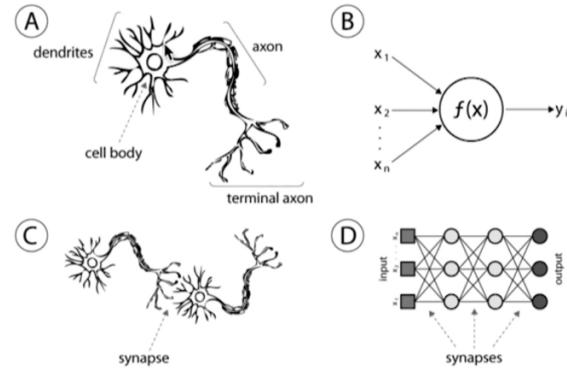


# Yapay Sinir Ağları (YSA) Artificial Neural Network (ANN)

- Sinirbilim ve Bilgisayar Bilim (A)
- Nöron (Neuron) (B)
- Sinapsis (Synapsis) (C)
- YSA Çalışma Mantığı (örnek üzerinden) (D)
- 



Acaba biz insan beynindeki nöron yapısını bilgisayarda modelleyerek, insan gibi çalışan bilgisayar yapabilir miyiz ?

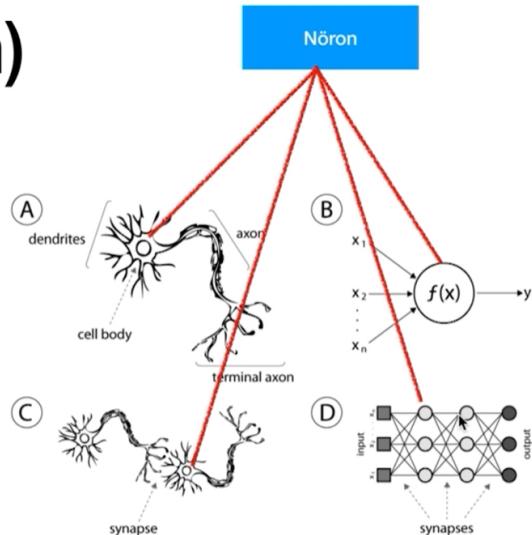
Uzun yıllar önce tasarılanıyor fakat öncelerde bu kadar parlak olmamasının sebebi görece yavaş çalışıyordu ve fazla kaynak istiyor.

Bugün ise bilgisayarlar hızlandı ve GPU'nun çok daha yaygın olarak bulunabilmesi(ucuz)

Neural Network'un önünü açan gelişmeler

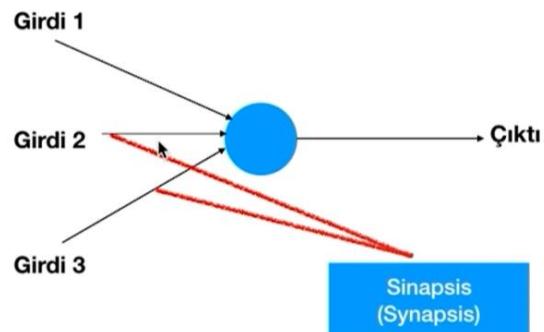
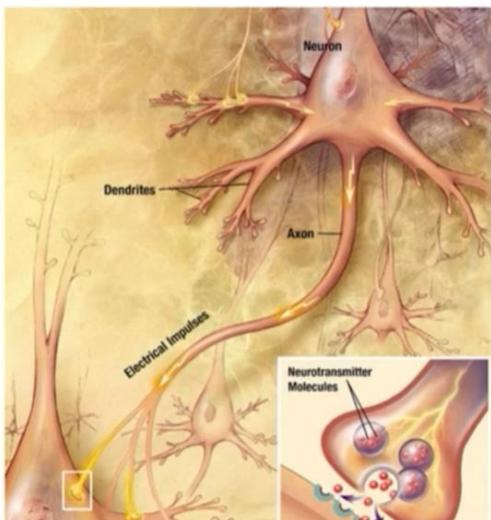
## Nöron (Neuron)

- Sinirbilim ve Bilgisayar Bilim (A)
- **Nöron (Neuron) (B)**
- Sinapsis (Synapsis) (C)
- YSA Çalışma Mantığı (örnek üzerinden) (D)
- YSA ile öğrenmek ne demektir ve bir örnek üzerinden öğrenme
- Gradient Descent (Gradyan iniş)
- Stochastic Gradient Descent
- Backpropagation

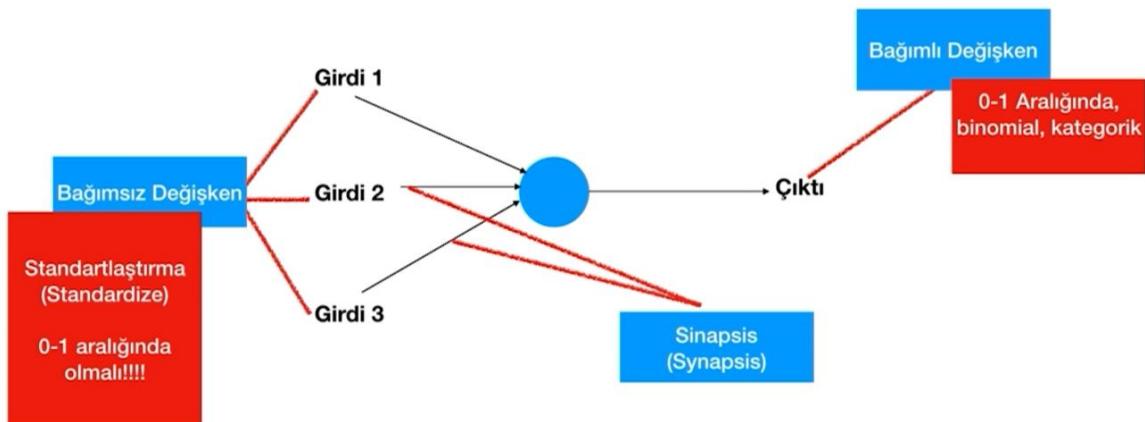


Sinapsislerden gelen inputların karar verildiği yerdir.

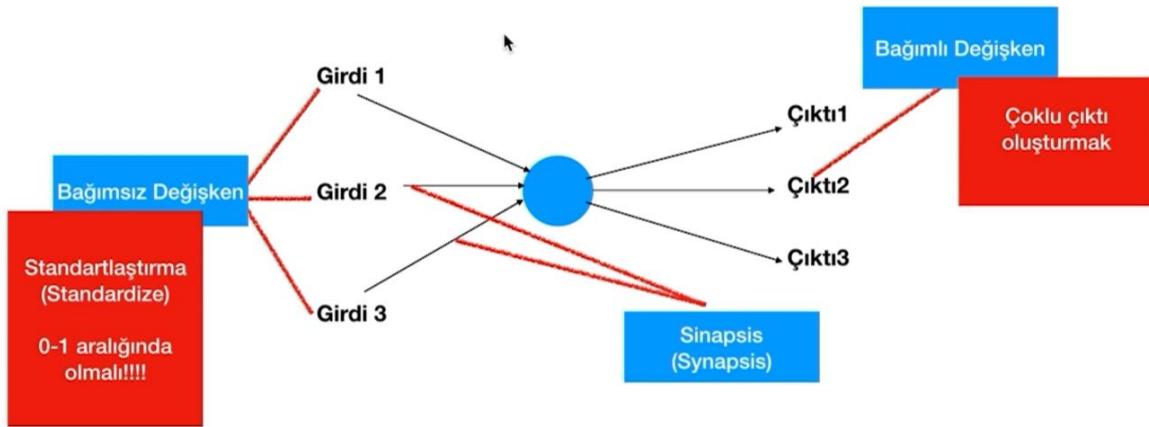
# Nöron (Neuron)



