



Yapay Sinir Ağları

Sınıflandırma Teknikleri

- Uzaklık Temelli (Nearest-Neighbor) yöntemler
- Karar Ağacı(Decision Tree) yöntemleri
- Olasılık temelli yöntemler (Naïve Bayes)
- Support Vector Machines
- **Yapay Sinir Ağları (Artificial Neural Networks)**
- Ensemble Yöntemler (Bagging - Boosting)
 - Rastgele Ormanlar (Random Forest)
 - AdaBoost ve XGBoost(eXtreme Gradient Boosting)

Artificial Neural Networks/Deep Learning

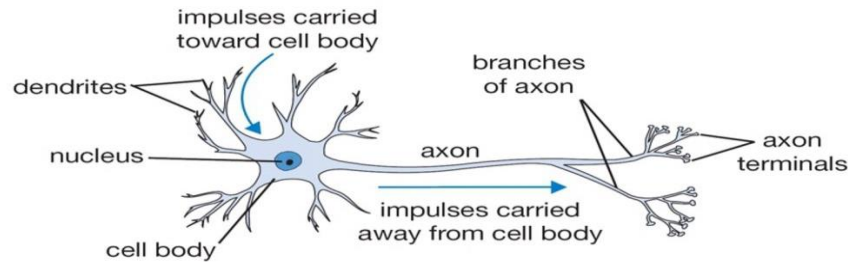
Neural Network Türleri

- Multi Layer Perceptron
- Recurrent neural network
- Convolutional neural network

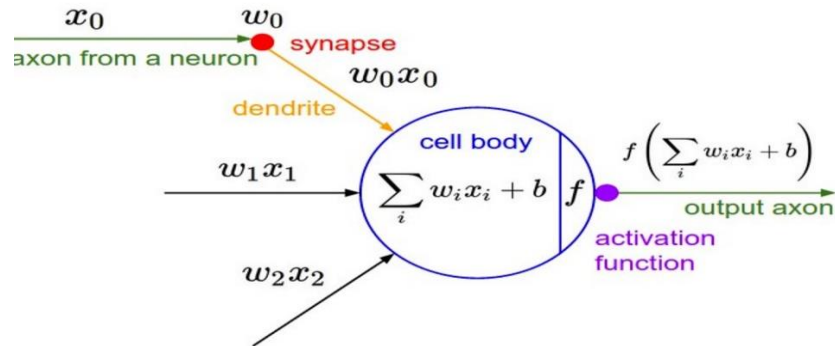
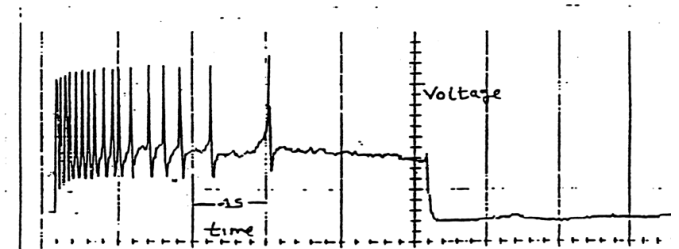
Neural Networks Uygulamaları

- El yazısı karakter tanımlama.
- Görüntü işleme.
- Borsa stok tahmini
- ve binlerce farklı uygulama

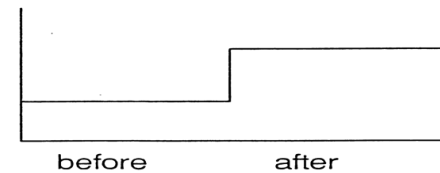
“Gerçek” ve Yapay Nöron



real output

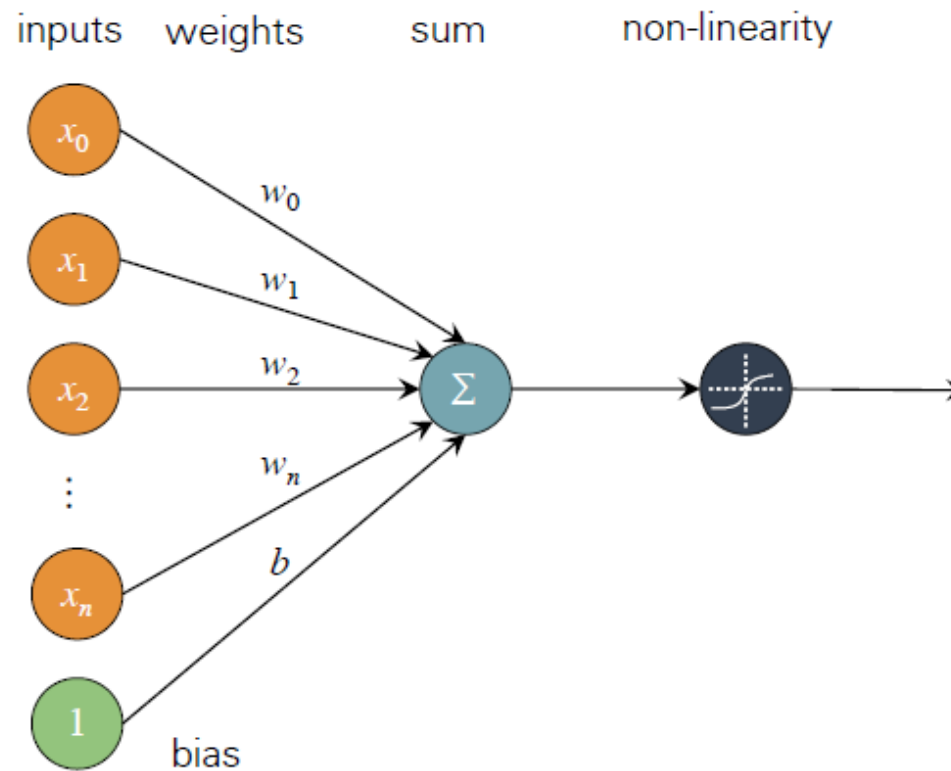


artificial output



<http://cs231n.github.io/neural-networks-1/>

Perceptron



Perceptron Forward Pass

- Neuron pre-activation
(or input activation)

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron output activation:

