Linear Regression

رگرسیون خطی (Linear Regression) :

یکی دیگر از الگوریتم ها و مدل های یادگیری ماشین ، رگرسیون خطی است. در این مدل اگر داده ها را به صورت نقاط مختلف در صفحه مختصات رسم کنیم که x در واقع همان ویژگی (Feature) و y همان Label است. سپس خطی بر داده ها رسم می شود. حال داده جدید که مقدار ویژگی آن را می دانیم را به مدل به عنوان ورودی می دهیم و با توجه به خط رسم شده خروجی یا همان Label توسط مدل پیش بینی می شود.

معادله خط رسم شده به شکل زیر است :

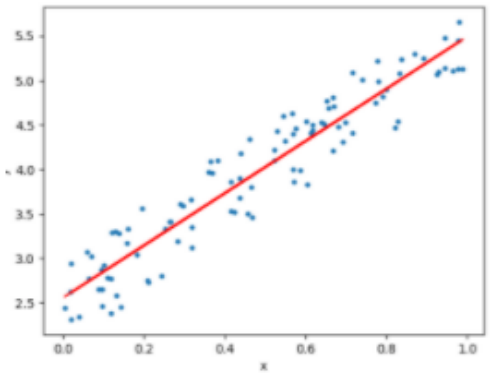
$$h_\theta(x) = \theta_0 + \theta_1 x$$

در این معادله به h اصطلاحا تابع فرضیه (Hypothesis Function) می گویند که مقدار آن برابر با همان y پیش بینی شده یا Label پیش بینی شده است.

X همان ویژگی است.

θ_1, θ_0 ضرایب معادله هستند که اصطلاحا به آن ها پارامترهای مدل می گویند. مقادیر آن ها با استفاده از داده های آموزش بدست می آید. پس طی فرآیند آموزش مقادیر پارامترهای مدل بدست می آید.

θ_0 را بایاس هم می نامند که در ادامه بیشتر با آن آشنا خواهیم شد.



ممکن است داده ها بیشتر از یک ویژگی داشته باشند (رگرسیون خطی چند متغیره) که در این صورت می توانیم معادله خط را به صورت کلی به شکل زیر بنویسیم :

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

که n تعداد ویژگی ها است.

فرم برداری معادله خط رسم شده نیز به شکل زیر است :

$$\hat{y} = h_{\theta}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

تا به اینجا با مدل خطی رگرسیون آشنا شدیم ، حال باید این مدل را آموزش دهیم. آموزش مدل به معنی تنظیم پارامترهای آن مدل است تا مدل متناسب با داده های آموزش باشد.

- **تابع هزینه (Cost Function) :**

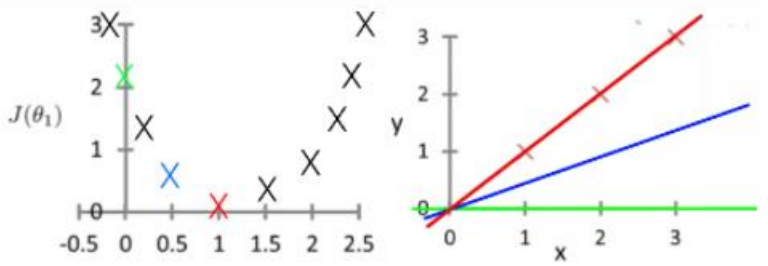
به واسطه تابع هزینه می توانیم خطای بین داده های آموزشی و تابع فرضیه را بدست بیاوریم. هدف ما این است که مقادیر پارامترها را طوری بدست بیاوریم که مقدار خطا یا هزینه حداقل بشود تا مدل بر داده های آموزشی بهترین تطابق را داشته باشد. بدین معنی است که خروجی مدل (مقداری که مدل پیش بینی می کند) به Label هایی که به عنوان دیتا داریم حد الامکان نزدیک باشد.

