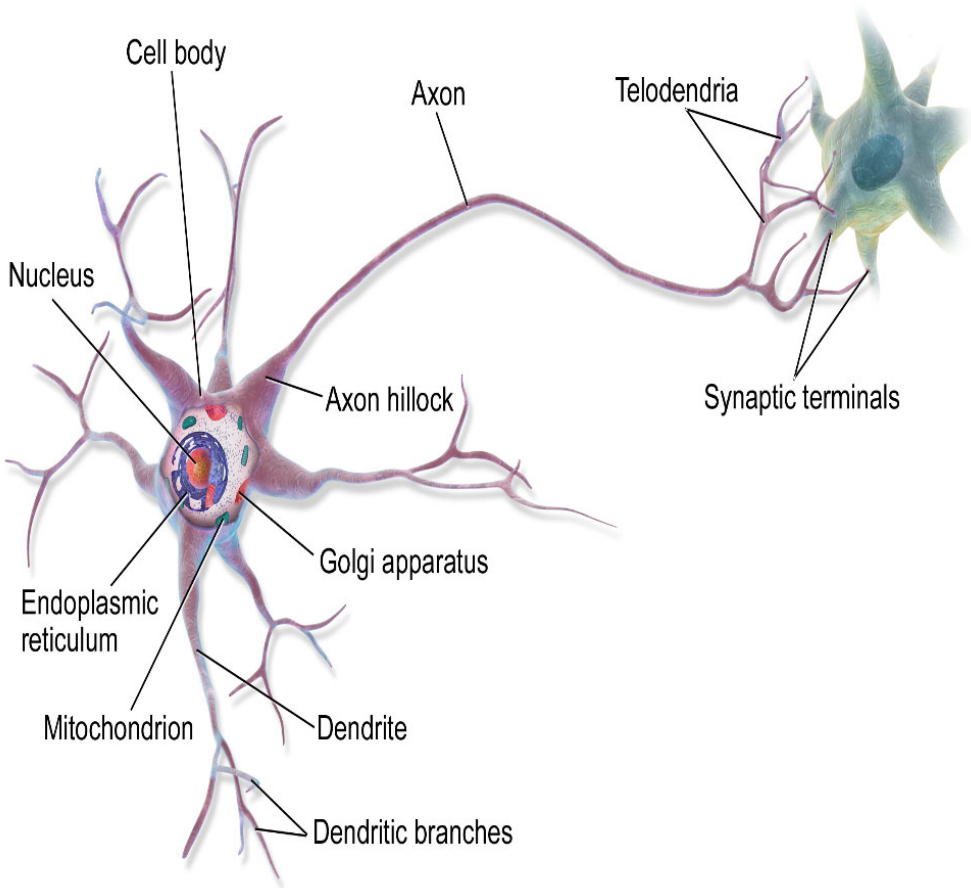
Basit Sinir Ağları ve Nöral Dil Modelleri

Sinir Ağları / Neural Networks

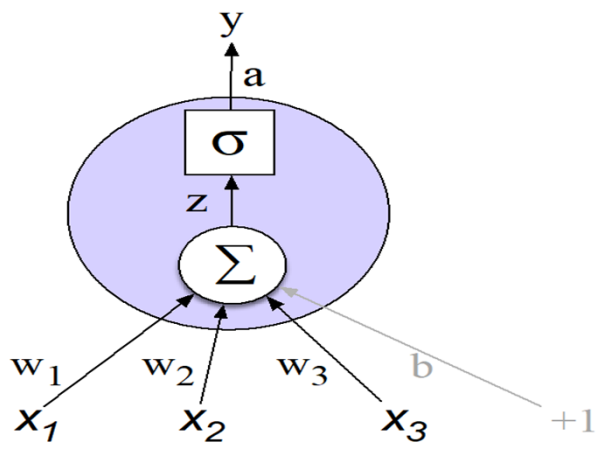
Beynimiz



BruceBlaus
https://commons.wikimedia.org/w/index.php?curid=28761830

- Sinir ağlarını «nöral" olarak adlandırmanın nedeni, 1943 yılında insan nöronunun basitleştirilmiş bir modeli olan ve önermeler mantığıyla tanımlanan *McCulloch-Pitts nöronuna* dayanır.
- Ancak günümüzde dil işlemede modern kullanım artık bu erken biyolojik örneği kullanmıyor. "sinirsel/ nöral " adı artık sadece tarihsel bir örnek olarak kaldı.

Sinir Ağları



- Modern bir sinir ağı, ayrı ayrı bir giriş değeri vektörünü alan ve tek bir çıktı değeri üreten hesaplama birimlerinden oluşan bir ağdır.
- Hesaplama olarak her bir birim katmanından diğerine yinelemeli olarak ilerlenmesi nedeniyle bu ağlar **feed forward network** olarak adlandırılır.
- Modern sinir ağları derin öğrenme gerçekleştirir, çünkü bu ağlar çok sayıda katmana sahiptir.
- Bir sinir ağının yapı taşı ise tek bir hesaplama birimidir.
- Bir birim, girdi olarak bir dizi gerçek değerli sayı alır, bunlar üzerinde bazı hesaplamalar yapar (ağırlıklar ve sapmalar ve doğrusal olmayan bir dönüşümle birleştirerek) ve bir çıktı üretir.

