

İleri Yapay Zeka'ya Giriş

Hazırlayan: Doç. Dr. Yunus Egi



Ana Başlıklar

- Yapay Zeka Patlaması
- Derin öğrenmenin tarihi ve gelişimi
- Yapay Zeka ve Alt Katmanları
- Programlama ve Makine Öğrenme
- Öğrenme Çeşitleri
- Derin öğrenme nedir?
- Makine öğrenimi ve derin öğrenme arasındaki farklar
- Derin öğrenme uygulama alanları: Görüntü işleme, metin işleme, oyun oynama

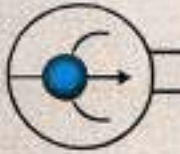
YAPAY ZEKA PATLAMASI!

Derin Öğrenme Devrimi!

Her şey tek bir kaynaktan başlar

YılGelişme / Kilometre Taşı Önemi

1943



McCulloch-Pitts Nöronu

Yapay sinir ağlarının ilk matematiksel modeli.

1943

1958

1998

2012

2012

2014

2017

2022



Backpropagation (Geri Yayılım)

Çok katmanlı ağları eğtmek için temel algoritma.



McCulloch-Pitts Nöronu

Rosenblatt's Perceptron

LeNet-5 (Yann LeCun)

AlexNet & ImageNet Devrimi

AlexNet & ImageNet Devrimi

GANs (Üretken Ağlar)

Transformer Mimarisi

ChatGPT'nin Lansmanı

Yapay sinir ağlarının ilk matematiksel modeli.

İlk gerçek yapay sinir ağı donanımı ve algoritması.

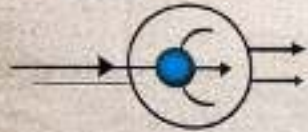
Modern Evrimsel Sinir Ağlarının babası.

Derin öğrenme çağı, GPU devrimi.

Derin öğrenme çağı, GPU devrimi.

"Yapay zeka hayal etmeye başladı".

Yapay zekanın ses, video ve oyun çağı.



Derin öğrenmenin yıllar içindeki gelişimini "görmek"



2015

Goodfellow ve diğerleri.



2018

Karras, Laine, Aila.



2020

MIT Derin Öğrenmeye Giriş

Paul Couvert'in  @ItsPaulAI

Abone

Düşününce bu oldukça çılgınca.

ChatGPT ile ilgili en büyük "vay be" anlarından biri, GPT-3.5'ten GPT-4'e yapılan güncelleme oldu.

Ve şimdi, telefonunuzda yerel olarak çalıştırılabilen, yalnızca 2,6 milyar parametreye sahip bu açık kaynaklı model... orijinal GPT-4'ten DAHA İYİ.

İnanılmaz.

LFM2-2.6B-Exp ile GPT-4'ün Karşılaştırmalı Testi

Kalite testi	LFM2-2.6B-Exp (San AI'dan)	GPT-4 (Orijinal)
IFEval (Kesin Talimatlara Uyuma)	88.13	79.0
Çoklu iF (Karmaşık Talimatlar)	73.05	65.0
IFBench (Bilgiyi Doğru Kullanma)	44.40	24.0
GPOA (Doktora Düzeyinde Akıl Yürütme)	41.31	35.7
MMLU-Pro (Zorlu Uzman Bilgisi)	53.87	63.7
AIIME25 (Ölçümler Seviyesi Matematiği)	22.67	15.0

 E-Tablolar'a aktar

25 Aralık 2023, 16:27 253.9K Görüntüleme

80 178 25 1.1K



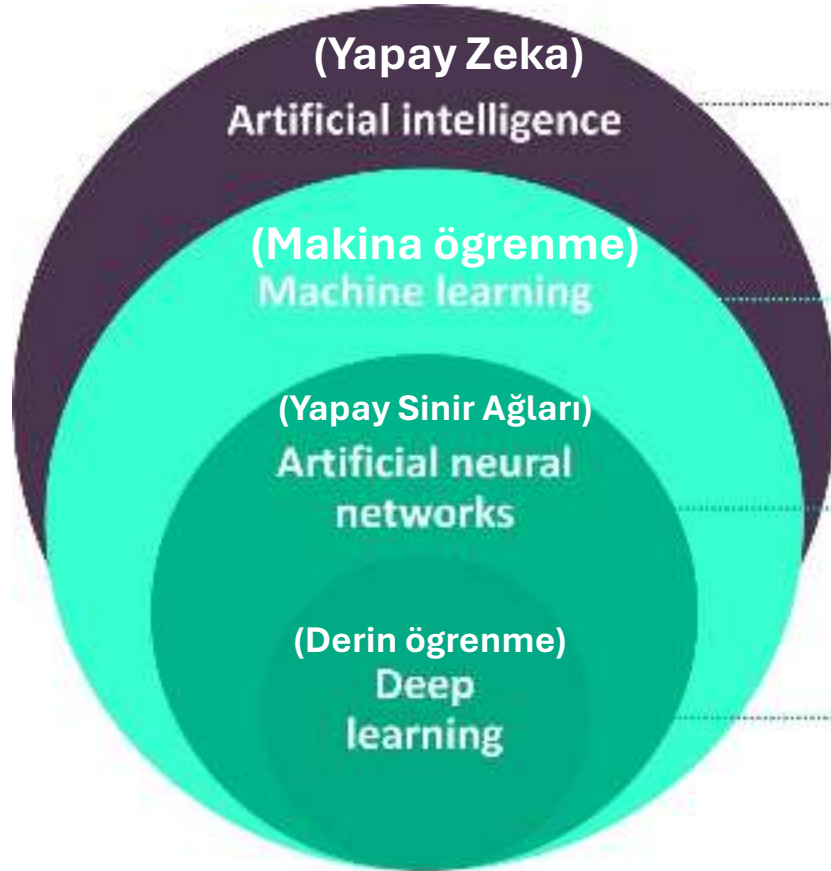


TensorFlow



PyTorch

YAPAY ZEKÂ ve ALT KATMANLARI



Yapay Zeka(AI)

Görevleri yerine getirmek için insan zekasını taklit eder. Genel bir terim.

Makina Öğrenme(ML)

Verilen görevlerin programlama gerektirmeden yerine getirilmesini sağlayan algoritmalar oluşturur. Yapay zekanın bir alt kumesi olarak alabiliriz.

Yapay Sinir Ağları(ANN)

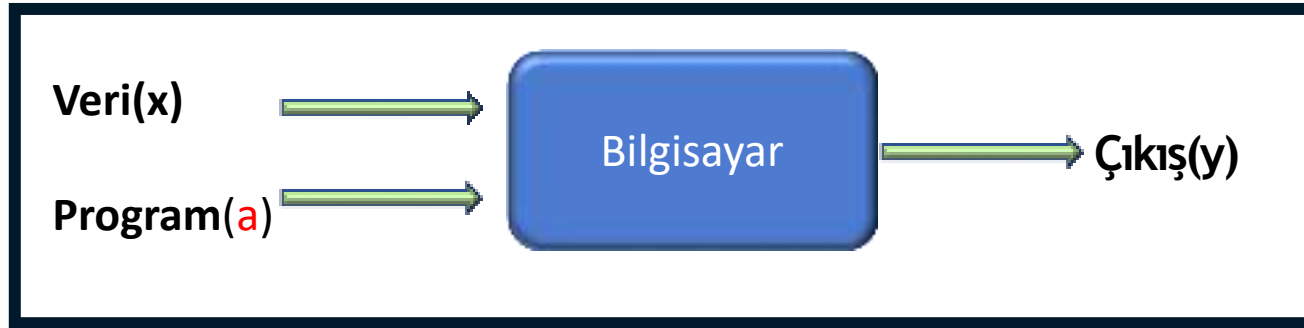
Sinir hücrelerinden ilham alınarak oluşturulan makina öğrenme metodudur.

Derin Öğrenme(DL)

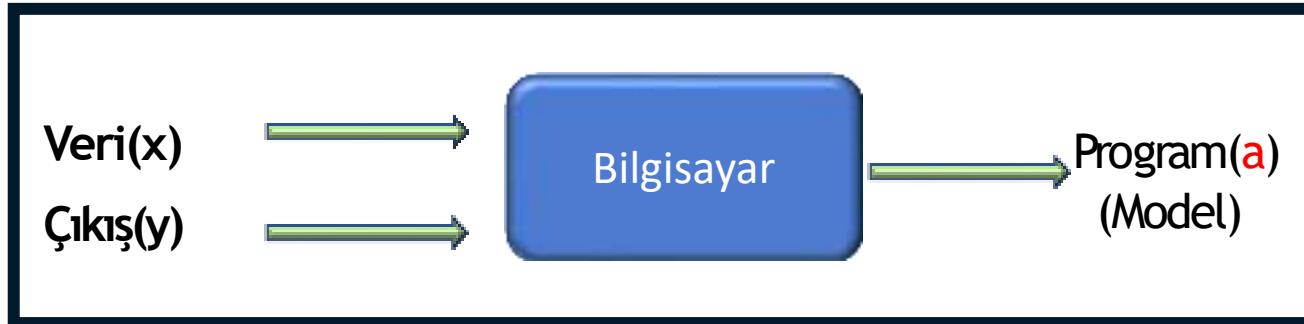
Yine bir makina öğrenme metodu olup yapay sinir ağlarının derin hali olarak adlandırılabilir.



Geleneksel Programlama

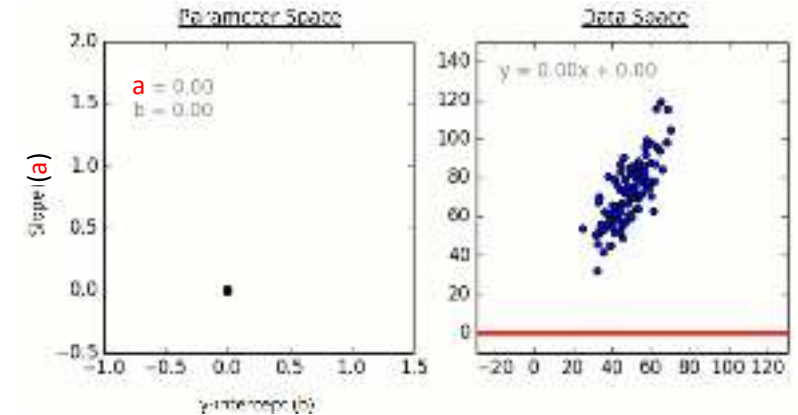
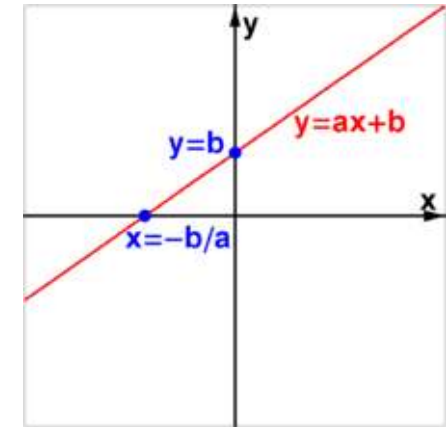


Makina Öğrenmesi



Doğrunun Denklemi

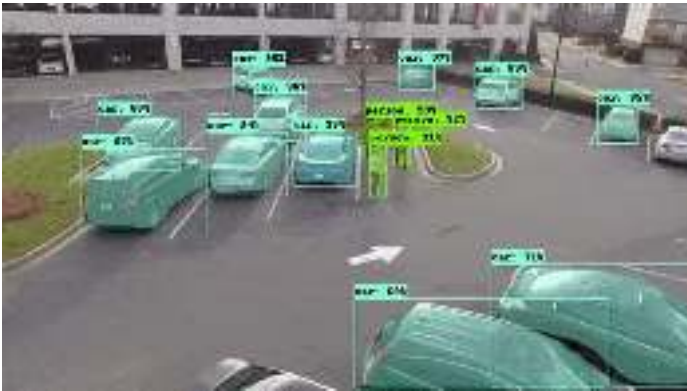
$$y = ax + b$$



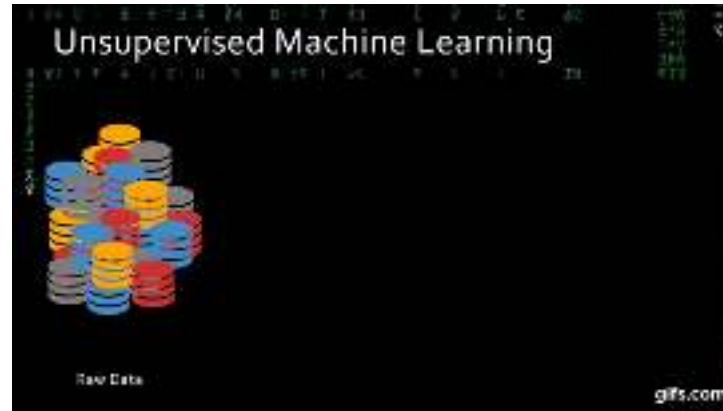
Öğrenme Çeşitleri

Öğrenme(Learning)

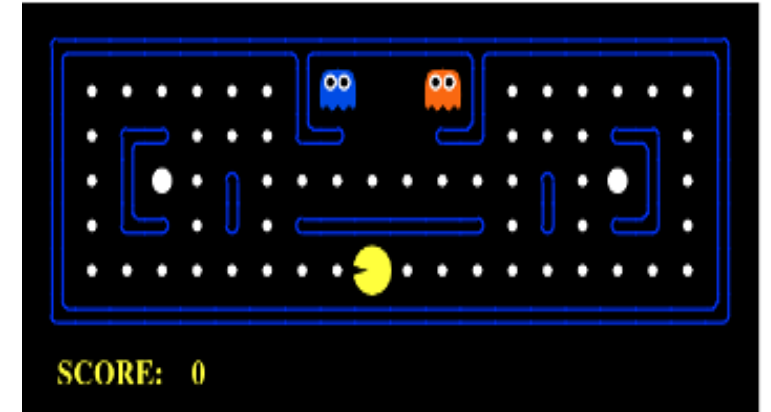
Gözetimli öğrenme
(Supervised Learning)



Gözetimsiz öğrenme
(unsupervised Learning)