با توجه به نکات بالا تابع هزینه به شکل زیر تعریف می شود :

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

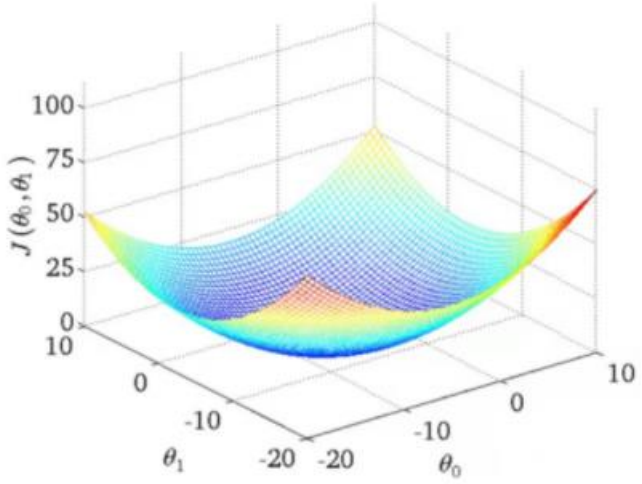
بدلیل آن که هدف حداقل کردن هزینه یا مقدار خطا بود پس باید تابع هزینه را مینیمم کنیم تا پارامترهای مناسب برای مدل بدست بیاید.

اگر θ_0 را صفر قرار دهیم و به θ_1 مقادیر مختلف بدهیم ، به ازای هر مقدار تابع فرضیه متفاوتی بدست می آید. حال اگر تابع هزینه را برای تمامی مقادیر بدست بیاوریم ، مشاهده خواهیم کرد در θ_1 برابر با ۱ ، مقدار تابع هزینه مینیمم است.

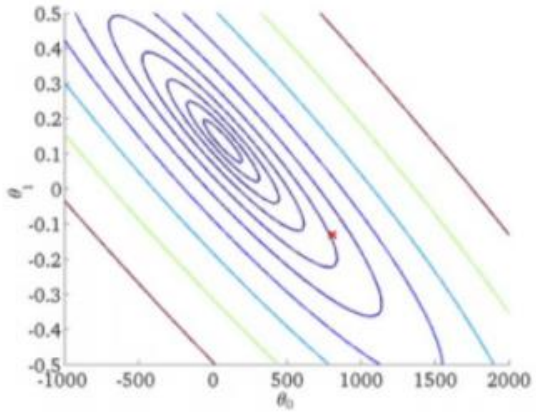


با توجه به نمودار بالا ، نمودار تابع هزینه برحسب θ_1 به صورت سهمی است.

حال اگر θ_0 را برابر با صفر قرار ندهیم و مقدار داشته باشد ، تابع هزینه برحسب دو پارامتر خواهد بود :



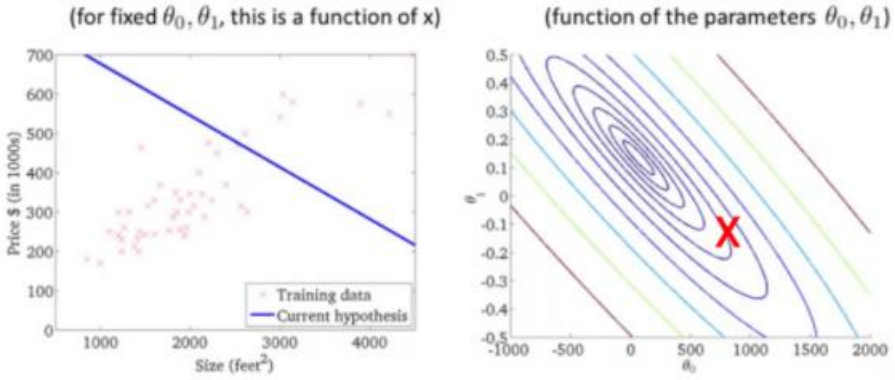
برای نمایش بهتر از نمودار کانتوری به جای شکل سه بعدی استفاده می کنیم :

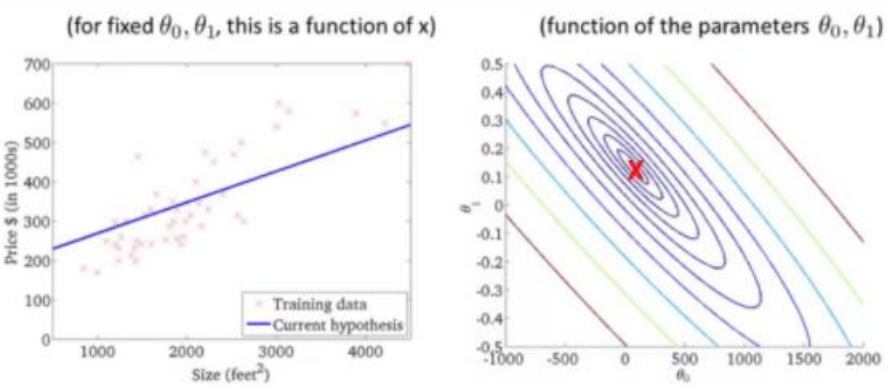


هر بیضی نشان دهنده مجموعه ای از نقاط است که مقدار تابع هزینه آن برحسب θ_0 و θ_1 های مختلف ، برابر است.