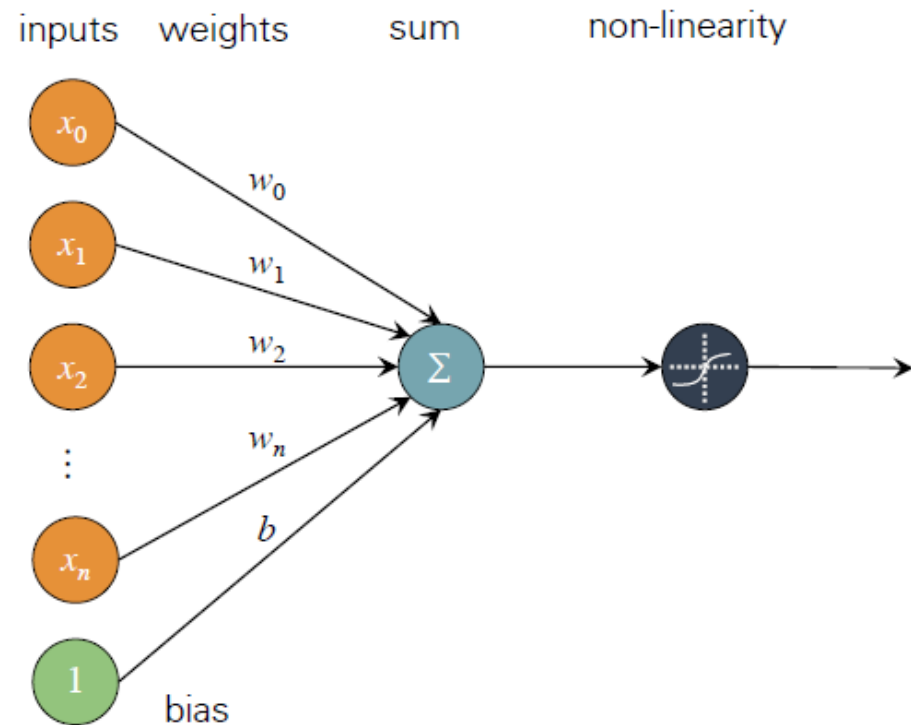
$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

where

\mathbf{w} are the weights (parameters)

b is the bias term

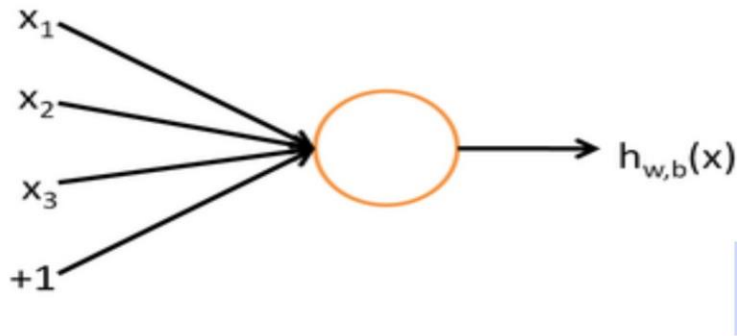
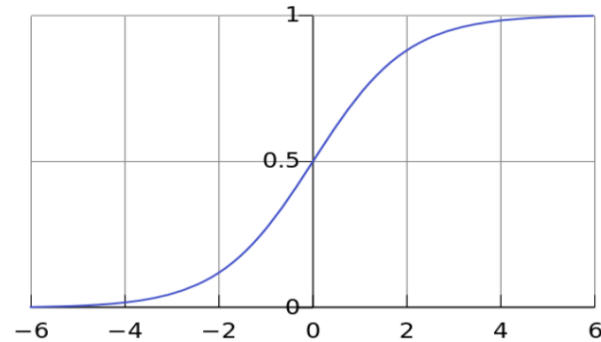
$g(\cdot)$ is called the activation function



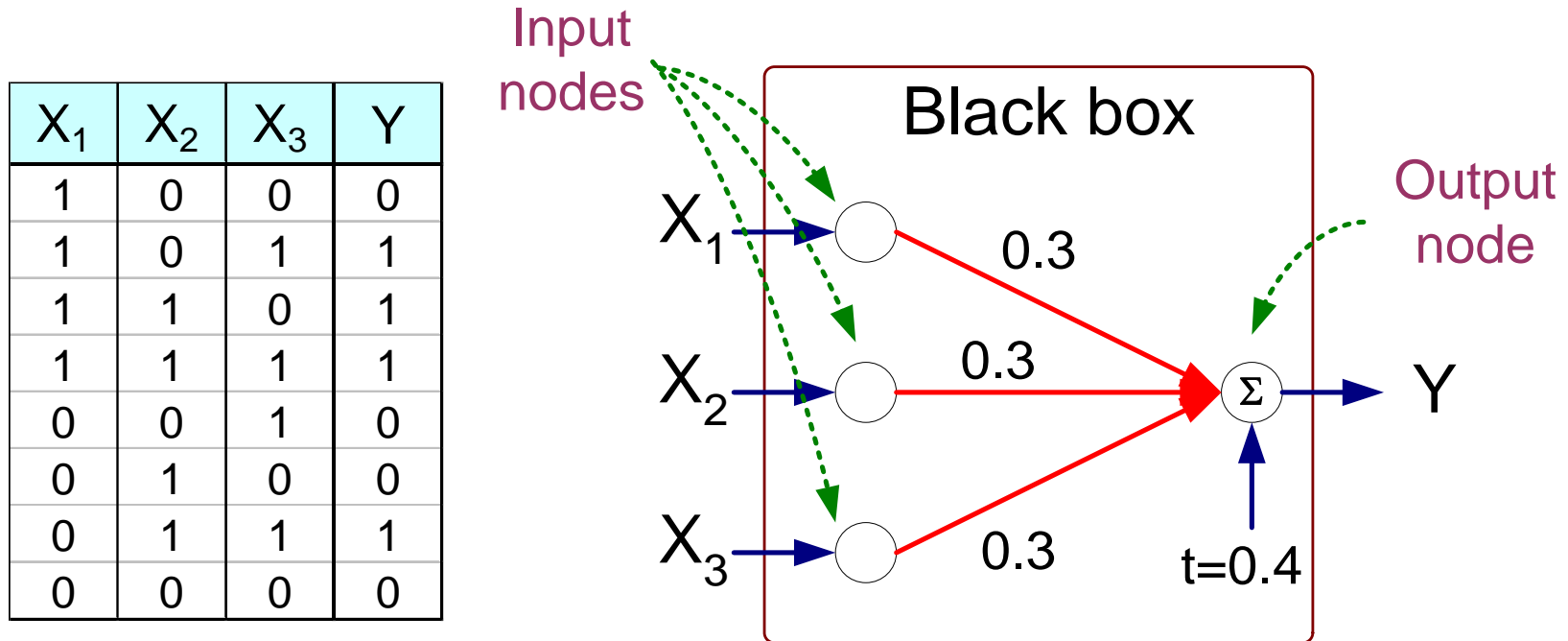
Tek bir nöron logistic regression oluşturur