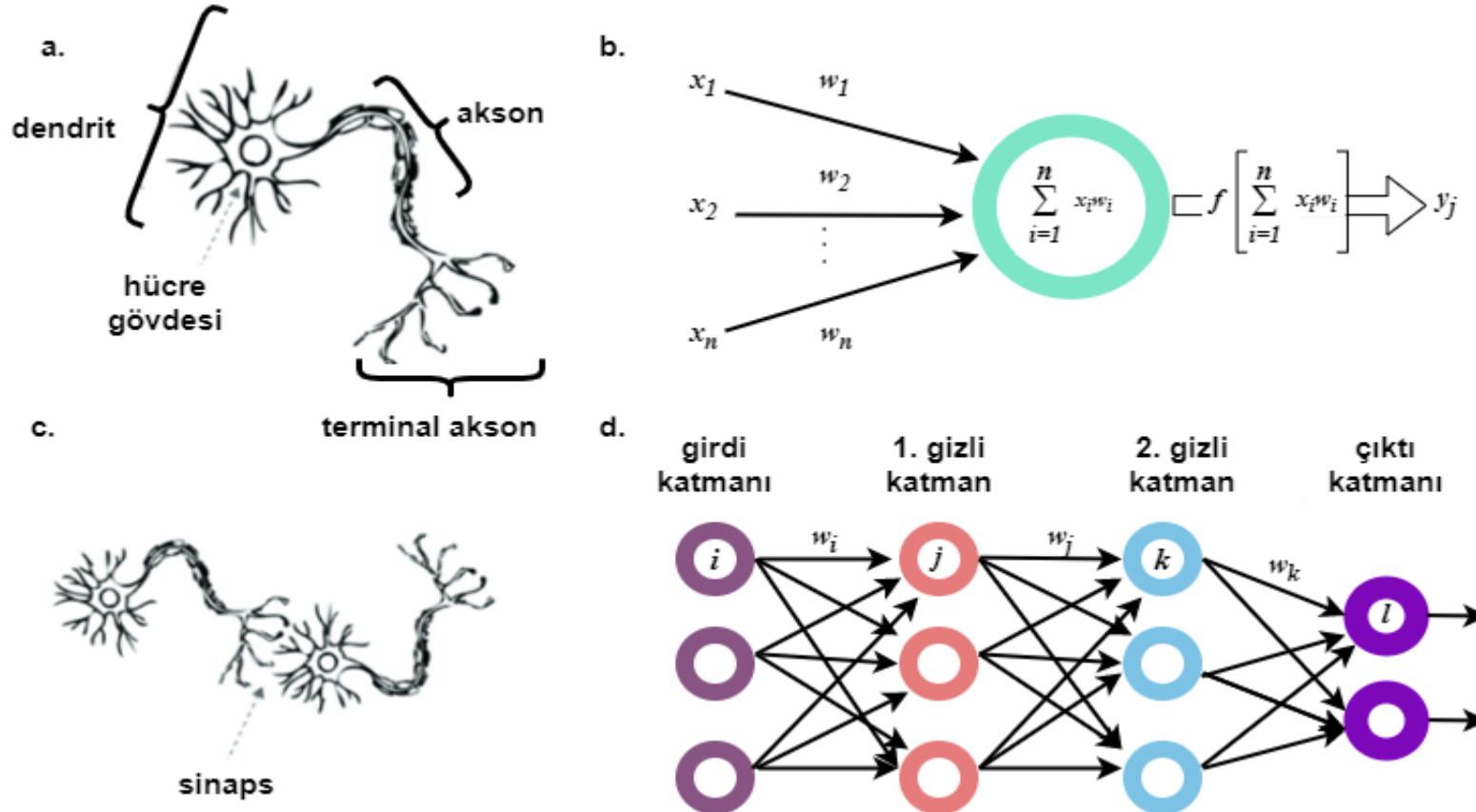


Pekiştirmeli öğrenme
(Reinforcement Learning)

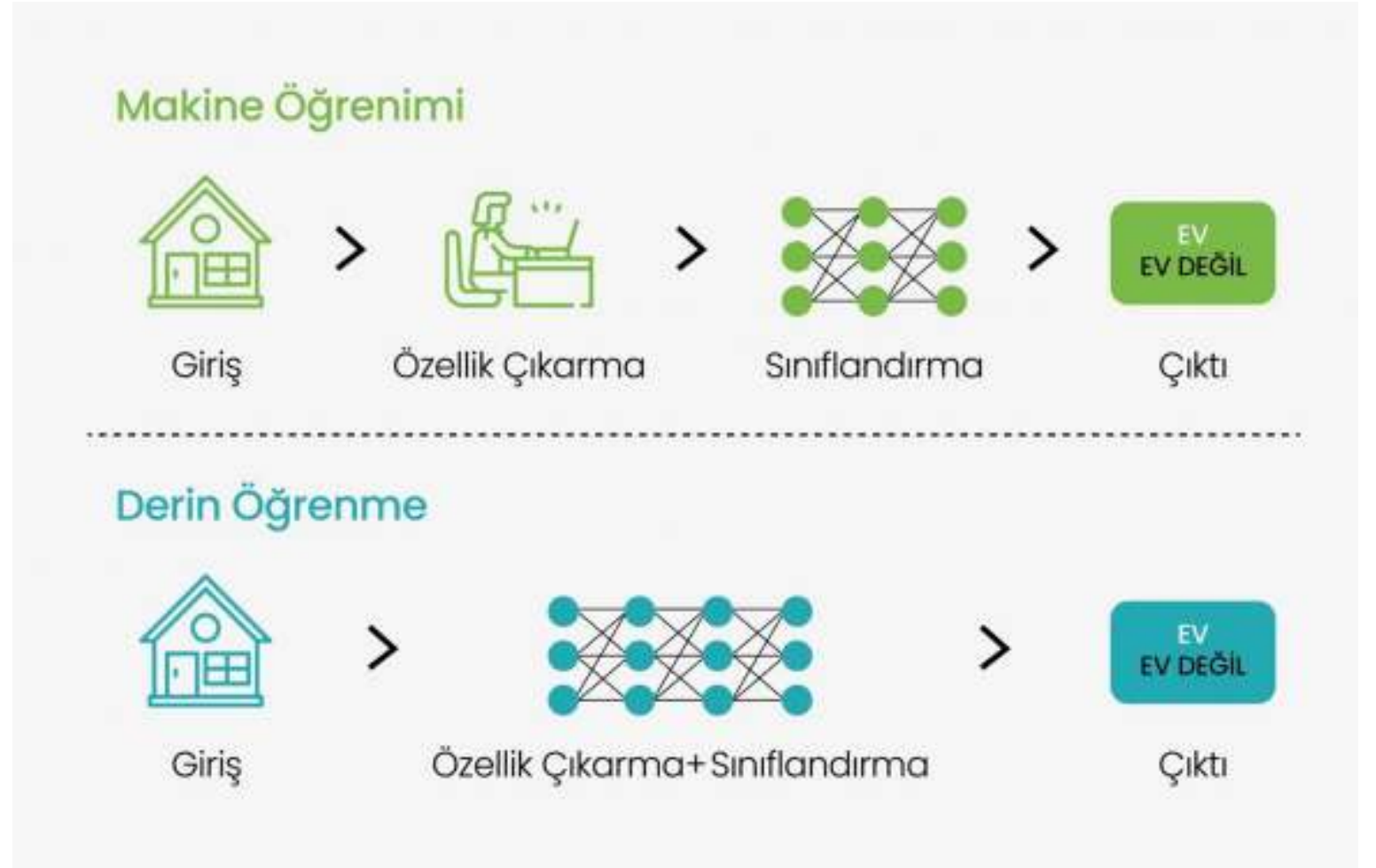


Derin öğrenme nedir?

- Derin öğrenme, büyük veri setlerinden öğrenen ve çok katmanlı sinir ağları (derin ağlar) kullanan bir makine öğrenimi alt kümesidir.

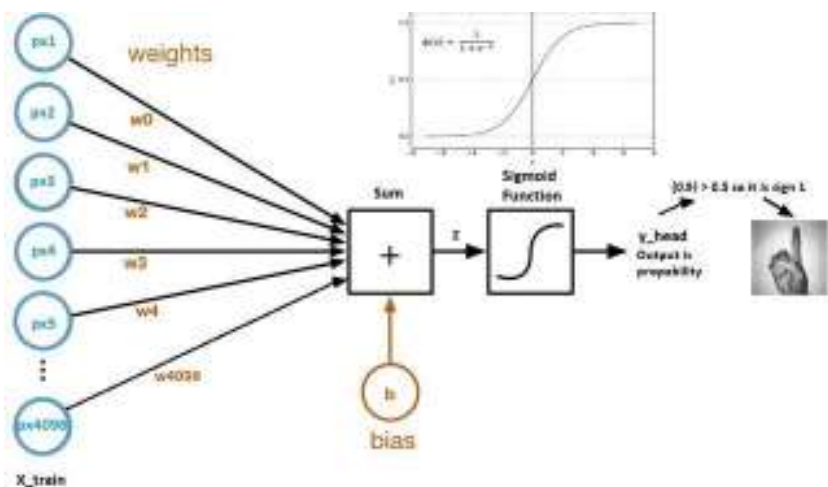


Makine öğrenimi ve derin öğrenme arasındaki farklar

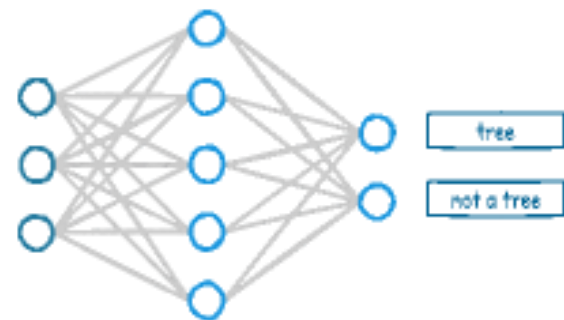


Kaynak: <https://www.turhost.com/blog/deep-learning-nedir/>

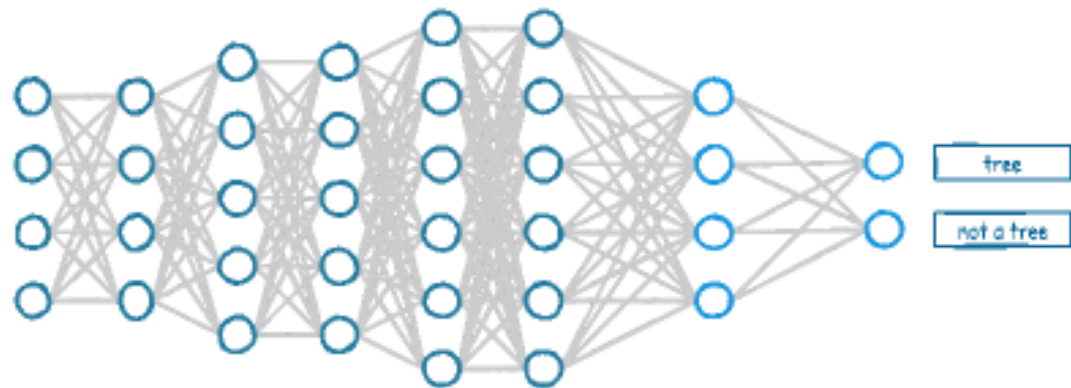
Makine öğrenimi ve derin öğrenme arasındaki farklar



Machine Learning



Deep Learning



Neden Şimdi?

Sinir ağıları onlarca yıl öncesine dayanıyor,
Peki neden şimdi daha çok kullanılmaya başlandı?



1. Büyük Veri

- Daha Büyük Veri Kümeleri
- Daha Kolay Toplama ve Depolama

IMAGENET



2. Donanım

- Grafik İşlem Birimleri (GPU'lar)
- Büyük Ölçekli Paralleleştirilebilir



3. Yazılım

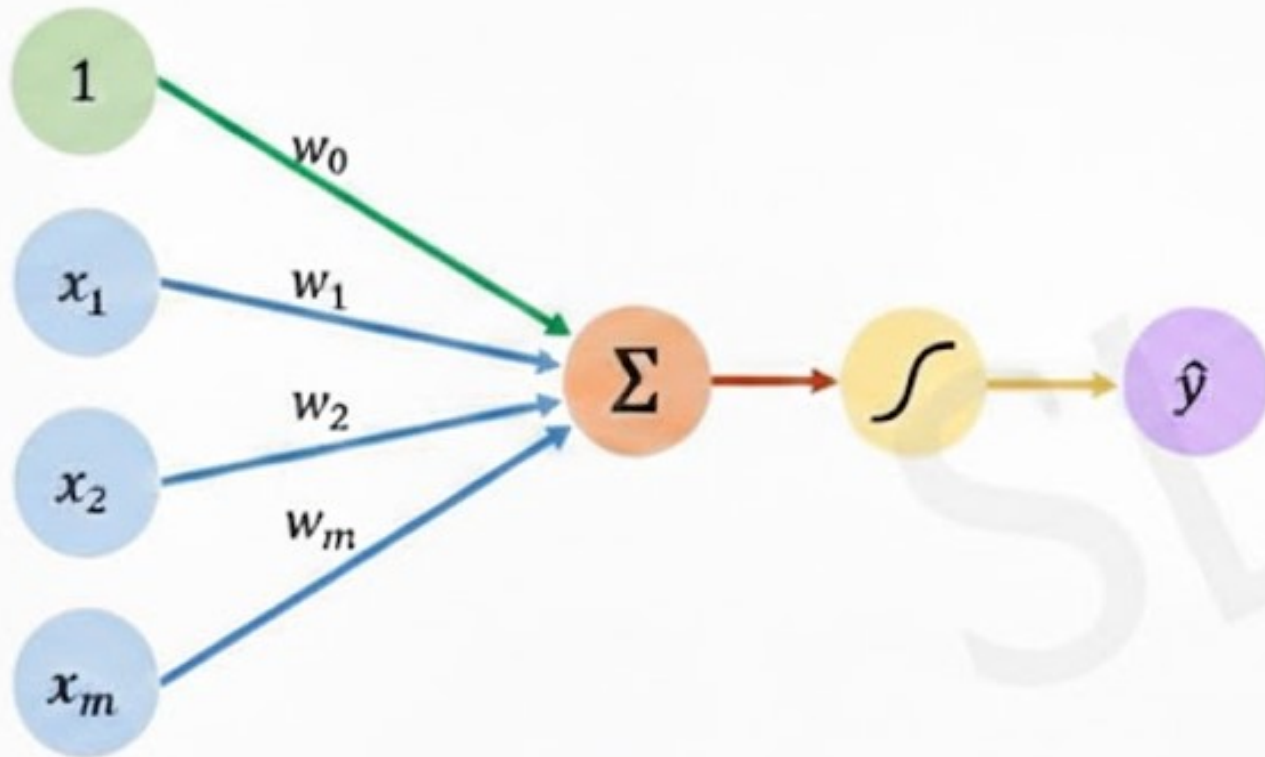
- Geliştirilmiş Teknikler
- Yeni Modeller
- Araç kutuları



