1

Where to use?

- Database mining
  Large datasets from growth of automation / web.
  E.g. Web click data, medical records, biology, engineering.
- Applications can't program by hand
  E.g. Autonomous helicopter, handwriting recognition
  Natural Language Processing, Computer Vision.
- Self-customizing programs
  E.g. Amazon, Netflix product recommendations
- Understanding human learning (brain, real AI).

What is machine learning?

- (1959): Field of study that gives computers the ability to learn without being explicitly programmed.
- 1998: Well-posed learning problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E

Types of ML algorithms

- Supervised learning
- Unsupervised learning

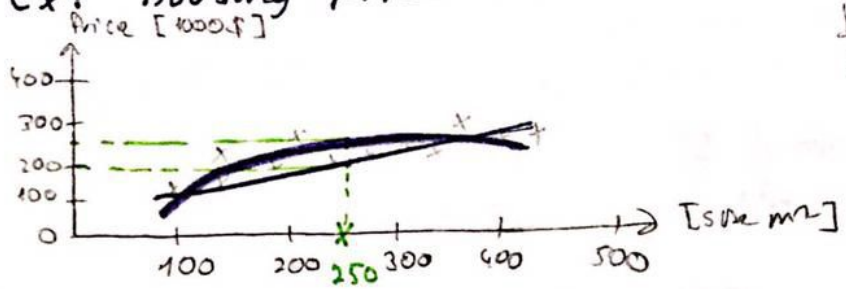
others: Reinforcement learning, recommender systems

Introduction ML WS

1

Supervised Learning

Ex: Housing price prediction



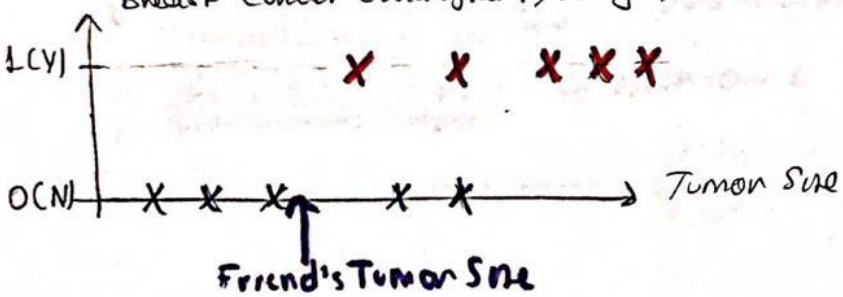
- Böyle bir dataset var diyelim, biz ise 250 m²'lik bir evin fiyatını tahmin etmeye çalışıyoruz.
- Bunun için datalardan bir straight line veya bir quadratic function geçirebilirim.

✡ Bu bir **supervised learning** örneğidir. Çünkü "Right Answers" is given to the algorithm.

✡ This problem is also a **Regression Problem**: Predicting continuous valued output (price)

Another supervised learning example:

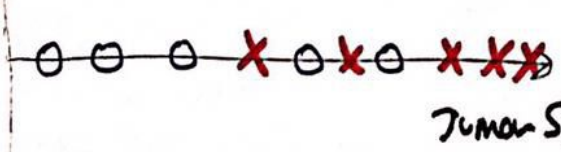
Breast Cancer (Malignant, benign)



- Örneğin bir arkadaşın morluklarını işaret ettiği boyutta bir tümörü var. Biz de bu tümörün malignant olma probabilitysini öğrenmek istiyoruz.

✡ **Classification Problem:** Discrete valued output (0 or 1)
Burada 2 den fazla grubu da clasify edebiliriz. Output 0, 1, 2, 3 ... şeklinde gidendir.

