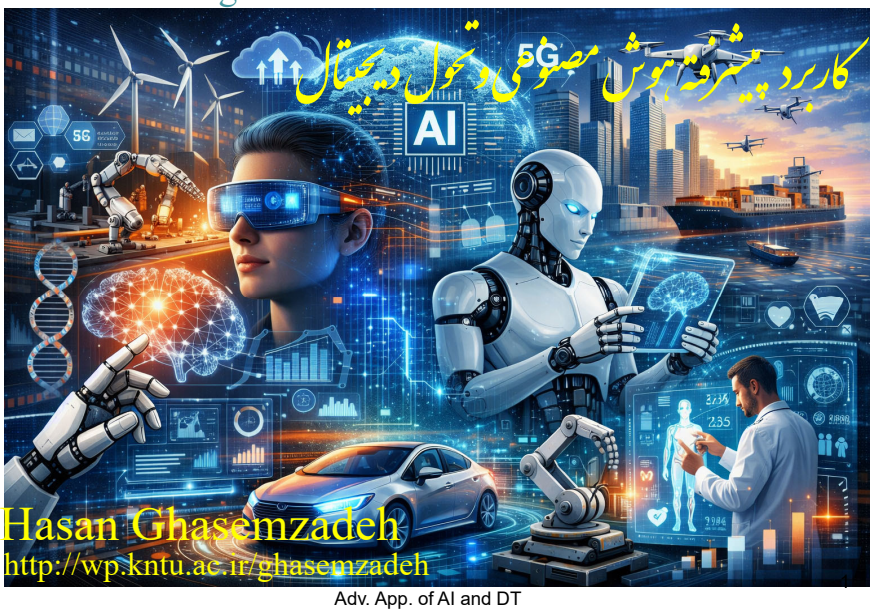
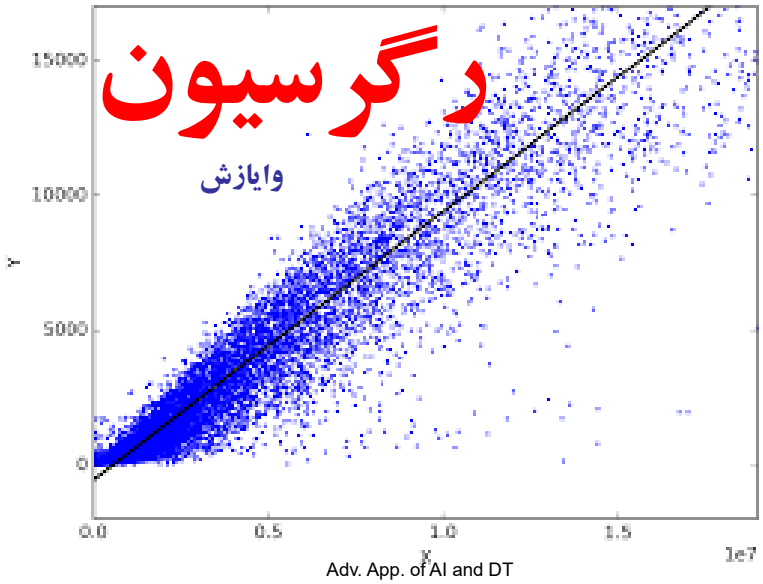
Advanced Application of Artificial Intelligence and Digital Transformation

K.N. Toosi University of Technology



Artificiel Intelligence and Soft computing

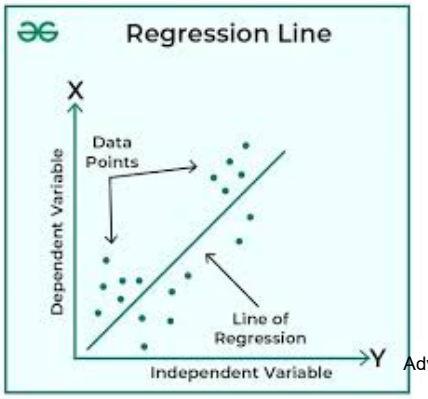
K.N.TOOSI University of Technology



یادگیری نظارت شده – رگرسیون

رگرسیون

رگرسیون یک نوع از یادگیری نظارت شده است که در آن الگوریتم یاد می‌گیرد بر اساس ویژگی‌های ورودی، مقادیری پیوسته را پیش‌بینی کند. مثال‌هایی از مسائل رگرسیون شامل پیش‌بینی قیمت سهام و قیمت مسکن می‌شوند.



الگوریتم‌های معروف:

رگرسیون خطی

رگرسیون چندجمله‌ای

رگرسیون ریج

رگرسیون درخت تصمیم

رگرسیون جنگل تصادفی

رگرسیون ماشین بردار پشتیبان

یادگیری نظارت شده– نمونه عمرانی رگرسیون

- **پیش‌بینی مقاومت بتن:**
با استفاده از ویژگی‌های مختلف مخلوط بتن، مانند درصد سیمان، سنگدانه، آب، و افزودنی‌ها

- **پیش‌بینی نشست زمین در پی‌های عمیق و تونل‌ها:**
با داشتن داده‌های مختلف از ویژگی‌های خاک، عمق تونل یا پی، و نوع ساختار

- **تخمین بار ترافیک بر پل‌ها:**
با تحلیل داده‌های مربوط به وزن وسایل نقلیه، تعداد وسایل نقلیه عبوری، و وضعیت آب و هوا

