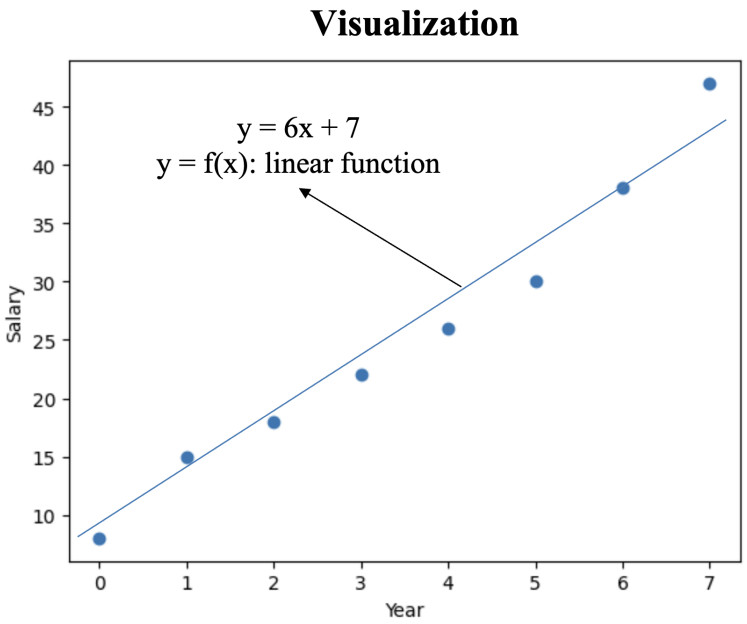
AI VIET NAM – COURSE 2023

# Project: Linear and non-linear Regressions

Ngày 9 tháng 7 năm 2023

**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**.



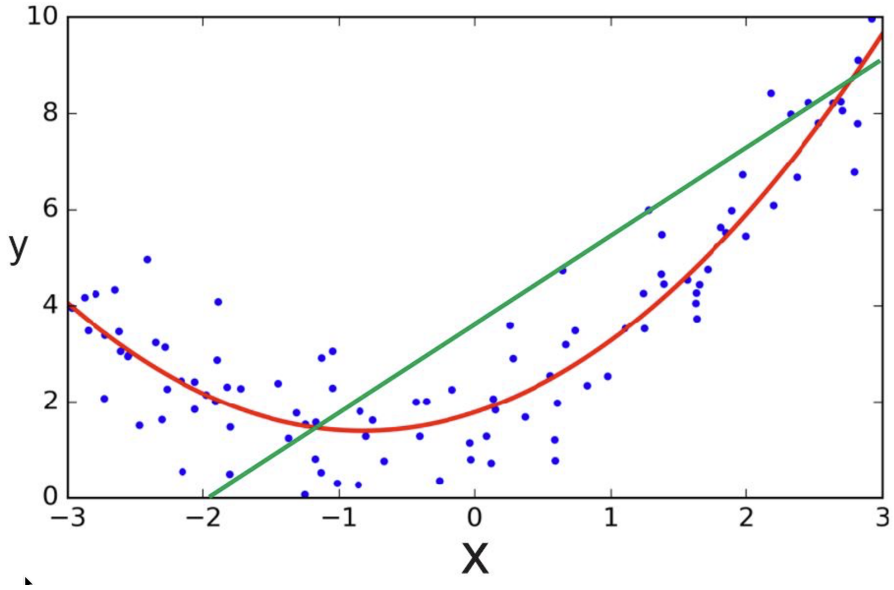

Hình 1: Mô tả dữ liệu dự đoán mức lương dựa vào số năm kinh nghiệm sử dụng mô hình hồi quy tuyến tính.

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.

```
class LinearRegression:
    def __init__(self, X_data, y_target, learning_rate=0.01, num_epochs=10000):
        self.X_data = X_data # Shape: (num_samples, num_features)
        self.y_target = y_target # Shape: (num_samples, )
        self.learning_rate = learning_rate
        self.num_epochs = num_epochs
        self.num_samples = self.X_data.shape[0]

        # Initial Coefficients
        self.theta = np.random.randn(self.X_data.shape[1])
        self.losses = []

    def compute_loss(self, y_pred, y_target):
        loss = (y_pred-y_target)*(y_pred-y_target)
        loss = np.mean(loss)
        return loss

    def predict(self, X_data):

        *** Your code here ***

        return y_pred

    def fit(self):
        for epoch in range(self.num_epochs):

            *** Your code here ***

            print(f'Epoch: {epoch} - Loss: {loss}')

        return {
            'loss': sum(self.losses)/len(self.losses),
            'weight': self.theta
        }
```