# Nöron (Neuron)



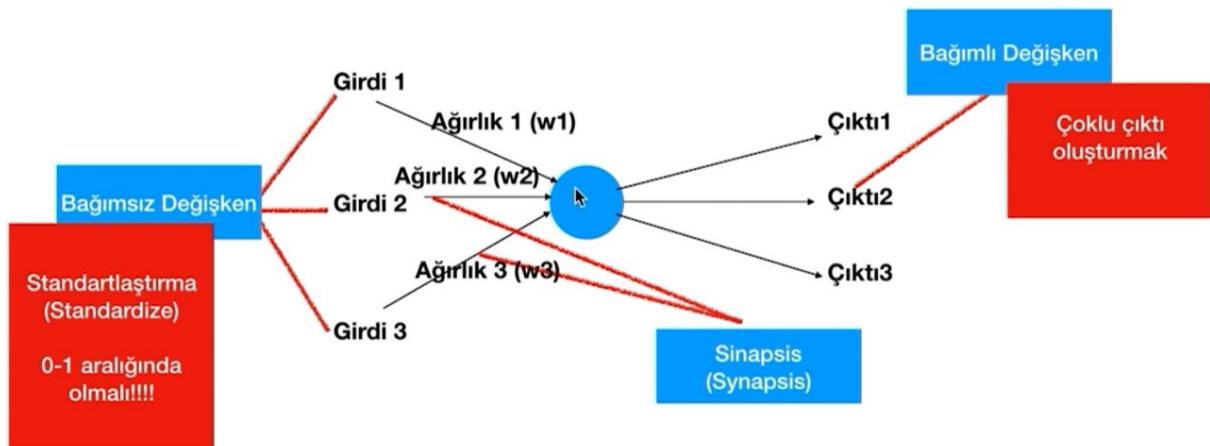
# Nöron (Neuron)



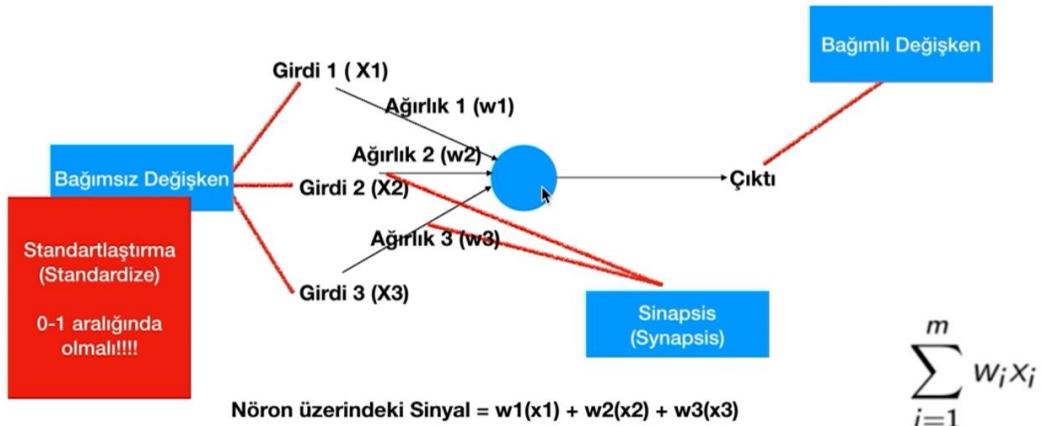
Sinapsisler üzerinde özel bir durum var, Ağırlıklar taşınıyor

Girdi bir sinyal olarak geliyor bu gelen sinyal norona taşınırken üzerinde bir çarpanla taşınabiliyor

# Nöron (Neuron)

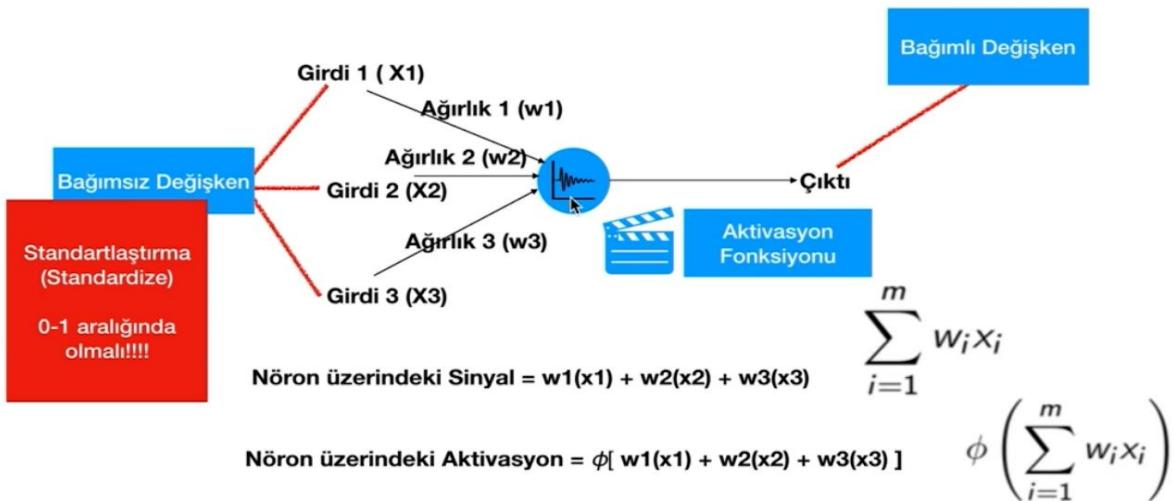


# Nöron (Neuron)



Nöron üstündeyi yükle(sinyala) göre karar veriyor.

# Nöron (Neuron)



Bir noronun aksiyon vermesi ile ilgili bir fonksiyon var

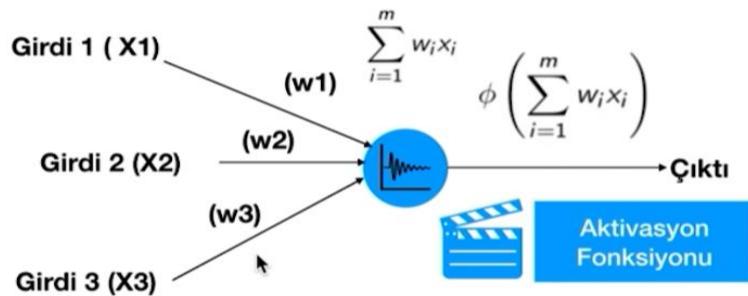
**Aktivasyon fonksiyonu;** basitce norunun karar vermesine etki eden fonksiyon

**ÖRNEK;** havada hafifce elinizi hareket ettirdiğinizde bir şey hissedilmez fakat hızlı şekilde ettirildiğinde rüzgari hissedersiniz, Bu hissi almak için gerekli sinir yapısı **belirli bir eşik değerini gecene kadar aktif olmuyor.**

Aktivasyon fonk'da bu şekilde noronun bir eylemi yapıp yapmayacağına karar vermesinde etkili.

### Activation Function

# Aktivasyon (Activation)



$$\text{Nöron üzerindeki Sinyal} = w_1(x_1) + w_2(x_2) + w_3(x_3)$$

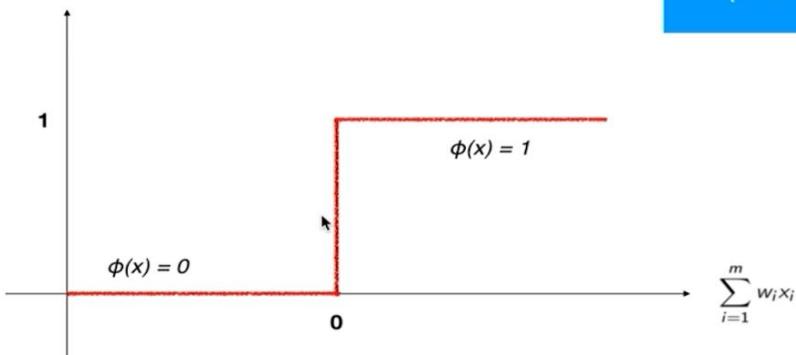
$$\text{Nöron üzerindeki Aktivasyon} = \phi[ w_1(x_1) + w_2(x_2) + w_3(x_3) ]$$

Nöronun üstünde sinapsislerden gelen ağırlıklı bir yük var, bu yük aktivasyon fonksiyonuna giriyo ve bu fonksiyon sinyali ateşliyor ya da ateşlemiyor.