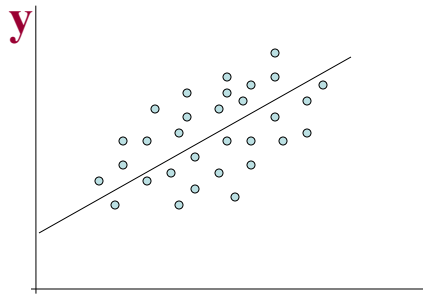
- **پیش‌بینی میزان آب جاری در رودخانه‌ها:**
با داشتن داده‌های هواشناسی (بارش، دما) و ویژگی‌های هیدرولوژیکی حوزه آبخیز

- **پیش‌بینی عمر مفید سازه‌ها:**
با تحلیل پارامترهای مختلف مانند نوع مصالح، شرایط محیطی و بارهای وارده

رگرسیون خطی

- رگرسیون خطی ساده ترین و پرکاربردترین نوع رگرسیون است. نمودار آن نشان دهنده یک خط راست است.
- برای شروع باید حدسی درباره وجود یک رابطه خطی وجود داشته باشد. نمودار Scatter Plot ایده ای اولیه درباره این موضوع می دهد.

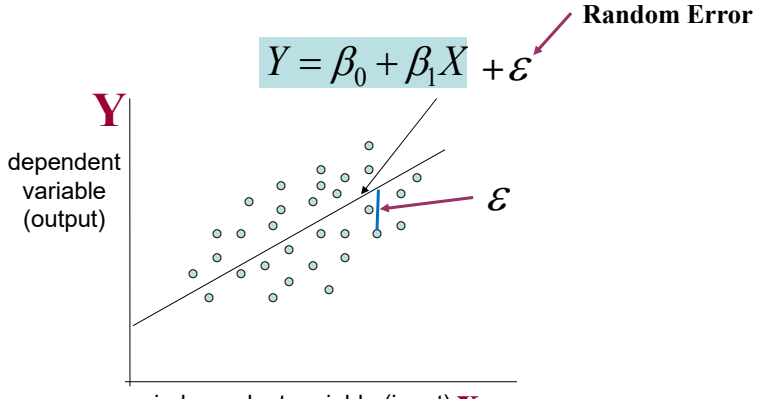
با دیدن این نمودار این ایده به ذهن می رسد که با افزایش x متغیر y هم افزایش می یابد و بلعکس.



Adv. App. of AI and DT 5

رگرسیون خطی

- روابط احتمالی همیشه دارای خطا می باشد که در اینجا آنرا با ε نمایش می دهیم.



Adv. App. of AI and DT 6

رگرسیون خطی

Objective function: we use sum squared error (SSE)

$\Sigma\ (predicted_i - actual_i)^2 = \Sigma\ (residue_i)^2$

$$Y = \beta_0 + \beta_1 X$$

- Minimize the objective function: we can take the partial derivates of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$\beta_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2} \qquad \beta_0 = \frac{\sum y - \beta_1 \sum x}{n}$$

Adv. App. of AI and DT 7

رگرسیون خطی

- مثال: فرض کنید هدف تخمین میزان پرداختی است که شخصی با درآمد معین برای مواد غذایی می پردازد. برای پاسخ به این مساله باید از چندین خانوار نمونه گیری کنیم(درآمد به میلیون).

X = [21, 35, 7, 33, 5, 14, 25] $\sum x = 140$

Y = [15, 21, 2, 13, 3, 8, 9] $\sum y = 71$

n =7

حل به روش رگرسیون

XY=[315, 735, 14, 429, 15, 112, 225] $\sum xy = 1845$

X^2= [441,1225,49,1089,25, 196, 625] $\sum x^2 = 3650$

$$\beta_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2} = 0.5 \qquad \beta_0 = \frac{\sum y - \beta_1 \sum x}{n} = 0.143$$

Adv. App. of AI and DT 8

رگرسیون خطی

محاسبه ضرایب خط رگرسیون

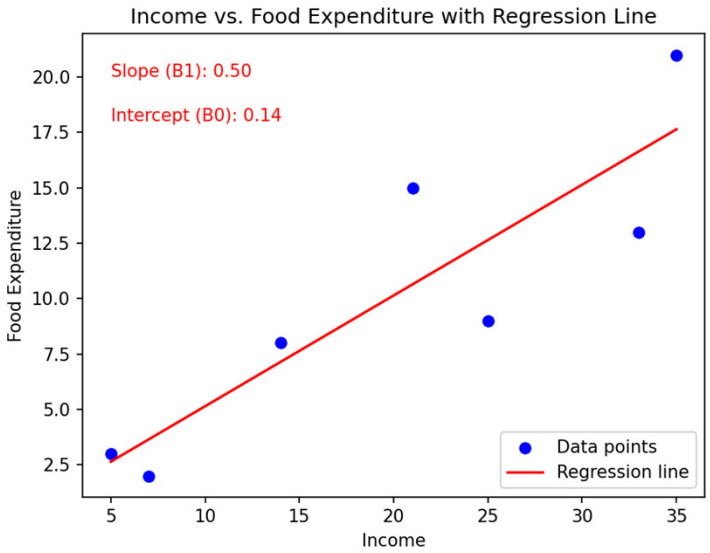
```
import matplotlib.pyplot as plt

x = [21, 35, 7, 33, 5, 14, 25]  # Income
y = [15, 21, 2, 13, 3, 8, 9]    # Food Expenditure

N = len(x)    # Number of data points

# Calculating products of x and y, and squares of x
xy = [x[i] * y[i] for i in range(N)]
x_squared = [x[i] ** 2 for i in range(N)]

# Calculating the sums
sum_x = sum(x)
sum_y = sum(y)
sum_xy = sum(xy)
sum_x_squared = sum(x_squared)

# Calculating the slope (B1) and intercept (B0)
B1 = (N * sum_xy - sum_x * sum_y) / (N * sum_x_squared - (sum_x) ** 2)
B0 = (sum_y - B1 * sum_x) / N
```


