Week 2

Multiple features

$n \rightarrow$ # of features

$x^{(i)} \rightarrow$ é o i-ésimo vector de features

$x_j^{(i)} \rightarrow$ é o j-ésimo elemento desse vetor.

Hypothesis function → tenta prever, por ex., o preço de uma casa.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \quad (x_0 = 1)$$

for convenience of notation, define $x_0^{(i)} = 1$

→ pense nisso como definir uma zero feature adicional. ($x_0 = 1$)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad n \times 1$$

Feature vector

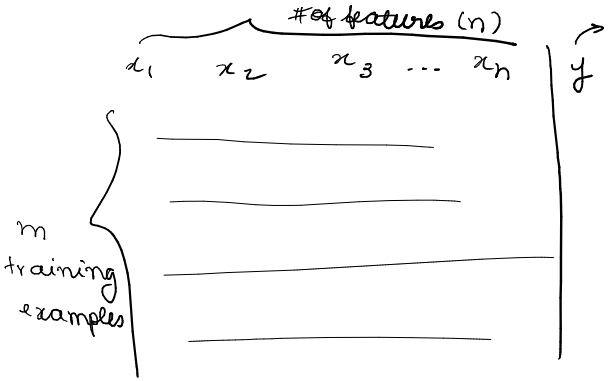
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad n \times 1$$

Parameters vector

$$h_\theta(x) = \theta^T x$$

↳ inner product

multivariate linear regression



Hypothesis function:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

How to fit the parameters?

Como usar GD for linear regression w/ multiple features?

Cost function

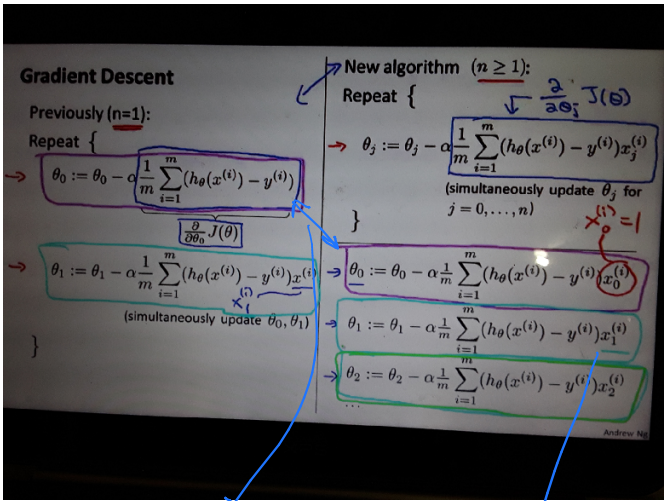
$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

GDescent:

Repeat {

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

} (simultaneously update $\forall j = 0, \cdots, n$)



São coisas idênticas! $x_0 = 1$

2 features, x_1 e x_2

$x_0 = 1$

todos os vetores (i) da 1ª feature

Practical tricks for making GD work well

→ **Feature Scaling** — divide by the range of the input values

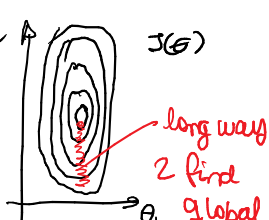
↳ se tem várias features - certifique-se de que elas estão numa escala similar (assumem ranges de valores similares)

↳ assim, GD converge + rápido

Ex: x_1 - assume valores de 0 - 2.000

x_2 - assume valores de 1 a 5

Ai plota cost function $J(\theta)$:



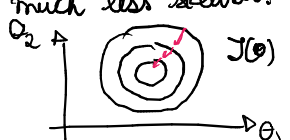
SOLUÇÃO

Scaling the features

$x_1 = \frac{\text{feature 1}}{2.000}$

$x_2 = \frac{\text{feature 2}}{5}$

Contours much less skewed!



Get every feature into approx. $-1 \leq x_i \leq 1$ range.

→ Quero + ou - entre -1 e 1, mas tudo bem se passar um pouco.

$$0 \leq x_1 \leq 3$$

$$-2 \leq x_2 \leq 0{,}5$$

$$-100 \leq x_3 \leq 100$$ → VERY different values!

$$-0{,}0001 \leq x_4 \leq 0{,}0001$$ → very different!

features poorly scaled!

