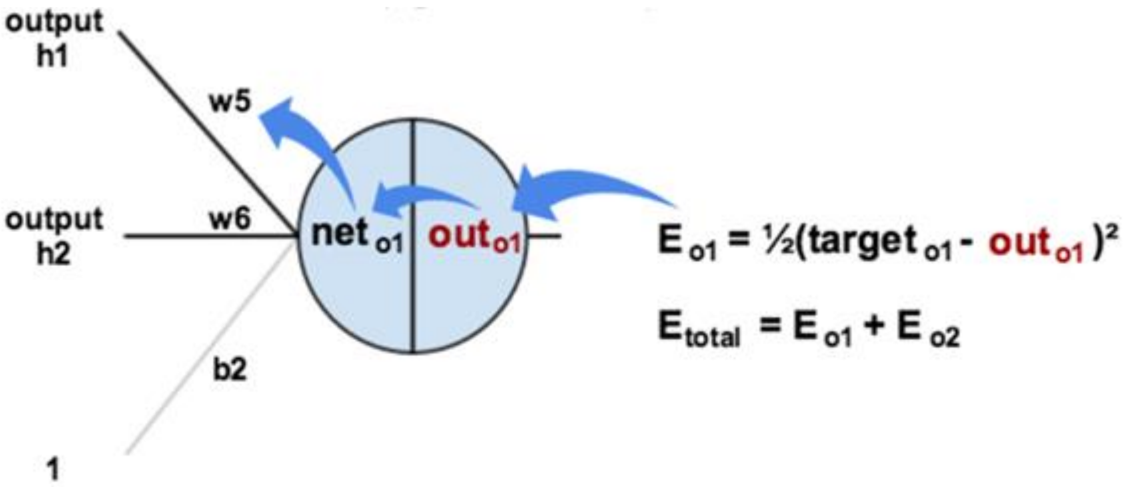
Örnek bir hesaplama (İleri yön hesaplama)

- $o_2net = out_{h1} * w_7 + out_{h2} * w_8 + b_2 * 1$
- $= 0.5933 * 0.5 + 0.5969 * 0.55 + 0.6 * 1 = 1.2249$
- $out_{o2} = {}^{1}\!/_{1+e^{-1.2249}} = 0.7729$
- $E_i = 0.5(T_i - outo_i)^2$
- Birinci ve ikinci çıkışlar için elde edilen hatalar;
- $E_{o1} = 0.5 * (0.01 - 0.7514)^2 = 0.2748$
- $E_{o2} = 0.5 * (0.99 - 0.7729)^2 = 0.0236$
- $E_{toplam} = E_{o1} + E_{o2} = 0.2748 + 0.0236 = 0.2984$

Örnek bir hesaplama (Geriye doğru hesaplama)

- İleri doğru hesaplama sonunda elde edilen hata değerinin, ağırlıkların ve bias değerlerinin güncellenmesinde kullanılacaktır. Örneğin w_5 ağırlığını düşünelim;
- Hatanın w_5 ağırlığına göre parçalı türevi ya da gradyanı;
- $\frac{\partial E_{toplam}}{\partial w_5}$ olarak ifade edilir.
- $\frac{\partial E_{toplam}}{\partial w_5} = \frac{\partial E_{toplam}}{\partial out_{o1}} \frac{\partial out_{o1}}{\partial o_1 net} \frac{\partial o_1 net}{\partial w_5}$

Örnek bir hesaplama (Geriye doğru hesaplama)



$$E_{toplam} = 0.5(T_1 - outo_1)^2 + 0.5(T_2 - outo_2)^2$$

$$\frac{\partial E_{toplam}}{\partial out_{o1}} = 2 * 0.5 * (T_1 - outo_1) * (-1) = outo_1 - T_1 = 0.7514 - 0.01 = 0.7414$$

$$outo_1 = \frac{1}{1 + e^{-o1net}}$$

$$\frac{\partial out_{o1}}{\partial o_1 net} = out_{o1} * (1 - out_{o1}) = 0.7514 * (1 - 0.7514) = 0.1868$$

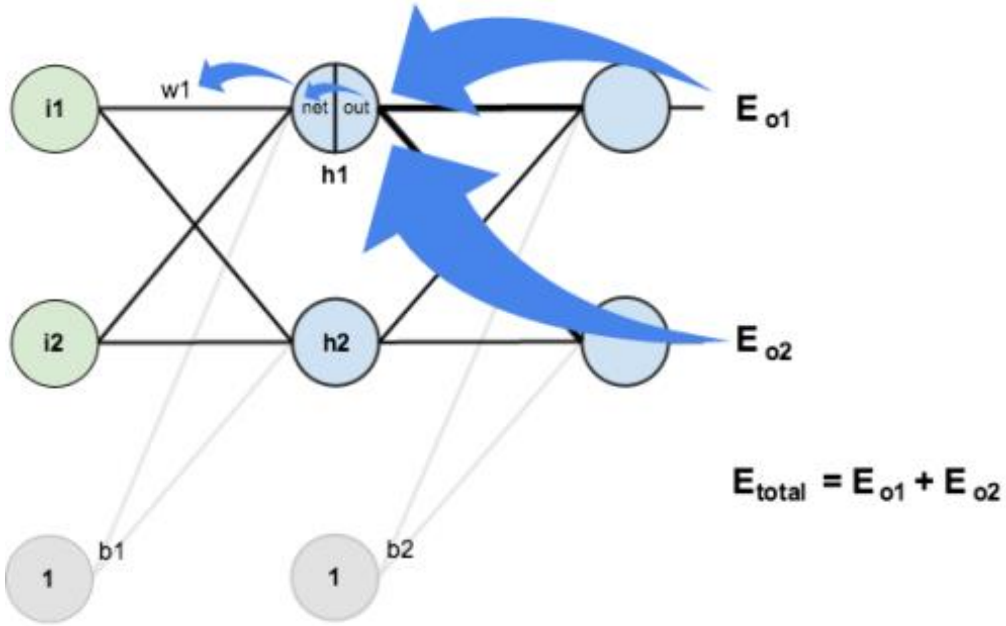
Örnek bir hesaplama (Geriye doğru hesaplama)

- $o_1 net = out_{h1} * w_5 + out_{h2} * w_6 + b_2 * 1$ olduğuna göre;
- $\frac{\partial o_1 net}{\partial w_5} = out_{h1} = 0.5933$
- Eğer bütün parçalar birleştirilirse;

- $\frac{\partial E_{toplam}}{\partial w_5} = 0.7414 * 0.1868 * 0.5933 = 0.0822$
- Böylece yeni w_5 ağırı şöyle hesaplanır; burada η öğrenme oranı olarak adlandırılır.
- $w^n{}_5 = w_5 - \eta * \frac{\partial E_{toplam}}{\partial w_5} = 0.4 - 0.5 * 0.0822 = 0.3589$

Örnek bir hesaplama (Geriye doğru hesaplama)

- Şimdide ara katman ile giriş katmanı arasındaki ağırlıkların yenilenmesine bakalım.
- $\frac{\partial E_{toplam}}{\partial w_1} = \frac{\partial E_{toplam}}{\partial out_{h1}} \frac{\partial out_{h1}}{\partial h_1 net} \frac{\partial h_1 net}{\partial w_1}$
- $\frac{\partial E_{toplam}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$



Örnek bir hesaplama (Geriye doğru hesaplama)

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial o_1 net}\frac{\partial o_1 net}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial o_1 net} = \frac{\partial E_{o1}}{\partial out_{o1}}\frac{\partial out_{o1}}{\partial o_1 net} = 0.7414 * 0.1868 = 0.1385$$

$$\frac{\partial o_1 net}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial o_1 net}\frac{\partial o_1 net}{\partial out_{h1}} = 0.1385 * 0.40 = 0.0554$$

Örnek bir hesaplama (Geriye doğru hesaplama)

- Aynı işlemler $\frac{\partial E_{o2}}{\partial out_{h1}}$ için de yapılırsa;
- $\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019$ olarak hesaplanır.
- Böylece;
- $\frac{\partial E_{toplam}}{\partial out_{h1}} = 0.0364$ elde edilir ve $\frac{\partial out_{h1}}{\partial h_1 net}, \frac{\partial h_1 net}{\partial w_1}$ nin bulunmasını gerektirir.

Örnek bir hesaplama (Geriye doğru hesaplama)

$$out_{h1} = \frac{1}{1 + e^{-h1(net)}}$$

$$\frac{\partial out_{h1}}{\partial h_1 net} = out_{h1} * (1 - out_{h1}) = 0.5933 * (1 - 0.5933) = 0.2413$$

$$h_1 net \ = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial h_1 net}{\partial w_1} = i_1 = 0.05$$

Örnek bir hesaplama (Geriye doğru hesaplama)

- $\frac{\partial E_{toplam}}{\partial w_1} = \frac{\partial E_{toplam}}{\partial out_{h1}} \frac{\partial out_{h1}}{\partial h_1 net} \frac{\partial h_1 net}{\partial w_1}$
- $\frac{\partial E_{toplam}}{\partial w_1} = 0.0364 * 0.2413 * 0.05 = 0.0004$ elde edilir.

