



---

**2102470 Học máy**

**Bài giảng: Xấp xỉ hàm**

Chương 2: Xấp xỉ hàm và phân lớp

# Ôn lại bài học trước

---

- Bạn có nhớ ? % ?

# Nội dung chính

---

- 2.1 Khái niệm về xấp xỉ hàm và phân lớp.
  - 2.1.1 Xấp xỉ hàm
  - 2.1.2 Phân lớp
- 2.2 Bài toán dùng xấp xỉ hàm
  - 2.2.1 Mô tả bài toán
  - 2.2.2 Hàm mục tiêu
  - 2.2.3 Các giải thuật hồi quy
  - 2.2.4 Ví dụ về bài toán xấp xỉ hàm

## 2.2.3 Các giải thuật hồi quy

---

# Gradient descent

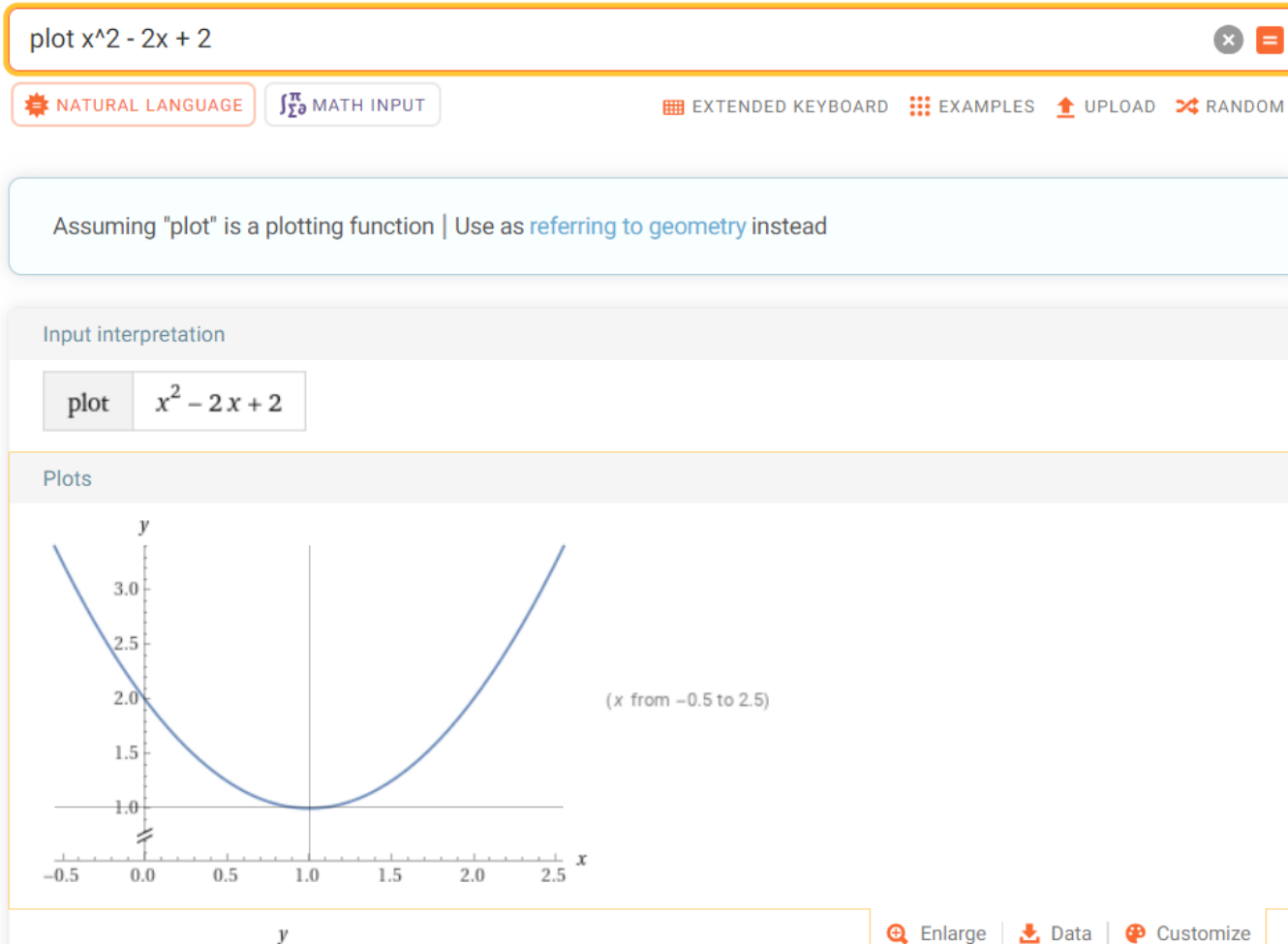
---

- Ví dụ 1: Khảo sát hàm số  $f(x) = x^2 - 2x + 2$

# Gradient descent

$$f(x) = x^2 - 2x + 2$$

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA



<https://www.wolframalpha.com/input?i=plot+x%5E2+-+2x+%2B+2>

# Gradient descent

---

$$f(x) = x^2 - 2x + 2$$

$$f'(x) = 2x - 2$$

$$x(n+1) = x(n) - \eta f'(x(n))$$

$\eta > 0$  Tốc độ học, learning rate

$x(0)$  Giá trị ban đầu



# Gradient descent

---

$$f(x) = x^2 - 2x + 2$$

$$f'(x) = 2x - 2$$

$$x(n+1) = x(n) - \eta f'(x(n))$$



$$\eta = 0.1 > 0$$

$$x(0) = 3$$

Số bước lặp?

$$|f'(x(n))| \leq \varepsilon = 10^{-3}$$

# Gradient descent

---

Sau khi tính toán hãy đưa ra nhận xét khi lựa chọn các giá trị khác nhau của:

$\eta > 0$  Tốc độ học, learning rate

$x(0)$

# Gradient descent

---

- 1 thuật toán tối ưu có khả năng dùng được cho nhiều vấn đề
- Ý tưởng: điều chỉnh các tham số (một cách lặp đi lặp lại ~ phép lặp) để giảm thiểu hàm mất mát/chi phí thông qua gradient