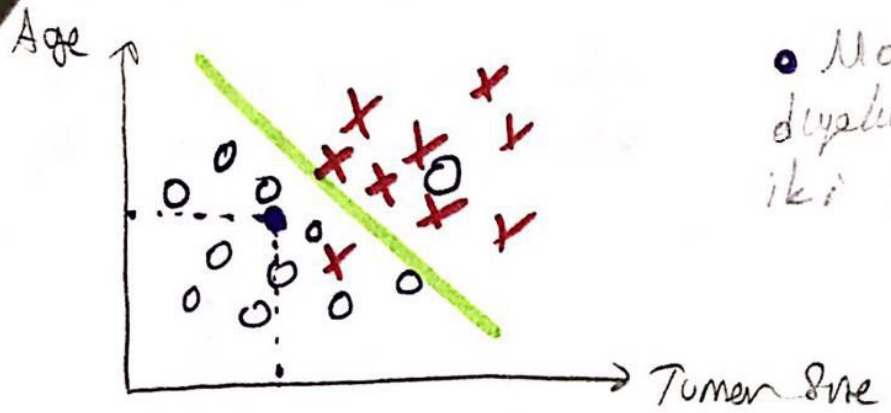
✡ classification problemde data farklı şekilde de çizilebilir:



- Böylece aynı datayı tek eksende göstermiş oldum.

There is only one **Feature** or **Attribute** (Tumor Size

Multifeatured Classification Problem



- Mor noktaya denk düşen bir tümör var diyelim. Learning algorithmin yapacağı şey iki grubu bir çizgi ile ayırmak

* daha fazla feature da kullanılabilirdi.

Sonuç olarak:

- **Supervised** için doğru cevapla veriliyor (Evet fuyak veya tümörün B/M olusu)
- **Classification**: Discrete output
- **Regression**: Continious output

Unsupervised Learning

- Supervised için data plot şuna benziyordu:

- Unsupervised için ise data set için label olmaz hen datanın neye denk geldiği söylenmez.

- İstenilen şey şu: Here is the data-set can you find some structure in the data. Learning algorithm bu durumda iki cluster oluşturabilir. Buna clustering denir.

Cocktail Party ~~Algorithm~~

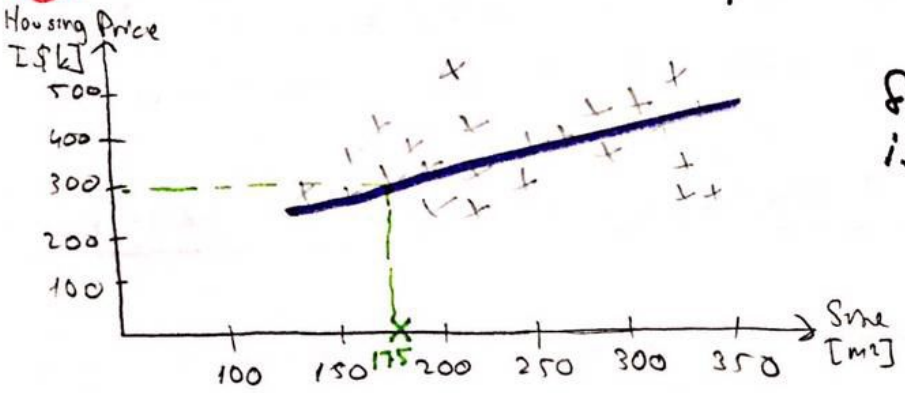
- iki kişi 1 den 10'a saysın. iki mikrofon olsun biri ing. sayarken diğeri italyanca sayarsa yakın olsun. sonuçlarında da iki farklı sayım duyulur bunun resmi daha yüksek.

* Take this recordings give to an unsupervised learning algorithm and tell the algorith find structures.
* What algorithm will do is, listen them and say: sound like two recordings are added together, maneuver cocktail party algorithm will seperate them.

* Bu algorithm 1 satırlık bir algorithmdir!
* Octave ve Matlab ile machine learning alg. uygulaması daha kolay olur.

Model Representation

Let's look at a housing price prediction application.



* 175 m2'lik evimin fiyatini tahmin etmek istiyorsa.

One thing we could do is fit a MODEL. Maybe fit a straight line to this data.

Remember, this is an example of supervised learning algorithm. Moreover this is an example of regression problem.

For supervised learning we have a training set and our job is to learn from this data to predict prices.

Here are some notations

Size in m² (x)	Price in $k (y)
205	307
257	335
487	405
323	370
...	...

$m = 47$

- m: # of training examples
- x's: "input" variable / features
- y's: "output" variable / "target" variable
- (x, y): one training example
- $(x^{(i)}, y^{(i)})$: ith training example

$x^{(1)} = 205$
$y^{(1)} = 307$
⋮

We feed our training to our learning algorithm. The job of the learning algorithm is to output a function h (hypothesis function). The job the hypothesis is estimating a price by any given input size.

