- ML W2 -

Multivariate Linear Regression

- Onceki ornekte 1 feature vardı o da ev boyutu idi buna karşılık fiyat bilgisi (output) vardı.
- Şimdi birden fazla feature olan durumlara bakacağız.

Size (feet²) x_1	# of bedrooms x_2	# of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
⋮	⋮	⋮	⋮	⋮

$m = 47$ # of training examples

- n: # of features
- $x^{(i)}$: features of i^{th} training example (a vector)
- $x^{(i)}_j$: value of feature j in i^{th} training example
  - $x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$
  - $x^{(2)}_3 = 2$

Form of the Hypothesis: Önceden 1 feature vardı bu yuzden $h(x) = \theta_0 + \theta_1 x$ idi.

Şimdi ise $h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

E.g. $h(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

For convenience of notation, DEFINE $x_0 = 1$ $(x_0^{(1)} = 1)$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta^T \cdot x = \theta_0 \cdot x_0 + \theta_1 x_1 + \dots \theta_n x_n \quad (x_0 = 1)$$

Form of the Multivariate Linear Regression Hypotheses.

- Gradient Descent For Multiple Variables -

Hypothesis: $h(x) = \theta^T \cdot x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \dots \theta_n \Rightarrow \theta_{n+1 \times 1}$ vector

Cost Function:

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

$\Rightarrow J(\theta)$ ↳ vector

Gradient Descent:

Repeat {

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

} (simultaneous update for every $j = 0, \dots n$)

GDA for Lin. Reg. with 1 feature ($n=1$)

Repeat {

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

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

} Simultaneous update for θ_0, θ_1.

GDA for Linear Regression with Multiple Features ($n \geq 1$)

Repeat {

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

} Simultaneous update for θ_j ($j = 0, \dots n$)

Intuition ⇕

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right) \cdot x_0^{(i)}$ ($x_0^{(i)} = 1$)

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right) \cdot x_1^{(i)}$

$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right) \cdot x_2^{(i)}$

⋮

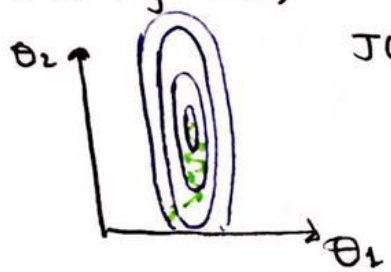
Gradient Descent In Practice I: Feature Scaling -

(*) We will talk about some practical tricks for making GDA works well. First let's talk about Feature Scaling.

Idea: Make sure features are on a similar scale.

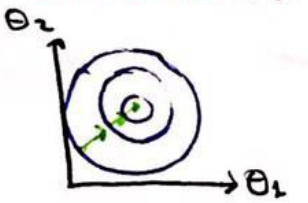
E.g. x_1 = size (0-2000 feet²), x_2 = # of bedrooms (1-5)

Ignoring θ_0, the cost function contour plot would look like:



- Very tall and skinny
- GDA çalıştırılırsa global minimum'un bulunması çok zaman alır.
- Şeklin böyle olmasının sebebi şu: θ_1 deki küçük bir değişim bile x_1 değerleri çok büyük olduğundan büyük cost function değişimleri yaratır.
- GDA iksüreklili oscille olur ve global min'e ulaşmak uzun zaman alır

Solution: • $x_1 = \text{size (feet}^2) / 2000$ • $x_2 = \frac{\#\text{ of bedrooms}}{5}$



- Countor plot ince uzun gönmez yandaki gibi çemberlerden oluşur. Böylece GDA ile global min'e giden yol osile olmayan düzgün bir path olur ve GDA nın sonuca ulaşması daha az zaman alır.

(*) Feature scaling ile $0 \le x_1 \le 1$ ve $0 \le x_2 \le 1$ haline getirdik böylece GDA çok daha hızlı şekilde convenge eder.
(SCALING İÇİN MAX VALUE'ye Bölüdük)

General Feature Scaling

Get every feature into a approximately $-1 \le x_i \le 1$ range.