کوچکترین بیضی ، مینیمم تابع هزینه را نشان می دهد و هر چه از آن دورتر می شویم تابع فرضیه تناسب کمتری با دیتاهای آموزشی دارد.

برای مثال :





شکل دوم به نسبت شکل اول تناسب بیشتر و بهتری دارد ولی اگر دقت کنید متوجه می شوید کاملا متناسب نیست. برای تناسب کامل و رسیدن به مینیمم تابع هزینه نیاز به الگوریتم خاصی است. الگوریتمی مانند گرادیان کاهشی (Gradient Descent).

❖ **پروژه :**

تا به اینجا با الگوریتم رگرسیون خطی آشنا شدیم ، حال می خواهیم یک پروژه پیاده سازی کنیم.

می خواهیم با استفاده از الگوریتم رگرسیون خطی ارتباط بین فروش و تبلیغات محصولات را پیدا کنیم.

مراحل پیاده سازی پروژه به صورت زیر است :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
url = "C:/Users/Asus/Documents/AI/Linear Regression/Sales & Advertising/Dataset.csv"
df = pd.read_csv(url, header=None)
```

Exploratory Data Analysis

```
print(df.shape)
```

```
...
```

```
print(df.head())
```

```
...
```

```
df.columns = ['Sales', 'Advertising']
```

```
print(df.head())
```

```
...
```

```
print(df.info())
```

```
...
```

```
print(df.describe())
```

```
X = df['Sales'].values
y = df['Advertising'].values
```

Visual Exploratory Data Analysis

```
plt.scatter(X, y, color = 'blue', label='Scatter Plot')
plt.title('Relationship between Sales and Advertising')
plt.xlabel('Sales')
plt.ylabel('Advertising')
plt.legend(loc=4)
plt.show()
```

```
print(X.shape)
print(y.shape)
```

```
X = X.reshape(-1,1)
y = y.reshape(-1,1)
```

```
print(X.shape)
print(y.shape)
```

Train Test Split

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

Model

```
lm = LinearRegression()

lm.fit(X_train,y_train)

y_pred = lm.predict(X_test)
```

```
a = lm.coef_
b = lm.intercept_,
print("Estimated model slope, a:" , a)
print("Estimated model intercept, b:" , b)
```

Making Predictions

```
lm.predict(X)[0:5]
```

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("RMSE value: {:.4f}".format(rmse))
```

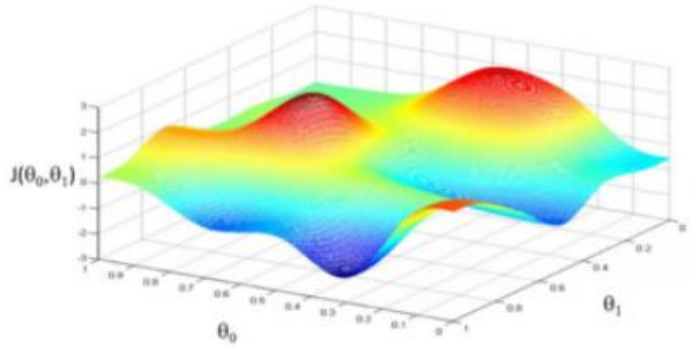
```
plt.scatter(X, y, color = 'blue', label='Scatter Plot')
plt.plot(X_test, y_pred, color = 'black', linewidth=3, label = 'Regression Line')
plt.title('Relationship between Sales and Advertising')
plt.xlabel('Sales')
plt.ylabel('Advertising')
plt.legend(loc=4)
plt.show()
```

❖ **گرادیان کاهشی (Gradient Descent) :**

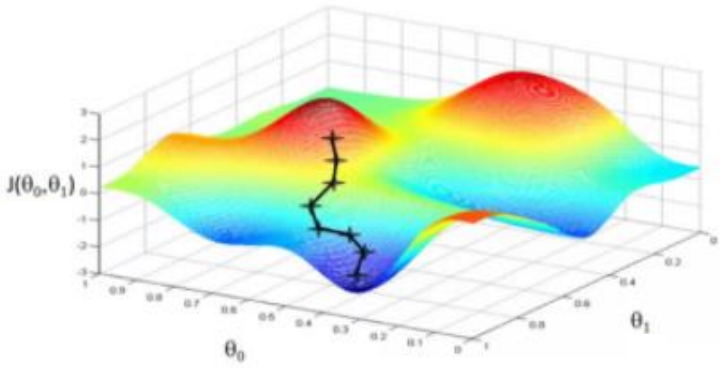
با استفاده از گرادیان کاهشی ، مقادیر θ_0 و θ_1 را طوری بدست میاوریم که تابع هزینه مینیمم شود تا مدل با دیتاهای آموزشی به خوبی تناسب داشته باشد.

