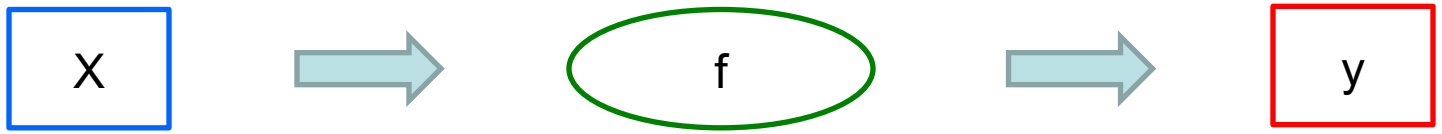
CHƯƠNG 4:
HỒI QUY (REGRESSION) VÀ CÁC THUẬT TOÁN MÁY HỌC

MÔ HÌNH VÀ THUẬT TOÁN MÁY HỌC

Huấn luyện mô hình



- Vấn đề đặt ra: đi tìm f dựa vào X và y
  + X và y ở đây chính là dữ liệu huấn luyện.
- Để tìm được f: dựa vào dữ liệu huấn luyện ➔ huấn luyện mô hình.
- Mục tiêu: dự đoán dữ liệu X' mới chưa biết nhãn trong tương lai.
- Vấn đề: làm sao biết được f tốt ➔ đánh giá mô hình.
- Cơ sở đánh giá: các độ đo đánh giá (evaluation metric).

MÔ HÌNH LÀ GÌ

- Mô hình về bản chất là một bộ các tham số (parameters).
- Mục tiêu đặt ra: tìm ra bộ tham số tối ưu nhất ➔ công việc của các thuật toán máy học.
- Để tìm được tham số tối ưu nhất: cần dựa trên độ đo đánh giá.
  + Đối với bài toán phân lớp, mô hình được xem là tối ưu nhất khi số dữ liệu bị dự đoán sai là ít nhất.
- Hàm mất mát (loss function) là hàm số mô tả mức độ sai lệch giữa dữ liệu dự đoán và dữ liệu thực tế.
- Mô hình tốt nhất ➔ dữ liệu bị dự đoán sai là ít nhất ➔ hàm mất mát đạt giá trị nhỏ nhất.

➔ Tối ưu hàm mất mát là công việc của một thuật toán máy học.

Hàm mất mát (loss function)

– Gọi θ là tập tham số của mô hình máy học.

– $\mathcal{L}(\theta)$ là hàm mất mát của mô hình.

Bài toán đi tìm tập tham số tối ưu của mô hình được định nghĩa toán học như sau:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}(\mathcal{L}(\theta))$$

tập tham số tối ưu

Quá trình này còn được gọi là huấn luyện mô hình (learning).

LINEAR REGRESSION

Bài toán hồi quy

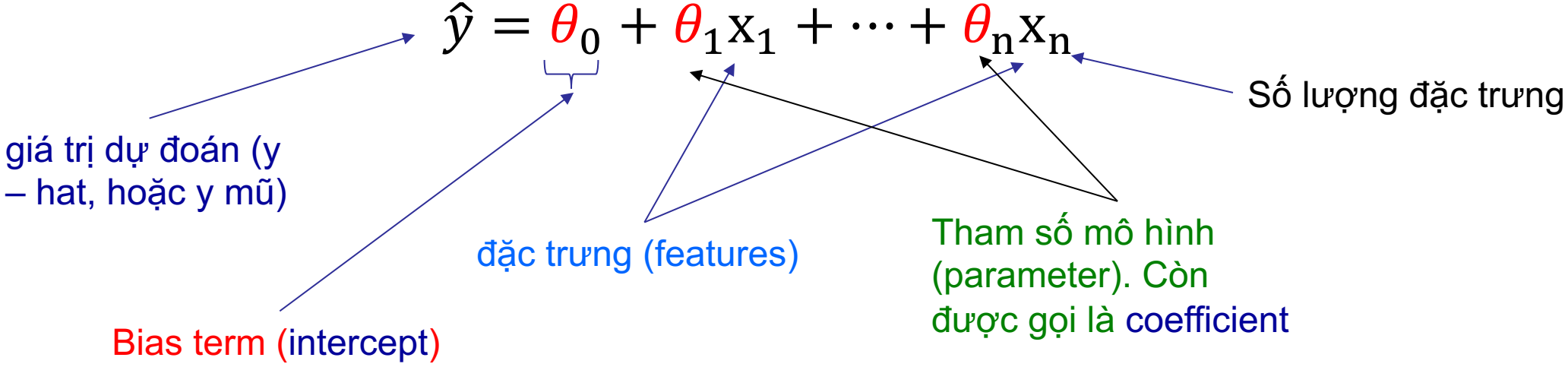
- Đầu vào: Dữ liệu đầu vào.
- Đầu ra: một giá trị số thực.

VD: Bài toán dự đoán giá nhà (Bộ dữ liệu **California Housing**)

+ Đầu vào: 1 khu vực bất kỳ và các dữ kiện khác đi kèm.
+ Đầu ra: Giá nhà trung bình.