Sinir (Nöral) Ağları

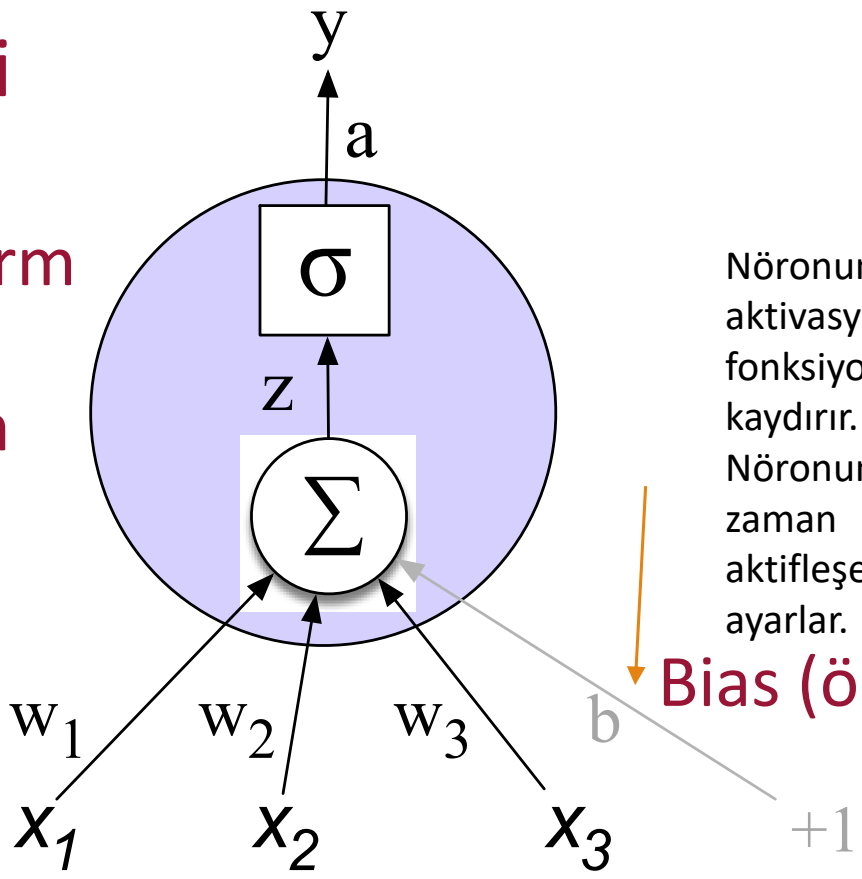
Çıktı Değeri

Non-linear transform

Ağırlıklı Toplam

Ağırlıklar

Giriş Tabakası



Nöronun aktivasyon fonksiyonunu kaydırır. Nöronun ne zaman aktifleşeceğini ayarlar.

Bias (önyargı)

$$z = b + \Sigma\, w_i\, x_i$$

Ağırlıklar ve bias, eğitim sorasında otomatik olarak ayarlanır.

Yapay Sinir Ağlarında Öğrenme Süreci: Ağırlıklar, Girdiler ve Bias

Öğrenme, forward propagation (ileri yönlü yayılım) ya da back propagation (geri yayılım) ile gerçekleşir.

Girişlerin ağırlıklı toplamı ile önyargı (bias) değeri toplanır.

$$z = b + \sum w_i x_i$$

$$z = b + w.x$$

z yerine, lineer olmayan bir aktivasyon fonksiyonu da kullanılabilir. $y = \alpha f(z)$

- Bir sinir birimi, girdilerinin ağırlıklı toplamını alır; bu toplamda önyargı (bias) terimi adı verilen ek bir terim bulunur.
- $x_1 \ldots x_n$ girdileri verildiğinde, bir birimin karşılık gelen ağırlıkları $w_1 \ldots w_n$ ve bir önyargısı (bias) b vardır.
- Ağırlıklı toplam z vektörel olarak ifade edilir.
- Bu ağırlıklı toplam, yani bir ağırlık vektörü w ile bir girdi vektörü x'in çarpımından hesaplanan bir z skaleri önyargı b değeri ile birlikte hesaplanır.
- x'in doğrusal bir fonksiyonu olan z'yi çıktı olarak kullanmak yerine, sinir birimleri z'ye doğrusal olmayan bir fonksiyon f uygular.
- Bu fonksiyonun çıktısına birim için aktivasyon değeri α denir. Tek bir birim modellendiği için, düğümün aktivasyonu aslında ağın çıktısıdır ve bu $y = \alpha f(z)$ olarak adlandırılır.

Lineer olmayan Aktivasyon Fonksiyonları

Lojistik regresyon için sigmoid fonksiyonu 1 bölü 1 artı exp (-z) şeklinde yazılır.

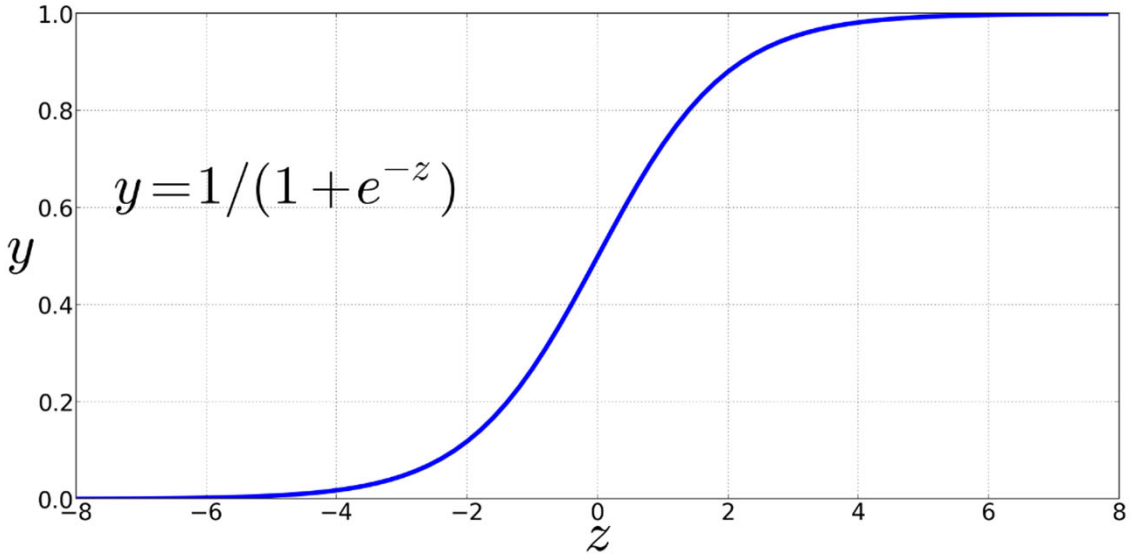
Sigmoid fonksiyonunun avantajı çıktıyı [0,1] aralığına eşleştirmesidir. ,

Bu da aykırı değerleri 0 veya 1'e doğru sıkıştırmada faydalıdır.

Türevlenebilirmesi (differentiable) nedeni ile , öğrenmede kullanışlıdır.