روند کلی که در الگوریتم گرادیان کاهشی طی می شود بدین صورت است که ابتدا مقدار پارامترهای θ_0 و θ_1 را حدس می زنیم. برای مثال هر دو را صفر قرار می دهیم. سپس θ_0 و θ_1 را به صورت جزئی تغییر می دهیم تا تابع هزینه کاهش پیدا کند. این روند را تا زمانی تکرار می کنیم که به مینیمم محلی یا مینیمم کلی تابع هزینه برسیم.

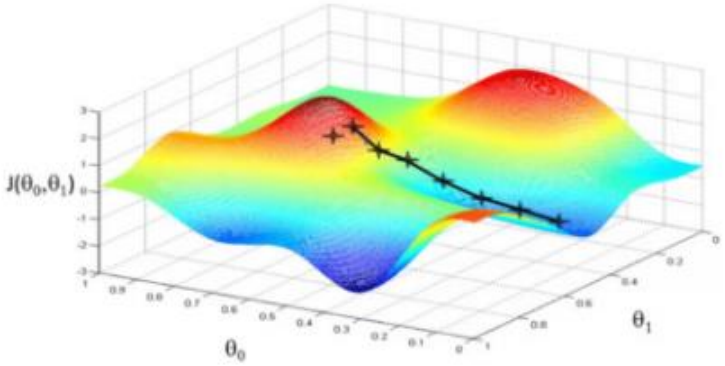
تصور کنید بر روی تپه های قرمز رنگ ایستاده اید و می خواهید پایین بروید.



مسیر زیر را طی می کنید.



اما ممکن است از کمی آن طرف تر بروید و مسیر دوم را طی کنید.



مفهوم کلی گرادیان کاهشی بدین شکل بود و تعریف ریاضیاتی گرادیان کاهشی به شکل زیر است.

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

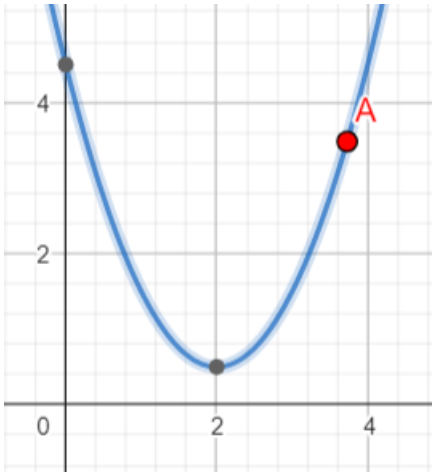
رابطه بالا تا زمانی تکرار می شود که به مینیمم تابع هزینه برسیم.

نکته ! در رابطه بالا α نرخ یادگیری (Learning Rate) است.

اگر رابطه ریاضی را با مثال تپه های قرمز مقایسه کنیم ، α میزان سرعت حرکت ما در پایین آمدن از تپه ها است (یا میزان آن که قدم هایمان را چقدر بلند و یا سریع برداریم). همچنین عبارت مشتق جهت حرکت ما خواهد بود.

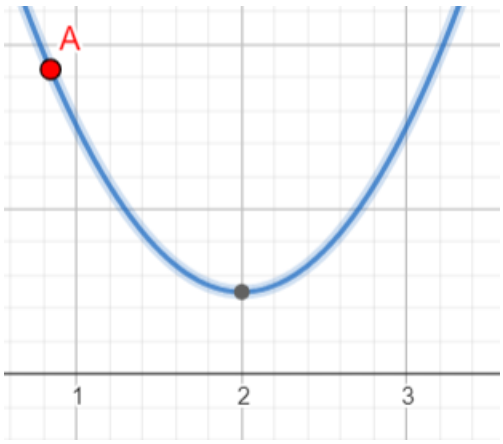
در ادامه قرار است که نرخ یادگیری و عبارت مشتق را بهتر و عمیق تر درک کنیم.

