MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 10. Linear Regression

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail: jchun@mju.ac.kr

Lecture Note: http://lms.mju.ac.kr

Agenda

- Classification concept review
- Linear Regression
- Logistic Regression

Classification Concept Review

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Regression vs. Classification

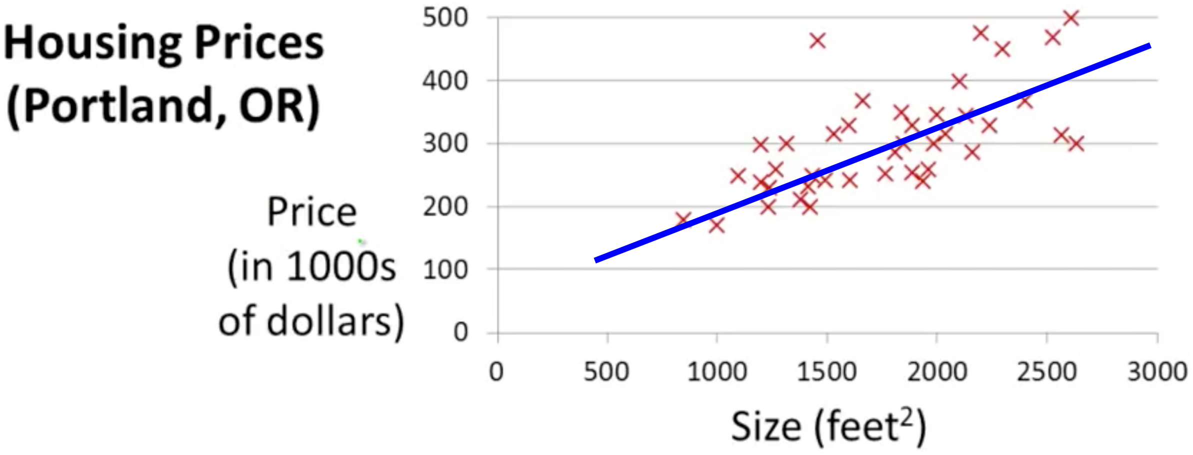
- Regression problem
  - predict results within a continuous output
  - map input variables to some continuous function
  - E.g., Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output.

- Classification problem
  - predict results in a discrete output
  - map input variables into discrete categories (e.g., positive, negative reviews)
  - E.g., Given a patient with a tumor, predict whether the tumor is malignant or benign.

LINEAR REGRESSION

Regression Problem – an example

- Predict Housing Prices



<u>Supervised Learning</u>

Given the "right answer" for each example in the data.

<u>Regression Problem</u>

Predict real-valued output

Classification: Discrete- Valued

Training Set

- Training set of housing prices

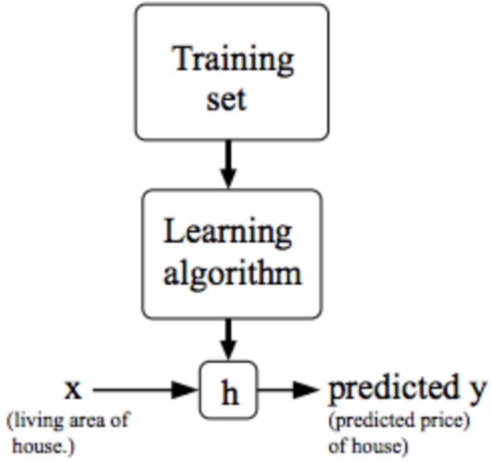
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

$m = 47$

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

Supervised Learning Problem

- Our goal is,
  - Given a training set, to learn a function h : X → Y so that h(x) is a "good" predictor for the corresponding value of y.
- For historical reasons, this function h is called a hypothesis.



How do we represent *h*?

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

Shorthand: $h(x)$

Linear regression with one variable

Univariate linear regression

When the target variable is continuous,
we call the learning problem a **regression** problem!