رگرسیون خطی

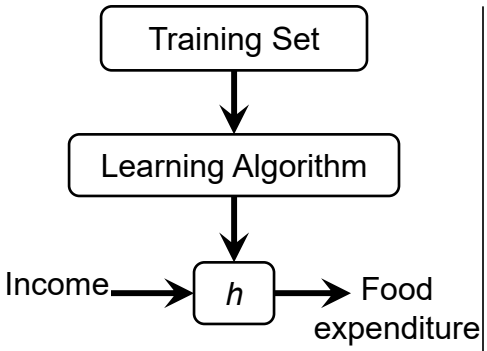
رسم خط رگرسیون

```
# Plotting the scatter plot
plt.scatter(x, y, color="blue", label="Data points")

# Calculating the regression line values
x_values = range(min(x), max(x) + 1)
y_values = [B0 + B1 * xi for xi in x_values]

# Plotting the regression line
plt.plot(x_values, y_values, color="red", label="Regression line")

# Adding labels and title
plt.xlabel("Income")
plt.ylabel("Food Expenditure")
plt.title("Income vs. Food Expenditure with Regression Line")
plt.legend()

# Annotating the slope and intercept
plt.text(min(x), max(y) - 1, f"Slope (B1): {B1:.2f}", color="red")
plt.text(min(x), max(y) - 3, f"Intercept (B0): {B0:.2f}",
color="red")

# Displaying the plot
plt.show()
```

رگرسیون خطی



رگرسیون خطی

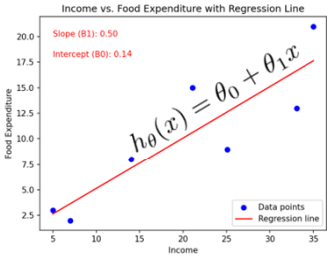


Hypothesis:

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

How do we represent *h* ?

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

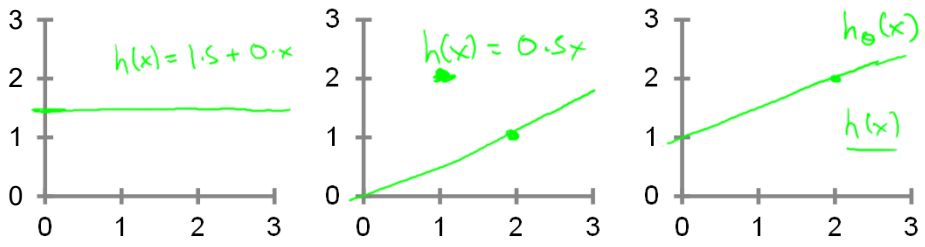


Linear regression with one variable.
Univariate linear regression.

How to choose parameters θ_i's ?

تعیین ضرایب رگرسیون خطی

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



$\theta_0 = 1.5$, $\theta_1 = 0$ | $\theta_0 = 0$, $\theta_1 = 0.5$ | $\theta_0 = 1$, $\theta_1 = 0.5$

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

تعیین ضرایب رگرسیون خطی

Hypothesis:

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

Parameters:

θ_0, θ_1

Cost Function:

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

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

Simplified

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

$\theta_0 = 0$

θ_1

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

$\theta, x^{(i)}$

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

با حداقل کردن تابع هزینه ضرایب خط رگرسیون تعیین می شوند

تعیین ضرایب رگرسیون با حداقل کردن تابع هزینه

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

Parameters: θ_0, θ_1

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

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

Adv. App. of AI and DT 15

حداقل کردن تابع هزینه

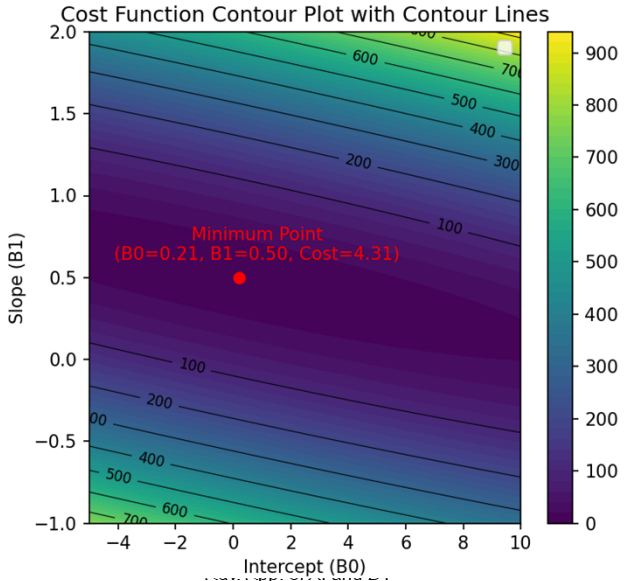
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Data from the table
x = np.array([21, 35, 7, 33, 5, 14, 25])  # Income
y = np.array([15, 21, 2, 13, 3, 8, 9])    # Food Expenditure