# Gradient descent

---

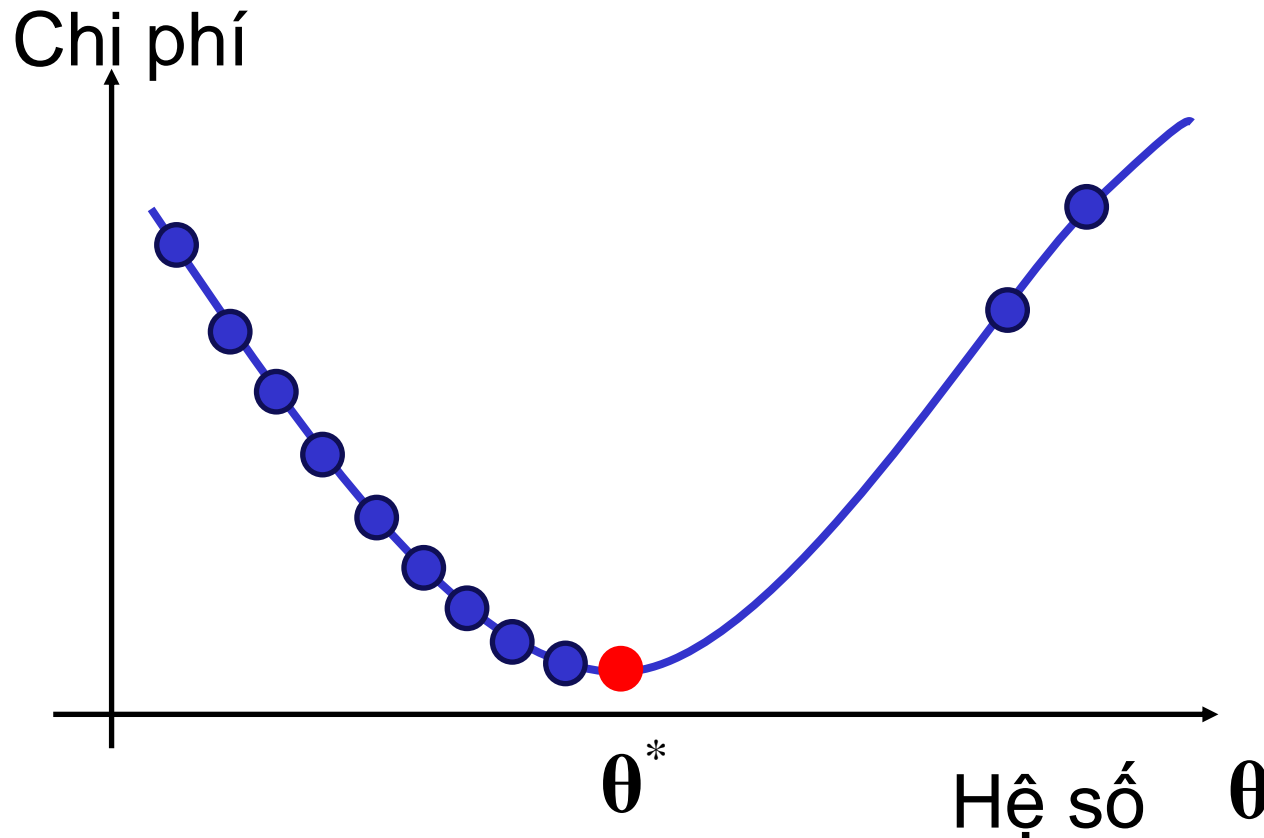


- Xác định độ dốc (cục bộ) của hàm mất mát, đối với vector hệ số  $\theta$
- Chọn đi theo hướng độ dốc giảm dần
- Đạt được mức thấp nhất khi độ dốc = 0

# Gradient descent

---

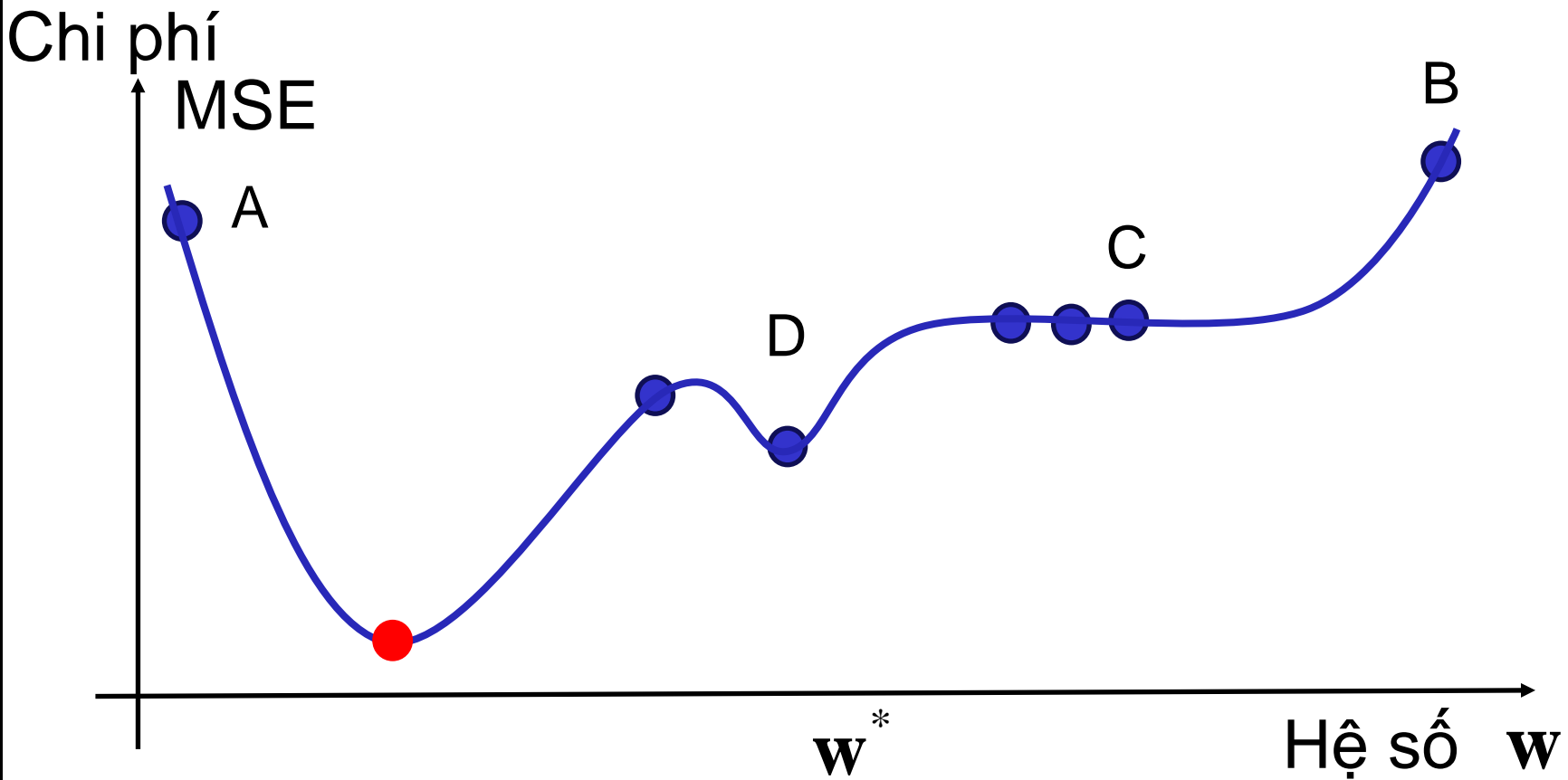
- Khởi tạo ngẫu nhiên  $\theta$
- Từng bước (nhỏ) để giảm hàm chi phí
- Lặp lại cho tới khi đạt tới giá trị tối thiểu