Cost Function

- Problem to solve
  - how to fit the best possible straight line to our data

Training Set

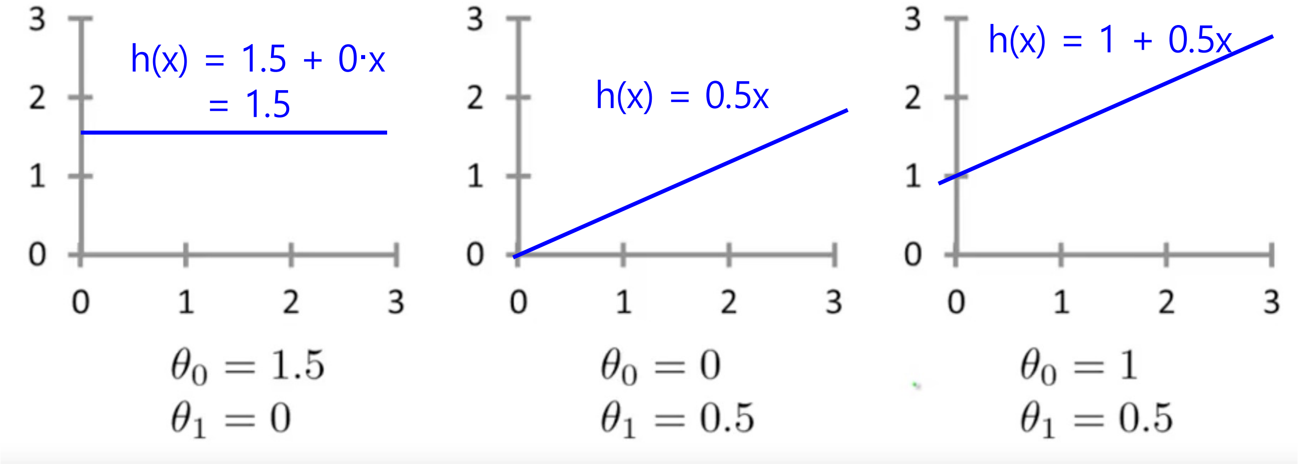
Size in feet^2 (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

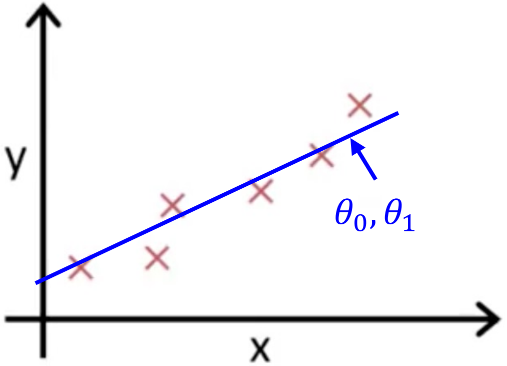
  - θ_i's : Parameters
  - How to choose θ_i's?

Cost Function

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



Cost Function



$$\underset{\theta_0, \theta_1}{\text{Minimize}} \ \frac{1}{2m} \sum_{i=1}^{m} (\underbrace{h_\theta(x^{(i)})}_{h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}} - y^{(i)})^2$$

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

$$\underset{\theta_0, \theta_1}{\text{Minimize}} \ \underbrace{J(\theta_0, \theta_1)}$$

Cost Function Squared Error Function

- Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training samples (x, y)

