Introduction to ML
Lecture 4: Linear Regression

Hamed Tabkhi
Department of Electrical and Computer Engineering,
University of North Carolina Charlotte (UNCC)
htabkhiv@uncc.edu

Linear regression

- In regression problem, the task is:
  approximate a mapping function (h) from input variables (x) to continuous output variables

- In this lecture, we work on linear regression models

Linear regression

Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on $X_1, X_2, \ldots X_p$ is linear.

$$y = \underbrace{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots}_{\text{linear combination}} \beta_k x_k$$

response or dependent variable

linear combination

predictor or independent variables

β unknown model parameters are estimated from the data

UNC CHARLOTTE

Linear regression

- A linear regression model follows a very particular form. In statistics, a regression model is linear when all terms in the model are one of the following:
  - The constant
  - A parameter multiplied by an independent variable (IV)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k$$

Quiz (1)

- Linear or non-linear regression model

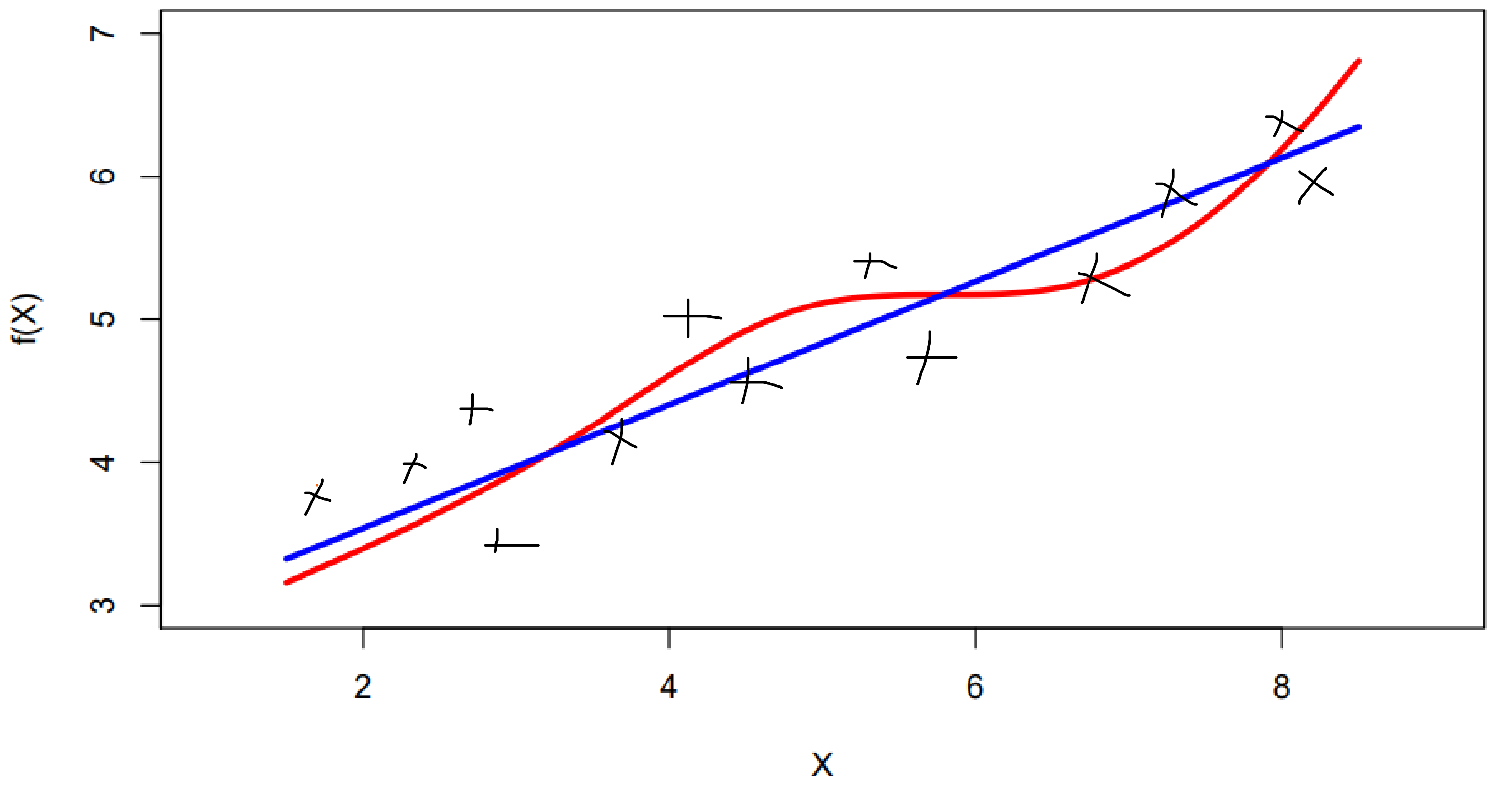
$$\theta_1 * X^{\theta_2}$$ non-linear

$$\theta_1 * cos(X + \theta_4) + \theta_2 * cos(2 * X + \theta_4) + \theta_3$$ non-linear

$Y = b_o + b_1X_1 + b_2X_1^2$ Non-linear

UNC CHARLOTTE

Quiz (2)



Q: Are the true regression functions always linear ?

Linear regression

- Why should we assume that relationships between variables are linear?
  - Because linear relationships are the simplest non-trivial relationships that can be imagined (hence the easiest to work with), and.....
  - Because the "true" relationships between our variables are often at least approximately linear over the range of values that are of interest to us, and...
  - Even if they're not, we can often transform the variables in such a way as to linearize the relationships.