Linear regression

- Là mô hình hồi quy tuyến tính đơn giản, thuộc dạng học máy có giám sát.
- Mô hình hồi quy tuyến tính được định nghĩa thông qua một hàm tuyến tính như sau:



California Housing

instance – feature vector

đặc trưng (features)

giá trị thực (target value)

X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 y

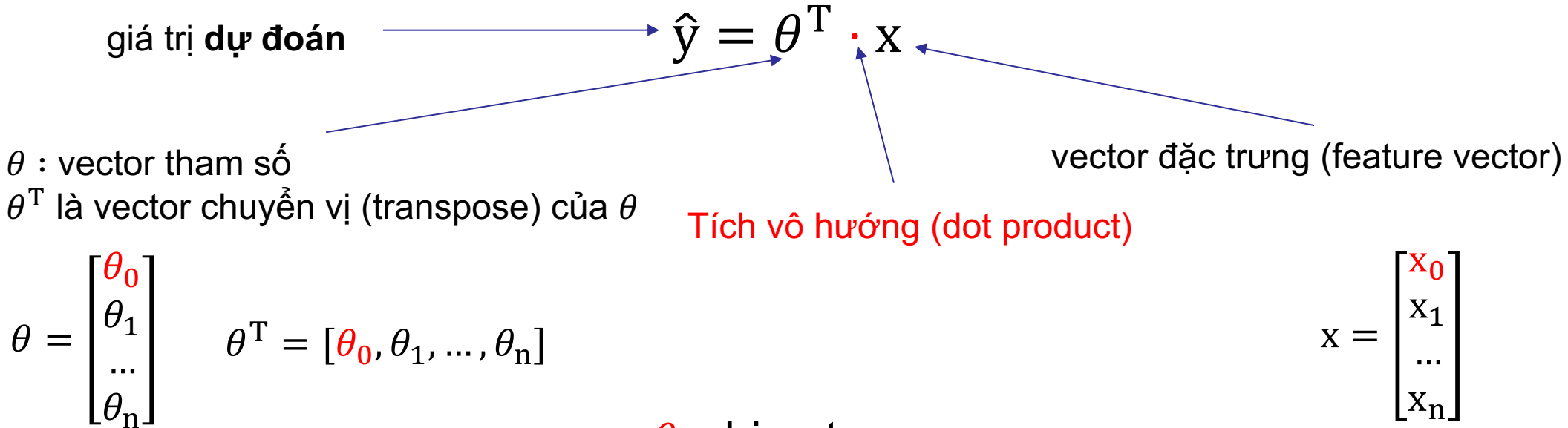
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0

Ý nghĩa từng thuộc tính

1. longitude: kinh độ.
2. latitude: vĩ độ.
3. housingMedianAge: tuổi của ngôi nhà.
4. totalRooms: Tổng số phòng.
5. totalBedrooms: Số phòng ngủ.
6. population: tổng dân số cư trú.
7. households: tổng số hộ gia đình.
8. medianIncome: Thu nhập trung bình của từng hộ gia đình.
9. medianHouseValue: Giá nhà.
10. ocean_proximity: Vị trí nhà (trong đất liền, giáp biển, giáp vịnh, trên đảo).

Linear regression dạng vector hoá

Mô hình Linear Regression được viết lại ở dạng vector hoá như sau:



θ_0: bias term

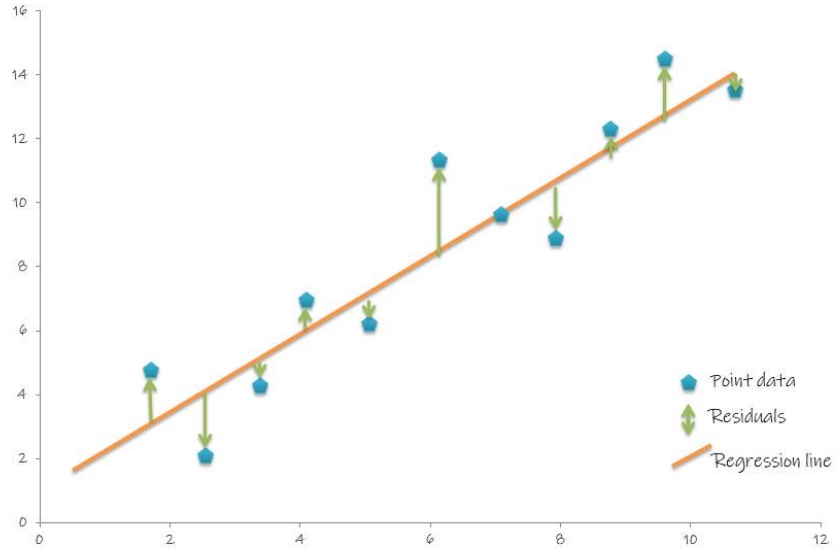
$x_0 = 1$

Độ đo dùng cho bài toán hồi quy

— Độ đo dùng cho bài toán hồi quy: Mean Square Error (MSE).

