Báo cáo thực tập ngày 18/06/2024

Nguyễn Tiến Đạt - Thực tập sinh AI

1 Hồi quy tuyến tính - Linear Regression

1.1 Model

Giả sử giá nhà chỉ phụ thuộc vào diện tích căn nhà. Ta có dữ liệu về diện tích và giá nhà của 30 căn nhà như sau:

Diện tích	Giá nhà
30	448.524
32.4138	509.248
34.8276	535.104
...	...

Liệu rằng khi có một căn nhà với thông số 50 mét vuông thì ta có dự đoán được giá y của căn nhà đó không? Nếu có thì hàm dự đoán $y = f(x)$ sẽ có dạng như thế nào? Ở đây, vectơ đặc trưng $x = [30, 32.4138, 34.8276, ...]^T$ là một vectơ cột chứa thông tin đầu vào, đầu ra y là một vectơ vô hướng.

Ta có thể mô hình hoá giữa quan hệ đầu ra và đầu vào bằng một hàm tuyến tính:

$$y \approx \hat{y} = f(x) = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n = x^Tw$$

trong đó $w = [w_0, w_1, w_2, ..., w_n]^T$ là vectơ hệ số mà máy cần đi tìm đồng thời là tham số mô hình của bài toán. Mối quan hệ $y \approx \hat{y} = x^Tw$ là một mối quan hệ tuyến tính.

Vì vậy linear regression là một bài toán dự đoán giá trị đầu ra dựa trên vectơ đặc trưng của đầu vào.

1.2 Loss function

Việc tìm trọng số $w = [w_0, w_1, w_2, ..., w_n]^T$ không được máy thực hiện bằng các kĩ năng tính toán như con người mà ban đầu phải chọn một giá trị ngẫu nhiên rồi được chỉnh dần.

Giả sử ban đầu máy chọn ngẫu nhiên giá trị trọng số là $w_0 = 0, w_1 = 1$. Như vậy máy xây dựng được một đường thẳng có dạng $y = x$ và dự đoán xem giá của ngôi nhà $x(m^2)$ có giá bao nhiêu dựa trên đường thẳng vừa tìm được.

Vậy thì với hàm dự đoán $y = x$ thì tại $x = 30$ (Diện tích $30(m^2)$ giá thật là 448.524 triệu nhưng giá mà model dự đoán chỉ là 30 triệu.

Nên giờ cần một hàm để đánh giá mô hình với bộ tham số $(w_0, w_1) = (0, 1)$ có tốt hay không. Với mỗi điểm dữ liệu (x_i, y_i) độ chênh lệch giữa giá thật và giá dự đoán được tính bằng $\frac{1}{2}(\hat{y}_i - y_i)^2$. Vì vậy độ chênh lệch trên toàn bộ dữ liệu bằng tổng chênh lệnh của từng điểm:

$$J = \frac{1}{2} * \frac{1}{N}\left(\sum_{i=1}^{N}(\hat{y}_i - y_i)^2\right)$$

J được gọi là loss function, hàm để đánh giá xem bộ tham số hiện tại có tốt với dữ liệu hay không.

Nhận xét:

- J không âm.
- J càng nhỏ thì đường thẳng càng gần với các điểm dữ liệu. $J = 0$ thì đường thẳng đi qua tất cả các điểm dữ liệu.

=> Để tối ưu hoá hàm mất mát ta cần tìm trọng số w sao cho hàm J đạt giá trị nhỏ nhất.

1.3 Gradient Descent

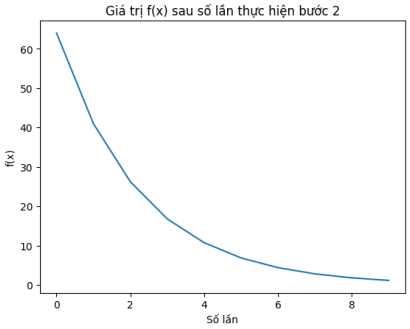
Gradient Descent là thuật toán tìm giá trị nhỏ nhất của hàm số $f(x)$ dựa trên đạo hàm.

Các bước thực hiện:

- Khởi tạo giá trị $x = x_0$ tuỳ ý.
- Gán $x = x - learning\_rate * f'(x)$. (learning_rate > 0).
- Tính lại $f(x)$: Nếu $f(x)$ đủ nhỏ thì dừng lại, ngược lại tiếp tục bước 2.

Thuật toán sẽ lặp lại bước 2 một số đủ lớn (100,1000,...tuỳ thuộc vào bài toán và hệ số learning_rate cho đến khi $f(x)$ đạt giá trị đủ nhỏ).