# Number of data points
M = len(x)  # Changed from N to M

# Defining the cost function
def cost_function(B0, B1):
    return (1 / (2 * M)) * np.sum((y - (B0 + B1 * x)) ** 2)

# Generating values for B0 and B1
B0_values = np.linspace(-5, 10, 1000)
B1_values = np.linspace(-1, 2, 2000)
B0_mesh, B1_mesh = np.meshgrid(B0_values, B1_values)

# Calculating cost values for each combination of B0 and B1
cost_values = np.zeros(B0_mesh.shape)
for i in range(B0_mesh.shape[0]):
    for j in range(B0_mesh.shape[1]):
        cost_values[i, j] = cost_function(B0_mesh[i, j], B1_mesh[i, j])

# Finding the minimum point
min_index = np.unravel_index(np.argmin(cost_values), cost_values.shape)
B0_min = B0_mesh[min_index]
B1_min = B1_mesh[min_index]
min_cost = cost_values[min_index]
```

Adv. App. of AI and DT 16

حداقل کردن تابع هزینه

```
# Plotting the 2D contour plot of the cost function
plt.figure(figsize=(12, 5))

# 2D Contour Plot with contour lines
plt.subplot(1, 2, 1)
cp = plt.contourf(B0_mesh, B1_mesh, cost_values, levels=50,
cmap='viridis')
plt.colorbar(cp)
contour_lines = plt.contour(B0_mesh, B1_mesh, cost_values, levels=10,
colors='black', linewidths=0.5)
plt.clabel(contour_lines, inline=True, fontsize=8)
plt.scatter(B0_min, B1_min, color="red")
plt.annotate(f"Minimum Point\n(B0={B0_min:.2f}, B1={B1_min:.2f},
Cost={min_cost:.2f})",
             (B0_min, B1_min),
             textcoords="offset points",
             xytext=(10, 10),  # Adjust these values to move the text
             ha='center',
             color="red")
plt.xlabel("Intercept (B0)")
plt.ylabel("Slope (B1)")
plt.title("Cost Function Contour Plot with Contour Lines")
plt.legend()
```

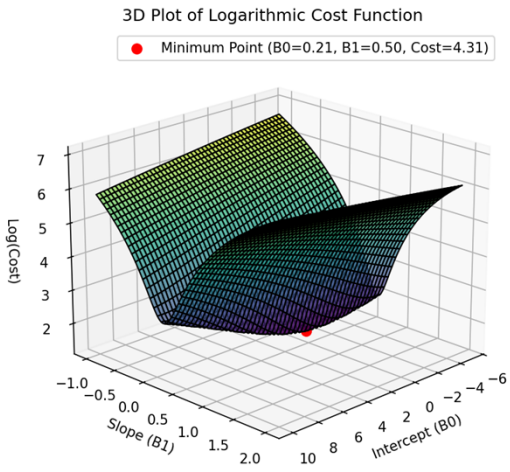

حداقل کردن تابع هزینه



حداقل کردن تابع هزینه

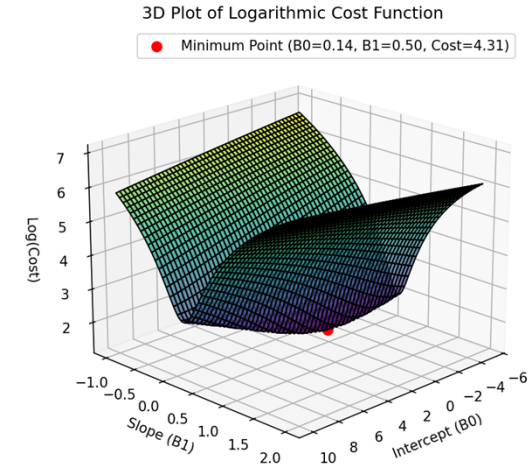
```
# Logarithmic scale for cost values (add a small value to avoid log(0))
log_cost_values = np.log(cost_values + 1e-10)  # Adding a small constant
to avoid log(0)

# 3D Surface Plot with logarithmic cost values
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(B0_mesh, B1_mesh, log_cost_values, cmap='viridis',
edgecolor='k', alpha=0.7)
ax.scatter(B0_min, B1_min, np.log(min_cost + 1e-10), color="red", s=50,
label=f"Minimum Point (B0={B0_min:.2f}, B1={B1_min:.2f},
Cost={min_cost:.2f})")

# Set the best view for the 3D plot
ax.view_init(elev=20, azim=45)  # Adjust these values to find the best
angle
# Labels and title
ax.set_xlabel("Intercept (B0)")
ax.set_ylabel("Slope (B1)")
ax.set_zlabel("Log(Cost)")
ax.set_title("3D Plot of Logarithmic Cost Function")
ax.legend()

# Print minimum values
print("Optimal Intercept (B0):", B0_min)
print("Optimal Slope (B1):", B1_min)
print("Minimum Cost:", min_cost)

plt.show()   # Show plots
```

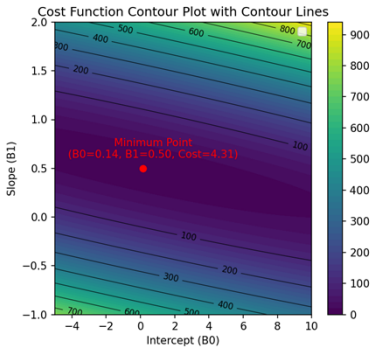

حداقل کردن تابع هزینه




