

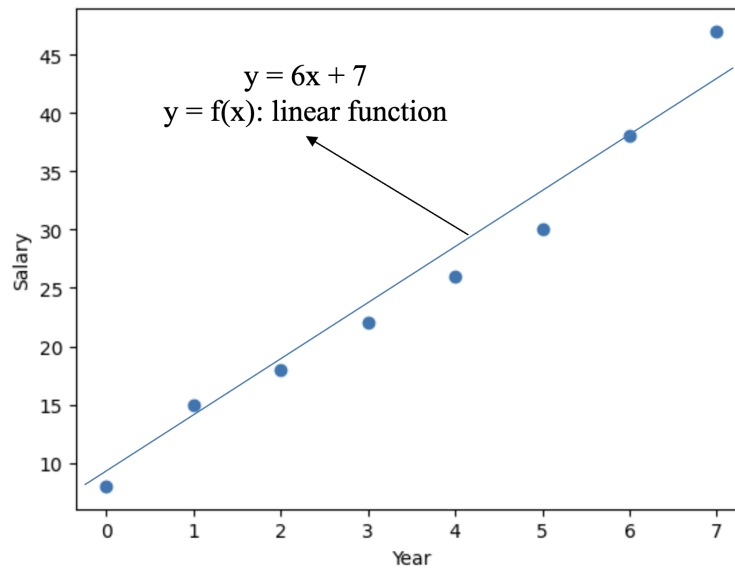
# Project: Sales Prediction

Ngày 18 tháng 10 năm 2024

**Hồi quy tuyến tính - Linear Regression** là mô hình phân tích mối quan hệ phụ thuộc của  $Y$  với một hay nhiều biến  $X$  sử dụng hàm tuyến tính (hàm đa thức bậc 1). Các tham số của mô hình (hay hàm số) được ước lượng từ dữ liệu huấn luyện.

Mô hình hồi quy tuyến tính cho bài toán dự đoán mức lương dựa vào số năm kinh nghiệm được mô tả như sau:  $salary = year * w + b$ . Phương trình tuyến tính xác định bởi 2 tham số  $w$  và  $b$ .

## Visualization



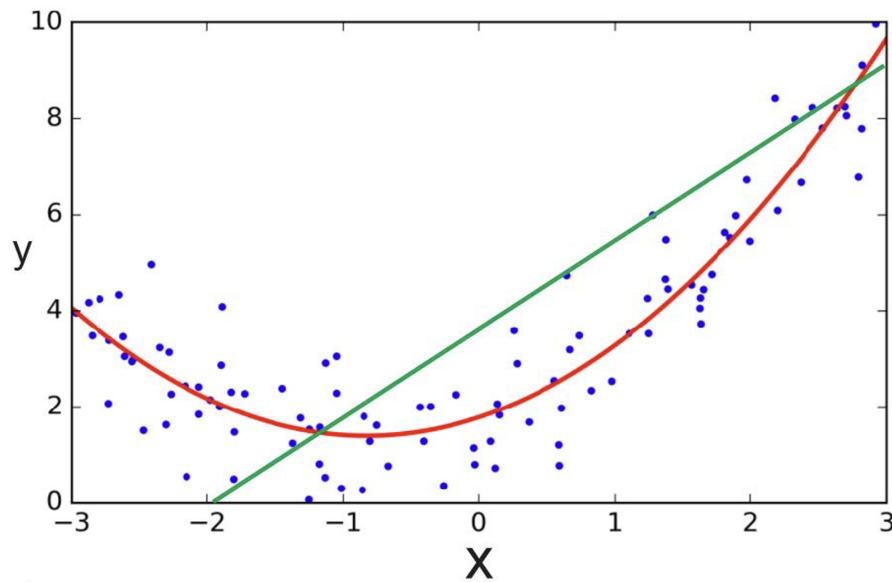
Hình 1: Mô tả dữ liệu dự đoán mức lương dựa vào số năm kinh nghiệm sử dụng mô hình hồi quy tuyến tính.