Training Set → Learning Algorithm → h (hypothesis)

size of house (x) → h (hypothesis) → y → Estimated price

Hypothesis h is a function maps x's to y's

How do we represent h?

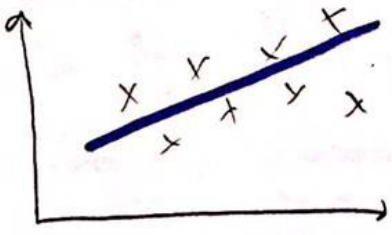
Our initial choice will be

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

shorthand h(x) instead of $h_\theta(x)$

Note that this h is a straight line function.

Zamanla daha complex nonlinear hypotheses-leri de kullanacağız. Başlangıç için bunu kullanıyoruz!



Linear Regression with one variable (x)

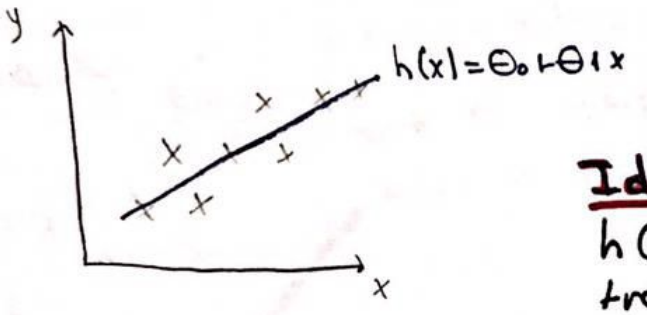
Cost Function

Cost function will let us figure out how to fit the best possible straight line to our data.

m: # of training examples olduğunu unutma.

Hypothesis: $h(x) = \theta_0 + \theta_1 x$ demiştik.

How to choose PARAMETERS $(\theta_0, \theta_1, \dots)$?



- How do we come up with θ_0 and θ_1 ?

Idea: Choose θ_0, θ_1 so that $h(x)$ is close to y for our training examples.

I want to minimize $\sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)}\right)^2$ over θ_0, θ_1.

Result: If I can minimize the COST FUNCTION

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)}\right)^2$$

over θ_0 and θ_1, sonucta elde edilen hipotezim başarılı bir hipotez olacaktır.

Burada cost function Squared Error Function olarak seçildi. Linear Regression problemlerinde bu popülerdir ve işe yarar.

Kısaca cost function tüm training ex.s için hata karesinin toplanması ile oluşur.

- Cost Function Intuition I -

In order to visualize cost function we will use a simplified hypothesis:

$h(x) = \theta_1 x$ assume $\theta_0 = 0$

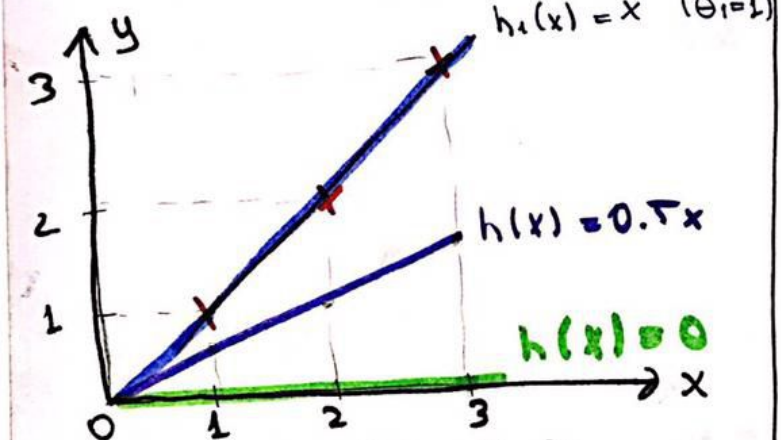
Hypothesis $h(x)$

- For fixed θ_1, it is a function of x.

Cost Function $J(\theta_1)$

- It is a function of parameter θ_1. (For fixed training set)

- Now let's say we have a training set with $m = 3$ as below. We will find the value of $J(\theta_1)$ for different θ_1 values.