+ Sự khác biệt giữa giá trị dự đoán $\hat{y}$ và giá trị thực tế y.

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



Xây dựng hàm mất mát

– Hàm mất mát (loss function) được xây dựng dựa trên độ đo MSE như sau:

$$\mathcal{L}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot x^{(i)} - y^{(i)}\right)^2$$

Mục tiêu: $\theta^* = \operatorname{argmin}_\theta(\mathcal{L}(\theta)) = \operatorname{argmin}_\theta\left(\frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot x^{(i)} - y^{(i)}\right)^2\right)$

Ký hiệu (notation):

- $x^{(i)}$: vector đặc trưng điểm dữ liệu thứ i trong bộ dữ liệu.
- m: số lượng điểm dữ liệu.
- $y^{(i)}$: giá trị của điểm dữ liệu thứ i trong bộ dữ liệu.

Tối ưu hàm mất mát

- Hàm mất mát: $\mathcal{L}(\theta) = \frac{1}{2\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \left(\theta^{\mathrm{T}} \cdot x^{(i)} - \mathrm{y}^{(\mathrm{i})} \right)^2$
- Biểu diễn dưới dạng vector hoá (thay phép cộng bằng phép nhân các vector):

$$\mathcal{L}(\theta) = \|(X\theta - \mathrm{y})\|_2^2 = \frac{1}{2\mathrm{m}} (X\theta - \mathrm{y})^{\mathrm{T}} (X\theta - \mathrm{y})$$

Để đơn giản, lúc khai triển $\mathcal{L}(\theta)$ có thể bỏ qua phân số $\frac{1}{2\mathrm{m}}$

Tối ưu hàm mất mát

$$\mathcal{L}(\theta) = (X\theta - \mathrm{y})^{\mathrm{T}}(X\theta - \mathrm{y}) = \left((X\theta)^{\mathrm{T}} - \mathrm{y}^{\mathrm{T}}\right)(X\theta - \mathrm{y})$$

$$\mathcal{L}(\theta) = (X\theta)^{\mathrm{T}}X\theta - (X\theta)^{\mathrm{T}}\mathrm{y} - \mathrm{y}^{\mathrm{T}}X\theta + \mathrm{y}^{\mathrm{T}}\mathrm{y}$$

$$\mathcal{L}(\theta) = \mathrm{X}^{\mathrm{T}}X\theta^{\mathrm{T}}\theta - \mathrm{X}^{\mathrm{T}}\theta^{\mathrm{T}}\mathrm{y} - \mathrm{y}^{\mathrm{T}}X\theta + \mathrm{y}^{\mathrm{T}}\mathrm{y}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = 2\mathrm{X}^{\mathrm{T}}X\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y}$$

$$\nabla_{\mathbf{w}} f(\mathbf{w})^{\mathrm{T}} f(\mathbf{w}) = \nabla_{\mathbf{w}} \|f(\mathbf{w})\|_2^2 = 2\nabla_{\mathbf{w}}(f) f$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = 0 \Leftrightarrow 2\mathrm{X}^{\mathrm{T}}X\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y} = 0 \Leftrightarrow \mathrm{X}^{\mathrm{T}}X\theta = \mathrm{X}^{\mathrm{T}}\mathrm{y}$$

Normal Equation

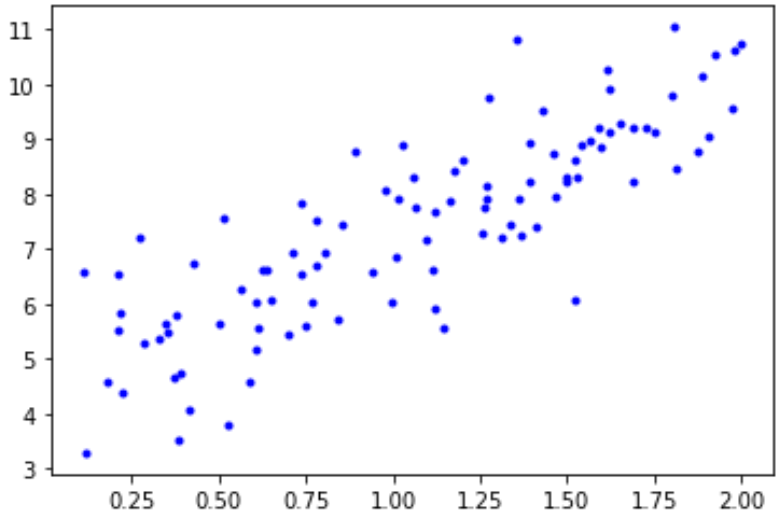
Suy ra: $\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$

Tham khảo các công thức đạo hàm: https://ccrma.stanford.edu/~dattorro/matrixcalc.pdf

Minh hoạ

– Dữ liệu: Sinh ra 100 giá trị X và y ngẫu nhiên.

```
import numpy as np
import matplotlib.pyplot as plt

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

plt.plot(X, y, "b.")
```