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

- $w_1^n = w_1 - \eta * \frac{\partial E_{toplam}}{\partial w_1} = 0.15 - 0.5 * 0.004 = 0.1498$

Geri Yayılım Algoritmasının MATLAB Uygulaması

```
%% Giriş parametreleri ve değerleri;
X  = [0.05 0.10];   % İki boyutlu giriş vektörü
Wi = [0.15 0.20     % Giriş ile gizli katman arasındaki ağırlıklar
      0.25 0.30];
Bh = [0.35 0.35];   % Gizli katmanı biası
Bo = [0.60 0.60];   % Çıkış katmanı biası
Wh = [0.40 0.45     % Gizli katman ile çıkış katmanı arasındaki
      0.50 0.55];   % ağırlıklar
mu = .01;           % Öğrenme oranı
T  = [0.01 0.99]'; % Arzu edilen çıkış
```

Geri Yayılım Algoritmasının MATLAB Uygulaması

```
for i = 1:1000
h_net  = Wi*X'+Bh'; % Gizli katman nöronlarında elde edilen değerler
oh_net = 1./(1+exp(-h_net))';  % Gizli katman çıkışı
o_net  = Wh*oh_net'+Bo'; % Çıkış katman nöronlarında elde edilen
                         % değerler
oo_net = 1./(1+exp(-o_net));   % Çıkış katmanı çıkışı
Etotal = mean(0.5*sum((T-oo_net').^2)); % Toplam hata
plot(i,Etotal,'r.-')
      if mod(i,200)==0
      text(i,Etotal,num2str(Etotal));
      end
```

Geri Yayılım Algoritmasının MATLAB Uygulaması

```
title(['iteration number: ',num2str(i)]);
xlim([0 1000])
ylim([0.28 0.31])
grid on
hold on
drawnow
dEh  = (oo_net-T).*(oo_net.*(1-oo_net)).*oh_net;
Wh_n = Wh-mu*dEh;
dEi  = sum((oo_net-T).*(oo_net.*(1-oo_net)).*Wh).*(oh_net.*(1-oh_net)).*X;
Wi_n = Wi-mu*dEi;
Wh   = Wh_n;
Wi   = Wi_n;
end
```

MATLAB Neural Networks Örnekleri

```
close all, clear all, clc, format compact

% Neuron weights
w = [4 -2]
% Neuron bias
b = -3
% Activation function
 func = 'tansig'
% func = 'purelin'
% func = 'hardlim'
% func = 'logsig'
```

```
p = [2 3]
```

MATLAB Neural Networks Örnekleri

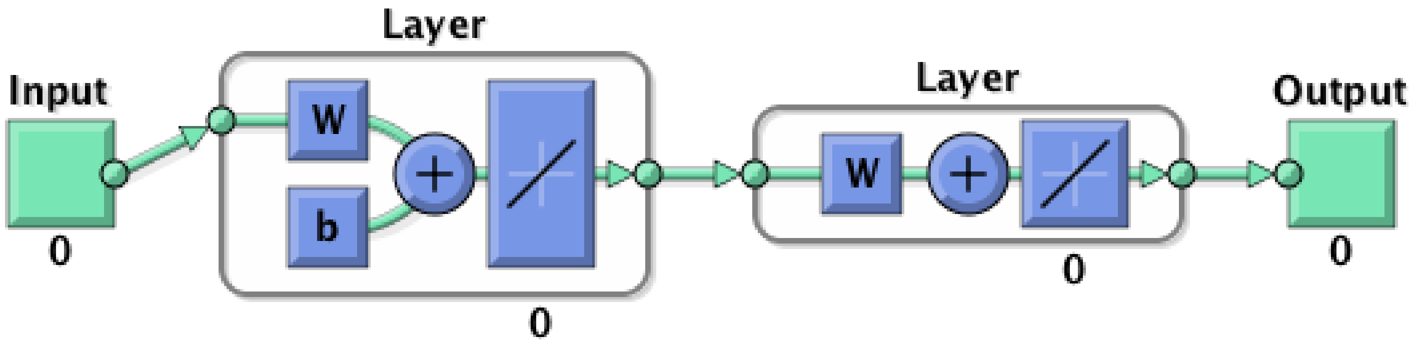
```
activation_potential = p*w'+b
```

```
neuron_output = feval(func, activation_potential)
```

İstenilen bir ağ mimarisini oluşturma

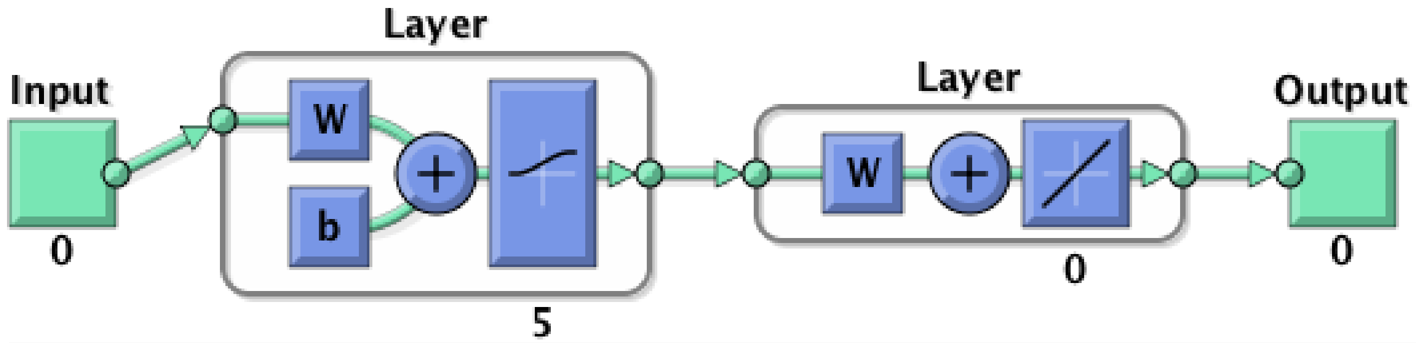
```
% Ağı oluştur
net = network( ...
1,            ... % numInputs,    Giriş sayısı,
2,            ... % numLayer,     Katman sayısı
[1; 0],       ... % biasConnect,  Bias bağlantıları
[1; 0],       ... % layerConnect, Katman bağlantıları
[0 0; 1 0],   ... % layerConnect, numLayers-by-numLayers Boolean matrix
[0 1]         ... % outputConnect, 1-by-numLayers Boolean vector
);
% View network structure
view(net);
```

MATLAB Neural Networks Örnekleri

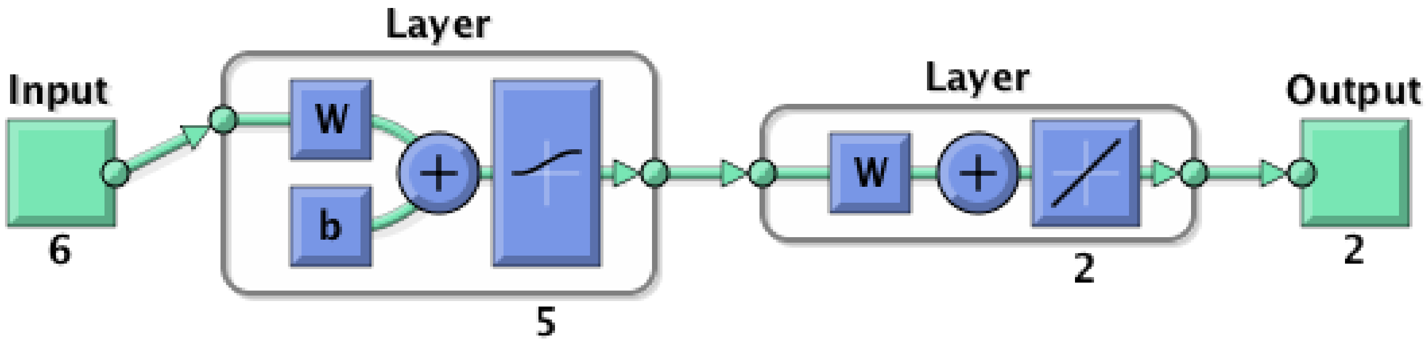



```
% number of hidden layer neurons
net.layers{1}.size = 5;

% hidden layer transfer function
net.layers{1}.transferFcn = 'logsig';
view(net);
```

MATLAB Neural Networks Örnekleri




```
inputs  = [1:6]' % input vector (6-dimensional pattern)
outputs = [1 2]' % corresponding target output vector

net = configure(net,inputs,outputs);
view(net);
```