(*) The numbers -1 and 1 aren't important. (0)-(3) aras. bir x_1 ve (-2)-(0.5) aras. bir x_2 var diyelim sorun olmaz aşağı yukarı (-1)-(1) aralığına yakın tutar.

(*) Ama biri $-100 \le x_3 \le 100$ ise mesela bu sıkıntı olur veya $-0.0001 \le x_4 \le 0.0001$ de sıkıntı olur.

-3 to 3
-1/3 to 1/3

Scaling için feature'ı max value'suna böldük!

-Feature Scaling with Mean Normalization-

Biraz önce Feature Scaling için her feature'ı kendi maximum değerine bölduk.

Burada ise Mean Normalization methodu kullanıcaz

Mean Normalization: Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$)

E.g. • $x_1 = \frac{\text{size} - 1000}{2000}$ (assuming average $x_1 = 1000$)

• $x_2 = \frac{\#\text{ of bedrooms} - 2}{5}$ (Assuming average $x_2 = 2$)

General Form

$$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1} \quad \Big| \quad x_2 \leftarrow \frac{x_2 - \mu_2}{s_2}$$

- μ_1: Avg. value of x_1 in training set
- s_1: Range (max-min) (or standard deviation)

Features bu şekilde yazılınca:

$$-0.5 \leq x_1 \leq 0.5$$

$$-0.5 \leq x_2 \leq 0.5$$

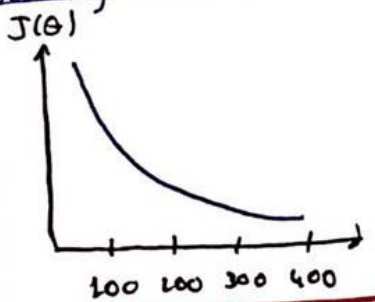
Aralığı gerçekleniz veya buna yakla-şılır. Böylece GDA daha hızlı çalışır!

- Gradient Descent in Practice II: Learning Rate -

GDA nın daha hızlı ve doğru çalışması için ilk iki "feature scaling" idi, ikincisi ise Learning Rate ile ilgili olacak!

1) How to make sure GDA is working correctly?
2) How to choose learning rate, α?

Making sure GDA is working correctly



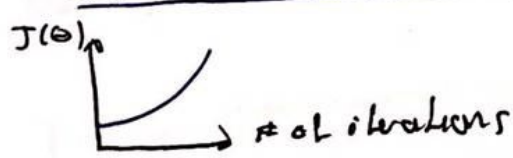
Belirli iterasyonlar sonucunda elde edilen θ'lar ile $J(\theta)$ hesaplabilir. Eğer $J(\theta)$ gideneh azalıyorsa algoritma doğru yolda demektir.

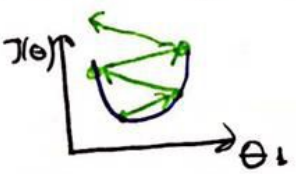
$J(\theta)$ should decrease after every iteration.

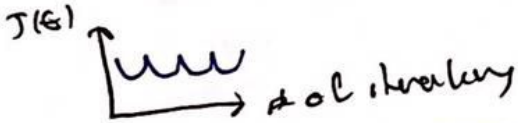
Automatic Converge Test (For $J(\theta)$)

Declare convergence if $J(\theta)$ decreases by less than some small value Σ in one iteration.

If α is too big: GDA may not work:







Result and Method:

- If α is too small: slow convergence
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

α seçmek için: $\alpha = 0.001 \rightarrow 0.003 \rightarrow 0.01 \rightarrow 0.03 \rightarrow 0.1 \ldots$ şeklinde teh teh deneyip **$J(\theta)$ - iterations** grafiğine bakılır. En hızlı converge eden seçilebilir.