Minh hoạ

– Thêm vào bias term cho từng dữ liệu đầu vào X

```
X_b = np.c_[np.ones((100, 1)), X]
```

– Tính toán theta_best: $\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$

Dùng hàm *linalg.inv* để tính nghịch đảo và *np.dot* để tính tích vô hướng.

```
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

Minh hoạ

– Dùng theta_best tìm được dự đoán cho dữ liệu mới.

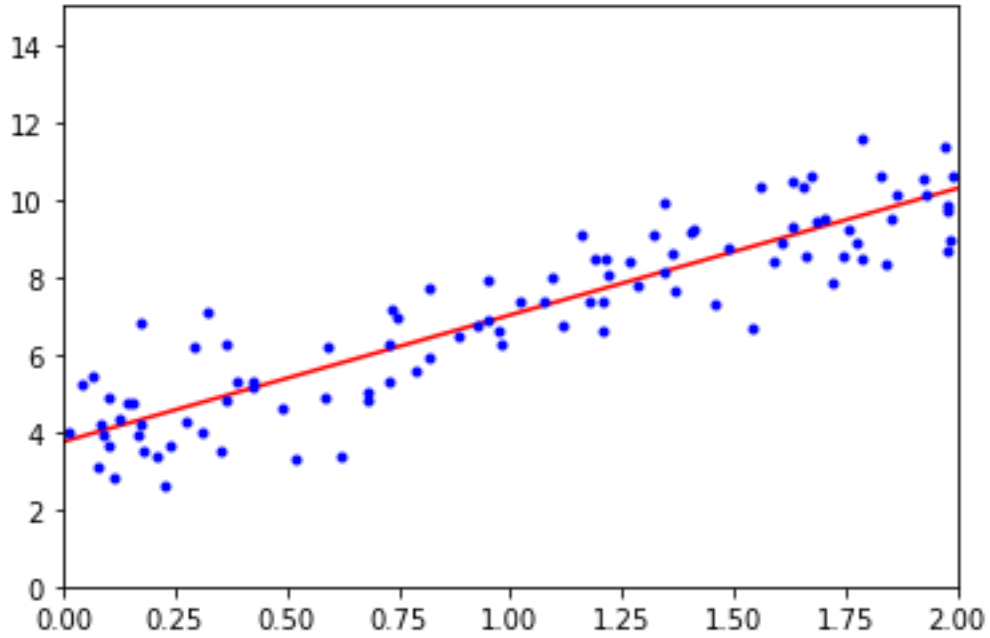
```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]
y_predict = X_new_b.dot(theta_best)
y_predict
```

Ghi chú: `X_new` là dữ liệu mới cần dự đoán

Minh hoạ

— Mô phỏng kết quả bằng đường hồi quy:

```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```



Sử dụng thư viện sklearn

```
1. from sklearn.linear_model import LinearRegression


2. lin_reg = LinearRegression()
3. lin_reg.fit(X, y)
4. lin_reg.intercept_, lin_reg.coef_
5. y_sklearn_pred = lin_reg.predict(X_new)
```

Tìm được giá trị θ_0 (bias) và tập tham số $\theta_1 \dots \theta_n$ (parameter) của mô hình.

*Dùng hàm `plot` trong `matplotlib` kiểm tra xem đường hồi quy của **y_predict** trước đó và **y_sklearn_pred** có giống nhau hay không ?*

Đánh giá mô hình hồi quy

- Tổng bình phương tất cả sai lệch giữa y_i và giá trị trung bình (Total sum squares – TSS):

$$TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

- Tổng bình phương các sai lệch giữa giá trị dự đoán của biến phụ thuộc y và giá trị trung bình (Explained sum of squares – ESS):

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

- Tổng bình phương sai số (Residual sum of squares - RSS):

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

Đánh giá mô hình hồi quy

- Ta có: **TSS = ESS + RSS**
- Hệ số phụ thuộc tuyến tính:

$$R^2 = 1 - \frac{RSS}{TSS}$$

- Hệ số R^2 càng cao ➔ mô hình càng giải thích được sự biến động của biến phụ thuộc.

VD: $R^2 = 0.85$ thì biến độc lập giải thích được 85% sự thay đổi của biến phụ thuộc, 15% còn lại là do yếu tố ngẫu nhiên.

Đánh giá mô hình hồi quy

- Khi sử dụng nhiều biến độc lập trong mô hình hồi quy thì số bậc tự do sẽ giảm (df=m-n), với *m là kích thước dữ liệu, n là số lượng hệ số mô hình* → đưa thêm bậc tự do của các tổng bình phương vào công thức hệ số xác định.
- Hệ số phụ thuộc điều chỉnh (Adjusted R-squared):

$$\overline{\mathrm{R}}^2 = 1 - \frac{\frac{RSS}{m-n}}{\frac{TSS}{(m-n)}} = \mathrm{R}^2 + (1-\mathrm{R}^2)\frac{1-\mathrm{n}}{\mathrm{m}-\mathrm{n}}$$