$$h_{w,b}(x) = f(w^T x + b) \leftarrow$$

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



Artificial Neural Networks (ANN)



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Neural Network

Training Data	Course Hours	IQ Test Result	Education	Sex	Result
	28	109	High School	M	1
	20	72	High School	M	0
	12	121	Undergraduate	F	1
	8	86	Graduate	M	0
	32	94	Undergraduate	F	1
Predictors					Predict the outcome
	28	67	Graduate	M	?

Network

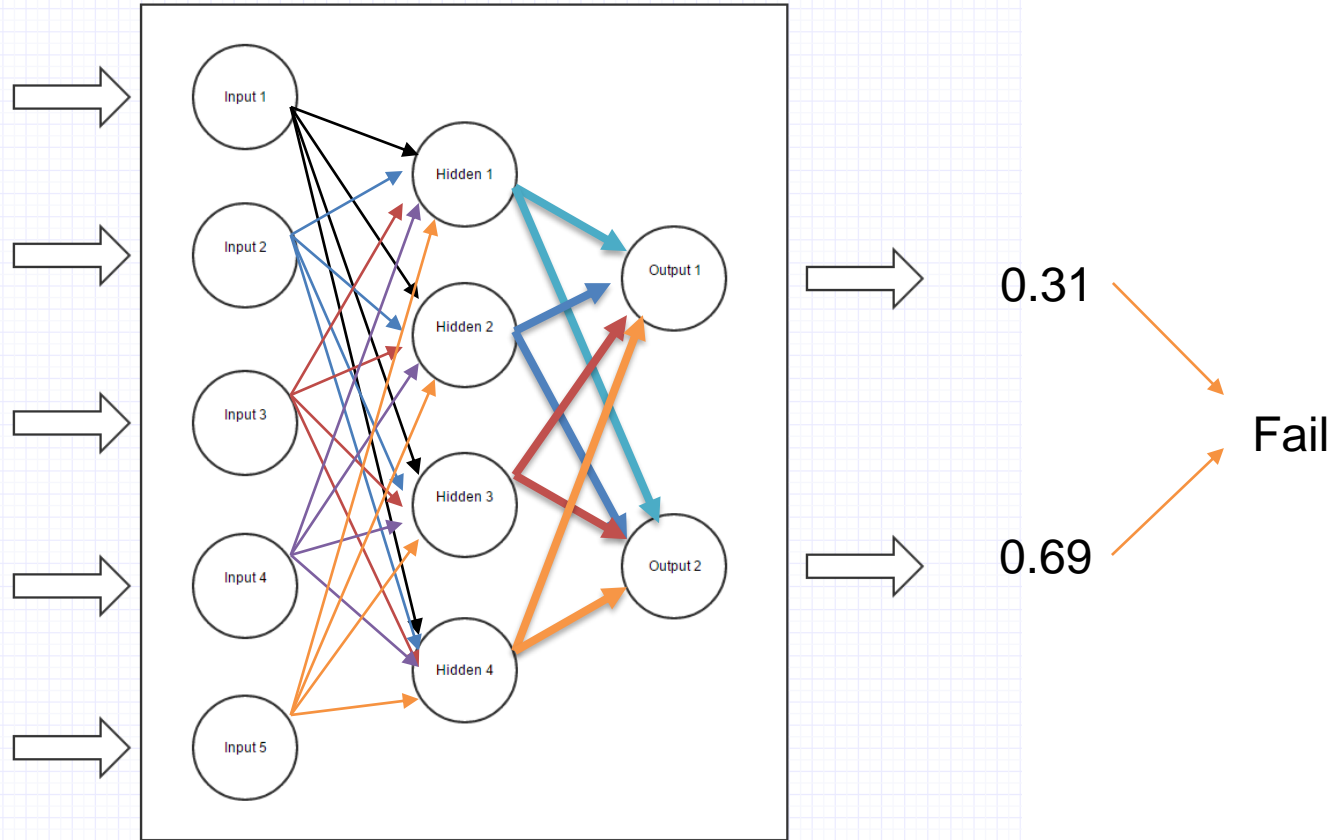
Course Hours: 2.8

IQ Test Result: 6.7

Education: 1.0

0.0

Sex : -1.0



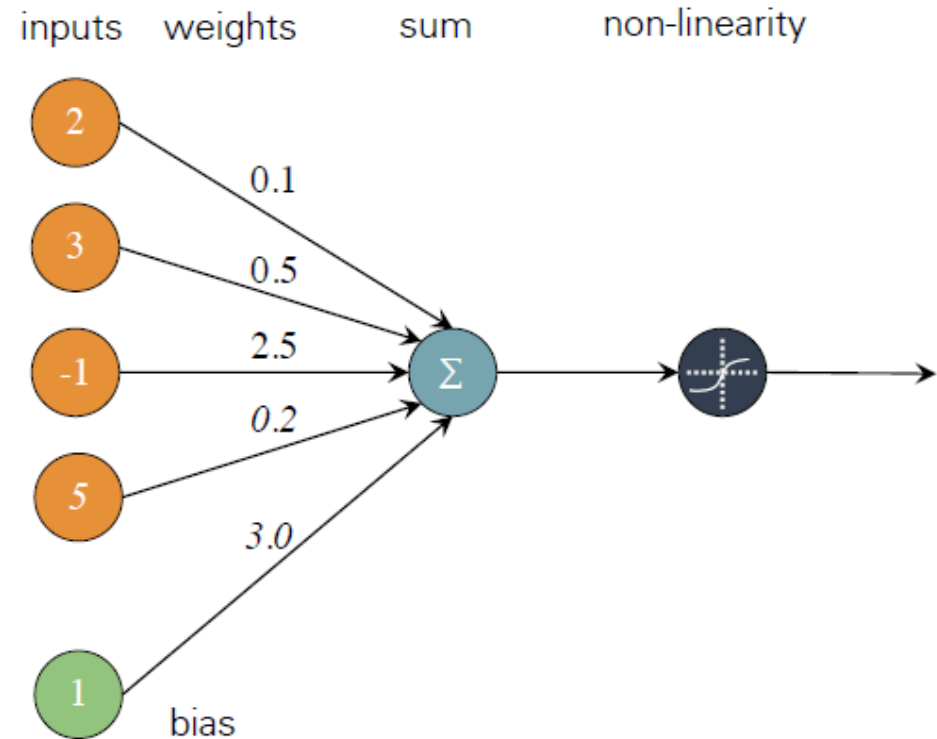
Perceptron Forward Pass

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

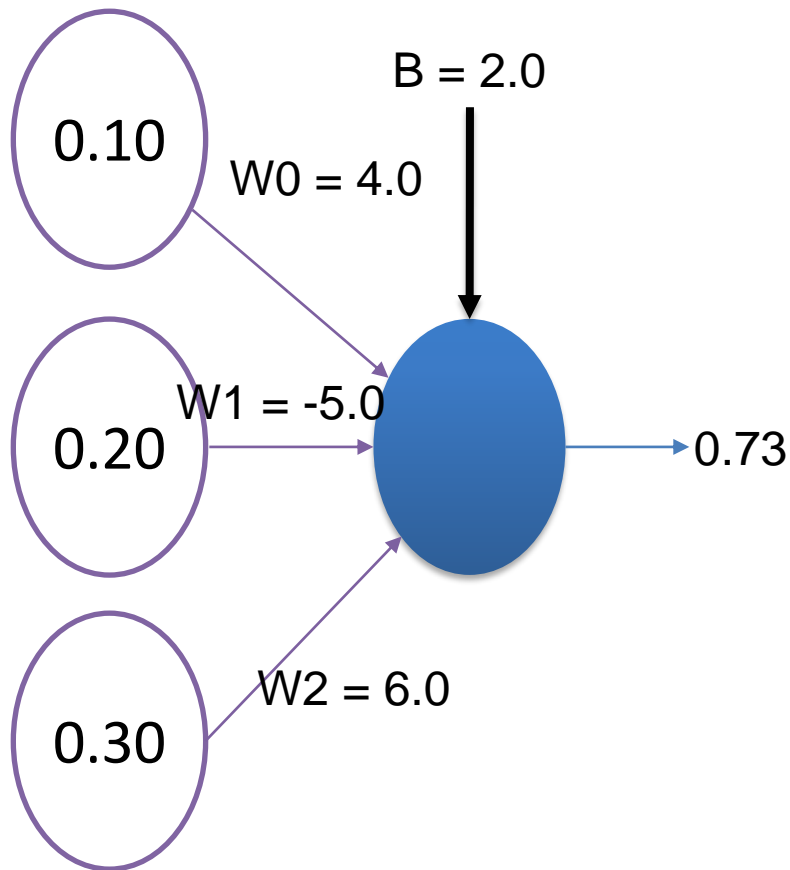
