Lecture 2

Linear Regression

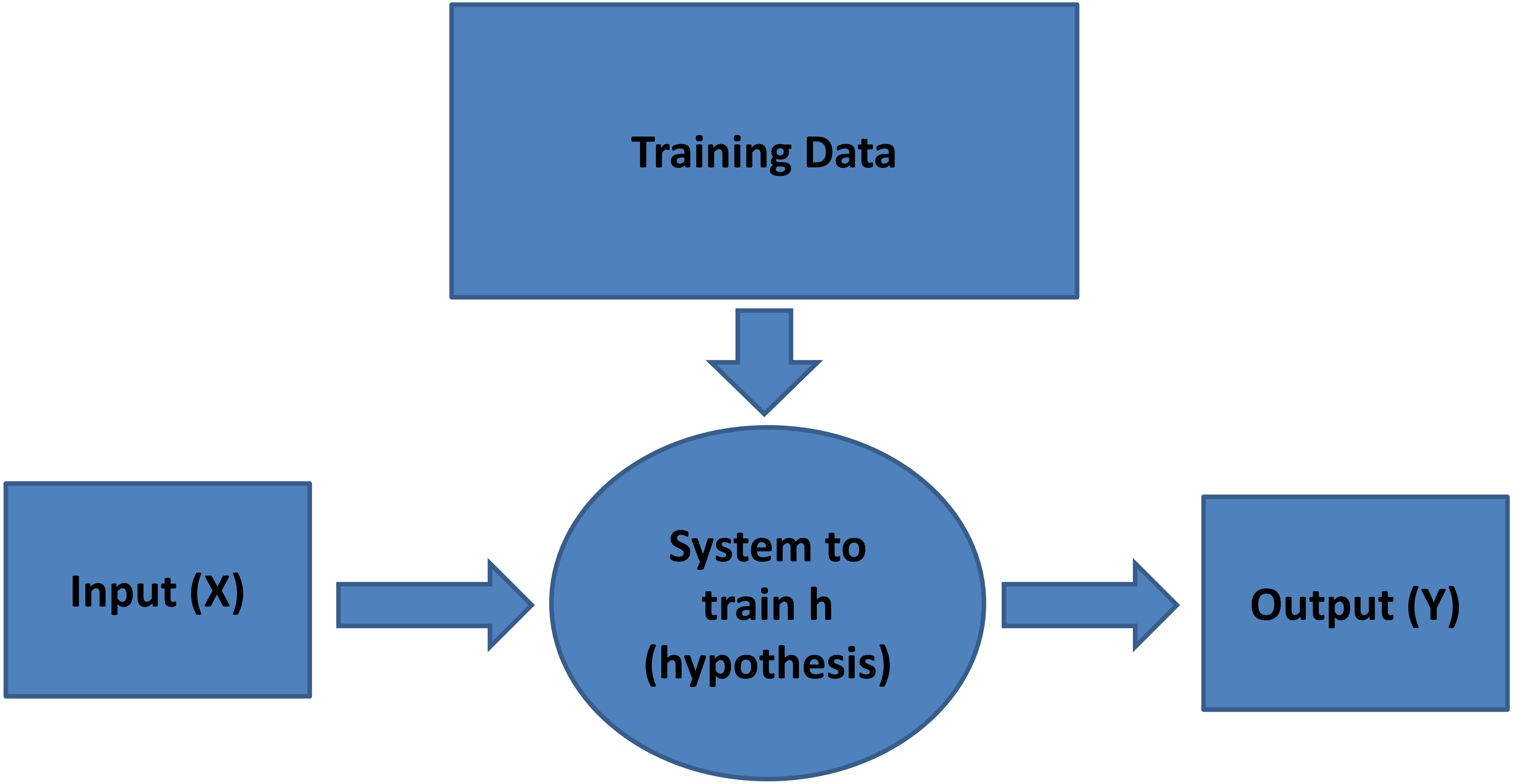
Dr. Le Huu Ton

Hanoi, 09/2016

Outline

- Linear Regression
- Gradient Descent?
- Multi Feature Representation
- Questions and Answers

Review of Machine Learning



For each input *x*, output *y* = *h* (*x*)

Linear Regression

$(x^{(2)}, y^{(2)})$

Size (m^2)	Price (billion VND)
30	2.5
43	3.4
25	1.8
51	4.5
40	3.2
20	1.6

$(x^{(5)}, y^{(5)})$

Table 1: Training data of housing price in Hanoi

In Supervised Learning, each training data consists of 2 elements: input x (features) and output y (response)