BÀI TẬP

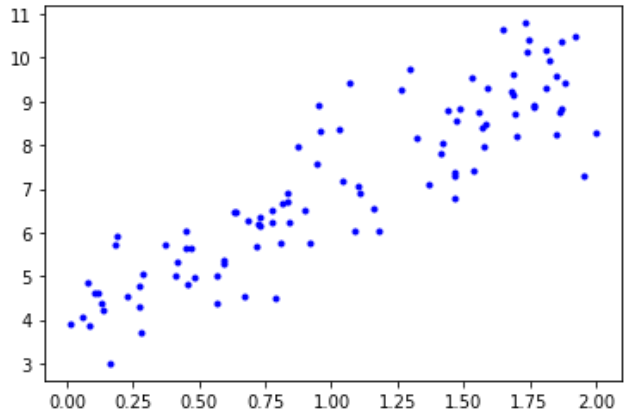
- Xây dựng mô hình hồi quy tuyến tính cho bài toán dự đoán giá nhà dựa trên bộ dữ liệu California Housing.

  Ghi chú: Chỉ cần xét thuộc totalRooms - tổng số phòng.

Hồi quy đa thức

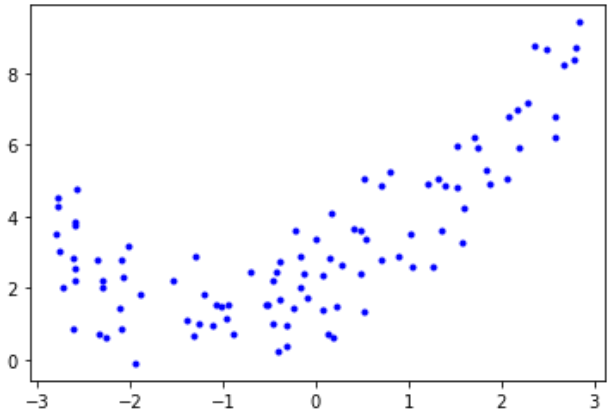
- Hồi quy đa thức (Polynomial regression) là dạng đặc biệt của hồi quy tuyến tính, khi các giá trị trong dữ liệu được phân bố theo dạng phi tuyến (non-linear).

```
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```



tuyến tính (linear)

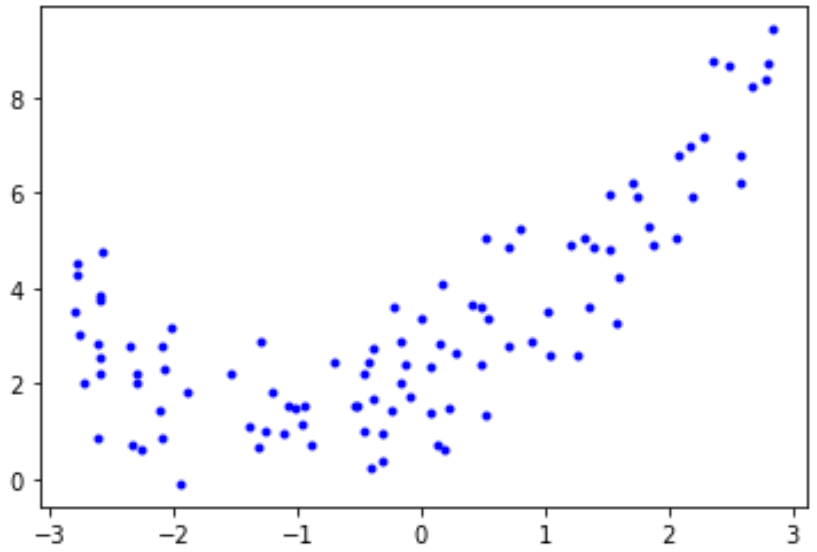
```
m=100
X = 6 * np.random.rand(m, 1) - 3
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
```



phi tuyến (non-linear)

Sinh dữ liệu huấn luyện

```
1. m=100
2. X = 6 * np.random.rand(m, 1) - 3
3. y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
```



Dữ liệu huấn luyện dạng phi tuyến

- Chuyển dữ liệu tuyến tính ban đầu sang dạng dữ liệu phi tuyến tính: lấy luỹ thừa 2 các giá trị ban đầu.
- Sử dụng lớp `PolynomialFeatures` trong sklearn.