Dựa vào dữ liệu huấn luyện thông qua phương pháp học gradient descent để tìm bộ tham số tối ưu cho phương trình tuyến tính. Trong mỗi lần học, từ dữ liệu huấn luyện tính toán giá trị dự đoán, sau đó kết hợp với giá trị cần dự đoán trong thực tế để tính toán hàm mất mát (loss), rồi cập nhật lại tham số mô hình dựa vào tính đạo hàm. Quá trình huấn luyện cần chọn lựa các tham số ảnh hưởng đến kết quả của mô hình như tốc độ học (learning rate), giá trị khởi tạo của tham số và số vòng lặp thực hiện cập nhật trọng số).

Tuy nhiên, một trong những nhược điểm của mô hình hồi quy tuyến tính là giá trị dự đoán sự phụ thuộc tuyến tính của các giá trị đầu vào và tham số mô hình. Điều này dẫn tới khả năng tổng quát hoá của mô hình kém và không tối ưu cho dữ liệu trong thực tế. Vì vậy, mở rộng mối quan hệ phụ thuộc tuyến tính thông qua xây dựng hàm tuyến tính bằng các hàm phi tuyến sẽ giúp mô hình có tính tổng quát hơn. Các mô hình sử dụng các hàm phi tuyến với các giá trị đặc trưng đầu vào được gọi là hồi quy

phi tuyến (Nonlinear Regression). Một số phi tuyến điển hình như hàm đa thức (polynomial), hàm mũ hoặc hàm logarit,...

Ví dụ, hàm đa thức bậc 2 cho bài toán dự đoán lương dựa vào năm kinh nghiệm được mô tả như hình sau:



Hình 2: Mô tả dữ liệu dự đoán mức lương dựa vào năm kinh nghiệm sử dụng mô hình hồi quy đa thức bậc 2.

# 1. Linear regression

Để củng cố kiến thức về hồi quy tuyến tính, dựa vào file colab và đoạn code sau đây, để trả lời các câu hỏi.

```

1 class CustomLinearRegression:
2     def __init__(self, X_data, y_target, learning_rate=0.01, num_epochs=10000):
3         self.num_samples = X_data.shape[0]
4         self.X_data = np.c_[np.ones((self.num_samples, 1)), X_data]
5         self.y_target = y_target
6         self.learning_rate = learning_rate
7         self.num_epochs = num_epochs
8
9         # Initial weights
10        self.theta = np.random.randn(self.X_data.shape[1], 1)
11        self.losses = []
12
13    def compute_loss(self, y_pred, y_target):
14        *** Your Code Here ***
15        return loss
16
17    def predict(self, X_data):
18        *** Your Code Here ***
19        return y_pred
20
21    def fit(self):
22        for epoch in range(self.num_epochs):
23
24            *** Your Code Here ***
25
26            if (epoch % 50) == 0:
27                print(f'Epoch: {epoch} - Loss: {loss}')
28
29        return {
30            'loss': sum(self.losses)/len(self.losses),
31            'weight': self.theta
32        }

```

**Câu hỏi 1** Dòng code nào sau đây giúp hoàn thiện hàm *predict()*:

- a) `y_pred = X_data.dot(self.theta)`
- a) `y_pred = X_data.dot(theta)`
- c) `y_pred = X_data.dot(self.theta)**2`
- d) `y_pred = X_data**2.dot(self.theta)`

**Câu hỏi 2** Sắp xếp các đoạn code sau đây để hoàn thiện code trong hàm *fit()* cho quá trình huấn luyện mô hình Linear Regression:

```

1 A)
2 # compute gradient
3 loss_grd = 2*(y_pred-self.y_target)/self.num_samples
4 gradients = self.X_data.T.dot(loss_grd)
5
6 B)
7 # compute loss
8 loss = self.compute_loss(y_pred, self.y_target)
9 self.losses.append(loss)
10

