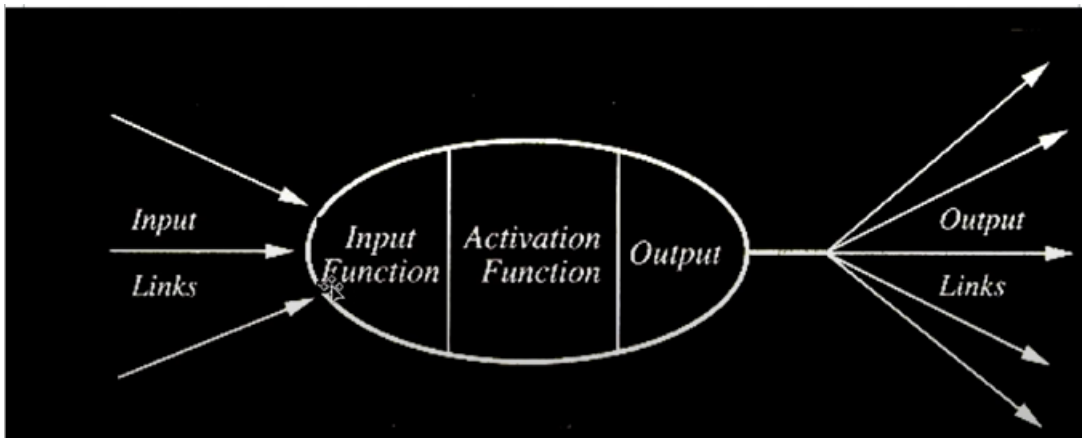
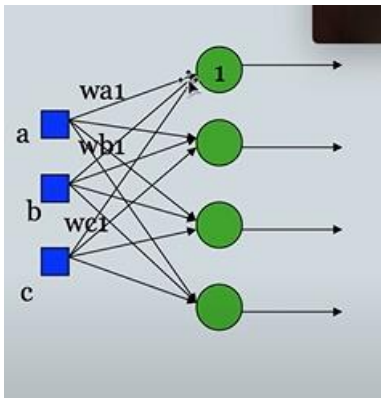


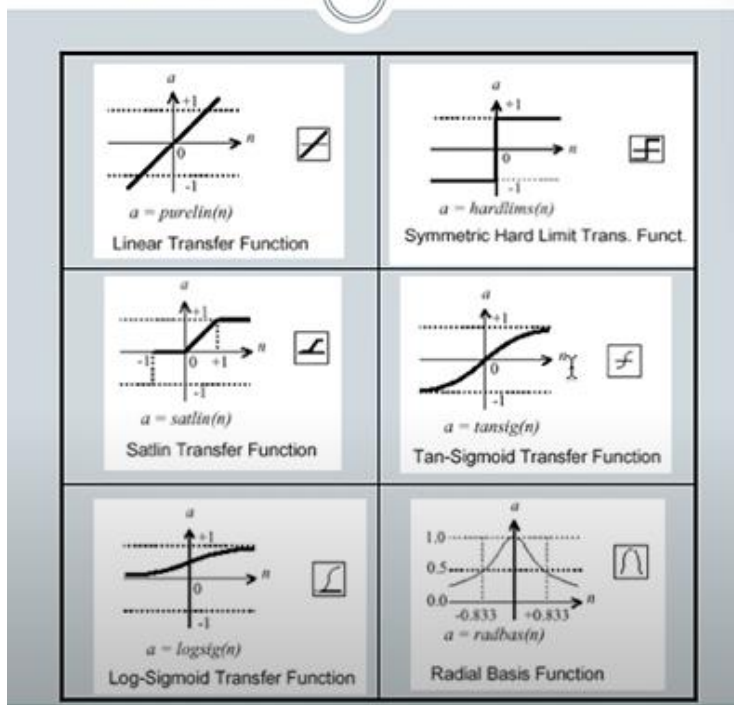
# Neural Networks

A NN is a machine learning approach inspired by the way in which brain performs a particular learning task:

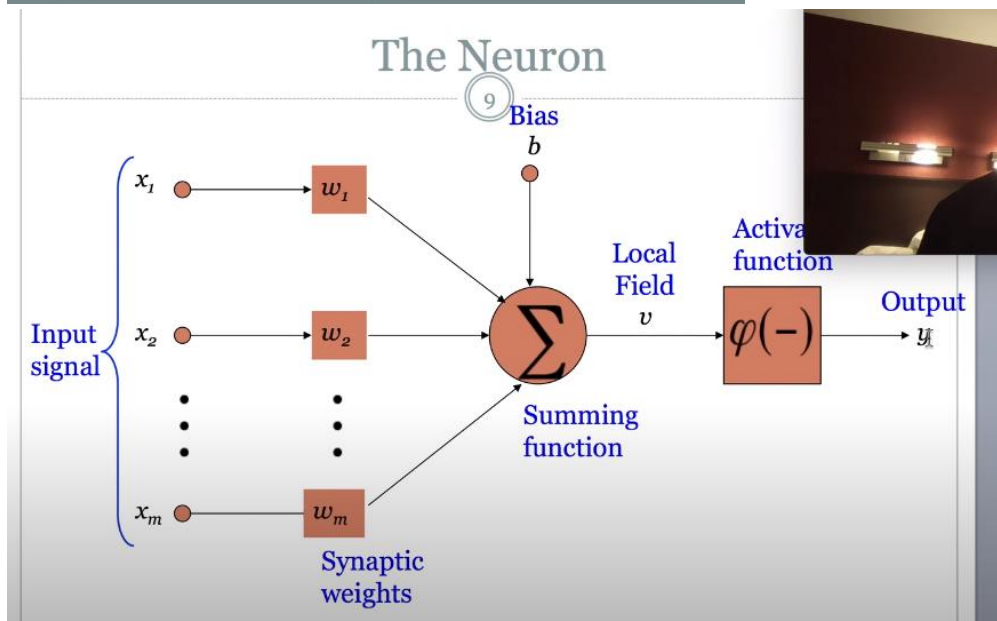
- Knowledge about the learning task is given in the form of examples
- Inter neuron connection strengths(weights) are used to store the acquired information (the training examples)
- During the learning process the weights are modified in order to particular learning task correctly on the training examples.



# Activation Function

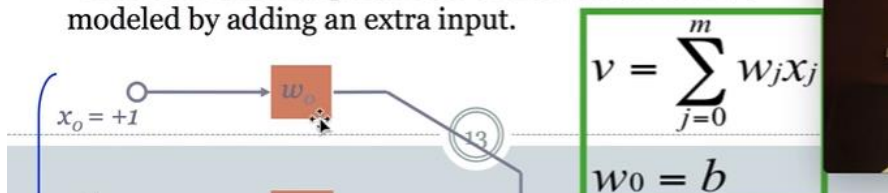


## The Neuron



## Bias as extra input

- Bias is an external parameter of the neuron. Can be modeled by adding an extra input.



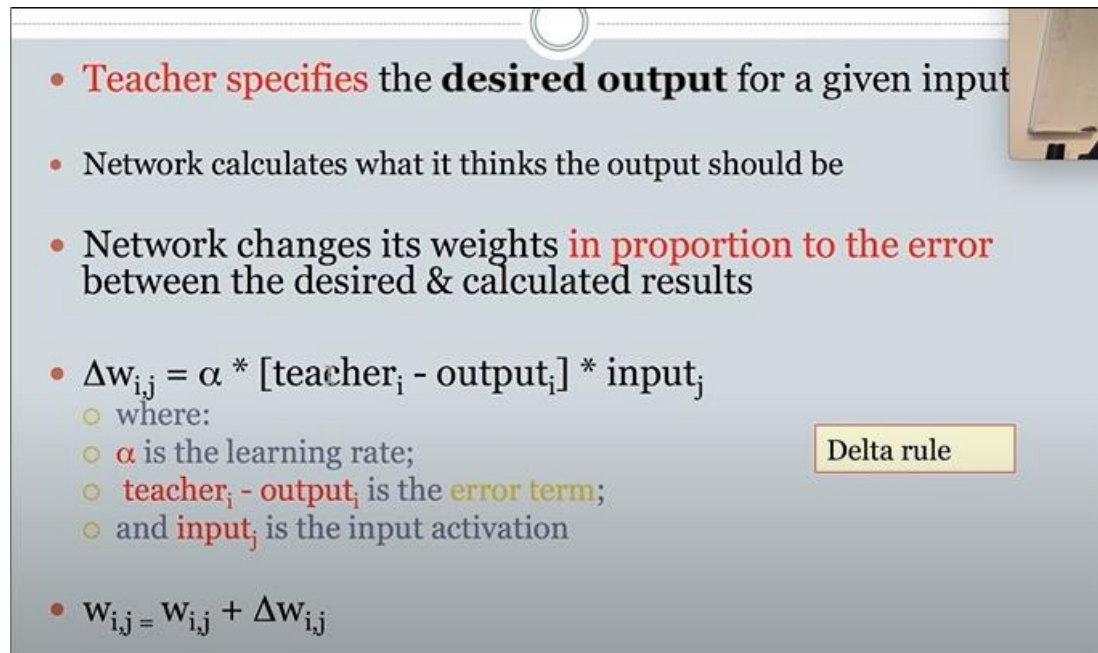
Three different classes of network architectures :