# Gradient descent

---

- Tốc độ hội tụ của GD phụ thuộc vào
  - + giá trị khởi tạo ban đầu
  - + learning rate



# Learning rate

---

- Tốc độ học nhỏ
  - Cần nhiều lần lặp => mất nhiều thời gian
  - Có thể không tìm được giá trị nhỏ nhất
- Tốc độ học lớn
  - Thay đổi quá nhiều sau mỗi vòng lặp
  - Có thể làm thuật toán không thể hội tụ

# Batch GD

---

- Batch GD sử dụng toàn bộ lô dữ liệu huấn luyện cho mỗi bước
- Tương đối chậm khi các bộ dữ liệu huấn luyện lớn

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} (MSE(\boldsymbol{\theta}))$$

$\eta$  : tốc độ học

- Cần tìm tốc độ học tốt: có thể dùng tìm kiếm theo lưới
  - + Giới hạn số lượng lặp
  - + Thiết lập 1 giá trị lặp lớn nhưng sẽ chú ý để dừng lại khi vector gradient trở nên quá nhỏ



# Stochastic GD

---

- SGD chọn 1 mẫu ngẫu nhiên trong tập huấn luyện tại mỗi bước và tính toán gradient dựa trên chỉ 1 mẫu đó
  - Tốc độ nhanh hơn nhiều
  - Có thể huấn luyện trên các tập dữ liệu rất lớn
  - Mỗi bước huấn luyện cũng ngẫu nhiên hơn so với Batch GD
  - Có thể giúp tránh được những tối ưu cục bộ

# Stochastic GD

---

- Khi sử dụng SGD, các tập huấn luyện phải độc lập và được phân phối giống nhau để đảm bảo về trung bình, các tham số được đưa về mức tối ưu toàn cục.
  - Cần xáo trộn các mẫu trong quá trình huấn luyện
    - ví dụ: chọn ngẫu nhiên từng mẫu hoặc xáo trộn tập huấn luyện ở đầu mỗi **epoch**\*

\*“**epoch**“: đề cập đến một lần hoàn thành, khi mô hình được huấn luyện từ toàn bộ tập dữ liệu huấn luyện nhằm tinh chỉnh các tham số của mô hình

Trong quá trình huấn luyện, các tham số của mô hình được điều chỉnh để giảm thiểu hàm mất mát bằng cách lặp lại trên tập dữ liệu nhiều lần.

## Mini-batch GD

---

- Mini-batch: lô nhỏ ngẫu nhiên các mẫu ( $k$  mẫu  $\sim$  batch size,  $1 < k \ll N$ )
- Mini-batch GD tính toán gradient trên các mini-batch
- Ưu điểm so với SGD
  - Có thể tăng hiệu suất thông qua việc tối ưu hóa phần cứng (ví dụ khi sử dụng GPU) khi thực hiện các tính toán ma trận