**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 nào sau đây là đúng để hoàn thiện code trong hàm *fit*():
Đoạn A:

```
# compute gradient
k = 2*(y_pred-self.y_target)
gradients = self.X_data.T.dot(k)/self.num_samples
```

Đoạn B:

```
# compute loss
loss = self.compute_loss(y_pred, self.y_target)
self.losses.append(loss)
```

Đoạn C:

```
# predict
y_pred = self.predict(self.X_data)
```

Đoạn D:

```
# update weight
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:

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

Chọn đáp án đúng trong các đáp án sau đây:

```
(a) def r2score(y_pred, y):
        rss = np.sum((y_pred - y) ** 2)
        tss = np.sum((y-y.mean()))
        r2 = 1 - (rss / tss)
        return r2
```

```
(b) def r2score(y_pred, y):
        rss = np.sum((y_pred - y))
        tss = np.sum((y-y.mean()) ** 2)
        r2 = 1 - (rss / tss)
        return r2
```

```
(c) def r2score(y_pred, y):
        rss = np.sum((y_pred - y) ** 2)
        tss = np.sum((y-y_pred.mean()) ** 2)
        r2 = 1 - (rss / tss)
        return r2
```

```
(d) def r2score(y_pred, y):
        rss = np.sum((y_pred - y) ** 2)
        tss = np.sum((y-y.mean()) ** 2)
        r2 = 1 - (rss / tss)
        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:

```
y_pred = np.array([1, 2, 3, 4, 5])
y = np.array([1, 2, 3, 4, 5])
r2score(y_pred, y)
```

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

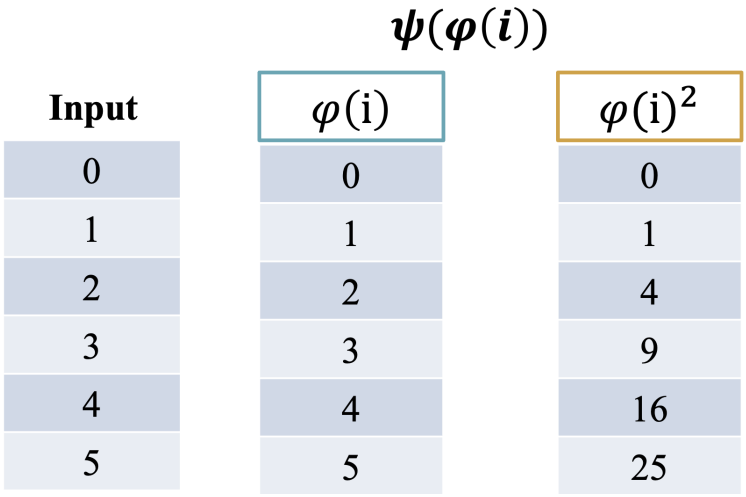
# 2. Polynomial Regression

**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.




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:**

```
def create_polynomial_features(X, degree=2):
    """Creates the polynomial features
    Args:
        X: A array tensor for the data.
        degree: A intege for the degree of
        the generated polynomial function.
    """
    ** Your code here **
    return X_new

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

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

```
(a) X_new = X
    for d in range(2, degree+1):
        X_new = np.c_[X_new, np.power(X, d)]
```

```
(b) X_new = X
    for d in range(2, degree+1):
        X_new = np.c_[X_new, np.add(X, d)]
```

```
(c) X_new = X
    for d in range(2, degree+1):
        X_new = np.c_[X_new, np.square(X, d)]
```

```
(d) X_new = X
    for d in range(1, degree+1):
        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 lẽ với nhau. Quá trình này được mô tả như hình sau:**

$$(a+b)^2 => a^2 + b^2 + ab + a + b + 1$$

Multivariable

| $X$ | | $\varphi(i)$ | | $\varphi(i)^2$ | | |
|---|---|---|---|---|---|---|
| 0 | 2 | 0 | 2 | 0 | 4 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| 3 | 1 | 3 | 1 | 9 | 1 | 3 |
| 4 | 2 | 4 | 2 | 16 | 4 | 8 |
| 5 | 1 | 5 | 1 | 25 | 1 | 5 |