- 1- Single-layer feed-forward
- 2- Multi-layer feed-forward
- 3- Recurrent (feed-backward , cyclic)

The architecture of a neural network is linked with the learning algorithm used to train

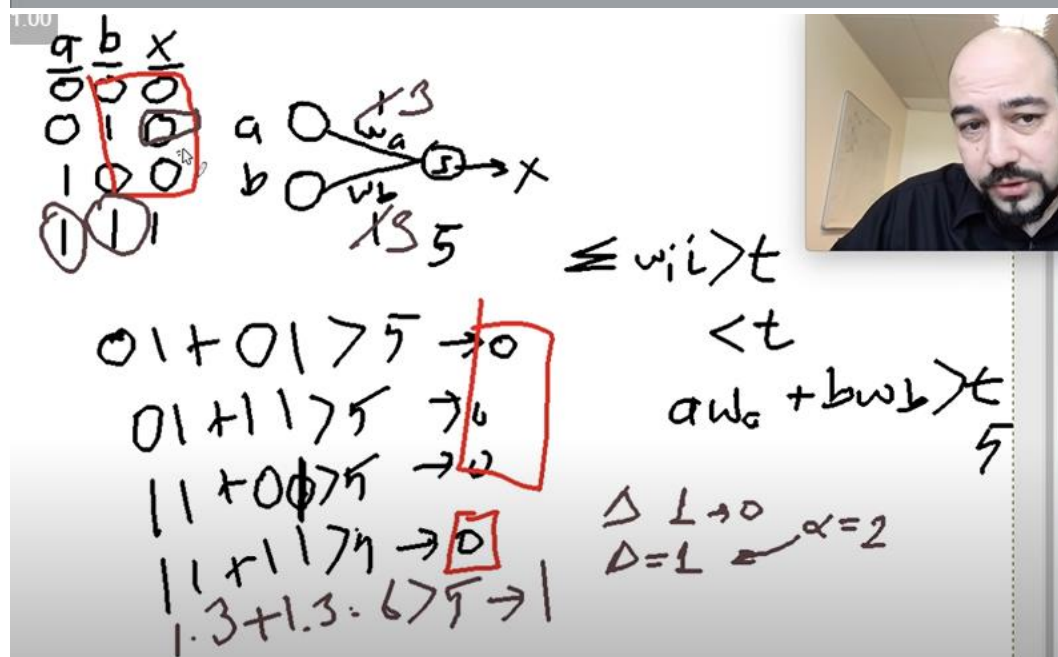
Perceptron : single-layer , there is no inter-layer(hidden layers)

#### Perceptron Learning Rule



- **Teacher specifies** the **desired output** for a given input
- Network calculates what it thinks the output should be
- Network changes its weights **in proportion to the error** between the desired & calculated results
- $\Delta w_{ij} = \alpha * [\text{teacher}_i - \text{output}_i] * \text{input}_j$ 
  - where:
  - $\alpha$  is the learning rate;
  - **teacher<sub>i</sub> - output<sub>i</sub>** is the **error term**;
  - and **input<sub>j</sub>** is the input activation
- $w_{ij} = w_{ij} + \Delta w_{ij}$

Delta rule



Handwritten notes illustrating the perceptron learning rule:

**Diagram:** A perceptron with inputs  $a$  and  $b$ , weights  $w_a$  and  $w_b$ , and a threshold  $t=5$ . The output is  $x$ .