Sigmoid

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



Hesaplamanın Final Fonksiyon Birimi

$$y = s(w \cdot x + b) = \frac{1}{1 + \exp(-(w \cdot x + b))}$$

Sigmoid aktivasyon fonksiyonuna sahip basit bir birimin, x girdisinden w ağırlık vektöründen ve b sapma teriminden hesapladığı fonksiyon.

Final Birimi (tekrar)

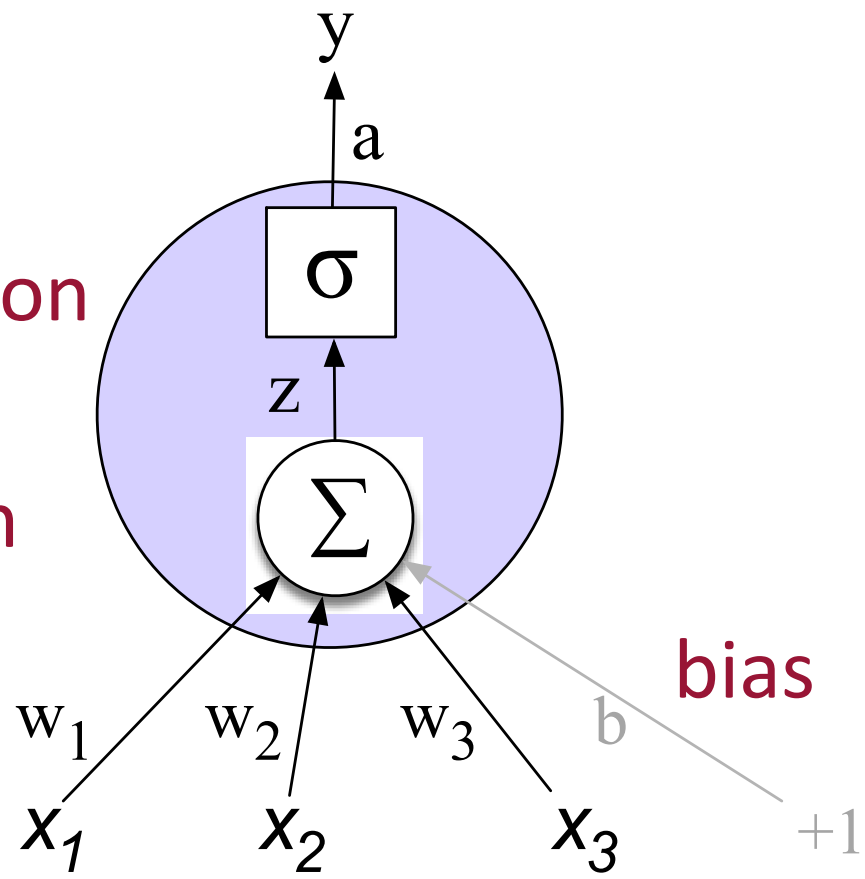
Output value

Non-linear activation function

Weighted sum

Weights

Input layer



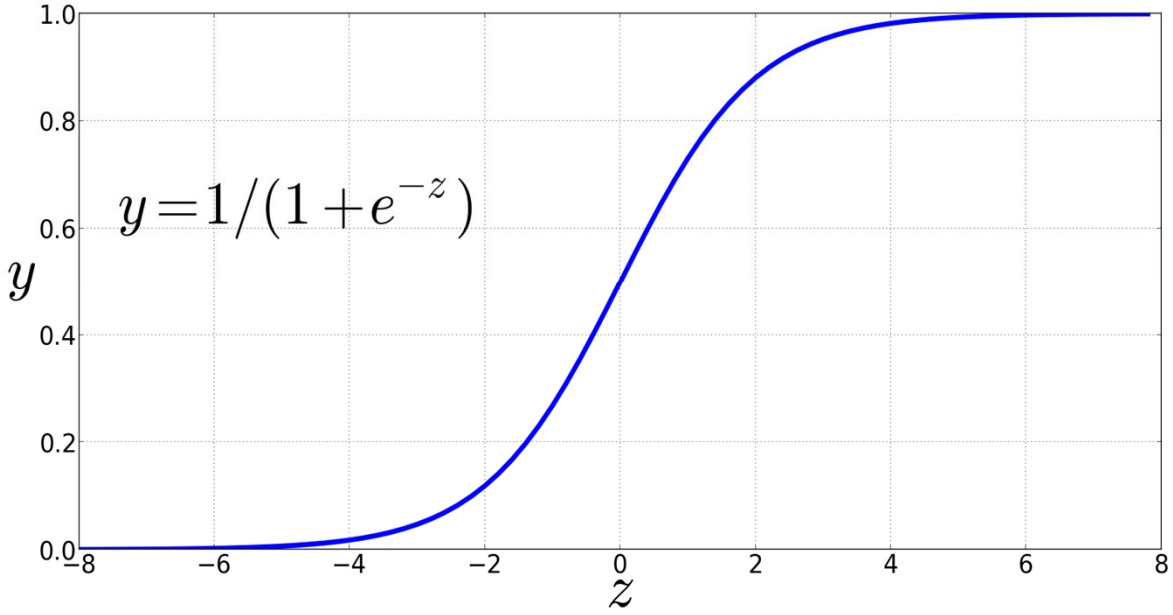
Non-Linear Activation Functions

f olarak seçilen non lineer fonksiyonlardan biri sigmoid fonksiyonu logistic regression için kullanılır.

Sigmoid

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

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$



Çıktıya **aktivasyon** değeri denir. 0 ile 1 arasında normalize edilmiştir.

