Машинно обучение

Лабораторно упражнение 5

Линейна регресия с множество променливи. Мащабиране на променливи

Целта на упражнението е да се реализира линейна регресия, в която хипотезата включва повече от една променлива. Ще демонстрираме как се променя изпълнението на градиентното спускане при предварително мащабиране на входните променливи.

1. Линейна регресия с множество променливи

Ще построим хипотеза за предсказване на цените на къщи по два зададени параметъра – площ и брой спални.

Задачата е пример за линейна регресия с множество променливи. Търсената хипотеза в случая се дефинира с уравнение:

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

, където:

x_1 – параметър за площ на къща

x_2 – параметър за брой спални

$\theta_0, \theta_1, \theta_2$ – параметри в хипотезата.

Функцията на цената и градиентното спускане се реализират по същия начин както при линейна регресия с една променлива.

Функцията на цената:

$$J(\theta_0, \theta_1, \theta_2) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

, където

$J(\theta_0, \theta_1, \theta_2)$ – функцията на цената с параметри $\theta_0, \theta_1, \theta_2$

m – брой входни данни

$x^{(i)}$ – i-та стойност за x от входните данни (training data)

$y^{(i)}$ – i-та стойност за y от входните данни

Алгоритъмът градиентно спускане търси минимум на цената J. Състои се от итерации, на всяка от които се променят стойностите на θ по следния начин:

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

, където

θ_j – параметрите θ при j = 0, 1, 2

α – скорост на обучението (learning rate)

Файлът с реализация на задачата е regr_multiple.py. Данните за обучение на регресията са в regr_multiple.txt.

Стартирайте дадения скрипт. Ще получите следните стойности:

Минимална цена J: 56.7455026052

Стойности за theta: [9.112934 7.9060503 6.97274069]

Задача 1. Визуализирайте изменението на функцията на цената в итерациите на градиентното спускане. Стартирайте скрипта и наблюдавайте графиката. Приблизително на коя итерация градиентното спускане достига минимум на цената?

2. Мащабиране на променливи

Когато входните данни са в различни диапазони, се прилага т.нар. мащабиране (feature scaling) на стойностите на входните променливи. Целта е по-бързо (с по-малък брой итерации) достигане на минимум на ф-та J.

При мащабирането стойностите на входните променливи се коригират по следния начин:

$$x_i := \frac{x_i - \mu_i}{s_i}$$

, където:

x_i – стойност на променлива *i*

μ_i – средна стойност от всички стойности на променливата *i*

s_i – стандартно отклонение

Стандартното отклонение показва степента на отклонение на стойностите на променливата *i* от средната стойност.

Средна стойност в Python се изчислява с функцията np.mean(), а стандартно отклонение с функцията np.std().

Задача 2. Направете мащабиране на стойностите на променливите площ и брой спални (т.е. x1 и x2). Тези стойности се намират съответно в колони 0 и 1 на матрицата X. Изпълнете скрипта.

Би трябвало да получите следните стойности:

Минимална цена J = 20.4328270993

Стойности за theta: [10.93700567 -0.65006151 34.0412563]

Отново наблюдавайте графиката на функцията на цената. Има ли разлика в броя на итерациите за достигане на минимум в сравнение с варианта без мащабиране на x1 и x2?

Задача 3. Експериментирайте с параметрите на регресията скорост на обучение (alpha) и брой итерации (iterations). Препоръчително е стойностите на alpha да се променят по следния начин: 0.01, 0.03, 0.1, 0.3. Пробвайте различни стойности на двата параметъра. Наблюдавайте как се изменя цената в графиката.

Задача 4. Направете мащабирането на променливите универсално (не само за x1 и x2), т.е. да работи при добавяне на нови променливи в задачата без да се променят индексите за матрицата X в кода.

Например методът shape, приложен върху NumPy обект, връща tuple – (брой редове, брой колони), т.е. следното извикване ще намери броя на колоните на матрицата X:

```
columns_number = X.shape[1]
```