Linear regression

- Let's start with an example

Predict house price

Training set

(x ft², $ y K)

(3883 ft², $432K)

(1668 ft², $218K)

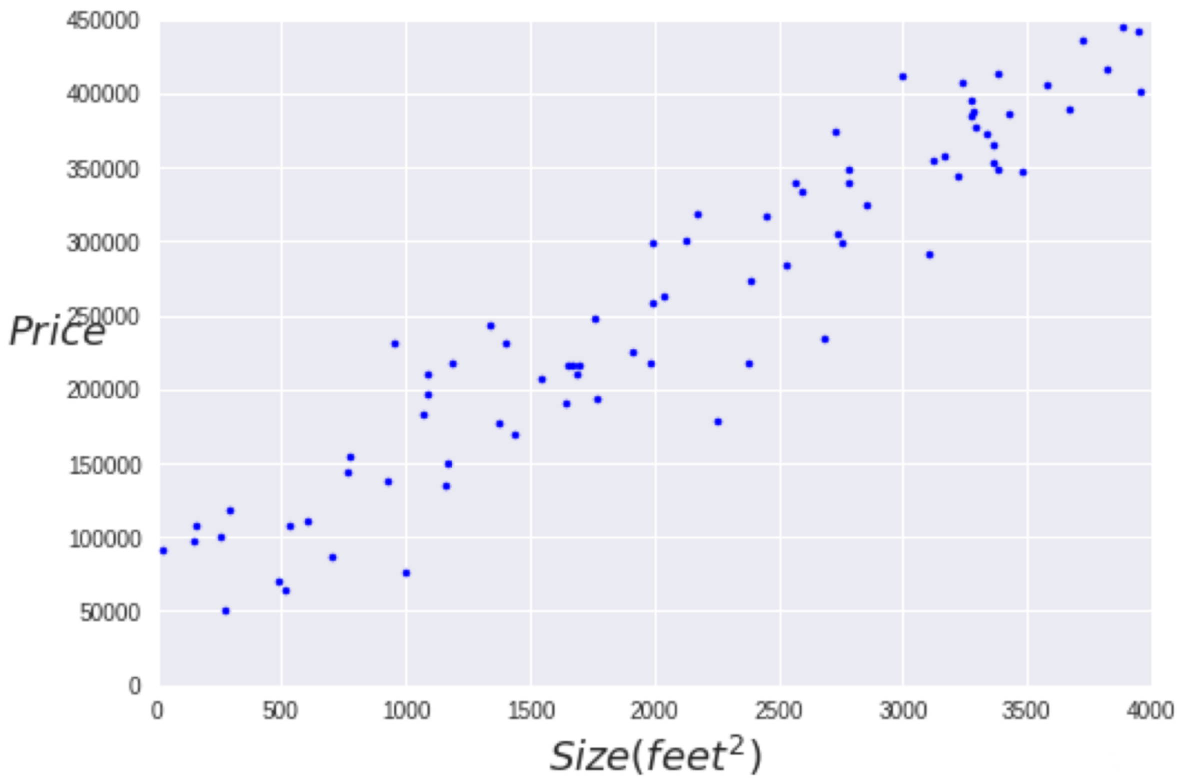
(3577 ft², $366K)

(1668 ft², $218K)

(765 ft², $123K)

(3822 ft², $493K)