Sayısal Örnek

w = [0.2, 0.3, 0.9]

b = 0.5

x = [0.5 ,0.6, 0.1]

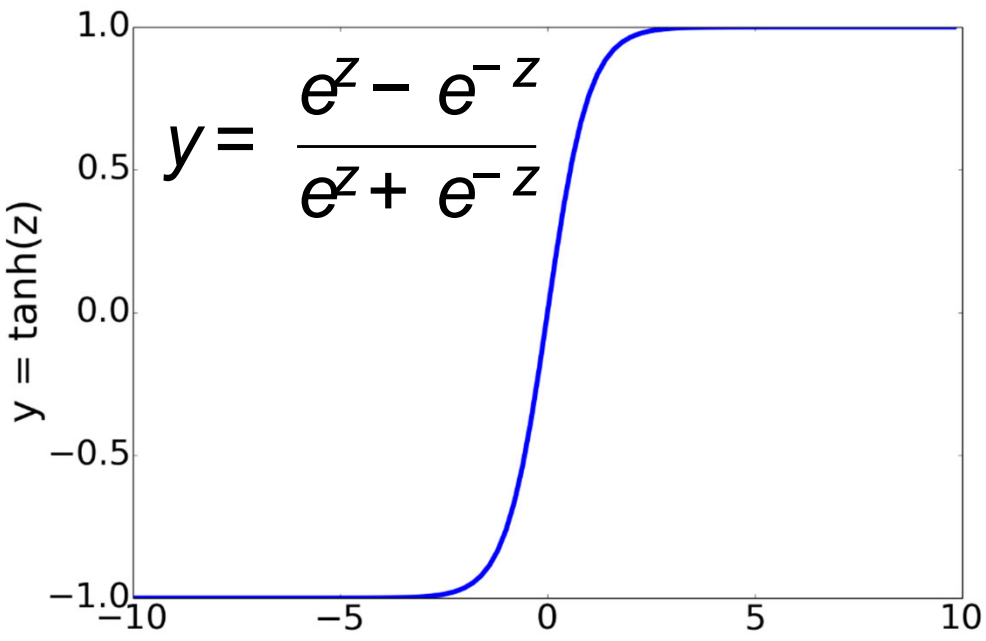
olsun. Sonuç:

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}} = \frac{1}{1 + e^{-(.5*.2+.6*.3+.1*.9+.5)}} = \frac{1}{1 + e^{-0.87}} = .70$$

Non-lineer aktivasyon fonksiyonu olarak Sigmoid fonksiyonu fazla tercih edilmez.

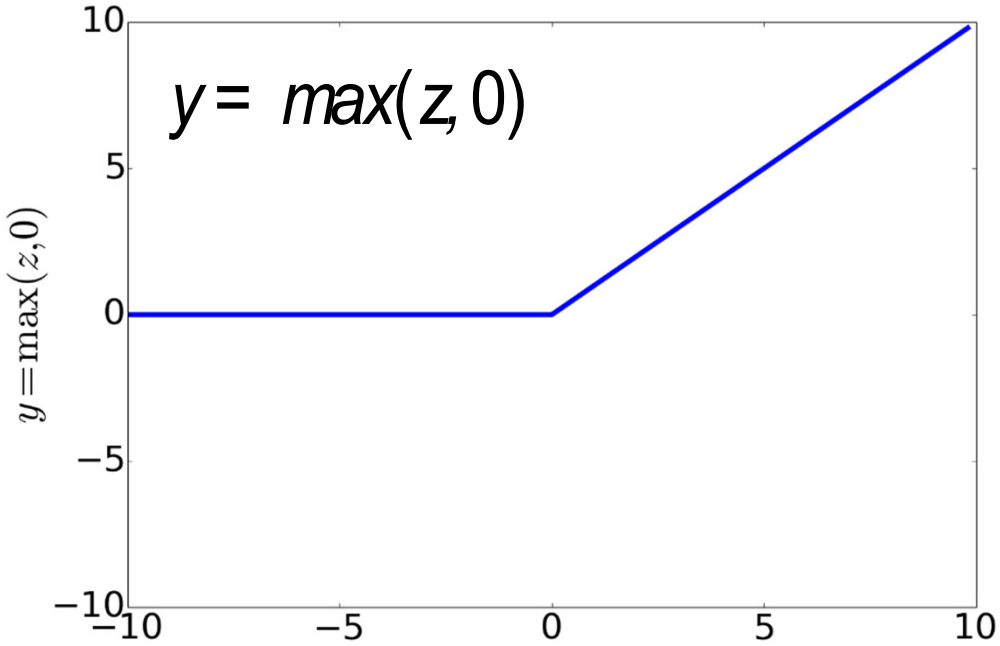
- Sigmoid fonksiyonu aktivasyon fonksiyonu olarak çok yaygın kullanılmamaktadır.
- **Sigmoid** fonksiyonuna çok benzeyen ve neredeyse her uygulama için daha iyi olan fonksiyon **tanh** fonksiyonudur
- tanh, -1 ile +1 arasında değişmesi nedeni ile sigmoid fonksiyonunun bir varyantıdır
- tanh fonksiyonu, düzgün bir şekilde türevlenebilmesi ve aykırı değerleri ortalamaya doğru yönlendirmesi gibi özelliklere sahiptir.

Non -Lineer Aktivasyon Fonksiyonları



tanh

Most Common:



ReLU

Rectified Linear Unit

$$y = \mathrm{ReLU}(z) = max(z, 0)$$

- En basit ve en yaygın kullanılan aktivasyon fonksiyonu, düzeltilmiş doğrusal birim /rectified linear unit (ReLU) fonksiyonudur.
  - z pozitif olduğunda z ile aynıdır, aksi halde 0'dır
- ReLU fonksiyonu, doğrusala çok yakın olması nedeni ile özellikleri öğrenme için uygundur.
- Sigmoid veya tanh fonksiyonlarında, büyük z değerleri 1'e ve türevleri 0'a çok yakın olan y değerleri verir.
- Sıfır türevler öğrenme için sorunlara yol açar: 0 olan gradyanlar, hata sinyalini giderek küçültür; değer eğitim için kullanılamayacak kadar küçülebilir; bu soruna **kaybolan gradyan /vanishing gradient** problemi denir.
- ReLU'nun yüksek z değerleri için türevi 0'a çok yakın değildir, 1'dir.

Basit Sinir Ağları ve Nöral Dil Modelleri

XOR problemi

- Sinir ağlarının gücü, bu birimlerin daha büyük ağlar oluşturmak üzere birleştirilmesindedir.
- Çok katmanlı ağlara duyulan ihtiyacın en önemli örneklerinden biri, Minsky ve Papert'in (1969) tek bir sinir birimi girdisinin bazı çok basit fonksiyonlarını hesaplayamadığına ilişkin o yıllardaki kanıtıdır."