- For $\theta_1 = 1$, $h(x) = x$ so

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{6} \cdot [0^2 + 0^2 + 0^2] = 0 \;///$$

- For $\theta_1 = 0.5$

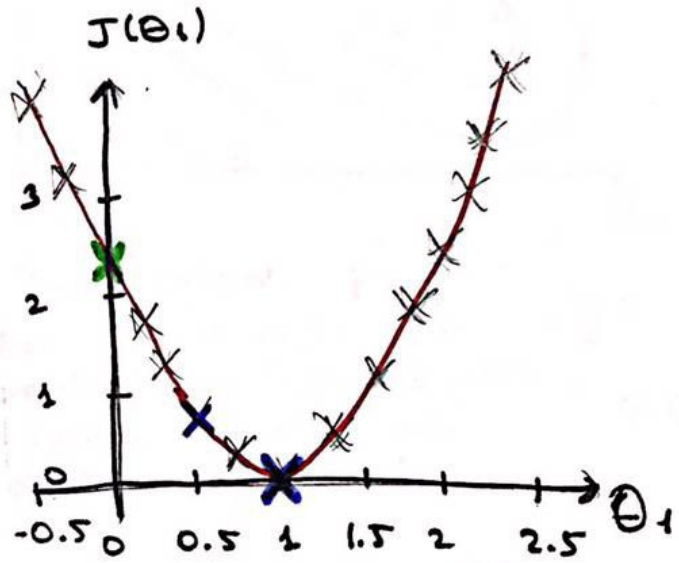
$$J(\theta_1) = \frac{1}{6} \cdot [0.5]^2 + (1)^2 + (1.5)^2]$$

$$\simeq 0.58 \;///$$

- For $\theta_1 = 0$

$$J(\theta_1) = \frac{1}{6} \cdot [1^2 + 2^2 + 3^2]$$

$$\simeq 2.3 \;///$$



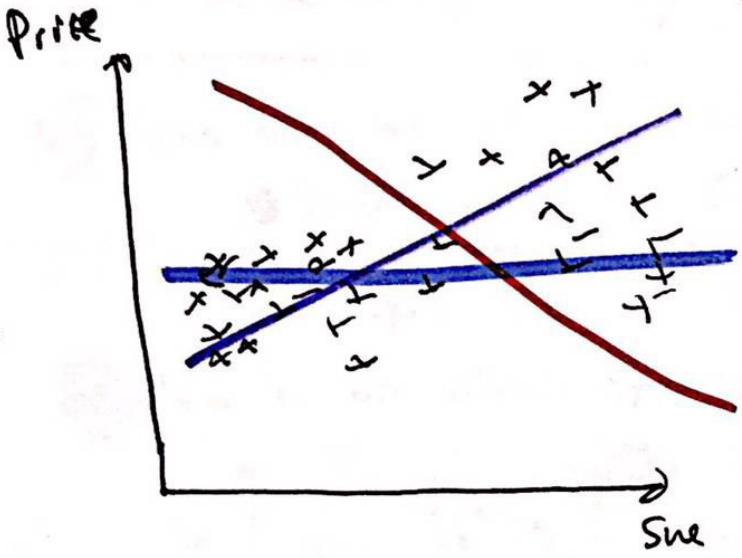
- Üç noktada hesaplandı, daha fazla hesaplanırsa da minimum değerden uzaklaştıkça büyüyen bir fonksiyon şekli ortaya çıkar.