```
1. from sklearn.preprocessing import PolynomialFeatures
2. poly_features = PolynomialFeatures(degree=2,
   include_bias=False)
3. X_poly = poly_features.fit_transform(X)
```

Dữ liệu huấn luyện gồm: (X_poly, y)

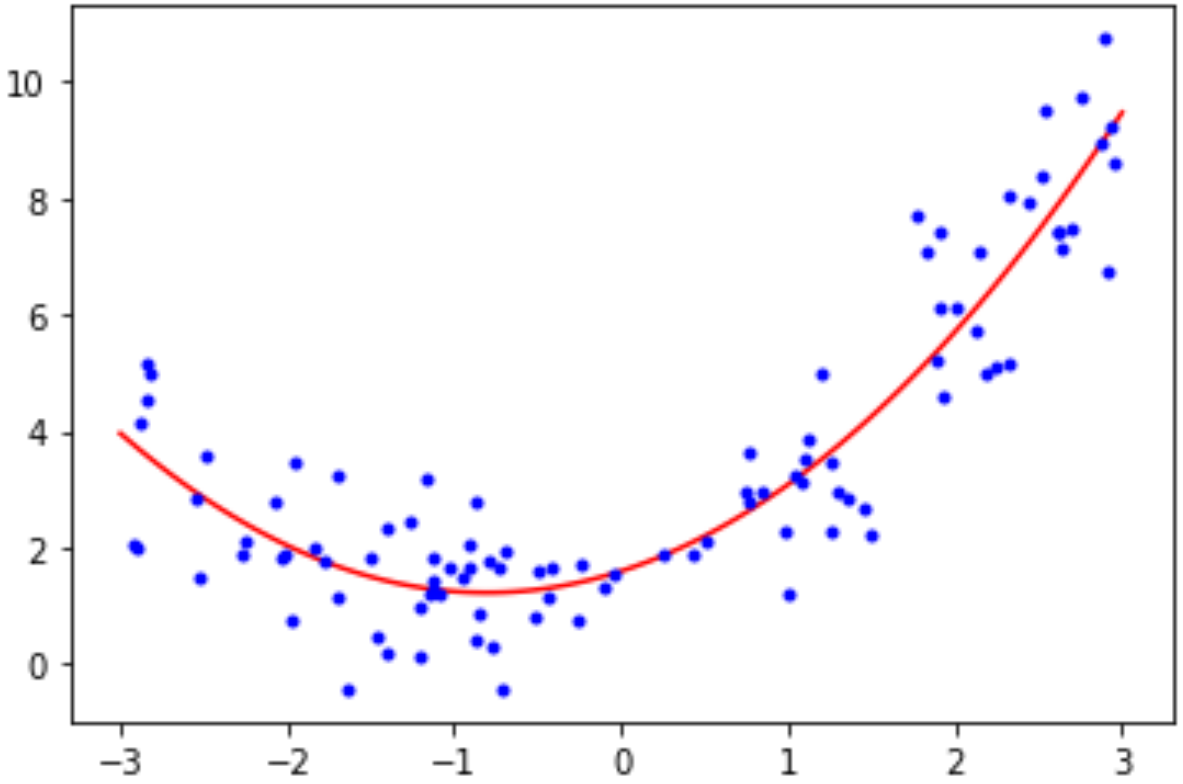
Dữ liệu dự đoán dạng phi tuyến

```
1. from sklearn.preprocessing import PolynomialFeatures


2. X_new = np.linspace(-3, 3, 100).reshape(100, 1)
3. poly_features = PolynomialFeatures(degree=2,
   include_bias=False)
4. X_poly_new = poly_features.fit_transform(X_new)
```

Huấn luyện mô hình và dự đoán