XOR problemi

Minsky and Papert (1969)

Sinir birimleri iki girdisi olan basit fonksiyonları hesaplayabilir mi?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Perceptrons

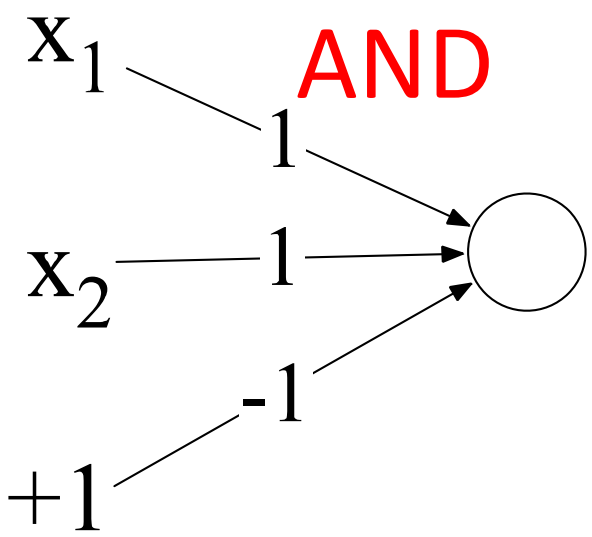
İlk örnek, ikili çıktısı olan ve doğrusal olmayan aktivasyon fonksiyonu da içermeyen en basit bir sinir birimi algılayıcı (perceptron) için gösterilmektedir.

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

Bir algılayıcının çıktısı y, 0 veya 1'dir; aynı ağırlık w, aynı giriş x ve önyargı (bias) b kullanılır.

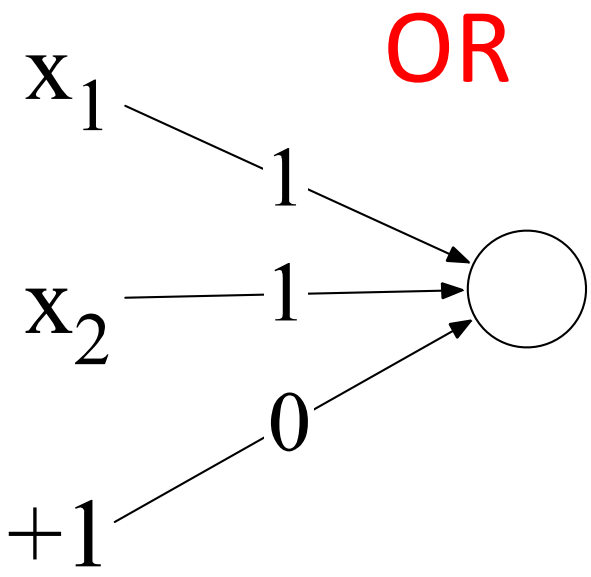
Algılayıcılar için AND - OR işlemleri

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$



AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1



OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Bu ağırlıklar/sapmalar, fonksiyonları uygulayabilecek sonsuz sayıda olası ağırlık ve sapma kümesinden sadece biridir.

Algılayıcılar: Lineer Sınıflandırıcılar

Perceptron denklemi: $w_1x_1 + w_2x_2 + b = 0$

(standart lineer format) $x_2 = (-w_1/w_2)x_1 + (-b/w_2)$

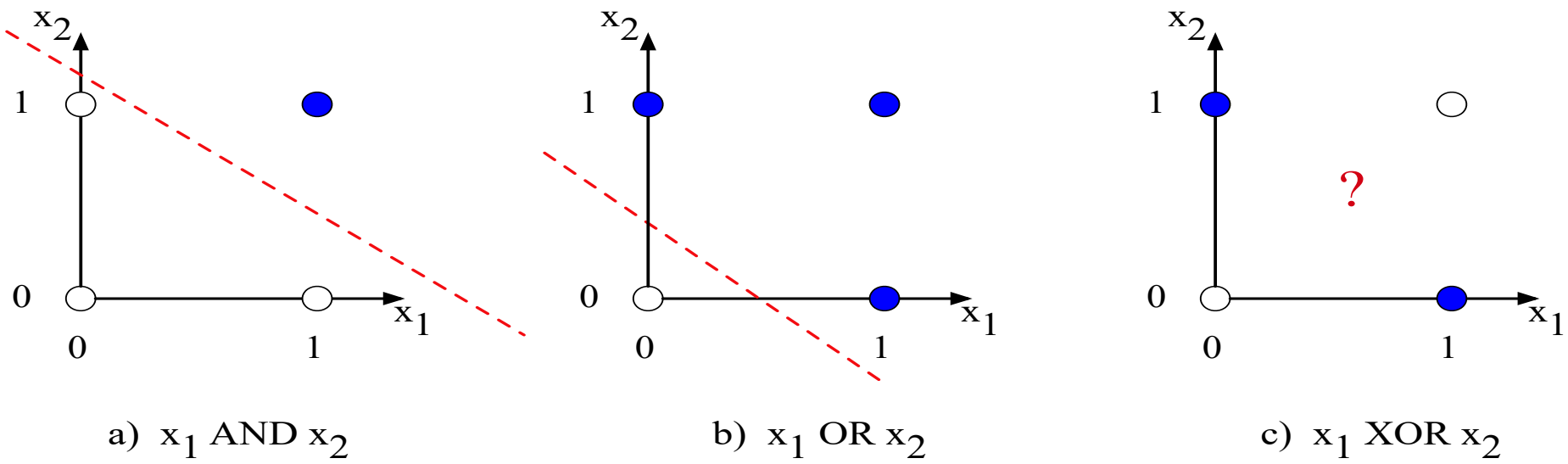
Doğru «decision boundary» olarak davranır.

Çıktı 0 değerindedir; eğer tüm girişler doğrunun bir tarafında ise

çıktı 1 değerindedir; eğer tüm girişler doğrunun diğer tarafında ise

Eğer ikiden daha fazla giriş varsa, «decision boundary» doğru değildir; hiper düzlem olur. Uzay iki kategoriye ayrılacak şekilde aynı mantıkla çalışır.