```
# Generating values for B0 and B1
B0_values = np.linspace(-5, 10, 1000)
B1_values = np.linspace(-1, 2, 500)
```

حداقل کردن تابع هزینه






```
# Generating values for B0 and B1
B0_values = np.linspace(-5, 10, 1000)
B1_values = np.linspace(-1, 2, 2000)
```

ضرایب با توجه به دقت حل مساله کمی متفاوت خواهند بود

حداقل کردن تابع هزینه
نزول در راستای گرادیان

Gradient descent

نزول در راستای گرادیان

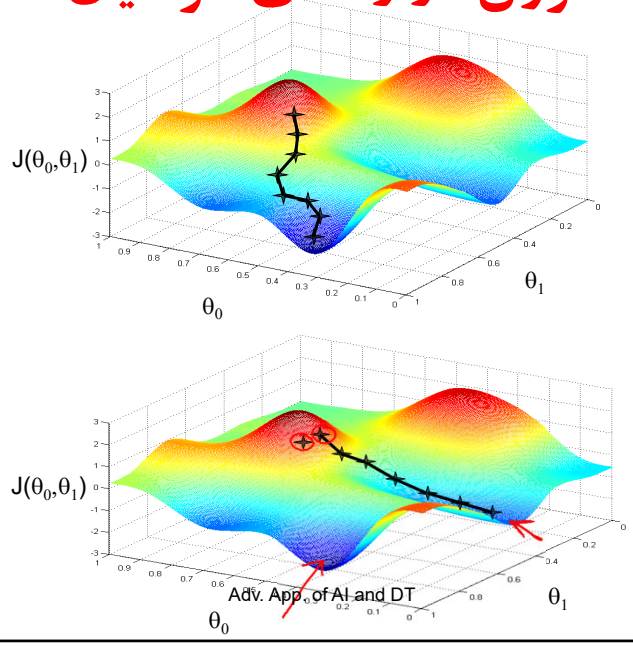
Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

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

Outline:

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

نزول در راستای گرادیان



نزول در راستای گرادیان

Gradient descent algorithm

repeat until convergence {

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

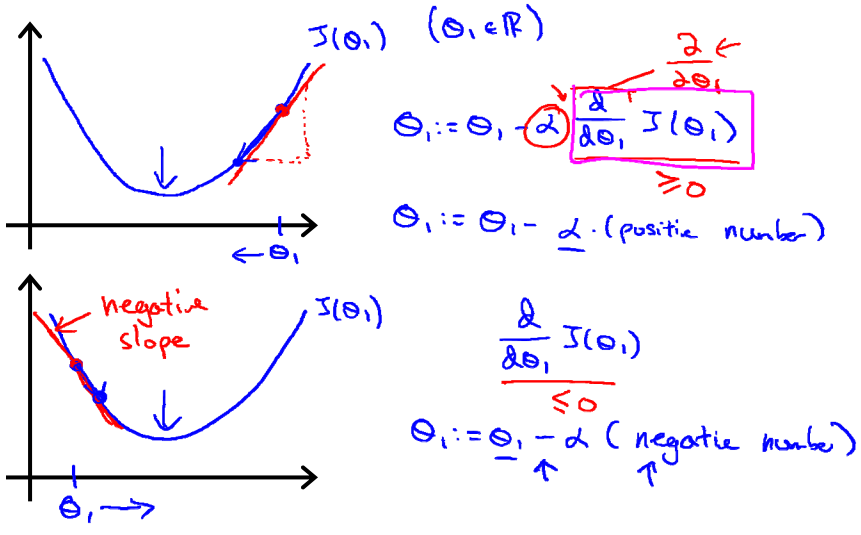
}

Learning rate

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

Adv. App. of AI and DT 25

نزول در راستای گرادیان

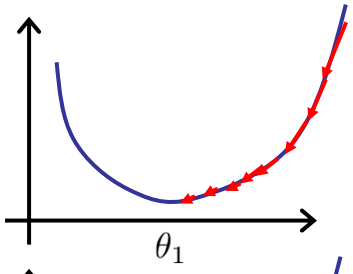


Adv. App. of AI and DT 26

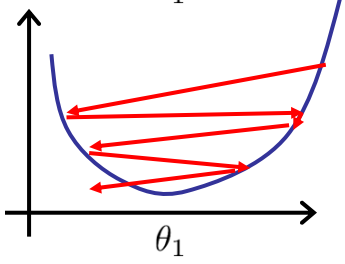
نزول در راستای گرادیان

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

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



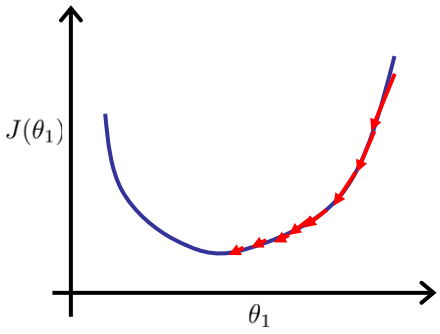
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



نزول در راستای گرادیان

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.

نزول در راستای گرادیان

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

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

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

نزول در راستای گرادیان

Gradient descent algorithm

repeat until convergence {

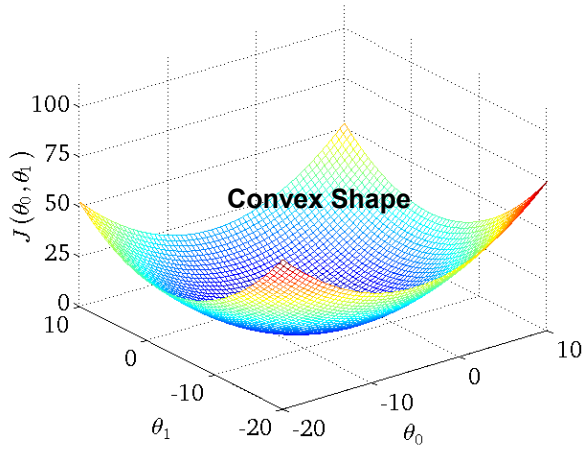
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

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

}

update θ_0 and θ_1 simultaneously

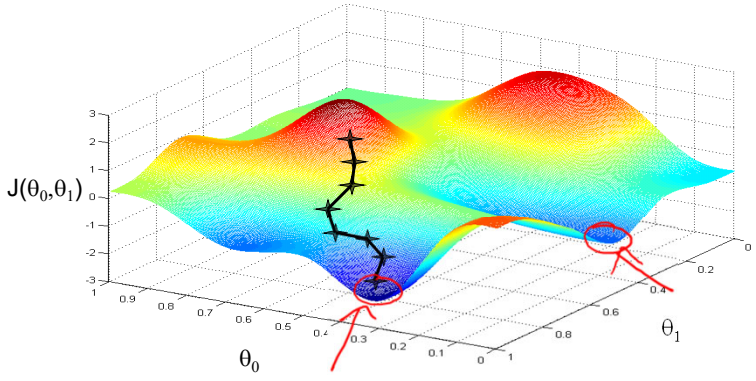
نزول در راستای گرادیان



“Batch” Gradient Descent

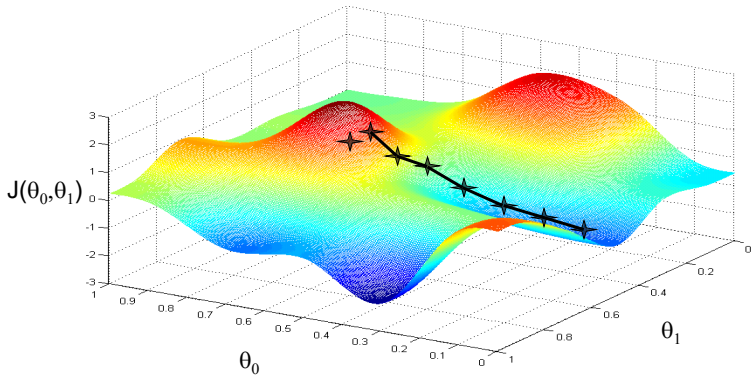