$$\begin{aligned} h(\mathbf{x}) = g(& (2*0.1) + \\ & (3*0.5) + \\ & (-1*2.5) + \\ & (5*0.2) + \\ & (1*3.0) \\ &) \end{aligned}$$

$$h(\mathbf{x}) = g(3.2) = \sigma(3.2)$$

$$\frac{1}{1 + e^{-3.2}} = 0.96$$



Feed-Forward ve Aktivasyon

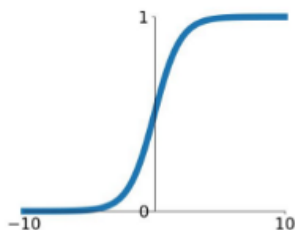


- $0.10 * 4.0 + 0.20 * -5.0 + 0.30 * 6.0 + 2.0 * 1 = 3.2$
- $\text{Activation}(3.2) = 0.73$
- $\text{Output} = 0.73$

Activation Functions

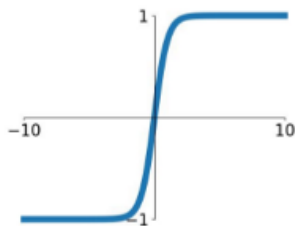
Sigmoid

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



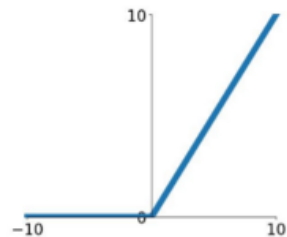
tanh

$$\tanh(x)$$



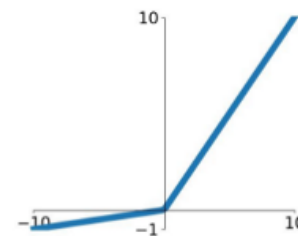
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

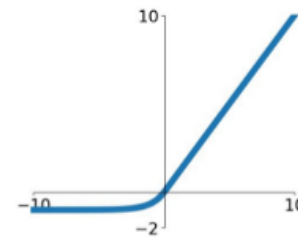


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Multi-Layer Perceptron(MLP)

