

Hafta 06 - Sinir Ağları

**SİB 552 - Siber Güvenlik İçin Veri Madenciliği**

Bilgisayar Mühendisliği

Siber Güvenlik Yüksek Lisans Programı

**Dr. Ferhat Özgür Çatak**  
ozgur.catak@tubitak.gov.tr

Gebze Teknik Üniversitesi  
2019 - Bahar

# İçindekiler

- 1 Neuron Model
  - Tek-Girişli Sinir
  - Aktivasyon Fonksiyonları
  - Sinir Ağı ile Sınıflandırma
  - Çok Sınıflı Sınıflandırma
  - Sinir Ağı Öğrenimi
  - Sinir Ağı Öğrenimi: Sınıflandırma
- 2 Ağ Mimarileri
  - Multiple-Input Neuron
  - Yapay Sinir Ağı Katmanları
  - Multiple Layers of Neurons
- 3 Derin Öğrenme
  - Giriş
  - Autoencoders

# İçindekiler

1

## Neuron Model

- Tek-Girişli Sinir
- Aktivasyon Fonksiyonları
- Sinir Ağı ile Sınıflandırma
- Çok Sınıflı Sınıflandırma
- Sinir Ağı Öğrenimi
- Sinir Ağı Öğrenimi: Sınıflandırma

- Multiple-Input Neuron

2

## Ağ Mimarileri

- Yapay Sinir Ağı Katmanları
- Multiple Layers of Neurons

3

## Derin Öğrenme

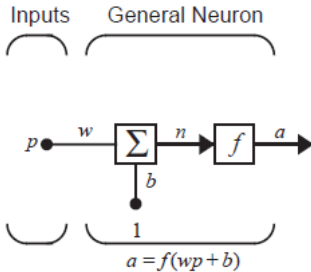
- Giriş
- Autoencoders

# Tek-Girişli Sinir

## Single-Input Neuron

### Single-Input Neuron

- ▶ Girdi  $p \in \mathbb{R}$ , bir ağırlık  $w \in \mathbb{R}$  ile çarpılarak  $wp$  formatına dönüştürülür.
- ▶ Başka bir girdi 1, *bias*  $b$  ile çarpılarak bir toplayıcıya verilir.
- ▶ Bu toplam  $n$  bir aktivasyon fonksiyonuna  $f$  verilerek sinir çıktısı  $a$  elde edilir.



### Neuron Output

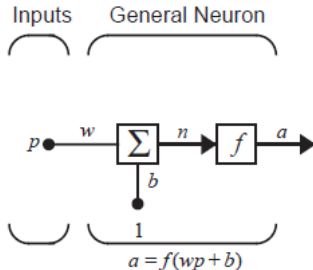
- ▶ Neuron çıktısı

$$a = f(wp + b)$$

Şekil: Single-Input Neuron

## Single-Input Neuron

- Girdi  $p \in \mathbb{R}$ , bir ağırlık  $w \in \mathbb{R}$  ile çarpılarak  $wp$  formatına dönüştürülür.
- Başka bir girdi  $1$ , *bias*  $b$  ile çarpılarak bir toplayıcıya verilir.
- Bu toplam  $n$  bir aktivasyon fonksiyonuna  $f$  verilerek sinir çıktısı  $a$  elde edilir.



- ▶  $w = 3, p = 2, b = -1.5$
- ▶  $a = f(3 \times 2 - 1.5) = f(4.5)$
- ▶ Gerçek sonuç aktivasyon fonksiyonuna  $f$  bağlı olarak değişiklik gösterecektir.

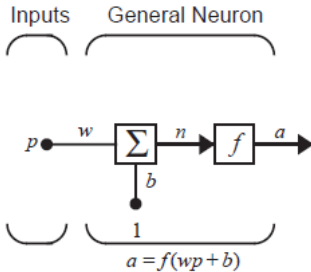
**Şekil:** Single-Input Neuron

# Tek-Girişli Sinir

## Single-Input Neuron

### Single-Input Neuron

- ▶ Girdi  $p \in \mathbb{R}$ , bir ağırlık  $w \in \mathbb{R}$  ile çarpılarak  $wp$  formatına dönüştürülür.
- ▶ Başka bir girdi 1, *bias*  $b$  ile çarpılarak bir toplayıcıya verilir.
- ▶ Bu toplam  $n$  bir aktivasyon fonksiyonuna  $f$  verilerek sinir çıktısı  $a$  elde edilir.