```
1. from sklearn.linear_model import LinearRegression


2. lin_reg = LinearRegression()
3. lin_reg.fit(X_poly, y)
4. lin_reg.intercept_, lin_reg.coef_
5. y_poly_pred = lin_reg.predict(X_poly_new)
```

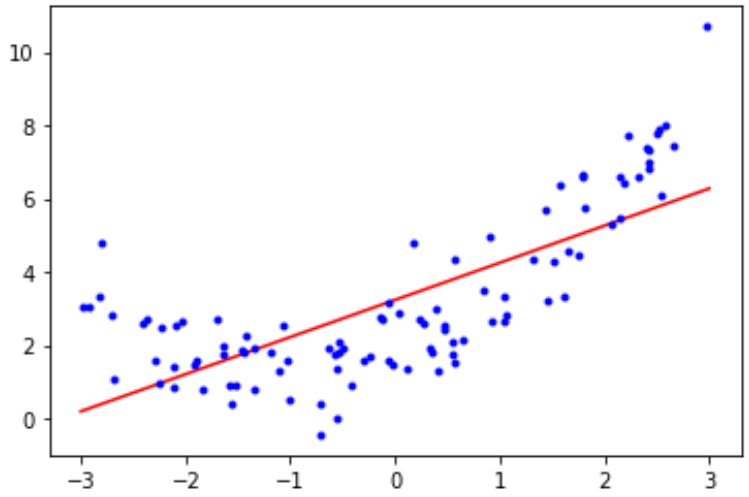
Kết quả



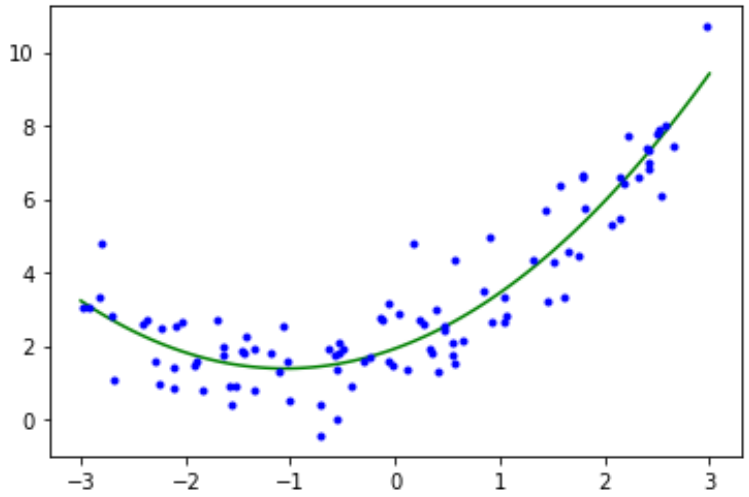
Màu xanh: dữ liệu bạn đầu
(X_poly, y)

Đường màu đỏ: dữ liệu dự đoán bởi mô hình
(X_new_poly, y_pred)

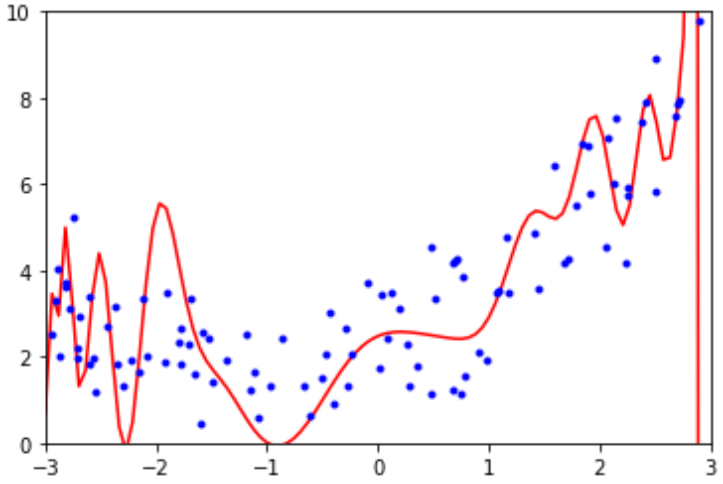
Higher degree Polynomial Regression



degree = 1



degree = 2



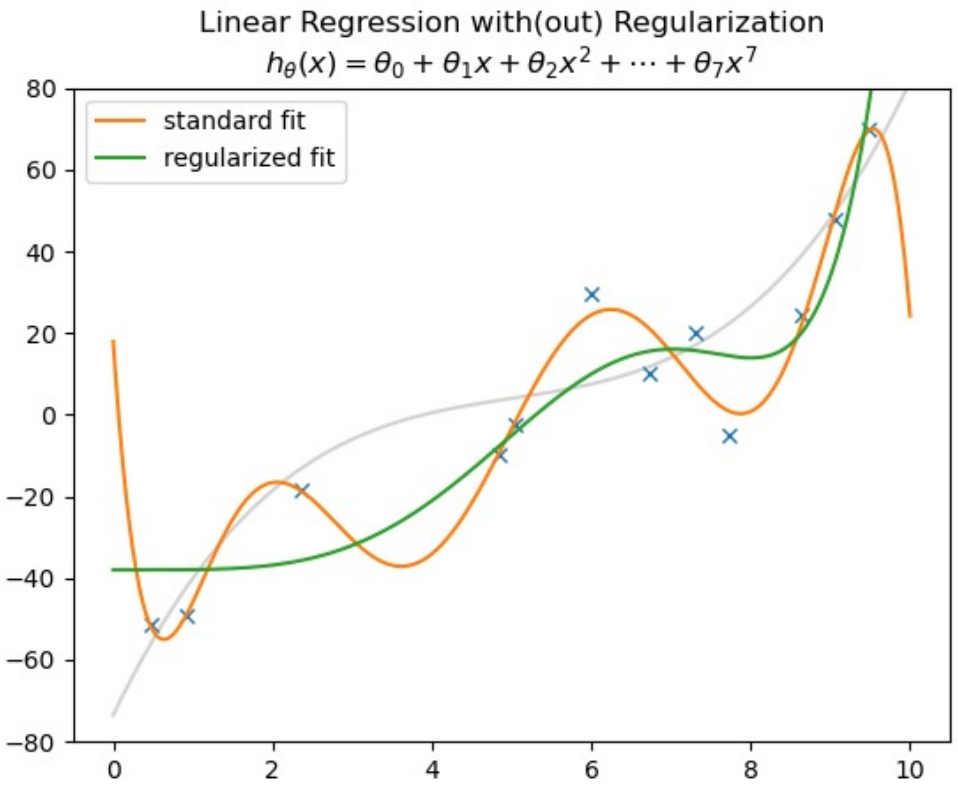
degree = 30

RIDGE REGRESSION

Regularization

- Regularization (chính quy hoá) là thủ thuật thêm vào một lượng giá trị nhằm điều chỉnh lại trọng số của mô hình với mục đích ngăn chặn *overfiting*.
- Giá trị thêm vào được gọi là thành phần điều chuẩn (regularization term).

➔ *Là một trong các thủ pháp giảm quá khớp cho mô hình !!*



Một số dạng tham số điều chuẩn

Regularization Term	Method
L2 norm: $\|\boldsymbol{\beta}\|_2$	ridge regression
L1 norm: $\|\boldsymbol{\beta}\|_1$	LASSO
Lq norm: $\|\boldsymbol{\beta}\|_q$	bridge regression
weighted L1 norm: $\sum_{j=1}^{p} w_j \left\|\beta_j\right\|$	adaptive LASSO
$c_1\|\boldsymbol{\beta}\|_1 + c_2\|\boldsymbol{\beta}\|_2^2$	elastic net
$\sum_{g=1}^{G} \sqrt{p_g}\|\beta_g\|_2$	group LASSO

Ridge regression

- Hồi quy Ridge bản chất là hồi quy tuyến tính, tuy nhiên có thêm vào thành phần điều chuẩn.
- Như vậy, hàm mất mát cho hồi quy Ridge được viết lại như sau:

$$\mathcal{L}(\theta) = \frac{1}{\mathrm{m}} \|X\theta - \mathrm{y}\|_2^2 + \alpha||\theta||_2^2$$

- Tìm tham số θ để $\mathcal{L}$ nhỏ nhất:

$$\hat{\theta} = \mathrm{argmin}_\theta \left(\frac{1}{\mathrm{m}} \|X\theta - \mathrm{y}\|_2^2 + \alpha||\theta||_2^2 \right)$$

Tìm nghiệm tối ưu

– Đạo hàm theo trọng số θ:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial(\|X\theta - \mathrm{y}\|_2^2)}{\partial \theta} + \frac{\partial(||\theta||_2^2)}{\partial \theta} = \frac{\partial((X\theta - \mathrm{y})^{\mathrm{T}}(X\theta - \mathrm{y}))}{\partial \theta} + \frac{\partial(||\theta||_2^2)}{\partial \theta}$$