⋮

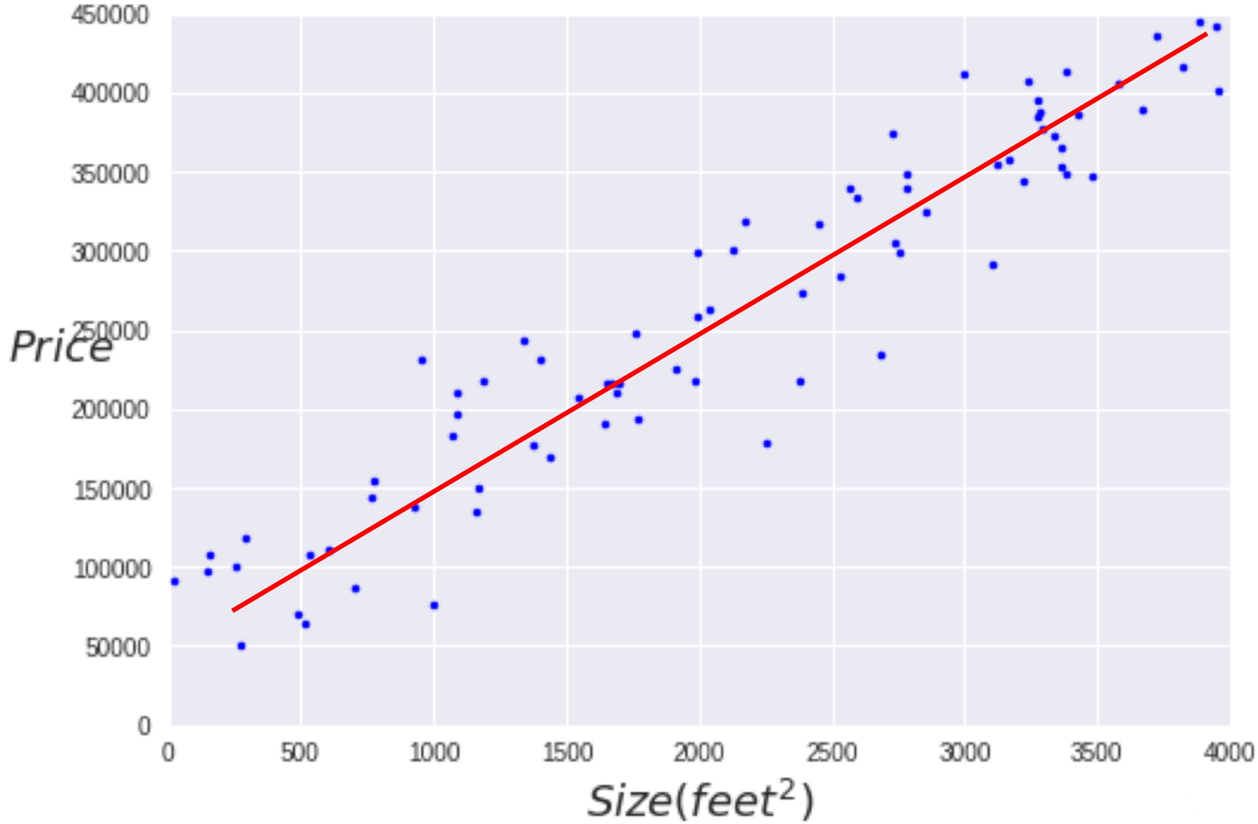


Credit: the following slides adapt from Andrew Ng and Behnam Kia

UNC CHARLOTTE

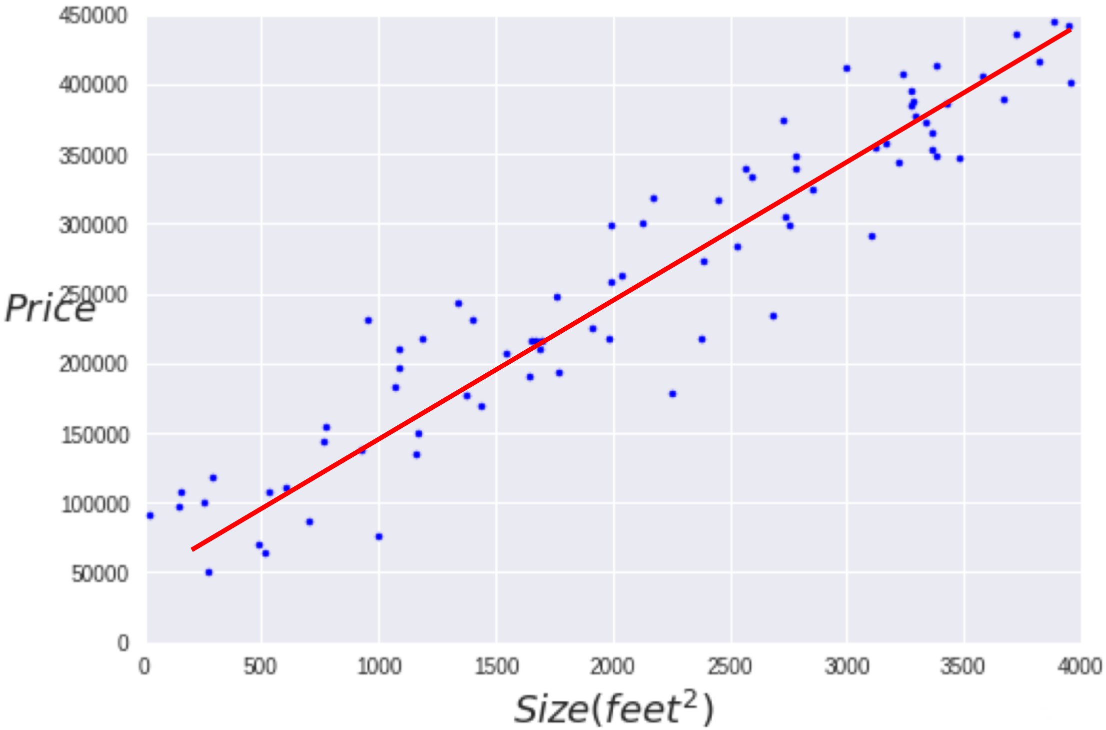
Linear regression

- Linear regression with one variable (x)



Linear regression

- Linear regression with one variable (x) – simple linear regression



Linear function

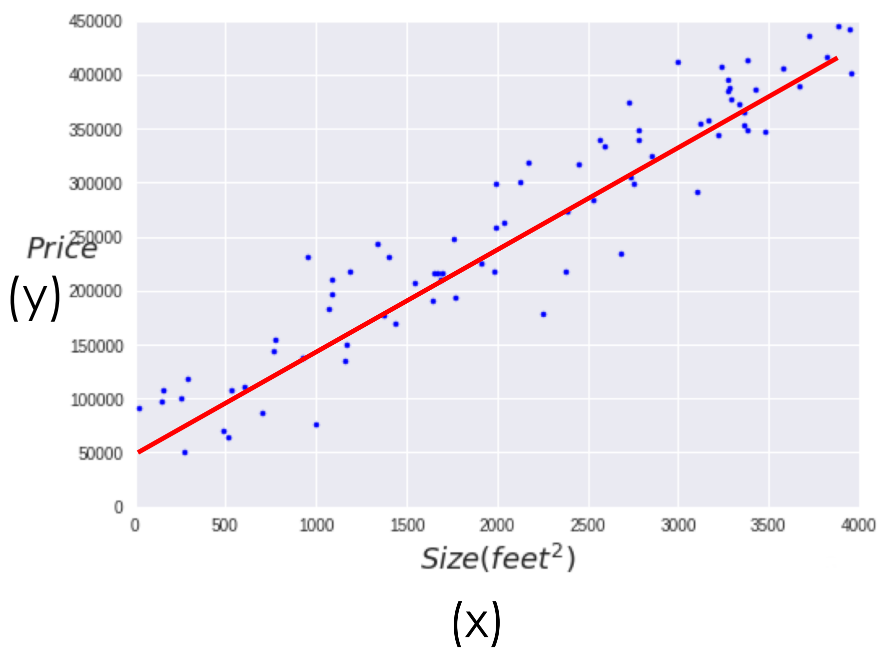
$$h(x) = \theta_0 + \theta_1 x$$

θ_0, θ_1 are parameters

Linear regression

- How to choose the parameters?

- Idea: choose θ_0, θ_1 so that h(x) is close to y for our training examples (x, y)



Linear regression

- Let's formalize the problem

Goal: Obtaining a Linear Regression Model

Input: x (the size of house)

Target: y (the price of the house),

Linear model:

$$h = \theta_0 + \theta_1 x$$

UNC CHARLOTTE

Linear regression

- **Cost Function: How well or poorly a model (Hypothesis) explains the training data**