“Batch”: Each step of gradient descent uses all the training examples.

نزول در راستای گرادیان



Training and Validation Data

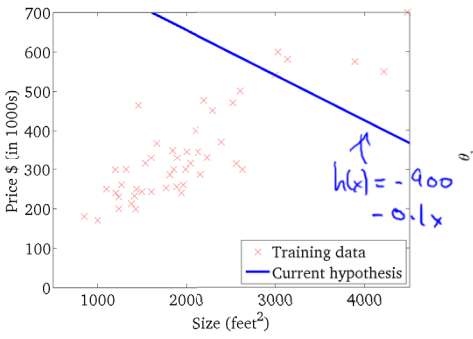
نزول در راستای گرادیان



Training and Validation Data

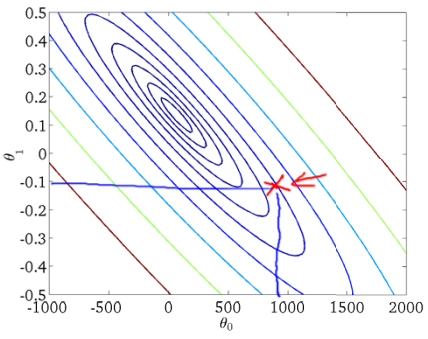
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



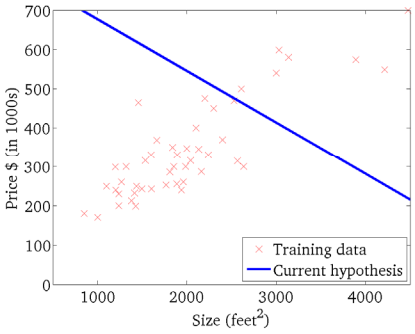
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



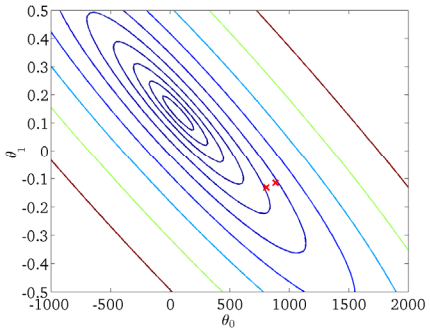
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

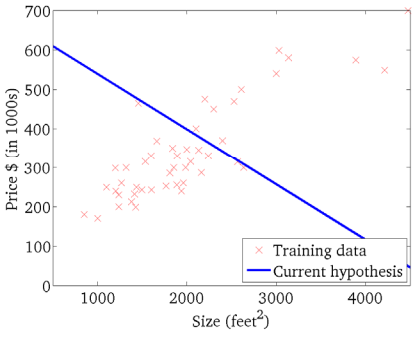
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 35

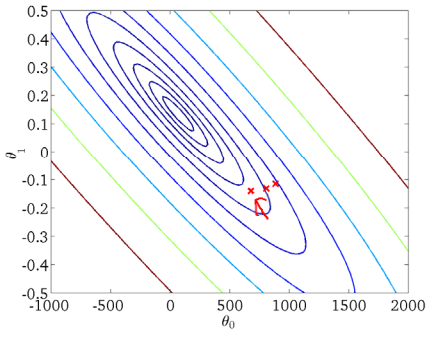
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

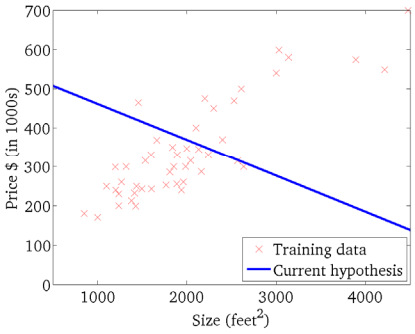
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 36

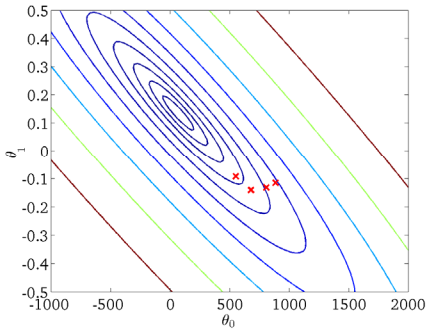
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

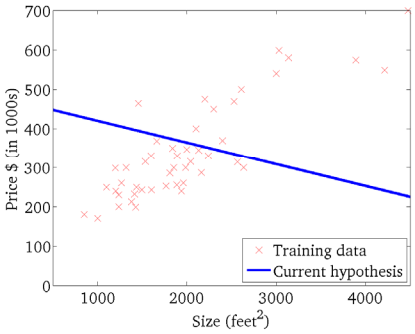
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 37

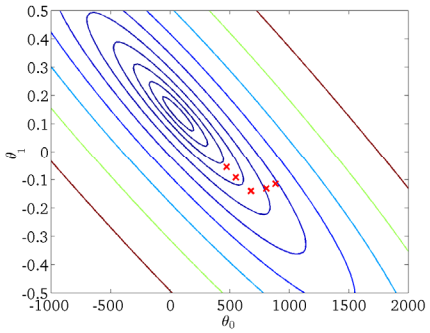
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

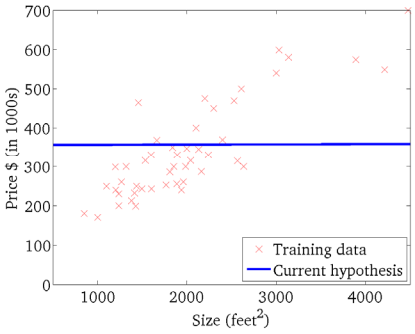
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 38

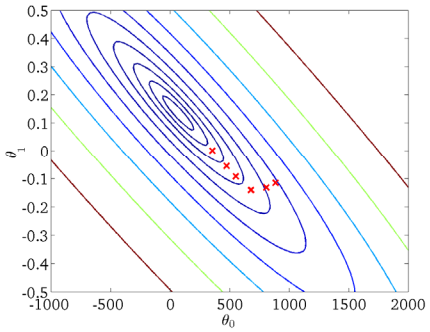
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

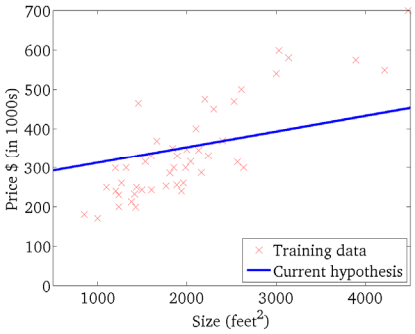
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 39

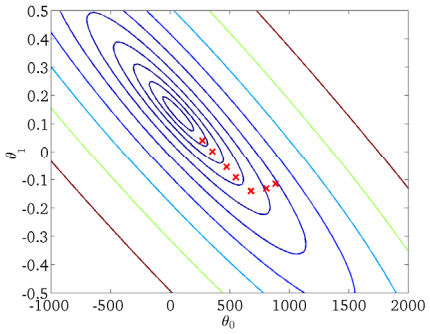
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

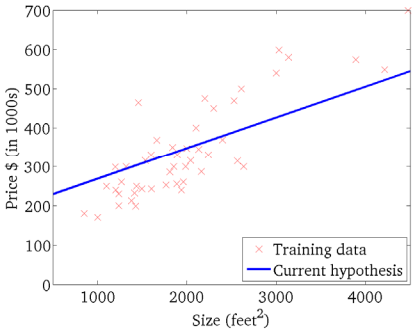
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 40

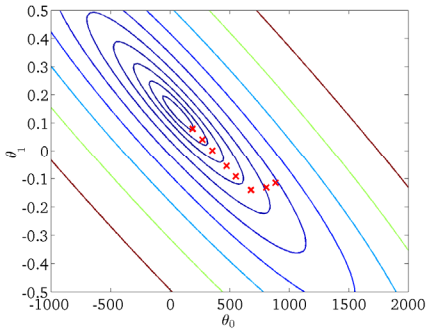
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

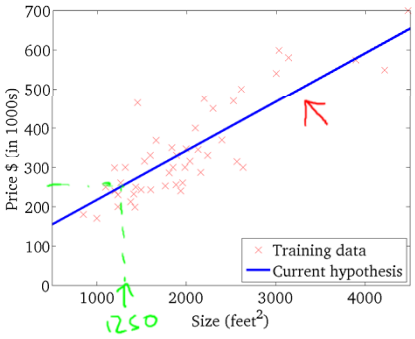
(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 41

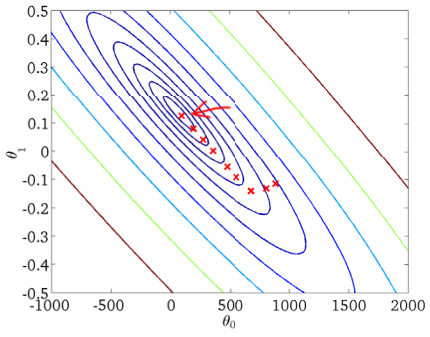
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Adv. App. of AI and DT 42

رگرسیون یک متغیره

آنالیز رگرسیون (واکاوی وایازشی)