Notation

(x,y) : One training data

$(x^{(i)}, y^{(i)})$: the i^{th} training data

Linear Regression

Plotting of training data

Linear Regression: Assume that the output ***y*** is a linear function of input ***x***

$$y=h(x)=a*x+b$$

Linear Regression

Objective:

Learning the function *y=h(x)=a*x+b*, such as it returns the minimize error- cost function for the training data (optimization problem)

Find the coefficient *a,b* to minimize the cost function

Cost Function:

The error for training data:

Size(m^2)	Price (b.VND)
30	2.5
43	3.4
25	1.8
51	4.5
40	3.2
20	1.6

$$e^{(1)} = \frac{1}{2}(h(x^{(1)}) - y^{(1)})^2 = \frac{1}{2}(30a + b - 2.5)^2$$

$$e^{(2)} = \frac{1}{2}(h(x^{(2)}) - y^{(2)})^2 = \frac{1}{2}(43a + b - 3.4)^2$$

.

.

.

$$e^{(m)} = \frac{1}{2}(h(x^{(m)}) - y^{(m)})^2 = \frac{1}{2}(x^{(m)}a + b - y^{(m)})^2$$

The cost function is define as:

$$E = \frac{1}{m}(e^{(1)} + e^{(2)} + \ldots + e^{(m)}) = \frac{1}{m}\sum_{i=1}^{m} e^{(i)} = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

$$E = \frac{1}{2m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})^2$$

Gradient Descent

Objective:

Use the gradient function to find a minimum of a function

$$E = \frac{1}{2m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})^2$$

Note that E is a function of a and b, we have only 2 variables a and b.

Idea:

Choose random number for a and b, the algorithm is implemented in many steps. At each step, modify a and b such as the cost function is reduced

$$a := a - \alpha \frac{\partial}{\partial_a} E(a) \qquad b := b - \alpha \frac{\partial}{\partial_b} E(b)$$

Gradient Descent

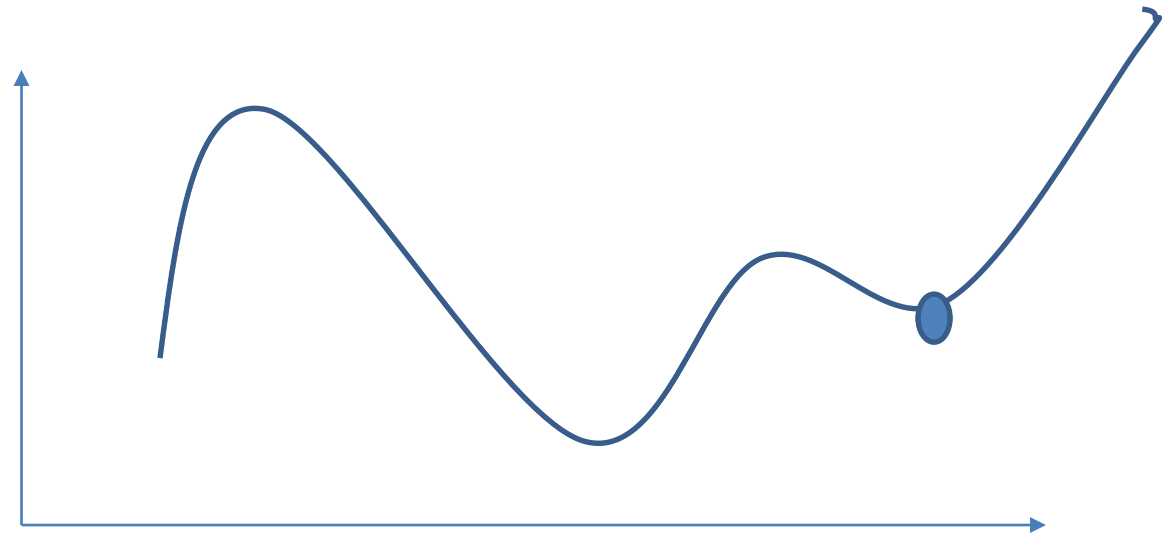
A demonstration and explanation of Gradient Descent algorithm can be found at the following website:

http://www.onmyphd.com/?p=gradient.descent

Gradient Descent

Suppose that (x0,y0) is a local minimum of the cost function, what will 1 iteration of gradient descent do?