$$= 2\mathrm{X}^{\mathrm{T}}X\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y} + 2\alpha\theta$$

$$= 2\mathrm{X}^{\mathrm{T}}(X\theta - \mathrm{y}) + 2\alpha\theta$$

– Áp dụng thêm tính chất $\mathrm{I}\theta=\theta$, với I là ma trận đơn vị, ta được:

$$\frac{\partial \mathcal{L}}{\partial \theta} = 2\mathrm{X}^{\mathrm{T}}X\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y} + 2\alpha\mathrm{I}\theta = 2(X^{\mathrm{T}}X + \alpha\mathrm{I})\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y}$$

Tìm nghiệm tối ưu

$$\frac{\partial \mathcal{L}}{\partial \theta} = 0 \Leftrightarrow 2(X^{\mathrm{T}}X + \alpha \mathrm{I})\theta - 2\mathrm{X}^{\mathrm{T}}\mathrm{y} = 0$$

$$\Leftrightarrow (X^{\mathrm{T}}X + \alpha \mathrm{I})\theta = \mathrm{X}^{\mathrm{T}}\mathrm{y}$$

$$\Leftrightarrow \boldsymbol{\theta} = (\boldsymbol{X^T X} + \boldsymbol{\alpha I})^{-1}\boldsymbol{X^T y}$$

(Định thức khác 0)

— Ma trận $(X^{\mathrm{T}}X + \boldsymbol{\alpha}\mathrm{I})$ sẽ không suy biến nếu $\boldsymbol{\alpha}$>0.

— Giá trị $\boldsymbol{\alpha}$ đóng vai trò như một thành phần điều chỉnh:

+ $\boldsymbol{\alpha}$ lớn: gia tăng ảnh hưởng của thành phần điều chỉnh.

+ $\boldsymbol{\alpha}$ = 0: quay về hồi quy tuyến tính ban đầu.

+ $\boldsymbol{\alpha}$ nhỏ: giảm ảnh hưởng của thành phần điều chỉnh.

Một số dạng hồi quy khác liên quan

- Hồi quy Lasso: $\mathcal{L}(\theta) = \frac{1}{\mathrm{m}} \|X\theta - \mathrm{y}\|_2^2 + \alpha \|\theta\|_1$ (L1 norm)
- Hồi quy Ridge: $\mathcal{L}(\theta) = \frac{1}{\mathrm{m}} \|X\theta - \mathrm{y}\|_2^2 + \alpha ||\theta||_2^2$ (L2 norm)
- Hồi quy ElasticNET: dạng tổng quát của Lasso và Ridge:

$$\mathcal{L}(\theta) = \frac{1}{\mathrm{m}} \|X\theta - \mathrm{y}\|_2^2 + \alpha \left[\lambda \|\theta\|_1 + \frac{1-\lambda}{2} ||\theta||_2^2\right]$$

➔ α và λ là 2 siêu tham số (hyper parameter) của mô hình hồi quy Ridge, hồi quy lasso và ElasticNet.

- Tinh chỉnh 2 siêu tham số bằng các chiến lược tinh chỉnh tham số như GridSearchCV.

TÀI LIỆU THAM KHẢO

Chương 4 của sách: *Hands-on Machine Learning with ScikitLearn, Keras & TensorFlow.*