One feature (Univariate).

Size (m^2) x	Price (Milliard Toman) y
210.4	46
141.6	23.2
153.4	31.5
85.2	17.8
…	…

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

رگرسیون خطی تک متغیره

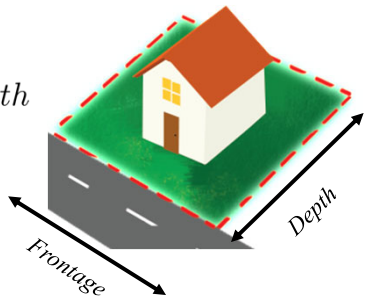
متغیر x

رگرسیون چند متغیره

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

$$Area = Frontage * Depth$$



گاهی اوقات یک متغیر ممکن است به چند متغیر
تبدیل شود و جواب بهتری از رگرسیون بدست آید

رگرسیون چند متغیره

Multiple features (variables). آنالیز رگرسیون (واکاوی وایازشی)

Size (m²)	Number of bedrooms	Number of floors	Age of home (years)	Price (Milliard Toman)
متغیرها ⟵ x_1	x_2	x_3	x_4	y
210.4	5	1	45	46
141.6	3	2	40	23.2
153.4	3	2	30	31.5
85.2	2	1	36	17.8
…	…	…	…	…

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

متغیر x_i یک بردار است

هر متغیر j نمونه یا مثال دارد که با آنها برنامه آموزشی خواهد دید تا y را حدس بزند

رگرسیون چند متغیره

Multiple features (variables).

متغیرها ⟵	Size (m²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (Milliard Toman) y
	210.4	5	1	45	46
	141.6	3	2	40	23.2
مثال سوم ⟵ X^3	153.4	3	2	30	31.5
	85.2	2	1	36	17.8
	300	4	1	38	54
	289	5	3	20	60
Notation:	…	…	…	…	…

n = number of features — $n = 4$ — تعداد متغیرها

$x^{(i)}$= input (features) of i^{th} training example. — $x^3 = [153.4\ 3\ 2\ 30]^T$

$x_j^{(i)}$= value of feature j in i^{th} training example. — $x^3_4 = 30$

نام متغیر در زیرنویس و شماره مثال در بالانویس می آید

$y = [46\ 23.2\ 31.5\ 17.8\ 54\ 60 \ldots]^T$

رگرسیون چند متغیره

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

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

For convenience of notation, define $x_0 = 1$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1} \quad X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1} \quad h_\theta = \theta^T X = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_\theta = \theta^T X = \sum_0^n \theta_n x_n$$

تعداد معادلات؟
تعداد مجهولات؟

training set: design matrix *X*