Our Goal

- Have some function $J(\theta_0, \theta_1)$

  We want $\min\limits_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

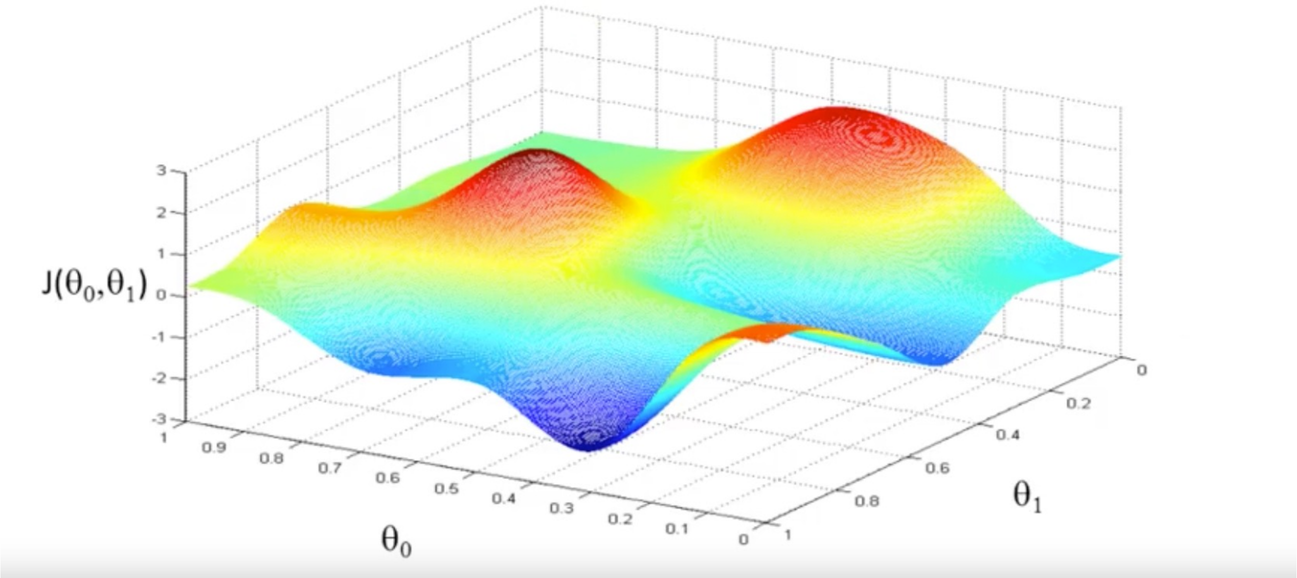
$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

$\min\limits_{\theta_0,\ldots,\theta_n} J(\theta_0, \ldots, \theta_n)$

- Outline:
  - Start with some θ_0, θ_1 (e.g., $\theta_0 = 0, \theta_1 = 0$)
  - Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

An algorithm called "Gradient Descent" for minimizing the cost function J for for the linear regression!

Gradient Descent



Gradient Descent Algorithm

repeat until convergence {

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

}

Learning Rate

Simultaneously update
θ_0 and θ_1

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

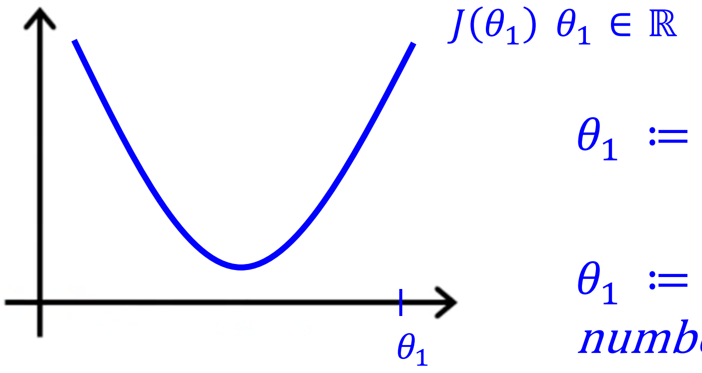
Gradient Descent Algorithm Intuition

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

- Assume a simpler case, we want to minimize the cost function with just one parameter

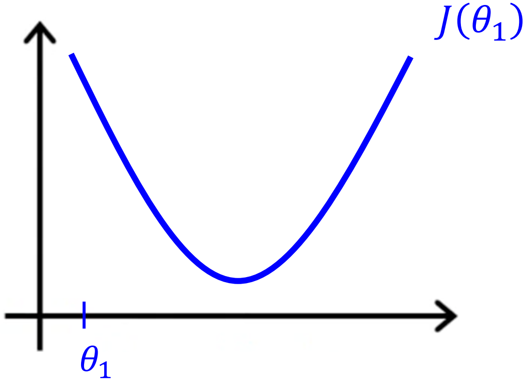
$$\min_{\theta_1} J(\theta_1), \;\; \theta_1 \in \mathbb{R}$$