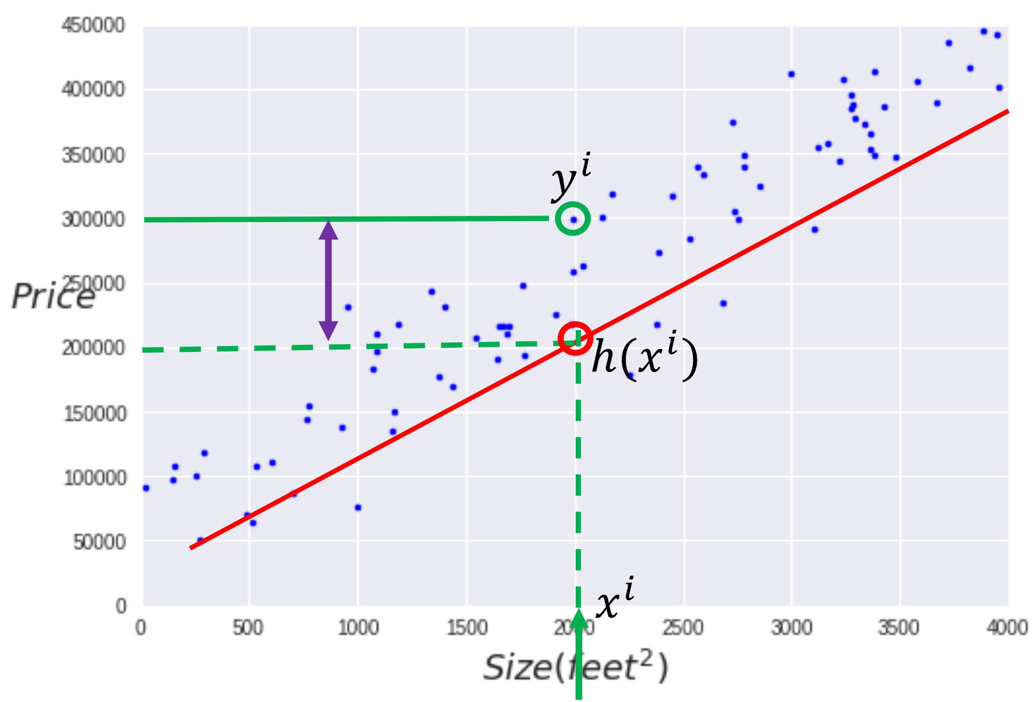
Sum of the squared Euclidean distance

$J(\theta_0, \theta_1) = \sum_i \left(h(x^i) - y^i\right)^2$ (x^i, y^i) is the ith training sample

$$h(x^i) = \theta_0 + \theta_1 x^i$$

$i \in [1, m]$

m: total number of training samples



Linear regression

- Problem to solve

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

Parameters: θ_0, θ_1

m: total number of training samples
½: mathematical convenience

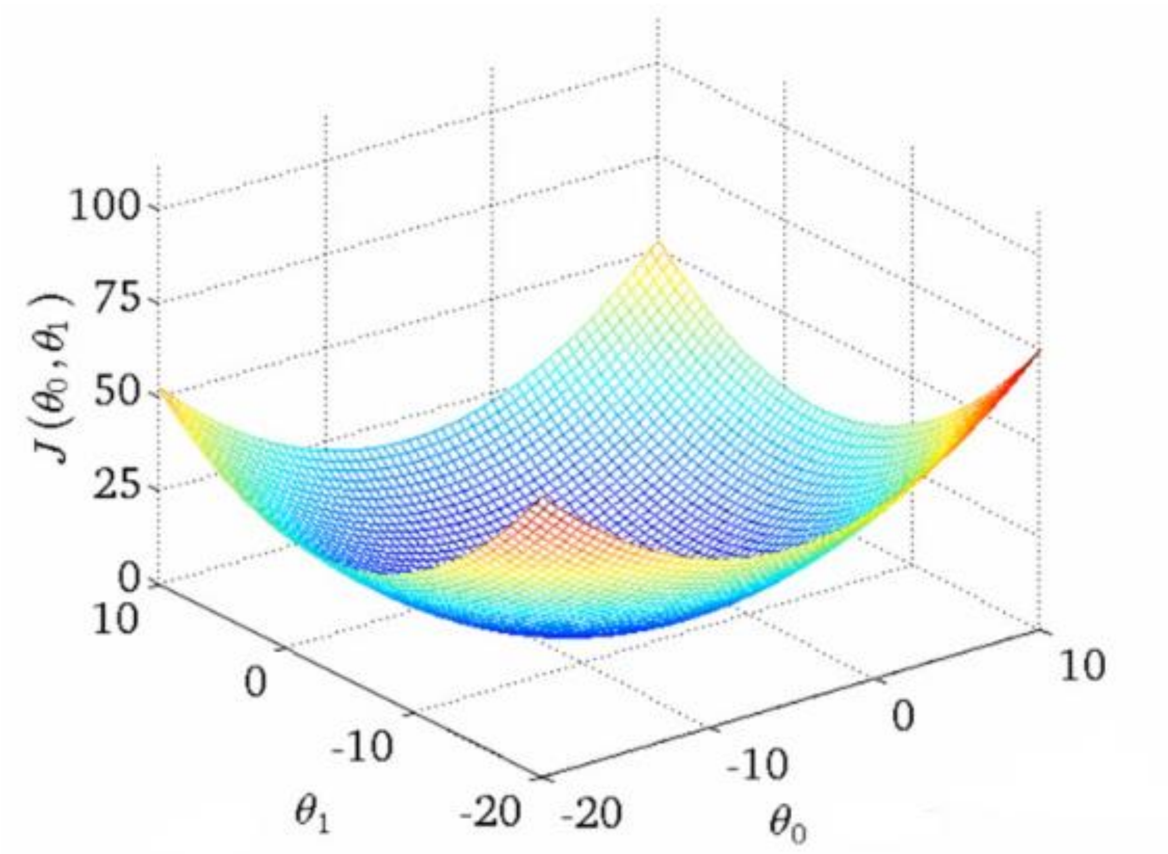
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Cost function