Şekil: Single-Input Neuron

### Neuron Output

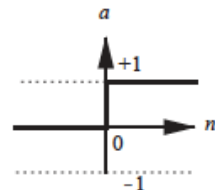
- ▶ Gerçek sonuç aktivasyon fonksiyonuna  $f$  bağlı olarak değişiklik gösterecektir.
- ▶  $w$  ve  $b$  parametreleri öğrenme kuralları uygulanarak bulunacaktır

# Aktivasyon Fonksiyonları I

## Activation Functions

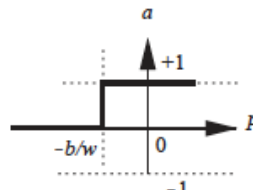
### Hard Limit

- Fonksiyon parametresinin belirli bir değerden az olması durumunda 0, diğer durumlarda 1 olacaktır.



$$a = \text{hardlim}(n)$$

Hard Limit Transfer Function



$$a = \text{hardlim}(wp + b)$$

Single-Input *hardlim* Neuron

Şekil: Hard Limit Aktivasyon Fonksiyonu

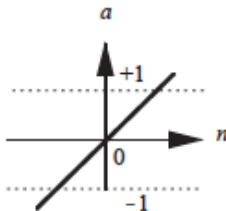
# Aktivasyon Fonksiyonları II

## Activation Functions

### Doğrusal (linear)

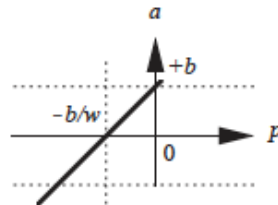
- Çıktı değeri, girdi değeri olan fonksiyondur.

$$a = n$$



$$a = \text{purelin}(n)$$

Linear Transfer Function



$$a = \text{purelin}(wp + b)$$

Single-Input *purelin* Neuron

Şekil: Doğrusal Aktivasyon Fonksiyonu

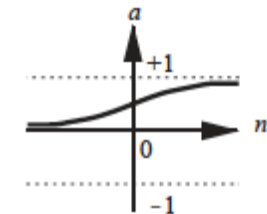


# Aktivasyon Fonksiyonları III

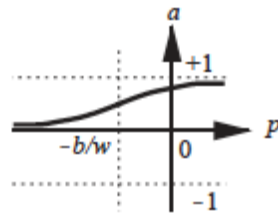
## Activation Functions

### Sigmoid

- Çıktı değeri  $[0, 1]$  aralığındadır.  $\frac{1}{1+e^{-n}}$



$$a = \text{logsig}(n)$$



$$a = \text{logsig}(wp + b)$$

Şekil: Sigmoid Aktivasyon Fonksiyonu

# Aktivasyon Fonksiyonları IV

## Activation Functions

### Diğer Aktivasyon Fonksiyonları

- ▶ Exponential Linear Unit (ELU)

$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (1)$$

- ▶ Rectified Linear Unit (ReLU)

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2)$$

- ▶ TanH

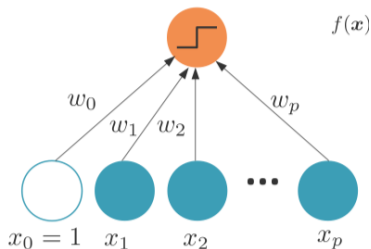
$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (3)$$

- ▶ Softplus

$$f(x) = \log_e(1 + e^x) \quad (4)$$

- ▶ Diğerleri için: <https://keras.io/activations/>

# Sinir Ağı ile Sınıflandırma

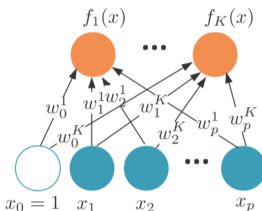


$$h(\mathbf{x}) = f\left(w_0 + \sum_{i=1}^p w_i x_i\right) = f(\mathbf{w}^T \mathbf{x}) \quad (5)$$

# Lab-1

## Çok Sınıflı Sınıflandırma

### Multiclass classification



- Modelin sınıf ataması  $C_k$

$$f_k(\mathbf{x}) = \max_{l \in \{1, \dots, K\}} f_l(\mathbf{x})$$

- Sınıf etiket olasılıkları bulunması için **softmax** kullanılır.