Decision Boundaries



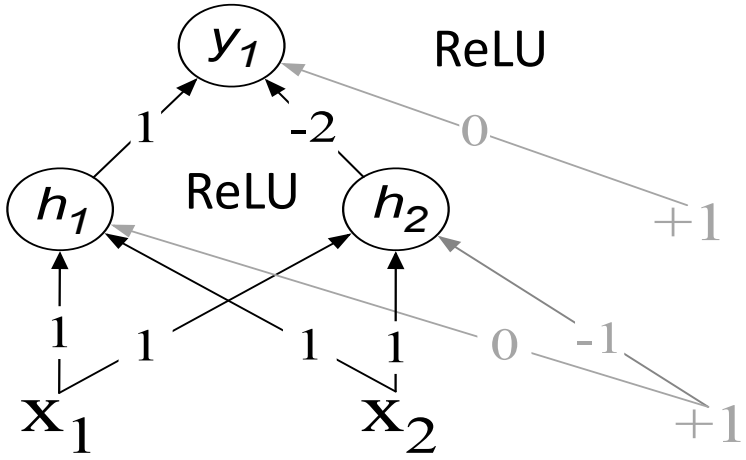
- Olası mantıksal girdileri (00, 01, 10 ve 11) ve bir AND ve bir OR sınıflandırıcı için olası bir parametre kümesi tarafından çizilen doğru göülmektedir.
- XOR'un pozitif durumlarını (01 ve 10) negatif durumlardan (00 ve 11) ayıran bir doğru çizilememektedir.
- XOR, **doğrusal olarak ayrılabilir** bir fonksiyon değildir. Bir eğri veya başka bir fonksiyonla bir sınır çizilebilir; diğer bir ifade ile tek bir doğru olmuyor.

XOR Probleminin Çözümü

XOR, tek bir algılayıcı (perceptron) ile hesaplanamaz. XOR katmanlı ağ birimleri ile hesaplanır.

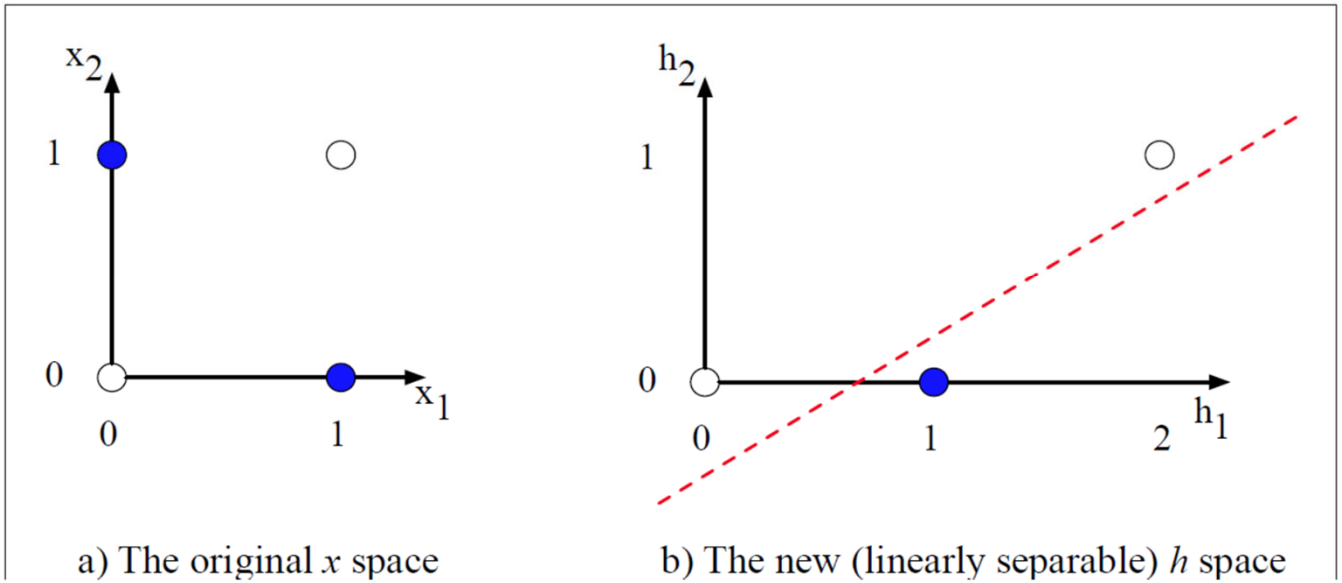
XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

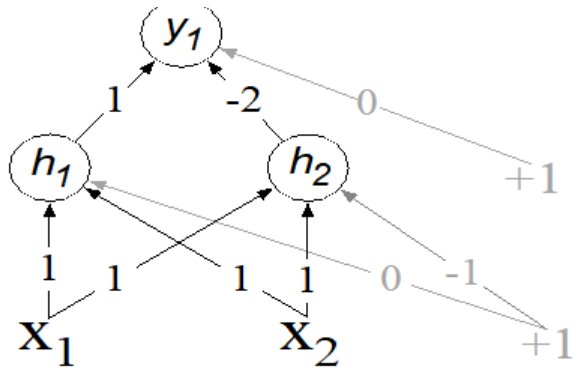


Bu yapının iki ki katmanında üç ReLU birimi vardır: h1, h2 gizli katmanları ve y1 çıktısıdır.
Oklar her birim için w ağırlıklarıdır; b sapması +1 olarak sabitlenmiştir ve bir birimin ağırlığıdır.

Giriş/input x = [0, 0] iken her giriş değeri uygun bir ağırlıkla çarpılarak toplanır ve b sapma (ön yargı) değeri eklenerek [0 -1] vektörü elde edilir. Daha sonra ReLU uygulanarak [0 0] elde edilir ve sonuç değeri 0 olur.
Sonuç olarak w ağırlıkları ve önyargılar (biases) XOR işlemini gerçekleştirir.



a) The original x space b) The new (linearly separable) h space



XOR hesaplamaları sonunda:

[0 1] ve [1 0] giriş değerleri için y değerleri 1;

[0 0] ve [1 1] giriş değerleri için y değerleri 0 olur.

h1 ve h2 (gizli düğüm) çıktılarının eldesi

- İki giriş noktası x = [0 1] ve x = [1 0]'ın XOR çıktısı = 1 'dir ve gizli temsilleri tek bir noktada h = [1 0]'da birleştirilmiştir.
- Bu birleştirme, XOR'un pozitif ve negatif durumlarını doğrusal olarak ayırmayı kolaylaştırır.
  - Başka bir deyişle, ağın gizli katmanıyla giriş iyi bir betimleme oluşturabilir.
  - Bu örnekte sadece ağırlıkları belirtilmiştir.
- Gerçek örnekler için sinir ağlarının ağırlıkları, hata geri yayılım algoritması / error backpropagation algorithm kullanılarak otomatik olarak öğrenilecektir.
  - Buradan da gizli katmanlar uygun simgelenişler oluşturulması öğrenilecektir.