- Hemen hemen bütün fonksiyonlar aktivasyon fonksiyonu olarak kullanılabilir ama bazıları daha çok kullanılır. Bunlardan Bazıları;
- Threshold Fonksiyonu(Adım fonksiyonu), Sigmoid Function

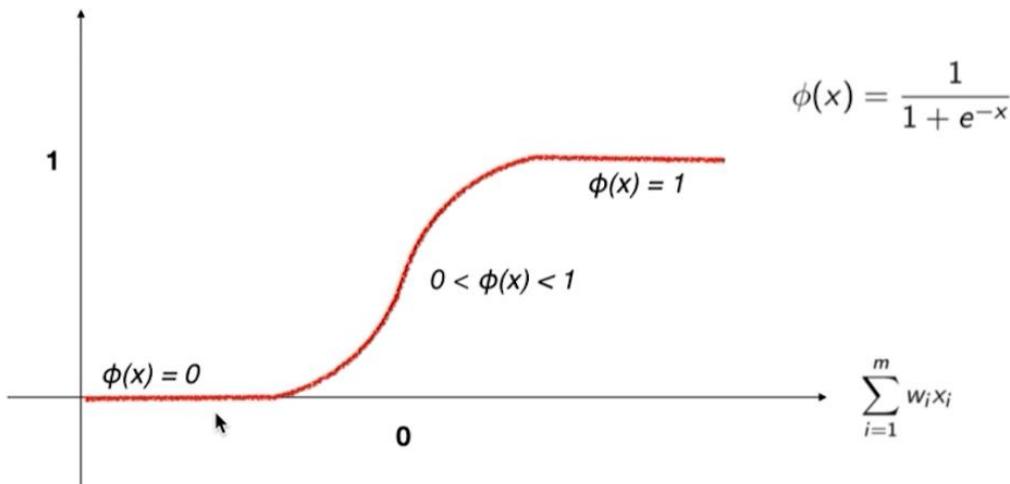
# Adım Fonksiyonu (Step Function)

Eşik Fonksiyonu  
(Threshold Function)



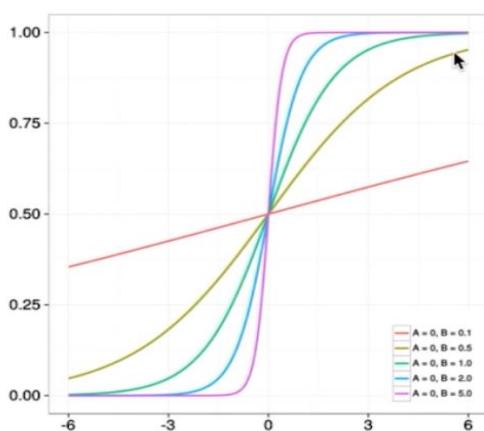
Belirli bir eşik değeri geçince atlama yapması

# S Fonksiyonu (Sigmoid Function)



Step fonksiyonunda sadece 0 ve 1 vardı fakat burada 0'dan 1'e değerler alabilir.

# Sigmoid Fonksiyon



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

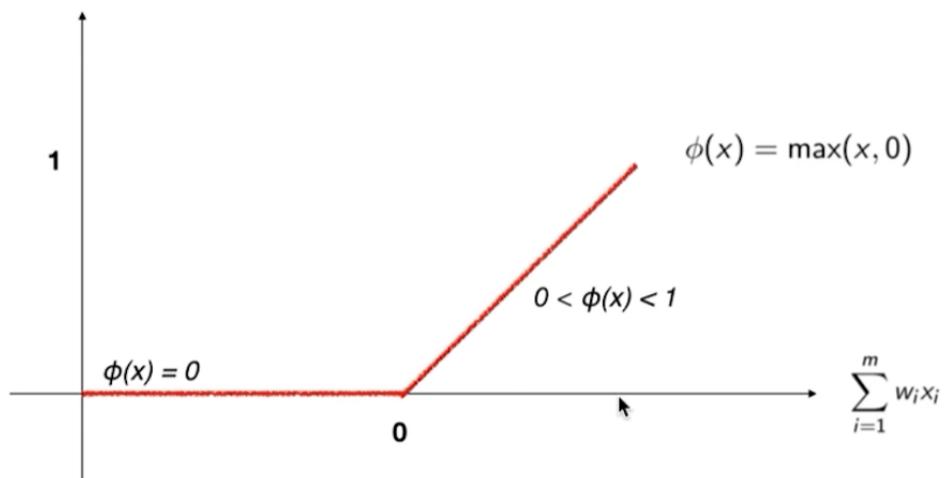
$$t = \beta_0 + \beta_1 x \quad t = A + Bx$$

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m = \beta_0 + \sum_{i=1}^m \beta_i x_i$$

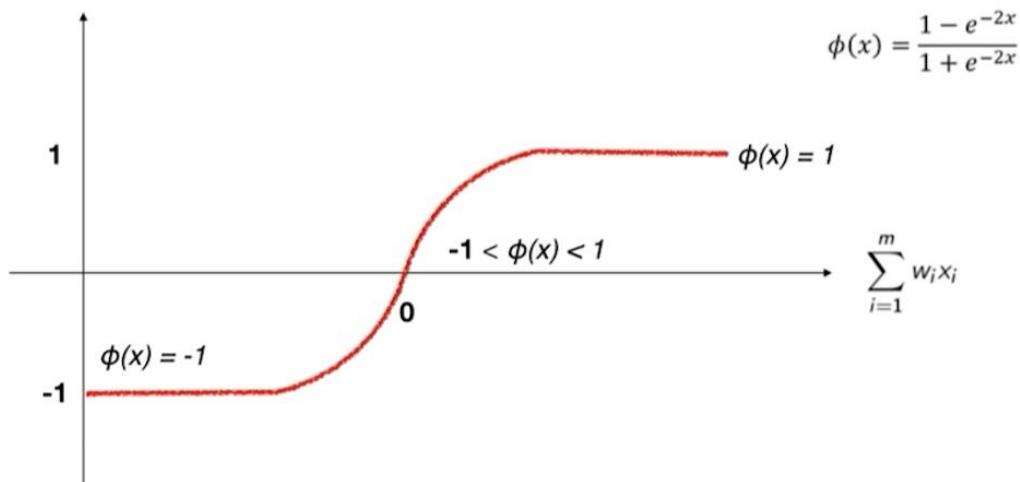
Hatırlatma: Logistic Regression

# Düzleştirilmiş Fonksiyon (Rectifier)

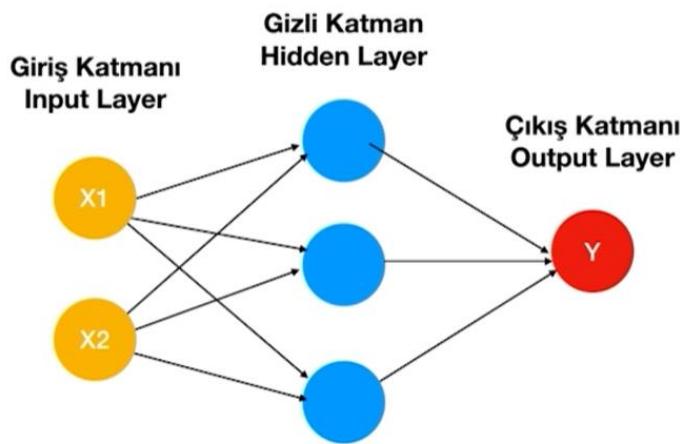


Rectifier'in özelliği eşik değerinden üstüne çıktığı anda değeri olduğu gibi yansıtması

# Hiperbolik Tanjant ( $\tanh$ ) (Hyperbolic Tangent)



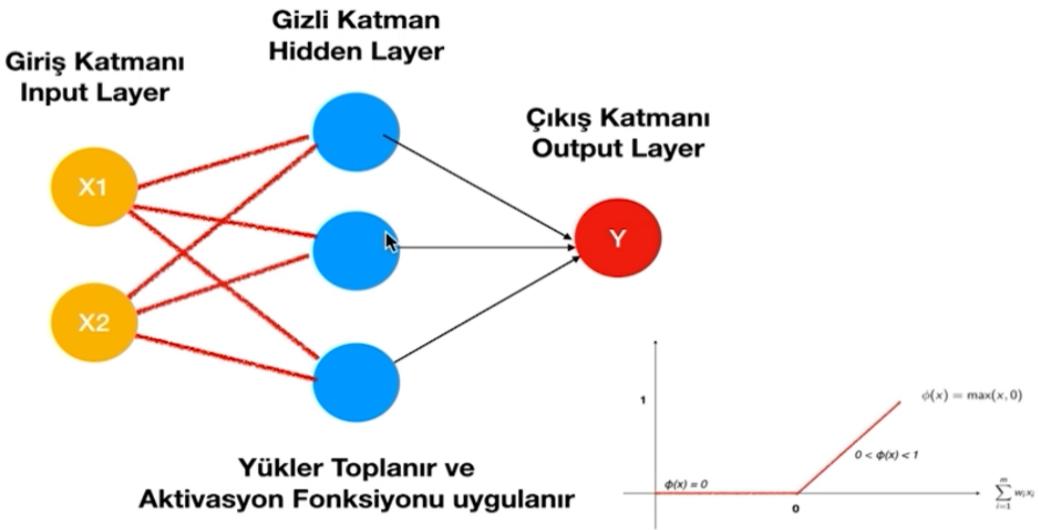
## Yapay Sinir Ağlarında Katman Kavramı (Layer)



Gizli katman sayısı 1'den fazla olabilir aynı şekilde nöron sayısı da 1'den fazla olabilir.