- Hidden layer pre-activation:

$$\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$$

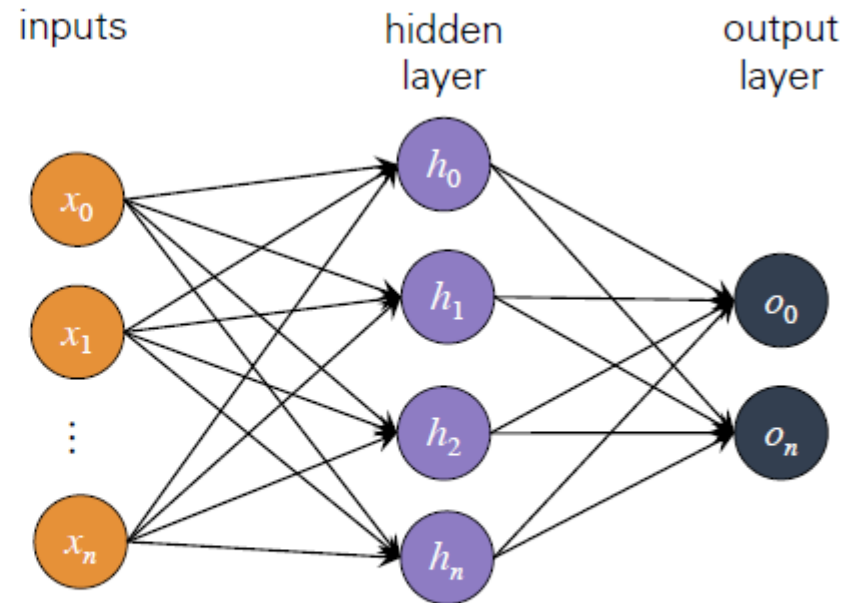
$$\left(a(\mathbf{x})_i = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j\right)$$

- Hidden layer activation:

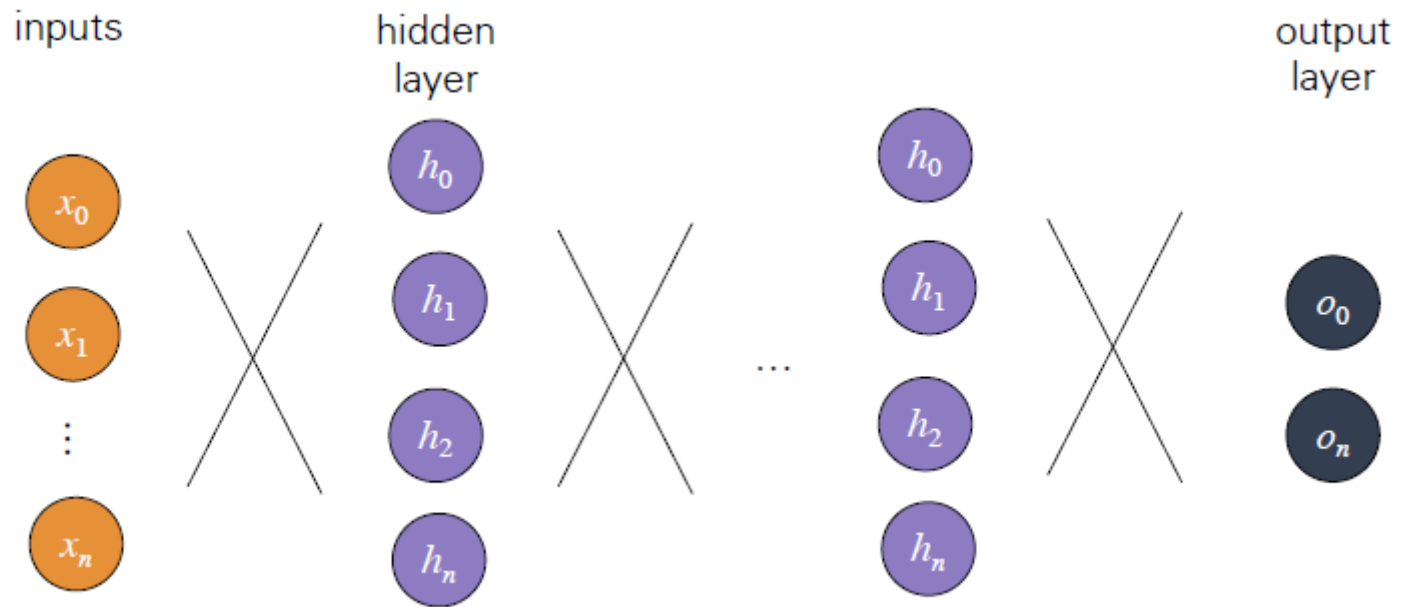
$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$$

- Output layer activation:

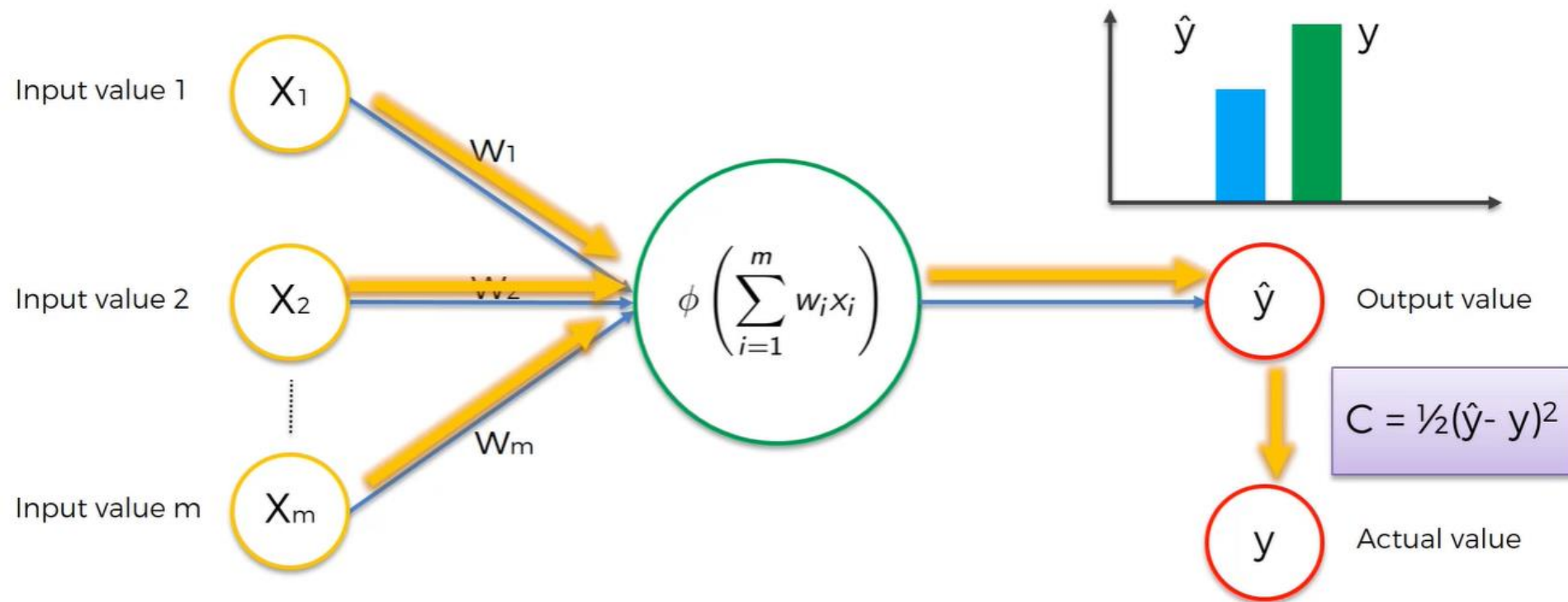
$$\mathbf{o}(\mathbf{x}) = \mathbf{o} \left(b^{(2)} + \mathbf{w}^{(2)\top} \mathbf{h}^{(1)} \mathbf{x} \right)$$