Sonuç olarak:

Sinir ağlarının girişlerin uygun betimlemelerini otomatik olarak öğrenebileceği önyargısı (sezgisi) gösterilmiş olur.

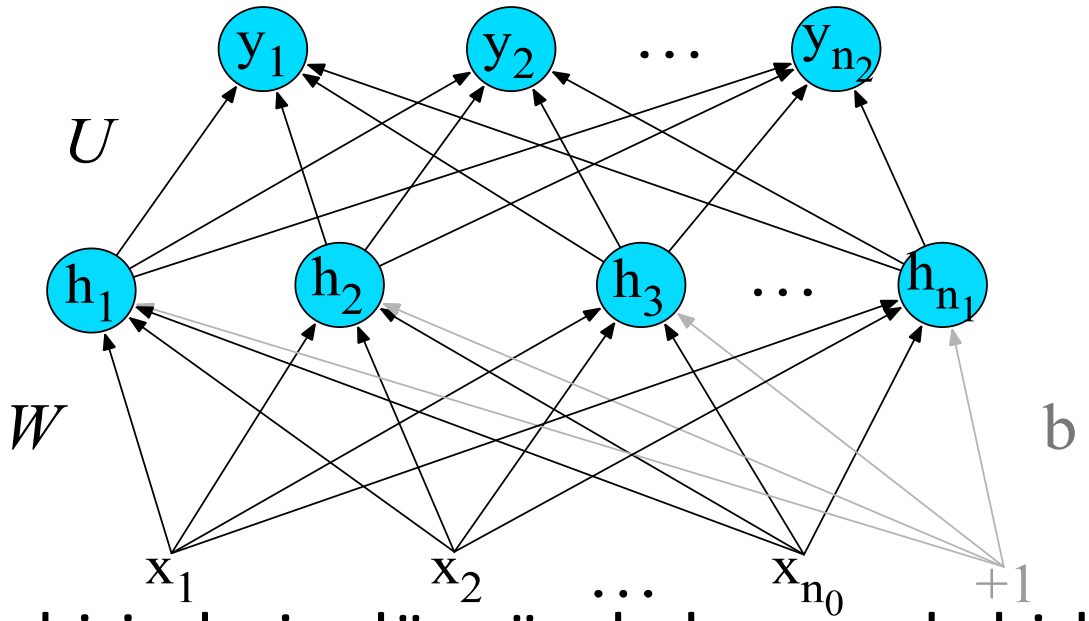
Basit İleri Beslenmeli Sinir Ağları / Feedforward Neural Networks & Nöral (Sinirsel) /Neural Dil Modelleri

- Bu yapı ile basit ileri beslemeli ağ tanımlanır.
- Doğal dil işleme (NLP) uygulamaları çok daha karmaşık ağlar kullanır.
  - Örneğin RNN, Transformer, dikkat mekanizması (attention mechanism?) kullanırken, basit ileri beslemeli ağlar hesaplamalarda önemli bir araç olmayı sürdürür.

<u>Özetle:</u>

- Basit ileri beslemeli ağ, karmaşık sinir ağları tiplerinin temelini oluşturur.

İleri beslemeli Sinir Ağları/Feedforward Neural Networks



İleri beslemeli ağ, birimlerin döngüsel olmayarak birbirlerine bağlı olduğu çok katmanlı bir ağdır; her katmandaki birimlerden gelen çıktılar bir üst katmandaki birimlere iletilir ve alt katmanlara hiçbir çıktı geri iletilmez.

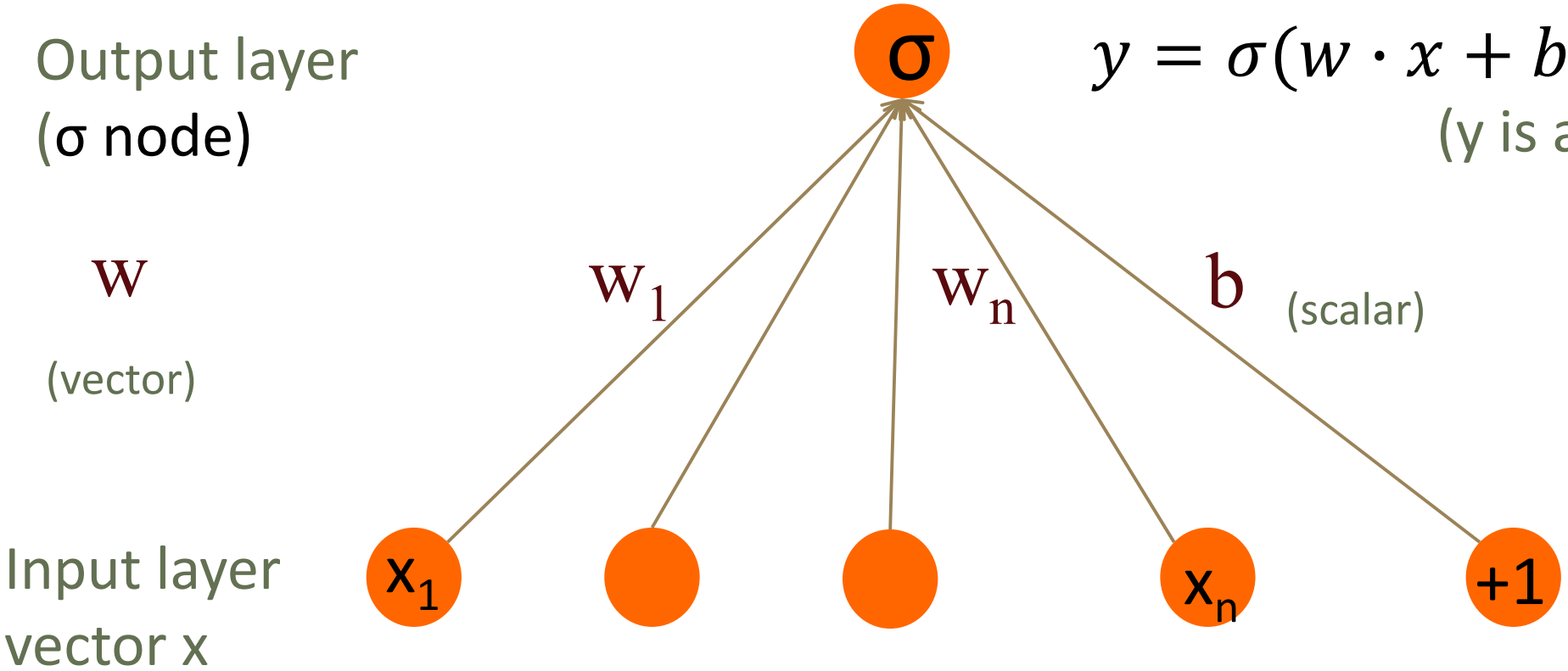
İleri beslemeli ağlara çok katmanlı algılayıcılar /multi-layer perceptrons (MLP) adı verilir.