The perceptron

Derin öğrenmenin yapısal yapı taşı

Perceptron: İleri Yayılım



Girişler

Ağırlıklar

Toplam

Doğrusal Olmama

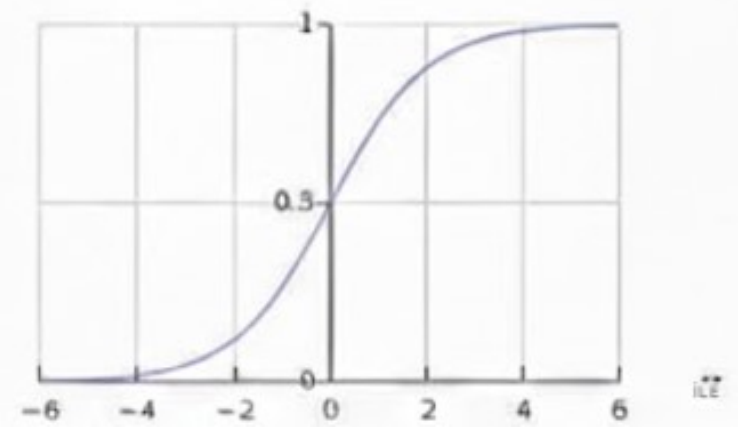
Çıkış

Etkinleştirme İşlevleri

$$\hat{y} = g(w_0 + \mathbf{X}^T \mathbf{W})$$

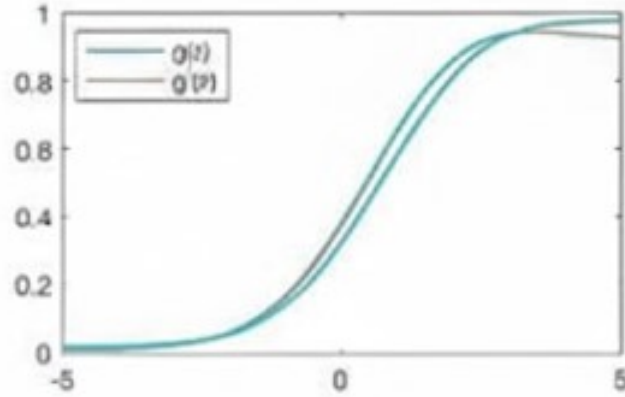
- Örnek: sigmoid fonksiyonu

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Ortak Etkinleştirme İşlevleri

Sigmoid Fonksiyonu



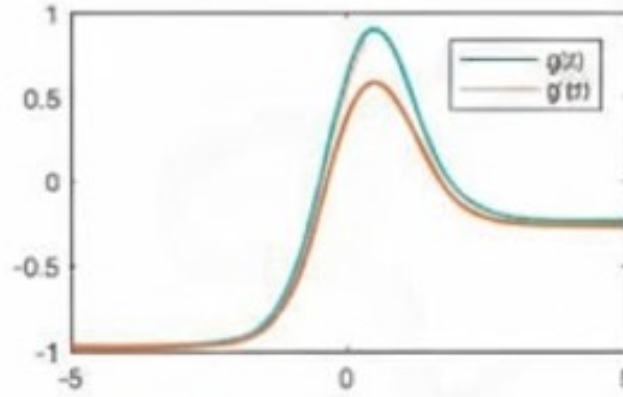
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

 `tf.math.sigmoid(z)`

 `torch.sigmoid(z)`

Hiperbolik Teğet



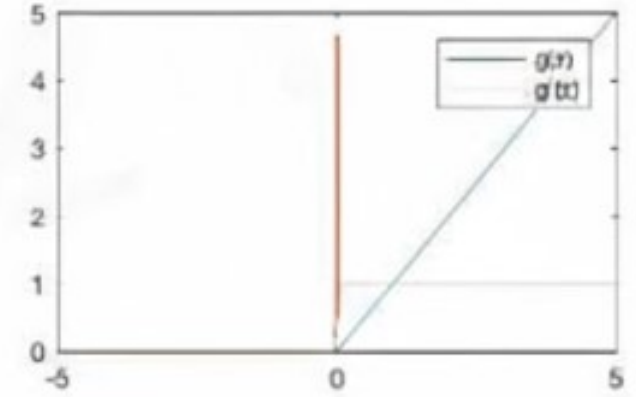
$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

 `tf.math.tanh(z)`

 `torch.tanh(z)`

Doğrultulmuş Doğrusal Birim (ReLU)



$$g(z) = \max(0, z)$$

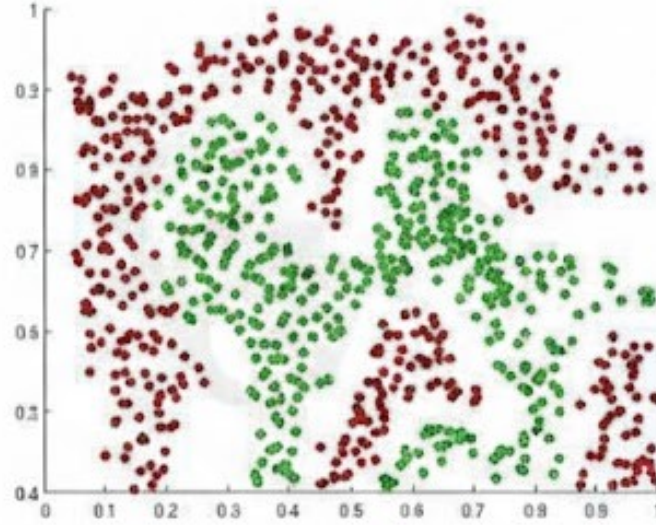
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

 `tf.nn.relu(z)`

 `torch.nn.ReLU(z)`

Aktivasyon Fonksiyonlarının Önemi

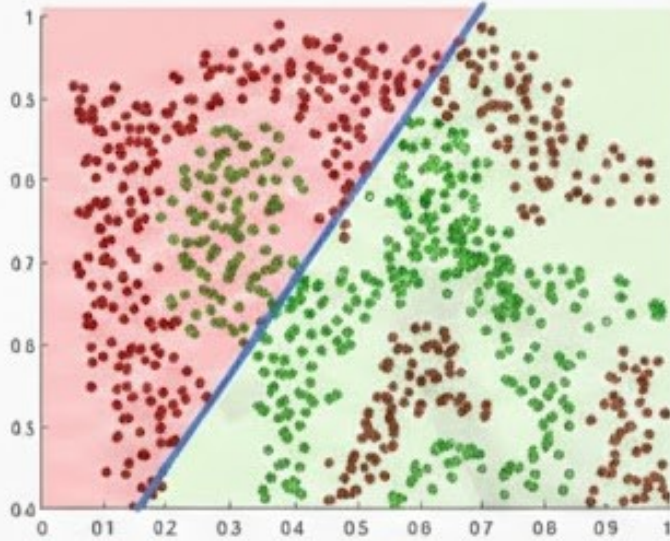
Aktivasyon fonksiyonlarının amacı, ağı doğrusal olmayan özellikler kazandırmaktır.



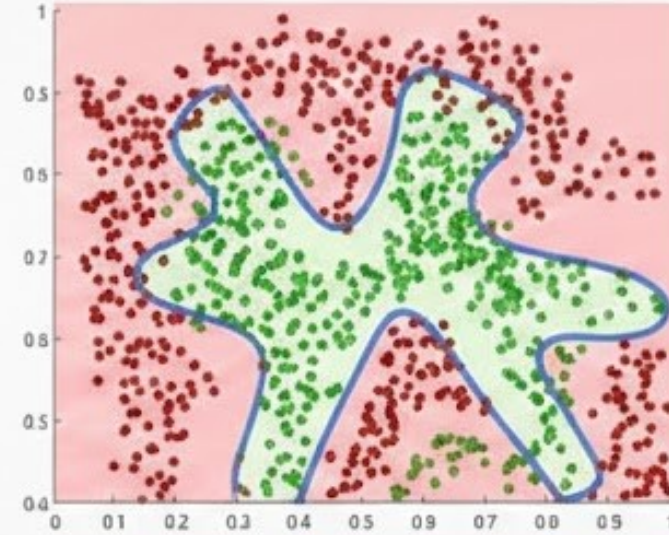
Peki ya yeşil ve kırmızı noktaları birbirinden ayırt edebilen
bir sinir ağı oluşturmak isteseydik?

Aktivasyon Fonksiyonlarının

Aktivasyon fonksiyonlarının amacı, ağ doğrusal olmayan özellikler kantır.

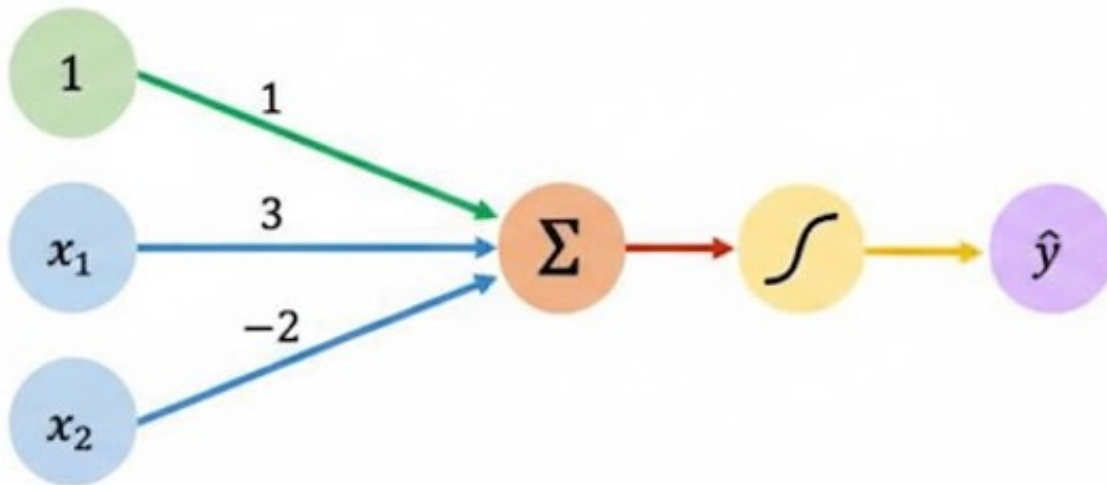


Doğrusal aktivasyon fonksiyonları, ağ boyutundan bağımsız olarak doğrusal kararlar üretir.



Doğrusal olmayan özellikler, keyfi olarak karmaşık fonksiyonları yaklaşık olarak hesaplamamıza olanak tanır.

Perceptron: Örnek

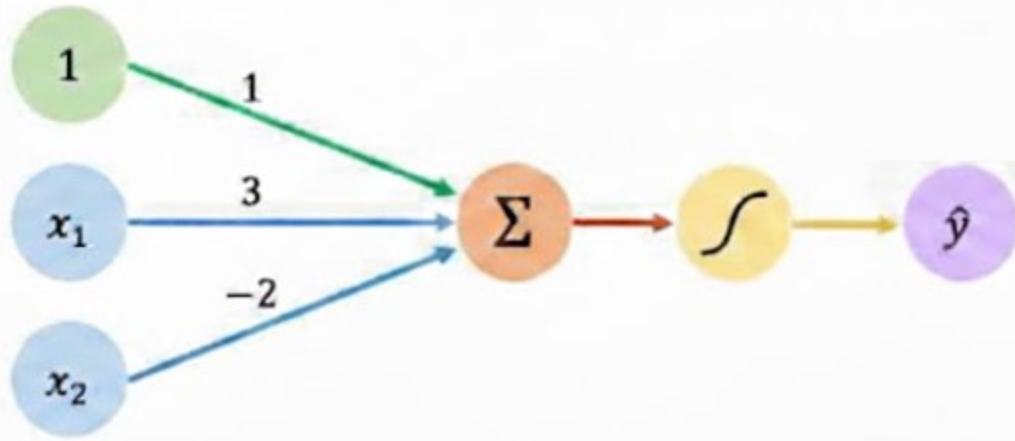


Elimizde şunlar var: $w_0 = 1$ $\mathbf{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

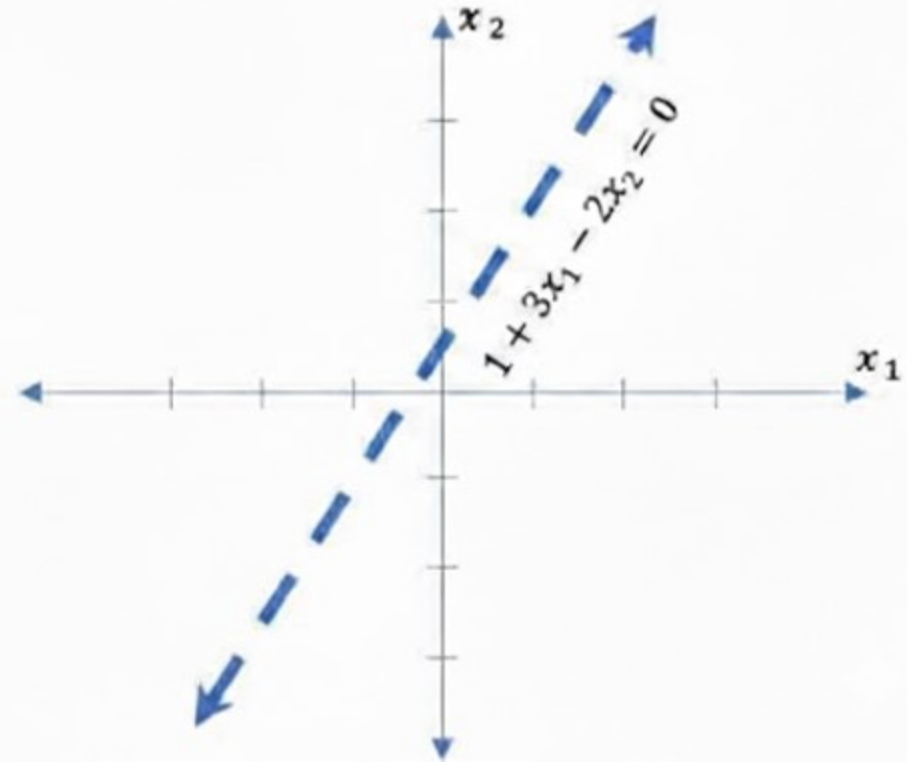
$$\begin{aligned}\hat{y} &= g(w_0 + \mathbf{X}^T \mathbf{W}) \\ &= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right) \\ \hat{y} &= g(1 + 3x_1 - 2x_2)\end{aligned}$$

Bu sadece 2 boyutlu bir çizgi!