تصور کنید تابع هزینه را فقط با θ_1 داریم و برای شروع ، نقطه زیر را به عنوان مقدار آن حدس می زنیم :



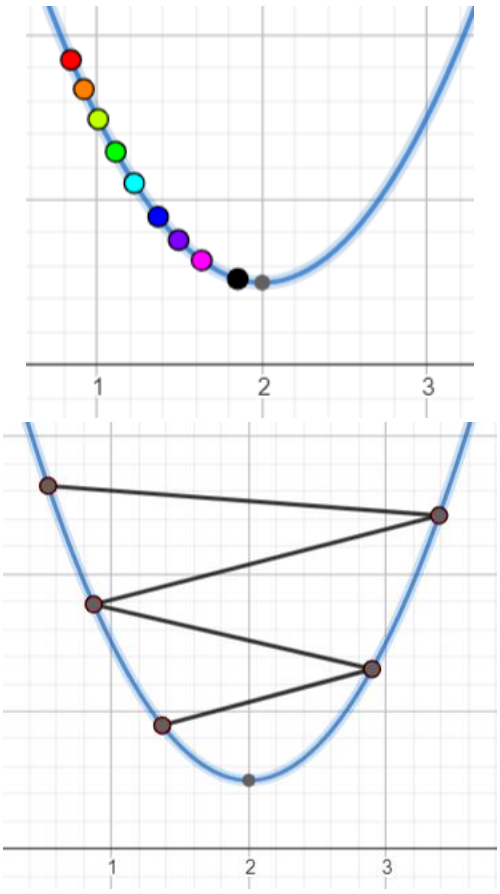
عبارت مشتق تانژانت این نقطه را محاسبه می کند سپس شیب خط را بدست میاوریم. شیب مثبت خواهد بود در نتیجه حاصل عبارت مشتق مثبت خواهد بود. از طرفی می دانیم نرخ یادگیری همیشه مثبت است و با توجه به رابطه نتیجه می گیریم از مقدار θ_1 کم خواهد شد ، این بدین معنی است که به سمت مینیمم نزدیک می شود.

حال اگر از نقطه زیر شروع کنیم :



شیب منفی خواهد بود که این بدین معنی است عبارت مشتق منفی خواهد شد. با توجه به رابطه نتیجه می گیریم که به مقدار θ_1 اضافه خواهد شد ، این بدین معنی است که به سمت مینیمم نزدیک می شود.

در شکل های زیر تفاوت بین دو مقدار نرخ یادگیری را مشاهده می کنیم :



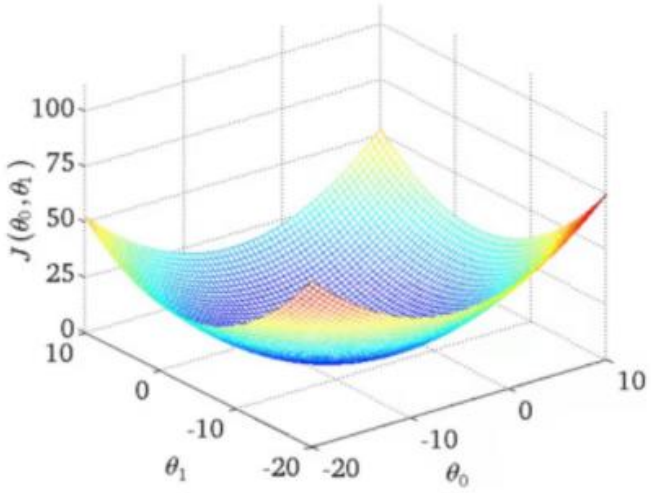
در شکل اول نرخ یادگیری مقداری کوچک دارد که باعث می شود خیلی کند تر به مینیمم برسیم و نیاز به قدم های بیشتری خواهیم داشت.
در شکل دوم نرخ یادگیری مقداری بزرگ دارد که باعث می شود اصلا به مینیمم نرسیم و روند گرادیان کاهشی واگرا می شود.

حال با استفاده از گرادیان کاهشی و تابع هزینه ، تابع فرضیه مناسب را پیدا می کنیم.
برای این کار باید تابع هزینه را در الگوریتم گرادیان کاهشی قرار دهیم :

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

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

نکته ! ممکن است بر اساس مثال تپه ها و پایین آمدن از مسیرهای مختلف به مینیمم های محلی مختلف برسیم و نتایج متفاوت باشد اما در رگرسیون خطی تابع هزینه همیشه محدب گونه است و از هر مسیری که به پایین حرکت کنیم در نهایت به یک مینیمم که مینیمم کلی است خواهیم رسید به همین دلیل استفاده از گرادیان کاهشی همیشه ما را به نقطه بهینه و نتیجه خوب و مناسب خواهد رساند.



