# Machine Learning

- Yapay zekanın bir alt dalıdır.
- Bilgisayarlar için yeni bir yetenektir.

Veri madenciliğinde çok fazla kullanılır. Günümüzde web ve otomasyonlardan birçok veri elde ediliyor. Örneğin; web tıklama verileri (kullanıcı kullanım stolaları), medical kayıtlar (tıbbi sonuçlar elde etmek), biyoloji (Gen ve dna dizilimi ile çıkarılabilecek sonuçlar), mühendislik.

Bir başka kullanım alanı elle programlayamadığımız uygulamalar. Örneğin; otonom helikopterler, elyazısı tanımlama, doğal dil işleme (NLP), computer vision.

Öğrenme algoritmaları ayrıca yaygın olarak kendini özelleştirme programları tarafından kullanılmaktadır. Örneğin; Netflix, Amazon gibi programların sizlere yeni film ve ürün önermeleri

- Bugün öğrenme algoritmaları, insan öğrenmesini ve beyni anlamak için kullanılmaktadır.

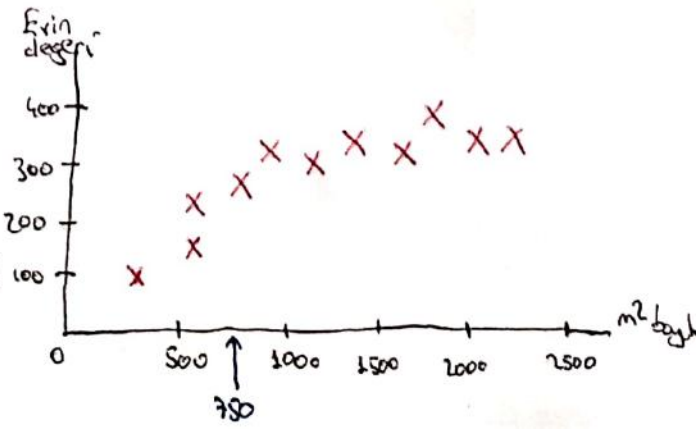
Machine Learning Algoritmaları

- Supervised Learning (Gözetimli-Denetimli)
- Unsupervised Learning
- Reinforcement Learning, recommender (tavsiye) system

"Tom Mitchell" bir bilgisayar programının T görevini, E ile tecrübe ederek P'nin başarı olasılıklarını ölçümleyeceğini söylemektedir.

## Supervised Learning

Bir örnek vererek açıklamaya çalışalım. Konut fiyatlarını tahmin etmek istediğinizi varsayalım.
Bölgedeki konutların veri kümesi yandaki gibidir. 750 $m^2$ evi olan bir arkadaşın olduğunu söyleyelim. Evini satmak istiyor ve ne kadara satabileceğini bilmek istiyor. Makine öğrenmesi nasıl işinize yaracak? Belki veriler üzerinden uygun bir doğru çizerek



Belki de verilere uygun bir 2. dereceden polinom çizerek yeni verileri tahmin edebileceğiz. Bunlardan hangisinin veya başka bir fonksiyonu nasıl seçeceğimizi ilerde göreceğiz.

Kısaca bu örnek, bir denetimli (supervised) öğrenme algoritmasına verilebilecek örnektir. Denetimli öğrenme, algoritmaya verdiğimiz veri setinde doğru cevapların olduğu verilerde bulunur. Bu verilere **label (etiket)** denir. Ev örneğinde de her bir verimiz için evin değerini verdik. Bu yüzden bu bir supervised problemidir.

Denetimli öğrenme problemleri; "regresyon" ve "classification" problemleri olarak ikiye ayırabiliriz. Bir regresyon probleminde, sürekli bir çıktı içindeki sonuçları tahmin etmeye çalışırız, yani girdi değişkenlerini bazı sürekli fonksiyonlarla eşlemeye çalışırız.
Bir clasification probleminde, ayrık bir çıktıdaki sonuçları tahmin etmeye çalışırız. Başka bir deyişle girdi değişkenlerini ayrık kategorilere sınıflandırmaya çalışırız.

Regresyon örnekleri; ev fiyatı tahmini, fotoğrafa göre yaş tahmini, stok tahmini...
Clasification örnekleri; Tümörün kötühuylu olup olmadığı, sitenin hacklenip hacklenmediği...

## Unsupervised Learning

Bu öğrenme algoritmasında, veri setlerinde etiketli veri bulunmaz. Bu öğrenme yönteminde veriler özelliklerine göre sınıflandırılır ve yeni gelen veriler bu sınıflandırılan verilere özelliklerine göre tahminde bulunurlar.

Örnek verecek olursak; sosyal medya analizi (reaksiyonlarınıza göre kimlerin ortaklaşınız olabileceği), pazar segmentasyonu (müşterilerin hangi pazar segmentasyonuna dahil olduğu), astronomik veri analizi.