$$f_k(\mathbf{x}) = \frac{e^{o_k}}{\sum_{l=1}^K e^{o_l}} \quad (6)$$

$$O_k = \mathbf{W}^{(k)} \cdot \mathbf{x} \quad (7)$$

- Herhangibir sınıfa ait olan değer diğerlerinden yüksek ise, bu sınıfın softmax değeri 1'e yaklaşacaktır.

```
>>> softmax([1,2])
[0.26894142, 0.73105858])
>>> softmax([10,20])
[0.0000453978687, 0.999954602])
```

```
>>> std_norm([1,2])
[0.3333333333333333, 0.6666666666666666]
>>> std_norm([10,20])
[0.3333333333333333, 0.6666666666666666]
```

# Sinir Ağı Öğrenimi

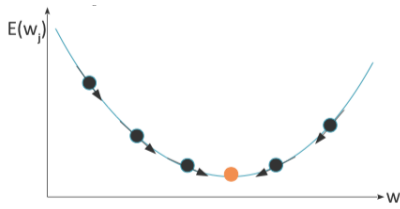
## Sinir Ağı Learning

### Genel Güncelleme Kuralı

$$\Delta w_j = -\eta \frac{\partial \text{Error}(f(\mathbf{x}^{(i)}, y_i))}{\partial w_j} \quad (8)$$

- Her bir eğitim örneğinden sonra, her bir ağırlık:

$$w_j^{(t+1)} = w_j^{(t)} + \Delta w_j^{(t)}$$



## Sinir Ağı Öğrenimi: Sınıflandırma

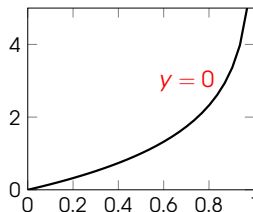
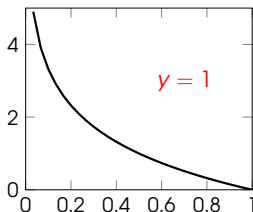
- **Sigmoid** çıktı:

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

$$\sigma(u) = \frac{1}{1 + \exp(-u)}$$

- Kayıp: **Cross-entropy error**

$$\ell \left( f(\mathbf{x}^{(l)}), y^{(l)} \right) = -y^{(l)} \log f(\mathbf{x}^{(l)}) - (1 - y^{(l)}) \log (1 - f(\mathbf{x}^{(l)})) \quad (9)$$



- ▶ İkili sınıflandırma için güncelleme kuralı

$$\Delta w_j = -\eta \frac{\partial \text{Error}(f(\mathbf{x}^{(i)}), y^{(i)})}{\partial w_j} \Rightarrow \Delta w_j = \eta (y^{(i)} - f(\mathbf{x}^{(i)})) x_j^i \quad (10)$$

# Sinir Ağı Öğrenimi: Sınıflandırma

- Güncelleme kuralı

$$\Delta w_j \leftarrow \eta \left( y^{(i)} - f(\mathbf{x}^{(i)}) \right) x_j^{(i)}$$

Update = Learning Rate  $\times$  (Actual - Predicted)  $\times$  Input

- Her bir eğitim örneği, her bir ağırlık için:

$$w_j^{(t+1)} = w_j^{(t)} + \Delta w_j^{(t)}$$



# Lab-2

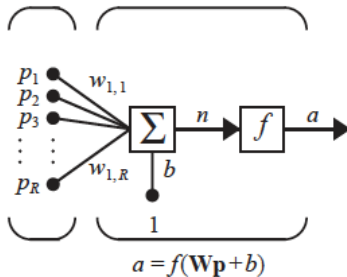
# Çok Girişli Nöron

## Multiple-Input Neuron

### Multiple-Input Neuron

- ▶  $R$  adet girdisi olan nöron
- ▶ Herbir girdi değeri  $p_1, \dots, p_R$ , bir ağırlık değeri  $w_1, \dots, w_R$  ile çarpılmaktadır.