**Truth Table:**

a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

**Calculations:**

For  $a=0, b=1$ :  $0 \cdot w_a + 1 \cdot w_b = 1 < 5 \rightarrow 0$

For  $a=0, b=0$ :  $0 \cdot w_a + 0 \cdot w_b = 0 < 5 \rightarrow 0$

For  $a=1, b=0$ :  $1 \cdot w_a + 0 \cdot w_b = 1 < 5 \rightarrow 0$

For  $a=1, b=1$ :  $1 \cdot w_a + 1 \cdot w_b = 2 < 5 \rightarrow 0$

For  $a=1, b=1$ :  $1 \cdot 3 + 1 \cdot 3 = 6 > 5 \rightarrow 1$

**Weight Updates:**

$\Delta w_a = 0$   
 $\Delta w_b = 1$   
 $\alpha = 2$

And işlemine sokulan iki değeri yapay sinir ağırları ile modellemek istiyoruz. Weight'ler ile input değerleri çarpılıp toplanır ve threshold (eşik değeri) aşılıyorsa aktivasyon

fonksiyonu 1 döndürür. Aşılmazsa 0 döndürür. Görselde sol tarafta sırasıyla değerler denenmiş fakat rastgele belirlediğimiz threshold (5) aşılamadığı için input değerleri 1 ve 1 iken sonucun da 1 çıkması beklenirken 0 gelmekte. Bu nedenle weight değerleri üzerinde güncelleme yaparak modelimizi güçlendirmemiz gerekiyor. Burada ise alpha değerinden faydalaniyor. Delta ise hatayı ifade ediyor. Örneği delta burada aktivasyon fonksiyonu sonucu 1 beklenirken 0 çıktığı için 1. Alpha ise 2 olsun. İki değeri çarpıp eski weight değerlerine eklediğimizde input'un 1-1 olduğu durumda threshold aşıldığı için beklenen değer alınıyor. Ve model güçlendirilmiş oldu.

Alpha değeri öğrenme oranını ifade eder. Öğrenme oranı çok yüksekse ulaşmak istediğimiz değere çok hızlı bir şekilde ulaşip hatta o değeri geçip sonrasında tekrar geri inmesi söz konusu olabilir. Alpha değeri düşük ise daha uzun bir sürede hedefe ulaşılır.

Bu anlatılan birkaç iterasyonda da olabilir. Amaç optimum modele ulaşana kadar bunun yapılmasıdır.

### Adjusting perceptron weights

- $\Delta w_{i,j} = \alpha * [teacher_i - output_i] * input_j$
- miss<sub>i</sub>** is (teacher<sub>i</sub> - output<sub>i</sub>)

	miss<0	miss=0	miss>0
input < 0	alpha	0	-alpha
input = 0	0	0	0
input > 0	-alpha	0	alpha

- Adjust each  $w_{i,j}$  based on  $input_j$  and  $miss_i$
- The above table shows adaptation.
- Incremental learning.

Perceptron'da çıktı sayısı kadar nöron olur , girdiler de ona bağlanır.

### Perceptron ile XOR problemi çözülebilir mi ?

Tek bir perceptron (veya tek katmanlı bir yapay sinir ağı), doğrusal olarak ayrılabilir problemleri çözebilir. XOR problemi ise doğrusal olarak ayrılabilir olmayan bir problemdir, yani iki sınıfı tek bir doğruyla ayırmak mümkün değildir. Bu nedenle, tek bir perceptron kullanarak XOR problemi çözülemez.

Ancak, çok katmanlı bir yapay sinir ağı (Multi-Layer Perceptron, MLP) kullanarak XOR problemi çözülebilir. MLP, en az bir gizli katmana sahip olduğunda, doğrusal olarak ayrılabilen problemleri çözebilir. XOR problemi için tipik olarak iki giriş, bir veya daha fazla gizli katman ve bir çıkış nöronundan oluşan bir ağ yapısı kullanılır.

Her bir gizli katman nöronu farklı kombinasyonları öğrenir ve aktivasyon fonksiyonları (örneğin, ReLU veya sigmoid) sayesinde ağın doğrusal olmayan özellikleri modellemesi mümkün olur. İki giriş ve bir çıkış nöronu olan bir XOR modeli için en azından bir gizli

katman ve genellikle iki gizli nöron kullanılır. Bu yapı, modelin XOR problemine ait doğrusal olmayan karar sınırlarını öğrenmesini sağlar.