# Logistic Regression

---

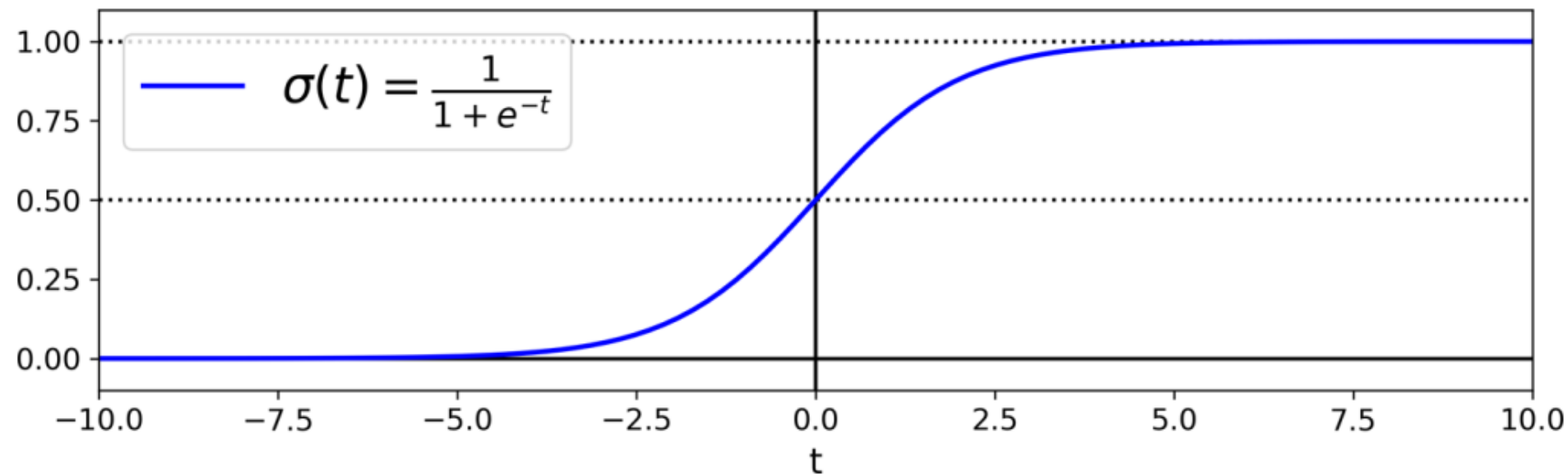
- Chú ý
  - Một số thuật toán hồi quy có thể dùng cho phân loại (và ngược lại)
- Logistic Regression
  - Thường được dùng để ước lượng xác suất một mẫu thuộc về một lớp cụ thể
  - Ví dụ: xác suất bức thư là thư rác là bao nhiêu?
    - Nếu xác suất ước tính  $\geq 50\%$   $\Rightarrow$  dự đoán rằng bức thư thuộc về lớp thư rác (positive class, được gán nhãn "1"),
    - Nếu không, dự đoán bức thư thuộc về lớp thư thường (negative class, được gán nhãn "0").
    - Bộ phân loại nhị phân

# Logistic Regression

---

- Nhắc lại về hàm logistic

$$\sigma(t) = \frac{1}{1 + \exp(-t)} = \frac{1}{1 + e^{-t}}$$



# Logistic Regression

---

- Ước lượng xác suất  $\hat{p} = \sigma(\boldsymbol{\theta}^T \mathbf{x})$

$$\hat{y} = \begin{cases} 0 & \hat{p} < 0.5 \\ 1 & \hat{p} \geq 0.5 \end{cases}$$

Hàm mục tiêu

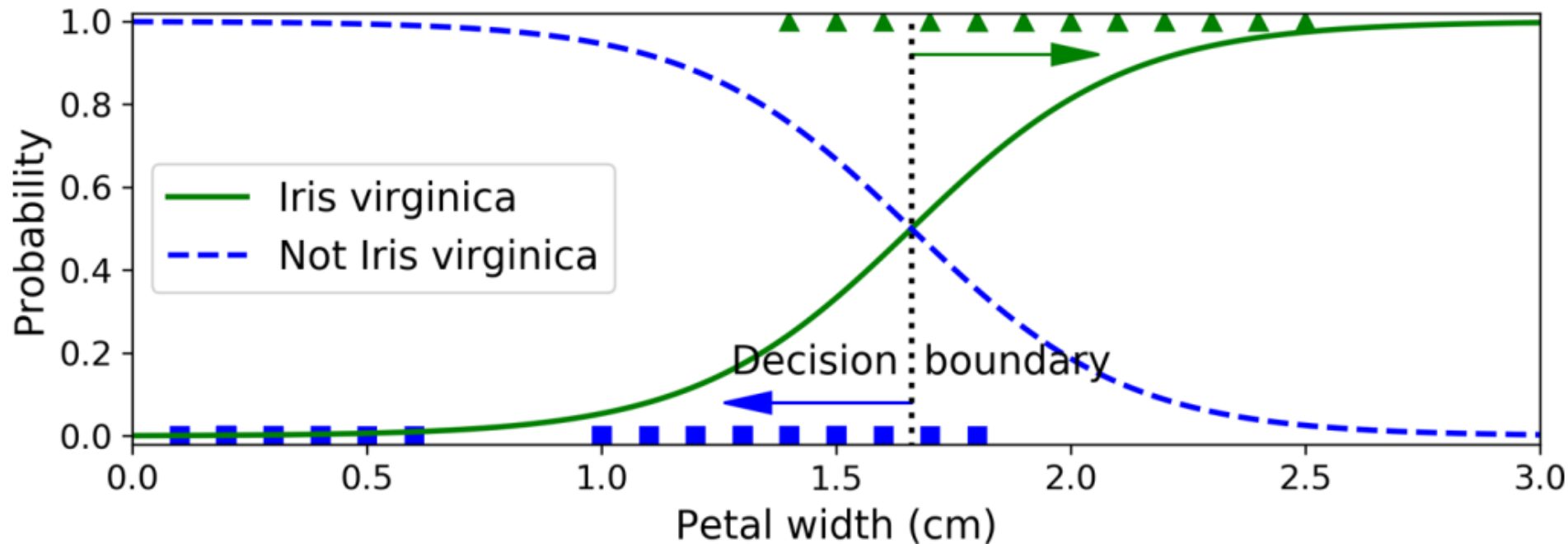
$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & y = 1 \\ -\log(1 - \hat{p}) & y = 0 \end{cases}$$

$$c(\boldsymbol{\theta}) = -y \log(\hat{p}) - (1 - y) \log(1 - \hat{p})$$

$$J(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{i=1}^M \left[ y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i) \right]$$