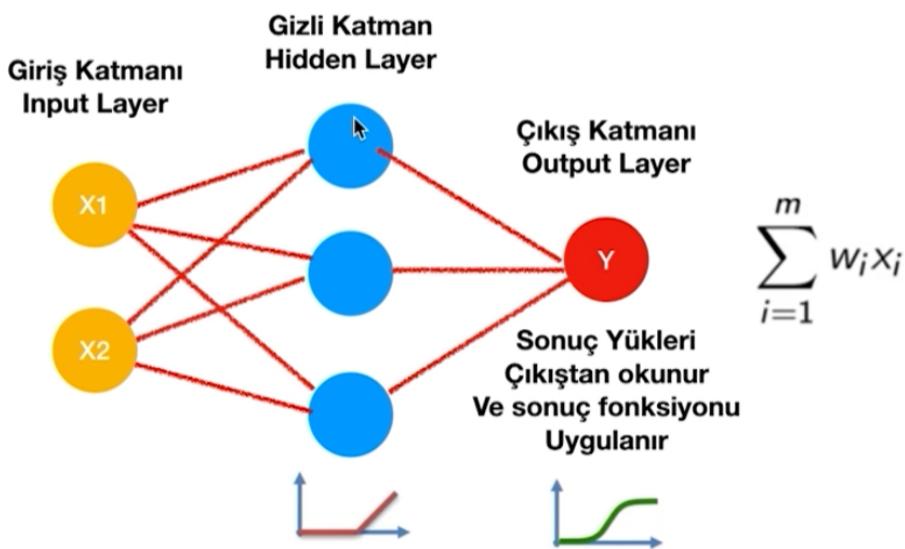
Her nöron bağımsız çalışıyor.

# Yapay Sinir Ağlarında Katman Kavramı (Layer)



Used function = rectifier [eşik değeri geçildikten sonra gelen sinayı aynı yansıtır]

# Yapay Sinir Ağlarında Katman Kavramı (Layer)



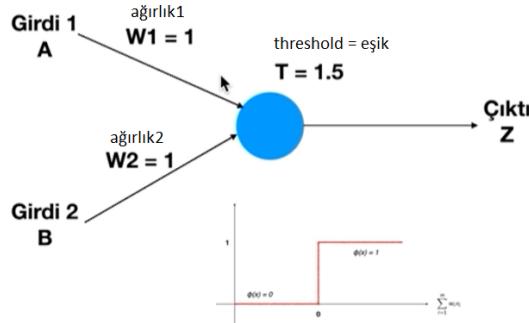
Cıkış katmanındaki nöron'da farklı bir aktivasyon fonksiyonu kullanılabilir.[sigmoid]

Her noronun aktivasyon fonksiyonu farklı olabilir

## Ve Kapısı (And Gate)

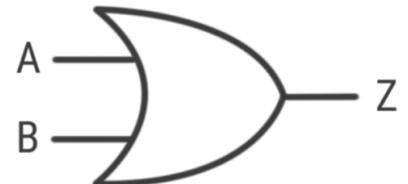


A	B	Z	W1 A	W2 B	W1A + W2B
0	0	0	0	0	0 < X
0	1	0	0	1	1 < X
1	0	0	1	0	1 < X
1	1	1	1	1	2 > 1.5 ✓

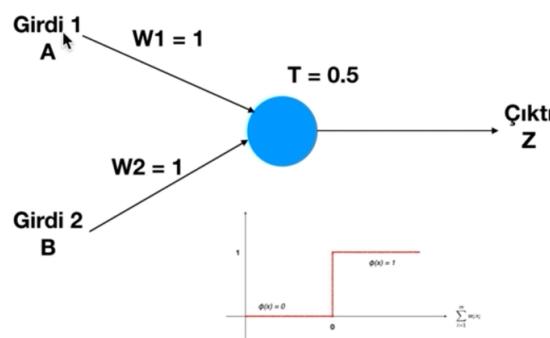


Neural network çalışma örneği.

## Veya Kapısı (Or Gate)

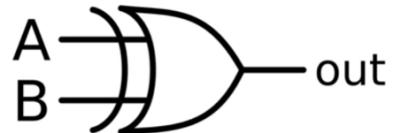


A	B	Z	W1 A	W2 B	W1A + W2B
0	0	0	0	0	0 < X
0	1	0	0	1	1 > ✓
1	0	1	1	0	1 > ✓
1	1	1	1	1	2 > ✓

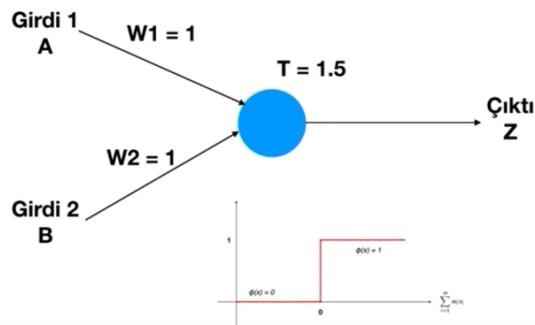


Xor Gate(Özel veya kapısı)

# Özel Veya Kapısı (Xor Gate)

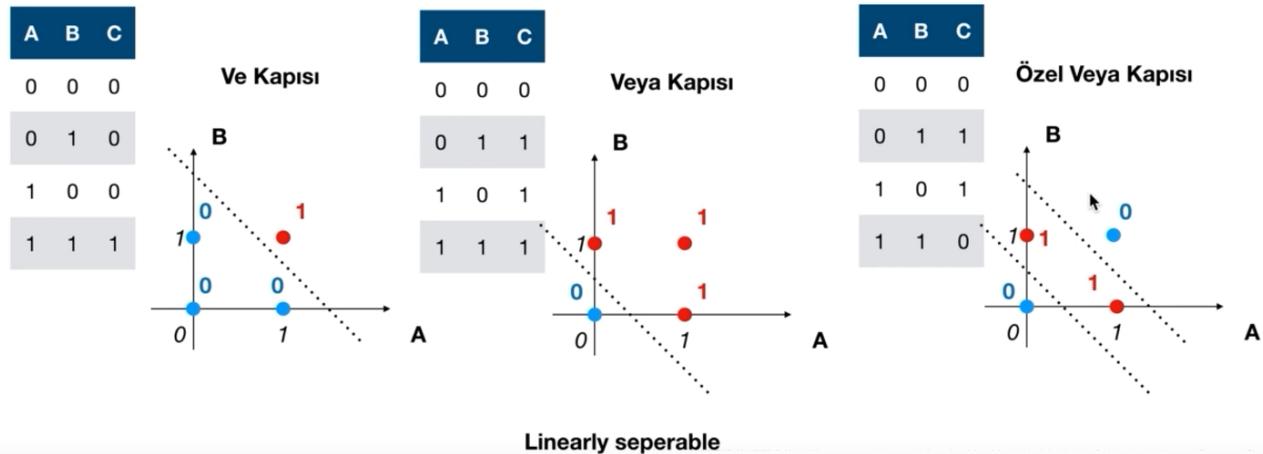


A	B	Z	W1 A	W2 B	W1A + W2B
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	1	1	2