Linear regression

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



Linear regression

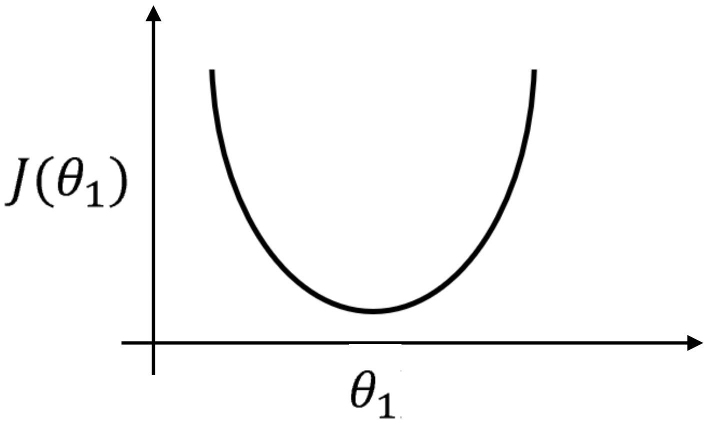
- Simplified hypothesis *h* for easier visualization

$Set\ \theta_0 = 0$

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

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

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



a **quadratic polynomial**

UNC CHARLOTTE

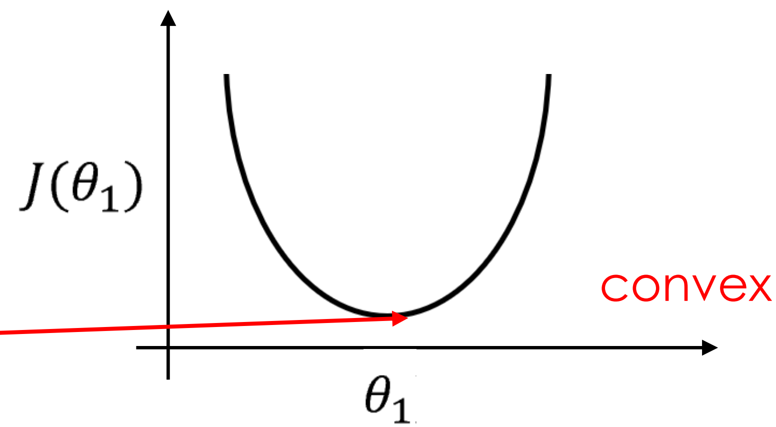
Gradient descent algorithm

- How to find the minimum point of cost function?
  - Gradient descent
    - Used all over machine learning for minimization

- But wait

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

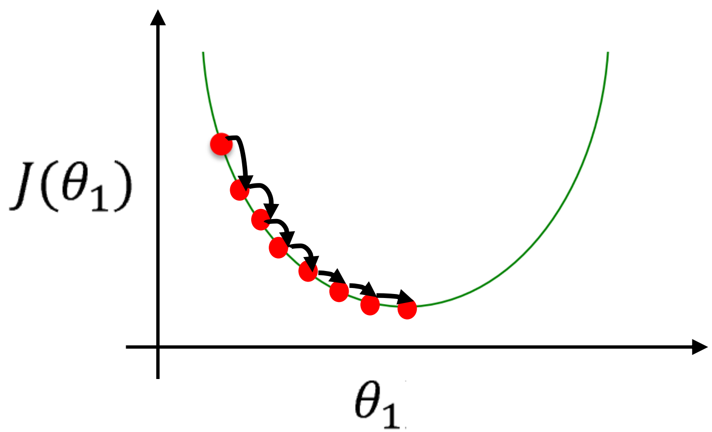
$$\frac{dJ(\theta_1)}{d\theta_1} = 0$$



Analytical Method

Gradient descent algorithm

- Start with an initial guess
- Keeping changing θ_1 a little bit to try and reduce $J(\theta_1)$



Gradient descent algorithm

Have a cost function $J(\theta_0, \theta_1)$

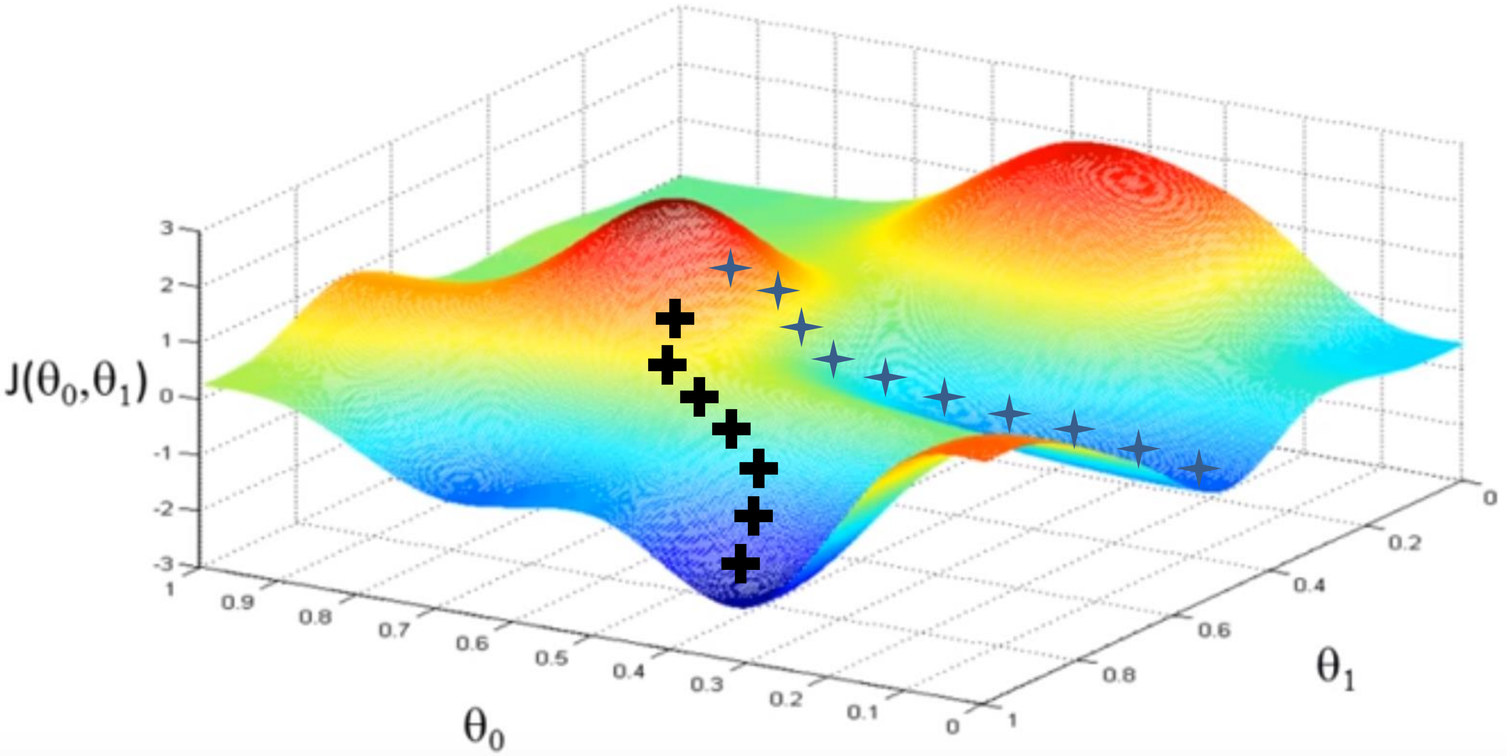
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- 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 a minimum