#### Inputs Multiple-Input Neuron



### Ağırlık Matrisi (Weight Matrix)

- ▶  $n = \mathbf{W}\mathbf{x} + b$
- ▶ Single neuron: **ağırlık matrisi tek satırdan oluşur.** Vektör
- ▶  $\mathbf{W} \in \mathbb{R}^p$
- ▶  $a = f(\mathbf{W}\mathbf{x} + b)$

Şekil: Multiple-Input Neuron

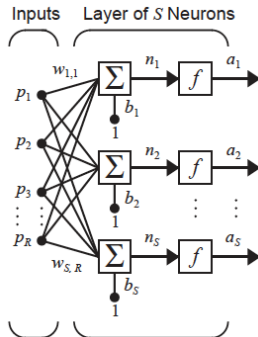
# İçindekiler

- 1 Neuron Model
  - Tek-Girişli Sinir
  - Aktivasyon Fonksiyonları
  - Sinir Ağı ile Sınıflandırma
  - Çok Sınıflı Sınıflandırma
  - Sinir Ağı Öğrenimi
  - Sinir Ağı Öğrenimi: Sınıflandırma
- 2 Ağ Mimarileri
  - Multiple-Input Neuron
  - Yapay Sinir Ağı Katmanları
  - Multiple Layers of Neurons
- 3 Derin Öğrenme
  - Giriş
  - Autoencoders

# Yapay Sinir Ağı Katmanları

## Katman (Layer)

- Tek katmanlı  $S$  adet sinir hücresi içeren ağ modeli
- Ağırlık matrisi  $\mathbf{W} \in \mathbb{R}^{S \times R}$



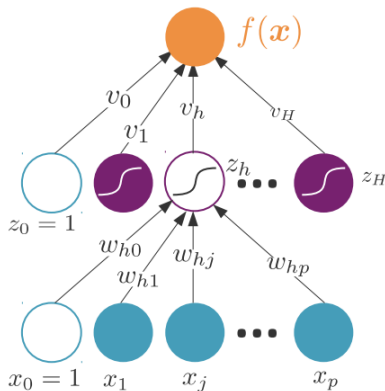
►  $\mathbf{a} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$

► **Ağırlık Matrisi**

$$\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1R} \\ W_{21} & W_{22} & \cdots & W_{2R} \\ \vdots & \vdots & \ddots & \vdots \\ W_{S1} & W_{S2} & \cdots & W_{SR} \end{bmatrix}$$

**Şekil:** Tek katmanlı sinir ağı (single layer neural network)

# Multiple Layers of Neurons I



- hidden unit  $h$ :

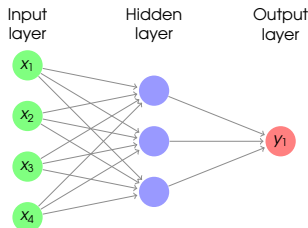
$$z_h = \frac{1}{1 + e^{-\mathbf{w}_h^T \mathbf{x}}}$$

- Ağın çıktı değeri

$$f(\mathbf{x}) = \mathbf{v}^T \mathbf{z}$$

$$= v_0 + \sum_{h=1}^H \frac{v_h}{1 + e^{-\mathbf{w}_h^T \mathbf{x}}}$$

# Multiple Layers of Neurons II



- İlk gizli katmanda bulunan ilk gizli birim ( the first hidden unit in the first hidden layer)

$$z_1^{[1]} = \mathbf{W}_1^{[1]T} \mathbf{x} + b_1^{[1]} \text{ ve } a_1^{[1]} = g(z_1^{[1]})$$

- **W**:ağırlık matrisi, **W**<sub>1</sub>: ağırlık matrisinin ilk satırı

- 
- $\mathbf{W}_1^{[1]} \in \mathbb{R}^3$  ve  $b_1^{[1]} \in \mathbb{R}$
  - Birinci gizli katmanda bulunan ikinci ve üçüncü gizli birimler

$$z_2^{[1]} = \mathbf{W}_2^{[1]T} \mathbf{x} + b_2^{[1]} \text{ ve } a_2^{[1]} = g(z_2^{[1]})$$

$$z_3^{[1]} = \mathbf{W}_3^{[1]T} \mathbf{x} + b_3^{[1]} \text{ ve } a_3^{[1]} = g(z_3^{[1]})$$

- Çıktı katmanı

$$z_1^{[2]} = \mathbf{W}_1^{[2]T} \mathbf{a}^{[1]} + b_1^{[2]} \text{ ve } a_1^{[2]} = g(z_1^{[2]})$$