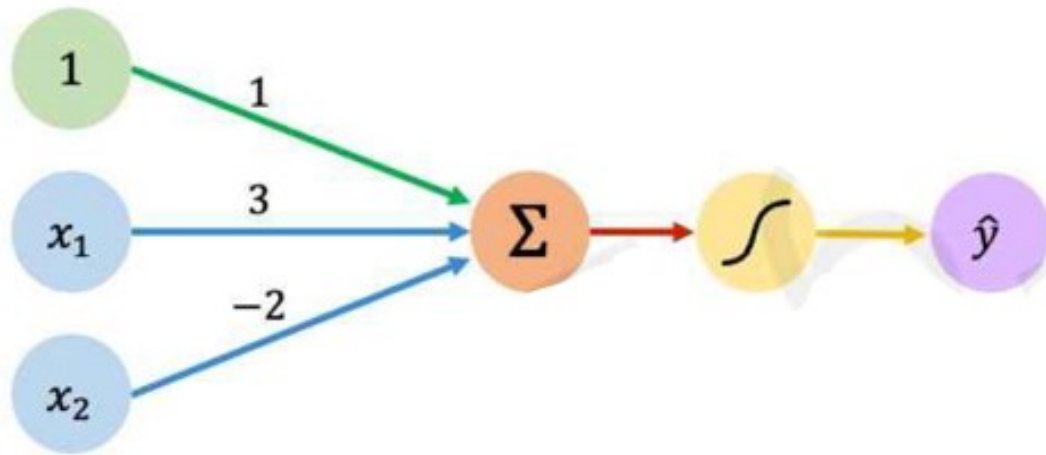
Perceptron: Örnek



$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

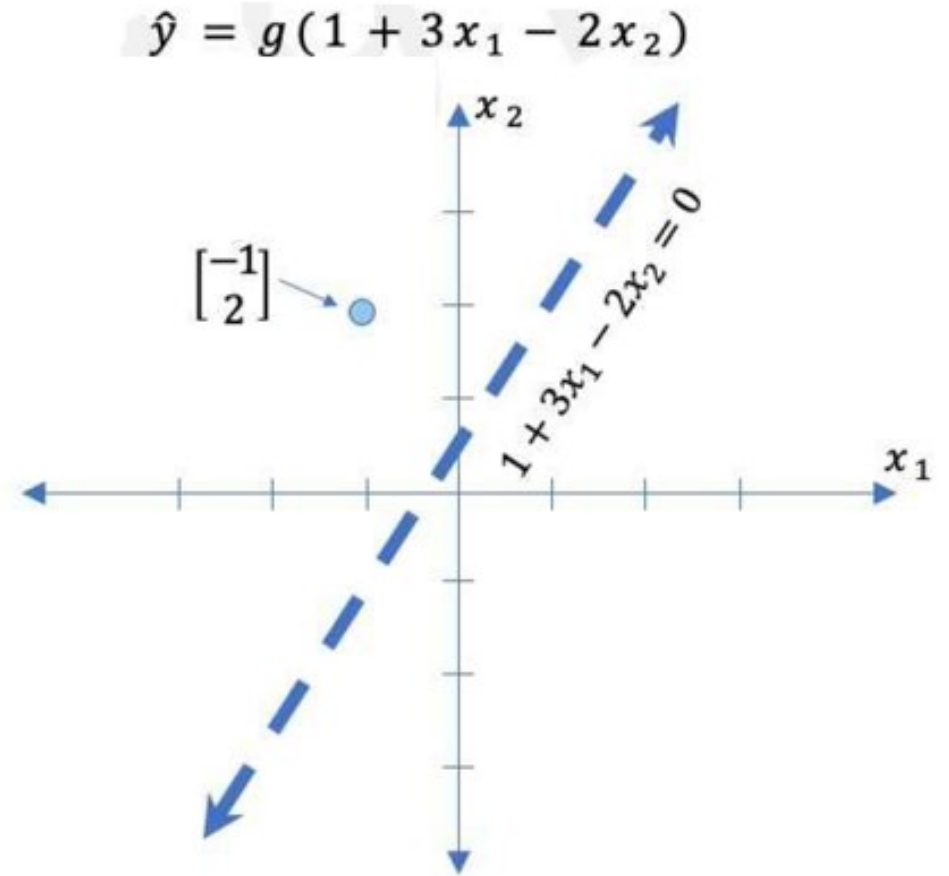


Perceptron: Örnek

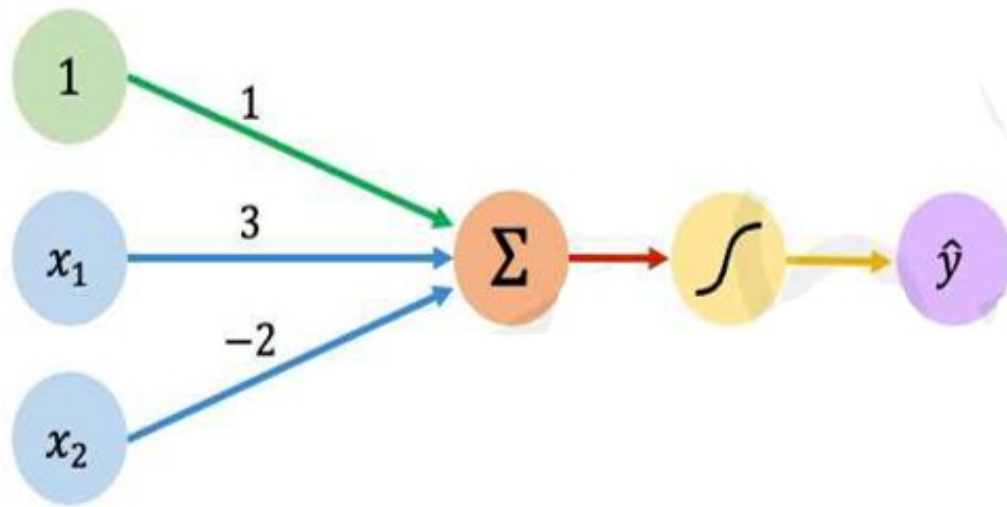


Assume we have input: $\mathbf{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

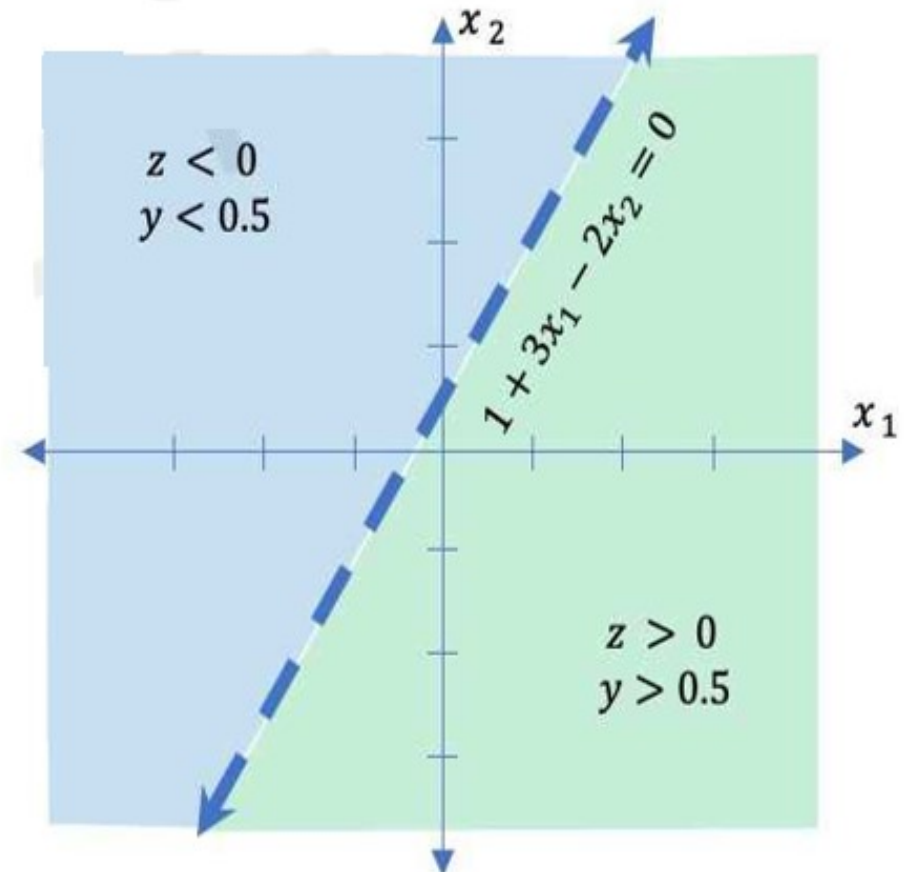
$$\begin{aligned}\hat{y} &= g(1 + (3 * -1) - (2 * 2)) \\ &= g(-6) \approx 0.002\end{aligned}$$