Gradient descent algorithm



Gradient descent algorithm

- A more formal definition

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

next

current

Gradient: partial derivative

α is learning rate (a positive constant)

Note: j is an index for parameters

Correct: Simultaneous update

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

UNC CHARLOTTE

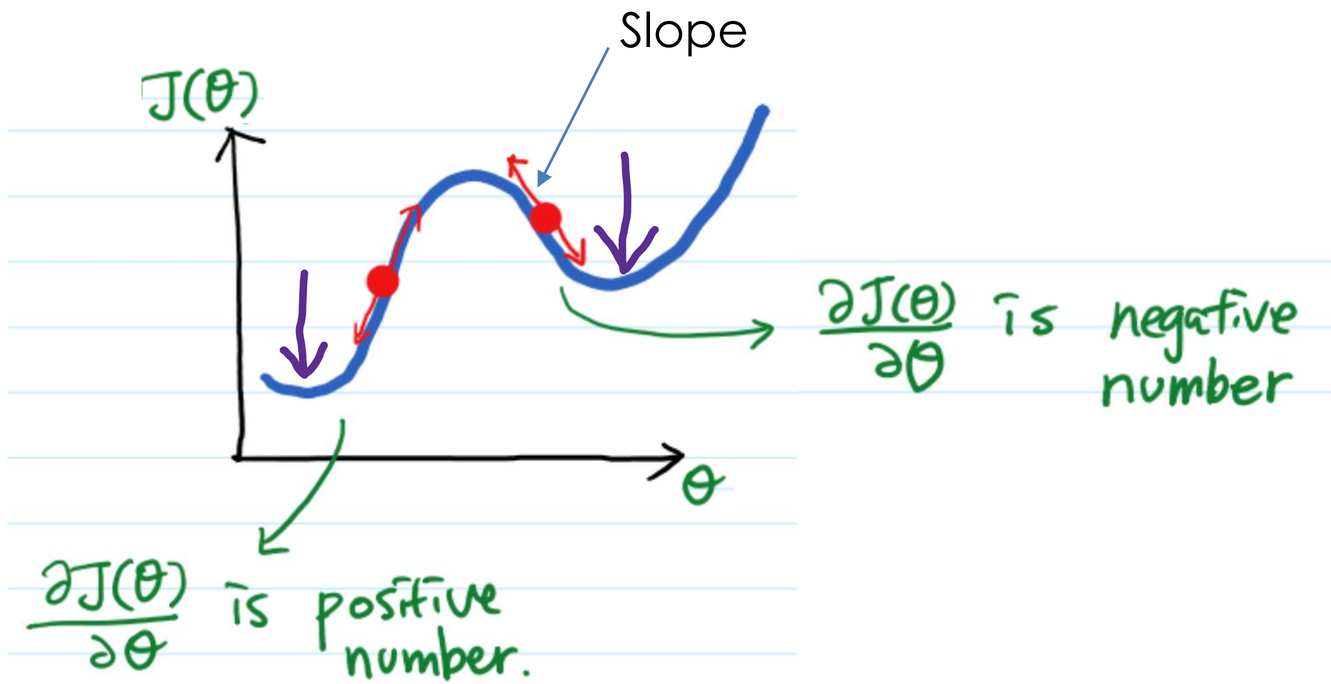
Gradient descent algorithm

- Q1: Why “ - ” the gradient in parameter update?
- Q2: How to set a proper value for the learning rate α ?

Gradient descent algorithm

- Why " - " the gradient in parameter update?
- Note that a gradient is a vector, so it has both of the following characteristics:
  - a direction
  - a magnitude
- The gradient always points in the direction of steepest increase in the loss function.
- The gradient descent algorithm takes a step in the direction of the negative gradient in order to reduce loss as quickly as possible.