Ví dụ: Chọn $x = 10, learning\_rate = 0.1$, giá trị $f(x) = x^2$ sẽ thay đổi qua các lần thực hiện bước 2 như sau:



Việc chọn learning_rate cũng vô cùng quan trọng:

- learning_rate nhỏ: mỗi lần hàm số giảm rất ít nên cần rất nhiều lần thực hiện bước 2.
- learning_rate hợp lý: sau một vài lần lặp lại bước 2 vừa phải thì hàm sẽ đạt giá trị nhỏ nhất.
- learning_rate lớn: khiến hàm "vọt" qua giá trị nhỏ nhất của hàm thay vì tiến dần đến (overshoot).

1.4 Áp dụng vào loss function

Tại 1 điểm (x_i, y_i) gọi $f(w_0, w_1) = \frac{1}{2}(w_0 + w_1x_i - y_i)^2$

Ta có:

$\frac{df}{dw_0} = w_0 + w_1x_i - y_i$

$\frac{df}{dw_1} = x_i(w_0 + w_1x_i - y_i)$

Do đó:

$\frac{dJ}{dw_0} = \sum_{i=1}^{N} w_0 + w_1x_i - y_i$

$\frac{dJ}{dw_1} = \sum_{i=1}^{N} x_i(w_0 + w_1x_i - y_i)$

1.4.1 Biểu diễn dưới dạng ma trận

với mỗi điểm (x_i, y_i) ta cần phải tính $(w_0 + w_1x_i - y_i)$ nên thay vì tính cho từng điểm dữ liệu ta sẽ biểu diễn dưới dạng ma trận.

X là ma trận n * 2, Y là ma trận n * 1 (n là số điểm dữ liệu trong dataset).

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ ... & ... \\ 1 & x_n \end{bmatrix}, \ y = \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{bmatrix}, w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\hat{y} = X * W = \begin{bmatrix} w_0 + w_1x_1 \\ w_0 + w_1x_2 \\ ... \\ w_0 + w_1x_n \end{bmatrix}$$

$X[:, i]$ là ma trận kích thước n * 1 lấy dữ liệu từ cột i của ma trận X.

$$X[:,1] = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$$

$sum(X[:,1]) = x_1 + x_2 + ... + x_n$
$\frac{dJ}{dw_0} = sum(\hat{y} - y)$

$\frac{dJ}{dw_1} = sum(X[:,1] \otimes (\hat{y} - y))$

$\frac{dJ}{dw} = X^T * (\hat{y} - y)$
($\otimes$ là phép toán tích element-wise cho ma trận)

1.5 Các loss function

MAE hay MSE là 2 loss function được sử dụng cho các mô hình hồi quy, đặc biệt là mô hình hồi quy tuyến tính.

- Mean Absolute Error (MAE) hay còn được gọi là L1 Loss được tính bằng tổng các giá trị tuyệt đối của hiệu giữa giá trị thực (y_i) và giá trị mô hình dự đoán ($\hat{y_i}$).

$$MAE = \frac{\sum |y_i - \widehat{y_i}|}{n}$$

- Mean Square Error (MSE) hay còn được gọi là L2 Loss được tính bằng tổng các bình phương của hiệu giữa giá trị thực (y_i) và giá trị mà mô hình dự đoán ($\hat{y_i}$).

$$MSE = \frac{\sum (y_i - \widehat{y_i})^2}{n}$$

So sánh giữa MAE & MSE: Để tìm cực tiểu của hàm số, ta tìm đạo hàm của hàm số rồi tìm điểm mà tại đó đạo hàm của hàm số bằng 0. Như vậy việc tối ưu hoá loss function sẽ cần một bước đạo hàm. Việc tìm đạo hàm của MSE sẽ đơn giản hơn rất nhiều so với MAE. Tuy nhiên MAE sẽ đem lại kết quả tốt hơn so với các dữ liệu có điểm outlier (điểm dị biệt - những điểm có giá trị khác xa so với phần còn lại của dữ liệu).