# Model ve Cost Function

## Model Gösterimi

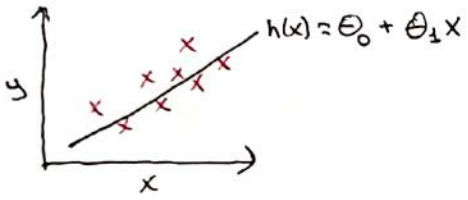
Lineer regresyon modeli olarak ilk örneğimiz ev fiyatı tahmin etme olsun.

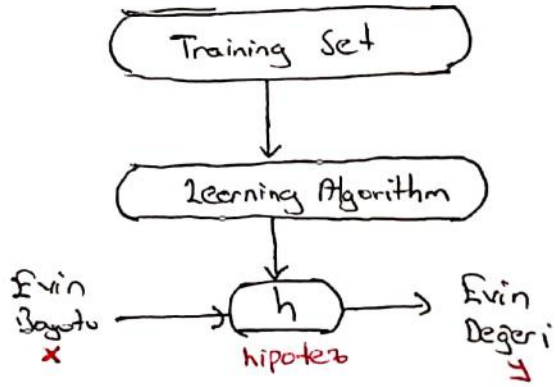
| (x) Size in feet$^2$(m) | (y) Price ($) |
|---|---|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

$m$ = veri setindeki örnek sayısı
$x$ = girdi değişkeni / features
$y$ = çıkış değişkeni / target variable
$(x, y)$ = veri setindeki tek bir örnek
$(x^{(i)}, y^{(i)})$ = veri setindeki özel bir örneği göstermek

$$h_\theta(x) = \theta_0 + \theta_1 x$$

shorthand = $h(x)$



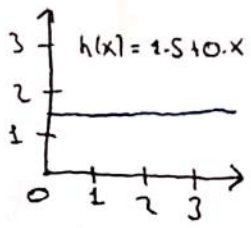



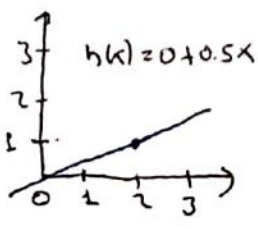

Burada bir veri setimiz ve modelimizi oluşturacak bir algoritmamız olacak. Algoritmadan çıkacak fonksiyona (hipotez) bağlı olarak tahmin işlemini gerçekleştireceğiz.

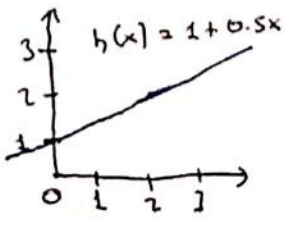
Yukarıdaki örneğimiz lineer regresyon modeline verilebilecek bir örnektir. Tek bir özelliği (feature) olduğu için tek değişkenli lineer regresyon'da diyebiliriz.

## Cost Function

Verilerimize mümkün olan en iyi düz çizgiyi nasıl yerleştirebileceğimizi anlamamızı sağlayacaktır. Burada $\theta_0$ ve $\theta_1$'in nasıl seçilmesi gerektiğine bakacağız. Çünkü bu değerler modelimizin parametrelerini dengeleyen değişkenlerdir. $\theta_0$ ve $\theta_1$'in farklı seçimleri farklı hipotezlerin çıkmasına neden olur.



Cost function'da tahmin değerinin çıktısı ile evin gerçek değeri arasındaki farkın karesini minimize hale getirmeye çalışarak en iyi çizgiyi bulabiliriz.

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

i = Veri setimizdeki örneklerin 1'den m'e kadar olanları.

(mean squared error)

Bu formül **kareler farkı** formülüdür. Bu formülde tahminim ve gerçek sonuç arasındaki farkların karesinin toplamının en az olmasını sağlamamız gerekecektir.

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \quad \text{Squared error}$$

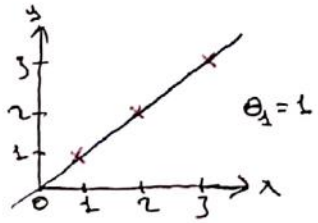
$$\underset{\theta_0,\theta_1}{\text{minimize}} \quad \underbrace{J(\theta_0,\theta_1)}_{\text{cost function}}$$

Birçok maliyet fonksiyonu var ancak en yaygın kullanılanlardan birisi squared error fonksiyonudur.

**Not** Hedef $\theta_0$ ve $\theta_1$'i minimize edebilmek.

**Not** Mümkün olan en iyi çizgi, noktaların çizgi ile olan dikey uzaklığının en az olmasıdır. En iyi çizgi uzaklığın "0" olması ile elde edilir.