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]])
```

a)
```
def create_polynomial_features(X, degree=2):
    """Creates the polynomial features
    Args:
        X: A array for the data.
        degree: A intege for the degree of
        the generated polynomial function.
    """
    X_mem = []
    for X_sub in X.T:
        X_new = X_sub
        for d in range(2, degree+1):
            X_new = np.c_[X_new, np.power(X_sub, d)]
        X_mem.extend(X_new.T)
    return np.c_[X_mem].T
```

b)
```
def create_polynomial_features(X, degree=2):
    """Creates the polynomial features
    Args:
```

```
            X: A array for the data.
            degree: A intege for the degree of
            the generated polynomial function.
        """
        X_mem = []
        for X_sub in X.T:
            X_sub = X_sub.T
            X_new = X_sub
            for d in range(2, degree+1):
                X_new = np.c_[X_new, np.power(X_sub, d)]
            X_mem.append(X_new.T)
        return np.c_[X_mem].T
```

c)
```
def create_polynomial_features(X, degree=2):
        """Creates the polynomial features
        Args:
            X: A array for the data.
            degree: A intege for the degree of
            the generated polynomial function.
        """
        X_mem = []
        for X_sub in X.T:
            X_sub = X_sub.T
            X_new = X_sub
            for d in range(2, degree+1):
                X_new = np.c_[X_sub, np.power(X_sub, d)]
            X_mem.extend(X_new.T)
        return np.c_[X_mem].T
```

d)
```
def create_polynomial_features(X, degree=2):
        """Creates the polynomial features
        Args:
            X: A array for the data.
            degree: A intege for the degree of
            the generated polynomial function.
        """
        X_mem = []
        for X_sub in X.T:
            X_sub = X_sub.T
            X_new = X_sub
            for d in range(2, degree+1):
                X_new = np.c_[X_new, np.power(X_sub, d)]
            X_mem.extend(X_new.T)
        return np.c_[X_mem].T
```

**Bộ dữ liệu dự đoán khối lượng của của cá (file Fish.csv) được mô tả như hình sau:**

| | Species | Weight | VerticalLen | DiagonalLen | CrossLen | Height | Width |
|---|---|---|---|---|---|---|---|
| **0** | Bream | 242.0 | 23.2 | 25.4 | 30.0 | 11.5200 | 4.0200 |
| **1** | Bream | 290.0 | 24.0 | 26.3 | 31.2 | 12.4800 | 4.3056 |
| **2** | Bream | 340.0 | 23.9 | 26.5 | 31.1 | 12.3778 | 4.6961 |
| **3** | Bream | 363.0 | 26.3 | 29.0 | 33.5 | 12.7300 | 4.4555 |
| **4** | Bream | 430.0 | 26.5 | 29.0 | 34.0 | 12.4440 | 5.1340 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **154** | Smelt | 12.2 | 11.5 | 12.2 | 13.4 | 2.0904 | 1.3936 |
| **155** | Smelt | 13.4 | 11.7 | 12.4 | 13.5 | 2.4300 | 1.2690 |
| **156** | Smelt | 12.2 | 12.1 | 13.0 | 13.8 | 2.2770 | 1.2558 |
| **157** | Smelt | 19.7 | 13.2 | 14.3 | 15.2 | 2.8728 | 2.0672 |
| **158** | Smelt | 19.9 | 13.8 | 15.0 | 16.2 | 2.9322 | 1.8792 |

159 rows × 7 columns

Hình 5: Dữ liệu bài toán dự đoán khối lượng của cá.

Trong đó trường thuộc tính *Species* thuộc kiểu dữ liệu text (các nhãn) - category. Vì vậy, các phương pháp mã hoá dữ liệu dạng nhãn thành số hoặc vector để xử lý trường thuộc tính này. Trong đó có hai phương pháp cơ bản điển hình là One Hot Encoding và Label Encoding.

**Câu hỏi 9** Đoạn code nào sau đây để chuyển trường dữ liệu *Species* theo phương pháp One Hot Encoding

```
a) encode_species = pd.get_dummies(df.Species)

b) encode_species = pd.get_dummies(~df.Species)

c) encode_species = pd.groupBy(df.Species)

d) encode_species = pd.groupBy(~df.Species)
```

**Câu hỏi 10** Đoạn code nào sau đây để chuyển trường dữ liệu *Species* theo phương pháp Label Encoding

```
a) df["Species"] = df["Species"].astype('category')
    label_ecnoding_species = df["Species"].cat.codes

b) label_ecnoding_species = df["Species"].cat.codes

c) df["Species"] = df["Species"].astype('numerical')
    label_ecnoding_species = df["Species"].cat.codes

d) label_ecnoding_species = df["Species"].cat.numerical
```

***- Hết -***