Gradient descent algorithm

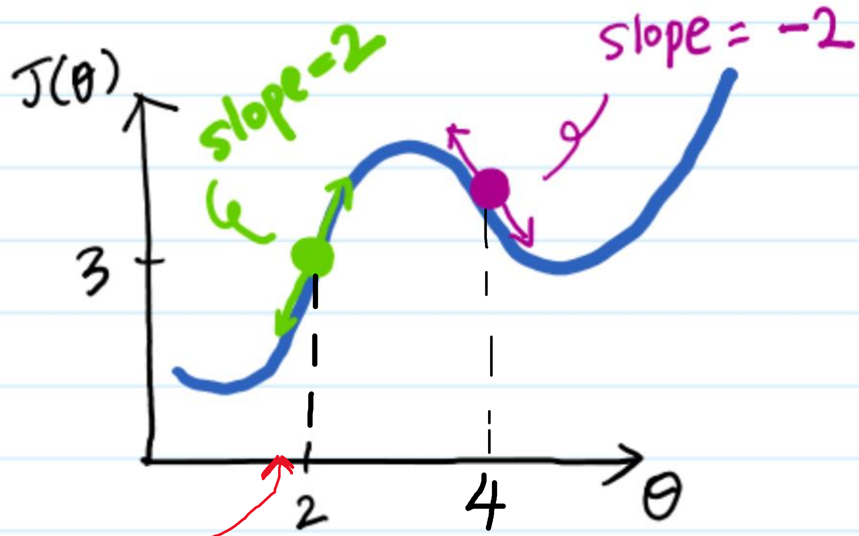


$$\theta = \theta - \alpha \cdot \text{slope}$$

The learning rate α is a positive constant

UNC CHARLOTTE

Gradient descent algorithm



$$\theta = \theta - \alpha \cdot \text{slope}$$

1.9 $\quad \theta_{new} = 2 - \alpha \cdot (2) \quad (\theta_{old} = 2)$

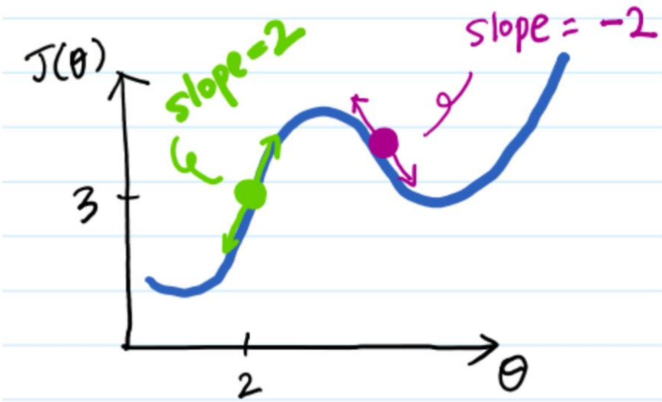
4.1 $\quad \theta_{new} = 4 - \alpha\,(-2) \quad (\theta_{old} = 4)$

$\alpha = 0.05$

UNC CHARLOTTE

Gradient descent algorithm

- How to choose a proper learning rate α ?

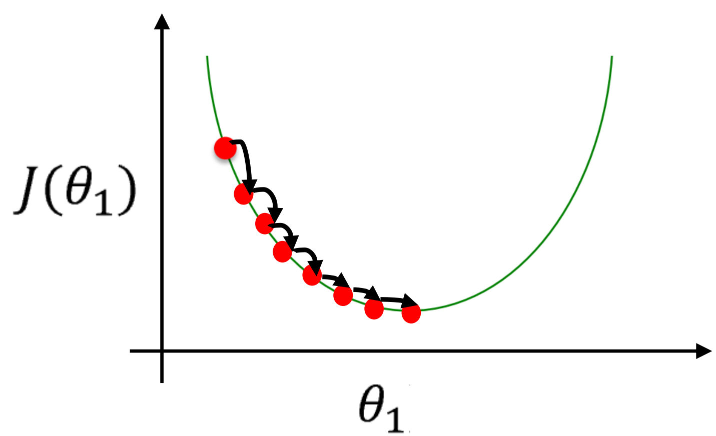


$\theta = \theta - \alpha \cdot \text{slope}$

1.9 $\theta_{new} = 2 - \alpha \cdot (2) \quad (\theta_{old} = 2)$

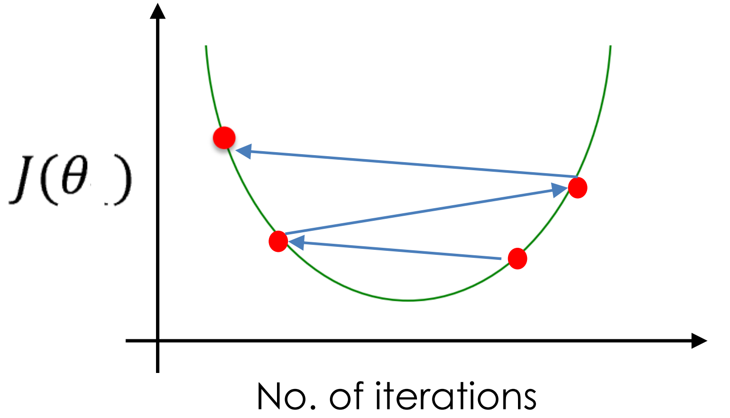
4.1 $\theta_{new} = 4 - \alpha\,(-2) \quad (\theta_{old} = 4)$