نکته ! رابطه ریاضیاتی گرادیان کاهشی برای رگرسیون خطی چند متغیره به شکل زیر است :

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 $j = 0, \ldots, n$)

$\}$

انواع گرادیان کاهشی به شکل زیر است :

۱- Batch Gradient Descent : در این حالت تمامی داده ها به الگوریتم گرادیان کاهشی داده می شود تا بهترین پارامترها را پیدا کند.

۲- Mini Batch Gradient Descent : در این حالت تعدادی از داده ها به الگوریتم گرادیان کاهشی داده می شود تا بهترین پارامترها را پیدا کند.

۳- Stochastic Gradient Descent : در این حالت تنها یکی از داده ها به الگوریتم گردایان کاهشی داده می شود تا بهترین پارامترها را پیدا کند.

نکته ! اصطلاحی وجود دارد به نام Epoch ، بدین معنی است که 1 Epoch ، یعنی یکبار دادن تمامی داده ها به الگوریتم. برای مثال اگر از Batch Gradient Descent استفاده کنیم ، برای حل مسئله نیاز به 1 Epoch است.

رگرسیون چند جمله ای (Polynomial Regression) :

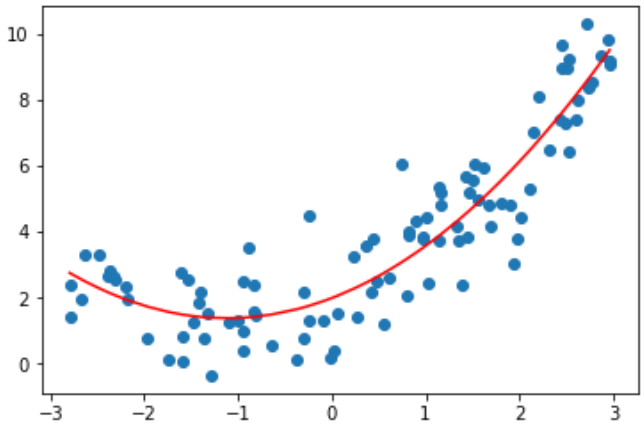
اگر تابع فرضیه خطی نباشد ، زیرا در صورت خطی بودن تناسب خوبی با داده ها نخواهد داشت ، در این شرایط می توان برای تابع فرضیه از توابع چند جمله ای استفاده کرد تا تناسب خوبی با داده ها ایجاد کرد.

برای مثال اگر فرض کنیم تابع فرضیه به صورت قبل باشد ، برای رگرسیون چند جمله ای می توانیم بر اساس x_1 تابع فرضیه درجه دوم به شکل زیر ایجاد کنیم :

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

و نیز می توانیم تابع فرضیه درجه سوم به شکل زیر داشته باشیم :

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



ممکن است به شکل زیر نیز داشته باشیم :

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3 \end{aligned}$$

❖ **معادله نرمال :**

یکی از روش ها برای پیدا کردن مینیمم تابع هزینه ، گرادیان کاهشی بود. روش معادله نرمال ، روشی است که بدون داشتن حلقه تکرار ، مینیمم تابع هزینه را پیدا می کند.

فرض کنید تابع هزینه به صورت زیر باشد :

$$J(\theta) = a\theta^2 + b\theta + c$$

برای پیدا کردن مینیمم تابع هزینه باید از تابع هزینه مشتق گرفته شود و حاصل را برابر صفر قرار داد. سپس مینیمم تابع هزینه بدست میاید.

$$\frac{\partial}{\partial x} J(\theta) \overset{\text{set}}{=} 0$$

توجه کنید که θ یک عدد حقیقی نیست بلکه یک بردار است. یعنی داریم :

$$\theta \in \mathbb{R}^{n+1} \qquad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0 \quad \text{(for every } j\text{)}$$

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

برای محاسبه θ هایی که تابع هزینه در آن ها مینیمم است ، به روش نرمال داریم :

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

نکته ! اگر از روش معادله نرمال استفاده شود ، نیاز به مقیاس بندی ویژگی ها نیست.

مقایسه گرادیان کاهشی و معادله نرمال :

Gradient Descent	**Normal Equation**
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large