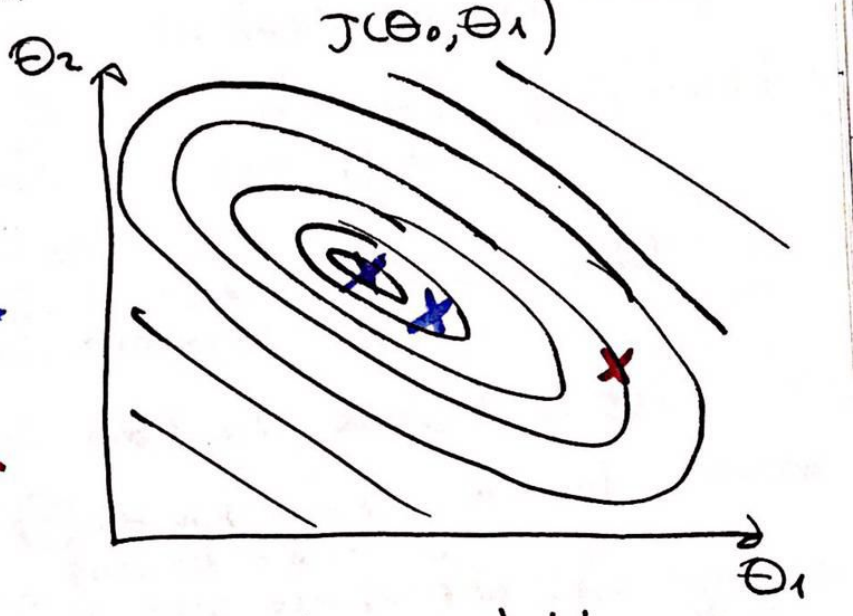
- $J(\theta)$ üzerindeki her noktanın bir hipotezin hata değerini temsil ettiğini unutma amaç ise en az hataya sahip cost function'ı bulmak.

? Cost function her zaman can şeklinde mi? Kısılları neden? Linear regression hep mi böyle? Non-linear olsa nasıl olur? Classification'da nasıl?

- Cost Function Intuition II -

Bu videoda $J(\theta_0, \theta_1)$ 'in plot'un nasıl olduğunu gösterdi ve hypothesis bağlantısını gösterdi





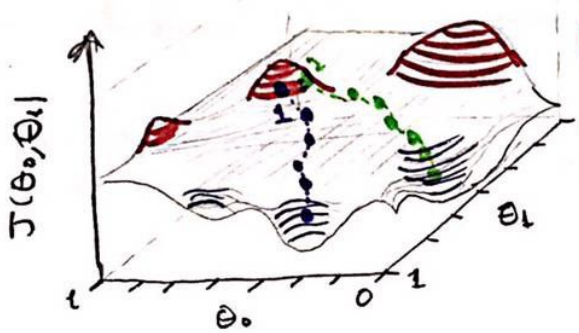
Bu contour plotlar her cincle aynı yüksekliğe denk gelir. Yani bu daire boyunca cost function $J(\theta_0, \theta_1)$ değeri aynıdır!

Sıradaki adım şu olmalı, bir efficient algorithm ile cost function'ı minimize etmek! Böylece en iyi hipotezi bulurut!

Gradient Descent

- "Gradient Descent" is an algorithm for minimizing the cost function.
- We use GD algorithm all over the machine learning not just for linear regression.
- Bu dersle GD alg. ile rastgele bir J function'ı minimize edeceğiz.
  - GD alg. $J(\Theta_0, \Theta_1, \dots \Theta_n)$ gibi genel formda da ise yarar!

- Assume we want to minimize $J(\Theta_0, \Theta_1)$
  - Start with some Θ_0, Θ_1
  - Keep changing Θ_0, Θ_1 to reduce $J(\Theta_0, \Theta_1)$ until we hopefully end up at minimum.
- Here is an intuition; what gradient descent does:



- Θ_0 ve Θ_1 için 1 numaralı noktadan başladık diyelim. GD ile yapılan şey şu kendi etrafımızda 360° dönüyorsun ve aşağı doğru inmek için hangi yöne doğru bir baby step atmamız gerektiğini buluyorsun.
- İlk adımı attıktan sonra aynı süreci tekrarlarız. Böylece mor yolu izleyerek bir local minimum'a ulaşırız. Burada önemli bir nokta şudur:
- Eğer initial Θ_0 ve Θ_1 farklı seçilseydi bambaşka bir local minimum'a ulaşabilirdik. Bu GDA'in bir özelliği bunu daha sonra tartışacağız.

That's the intuition in pictures
let's look at the math!

Gradient Descent Algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$ for $j = 0$ and $j = 1$

}

assignment — learning rate

We have to make a simultaneous update:

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

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

X WRONG implementation!

Simultaneous olmayan da muhtemelen çalışır ama gradient descent ilkiden ikincisinin kendine has başka özellikleri vardır.