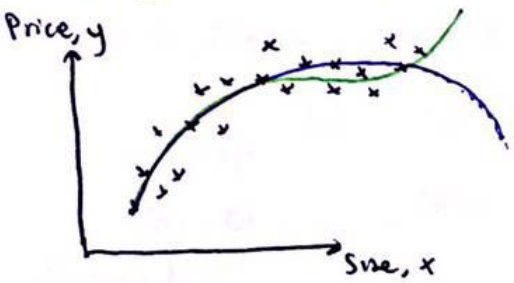
Choice of Features and Polynomial Regression -

Housing Price Prediction: Let's say we wanna predict house pricing:

- $h_\theta(x) = \theta_0 + \theta_1 \cdot \text{frontage} + \theta_2 \cdot \text{depth}$

- İki feature yerine yeni bir Area feature'ı tanımlayabilirim: Area, $x = (\text{frontage}) \cdot (\text{depth})$
- Hypothesis become: $h(x) = \theta_0 + \theta_1 x$

Polynomial Regression:



- Let's say we have training set like figure.
- There are few different models you might fit to this.
  - → $\theta_0 + \theta_1 x + \theta_2 x^2$ (blue)
  - → $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$ (green)

How do we fit a polynomial model to our data? (fit ile kast edilen $\theta_0, \theta_1, \dots \theta_n$ bulmak)

- Multivariant Linear Regression için öğrenilenler küçük bir modifikasyon ile polynomial regression için de kullanılabilir.
- Normalde $h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 x_2 + \theta_3 x_3$ idi
  $= \theta_0 + \theta_1 \cdot (\text{size}) + \theta_2 \cdot (\text{size})^2 + \theta_3 \cdot (\text{size})^3$

Trick: • $x_1 = \text{size}$ • $x_2 = (\text{size})^2$ • $x_3 = (\text{size})^3$ şeklinde tanımladıktan sonra bildiğim adımları kullanıp kübik hypothesis'i eniyileştirebilirim.

Uyarı: Bu durumda feature scaling çok önemli!

Çünkü size: 1-1000 arası ise size^2 1-10^6 size^3 ise 1-10^9 arası yani çok fark var. Feature scaling ile çözmek gerek.

- Yukarıdaki örnek için $h(x) = \theta_0 + \theta_1 \cdot (\text{size}) + \theta_2 \cdot \sqrt{\text{size}}$ kullanılabilir: ↳ böyle bir sonuç verir.
- İleride hangi feature'ların seçilmesi gerektiği ve nasıl fonksiyonlar/modüller kullanacağımızla ilgili algoritmaların da olduğunu göreceğiz.

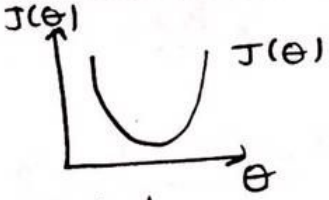
lytically

Normal Equation

NE bazı Linear Regression problemler için optimal parameter values'ı çözmek için daha iyi bir yoldur.

Normal Equation Method: A method to solve for θ analytically.

Intuition:



$J(\theta) = a\theta^2 + b\theta + c$ olduğunu varsayı, sadece 1 parametre olsun.

Bu cost function'ın minimumunu $\frac{d}{d\theta} J(\theta) = 0$ diyip bunu θ için çözerek bulabiliriz.

$J(\theta_0, \theta_1, \dots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$ için ise

$\frac{\partial}{\partial \theta_j} J(\theta) = 0$ (for every j)

$\theta_0, \theta_1, \dots, \theta_n$ için çözülür ve optimum parameters bulunabilir.

Example for Implementing Normal Equation Method:

$m = 4$
$n = 4$

x_0	size (feet^2) x_1	# of bedrooms x_2	# of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1464	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

features

outputs

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1464 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_{m \times 1}$$

$$\theta = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

Octave: `pinv(X' * X) * X' * y`

The resulting θ values will be optimum parameters!

Normal Equation Continue...

- Feature Scaling is NOT NECESSARY for this method!

When to use NE method when to use GDA?

GDA

- Need to choose α
- Need many iterations
- Works well even when n (features) is large

Normal Equation Method

- No need to choose α
- Don't need to iterate
- Since need to compute $(X^T \cdot X)^{-1}$ (nxn matrix), it is slow if n (features) is large.
  - → Until n = 1000 it is okay to use NE → n = 10000 eh? → More would be problem!

Classification algorithms (mesela logistic regression) algoritmaları bahsedince gorecegiz.

NE bunlarda ise yarama :)

Linear Regression ran ise ise yaran!