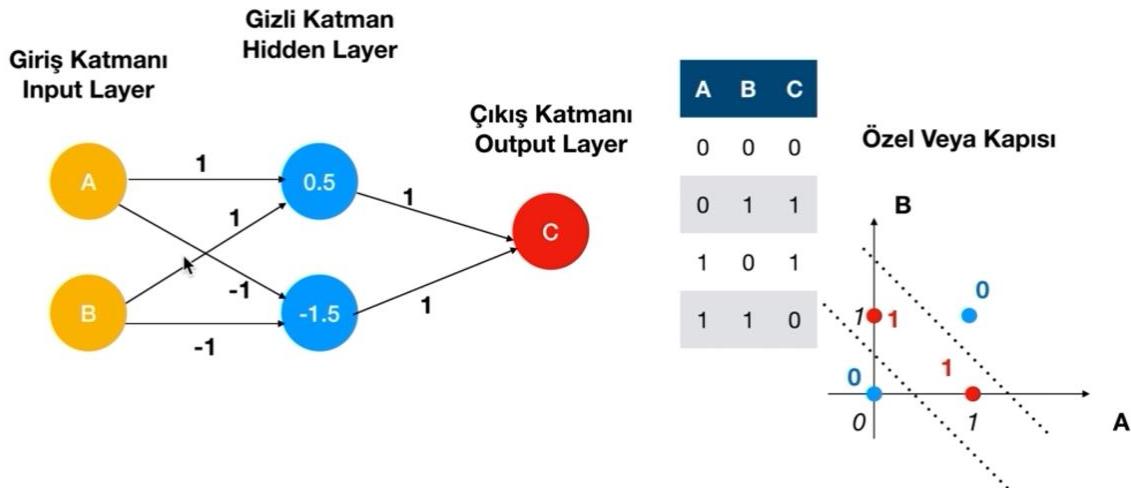
Bu problem tek nöron ile çözülmüyor ;

## 2 Boyutlu Uzay

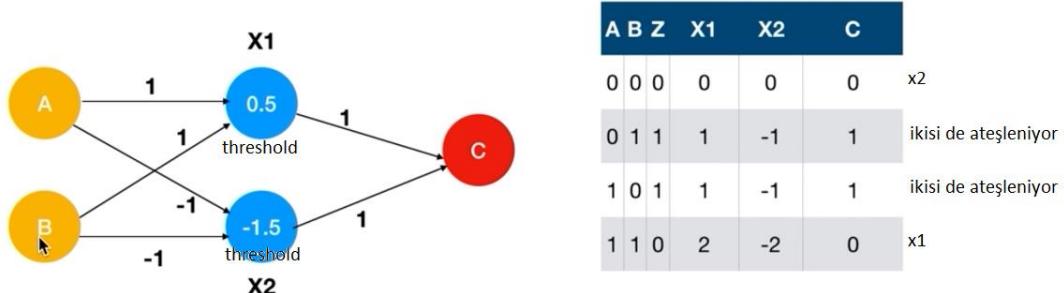


Xor gate doğrusal olarak ayrılamıyor

# 2 Boyutlu Uzay

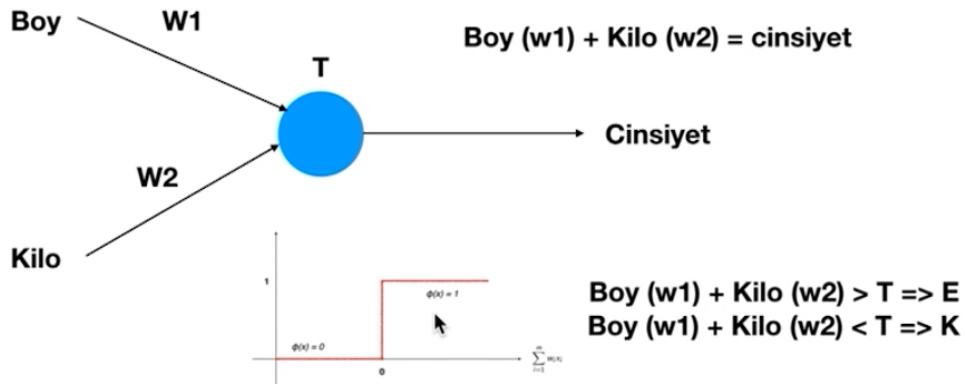


# 2 Boyutlu Uzay

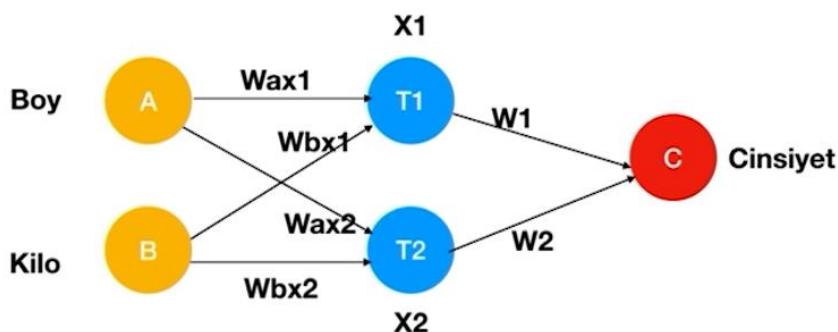


Burada and ve or kapılarını birleştiriyoruz

# Gerçek Örnek



# Gerçek Örnek



SVM'de olan çekirdek hilesi(kernel trick) = doğrusal olarak ayrılmayan verileri 3.boyuta taşıyarak ayırma

Bu problemi 1'den fazla nöron kullanarak yapay sinir ağlarıyla çözmek mümkün

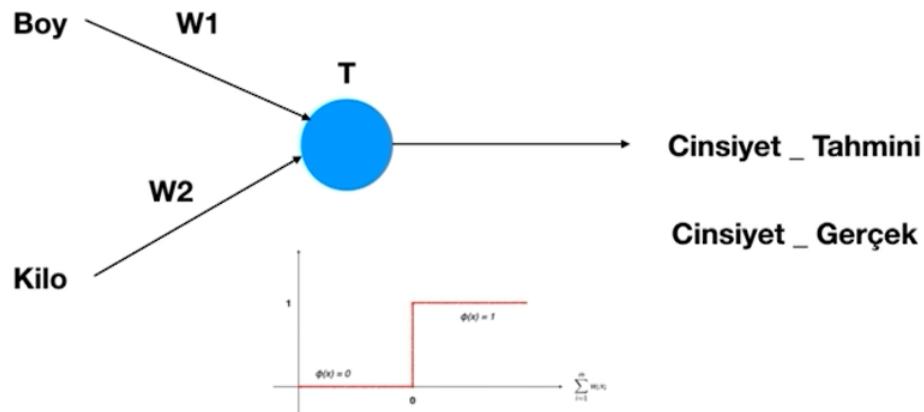
How does Artificial Neural Network learn?

Bunun için çok farklı yöntem var

En basitlerinden biri;

Perceptron(algılayıcı) kavramı

# YSA Nasıl Öğrenir?

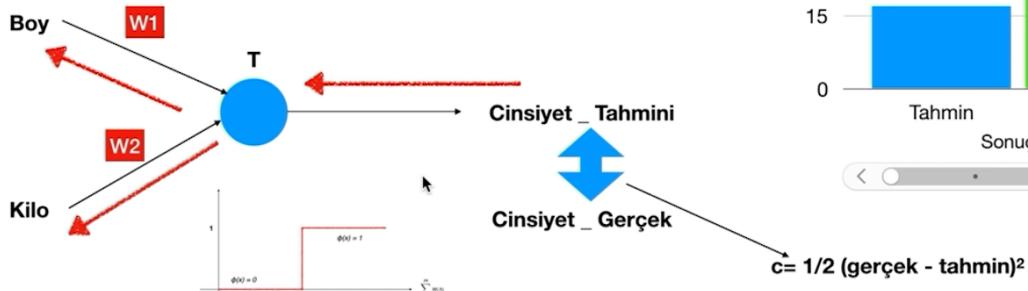


Cinsiyet tahimini yapıldıktan sonra;

Bu tahmin doğru ise yapay sinir ağında herhangi bir değişiklik olmuyor(öğrenilcek bi şey yok)

Bu tahmin yanlış ise veya gerçek ile tahmin arasında fark var ise Tahmin değerinden Gerçek değerini çıkartarak yapay sinir ağına tekrar gönderiyor ve yapay sinir ağının Ağırlıkları ve threshold değerini her seferinde düzenleyerek en iyi tahmin sonuçlarını bulmayı hedefliyor(geri besleme)

