**به نام خدا**

**پیاده‌سازی Gradient Descent (گرادیان کاهشی) برای رگرسیون خطی با توضیح کامل**

بیایید مرحله به مرحله گرادیان کاهشی را برای مدل رگرسیون خطی پیاده‌سازی کنیم، دقیقاً مشابه توضیح قبلی اما اینبار با روش بهینه‌سازی تکراری:

1. آماده‌سازی داده‌ها (همان مثال قبلی)

import numpy as np

*# داده‌های مثال*

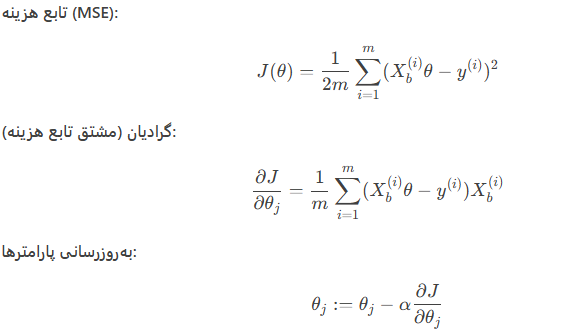
X = np.array([[2], [3], [4]]) *# ویژگی ورودی*

y = np.array([4, 6, 8]) *# خروجی واقعی*

*# اضافه کردن ستون 1 برای ضریب اریب*

X\_b = np.c\_[np.ones((3, 1)), X] *# X\_b = [[1, 2], [1, 3], [1, 4]]*

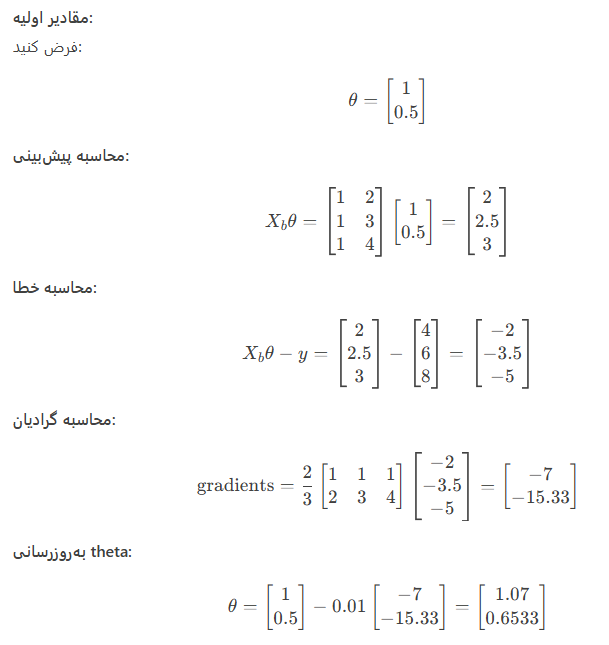
### فرمول‌های کلیدی



### پیاده‌سازی Gradient Descent

1. *# پارامترهای اولیه*
2. theta = np.random.randn(2, 1) *# مقادیر تصادفی [θ₀, θ₁]*
3. learning\_rate = 0.01 *# نرخ یادگیری*
4. n\_iterations = 1000 *# تعداد تکرار*
5. for iteration in range(n\_iterations):
6. *# محاسبه گرادیان*
7. gradients = 2/len(X\_b) \* X\_b.T.dot(X\_b.dot(theta) - y)
9. *# به‌روزرسانی theta*
10. theta = theta - learning\_rate \* gradients

### 4. تجزیه‌وتحلیل قدم‌به‌قدم (برای تکرار اول)



### 5. نتیجه نهایی

### بعد از 1000 تکرار:

print(theta) *# ≈ [0.], [2.] (همان نتیجه معادله نرمال)*

### 6. مقایسه با معادله نرمال

| **ویژگی** | **Gradient Descent** | **معادله نرمال** |
| --- | --- | --- |
| سرعت | آهسته (نیاز به تکرار) | سریع (محاسبه یک‌باره) |
| حافظه | مناسب داده‌های بزرگ | نیاز به معکوس ماتریس |
| دقت | وابسته به نرخ یادگیری | دقیق |
| پیچیدگی پیاده‌سازی | ساده | نیاز به محاسبات ماتریسی |

### 7. نکات کاربردی

* **انتخاب نرخ یادگیری (α):**
  + α خیلی بزرگ → واگرایی
  + α خیلی کوچک → یادگیری آهسته
  + معمولاً مقادیری مثل 0.01، 0.001 تست می‌شوند
* **معیار توقف:**
  + می‌توان وقتی تغییرات تابع هزینه کمتر از حد آستانه شد، توقف کرد:

if np.linalg.norm(gradients) < 1e-6:

break

### 8. کد کامل با رسم نمودار همگرایی

import matplotlib.pyplot as plt

theta\_path = [] *# ذخیره مسیر theta*

theta = np.random.randn(2, 1)

for iteration in range(n\_iterations):

gradients = 2/len(X\_b) \* X\_b.T.dot(X\_b.dot(theta) - y)

theta = theta - learning\_rate \* gradients

theta\_path.append(theta)

*# رسم نمودار همگرایی θ*

plt.plot(range(n\_iterations), [t[0] for t in theta\_path], label="θ₀")

plt.plot(range(n\_iterations), [t[1] for t in theta\_path], label="θ₁")

plt.xlabel("تعداد تکرارها")

plt.ylabel("مقدار θ")

plt.legend()

plt.show()

**خروجی:**  
نموداری که نشان می‌دهد چگونه θ₀ و θ₁ به مقادیر بهینه (۰ و ۲) همگرا می‌شوند.

### 9. چرا Gradient Descent؟

* برای داده‌های **بسیار بزرگ** که محاسبه معکوس ماتریس غیرممکن است.
* وقتی مدل‌های پیچیده‌تر داریم (مثل شبکه‌های عصبی).