1. **Leave x0 unchanged**
2. **Change x0 in random direction**
3. **Move x0 toward the global minimum**
4. **Decrease x0**



Gradient Descent

Calculate the derivations of the cost function: $\frac{\partial}{\partial_a} E(a)$ $\frac{\partial}{\partial_b} E(b)$

$$E = \frac{1}{2m} \sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)})^2$$

Given the following formula:

$$\frac{\partial}{\partial_x}(x^2) = 2x$$

$$\frac{\partial}{\partial_x}(f(x)^2) = 2f(x).\frac{\partial}{\partial_x} f(x)$$

Gradient Descent

$$\frac{\partial}{\partial_a} E(a) = \frac{1}{m} \sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)}).x^{(i)}$$

$$\frac{\partial}{\partial_b} E(b) = \frac{1}{m} \sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)})$$

Gradient Descent

Exercise:

Starting at a=0 and b= 0, α=0.01, what is the cost function?

Calculate the value of a and b after first iteration (first step). Confirm if the cost function is reduced or not?

Size(m^2)	Price (b.VND)
30	2.5
43	3.4
25	1.8
51	4.5
40	3.2
20	1.6

$$a := a - \alpha \frac{\partial}{\partial_a} E(a) \qquad b := b - \alpha \frac{\partial}{\partial_b} E(b)$$

$$\frac{\partial}{\partial_a} E(a) = \frac{1}{m} \sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)}).x^{(i)}$$

$$\frac{\partial}{\partial_b} E(b) = \frac{1}{m} \sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)})$$

Gradient Descent

Batch and Stochastic Gradient Descent

Batch gradient Descent:

Compute the Gradient Descent using the whole data set

Stochastic Gradient Descent:

Compute the Gradient Descent using 1 training example at a time

- Randomly reorder the training data
- Use $(x^{(1)},y^{(1)})$ to calculate the gradient descent in order to update a,b
- Use $(x^{(2)},y^{(2)})$ to update
- ……

Gradient Descent

<u>Mini-Batch Gradient Descent</u>

Compute the Gradient Descent for t example at a time ($1<t<m$)

Example:

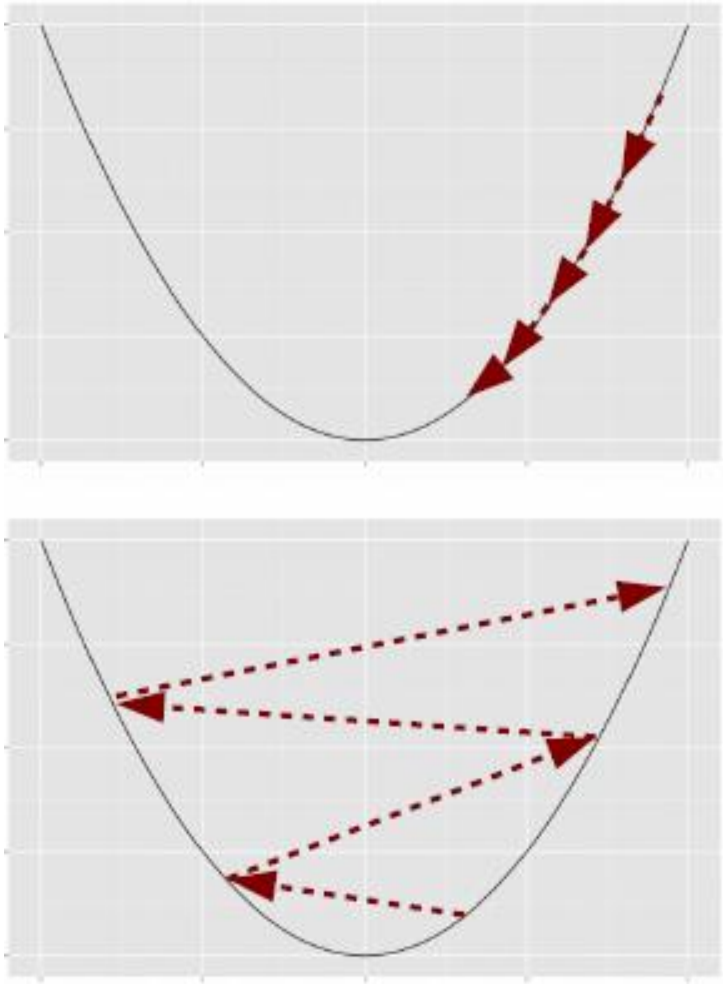
We have 1000 training data.

Step 1: Update the coefficient using 10 data 1-10

Step 2: Update the coefficient using 10 data 11-20

….