# Logistic Regression

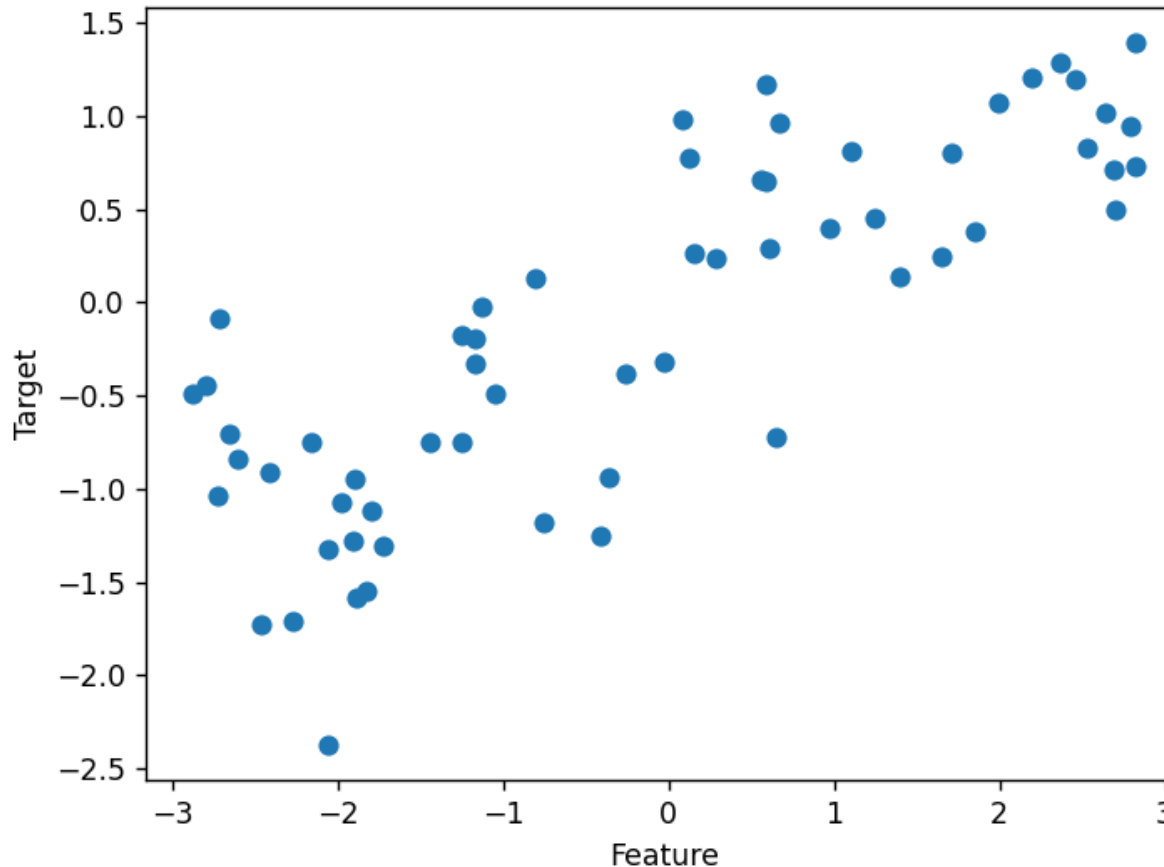
---



## 2.2.4 Ví dụ về bài toán xấp xỉ hàm

---

- Ví dụ 1: sử dụng tập dữ liệu wave [TLHT1]





# Ví dụ 1

```
▶ ~  
#from sklearn.preprocessing import add_dummy_feature  
from sklearn.linear_model import LinearRegression  
from sklearn.model_selection import train_test_split  
  
X, y = mglearn.datasets.make_wave(n_samples=100)  
# split the dataset into: training set and testset  
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)  
# training  
LR_model = LinearRegression().fit(X_train, y_train)  
  
print("Parameters of the linear regression model after training:")  
print(f"theta_0 = {LR_model.intercept_:.3f}")  
print(f"theta_1 = {LR_model.coef_[0]:.3f}")  
  
print("\nChecking the scores:")  
print(f"Training set score = {LR_model.score(X_train, y_train):.3f}")  
print(f"Test set score = {LR_model.score(X_test, y_test):.3f}")
```

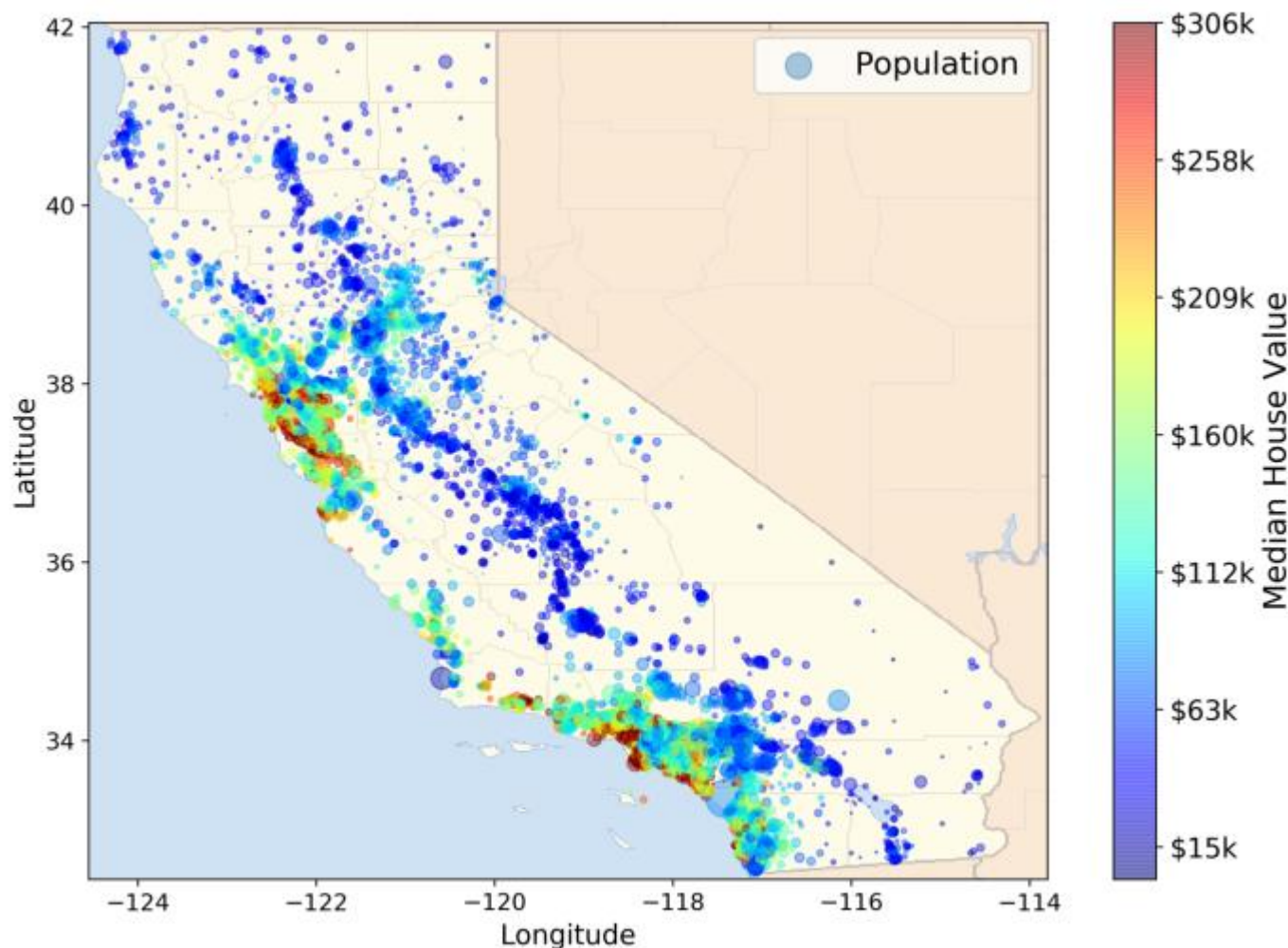
[19]