Deep Neural Networks (DNN)



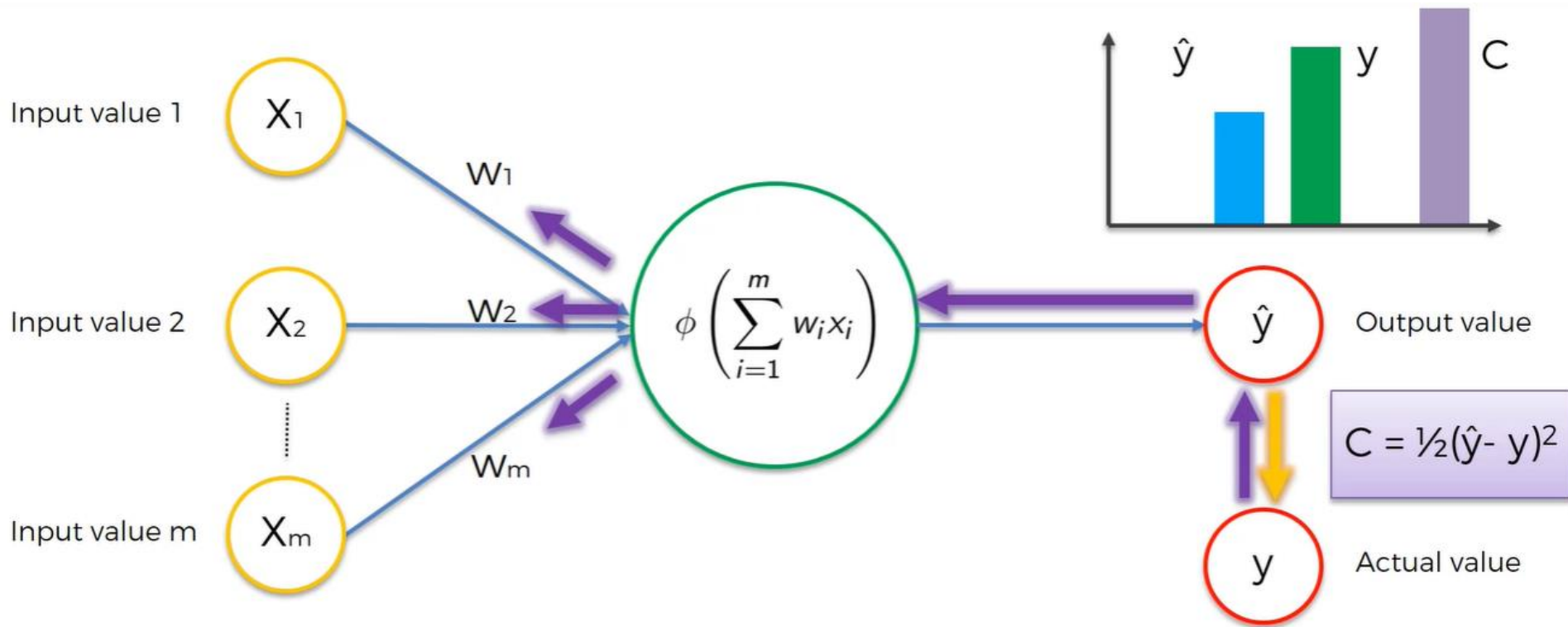
Training(Feed Forward)



Error ve Accuracy


- Mean Squared Error
- $E_{total} = \sum \frac{1}{2} (target - output)^2$

Training(Back Propagation)



Training

$$J(\theta) = \arg \min_{\theta} \frac{1}{T} \sum_t \underbrace{l(f(\mathbf{x}^{(t)}; \theta), y^{(t)})}_{\text{Loss function}} + \underbrace{\lambda \Omega(\theta)}_{\text{Regularizer}}$$