Gradient Descent Algorithm Intuition



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

$$\theta_1 := \theta_1 - \alpha \cdot (\textit{positive number})$$



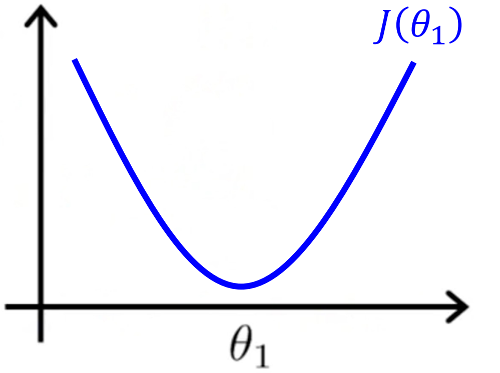
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha \cdot (\textit{negative number})$$

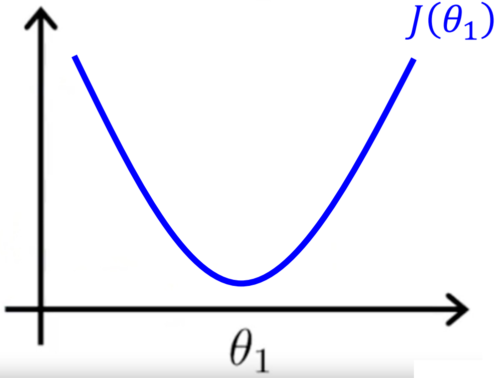
Learning Rate

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

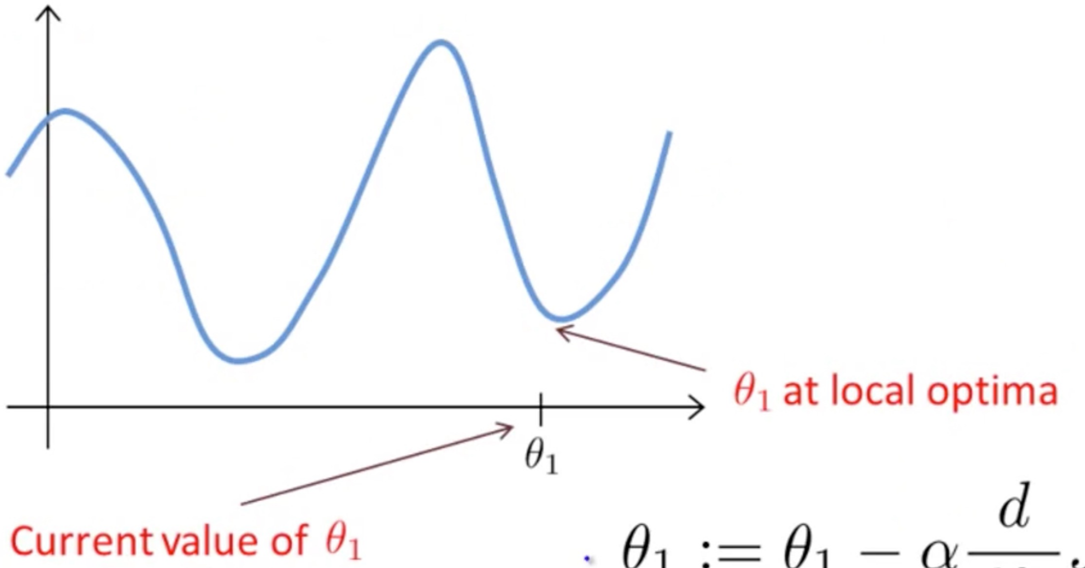
If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Local Minimum Problem

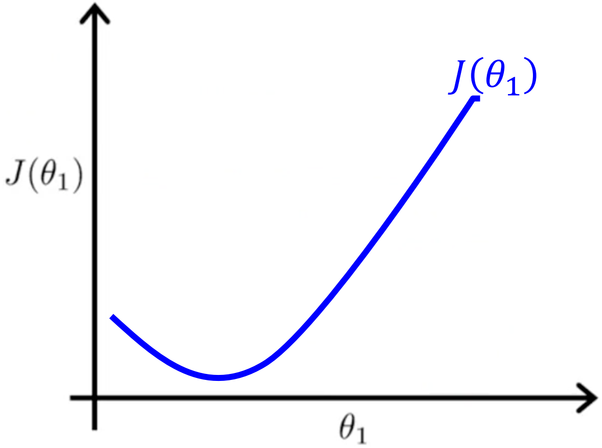


$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Local Minimum Problem

- Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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



As we approach a local minimum, gradient descent will automatically take smaller steps. So no need to decrease α over time.

