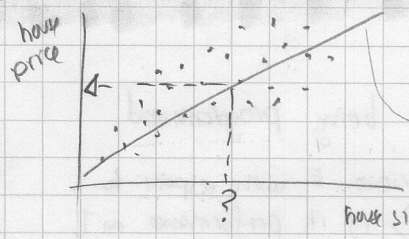
Linear regression of One Variable



→ Supervised learning → we have a data-set of price/size (if discrete, it would be a classification problem)

→ Regression problem → predict continuous output

size	price
2104	460
1416	232
1534	315
⋮	⋮

m: number of training examples

(x, y) - one training example

$(x^{(2)}, y^{(2)})$ - 2nd training example

x: input variable (features)

y: output variable (target)

Model

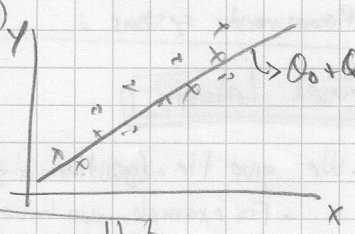
Training Set → Learning Algorithm → h (hypothesis function)

size → h → estimated value! (price)

Hypothesis function (one variable):

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

($h_\theta(x)$ for short; θ_0, θ_1: model parameters)



Linear Regression with one variable or Univariate linear regression

How do we find θ_0 & θ_1 parameter values?

- Informally, we want θ_0, θ_1 so that $h(x)$ is close to y for the training examples (x, y)
- Formally:

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

where $h_\theta(x^{(i)}) = \theta_0 + \theta_1 \cdot x^{(i)}$

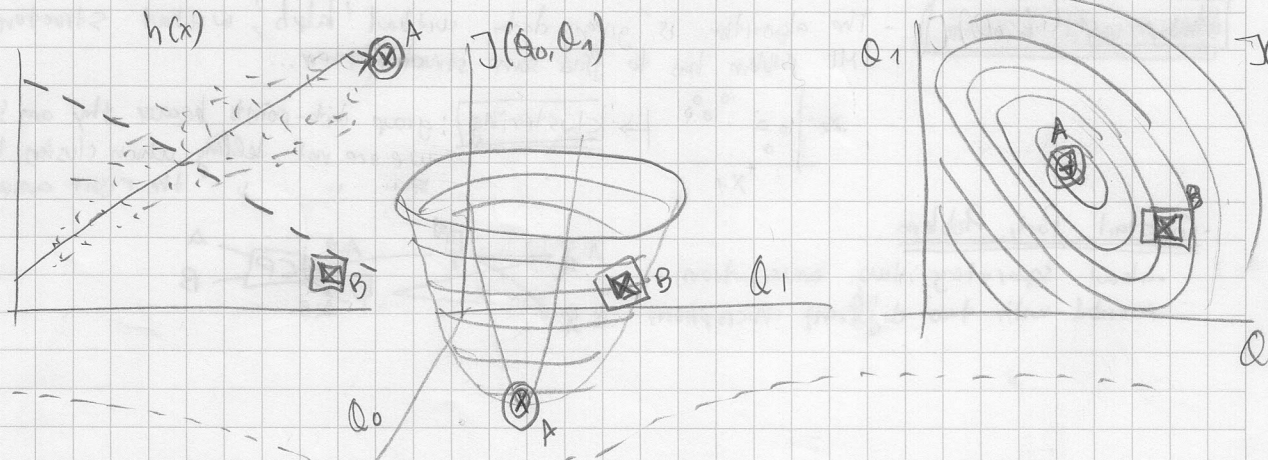
} Objective function to minimize

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

} Cost function to minimize

Square Error function → most common for regression problems

another notation



$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2} (h_\theta(x) - y)^2}_{Cost(h_\theta(x), y)} = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x), y)$$

→ sum, over m, of Cost

Cost: is what we want the algorithm to penalize; in this case we penalize with the square error

Gradient Descent

Algorithm to minimize, here used to minimize $J(\theta_0, \dots \theta_n)$ and get the right $\theta_0, \dots \theta_n$ parameter values that better match the regression line and the data-set.