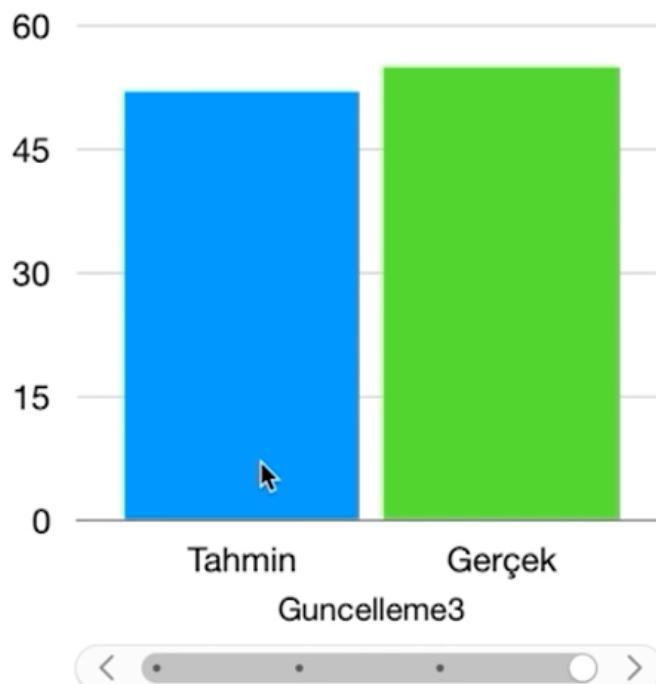
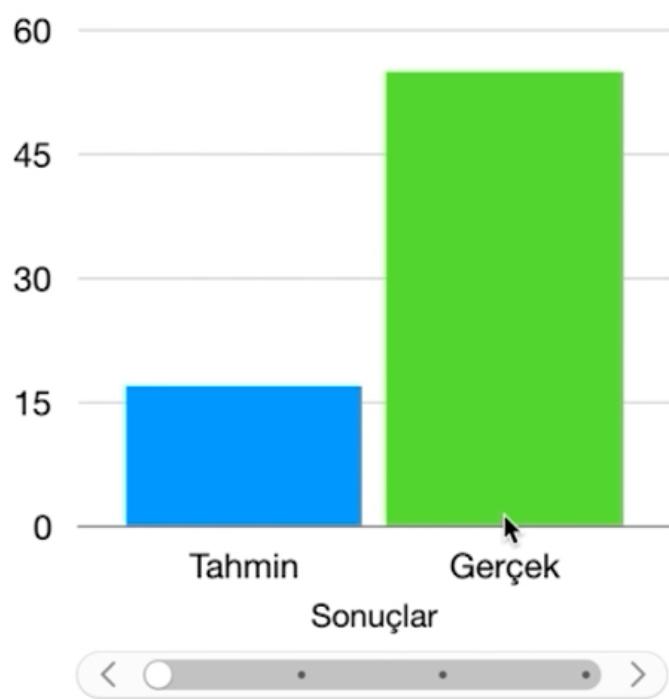
# Algılayıcı Kavramı (Perceptron)



Burada farklı bir problem daha var geri yansıtılacak değer "c" bir çarpanla çarpılarak gönderilebilir

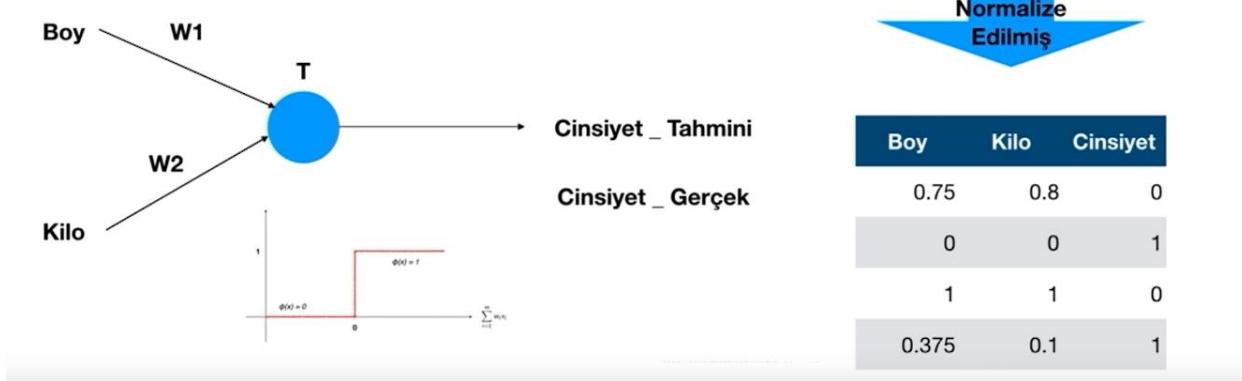
Cünkü **c'nin çok büyük olması** yapay sinir ağını olumsuz etkileyerek önceki bildiklerini de unutturabilir(ani şekilde sistemi etkileyebilir)

Veya **c'nin çok küçük olması durumunda** gerekli düzenlemeler yapılamaz



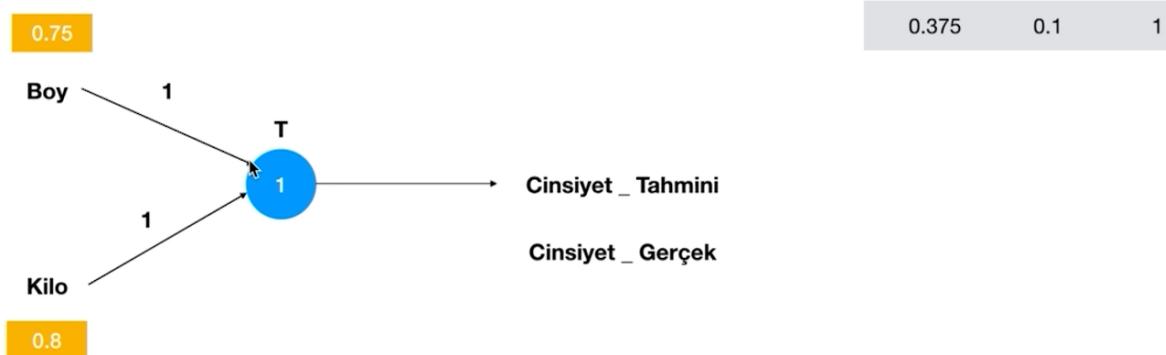
## Gerçek Örnek

# Algılayıcı Kavramı (Perceptron)



Öncelikle verilerin normalize edilmesi gerek [ 0 ve 1 arasında çalışır], ayrıca cinsiyet encode edilmiş.

# Algılayıcı Kavramı (Perceptron)

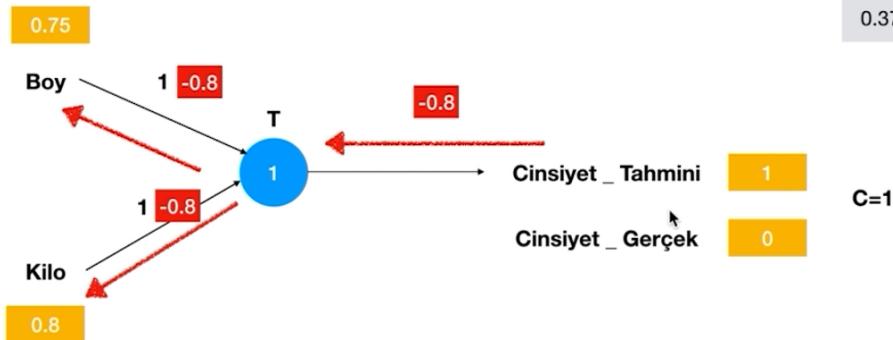


Ağırlık değerleri ve threshold'lara rastgele değerler verilebilir. Çünkü deep learning ve reinforced algoritmalarının mantığı;

Çok kötü bir algoritmayla çok kötü bir yerden başlıyor olsa bile sistem kendini eğiterek iyi bir yere gelecektir

Adımları tek tek inceleyelim.

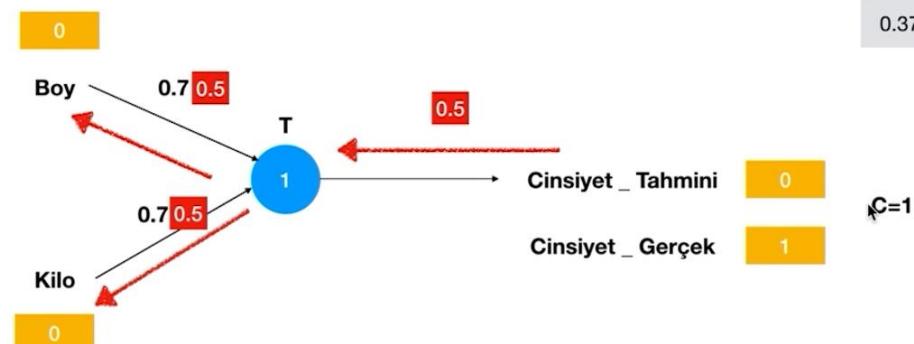
## Algılayıcı Kavramı (Perceptron)



Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	
1	1	0	
0.375	0.1	1	

Hatalı tahmin yapıldı ve ağırlıklar değişecek.