Gradient Descent for Linear Regression

Gradient descent algorithm

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

Linear Regression Model

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

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

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

Gradient Descent for Linear Regression

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

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

- $j = 0 : \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$
- $j = 1 : \frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}$

Gradient Descent for Linear Regression

repeat until convergence {

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

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

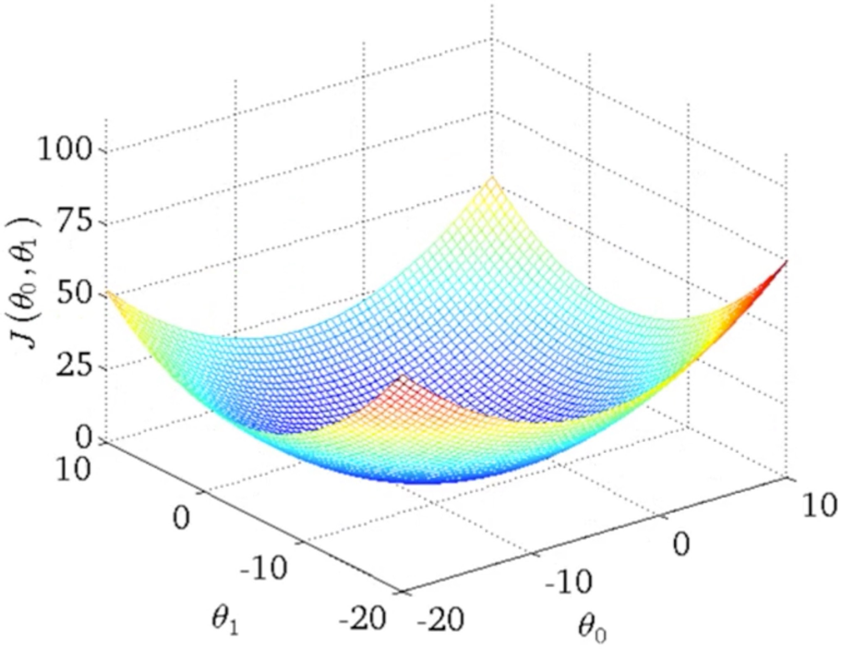
}

update
θ_0 and θ_1
Simultaneously

"Batch" Gradient Descent

"Batch": Each step of gradient descent uses all the training examples.

Gradient Descent for Linear Regression



Cost function for linear regression is always going to be a **"Convex Function"**!

Remarks

- Solving for the minimum of the cost function
  - Gradient descent: an iterative algorithm
  - Normal equation method: numerically solving for the minimum of the cost function without needing the multiple steps of gradient descent

- Gradient descent will scale better to larger data sets than that normal equation method.

- Generalization of gradient descent algorithm
  - Will make it much more powerful

MULTIVARIATE LINEAR REGRESSION

Multiple Features (variables)

Size (feet²) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
…	…

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

Multiple Features (variables)

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

- Notation
  - n = number of features
  - $x^{(i)}$= input (features) of i^{th} training example
  - $x_j^{(i)}$= value of feature j in i^{th} training example

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

Multivariate Linear Regression

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

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

e.g., $h_\theta(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_3 - 2x_4$

- General form:

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

For convenience of notation, define $x_0 = 1$

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

Multivariate Linear Regression

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

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

$$\mathbb{R}^{n+1}$$

$$\underbrace{[\theta_0\ \theta_1 \ldots \theta_n]}_{\theta^T} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

(n + 1) X 1 matrix

Gradient Descent

Previously (n=1):

Repeat {

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

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

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

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

}

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

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

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

$\ldots$

ACKNOWLEDGMENTS

- Many of the slides and examples are borrowed from the open course by Coursera “**Machine Learning**” by A. Ng which were modified into their current form.

END