✓ 0.9s

```
Parameters of the linear regression model after training:  
theta_0 = -0.023  
theta_1 = 0.404  
  
Checking the scores:  
Training set score = 0.593  
Test set score = 0.593
```

## Ví dụ 2

- Ví dụ 2: California Housing Prices dataset [TLHT2]

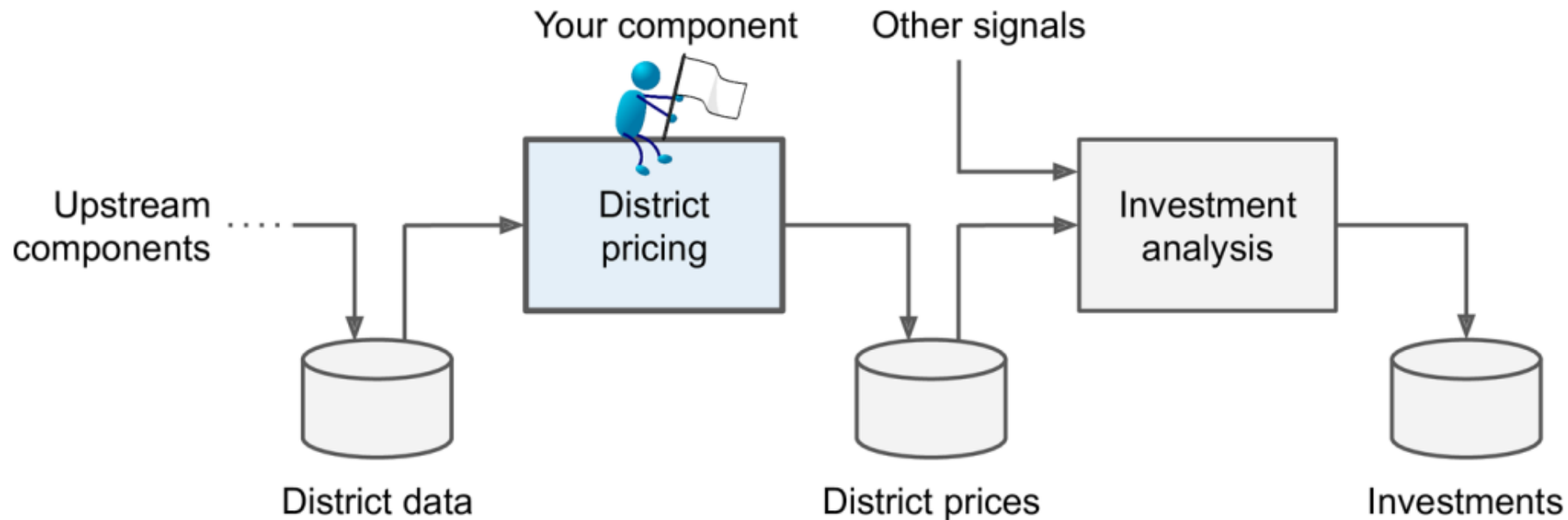


[https://github.com/ageron/handson-ml3/blob/main/02\\_end\\_to\\_end\\_machine\\_learning\\_project.ipynb](https://github.com/ageron/handson-ml3/blob/main/02_end_to_end_machine_learning_project.ipynb)