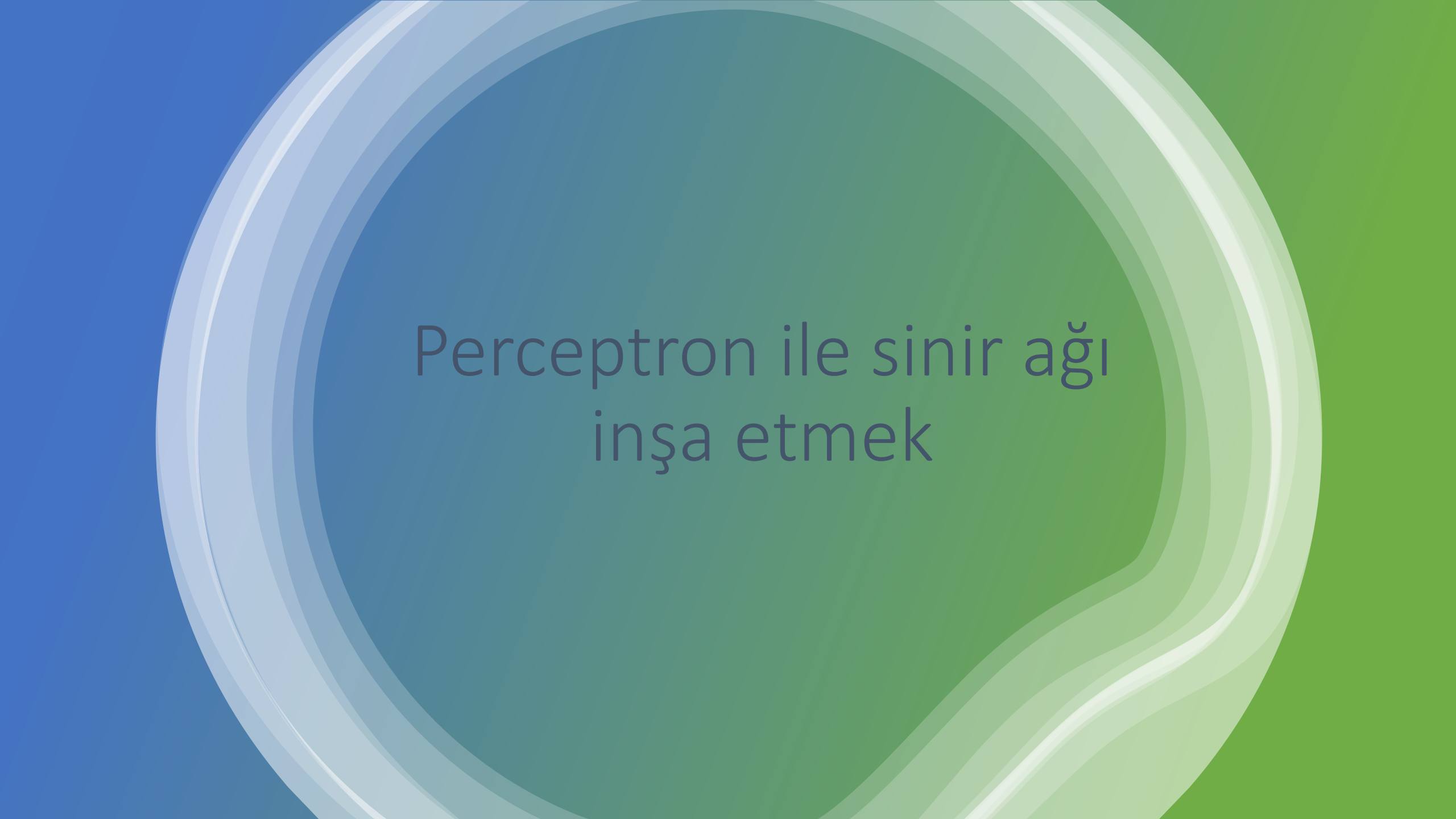


Perceptron: Örnek



$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

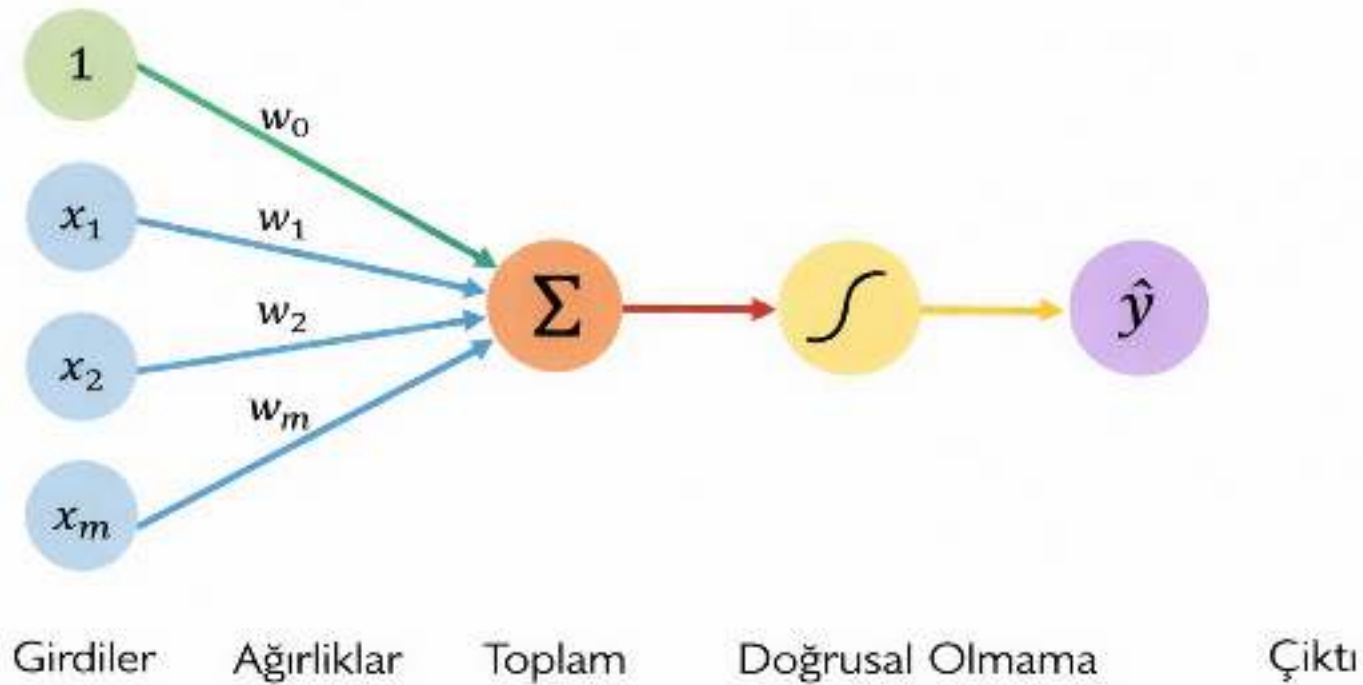




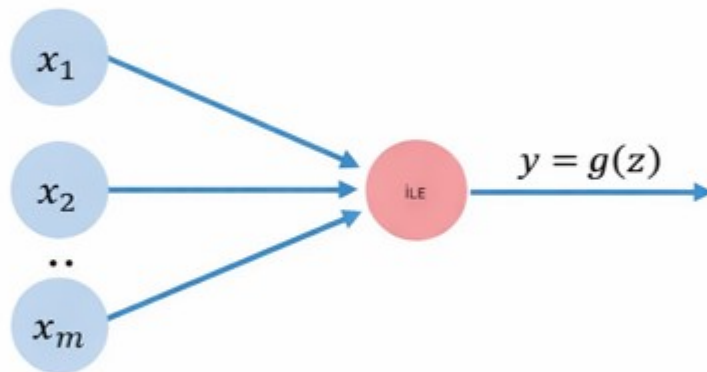
Perceptron ile sinir ağı
inşa etmek

Perceptron: Basitleştirilmiş

$$\hat{y} = g(w_0 + X^T W)$$



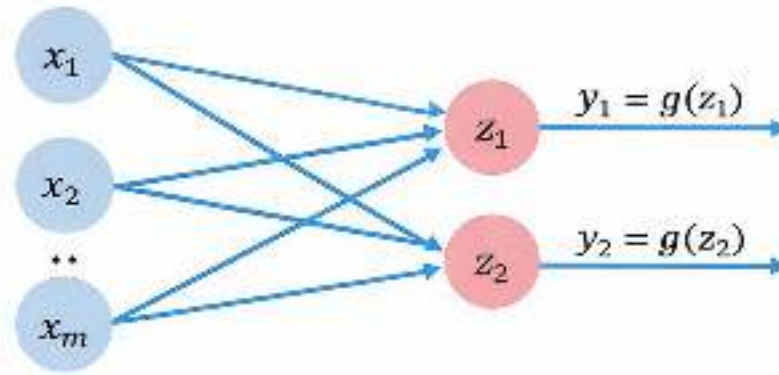
Perceptron: Basitleştirilmiş



$$z = w_0 + \sum_{j=1}^m x_j w_j$$

Çok Çıkışlı Perceptron

Tüm girişler tüm çıkışlara yoğun bir şekilde bağlı olduğundan, bu katmanlara Yoğun Katmanlar denir.



$$z_i = w_{0,i} + \sum_{j=1}^m x_j w_{j,i}$$



```
class MyDenseLayer(tf.keras.layers.Layer):
    def __init__(self, input_dim, output_dim):
        super(MyDenseLayer, self).__init__()

        # Ağırlıkları ve sapmayı (bias) ilklendirme
        self.W = self.add_weight([input_dim, output_dim])
        self.b = self.add_weight([1, output_dim])

    def call(self, inputs):
        # İleri yayılım (forward propagation)
        z = tf.matmul(inputs, self.W) + self.b

        # Doğrusal olmayan aktivasyon (sigmoid)
        output = tf.math.sigmoid(z)
        return output
```

```
class MyDenseLayer(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(MyDenseLayer, self).__init__()

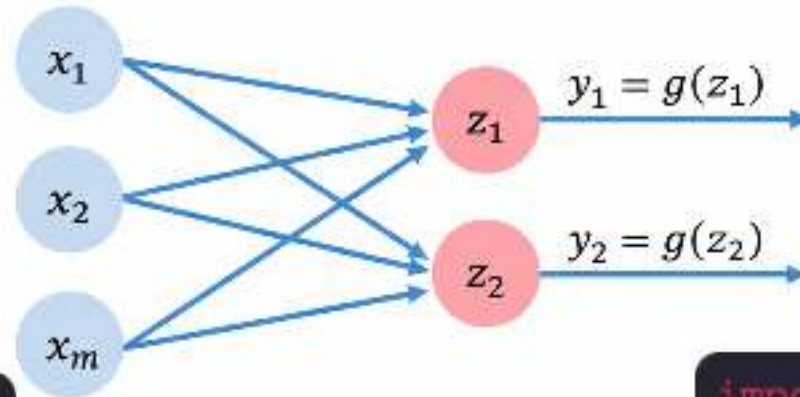
        # Ağırlıkları ve sapmayı (bias) ilklendirme
        self.W = nn.Parameter(torch.randn(input_dim, output_dim, requires_grad=True))
        self.b = nn.Parameter(torch.randn(1, output_dim, requires_grad=True))

    def forward(self, inputs):
        # İleri yayılım (forward propagation)
        z = torch.matmul(inputs, self.W) + self.b

        # Doğrusal olmayan aktivasyon (sigmoid)
        output = torch.sigmoid(z)
        return output
```

Çok Çıkışlı Perceptron

Tüm girişler tüm çıkışlara yoğun bir şekilde bağlı olduğundan, bu katmanlara Yoğun Katmanlar denir.

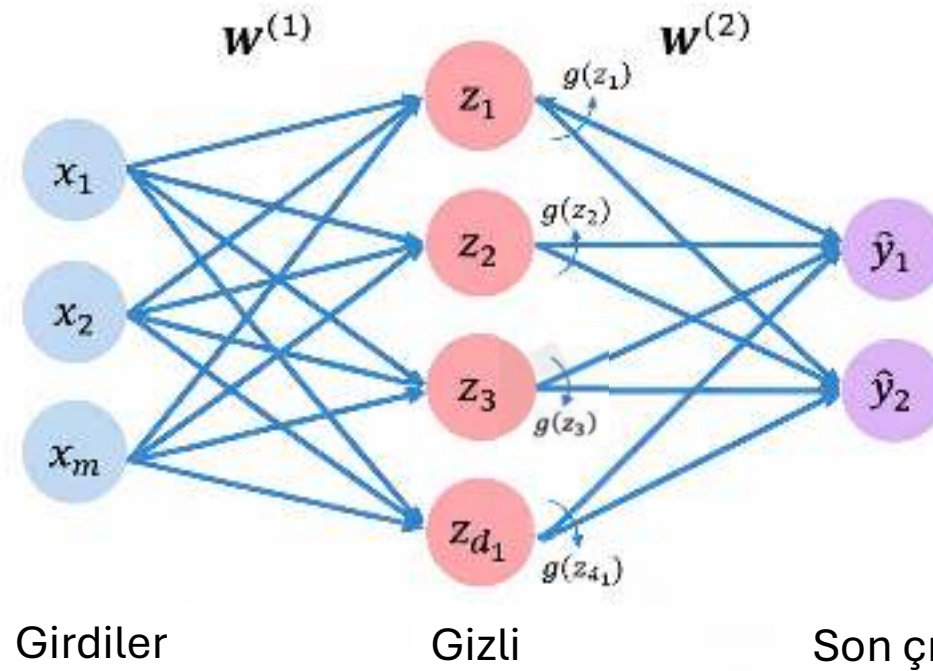


```
import tensorflow as tf  
  
layer = tf.keras.layers.Dense(  
    units=2)
```

$$z_i = w_{0,i} + \sum_{j=1}^m x_j w_{j,i}$$

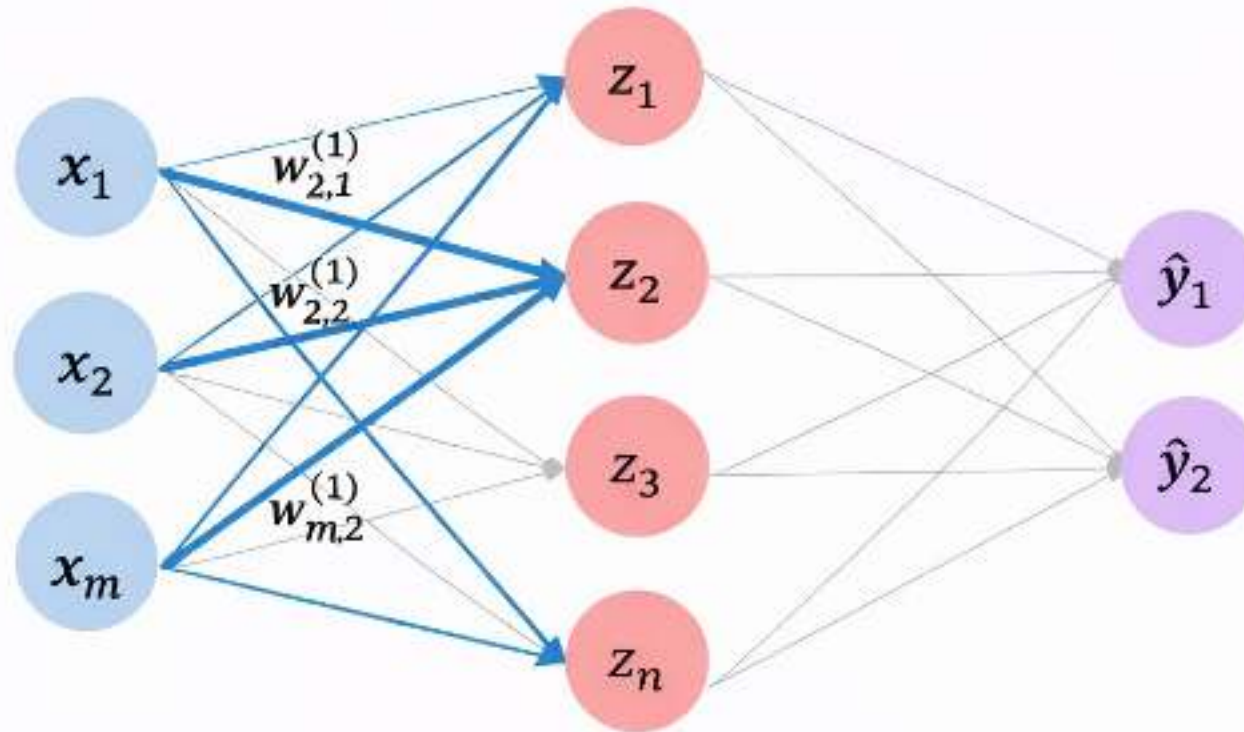
```
import torch.nn as nn  
  
layer = nn.Linear(in_features=m,  
    out_features=2)
```

Tek katmanlı Sinir Ağı



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^m x_j w_{j,i}^{(1)} \quad \hat{y}_l = g \left(w_{0,l}^{(2)} + \sum_{j=1}^{d_1} g(z_j) w_{j,l}^{(2)} \right)$$

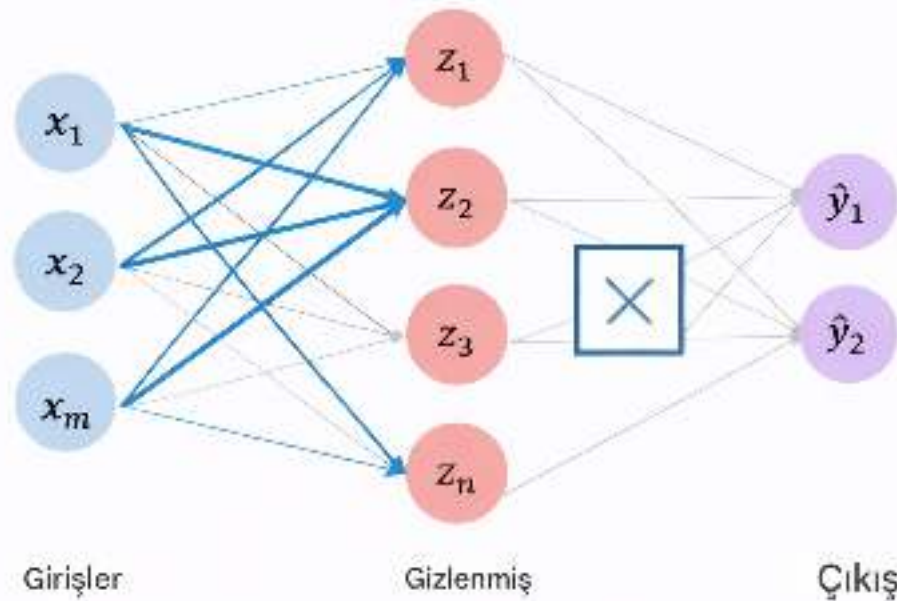
Tek katmanlı Sinir Ağı



$$\begin{aligned} z_2 &= w_{0,2}^{(1)} + \sum_{j=1}^m x_j w_{j,2}^{(1)} \\ &= w_{0,2}^{(1)} + x_1 w_{1,2}^{(1)} + x_2 w_{2,2}^{(1)} + x_m w_{m,2}^{(1)} \end{aligned}$$