```

3

2

```

11 C)
12 # predict
13 y_pred = self.predict(self.X_data)
14
15 D)
16 # update weight
17 self.theta = self.theta - self.learning_rate*gradients

```

Thứ tự đúng của các đoạn trên là:

- a) D - C - B - A
- b) C - D - B - A
- ~~c) C - B - A - D~~
- d) C - B - D - A

**Câu hỏi 3** Định nghĩa hàm tính hệ số xác định (Coefficient of determination) để đánh giá tính hiệu quả của mô hình hồi quy tuyến tính

Hệ số xác định ( $R^2$ ) được xác định dựa vào công thức sau:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Chọn đáp án đúng trong các đáp án sau đây:

```

1 A)
2 def r2score(y_pred, y):
3     rss = np.sum((y_pred - y) ** 2)
4     tss = np.sum((y - y.mean()) ** 2)
5     r2 = 1 - (rss / tss)
6     return r2
7
8 B)
9 def r2score(y_pred, y):
10    rss = np.sum((y_pred - y))
11    tss = np.sum((y - y.mean()) ** 2)
12    r2 = 1 - (rss / tss)
13    return r2
14
15 C)
16 def r2score(y_pred, y):
17    rss = np.sum((y_pred - y) ** 2)
18    tss = np.sum((y - y_pred.mean()) ** 2)
19    r2 = 1 - (rss / tss)
20    return r2
21
22 D)
23 def r2score(y_pred, y):
24    rss = np.sum((y_pred - y) ** 2)
25    tss = np.sum((y - y.mean()) ** 2)
26    r2 = 1 - (rss / tss)
27    return r2

```

**Câu hỏi 4** Cho giá trị  $y\_pred$  và  $y$  như sau dựa vào hàm  $r2score$  được định nghĩa ở câu 3, tính các giá trị  $r2score$  tương ứng sau đây:

```

1 # Case 1
2 y_pred = np.array([1, 2, 3, 4, 5])
3 y = np.array([1, 2, 3, 4, 5])

```

```
4 r2score(y_pred, y)
5
6 # Case 2
7 y_pred = np.array([1, 2, 3, 4, 5])
8 y = np.array([3, 5, 5, 2, 4])
9 r2score(y_pred, y)
```

Giá trị  $r2score(y\_pred, y)$  lần lượt là:

- a) -2.2 và 1.0
- b) 1.0 và -2.2
- c) 1.0 và -4.4
- d) -4.4 và 1.0

## 2. Polynomial Regression

**Câu hỏi 5** Hàm số nào sau đây không phải là hàm phi tuyến:

- a)  $y = 5x^2 + 3x$
- b)  $y = 6x + 7$
- c)  $y = x^3 + 2x^2 + 3$
- d)  $y = \ln(x)$

**Câu hỏi 6** Phát biểu nào sau đây là đúng nhất về polynomial regression

- a) Giá trị  $y$  phụ thuộc tuyến tính vào giá trị đầu vào  $X$
- b) Giá trị  $y$  phụ thuộc phi tuyến vào giá trị đầu vào  $X$  và tuyến tính vào giá trị trọng số  $\theta$
- c) Giá trị  $y$  phụ thuộc tuyến tính vào giá trị trọng số  $\theta$
- d) Giá trị  $y$  phụ thuộc phi tuyến vào giá trị đầu vào  $X$  và phi tuyến vào giá trị trọng số  $\theta$

### Polynomial Feature

Các đặc trưng phi tuyến cho hàm đa thức (polynomial feature) được tính toán như hình dưới. Với giá trị đầu vào  $x = 2$ , hàm đa thức bậc 3 sẽ tạo ra các feature tương ứng là  $x^1 = 2, x^2 = 4, x^3 = 8$ . Từ đó ta chuyển từ bài toán đơn biến chỉ có một giá trị  $x$  sang bài toán hồi quy tuyến tính đa biến.