MATLAB Neural Networks Örnekleri

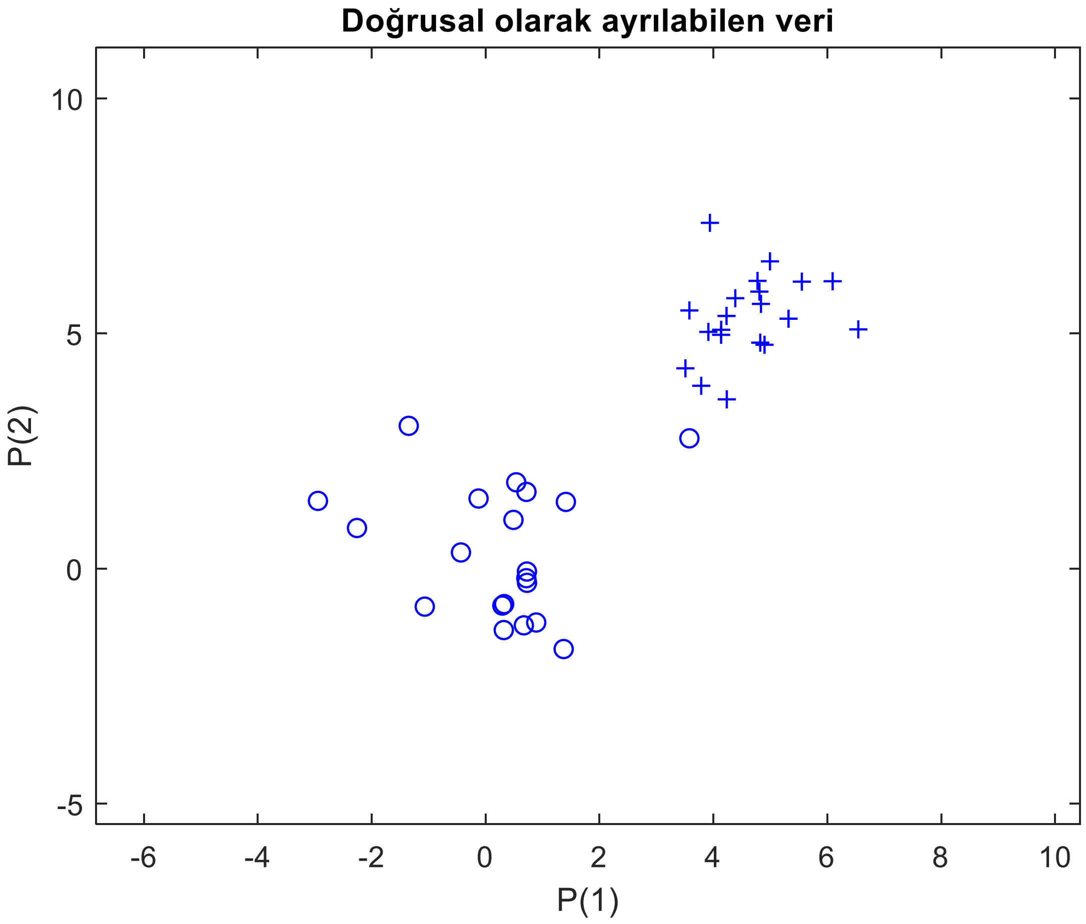
```
% initial network response without training
initial_output = net(inputs)

% network training
net.trainFcn = 'trainlm';
net.performFcn = 'mse';
net = train(net,inputs,outputs);

% network response after training
final_output = net(inputs)
```

MATLAB Neural Networks Örnekleri

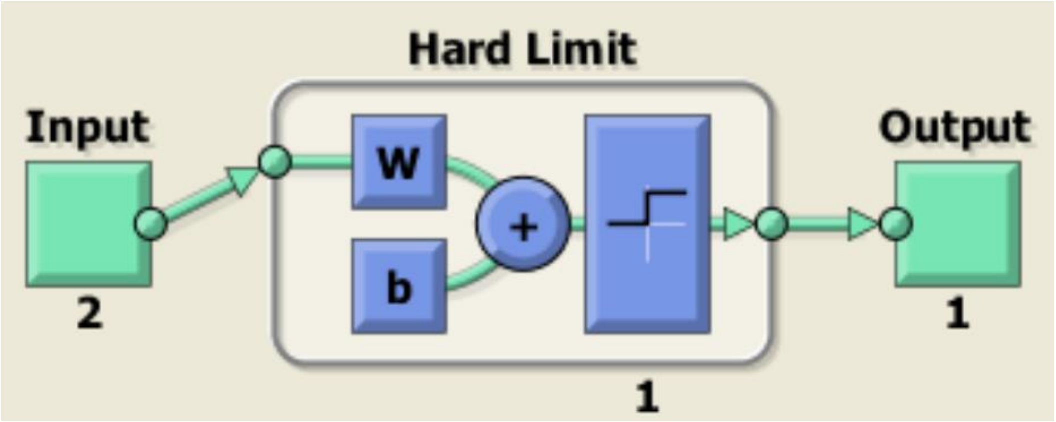
- Bir perceptron ile doğrusal ayrılabilen verinin sınıflandırılması

```
close all, clear all, clc, format compact

% number of samples of each class
N = 20;
% define inputs and outputs
offset = 5; % offset for second class
x = [randn(2,N) randn(2,N)+offset]; % inputs
y = [zeros(1,N) ones(1,N)];         % outputs

% Plot input samples with PLOTPV (Plot perceptron input/target vectors)
figure(1)
plotpv(x,y);
```

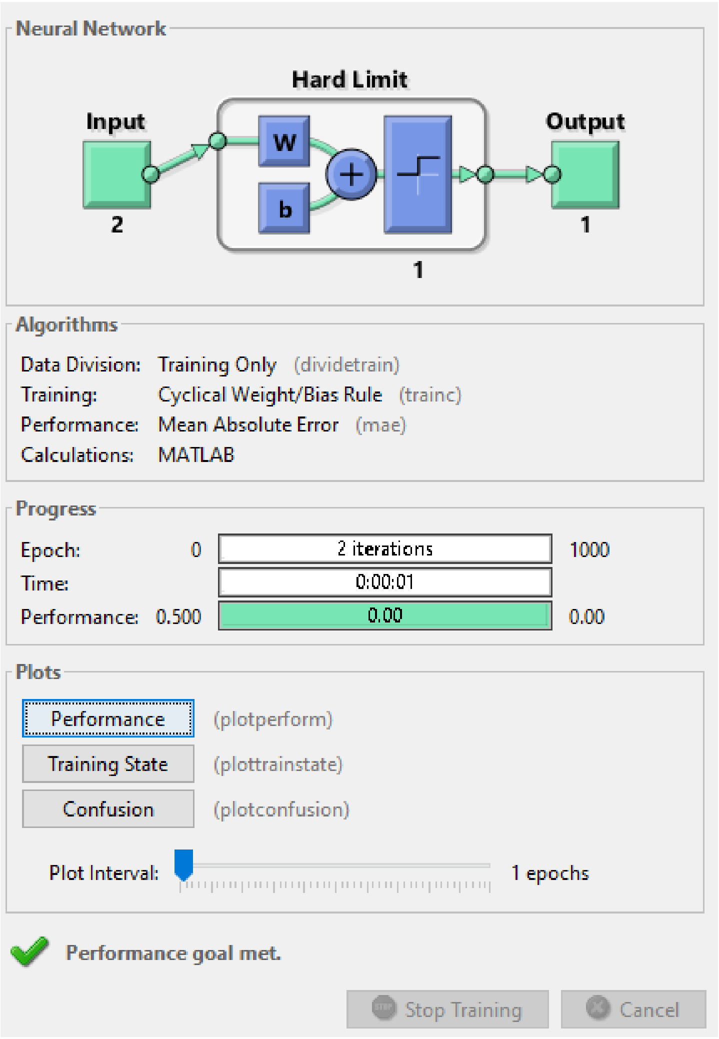
MATLAB Neural Networks Örnekleri



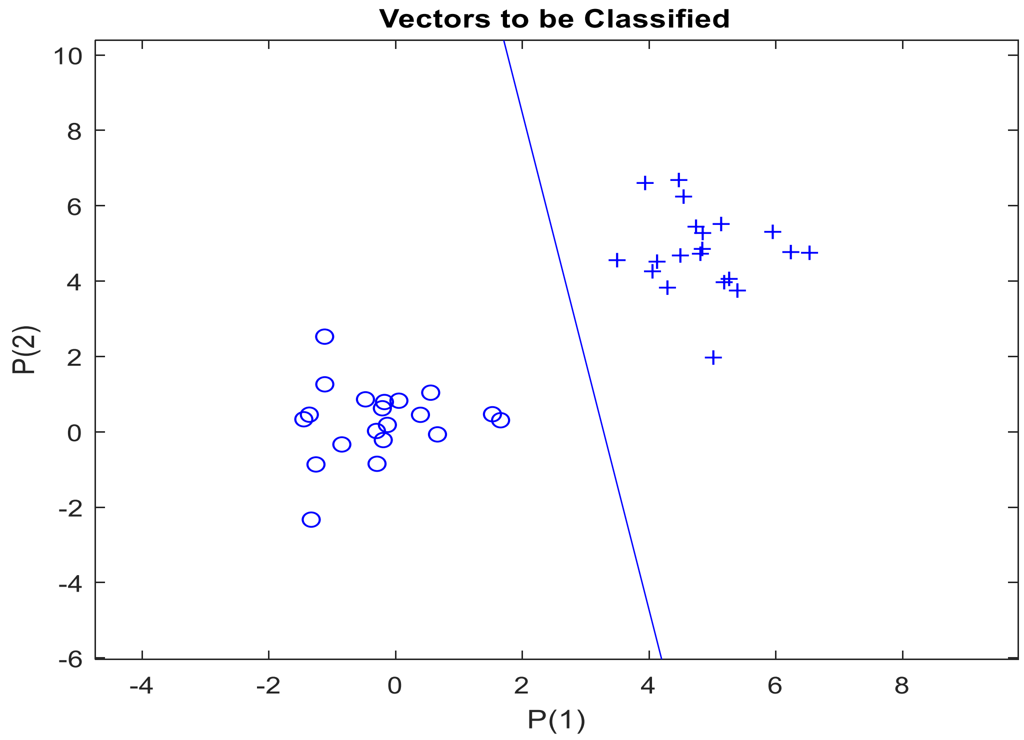

```
net = perceptron;
net = train(net,x,y);
view(net);
```