(in general)

Idea
1. initialize θ_0, θ_1 (to 0, for example)
2. keep changing θ_0, θ_1 so $J^{(\theta_0, \theta_1)}$ is reduced ~···~> minimum of J will be found

Formally:

repeat until convergence {

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

}

- $:=$ assignment operator (like '=')
- α learning rate
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ partial derivative
- $\frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x), y) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y)^2$

simultaneously update θ_0, θ_1

temp0 := $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} \dots$

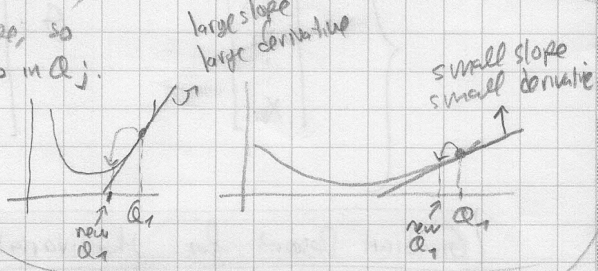
temp1 := $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} \dots$

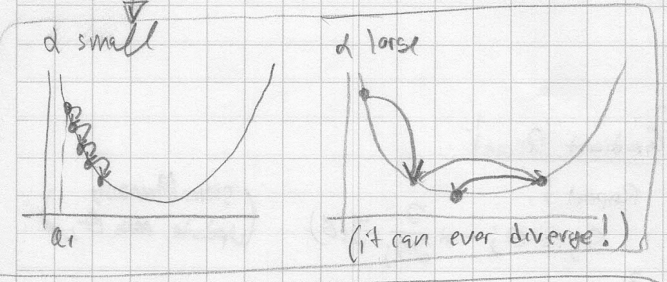
θ_0 := temp0

θ_1 := temp1

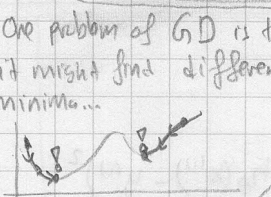
When the derivative is 0, we stop.
- it was the minimum!
- θ_j does not evolve

The larger the derivative, the larger the slope, so the larger the jump in θ_j.





▽ One problem of GD is that it might find different local minima...



we don't have to change the value of α as we advance, the derivative part already results in smaller steps the closer to the minimum we get

GD for Univariate LR

repeat until convergence { $\theta_0 + \theta_1 \cdot x^{(i)}$

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

$$\theta_1 := \theta_1 - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (\underbrace{h_\theta(x^{(i)}) - y^{(i)}}_{\theta_0 + \theta_1 \cdot x^{(i)}}) \cdot x^{(i)}$$

}

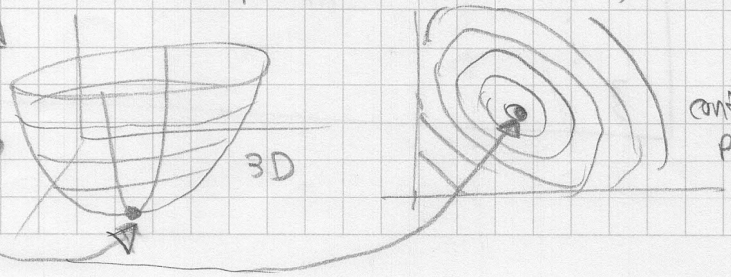
(update θ_0, θ_1 simultaneously)

→ called "Batch" gradient descend because at each step we use the entire set of training samples ($\sum_{i=1}^{m}$ ...)

▽ Applying Gradient Descent to Univariate Linear Regression
- does not have the local minima problem because the $J(\theta)$ function is Bowl-shaped

⇓

single local minimum = global minimum



Gradient Descent in Univariate Linear Regression

Linear Regression of Multiple Variables (or Multivariate LR)

Multiple variables = Multiple features → house size, num of floors, num of dorms, ...

size	dorms	floors	years	price
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
⋮	⋮	⋮	⋮	⋮
X_1	X_2	X_3	X_4	Y

Vector $X^{(2)}$ ← row 2

$n = 4$ features

$X_3^{(2)}$

Y: output variable to predict

Hypothesis $\Rightarrow h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_3 + \theta_4 \cdot x_4$ (size, dorms, floors, years)

$\theta_0 \cdot x_0$, $x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \Rightarrow h_\theta(x) = \theta^T \cdot X = [\theta_0 \theta_1 \dots \theta_n] \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Gradient Descent for Multivariate LR