$\psi(\varphi(i))$		
Input	$\varphi(i)$	$\varphi(i)^2$
0	0	0
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

Hình 3: Ví dụ mô tả quá trình tạo các polynomial feature từ một dữ liệu đầu vào.

Examples

Input:

```
X = np.array([[1],
               [2],
               [3]])
```

```
degree = 2
```

Output:

```
X_poly = np.array([[1, 1],
                   [2, 4],
                   [3, 9]])
```

Hoàn thành đoạn code sau đây để tạo các polynomial feature từ một dữ liệu đầu vào:

```
1 def create_polynomial_features(X, degree=2):
2     """Creates the polynomial features
3     Args:
4         X: A array tensor for the data.
5         degree: A intege for the degree of
6             the generated polynomial function.
7     """
8     ** Your code here **
9     return X_new
10
11 X = np.array([[1], [2], [3]])
```

**Câu hỏi 7** Chọn đáp án đúng trong các đáp án sau đây:

```
1 A)
2 X_new = X
3     for d in range(2, degree+1):
4         X_new = np.c_[X_new, np.power(X, d)]
5
6 B)
7 X_new = X
8     for d in range(2, degree+1):
9         X_new = np.c_[X_new, np.add(X, d)]
10
11 C)
12 X_new = X
13     for d in range(2, degree+1):
14         X_new = np.c_[X_new, np.square(X, d)]
15
16 D)
17 X_new = X
18     for d in range(1, degree+1):
19         X_new = np.c_[X_new, np.power(X, d)]
```

Xây dựng các đặc trưng phi tuyến cho mô hình đa biến thông qua việc xem xét các biến này phi tuyến riêng lẻ với nhau. Quá trình này được mô tả như hình sau:

1	$\varphi(i)$		$\varphi(i)^2$	
1	0	2	0	4
1	1	1	1	1
1	2	2	4	4
1	3	1	9	1
1	4	2	16	4
1	5	1	25	1

Hình 4: Ví dụ mô tả quá trình tạo các polynomial feature từ nhiều dữ liệu đầu vào và xem xét các dữ liệu đầu vào phi tuyến độc lập với nhau.

**Câu hỏi 8** Hàm nào trong các hàm sau đây phù hợp để xây dựng các đặc trưng phi tuyến được mô tả như hình trên:

```
Examples
Input:
X = np.array([[1, 2],
              [2, 3],
              [3, 4]])

degree = 2

Output:
X_poly = np.array([[1, 1, 2, 4],
                  [2, 4, 3, 9],
                  [3, 9, 4, 16]])
```

```
1 A)
2 def create_polynomial_features(X, degree=2):
3     """Creates the polynomial features
4     Args:
5         X: A array for the data.
6         degree: A integer for the degree of
7             the generated polynomial function.
8     """
9     X_mem = []
10    for X_sub in X.T:
11        X_new = X_sub
12        for d in range(2, degree+1):
13            X_new = np.c_[X_new, np.power(X_sub, d)]
14        X_mem.extend(X_new.T)
15    return np.c_[X_mem].T
16
17 B)
18 def create_polynomial_features(X, degree=2):
19     """Creates the polynomial features
20     Args:
21         X: A array for the data.
22         degree: A integer for the degree of
23             the generated polynomial function.
```



```

24     """
25     X_mem = []
26     for X_sub in X.T:
27         X_sub = X_sub.T
28         X_new = X_sub
29         for d in range(2, degree+1):
30             X_new = np.c_[X_new, np.power(X_sub, d)]
31         X_mem.append(X_new.T)
32     return np.c_[X_mem].T
33
34 C)
35 def create_polynomial_features(X, degree=2):
36     """Creates the polynomial features
37     Args:
38         X: A array for the data.
39         degree: A intege for the degree of
40             the generated polynomial function.
41     """
42     X_mem = []
43     for X_sub in X.T:
44         X_sub = X_sub.T
45         X_new = X_sub
46         for d in range(2, degree+1):
47             X_new = np.c_[X_sub, np.power(X_sub, d)]
48         X_mem.extend(X_new.T)
49     return np.c_[X_mem].T
50
51 D)
52 def create_polynomial_features(X, degree=2):
53     """Creates the polynomial features
54     Args:
55         X: A array for the data.
56         degree: A intege for the degree of
57             the generated polynomial function.
58     """
59     X_mem = []
60     for X_sub in X.T:
61         X_sub = X_sub.T
62         X_new = X_sub
63         for d in range(2, degree+1):
64             X_new = np.c_[X_new, np.power(X_sub, d)]
65         X_mem.extend(X_new.T)
66     return np.c_[X_mem].T