Xét ví dụ: Giả sử có 4 điểm dữ liệu 1, 2, 4, 33.
Tìm min của:
a)$L1 = |x-1| + |x-2| + |x-4| + |x-33|$
+) $-oo < x \leq 1 : L1 = 40 - 4x$ đạt min = 36 tại x = 1
+)$1 < x \leq 2 : L1 = 38 - 2x$ đạt min = 34 tại x = 2
+)$2 < x \leq 4 : L1 = 34$
+)$4 < x \leq 33 : L1 = 2x + 26 > 34$
+)$33 < x < +oo : L1 = 4x - 40$ đạt min > 4 * 33-40 = 92
Vậy L1 đạt min = 34 tại $1 < x \leq 2$.

b)$L2 = (x-1)^2 + (x-2)^2 + (x-4)^2 + (x-33)^2$
$L2' = 2(x - 1 + x - 2 + x - 4 + x - 33) = 0 => x = 10$
Vậy L2 đạt min = 710 tại x = 10

Trong 4 điểm dữ liệu thì 33 là outlier, L2 đạt min lớn hơn rất nhiều so với L1. L2 đạt min tại trung bình của các giá trị ($\frac{1+2+4+33}{4} = 10$). Trong khi L1 đạt min tại trung vị của các giá trị đó. Do đó L1 tốt hơn L2 với dataset có các outlier.

1.6 Giải bài toán

$w = [w_0, w_1, w_2, ..., w_m]^T$ là vectơ hệ số cần phải tối ưu, w_0 thường được gọi là *bias*.
$x^{(i)} = \left[1, x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)}\right]$ là dữ liệu thứ i trong bộ n số lượng dữ liệu, mỗi dữ liệu có m giá trị.

Biểu diễn tổng quát bài toán Linear Regression:
$X \in \mathbb{R}^{n*(m+1)}, w \in \mathbb{R}^{(m+1)*1}, y \in \mathbb{R}^{n*1}$

$$X = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ ... \\ x^{(n)} \end{pmatrix}, w = \begin{pmatrix} w_0 \\ w_1 \\ ... \\ w_m \end{pmatrix}$$

$$\hat{y} = Xw = \begin{pmatrix} w_0 + w_1 x_1^{(1)} + ... + w_m x_m^{(1)} \\ w_0 + w_1 x_1^{(2)} + ... + w_m x_m^{(2)} \\ ... \\ w_0 + w_1 x_1^{(n)} + ... + w_m x_m^{(n)} \end{pmatrix}$$

$$y - \hat{y} = \begin{pmatrix} y^{(1)} - \left(w_0 + w_1 x_1^{(1)} + ... + w_m x_m^{(1)}\right) \\ y^{(2)} - \left(w_0 + w_1 x_1^{(2)} + ... + w_m x_m^{(2)}\right) \\ ... \\ y^{(n)} - \left(w_0 + w_1 x_1^{(n)} + ... + w_m x_m^{(n)}\right) \end{pmatrix}$$

Lost function:

$$L = \frac{1}{2} * \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \widehat{y^{(i)}}\right)^2 = \frac{1}{2} * \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - x^{(i)} w\right)^2$$
Định nghĩa chuẩn 2: $||z||_2^2 = \left(z_1^2 + z_2^2 + ... + z_n^2\right)$

$$\Rightarrow L = \frac{1}{2} * \frac{1}{n} ||y - \hat{y}||_2^2 = \frac{1}{2} * \frac{1}{n} ||y - Xw||_2^2 = \frac{1}{2} * \frac{1}{n} * (y - \hat{y})^T * (y - \hat{y})$$

Giải:

$$+) x^{(i)} w = [1 \quad x_1^{(i)} \quad ... \quad x_m^{(i)}] \begin{bmatrix} w_0 \\ w_1 \\ ... \\ w_m \end{bmatrix} = w_0 + w_1 x_1^{(i)} + ... + w_m x_m^{(i)}$$

$$\Rightarrow \frac{d\left(x^{(i)} w\right)}{dw} = [\frac{d\left(x^{(i)} w\right)}{dw_0} \quad \frac{d\left(x^{(i)} w\right)}{dw_1} \quad ... \quad \frac{d\left(x^{(i)} w\right)}{dw_m}] = [1 \quad x_1^{(i)} \quad ... \quad x_m^{(i)}]$$

$$\Rightarrow \frac{d(Xw)}{dw} = X$$

$$+) w x^{(i)} = \begin{bmatrix} w_0 \\ w_1 \\ ... \\ w_m \end{bmatrix} [1 \quad x_1^{(i)} \quad ... \quad x_m^{(i)}] = w_0 + w_1 x_1^{(i)} + ... + w_m x_m^{(i)}$$

$$\Rightarrow \frac{d\left(w x^{(i)}\right)}{dw} = \begin{bmatrix} \frac{d\left(w x^{(i)}\right)}{dw_0} \\ \frac{d\left(w x^{(i)}\right)}{dw_1} \\ ... \\ \frac{d\left(w x^{(i)}\right)}{dw_m} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ ... \\ x_m^{(i)} \end{bmatrix} = x^{(i)T}$$