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

parameters → θ

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

Gradient Descent:

Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (simultaneously update each θ_j)

}

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$ (simultaneously update for all j)

}

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot \underbrace{x_0^{(i)}}_{1}$$

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

$$\theta_2 := \theta_2 + \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

⋮

}

(simultaneously update all θ_j)

Practical hints

- feature scaling so every feature fits aprox. into $-1 \leq x_i \leq +1$

$$x_i \leftarrow \frac{x_i}{max}$$

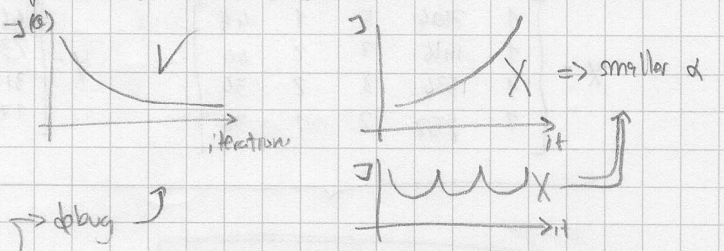
- mean normalization → so every feature has aprox. zero mean

$$x_i \leftarrow \frac{x_i - \mu_i}{s_i}$$

s_i → range, or standard dev.

Gradient Descent evolves better

- debugging → plots of the value of $J(\theta)$ versus iterations



- value of α? → debug

try → 0'001 (x3) → 0'003 (x3) → 0'01 (x3) → 0'03 (x3) → 0'1 (x3) → 0'3 (x3) → 1 ... } plot $J(\theta)$ vs iteration

than fine tune → 0'003 (x3) → 0'01 (x3) → 0'03 (x3) → 0'1 (x3) → 0'3 (x3) → 1 ... } plot $J(\theta)$ vs iteration

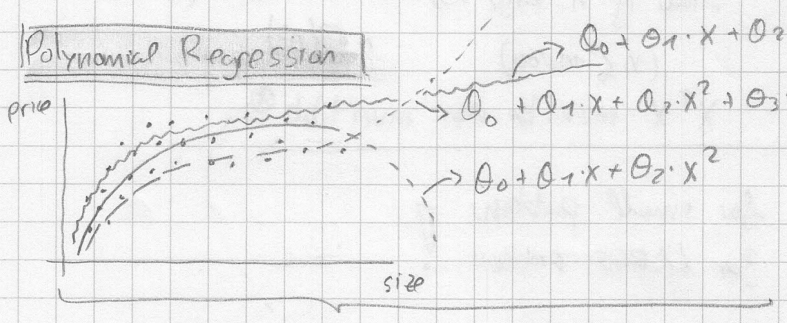
Choosing the right features (x_i)

- If we have width and height, better work with area

$$h_\theta(x) = \theta_0 + \theta_1 \cdot w + \theta_2 \cdot h \Rightarrow h_\theta(x) = \theta_0 + \theta_1 \cdot a$$

} combining features

Polynomical Regression



We can obtain this regression with the Multivariate Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_3 =$$
$$= \theta_0 + \theta_1 \cdot (size) + \theta_2 (size^2) + \theta_3 (size^3)$$

$x_1 = size$

$x_2 = size^2$

$x_3 = size^3$

} be careful with scaling! apply feature scaling!

Normal Equation

- Method to solve for θ analytically (alternative to Gradient Descent Algorithm)
  - ↳ in one 'mathematical' step

	size	rooms	floors	years	price
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
x_0	x_1	x_2	x_3	x_4	y

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} \cdot X^T \cdot y$$ ⇝ θ values that minimize $J(\theta)$

Gradient Descent		or		Normal Equation ?
Need to choose α	X		V	No parameters to choose
Needs many iterations	X		V	No iterations
Feature scaling	X		V	No need to scale features
Works well for large n (features) ($n \gtrsim 10000$)	V		X	Slow if n large ⇝ Needs to compute $(X^T X)^{-1}$ ($n \lesssim 10000$) ~ $O(n^3)$
	-		X	$X^T X$ might be non invertible (*)

Normal Equations for small problems !
Gradient Descent for LARGE problems !

(*) - octave 'pinv' will manage it
- if it has redundant features
- if $m \leq n$ ↳ reduce features (n) ↳ regularization
  - m ↑ num data experiments