### Örnek Python kodu:

```
import numpy as np
from sklearn.neural_network import MLPClassifier

# XOR için giriş ve çıktılar
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 1, 1, 0])

# MLPClassifier ile bir model oluştur
xor_model = MLPClassifier(hidden_layer_sizes=(2,), activation='tanh',
                          max_iter=1000, random_state=42)

# Modeli eğit
xor_model.fit(X, y)

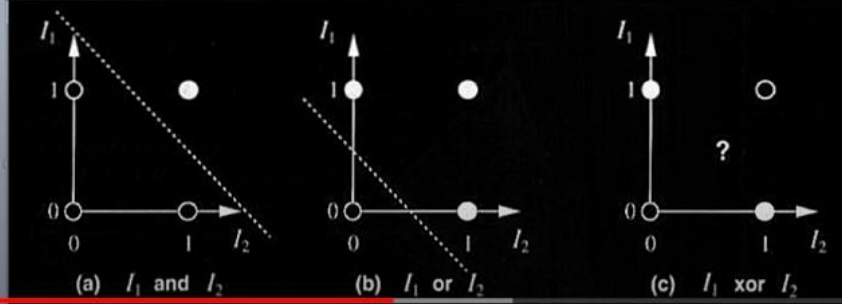
# XOR tahminleri yap
predictions = xor_model.predict(X)
print(predictions) # Çıkış: [0 1 1 0]
```

Bu kod, XOR problemini çözecek şekilde bir MLP sınıflandırıcı oluşturur ve eğitir.

`hidden_layer_sizes=(2,)` parametresi, tek bir gizli katman ve bu katmanda iki nöron olduğunu belirtir. `activation='tanh'` ise aktivasyon fonksiyonu olarak hiperbolik tanjant fonksiyonunun kullanılacağını ifade eder. Model yeterince iterasyonla (burada `max_iter=1000` olarak ayarlanmıştır) eğitildiğinde, XOR işlevini başarıyla öğrenir ve doğru tahminler yapabilir.

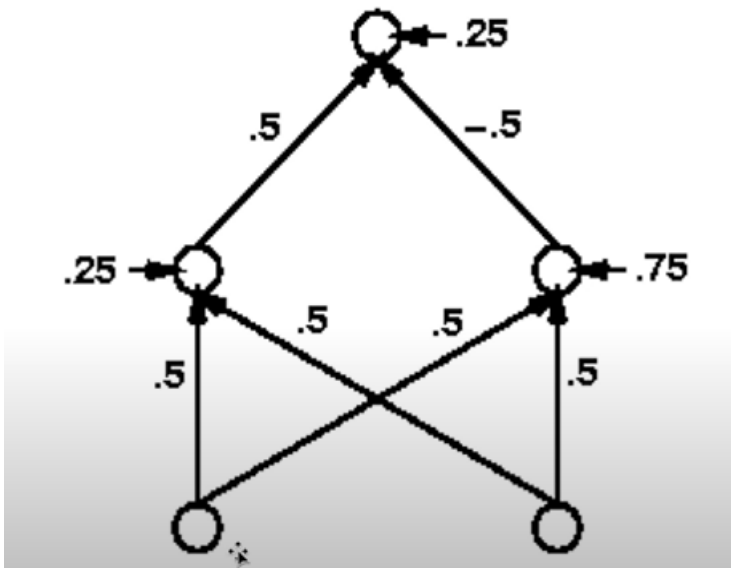
## The Separability Problem and EXOR trouble

Figure 19.9. Linear Separability in Perceptrons



Görselde anlaşıldığı üzere, xor kapısının sonuçlarına göre tek bir doğru ile 1 ve 0 outputlarını bölemiyoruz. Bu nedenle birden çok katman gerekiyor.

### XOR Solution



<https://www.youtube.com/watch?v=aircAruvnKk>  
<https://www.youtube.com/watch?v=IHZwWFHwa-w>  
<https://www.youtube.com/watch?v=llg3gGewQ5U>  
<https://www.youtube.com/watch?v=tIeHLnjs5U8>