$$\Rightarrow \frac{d(wX)}{dw} = X^T$$
$$+) (Xw)^T = w^T X^T$$

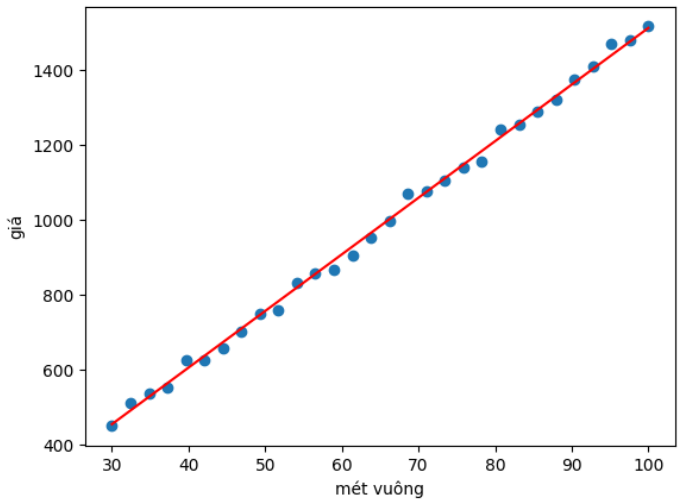
$$+) \frac{d\left(w^T X w\right)}{dw} = w^T X + \frac{d\left((Xw)^T w\right)}{dw} = w^T X + w^T X^T = w^T \left(X + X^T\right)$$

$$\Rightarrow \frac{dX}{dw} = 0, \ \frac{d(Xw)}{dw} = X, \ \frac{d(wX)}{dw} = X^T, \frac{d\left(w^T X w\right)}{dw} = w^T \left(X + X^T\right)$$

$A = (y - \hat{y})^T * (y - \hat{y}) = (y - Xw)^T * (y - Xw)$
$= \left(y^T - w^T X^T\right)(y - Xw) = y^T y - y^T X w - w^T X^T y + w^T X^T X w$
$\Rightarrow \frac{dA}{dw} = 0 - y^T X - \frac{d\left(\left(y^T X\right) w\right)^T}{dw} + 2 w^T X^T X$
$= -y^T X - \frac{d\left(\left(y^T X\right) w\right)}{dw} + 2 w^T X^T X$
$= -y^T X - y^T X + 2 w^T X^T X$
$A'_w = 0 \Rightarrow 2 y^T X = 2 w^T X^T X$
$\Rightarrow \left((Xw)^T X\right)^T = \left(y^T X\right)^T$
$\Rightarrow \left(X^T X\right) * w = X^T y$
$\Rightarrow w = \left(X^T X\right)^{-1} X^T y$

1.7 Thực hành

1.7.1 Hiển thị dữ liệu trên đồ thị



Các điểm dữ liệu được minh hoạ bởi điểm màu xanh được sắp xếp gần như theo một đường thẳng, vậy mô hình linear regression có khả năng cho kết quả tốt:

(giá nhà) = w_0 + w_1 * (diện tích)

1.7.2 Nghiệm trên công thức

Tính toán các hệ số w_0 và w_1 dựa vào công thức được xây dựng ở trên.

```
x = data[:, 0].reshape(-1, 1)
y = data[:, 1].reshape(-1, 1)
plt.scatter(x, y)
plt.xlabel('m^2')
plt.ylabel('price')
x = np.hstack((np.ones((N, 1)), x))
w = np.array([0.,1.]).reshape(-1,1)
numOfIteration = 100
cost = np.zeros((numOfIteration,1))
learning_rate = 0.00001
for i in range(0, numOfIteration):
    r = np.dot(x, w) - y
    cost[i] = 0.5*np.sum(r*r)
    w[0] -= learning_rate*np.sum(r)
    # correct the shape dimension
    w[1] -= learning_rate*np.sum(np.multiply(r
        , x[:,1].reshape(-1, 1)))
```

Bây giờ sử dụng mô hình để dự đoán giá một căn nhà có diện tích $50m^2$:

```
x1 = 50
y1 = w[0] + w[1] * x1
print('House price for 50m^2 using formula is:
    ', y1[0])
```

Kết quả:

```
    House price for 50m^2 using formula is
        :755.6826304680484
```

1.7.3 Nghiệm theo thư viện scikit-learn

```
x = data[:, 0].reshape(-1, 1)
y = data[:, 1].reshape(-1, 1)
lrg = LinearRegression()
lrg.fit(x, y)
x_test = [[50]]
y_pred = lrg.predict(x_test)
print('House price for 50m^2 using library is:
    ', y_pred[0][0])
```

Kết quả:

```
    House price for 50m^2 using library is
        :753.490271338277
```