Não precisa estar exatamente na mesma escala!

→ Mean normalization (in addition to feature scaling)

É bom p/ ter suas features perto da média zero.

(Não se aplica à feature zero)

$x_1 = \dfrac{\text{feature 1} - \text{média das features 1}}{2.000}$

range - máx val - min val

$-0{,}5 \leq x_1 \leq 0{,}5$ $\quad -0{,}5 \leq x_2 \leq 0{,}5$

x_1 → replaced by → $x_1 - \mu_1$ → avg val of x_1 in training set

S_1

↳ max - min

ou

Std da variável

Código do curso usarão Std

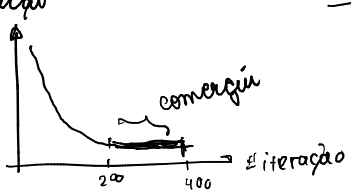
Runs much faster and converges in a lot fewer iterations.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial_j} J(\theta)$$

debugging - como ter certeza q gradient descent tá funcionando bem.

- Como escolher o α

- A fç do GD é achar o θ q minimiza $J(\theta)$
- Plota $J(\theta)$ de acordo com o # de iteração.
- Se tiver funcionando bem, $J(\theta)$ deve cair p/ cada iteração



Automatic convergence test

→ declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration

ε

→ mas é mt difícil achar um ε plausível

↓

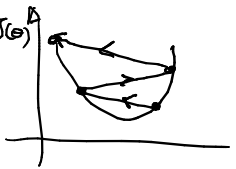
melhor plotar $J(\theta) \times$ # iter.

Casos ruins:

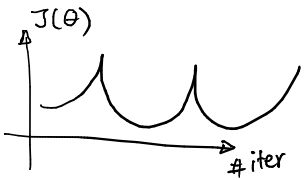
① 

→ GD not working

→ use smaller α



→ ajustar o passo

② 

J'm getting worse and worse

→ use smaller α

α mt pequeno tb demora um século p/ convergir

→ Sugestão: tentar um range de values p/ α

$\alpha = 0,001$ | $0,003$ | $0,01$ | $0,03$ | $0,1$ | $0,3$ | ⋮

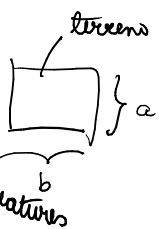
} Plotar $J(\theta)$ × #iter

↓

Pegue o valor de α que parece fazer $J(\theta)$ cair + rapidamente.

Acho um valor bom de α e aí tento diminuir um pouco p/ achar um bom learning rate p/ meu problema.

Choosing appropriate features



Suponha q tenho 2 features: a e b

→ posso criar minhas próprias features

$\underbrace{\text{área}}_{\text{nova feature}} = a \cdot b$

$h_\theta(x) = \theta_0 + \theta_1 \underbrace{x}_{\text{área}}$

Regressão Polinomial

→ uma linha não parece ser um bom modelo.

$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x_1 + \theta_2 x^2 + \theta_3 x^3$$

forma da 1ª hipótese: (com novas features agora)

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \rightarrow \text{linear}$$

$$\theta_0 + \theta_1 (\text{size}) + \theta_2 (\text{size})^2 + \theta_3 (\text{size})^3$$

x_1 x_2 x_3

Aplico o método do linear regression e ajusto um polinómio.

feature scaling becomes increasingly important!

size: 1 — 1000

size^2: 1 — 1000 000

size^3: 1 — 1000 000 000

features taking very ≠ ranges!

Em vez de ir p/ um modelo cúbico por ex, posso recombinar as features e permanecer

Tem algoritmos q escolhem p/ vc que features usar.

Normal Equation

- p/ alguns problemas de reg. linear, é uma forma melhor de resolver p/ o valor ótimo dos parâmetros θ.

- antes eu tava usando o gradient descent. minimizava cost function $J(\theta)$, podia levar mtas iterações.

- normal equation: método p/ resolver p/ θ analiticamente.

~~Algoritmo iterativo~~ → resolve p/ o θ ótimo.

Continua ...