MATLAB Neural Networks Örnekleri




```
figure(1)
plotpc(net.IW{1},net.b{1});
```



MATLAB Neural Networks Örnekleri

- Perceptron ile 4 sınıflı bir sınflandırma problemi

```
close all, clear all, clc, format compact

% number of samples of each class
K = 30;

% define classes
q = .6; % offset of classes
A = [rand(1,K)-q; rand(1,K)+q];
B = [rand(1,K)+q; rand(1,K)+q];
C = [rand(1,K)+q; rand(1,K)-q];
D = [rand(1,K)-q; rand(1,K)-q];
```

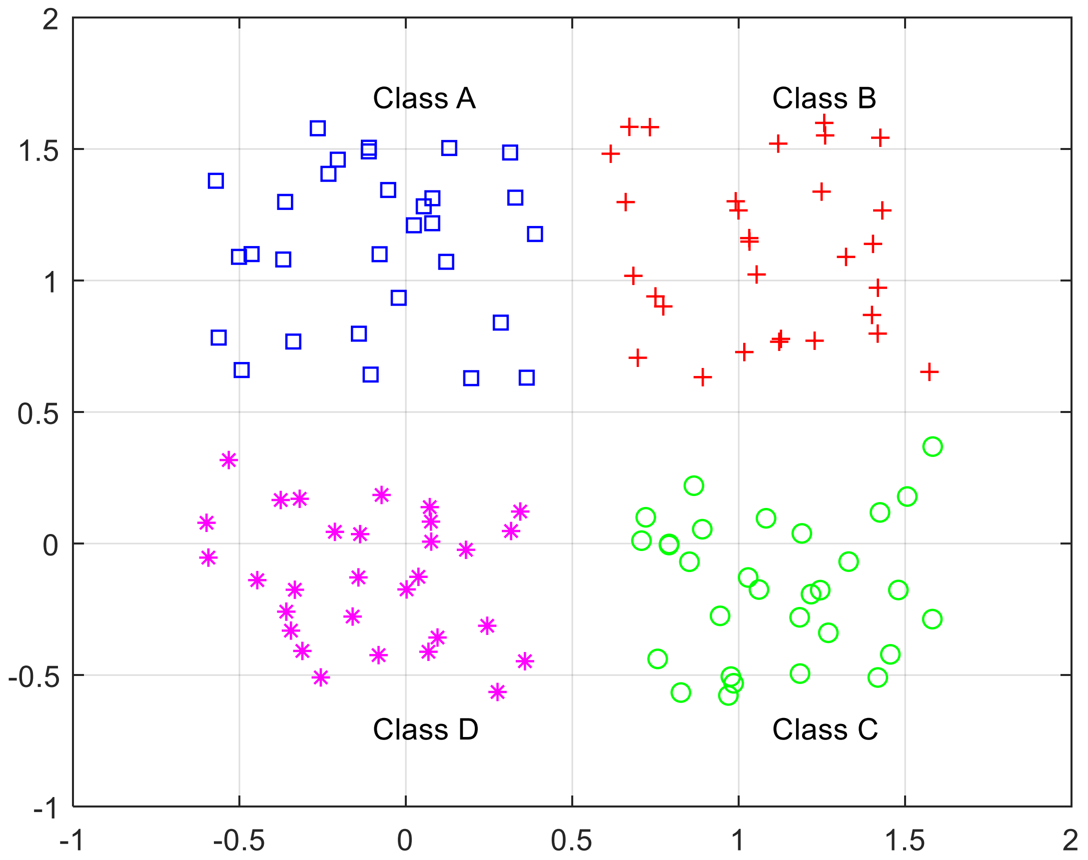
MATLAB Neural Networks Örnekleri

```
% plot classes
plot(A(1,:),A(2,:),'bs')
hold on
grid on
plot(B(1,:),B(2,:),'r+')
plot(C(1,:),C(2,:),'go')
plot(D(1,:),D(2,:),'m*')
```

```
% text labels for classes
text(.5-q,.5+2*q,'Class A')
text(.5+q,.5+2*q,'Class B')
text(.5+q,.5-2*q,'Class C')
text(.5-q,.5-2*q,'Class D')
```

```
% define output coding for classes
a = [0 1]';
b = [1 1]';
c = [1 0]';
d = [0 0]';
```

MATLAB Neural Networks Örnekleri



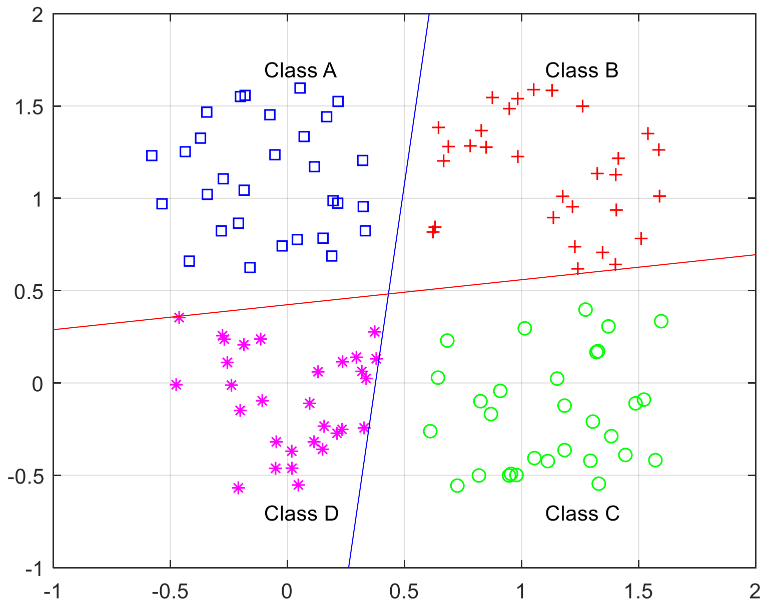

```
% define inputs (combine samples from all four classes)
P = [A B C D];
% define targets
T = [repmat(a,1,length(A))...
     repmat(b,1,length(B)) ...
     repmat(c,1,length(C)) ...
     repmat(d,1,length(D)) ];
```

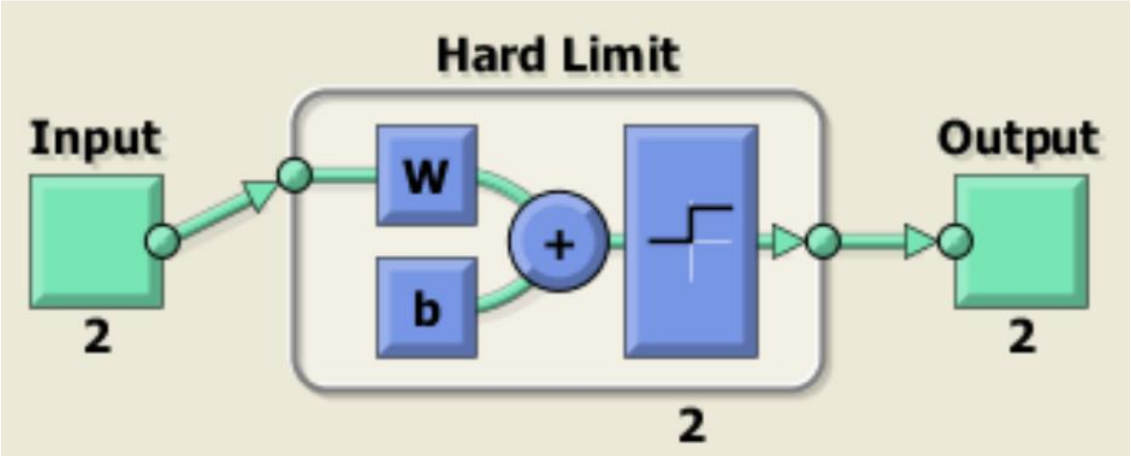
```
net = perceptron;
```

MATLAB Neural Networks Örnekleri

```
E = 1;
net.adaptParam.passes = 1;
linehandle = plotpc(net.IW{1},net.b{1});
n = 0;
while (sse(E) & n<1000)
    n = n+1;
    [net,Y,E] = adapt(net,P,T);
    linehandle = plotpc(net.IW{1},net.b{1},linehandle);
    drawnow;
end
% show perceptron structure
view(net);
```

MATLAB Neural Networks Örnekleri






```
% For example, classify an input vector of [0.7; 1.2]
p = [0.7; 1.2]
y = net(p)
```

```
p =
    0.7000
    1.2000
y =
     1
     1
```