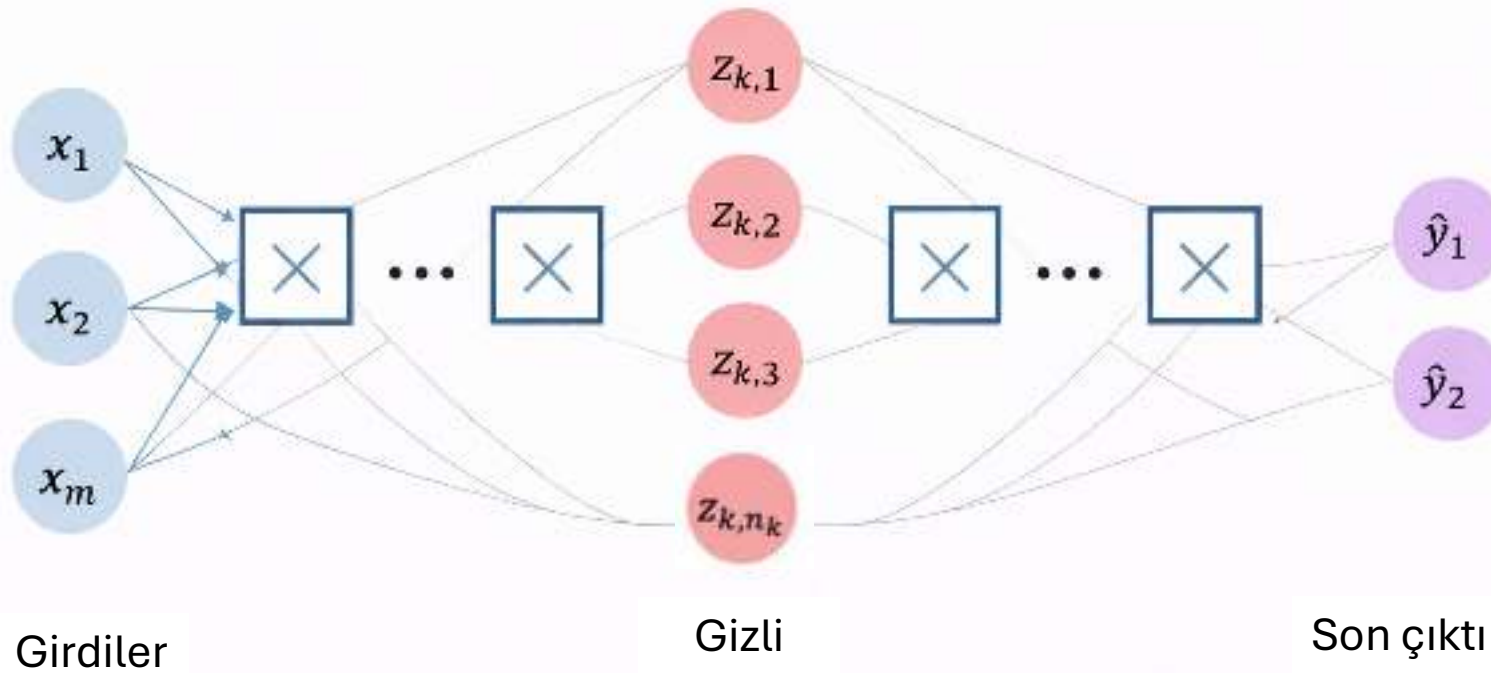
Çok Çıkışlı Algılayıcı

```
import tensorflow as tf  
  
model = tf.keras.Sequential([  
    tf.keras.layers.Dense(n),  
    tf.keras.layers.Dense(2)  
)
```



```
from torch import nn  
  
model = nn.Sequential(  
    nn.Linear(m, n),  
    nn.ReLU(),  
    nn.Linear(n, 2)  
)
```

Derin Sinir Ağı



```
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Dense(n1),
    tf.keras.layers.Dense(n2),
    ...
    tf.keras.layers.Dense(2)
])
```

```
from torch import nn

model = nn.Sequential(
    nn.Linear(m, n1),
    nn.ReLU(),
    ...
    nn.ReLU(),
    nn.Linear(nK, 2)
)
```

$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)}$$

Amazon Go Uygulaması





clideo.com

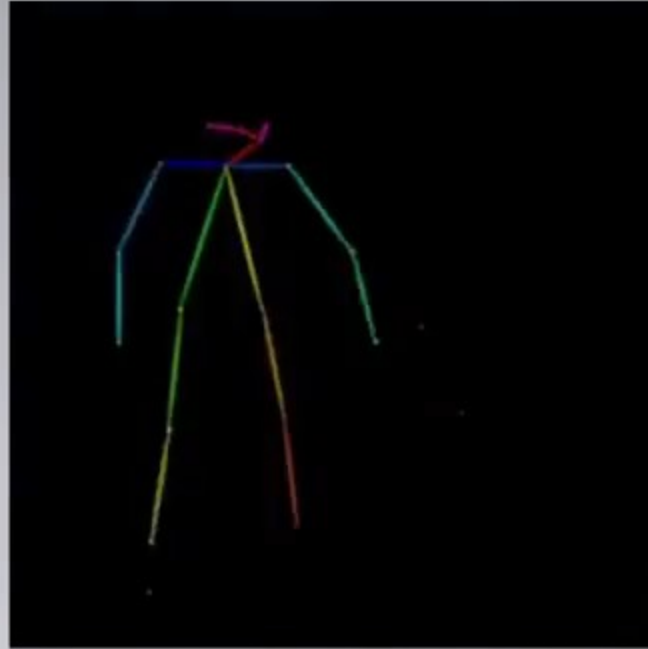
A close-up photograph of a person's hand holding a black stylus, writing on a tablet. The hand is positioned on the right side of the frame, with the stylus tip touching the screen. The tablet screen shows a digital writing application with a toolbar at the top containing icons for undo, redo, eraser, and other tools. The text being written is in a cursive script. The background is a wooden surface.

Here's to the crazy ones
the mavericks. The real geniuses
things differently. They're not
cared for the status quo. You can
them, glorify in victory. They
do it against all odds. Because they
the human race forward. And while
the crazy ones, we see genius. Because
crazy enough to think they can do
what we do.

Yazı Tanımlama

Yapay Zeka ile Pozisyon Tahmini

THE LOW CHAI



İnsansı Robot(Pekistirmeli Öğrenme)



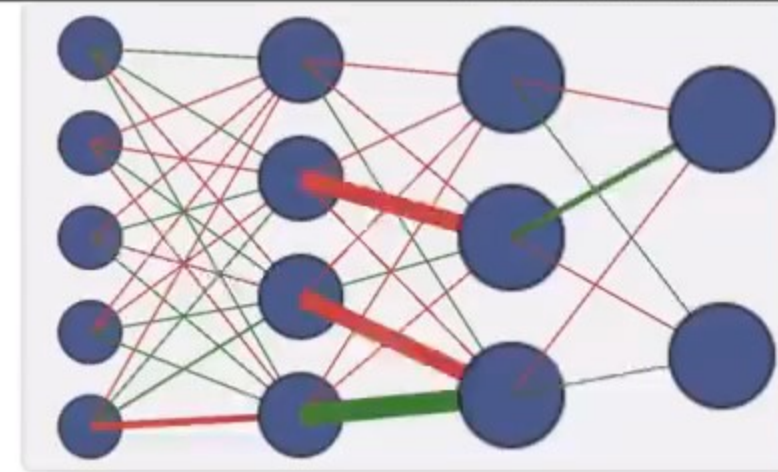
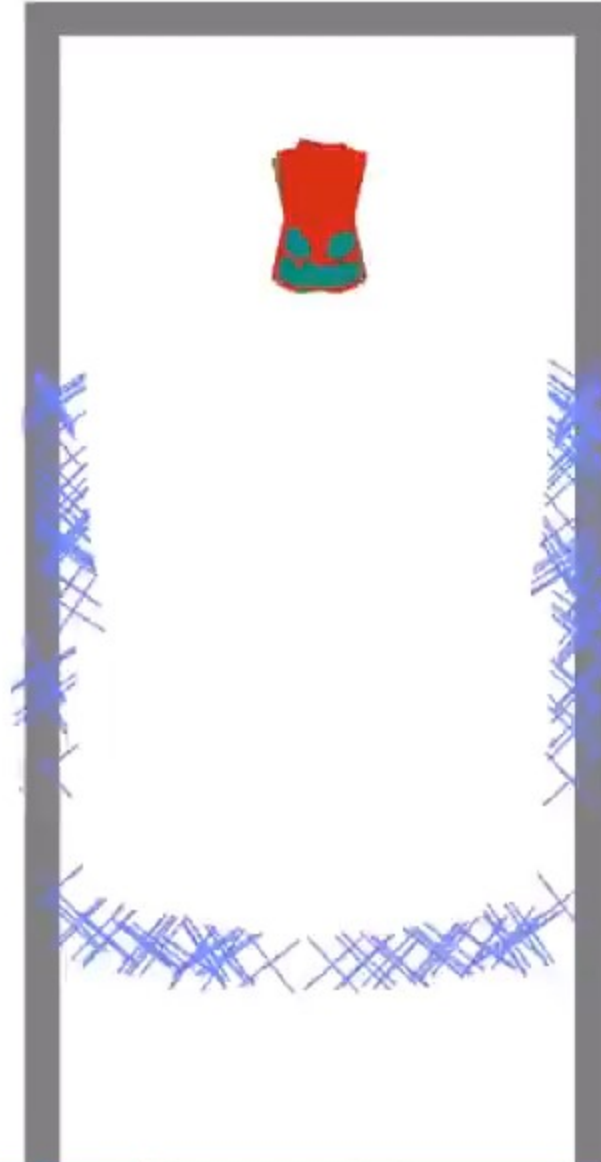
Boston Dynamics

Turn: -0.7932

Engine: 0.99999

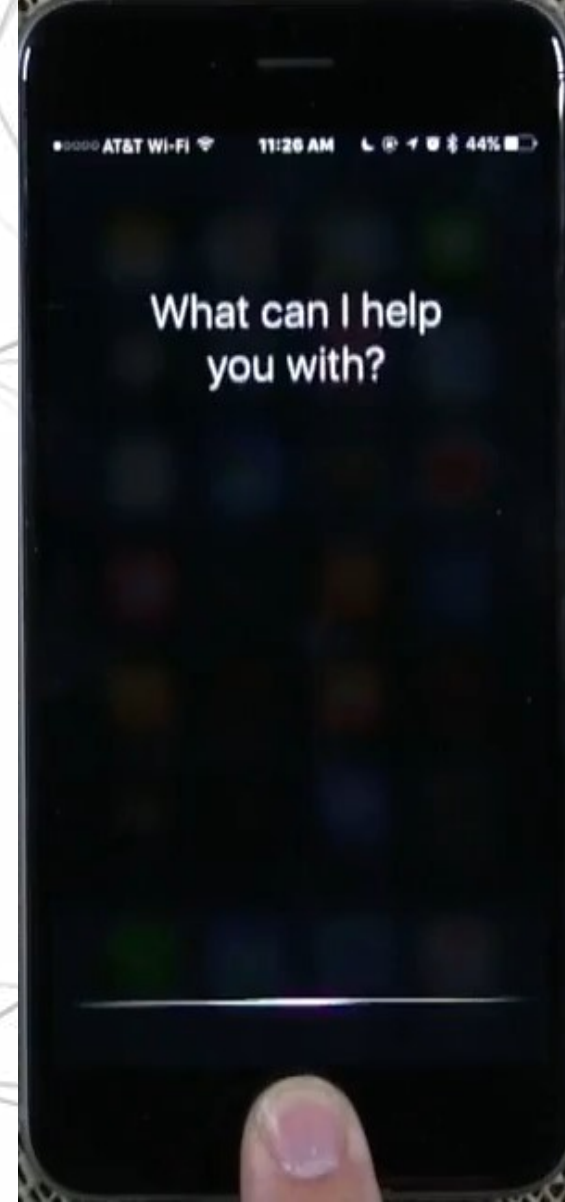
Fitness: 9.78920

Pekiştirmeli Öğrenme



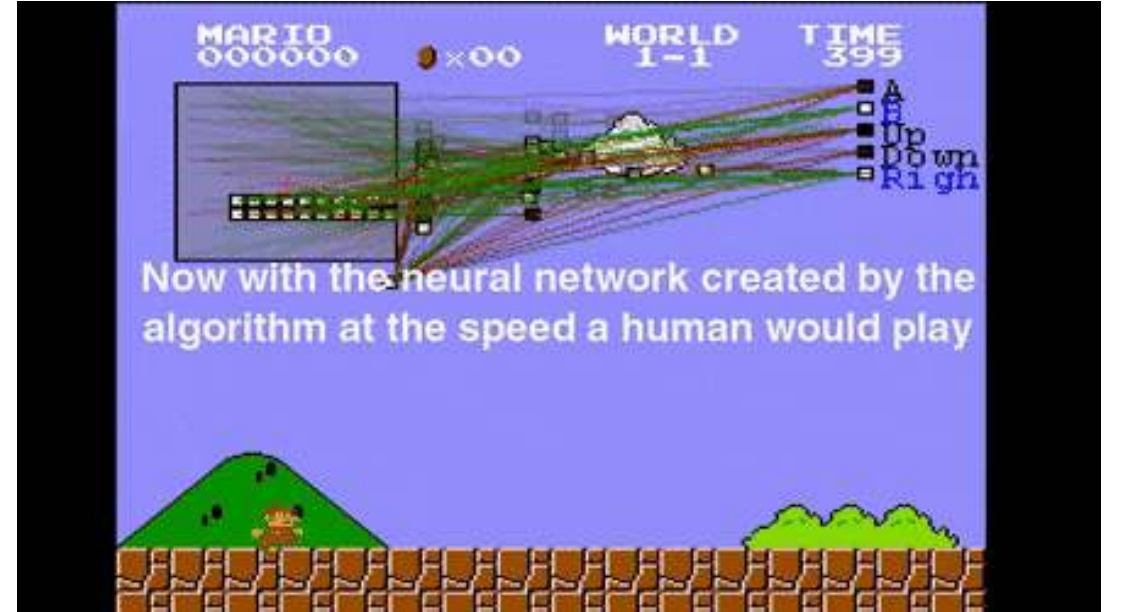
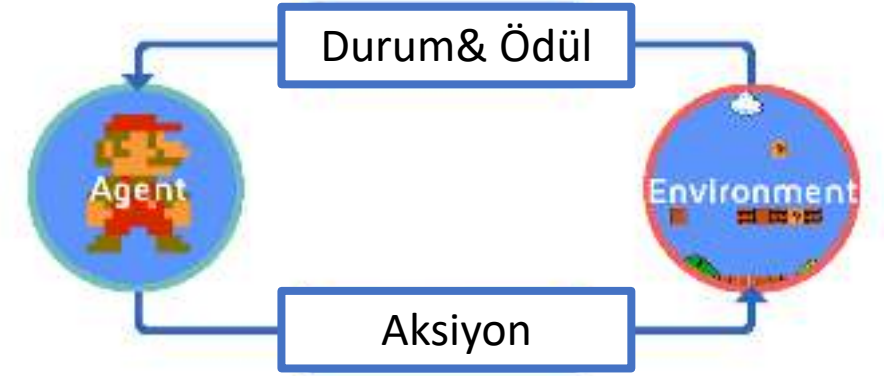
Generation: 1