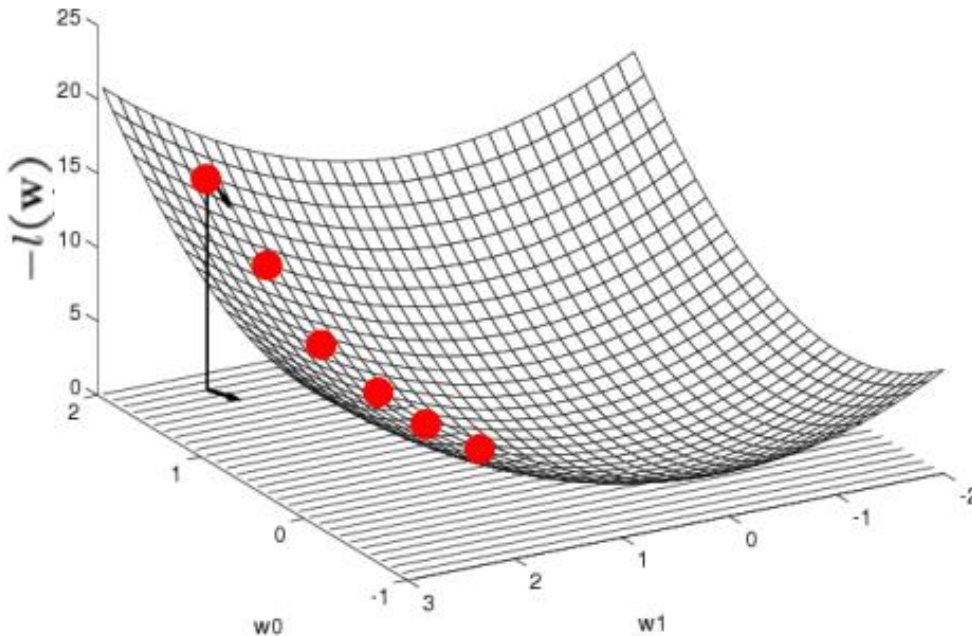


$$\theta = W_1, W_2 \dots W_n$$

- Learning is cast as optimization.
 - For classification problems, we would like to minimize classification error
 - Loss function can sometimes be viewed as a surrogate for what we want to optimize (e.g. upper bound)

Gradient descent

- Optimizing convex function: **Gradient descent (convex)**



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = \left[\frac{\partial l(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial l(\mathbf{w})}{\partial w_d} \right]'$$

Update rule: Learning rate, $\eta > 0$

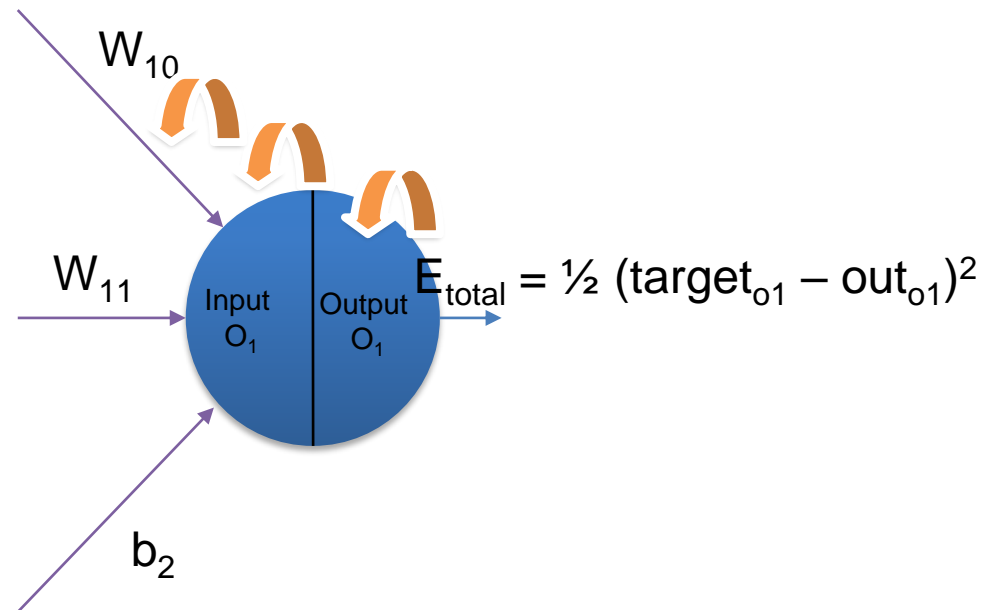
$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

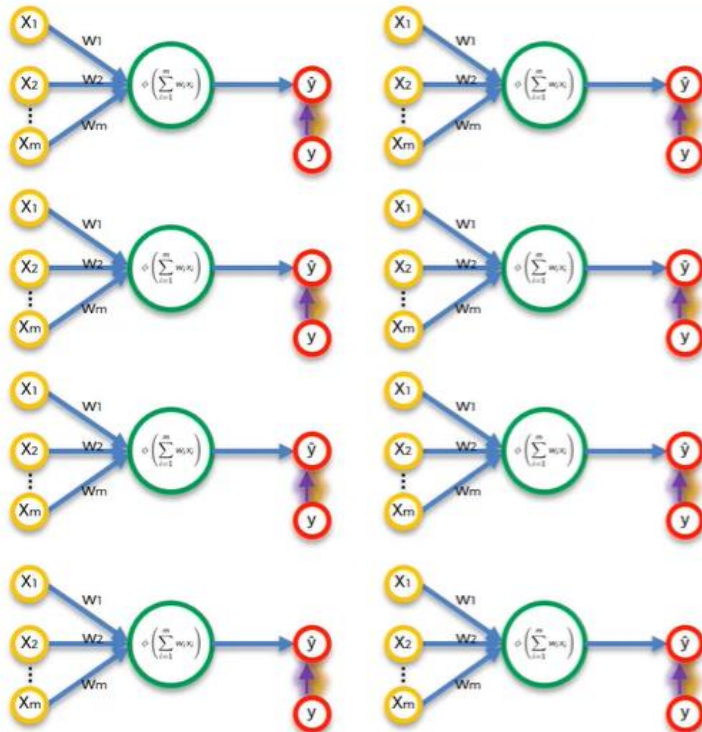
Training (Back Propagation)

- W_{10} ağırlık değerini değiştirmenin etkisini bulma

$$\frac{d E_{total}}{d W_{10}} = \frac{d E_{total}}{d out_{O_1}} * \frac{d out_{O_1}}{d in_{O_1}} * \frac{d in_{O_1}}{d W_{10}}$$



Training(Back Propagation)



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \frac{1}{2}(\hat{y} - y)^2$$