Gradient Descent



Big value of α may lead to the incensement of cost function and not convergence

⇒ Gradient Descent works if the cost function decrease at each step

Gradient Descent

Convergence:

How to know if a function is converged or not?

- Cost function is smaller than a predefined threshold
- After a big enough number of step
- Cost function decreased less than a predefine threshold

Gradient Descent

Summarization:

1. **Calculate the cost function**
2. **Select random value for coefficient a,b**
3. **Step by step modify a, b such as the cost function is decreased**

While (not converged)

do

$$a := a - \frac{\partial}{\partial_a} E(a) \qquad b := b - \frac{\partial}{\partial_b} E(b)$$

Multiple Input Representation

<u>Example:</u>

Consider the same example, but with more inputs

Size (m^2)	N^0 of floors	N^0 of rooms	Price (billion VND)
30	3	6	*2.5*
43	4	8	*3.4*
25	2	3	*1.8*
51	4	9	*4.5*
40	3	5	*3.2*
20	1	2	*1.6*

$x^{(i)}$: the input of i^{th} training data

$x_j^{(i)}$: the component j of i^{th} training data

Multiple Input Representation

Matrix representation:

$$y = h(x) = \theta_0 + \theta_1 + \theta_2 + \ldots + \theta_n$$

$$x = \begin{vmatrix} 1 \\ x_1 \\ \ldots \\ x_n \end{vmatrix} \qquad \theta = \begin{vmatrix} \theta_0 \\ \theta_1 \\ \ldots \\ \theta_n \end{vmatrix}$$

$$h(x) = [\theta_0 \; \theta_1 \; \theta_2 \ldots \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \theta^T x$$

Multiple Input Representation

Cost Function:

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

Multiple Input Representation

<u>Gradient Descent</u>

Start with random value of θ, step by step modify θ in order to decrease the cost function

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial_{\theta_j}} E(\theta)$$

$$\frac{\partial}{\partial_{\theta_j}} E(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial_{\theta_j}} (\theta^T x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$$

Multiple Input Representation

<u>Gradient Descent</u>

Start with random value of θ, step by step modify θ in order to decrease the cost function

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial_{\theta_j}} E(\theta)$$

$$\frac{\partial}{\partial_{\theta_j}} E(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial_{\theta_j}} (\theta^T x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$$

Normal Equations

Linear Regression:

Minimize the value of the cost function:

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

Normal Equations:

Solve the following equation to find out the optimized value of θ

$$\frac{\partial}{\partial_\theta} E(\theta) = 0$$

Normal Equations

Linear Regression:

Minimize the value of the cost function:

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

Normal Equations:

Solve the following equation to find out the optimized value of θ

$$\frac{\partial}{\partial_\theta}E(\theta) = 0 \qquad \forall\ j \in (0,1,\ldots,n), \frac{\partial}{\partial_{\theta_j}}E(\theta) = 0$$

Normal Equations

<u>Solution:</u>

Given a training set of m training example, each contain n inputs, we have the matrix X (m,n+1) of inputs and vector of output Y

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_n^{(2)} \\ & & \dots & \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \dots \\ (x^{(m)})^T \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

Solution of normal equations is:

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

- Using normal equation to find the value of coefficients theta for this training data

x	*y*
3	0
2	1
1	2

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

Homework

Size (m²)	N⁰ of floors	N⁰ of rooms	Price (billion VND)
30	3	6	*2.5*
43	4	8	*3.4*
25	2	3	*1.8*
51	4	9	*4.5*
40	3	5	*3.2*
20	1	2	*1.6*

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial_{\theta_j}} E(\theta)$$

$$\frac{\partial}{\partial_{\theta_j}} E(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial_{\theta_j}} (\theta^T x^{(i)} - y^{(i)})^2 \quad = \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$$

Polynomial Regression

Polynomial Regression

Output is an polynomial function of the input

For example

$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n$$

Assume

$$x_1 = x$$

$$x_2 = x^2$$

$$\ldots$$

$$x_n = x^n$$

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

Linear Regression

References

http://openclassroom.stanford.edu/MainFolder/VideoPage.php?course=MachineLearning&video=02.4-LinearRegressionI-GradientDescent&speed=100

Feature Rescale

Objective: Scale all features to the same scale, in order to have easier computation

Popular scale : [0,1],[-0.5,0.5]

x=x/max (x)

c= mean (x) x=x-c/max(x) =>