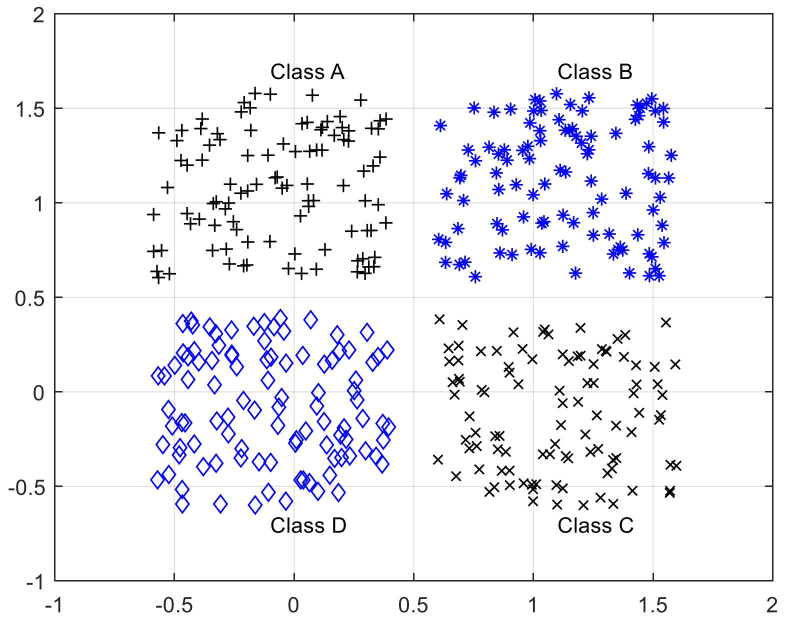
MATLAB Neural Networks Örnekleri

- Çok katmanlı perceptron yapısı ile 4 sınıflı veri sınıflandırma

```
close all, clear all, clc, format compact

% number of samples of each class
K = 100;
% define 4 clusters of input data
q = .6; % offset of classes
A = [rand(1,K)-q; rand(1,K)+q];
B = [rand(1,K)+q; rand(1,K)+q];
C = [rand(1,K)+q; rand(1,K)-q];
D = [rand(1,K)-q; rand(1,K)-q];
```

```
plot(A(1,:),A(2,:),'k+')
hold on
grid on
plot(B(1,:),B(2,:),'b*')
plot(C(1,:),C(2,:),'kx')
plot(D(1,:),D(2,:),'bd')
% text labels for clusters
text(.5-q,.5+2*q,'Class A')
text(.5+q,.5+2*q,'Class B')
text(.5+q,.5-2*q,'Class C')
text(.5-q,.5-2*q,'Class D')
```

MATLAB Neural Networks Örnekleri



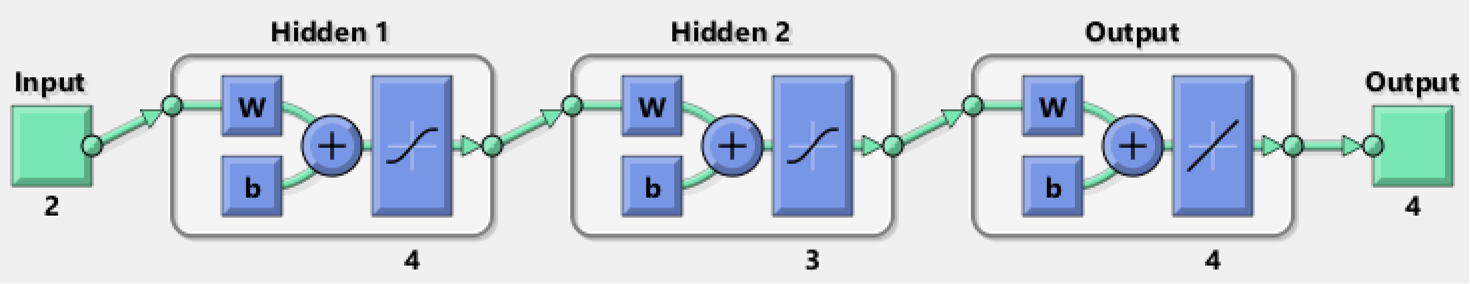

```
% coding (+1/-1) of 4 separate classes
a = [-1 -1 -1 +1]';
b = [-1 -1 +1 -1]';
d = [-1 +1 -1 -1]';
c = [+1 -1 -1 -1]';
```

```
% define inputs (combine samples from all four classes)
P = [A B C D];
% define targets
T = [repmat(a,1,length(A))...
     repmat(b,1,length(B)) ...
     repmat(c,1,length(C)) ...
     repmat(d,1,length(D)) ];
```

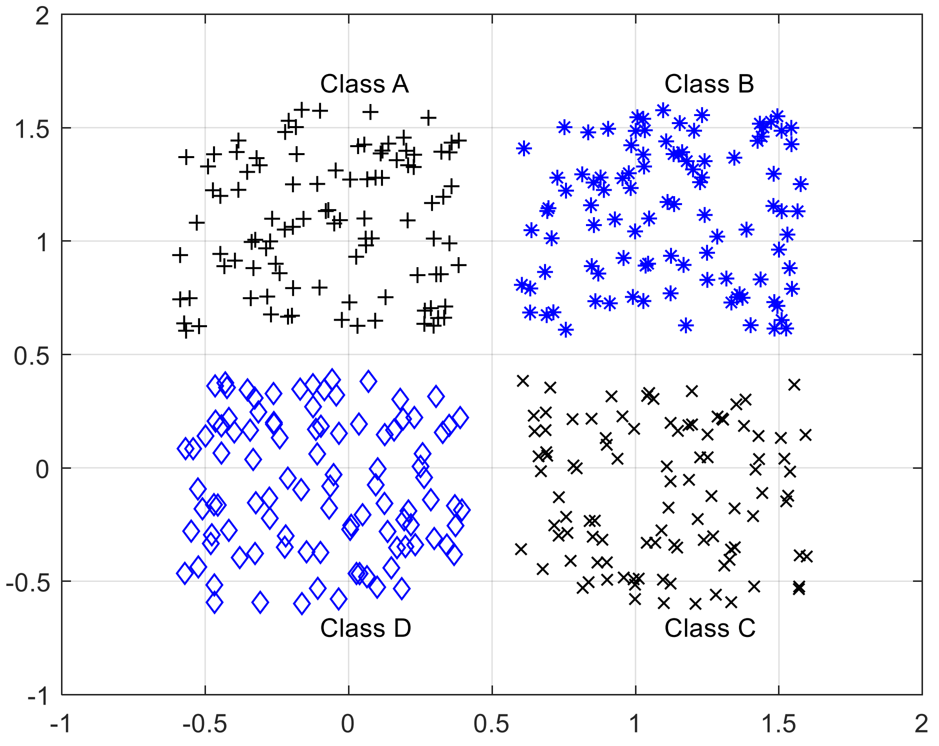
MATLAB Neural Networks Örnekleri

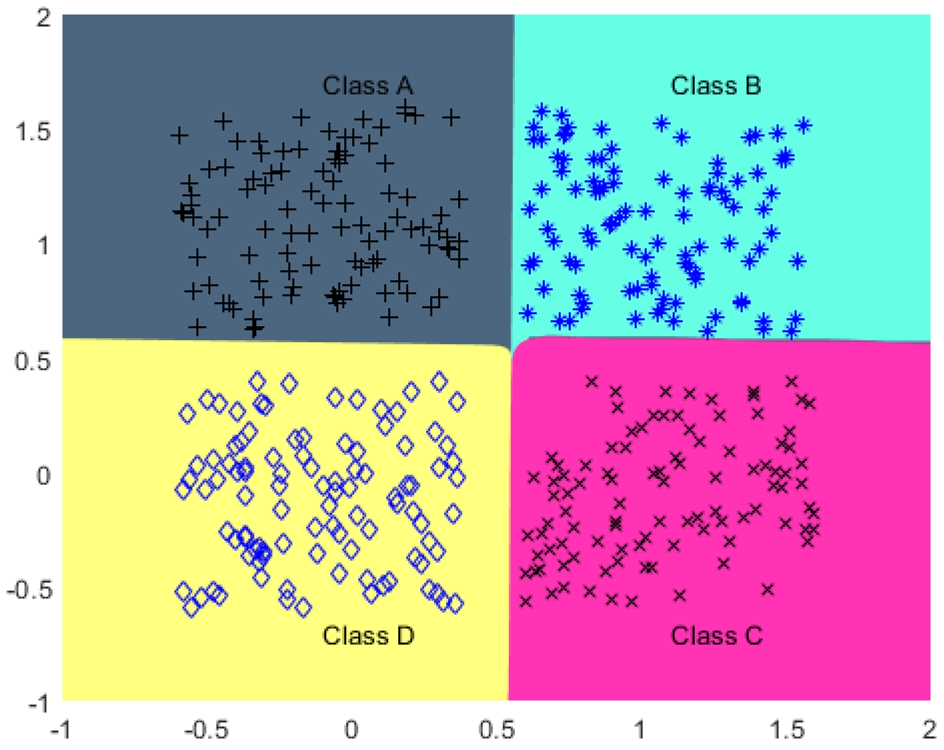
```
% create a neural network
net = feedforwardnet([4 3]);

% train net
net.divideParam.trainRatio = 1; % training set [%]
net.divideParam.valRatio   = 0; % validation set [%]
net.divideParam.testRatio  = 0; % test set [%]

% train a neural network
[net,tr,Y,E] = train(net,P,T);
```

```
% generate a grid
span = -1:.01:2;
[P1,P2] = meshgrid(span,span);
pp = [P1(:) P2(:)]';

% simualte neural network on a grid
aa = net(pp);
```