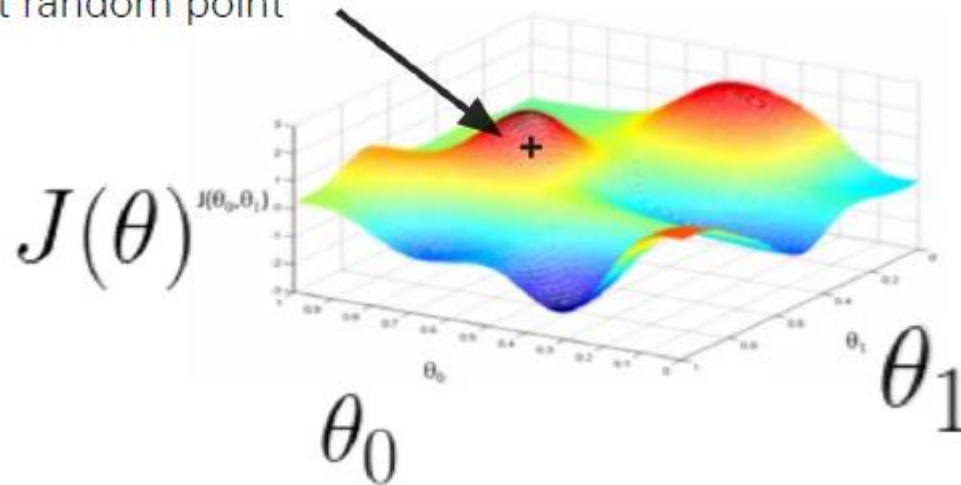


Training

Loss, modelin ağırlık parametreleri fonksiyonudur.
Nasıl minimize edilir?

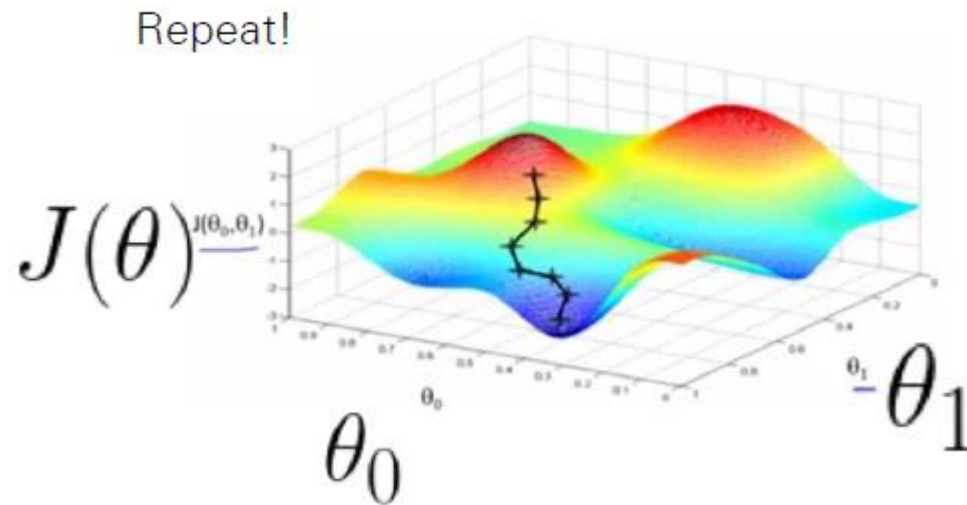
Compute: $\frac{\partial J(\theta)}{\partial \theta}$

Start at random point



Stochastic Gradient Descent (SGD)

Loss, azalan yönde iterative şekilde ilerler



Stochastic Gradient Descent (SGD)

- Initialize θ randomly
- For N Epochs
 - For each training batch $\{(x_0, y_0), \dots, (x_B, y_B)\}$:

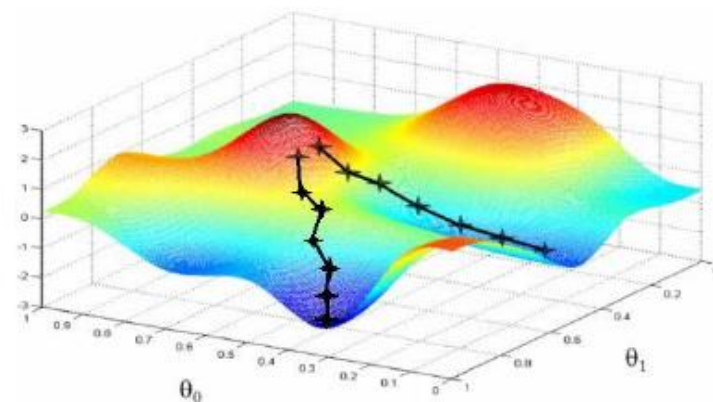
More accurate
estimate!

- Compute Loss Gradient:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{B} \sum_i^B \frac{\partial J_i(\theta)}{\partial \theta}$$

- Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$



Advantages:

- More accurate estimation of gradient
 - Smoother convergence
 - Allows for larger learning rates
- Minibatches lead to fast training!
 - Can parallelize computation + achieve significant speed increases on GPU's

Neural Network Sınıflandırıcı

- Zayıflıkları
 - **Uzun eğitim süresi**
 - Ağ topolojisi veya "yapı" gibi tipik olarak en iyi **deneysel** olarak belirlenen bir dizi **parametre** gerektirir.
 - **Kötü yorumlanabilirlik**: Öğrenilen ağırlıkların ve ağdaki "gizli birimlerin" arkasındaki sembolik anlamı yorumlamak zordur
- Güçlü tarafları
 - **Noise verilere** toleransı yüksek
 - Eğitim setinde **görülmemiş kalıpları** sınıflandırma yeteneği
 - **Sürekli değerli** girdi ve çıktılar için çok uygun
 - **Çok çeşitli** gerçek dünya verileri üzerinde başarılı
 - **Algoritmalar** doğası gereği **paraleldir**
 - **Sinir ağlarından kuralların çıkarılması** için son zamanlarda teknikler geliştirilmiştir.

Deep learning kullanılmalı?

- Çok noise varsa ve sorunun yapısı basitse -> hayır kullanma
- Az noise varsa ve sorunun yapısı karmaşıksa-> evet, deep learning kullan
- İyi ve basit bir temel algoritma ile başlamak faydalı:
 - Hangi algoritmayı iyi biliyorsanız onunla başlayın
 - Logistic regression, SVM, boosted decision tree gayet iyi çalışan yöntemler