Siri ile Ses Tanımlaması

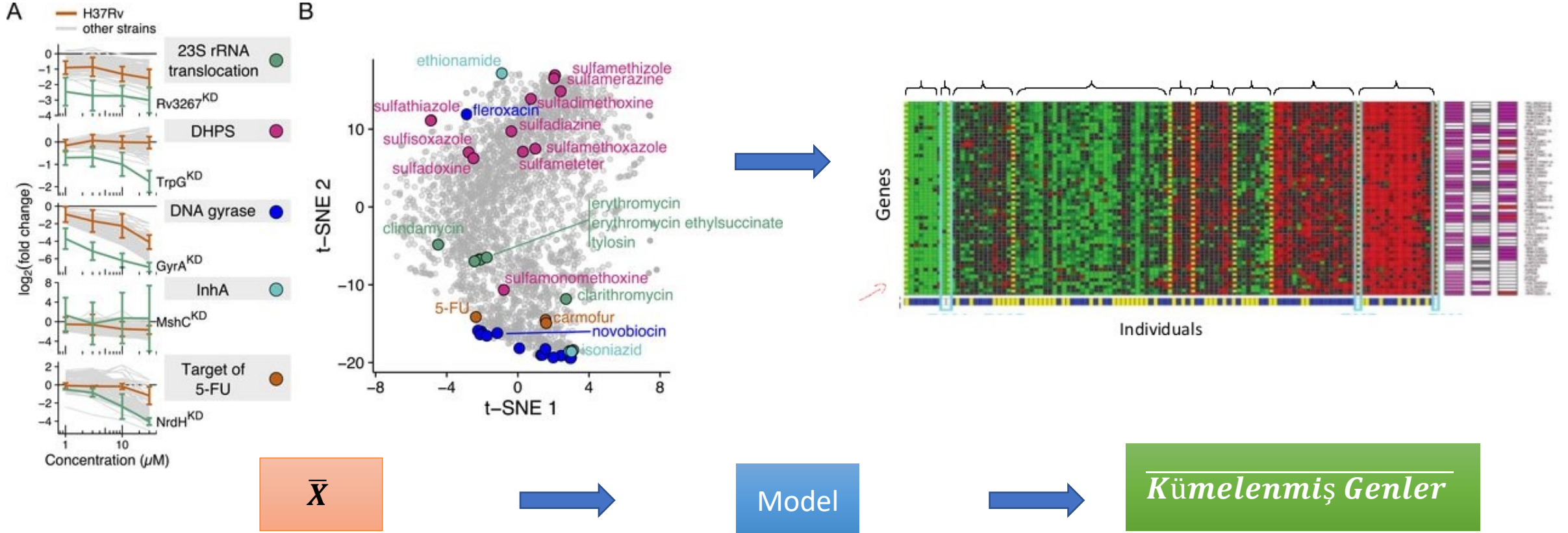


Pekiştirmeli Öğrenme(Reinforcement Learning)

Pekiştirmeli öğrenme optimal kararların yapıldığı bir bilim olarak da adlandırılır. Canlılar tarafından ödül odaklı davranışların formülize edilmesine yardımcı olur.

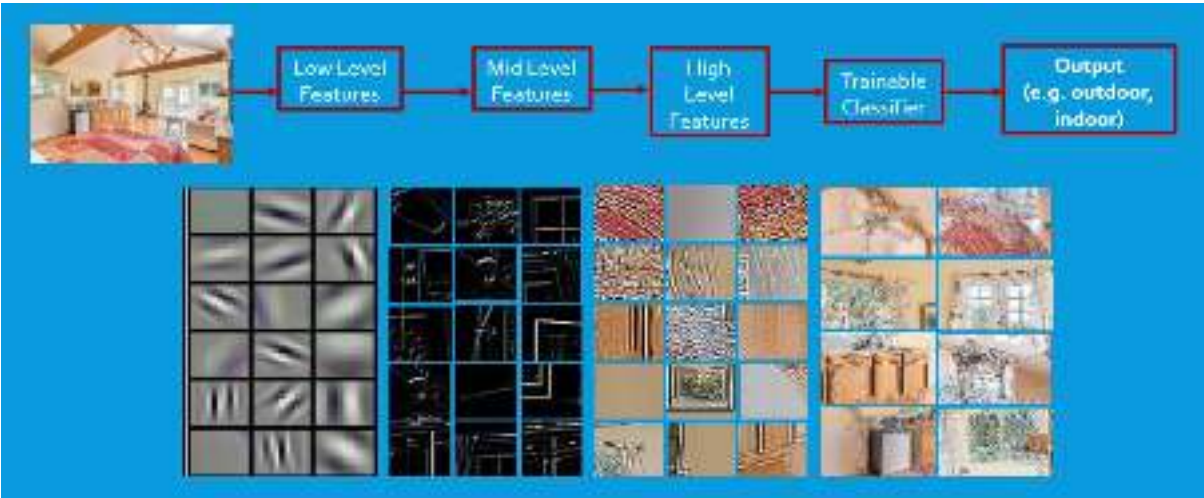


Gözetimsiz Öğrenme(Unsupervised Learning)



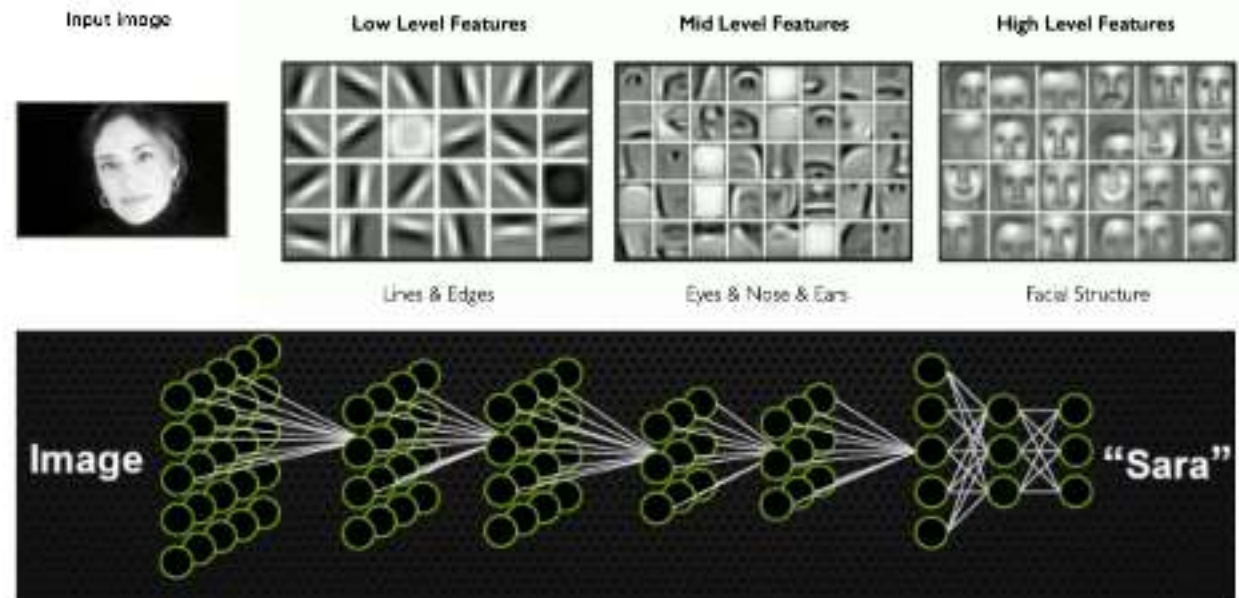
Öğrenme sadece önceden belli girdiler(X) üzerinden olur.

Özellik Çıkarımı



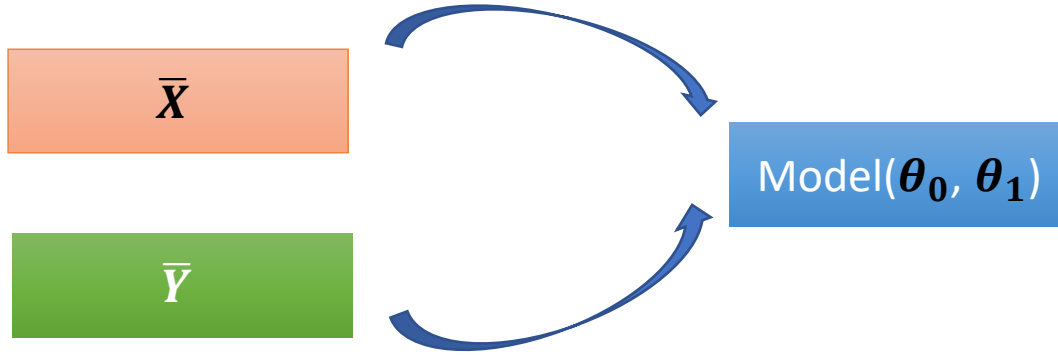
Kaynak: <https://www.kaggle.com/discussions/getting-started/133002>

Progressive extraction of features by a Neural Network

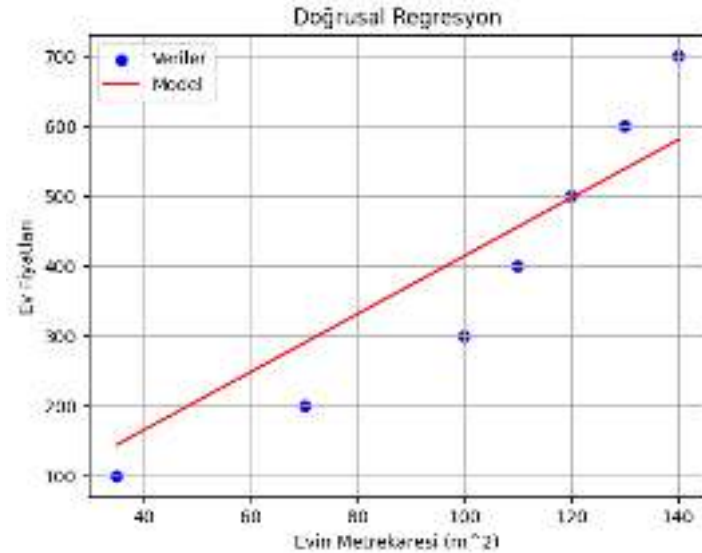


Kaynak : <https://www.linkedin.com/pulse/comprehensive-overview-deep-learning-ritika-dokania-7ksfc/>

Gözetimli Öğrenme(Supervised Learning)



Öğrenme önceden belli olan girdi(x) ve çıktılar(y) üzerinden olur.



Uyumsama(Regression)=Devamlı olarak çıkış(Y) değerini tahmin etme

Hipotez

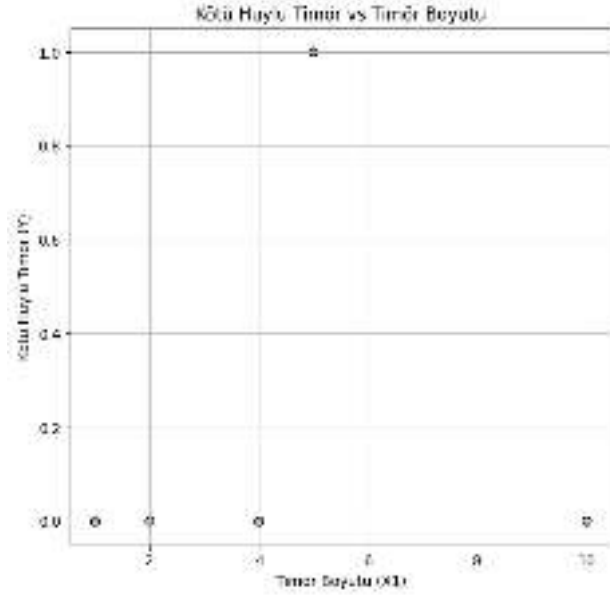
$$h(X)=\theta_0 + \theta_1 X$$

Hipotezimiz bir doğrunun denklemi olsun.

Fiyat(Y)	m2(X)
100	35
200	70
300	100
400	110
500	120
600	130
700	140

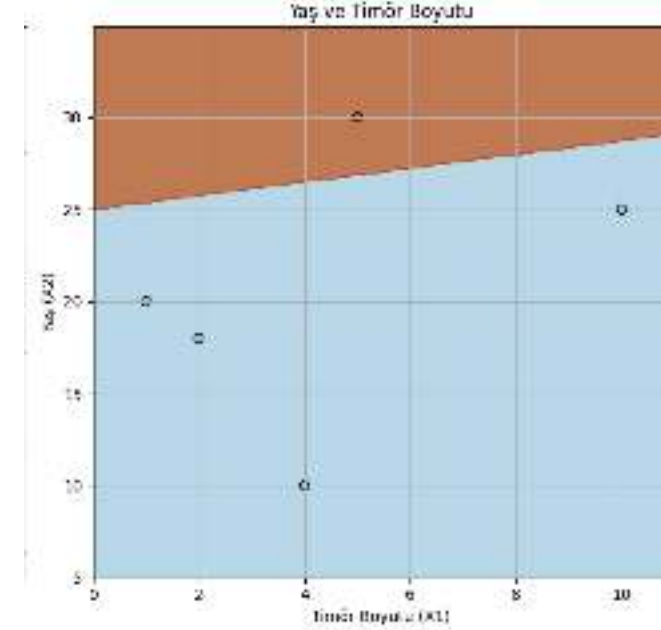
Gözetimli Öğrenme(Supervised Learning)

Sınıflandırma(Classification)=Doğru(1) yada yanlış (0) değerlerini verilen istatistiki tahmin üzerinden bularak etiketleme



Kötü huylu timör(Y) Timör boyutu(X)

0	1mm
0	2 mm
0	4mm
1	5mm
0	10 mm

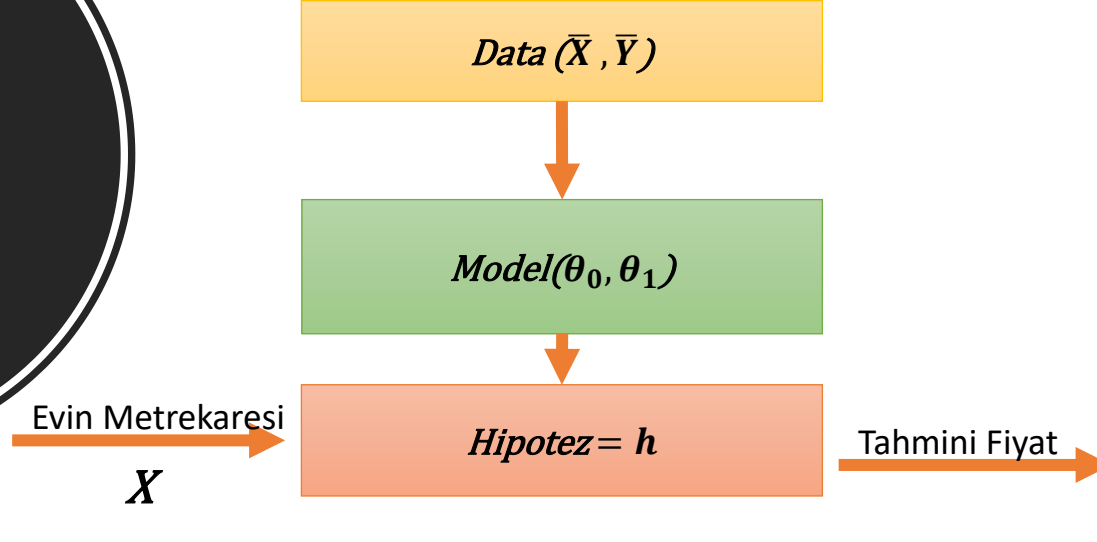


Kötü huylu timör(Y) Timör boyutu(X) yaş(X2)

0	1mm	20
0	2 mm	18
0	4mm	10
1	5mm	30
0	10 mm	25

Eğitim algoritması (Training Algorithm)

Makina öğrenme ile eğitim ve tahmin.