## Ví dụ 2

---


- B1: Nhận định vấn đề và nắm được bức tranh chung



ML pipeline

# Ví dụ 2

- B2: Thu thập dữ liệu
  - Download dữ liệu

 housing.csv

19.02.2022 13:41

Excel.CSV

1.391 KB


```
1 longitude,latitude,housing_median_age,total_rooms,total_bedrooms,population,households,
2 -122.23,37.88,41.0,880.0,129.0,322.0,126.0,8.3252,452600.0,NEAR BAY
3 -122.22,37.86,21.0,7099.0,1106.0,2401.0,1138.0,8.3014,358500.0,NEAR BAY
4 -122.24,37.85,52.0,1467.0,190.0,496.0,177.0,7.2574,352100.0,NEAR BAY
5 -122.25,37.85,52.0,1274.0,235.0,558.0,219.0,5.6431,341300.0,NEAR BAY
6 -122.25,37.85,52.0,1627.0,280.0,565.0,259.0,3.8462,342200.0,NEAR BAY
7 -122.25,37.85,52.0,919.0,213.0,413.0,193.0,4.0368,269700.0,NEAR BAY
8 -122.25,37.84,52.0,2535.0,489.0,1094.0,514.0,3.6591,299200.0,NEAR BAY
9 -122.25,37.84,52.0,3104.0,687.0,1157.0,647.0,3.12,241400.0,NEAR BAY
10 -122.26,37.84,42.0,2555.0,665.0,1206.0,595.0,2.0804,226700.0,NEAR BAY
11 -122.25,37.84,52.0,3549.0,707.0,1551.0,714.0,3.6912,261100.0,NEAR BAY
12 -122.26,37.85,52.0,2202.0,434.0,910.0,402.0,3.2031,281500.0,NEAR BAY
13 -122.26,37.85,52.0,3503.0,752.0,1504.0,734.0,3.2705,241800.0,NEAR BAY
14 -122.26,37.85,52.0,2491.0,474.0,1098.0,468.0,3.075,213500.0,NEAR BAY
15 -122.26,37.84,52.0,696.0,191.0,345.0,174.0,2.6736,191300.0,NEAR BAY
16 -122.26,37.85,52.0,2643.0,626.0,1212.0,620.0,1.9167,159200.0,NEAR BAY
17 -122.26,37.85,50.0,1120.0,283.0,697.0,264.0,2.125,140000.0,NEAR BAY
18 -122.27,37.85,52.0,1966.0,347.0,793.0,331.0,2.775,152500.0,NEAR BAY
19 -122.27,37.85,52.0,1228.0,293.0,648.0,303.0,2.1202,155500.0,NEAR BAY
20 -122.26,37.84,50.0,2239.0,455.0,990.0,419.0,1.9911,158700.0,NEAR BAY
21 -122.27,37.84,52.0,1503.0,298.0,690.0,275.0,2.6033,162900.0,NEAR BAY
22 -122.27,37.85,40.0,751.0,184.0,409.0,166.0,1.3578,147500.0,NEAR BAY
23 -122.27,37.85,42.0,1639.0,367.0,929.0,366.0,1.7135,159800.0,NEAR BAY
24 -122.27,37.84,52.0,2436.0,541.0,1015.0,478.0,1.725,113900.0,NEAR BAY
25 -122.27,37.84,52.0,1688.0,337.0,853.0,325.0,2.1806,99700.0,NEAR BAY
26 -122.27,37.84,52.0,2224.0,437.0,1006.0,422.0,2.6,132600.0,NEAR BAY
27 -122.28,37.85,41.0,535.0,123.0,317.0,119.0,2.4038,107500.0,NEAR BAY
28 -122.28,37.85,49.0,1130.0,244.0,607.0,239.0,2.4597,93800.0,NEAR BAY
29 -122.28,37.85,52.0,1898.0,421.0,1102.0,397.0,1.808,105500.0,NEAR BAY
30 -122.28,37.85,52.0,2888.0,588.0,1431.0,478.0,1.6161,100000.0,NEAR BAY
```

Normal text file length : 1.423.529 lines : 20.642 Ln : 2

## Ví dụ 2

---

- B2: Thu thập dữ liệu

 housing.csv

19.02.2022 13:41

Excel.CSV

1.391 KB

longitude, latitude,  
housing\_median\_age,  
total\_rooms, total\_bedrooms,  
population, households,  
median\_income, median\_house\_value,  
ocean\_proximity

## Ví dụ 2

- B3: Khám phá và trực quan hóa dữ liệu

```
In [5]: housing = load_housing_data()  
housing.head()
```

```
Out[5]:
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

```
>>> housing["ocean_proximity"].value_counts()  
<1H OCEAN      9136  
INLAND          6551  
NEAR OCEAN      2658  
NEAR BAY        2290  
ISLAND           5  
Name: ocean_proximity, dtype: int64
```

## Ví dụ 2

---

- B3: Khám phá và trực quan hóa dữ liệu

```
In [6]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude                20640 non-null float64
latitude                 20640 non-null float64
housing_median_age       20640 non-null float64
total_rooms              20640 non-null float64
total_bedrooms           20433 non-null float64
population               20640 non-null float64
households               20640 non-null float64
median_income            20640 non-null float64
median_house_value       20640 non-null float64
ocean_proximity          20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```