$\alpha = 0.05$



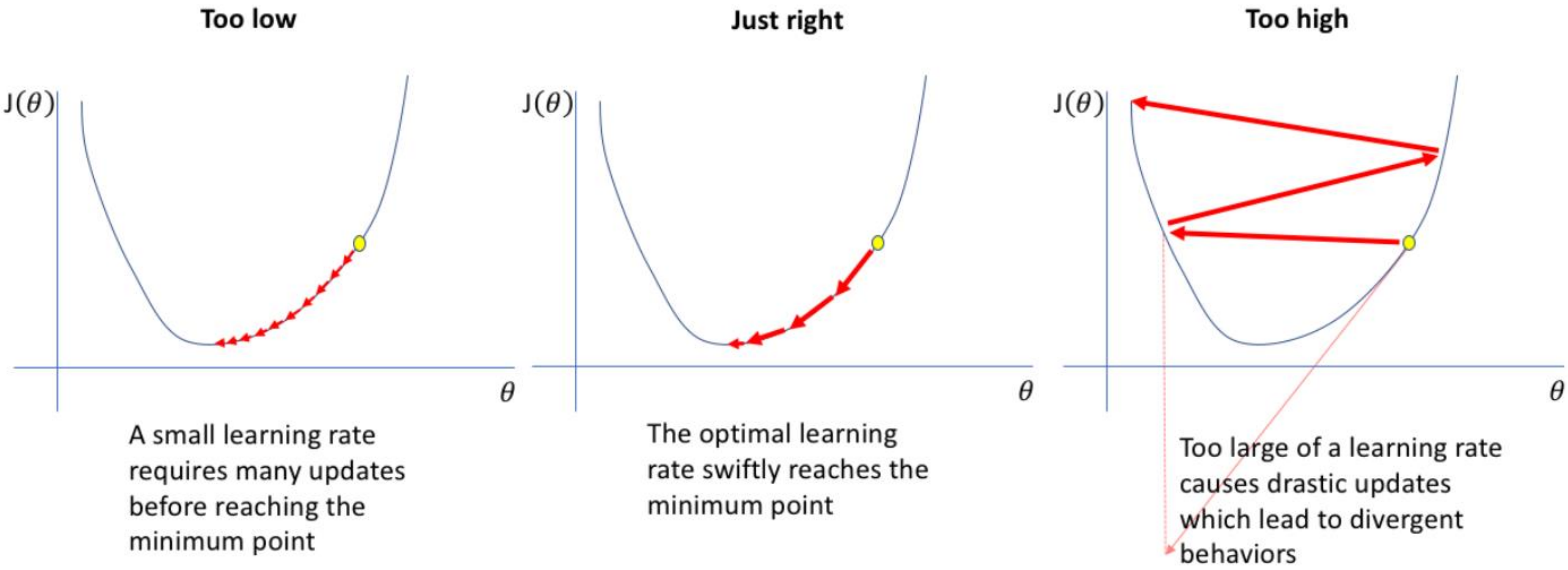
Gradually approach the minimum

Gradient descent algorithm



If the learning rate is too large, gradient decent can overshoot the minimum. It may fail to converge, or even diverge.

Gradient descent algorithm



Gradient descent algorithm

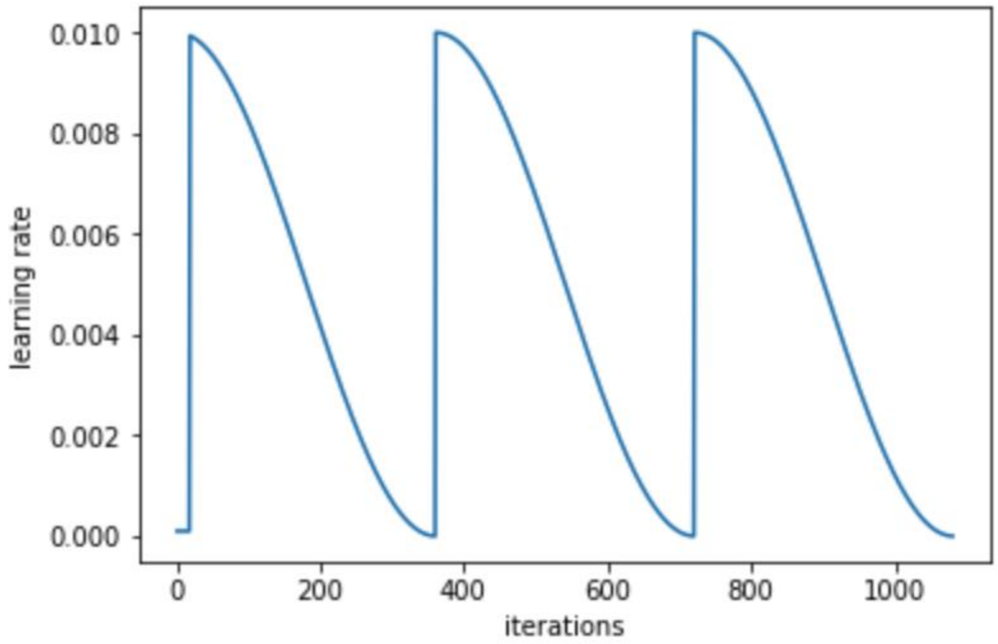
- Fixed learning rate
  - determined by trial and error
  - E.g., try a few values 0.1, 0.01, 0.001, …, i.e., parameter tuning via grid search
- To see if gradient descent is working, print out $J(\theta)$ each iteration
  - The value should decrease at each iteration
  - If it doesn't, adjust α

Gradient descent algorithm

- Recommended reading on learning rate

https://medium.com/@lipeng2/cyclical-learning-rates-for-training-neural-networks-4de755927d46

Learning rate annealing, initially large, decrease gradually



Cyclical Learning Rates for Training Neural Networks

Quiz

Statement
If the learning rate is too small, then gradient descent may take a very long time to converge.
If θ_0 and θ_1 are initialized at a local minimum, then one iteration will not change their values.
Even if the learning rate α is very large, every iteration of gradient descent will decrease the value of $f(\theta_0,\theta_1)$.
If θ_0 and θ_1 are initialized so that $\theta_0=\theta_1$, then by symmetry (because we do simultaneous updates to the two parameters), after one iteration of gradient descent, we will still have $\theta_0=\theta_1$.

Linear regression with multiple variables

More input variables (features)

Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	x_n	Price ($1000) y
2104	5	1	45	…	460
1416	3	2	40		232
1534	3	2	30		315
852	2	1	36		178

Each row is an n-dimensional data point (sample)

UNC CHARLOTTE

Linear regression with multiple variables

- Previous hypothesis or mapping function

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

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

For convenience of notation, define $x_0 = 1$.

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

Each data point is now a (*n+1*)-dimensional vector

UNC CHARLOTTE

Linear regression with multiple variables

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

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

m: total number of training samples

Cost function:

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

i-th sample

Corresponding label or GT value

- **Gradient descent**

Repeat {

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

}

(simultaneously update for every $j = 0, \ldots, n$)

i: index for sample

j: index for parameter

UNC CHARLOTTE

Linear regression with multiple variables

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ $x_0 = 1$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

m: total number of training samples

Cost function:

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

Repeat {

$$\theta_j := \theta_j - \alpha \underbrace{\boxed{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}}_{}$$

$\frac{\partial}{\partial \theta_j} J(\theta)$

(simultaneously update θ_j for $j = 0, \ldots, n$) }

i: index for sample

j: index for parameter

Linear regression with multiple variables

- Stopping criterion
  - Assume convergence when $||\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}||_2 < \boxed{\epsilon}$

A small constant (threshold)

  - Set a maximum number of iterations

Useful resource

- Gradient and partial derivatives
  - https://www.youtube.com/watch?v=GkB4vW16QHI

- A very detailed tutorial on Gradient Descent (Step-by-Step) with examples
  - https://www.youtube.com/watch?v=sDv4f4s2SB8