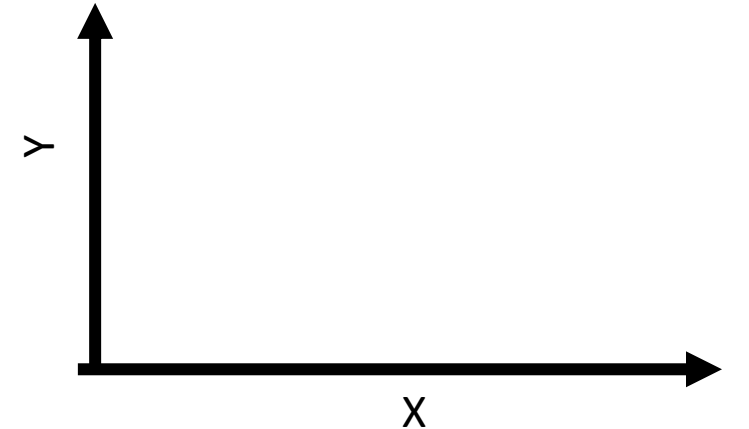
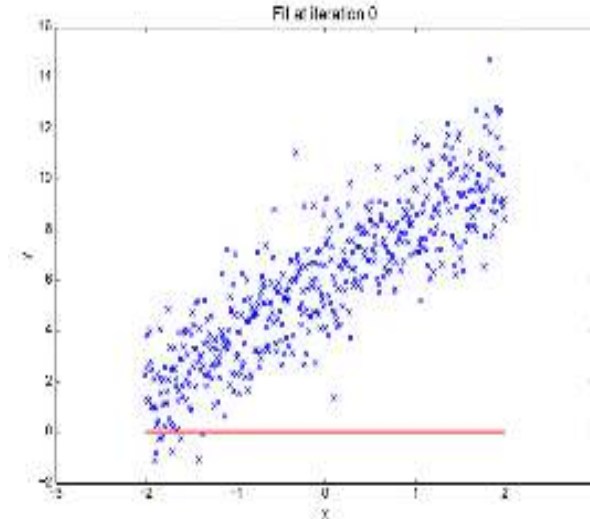


Burada önemli olan h fonksiyonunun nasıl bulunacağıdır.

Hipotez

$$h(X) = \theta_0 + \theta_1 X$$

Hipotezimiz bir doğrunun denklemi olsun.



Maliyet Hesabı(Cost) (Hata Hesabı)

- Tüm veri kümesini incelediğimizde, yapay zeka çıktıların ($\bar{h}$) gerçek çıktılarından (Y) ne kadar yanlış olduğunu gösteren bir fonksiyon oluşturabiliriz. Bu fonksiyona **Maliyet Fonksiyonu** denir.

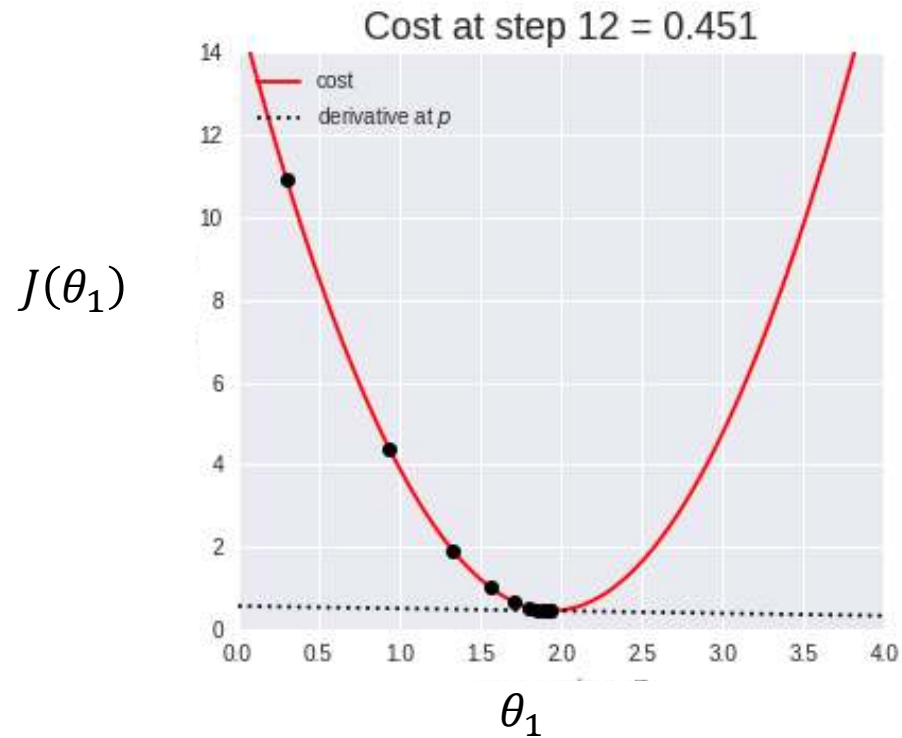


$$J = \frac{1}{2m} \sum_{i=1}^m (h - Y)^2$$

Maliyet Fonksiyonu(Cost Function)

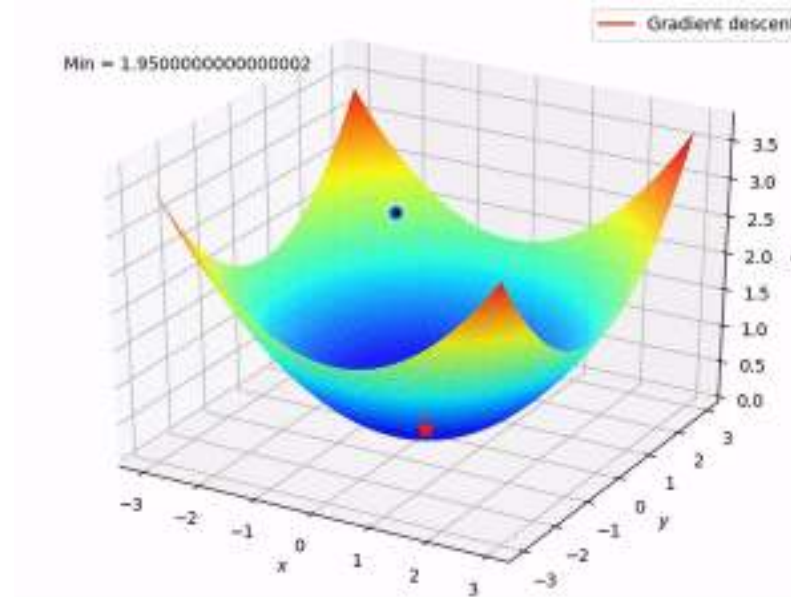
Basitleştirilmiş maliyet fonksiyonu

Tek Parametre



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h^i - Y^i)^2$$

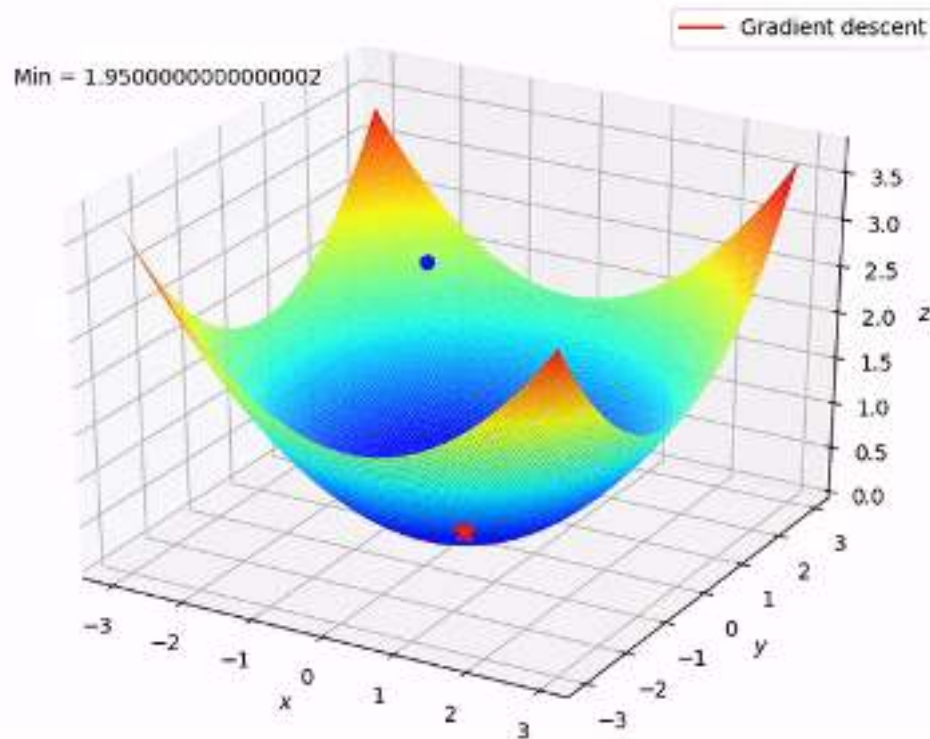
Çift Parametre



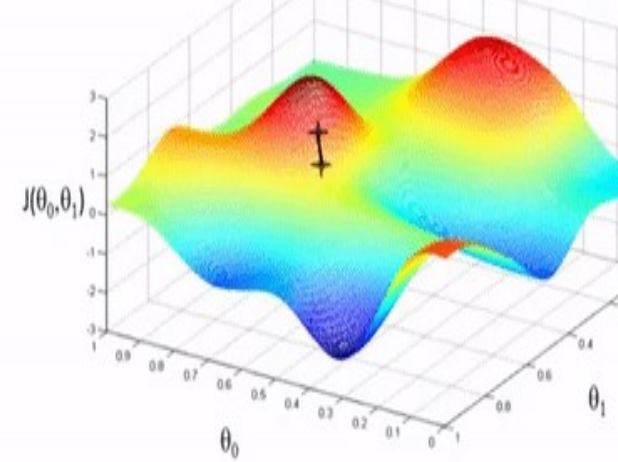
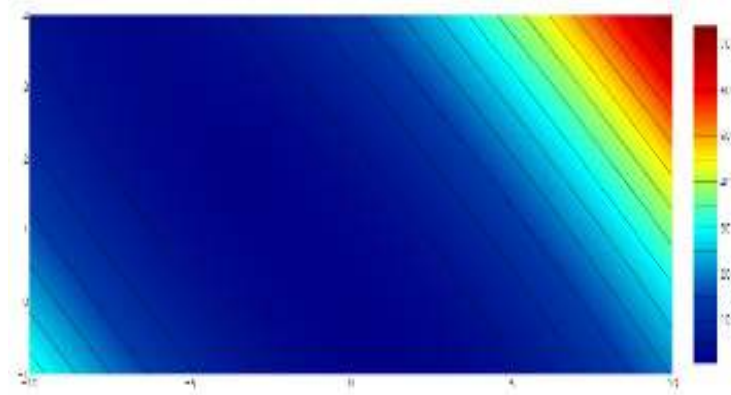
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h^i - Y^i)^2$$

Gradient düşümü(Gradient descent)

Tek Parametre



Çift Parametre



Gradient düşümü(Gradient descent)

$$\text{Amaç: } \min_{\theta_0, \theta_1} \{J(\theta_0, \theta_1)\}$$

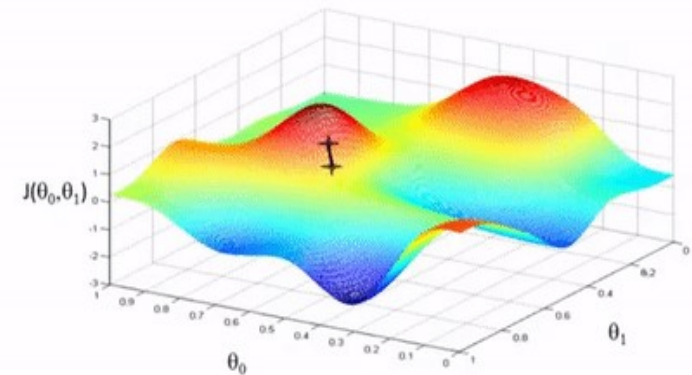
- θ_0 ve θ_1 ' e rastgele değerler vereceğiz.
Örnek : $\theta_0 = 0$ ve $\theta_1 = 0$
- θ_0 ve θ_1 değerlerini $J(\theta_0, \theta_1)$ minimum değeri verene kadar değiştir.

Algoritma:

En küçük değere ulaşana kadar yenile{

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}



Gradient düşümü(Gradient descent)

Algoritma:

En küçük değere ulaşana kadar yenile{

$$geçici_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$geçici_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 = geçici_0$$

$$\theta_1 = geçici_1$$

}