Gradient Descent Intuition

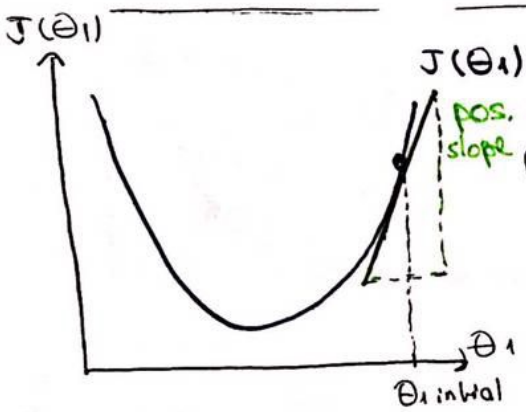
GD Algorithm was:

repeat until convergence { ← Derivative term

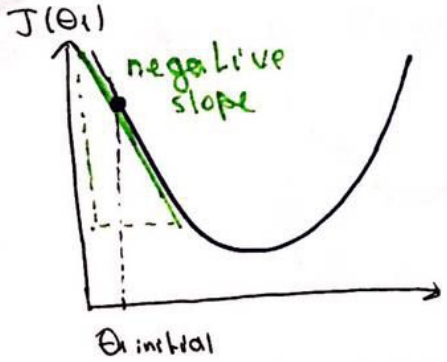
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (Simultaneously update $j=0$ and $j=1$)

} learning rate

- To understand what GDA is doing let's assume we have a hypothesis as $h(x) = \theta_1 \cdot x$ meaning $\theta_0 = 0$. Thus the cost function will be: $J(\theta_1)$ where $\theta_1 \in \mathbb{R}$.
- We know $J(\theta)$ was looking like below graph. What we want to do is understand how GDA works.



- $\theta_1 := \theta_1 - \alpha \left[\frac{\partial}{\partial \theta_1} J(\theta_1)\right]$ → ilgili θ_1 noktasında $J(\theta_1)$'in eğimi!
- Bu durumda derivative term ≥ 0 dır! Bu sebeple θ_1'in yeni değeri azalır yani minimum'a doğru slope ve α ile bağlantılı bir adım atar!
- α is always a positive number!



$\theta_1 = \theta_1 - \alpha \left[\frac{\partial}{\partial \theta_1} J(\theta_1)\right]$ → ≤ 0 !

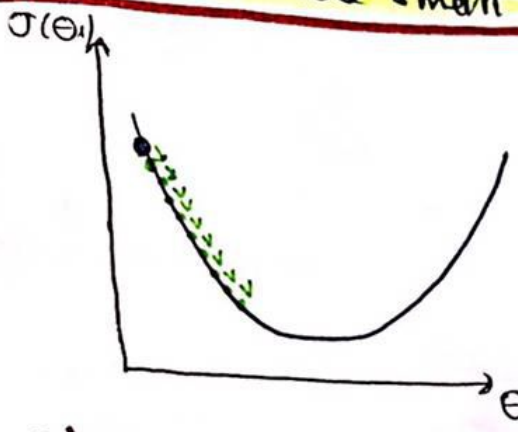
- Bu durumda yeni θ_1 değeri artacaktır böylece local minimum'a bir adım atılmış olur.

- Sonuç olarak GDA ile her durumda GDA ile θ parametreleri kendini yokuş aşağı bırakmaya yönelim- lidir! Bu şekilde minimum'a ulaşılır!

Now let's try to understand what learning rate α is doing...

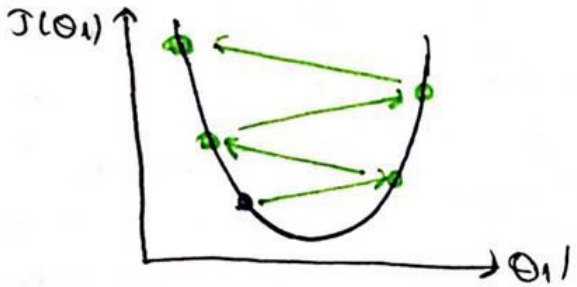
Learning Rate, α Intuition

1) If α is too small? => Gradient Descent can be slow.