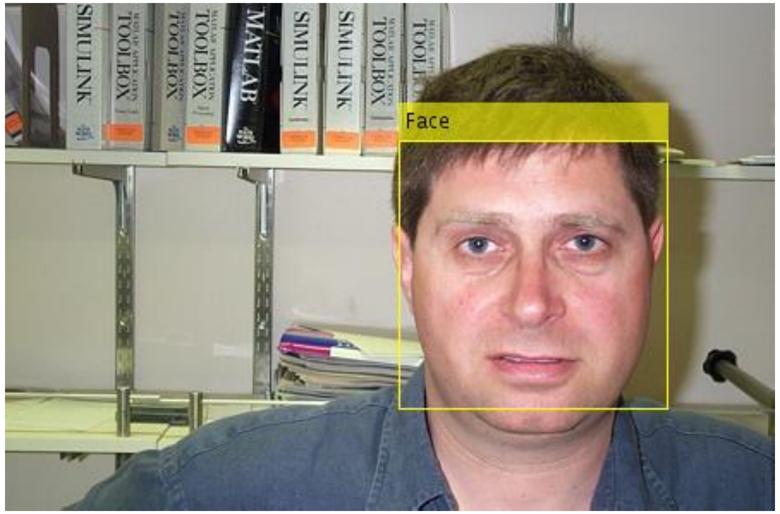


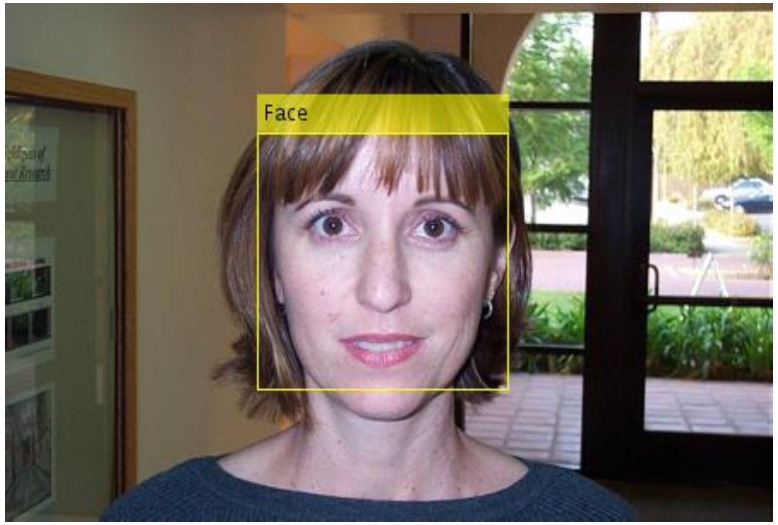
MATLAB Neural Networks Örnekleri





Matlab ile Yüz Tanıma






```
clc,clear
yol = '.\imgeler\';
liste = dir(yol);
liste = liste(3:end,1);
faceDetector = vision.CascadeObjectDetector();
for i=1:length(liste)
    img=imread([yol,'\',liste(i).name]);
    img = imresize(img,0.5);
    bbox            = step(faceDetector, img);
    ind=find(bbox(:,3)==max(bbox(:,3)));
    bbox=bbox(ind,:);
    videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Face');
    face_region = img(bbox(2):bbox(2)+bbox(3),bbox(1):bbox(1)+bbox(4),:);
    videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Face');
    figure, imshow(videoOut), title('Detected face');
    face_imgs(:,:,i) = rgb2gray(imresize(face_region,[32 32]));
end
```

Matlab ile Yüz Tanıma



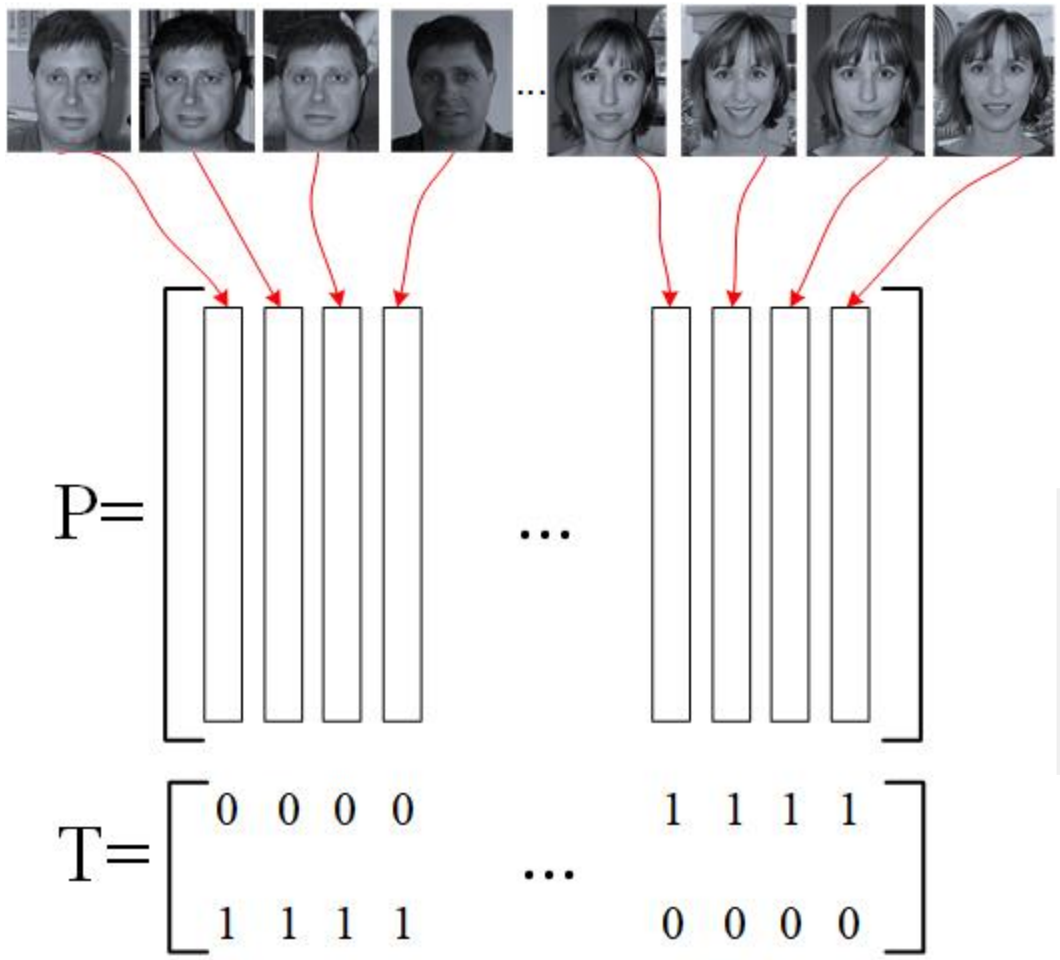

Ahmet ☺



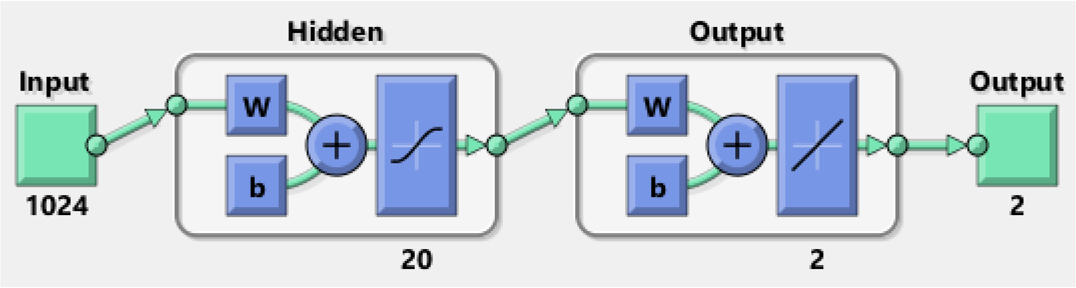

Ayşe ;)

Matlab ile Yüz Tanıma

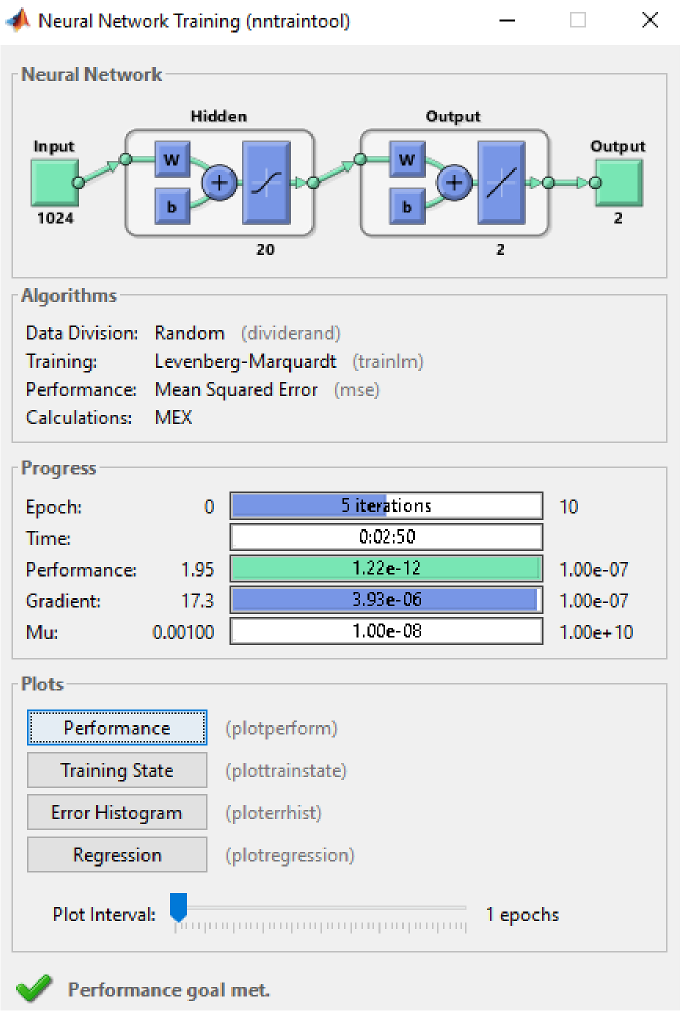
Matlab ile Yüz Tanıma




```
P = [];
for i=1:length(liste)
     im_vec=face_imgs(:,:,i);
     P=[P im_vec(:) ];
end
P= zscore(double(P));
a= [0 1]';
b= [1 0]';
T =[repmat(a,1,21) repmat(b,1,21) ];
```