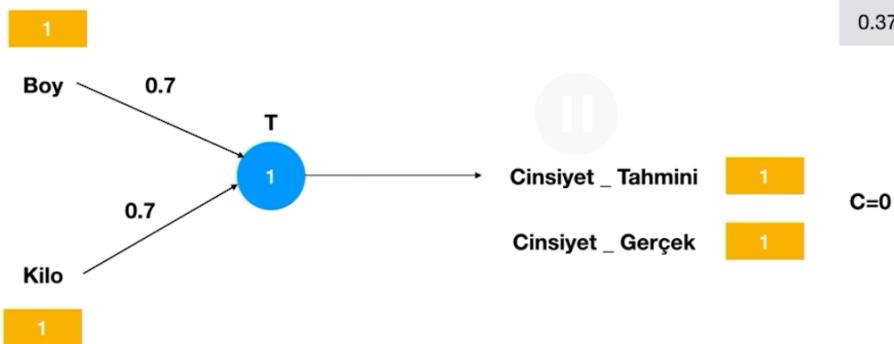
## Algılayıcı Kavramı (Perceptron)



Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	1
1	1	0	
0.375	0.1	1	

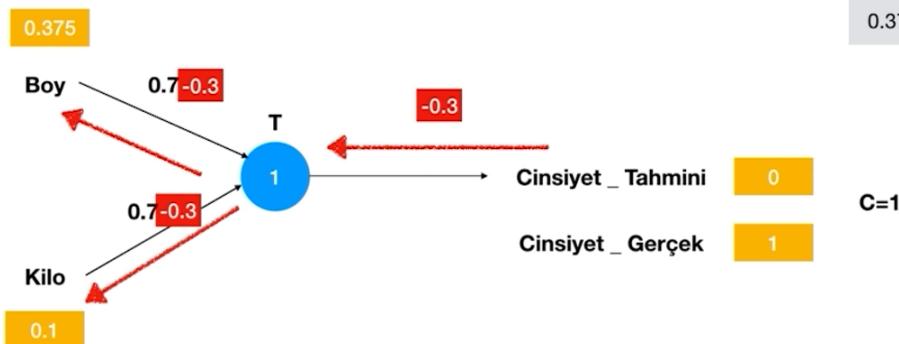
Tekrar hatalı tahmin ağırlıklar değişecek

# Algılayıcı Kavramı (Perceptron)



Doğru tahmin ceza yok, herhangi bir değişiklik yapılmayacak

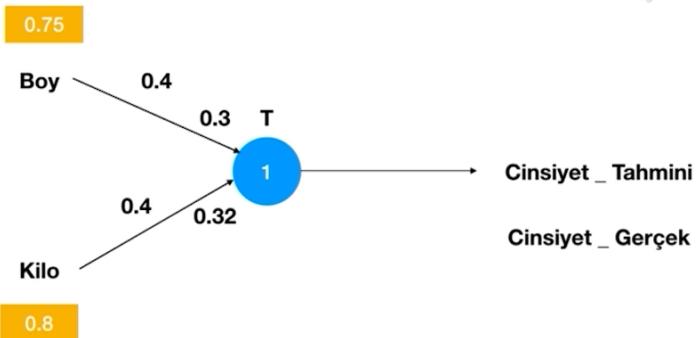
# Algılayıcı Kavramı (Perceptron)



Hata tekrar ağırlıklar değişecek.

Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	1
1	1	0	0
0.375	0.1	1	

# Algılayıcı Kavramı (Perceptron)



Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	1
1	1	0	0
0.375	0.1	1	1

ilk örneği tekrar verirsek?

0.75      0.8      0      1

Ağırlıkların en son hali bu şekilde

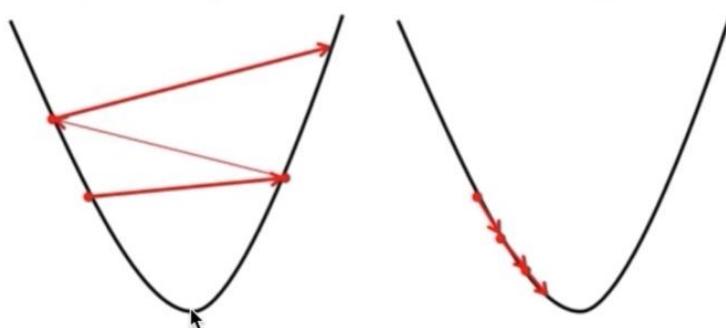
şimdi 1. örneği tekrar verirsek bu sefer doğru tahmin eder.

# Öğrenme Oranı ve Gradyan İniş

Büyük Öğrenme Oranı      Küçük Öğrenme Oranı

Big learning rate

Small learning rate



Önceden verdiğimiz cinsiyet tahmin örneğinde  $c(\text{hata}) = 1$  olsa bile her seferinde azalarak yansıtıldı [önce 0.8 sonra 0.5 sonra 0.3 gibi ]

NEDEN 1 OLARAK YANSITMADIK Paralel bir grafik elde ederdik bunun sonucu olarak;

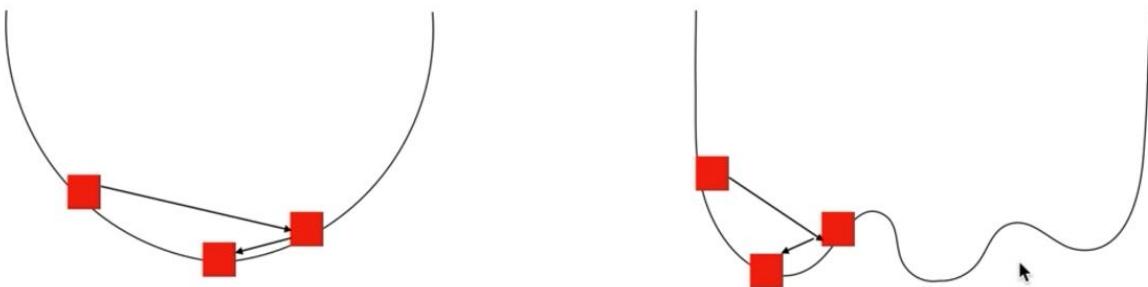
Hiçbir zaman ilerleme katedemeyebildik, Eğer hata oranını yanlış verirsek yüksek verirsek bizi olumsuz bir noktaya götürürebilir

Yapay sinir ağının doğru şekilde öğrenmesi için Gradyan descentend (yumuşak çıkış)

**Gradyan descentend:** Yapmış olduğumuz hataları, en iyi noktaya götürmesi ile ilgili nasıl adım atacağımız, adım değerlerinin [ağırlıklarını yansıtma] nasıl değişeceği ile ilgili bir yöntem

Optimum noktayı bulma problemi.

# Gradyan (Yerel ve Küresel En iyi)



Gradyan alçalısın problemi: bu örnekte olduğu gibi daha iyi bir noktaya gitmemizi engelleyebilir

## Gradyan Alçalış ve İki Farklı Yaklaşım

Boy	Kilo	Cinsiyet	C	
0.75	0.8	0	1	→ Alçalt
0	0	1	1	→ Alçalt
1	1	0	0	→ Değiştirme
0.375	0.1	1	1	→ Alçalt

Alçalt ←

Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	1
1	1	0	0
0.375	0.1	1	1

Stokastik Yaklaşım

Batch (Yığın) Yaklaşımı

stokastik yaklaşımında tek tek değerlere göre alçalt, değiştirme olurken

Batch(yığın) yaklaşımı : bütün veriye bakarak alçalt[tek bir kere çalışmıyor]