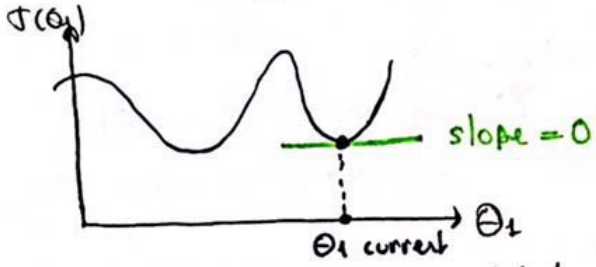


- α çok küçük olursa her update (θ) ile atılan adım boyutu çok küçük olacak bu yüzden minimum'a ulaşmak genelden farla zaman alabilir.

2) If α is too large? => Gradient descent can overshoot the minimum. It may fail to converge or even diverge



- If θ_1 is already at a local minimum:



$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

$$= \theta_1 - 0 = \theta_1 \;///$$

θ_1 stays unchanged !!!

- As we approached a local minimum, gradient descent will automatically take smaller steps (even if the α is fixed). So no need to decrease α over time

- Local minimuma yaklaştıkça eğim azalır bu yüzden update ile atılan adım α sabit olsa dahi küçülür.

Gradient Descent for Linear Regression with one Variable

Gradient Descent Algorithm

repeat until convergence {

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

for $j = 1$ and $j = 0$

}

Linear Regression Model & Cost Function

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

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

Apply gradient descent algorithm to this cost function!

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

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

Sonuçta burada J θ_0 ve θ_1'in bir fonksiyonu her bir farklı θ_0 ve θ_1 parametresi verin ayrı bir J değeri elde ediliyor! Sınıflı bir J fonksiyonu var ortada ve partial derivative alınabilir.

As a result GDA becomes:

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ $\quad$ $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

repeat until convergence {

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

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

}

Simultaneous update olmalı!

Cost function for Linear Regression is always going to be a "Convex Function". Meaning it doesn't have a local optima except for global optima! (Bowl Shape Cost Function)

Cost function şekli hipotese ve cost function formatına bağlı Linear Regression için hipotez $h(x) = \theta_0 + \theta_1 x + \dots$ şeklinde lineerden ise Cost Function Error Sevare!

Daha önce de bahsedildiği gibi her update ile Θ_0 ve Θ_1 değişecek sonuçta hipotez kendini minimum hataya doğru adım adım yaklaştıracaktır.

"Batch" Gradient Descent

Batch gradient descent algoritması ile her update iterasyonu için tüm training examples kullanılır.

- Böyle olmayan GDA'larda vardır. Daha sonra bunlardan da bahsedeceğiz.

Normal Equations Method

Bu method ile linear algebra kullanılarak $J(\Theta_0, \Theta_1)$'i minimum yapan Θ_0 Θ_1 değerleri iteratif çözüme gerek kalmadan bulunabilir. Bunu göreceğiz.

Larger data sets için GDA NEM'e göre daha iyi çalışıyor!

Matrix: Rectangular array of numbers.
A_{ij}: "i,j entry" in the i^{th} row, j^{th} column
Vector: An $n \times 1$ matrix.

Matrix Addition: Aynı boyuttaki matrisleri eleman elemana toplarız.

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0.5 \\ 4 & 10 \\ 3 & 2 \end{bmatrix}$$

Scalar Multiplication:

$$3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} \times 3$$

Matrix Vector Multiplication: $(A_{m \times n}) \times (x_{n \times 1}) = y_{m \times 1}$

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix}_{3\times 2} \begin{bmatrix} 1 \\ 5 \end{bmatrix}_{2\times 1} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}_{3\times 1}$$

$1 \times 1 + 3 \times 5 = 16$
$4 \times 1 + 5 \times 0 = 4$
$2 \times 1 + 1 \times 5 = 7$

Matrix Matrix Multiplication: $(A_{m \times n}) \times (B_{n \times o}) = C_{m \times o}$

Matrix Multiplication Properties:

- $A \times B \neq B \times A$ (not commutative)
- $A \times (B \times C) = (A \times B) \times C$ (Associative)
- $A \cdot I = I \cdot A = A$

Matrix Inverse: If A is an $m \times m$ matrix, and if it has an inverse:

$$A \cdot (A^{-1}) = (A^{-1}) \cdot A = I$$

Matrix Transpose: If $B = A^T$ and $A_{m \times n}$ then $B_{n \times m}$ and

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