**Örnek** $\theta_0$'ı "0" olarak alıp $\theta_1$'in maliyet fonksiyonunu hesaplayalım.

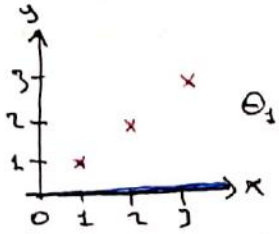



$h_\theta(x) = \theta_1 x$

$\theta_1 = 1$

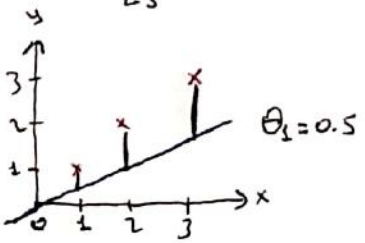
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2\cdot 3}\left(0^2+0^2+0^2\right) = 0$$



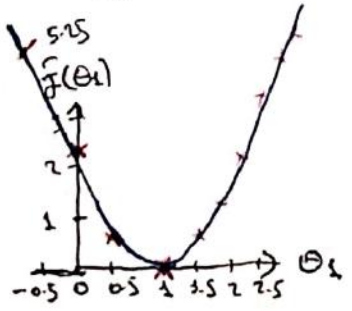

$\theta_1 = 0$

$$J(0) = \frac{1}{2\cdot 3}\left((0-1)^2+(0-2)^2+(0-3)^2\right) = \frac{14}{6} = 2.3$$




$\theta_1 = 0.5$

$$J(0.5) = \frac{1}{2\cdot 3}\left((0.5-1)^2+(1-2)^2+(1.5-3)^2\right) = 0.58$$




$\theta_1 = 1$ noktasında en minimizedir.

# Gradient Descent

Maliyet fonksiyonunun teta değerlerini minimuma getirebilmek için kullandığımız genel bir algoritmadır. Hemen hemen her cost function için kullanılır.

Gradient descent algoritması, bir tepeden en aşağı inmek için küçük küçük optimum atılan adımlara benzer.

$\alpha$ : learning rate

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

**Not** Learning rate ($\alpha$), yokuş aşağı giderken ne kadar büyüklükte bir adım atacağımızı belirler. $\alpha$ çok büyükse büyük adımlar atmış oluruz. $\alpha$ küçükse küçük adımlar atmış oluruz.

**correct update (eşzamanlı)**

$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := temp0$
$\theta_1 := temp1$

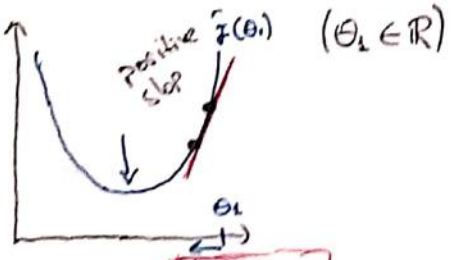
**incorrect update**

$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\theta_0 := temp0$
$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_1 := temp1$

**Not** Yokuş aşağı gitmek istediğimizde bize yönümüzü gösterecek maliyet fonksiyonunun kısmi türevidir. Tanjantın eğimi o noktada türevdir ve bize doğru ilerlemek için bir yön verir.

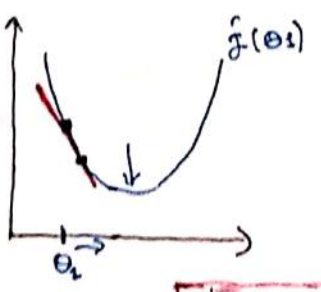
**Not** Gradient descent'te parametre güncellemeleri 0 olana kadar ve 0'a yakınsayana kadar devam eder. (Maliyet fonksiyonunun kısmi türevi)

**Örnek** Tek bir parametre için gradient descent'in tepkisi




$\theta_1 := \theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\geq 0}$

$\theta_1 := \theta_1 - \alpha$ . (positive number)




$\theta_1 := \theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$

$\theta_1 := \theta_1 - \alpha$ (negative number)

## Doğrusal Regresyon için Gradient Descent

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left( (h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left( \theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

$$\theta_0 := \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \cdot 2 \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{2m} \cdot 2 \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

} squared error'un kısmi türevleri

## Yakınsamaya Kadar Tekrar Edecek Gradient Descent

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

Doğrusal regresyon her zaman bir kase şeklinde maliyet grafiği çıkarır. Bu grafiğe dışbükey (konveks) fonksiyon denir. Bu fonksiyon tek bir global optimum değer barındırır.

Not: Lineer regresyon kullanıyorsanız, her zaman küresel optimum seviyeye ulaşacak.

Not: Bu gradient descent veri setindeki tüm verilere baktığı için toplu gradyen descent (Batch Gradient Descent) denir.

## Lineer Cebir

### Matrix - Matrix Çarpım

Birinci matrisin sütunu ile ikinci matrisin satırı aynı olmalıdır.

$$m \times n \;\; X \;\; n \times a \;=\; m \times a$$

$A \times B \neq B \times A$ (Değişme özelliği yok)

$I \times A = A \times I$

### Matrix Inverse

Matrisin tersi olması için kare matris olmalı.

### Matrix Transpose

Matrisin sütunlarını satır yaparak ters bir matris elde ederiz.

$$A_{ij} = B_{ji}$$