Boy	Kilo	Cinsiyet	C	
0.75	0.8	0	1	Alçalt
0	0	1	1	
1	1	0	0	Alçalt
0.375	0.1	1	1	

Mini Batch (Mini Yığın) Yaklaşımı  
Mini Yığın boyutu değişebilir

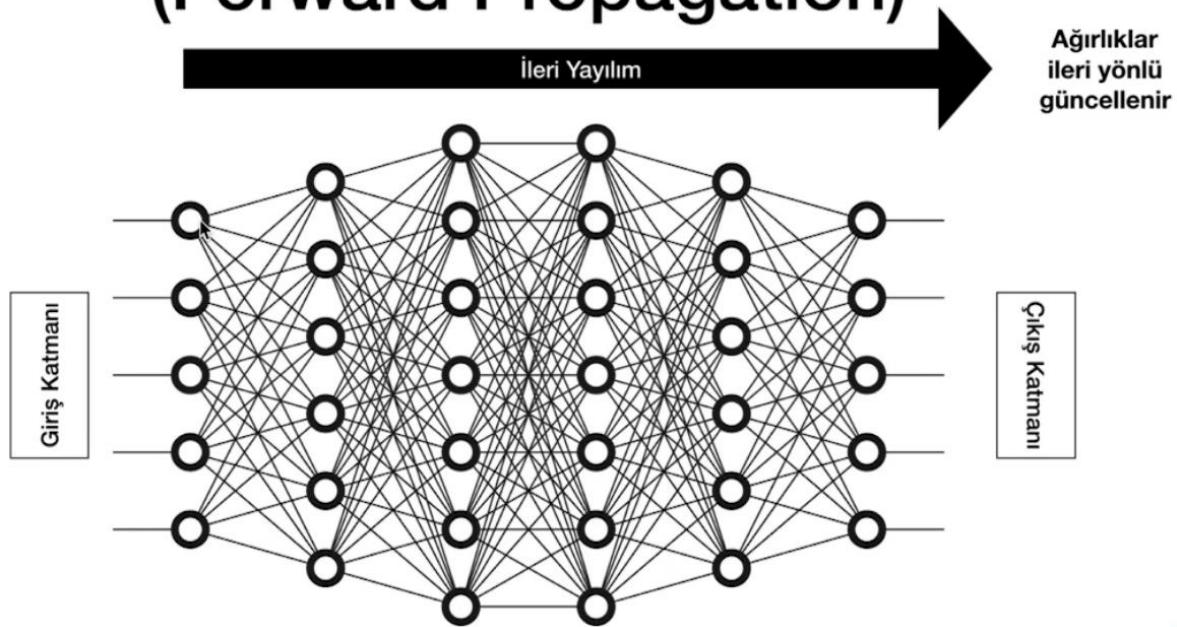
Iki

yaklaşım arasında bir yaklaşım olarak kabul edilebilir

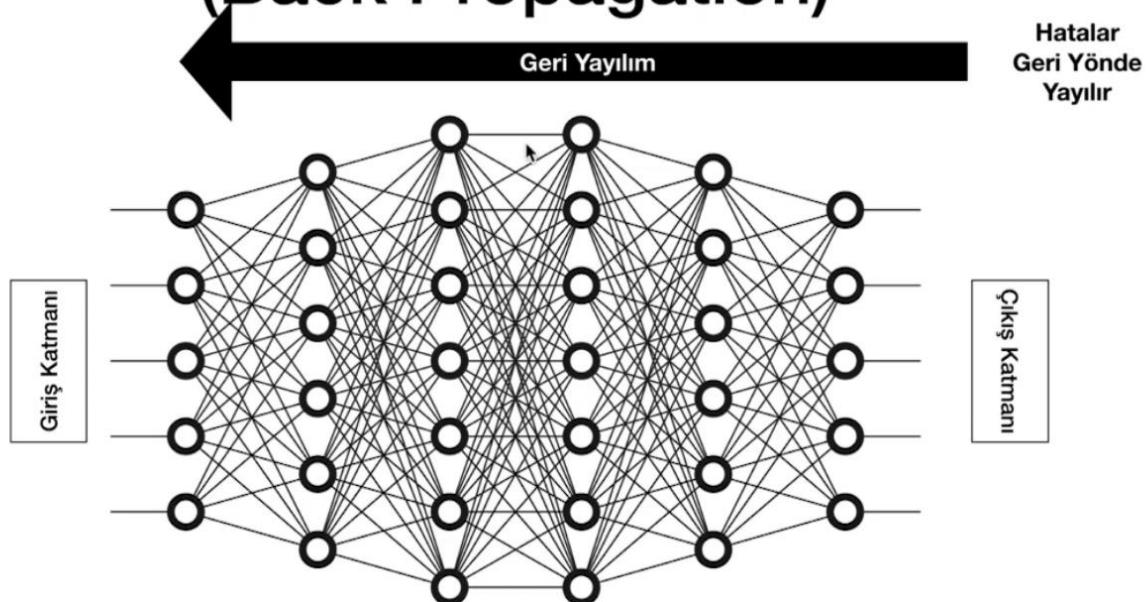
Mini yığınlar halinde karar veriyor.

YSA ÖĞRENME ALGORİTMASI VE İLERİ/GERİ YAYILIM

# İleri Yayılım (Forward Propagation)



# Geri Yayılım (Back Propagation)



# Algoritma Adımları (Backward Propagation)

- Adım 1: Bütün ağı rasgele sayılarla (sıfıra yakın ama sıfırdan farklı) ilkendir (threshold ve ağırlıkları)
- Adım 2: Veri kümelerinden ilk satır (her öznitelik bir nöron olacak şekilde) giriş katmanından verilir (boy, yaşı, kilo = 3 nöron)
- Adım 3: **İleri yönlü yayılım yapılarak**, YSA istenen sonucu verene kadar güncellenir. ( istenen veri çıktı mı? çıktımadı o zaman güncelle)
- Adım 4: Gerçek ve çıktı arasındaki fark alınarak hata (error) hesaplanır
- Adım 5: **Geri Yayılım yapılarak**, her sinapsis üzerindeki ağırlık, hatadan sorumlu olduğu miktarda değiştirilir. Değiştirilme miktarı ayrıca öğrenme oranına da bağlıdır. geri yayılım ise hangi sinapsis üzerinde ne kadar değişiklik yapacağız onu öğrenmiş oluyoruz
- Adım 6: Adım 1 - 5 arasındaki adımları istenen sonucu elde edene kadar güncelle (Takviyeli öğrenme (Reinforced Learning)) veya eldeki bütün verileri ilgili ağıda çalıştırıldıktan sonra bir seferde güncelleme yap (yığın öğrenme (batch learning))
- Adım 7: Bütün eğitim kümesi çalıştırıldıktan sonra bir çağ/tur (epoch) tamamlanmış olur. Aynı veri kümeleri kullanılarak çağ/tur tekrarları yapılır.

yapay sinir ağı üzerindeki hangi sinapsinin hatadan daha fazla sorumlu olduğunu ileri yayılım bize verebiliyor cunku her ağırlık farklı değer farklı

Makine öğrenmesinde hep bir preprocessing işlemi var verilerin birbirine göre özniteliklendirilmesi veya Dummy Variable Trap'den(birbiriyle aynı anlamdaki verilerin aynı anda verilmemesi) bahsettiğimiz **neural networkler bu durumlara karşı bağışıklığa sahip**

Neural Networkler ne kadar fazla öznitelik verirseniz verin bunların arasındaki bağlantıyı bulmak fazladan verdığınız neural networklerin sistemden temizlenmesi veya çıkarılması gereken öznitelikleri çıkarma gibi **işlemleri kendi içlerinde yapabiliyorlar**

**YPA algoritmalarında iki önemli parametre var;**

**Learning rate** = ağırlık güncellemelerinin ne kadar agresif veya yavaş olacağını belirleyen bir parametredir. Bu değer, her adımda ağırlıkların ne kadar değişeceğini kontrol eder. Yüksek bir öğrenme oranı hızlı öğrenmeyi sağlayabilir, ancak aşırıya kaçarsa ağın istikrarını bozabilir. Düşük bir öğrenme oranı ise daha istikrarlı ancak daha yavaş bir öğrenme süreci sağlar. Öğrenme oranı, ağın doğru yönde ilerlemesini ve optimal bir çözüm bulmasını sağlayacak şekilde dengelenmelidir. Bu nedenle, uygun bir öğrenme oranı seçimi, ağın başarısını önemli ölçüde etkileyebilir.

**Epoch** = kaç kere çalışacağı, ağın mevcut parametreleri (ağırlıklar ve biaslar) kullanarak tüm eğitim veri setini işleyip ardından gerçek sonuçlarla karşılaştırarak hata hesaplar ve bu hatayı azaltmaya çalışır. Her epoch, ağın genel performansını iyileştirmeye çalışır. Eğitim süreci boyunca bir veya daha fazla epoch kullanılarak ağın öğrenme yeteneği artar ve hedeflenen görevi daha iyi gerçekleştirebilir hale gelir. Ancak çok fazla epoch kullanmak, aşırı uyuma (overfitting) riskini artırabilir, bu da ağın eğitim verilerine çok fazla uyum sağlayıp genelleme yeteneğini kaybetmesine neden olabilir. Bu nedenle, optimal epoch sayısı genellikle deneme yanılma yöntemiyle belirlenir.