- $\mathbf{a}^{[1]} = [a_1^{[1]} a_2^{[1]} a_3^{[1]}]$

# Lab-3

# İçindekiler

- 1 Neuron Model
  - Tek-Girişli Sinir
  - Aktivasyon Fonksiyonları
  - Sinir Ağı ile Sınıflandırma
  - Çok Sınıflı Sınıflandırma
  - Sinir Ağı Öğrenimi
  - Sinir Ağı Öğrenimi: Sınıflandırma
- 2 Ağ Mimarileri
  - Multiple-Input Neuron
  - Yapay Sinir Ağı Katmanları
  - Multiple Layers of Neurons
- 3 Derin Öğrenme
  - Giriş
  - Autoencoders



# Derin Öğrenme

## Deep Learning

### Derin Öğrenme

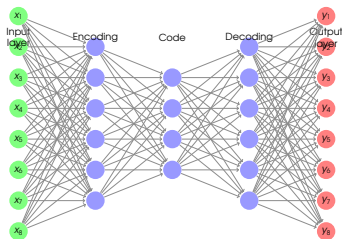
- ▶ Katman sayısının **fazla** olduğu, çok katmanlı sinir ağı modeli
- ▶ Oldukça karmaşık ağ oluşturulabilen parametrik bir model
- ▶ Ağırlıklar gradient descent ile öğrenilmektedir.
  - ▶ Lokal minimum problemi
- ▶ Her bir katman, verinin yeni bir gösterimini (**new representation**) öğrenmektedir.
  - ▶ **representation learning**

# Derin Öğrenme Tipleri

## (Derin) sinir ağlarının türleri

- ▶ Deep feed-forward (= multilayer perceptrons)
- ▶ Unsupervised networks
  - ▶ autoencoders / variational autoencoders (VAE) — learn a new representation of the data
  - ▶ deep belief networks (DBNs)
  - ▶ generative adversarial networks (GANs)
- ▶ Convolutional neural networks (CNNs)
  - ▶ image/audio modeling
- ▶ Recurrent Neural Networks (RNNs)
  - ▶ nodes are fed information from the previous layer and also from themselves
  - ▶ long short-term memory networks (LSTM) for sequence modeling.

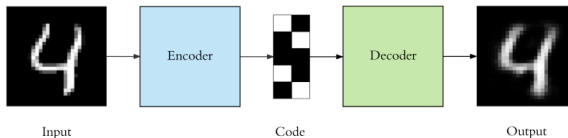
# Autoencoders I



► **Hidden layers:** Compact representation of input

►  $h(\mathbf{x}) \approx \mathbf{x}$

► Reconstruction error:  
 $\ell(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2$



# Autoencoders II

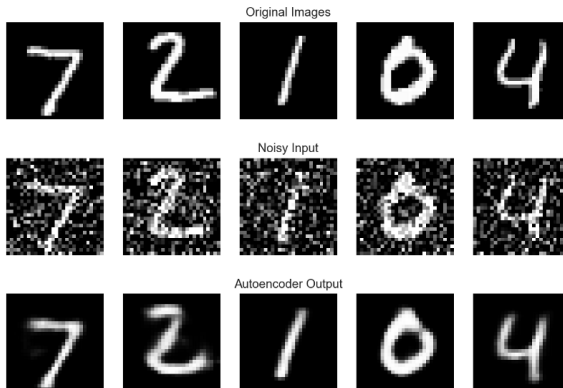
## Autoencoders Algorithm

- ▶ **Data-specific Compression:** Benzer veriyi sıkıştırmak için uygundur.
- ▶ **Lossy:** Giriş ile tam olarak aynı olmayacak, yakın ancak bozulmuş bir temsil (representation) olacaktır.
- ▶ **Unsupervised:** Veri kümesinden aynı veriyi üretmektedir.  
*self-supervised*

# Autoencoders III

## Denoising Autoencoders

- **Amaç:** Intelligent representation of the data
- Gürültü içeren veriden gürültünün temizlenmesi



# Autoencoders IV

## Sparse Autoencoders

- ▶ **Regularization**: Regularize the autoencoder by using a sparsity constraint
- ▶ Only a fraction of the nodes would have nonzero values: called **active nodes**

# Lab-4