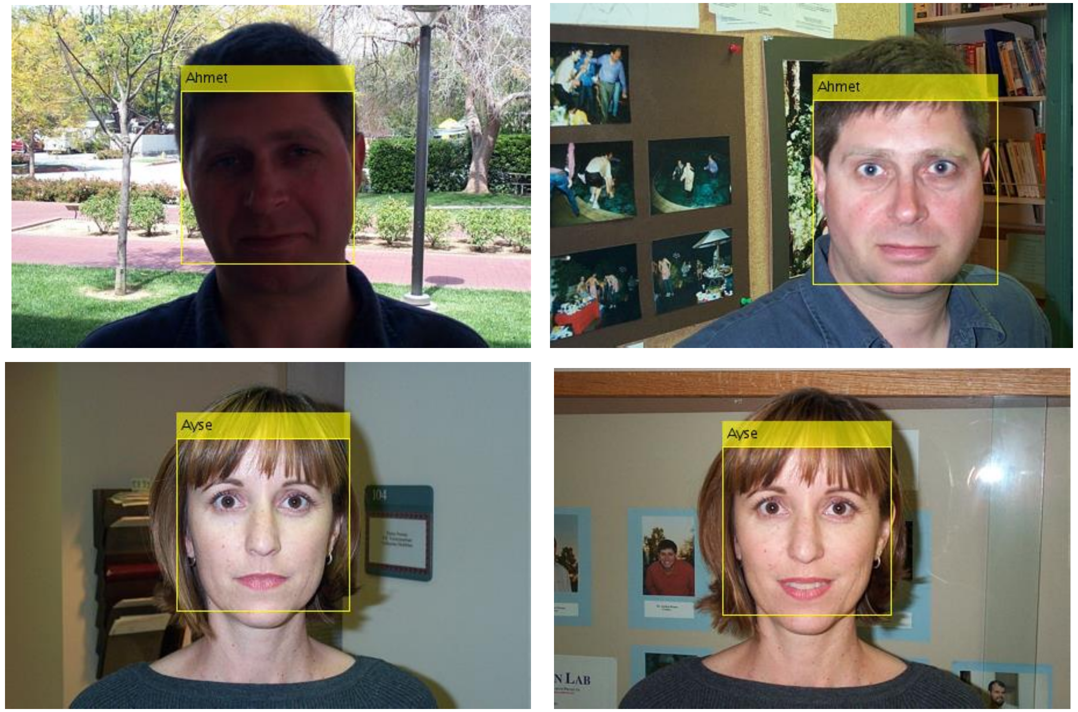
Matlab ile Yüz Tanıma



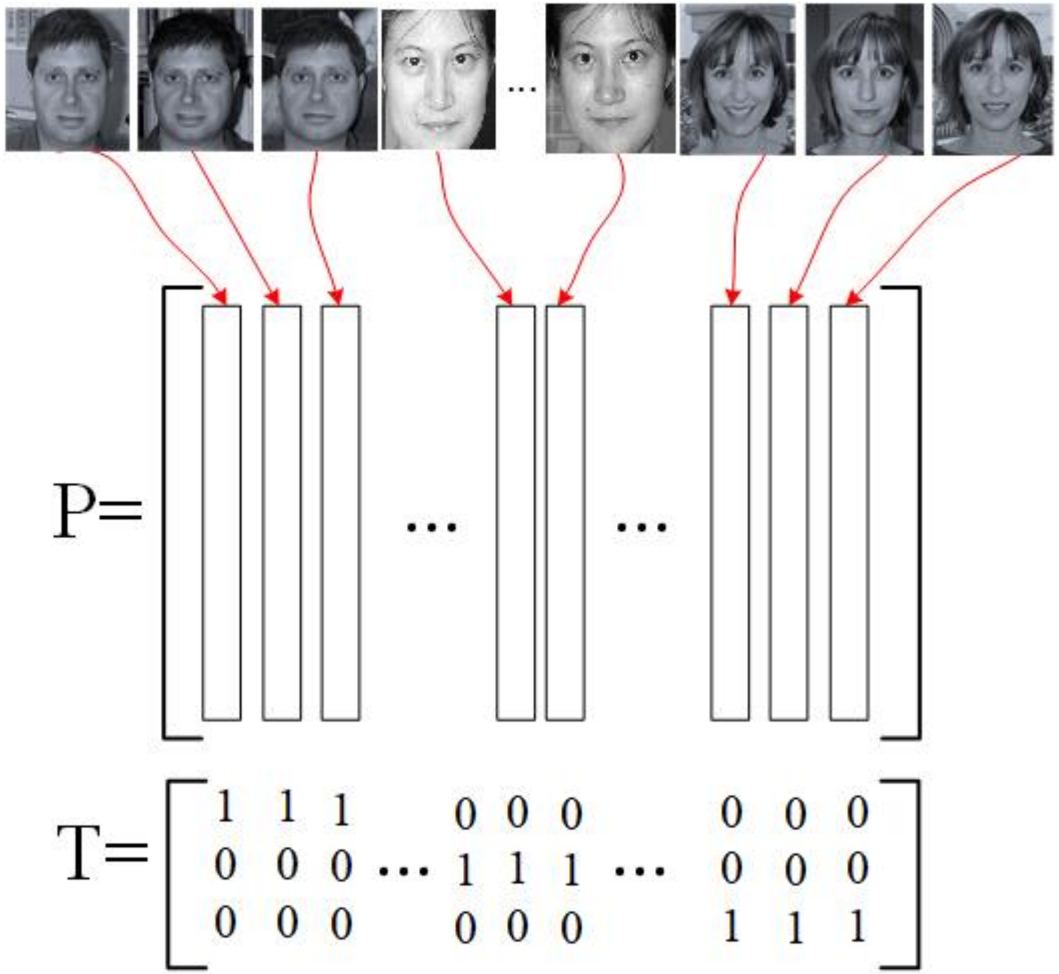

```
net = feedforwardnet(20);
net.divideParam.trainRatio = .75; % training set [%]
net.divideParam.valRatio = 0; % validation set [%]
net.divideParam.testRatio = .25; % test set [%]
net.trainFcn = 'trainlm';
net.layers{1}.transferFcn = 'tansig';
net.trainParam.goal = 1e-7;
net.trainParam.lr = 1e-3;
net.trainparam.epochs = 10;
[net,tr,Y,E] = train(net,P,T);
```

Matlab ile Yüz Tanıma

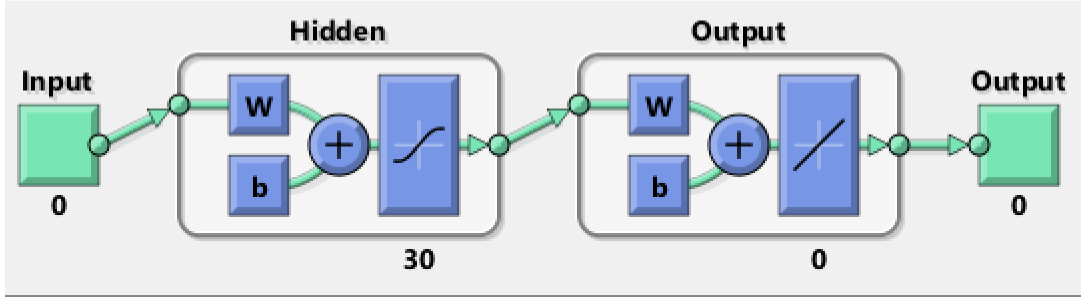
```
clc,clear
yol = '.\imgeler\';
liste = dir(yol);
liste = liste(3:end,1);
faceDetector = vision.CascadeObjectDetector();
load face.mat
a= [0 1]';
b= [1 0]';
for i=1:42
    img=imread([yol,'\',liste(i).name]);
    img = imresize(img,0.5);
    bbox            = step(faceDetector, img);
    ind=find(bbox(:,3)==max(bbox(:,3)));
    bbox=bbox(ind,:);
    face_region = img(bbox(2):bbox(2)+bbox(3),bbox(1):bbox(1)+bbox(4),:);
    face_imgs = rgb2gray(imresize(face_region,[32 32]));
    P = zscore(double(face_imgs(:)));
    out = net(P);
    if sum((out-a).^2)> sum((out-b).^2)
    videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Ayse');
    imshow(videoOut)
    else
    videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Ahmet');
    imshow(videoOut)
    end
     pause(1)
end
```