```

### 3. Sales Prediction

#### a. Load dataset

Bộ dữ liệu dự đoán giá quảng cáo của các chiến dịch khác nhau (file SalesPrediction.csv) có thể được tải về **tại đây** được mô tả như hình sau:

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181

Hình 5: Dữ liệu bài toán dự đoán giá quảng cáo.

#### b. Preprocessing

Trong đó trường thuộc tính *Influencer* thuộc kiểu dữ liệu text (các nhãn) - category. Vì vậy, sử dụng **one-hot encoding** để mã hoá dữ liệu dạng nhãn thành số hoặc vector để xử lý trường thuộc tính này

**Câu hỏi 9** Đoạn code nào sau đây để chuyển trường dữ liệu *Influencer* theo phương pháp One Hot Encoding. Gợi ý kết quả thu được như sau:

	TV	Radio	Social Media	Sales	Influencer_Macro	Influencer_Mega	Influencer_Micro	Influencer_Nano
0	16.0	6.566231	2.907983	54.732757	False	True	False	False
1	13.0	9.237765	2.409567	46.677897	False	True	False	False
2	41.0	15.886446	2.913410	150.177829	False	True	False	False
3	83.0	30.020028	6.922304	298.246340	False	True	False	False
4	15.0	8.437408	1.405998	56.594181	False	False	True	False

Hình 6: Dữ liệu sau khi mã hoá thông tin cho Influencer.

```
a) df = pd.get_dummies(df)
b) df = pd.get_dummies(~df)
c) df = pd.groupby(df)
d) df = pd.groupby(~df)
```

Sau đó, các giá trị Null sẽ được thay thế bởi giá trị trung bình cho mỗi thuộc tính và được chia thành 2 tập train : test với tập test 30%.

```
# Handle Null values
df = df.fillna(df.mean())

# Get features
X = df[['TV', 'Radio', 'Social Media', 'Influencer_Macro', 'Influencer_Mega', 'Influencer_Micro', 'Influencer_Nano']]
y = df[['Sales']]

# Train Test Split
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.33,
    random_state=0
)
```

### c. Feature Scaling

Tiếp theo, sử dụng StandardScaler để chuẩn hoá các đặc trưng.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_processed = scaler.fit_transform(X_train)
scaler.mean_[0]
```

**Câu hỏi 10** Kết quả của đoạn code trên xấp xỉ với kết quả nào sau đây? a) 53.99

- b) 18.22
- c) 3.33
- d) 0.25

### d. Polynomial Features

Tạo ra các đặc trưng phi tuyến bằng cách sử dụng hàm định nghĩa trong phần 2 hoặc thông qua thư viện Sklearn như sau:

```
from sklearn.preprocessing import PolynomialFeatures

poly_features = PolynomialFeatures(degree=2)

X_train_poly = poly_features.fit_transform(X_train_processed)
X_test_poly = poly_features.transform(X_test_processed)
```

### e. Training & Evaluation

Cuối cùng, sử dụng đặc trưng phi tuyến để huấn luyện và đánh giá mô hình:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

preds = poly_model.predict(X_test_poly)
r2_score(y_test, preds)
```

- Hết -