رگرسیون چند متغیره

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \qquad h_\theta = \theta^T X = \sum_0^n \theta_n x_n$$

$$J(\theta) = \frac{1}{2} \sum_1^j (h_\theta(x^{(j)}) - y^{(j)})^2$$

تابع هزینه بایستی حداقل شود

مشتق گیری (در این درس مورد نیاز نیست)

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T (X\theta - \vec{y}) \\ &= \frac{1}{2}\nabla_\theta \left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T \vec{y}\right) \\ &= \frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T \vec{y}\right) \\ &= \frac{1}{2}\nabla_\theta \left(\mathrm{tr}\,\theta^T X^T X\theta - 2\mathrm{tr}\,\vec{y}^T X\theta\right) \\ &= \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T \vec{y}\right) \\ &= X^T X\theta - X^T \vec{y} \end{aligned}$$

$$Ax = b \rightarrow A^T Ax = A^T b \Rightarrow x = \left(A^T A\right)^{-1} A^T b$$

$$X^T X\theta = X^T y$$

مشتق برابر صفر

معادله نرمال

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

تعیین ضرایب رگرسیون

رگرسیون چند متغیره

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

Cost function:

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

Gradient descent:

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

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

رگرسیون چند متغیره

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 $j = 0, \ldots, n$)

}

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

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

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

تمرین برنامه نویسی

تمرین پنجم : یک برنامه به زبان پایتون بنویسید رگرسیون خطی برای داده های فایل depth-temperature رسم کند.

Depth (m)	Temperature (c)
0.0	4.3
2.7	4.3
5.4	4.3
8.2	4.4
10.9	4.4
13.7	4.4
16.4	4.5
19.2	4.5
21.9	4.6
24.7	4.6
27.4	4.7
30.2	4.7
32.9	4.8
35.7	4.9

۱- به روش حداقل کردن تابع هزینه – رگرسیون خطی
۲- به روش نزول در راستای گرادیان
۳- زمان اجرای برنامه در دوحالت را مقایسه کنید
۴- نتایج را رسم نمایید