Matlab ile Yüz Tanıma




```
P = [];
for i=1:length(liste)
     im_vec=face_imgs(:,:,i);
     P=[P im_vec(:) ];
end
P= zscore(double(P));
a= [1 0 0]';
b= [0 1 0]';
c= [0 0 1]';
T =[repmat(a,1,21) repmat(b,1,21) repmat(c,1,21) ];
```

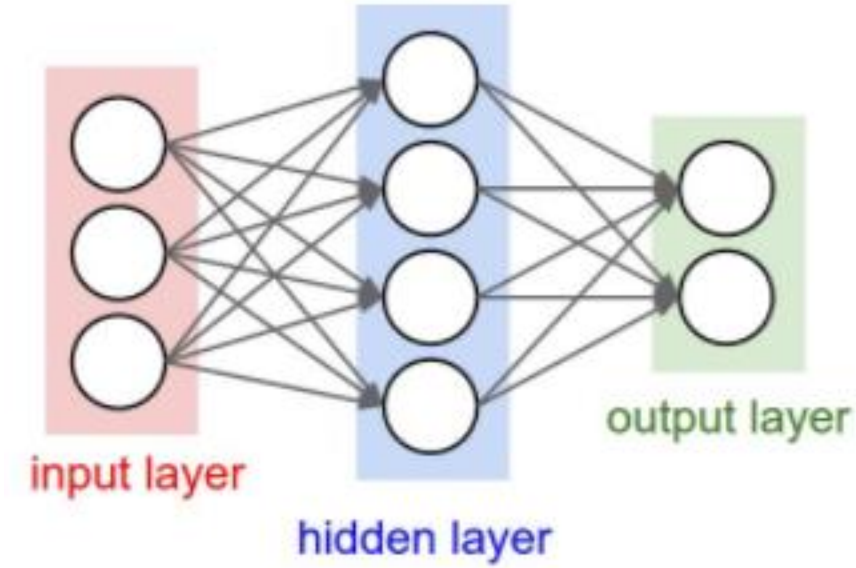


Matlab ile Yüz Tanıma

```
clc,clear
yol = '.\imgeler\';
liste = dir(yol);
liste = liste(3:end,1);
faceDetector = vision.CascadeObjectDetector();
load face.mat
a= [1 0 0]';
b= [0 1 0]';
c= [0 0 1]';
    for i=1:63
    img=imread([yol,'\',liste(i).name]);
    img = imresize(img,0.5);
    bbox            = step(faceDetector, img);
    ind=find(bbox(:,3)==max(bbox(:,3)));
    bbox=bbox(ind,:);
    face_region = img(bbox(2):bbox(2)+bbox(3),bbox(1):bbox(1)+bbox(4),:);
    face_imgs = rgb2gray(imresize(face_region,[32 32]));
    P = zscore(double(face_imgs(:)));
    out = abs(net(P));
        if find(out==max(out))==1
        videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Ahmet');
        imshow(videoOut)
        elseif find(out==max(out))==2
          videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Ayşe');
        imshow(videoOut)
        elseif find(out==max(out))==3
        videoOut = insertObjectAnnotation(img,'rectangle',bbox,'Zehra');
        imshow(videoOut)
        else
        videoOut = insertObjectAnnotation(img,'rectangle',bbox,'tanımlanamayan');
        imshow(videoOut)
        end
     pause(.3)
    end
```

Aşırı Öğrenme Makineleri (Extreme Learning Machines)

- Tek ara katmanlı ağ mimarisidir.
- Giriş-çıkış ilişkisini öğrenmek için eğitime İhtiyaç duymazlar. Öğrenme, en küçük kareler yöntemi ile 0 hata olacak şekilde halledelir eğer uygun şartlar yerine gelmiş ise…



N adet (x_i, y_i) çifti verilmiş olsun.

Bu arada $x_i \in R^n ve\ y_i \in R^n$ olabilir.

M adet ara katman nöronu için;

Aşırı Öğrenme Makineleri (Extreme Learning Machines)

$$\sum_{i=1}^{M} \beta_i g(w_i x_j + b_i) = o_i, j = 1, \dots, N$$

Burada, w_i, b_i $sırası\ ile\ ara\ katman\ ağırlıkları\ ve\ biasıdır\ ve\ \beta_i$ $ise\ ara\ katman\ ile\ çıkış\ katmanı\ arasındaki\ ağrılıklardır$. g, aktivasyon fonksiyonunu göstersin.

Eğer,

$$H = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_N.x_1 + b_N) \\ \vdots & \cdots & \vdots \\ g(w_1.x_M + b_1) & \cdots & g(w_N.x_M + b_N) \end{bmatrix}_{NxM}$$

ve

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_M^T \end{bmatrix}_{Mx1}$$

şeklinde ifade edilirse, çıkış $Y = H\beta$ elde edilir.

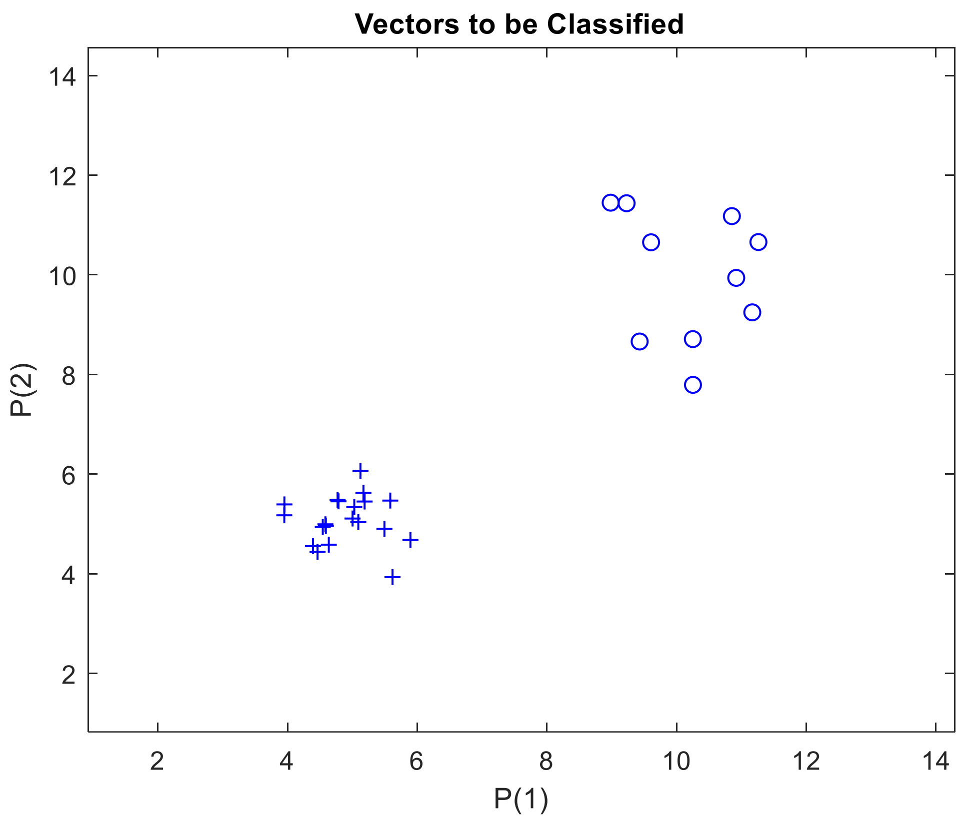
Aşırı Öğrenme Makineleri (Extreme Learning Machines)

Ara katman ile çıkış katmanı ağırlıkları $\hat{\beta} = H'Y$ şeklinde analitik olarak hesaplanır.

Aşırı öğrenme makinelerinde öğrenme işlemi, $w_i, b_i ve\ \beta_i$değerlerinin saklanması ile sağlanır. Öğrenmenin test edilmesi için; $Y = H\hat{\beta}$ hesaplanır.

Aşırı Öğrenme Makineleri (Extreme Learning Machines)

```
%% Eğitim
X=[normrnd(10,1,10,2);normrnd(5,.5,20,2)];
Y = [zeros(1,10) ones(1,20)]';
%plotpv(X',Y)
P= size(X,1);
N =size(X,2);
M = 15;% ara katman nöron sayısı
w= randn(M,N);
b= randn(M,1);
ind=ones(1,P);
bm= b(:,ind);
ac_func = 'purelin';
H = (w*X'+bm);
fH = feval(ac_func,H)
B = pinv(fH')*Y
Y_ussu= H'*B
```



Aşırı Öğrenme Makineleri (Extreme Learning Machines)

```
%% Test
Xt=[normrnd(10,1,10,2);normrnd(5,.5,25,2)];% test datası
NumberofTestingData = size(Xt,1);
ind=ones(1,NumberofTestingData);
bm=b(:,ind);
H1= w*Xt'+bm; fH1 = feval(ac_func,H1);
Yt = H1'*B
plot(Yt)
xlabel('Örnek Sayısı'),ylabel('Sınıf etiketi')
```