İkili Lojistik Regresyon (1 düzeyli ağ betimlemesi)

Output layer
(σ node)

$$y = \sigma(w \cdot x + b)$$

(y is a scalar)



Giriş tabakası (input layer) düzey olarak sayılmaz.

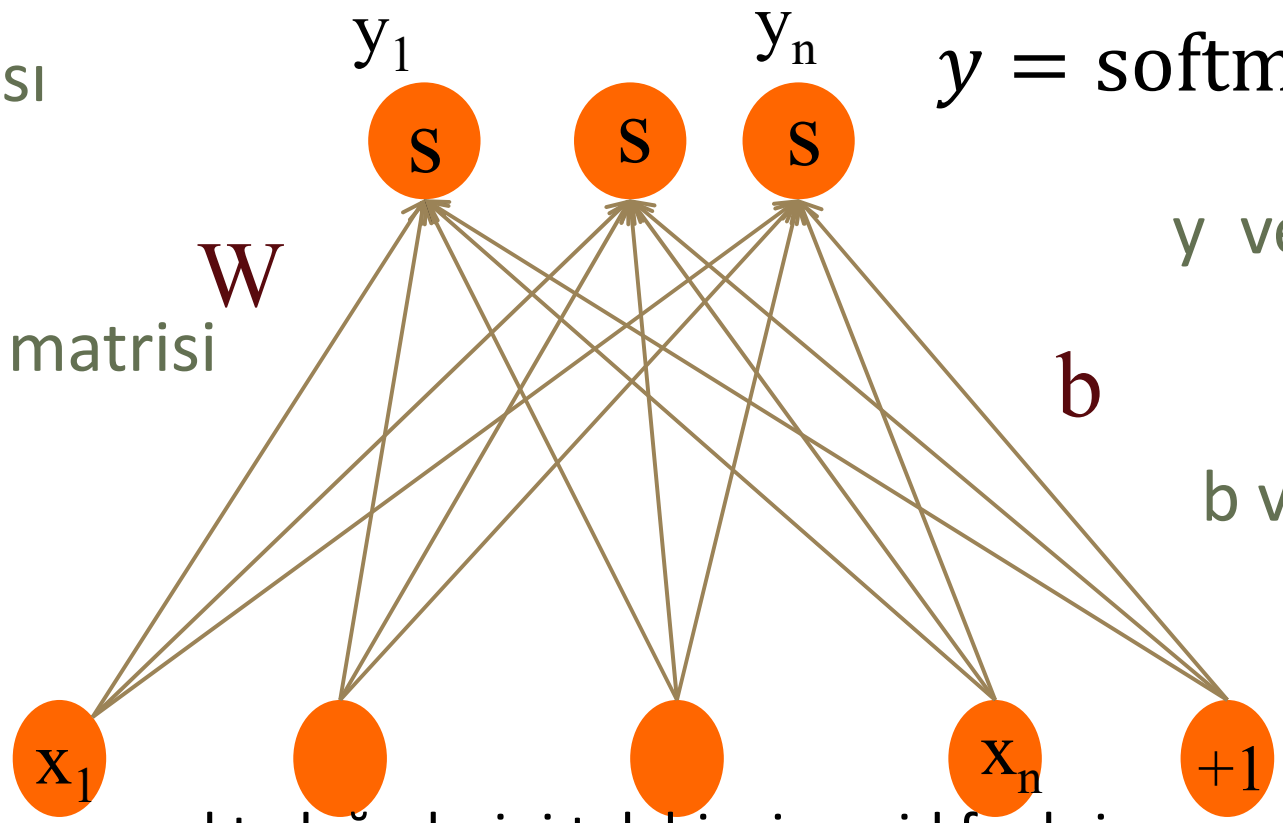
Lojistik Regresyon

- İkili lojistik regresyon tek katmanlı bir ağdır.
- Katmanları sayarken giriş katmanı sayılmaz.
  - x vektörü olarak adlandırılan ve katman 0 olarak giriş katmanı adlandırılır.
  - x vektörü ağırlık matrisi olan w ile çarpılır, doğrusal sapma/önyargı b değeri eklenir.
- 1. katman, çıkış katmanı, w çarpı x + b değerinin sigmoid fonksiyonu olarak bir skaler bir y değeri olarak hesaplanır.

Çoklu Lojistik Regresyon (Tek Katmanlı bir Ağ olarak)

Tümü bağlantılı tek katmanlı bir ağdır.

$$y = \mathrm{softmax}(Wx + b)$$



Çoklu lojistik regresyon, çıktı değerlerini tek bir sigmoid fonksiyonu yerine olasılıklara dönüştürmek için softmax fonksiyonunu kullanır.

Softmax Fonksiyonu: Genelleştirilmiş Sigmoid fonksiyonu

k boyutlu bir z vektörü için softmax aşağıdaki gibi hesaplanır:

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k}\exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k}\exp(z_i)}, \ldots, \frac{\exp(z_k)}{\sum_{i=1}^{k}\exp(z_i)}\right]$$

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k}\exp(z_j)} \quad 1 \le i \le k$$

Örneğin $z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$ vektörü verildiğinde

$\text{softmax}(z) = [0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$ olarak hesaplanır.