## Ví dụ 2

---

- B3: Khám phá và trực quan hóa dữ liệu

In [8]: `housing.describe()`

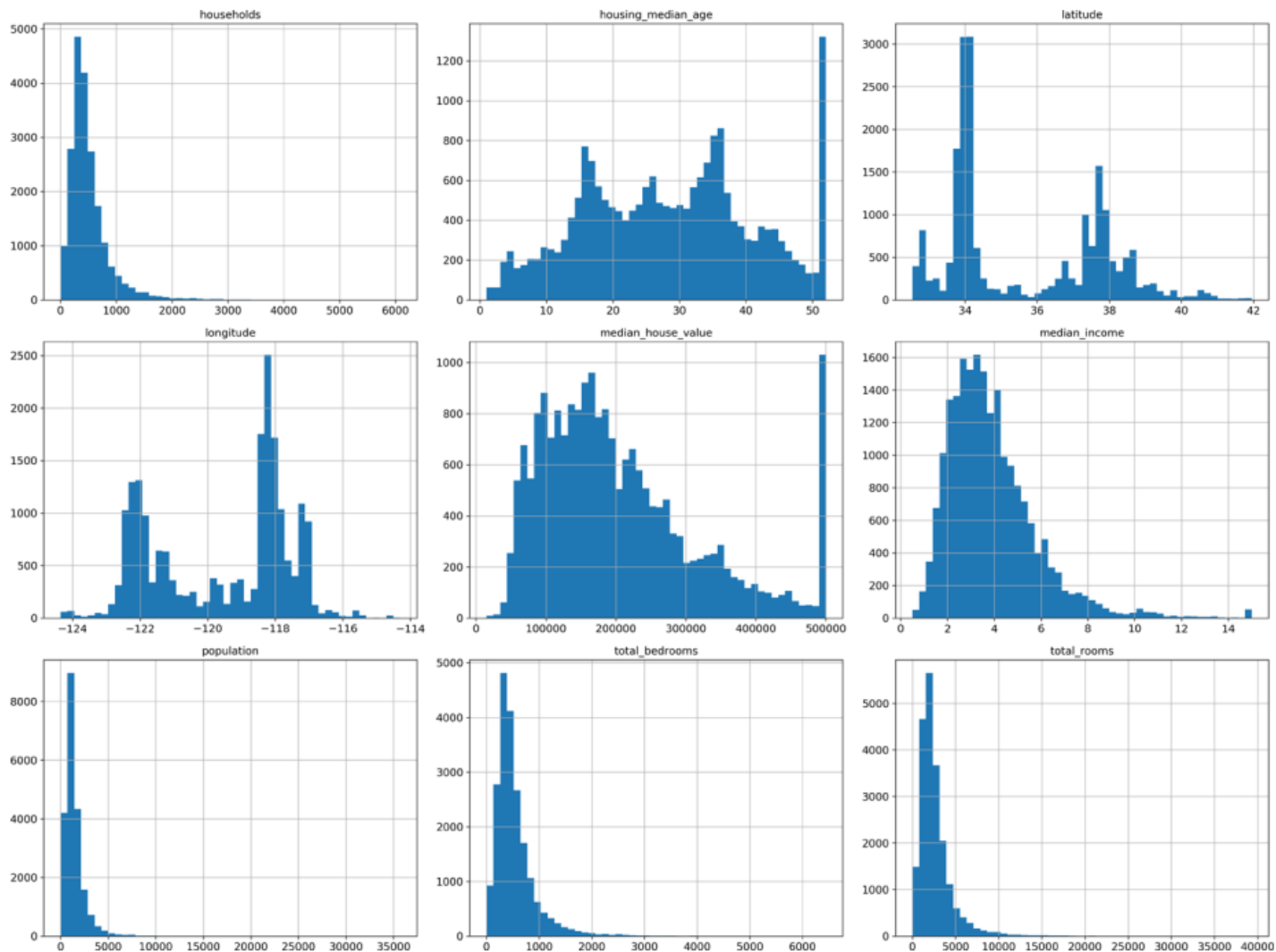
Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553
std	2.003532	2.135952	12.585558	2181.615252	421.385070
min	-124.350000	32.540000	1.000000	2.000000	1.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000



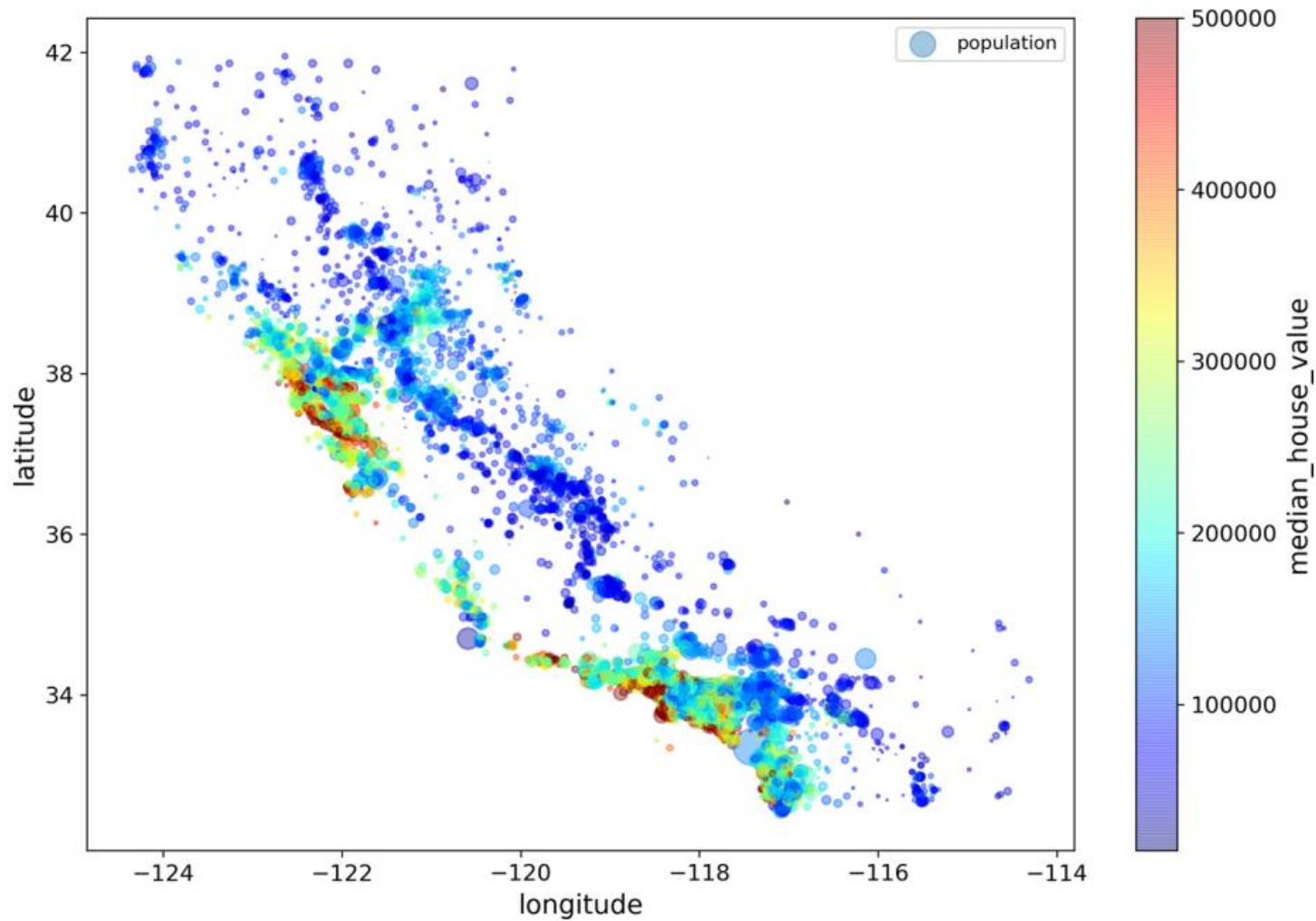
# Ví dụ 2

- B3: Khám phá và trực quan hóa dữ liệu



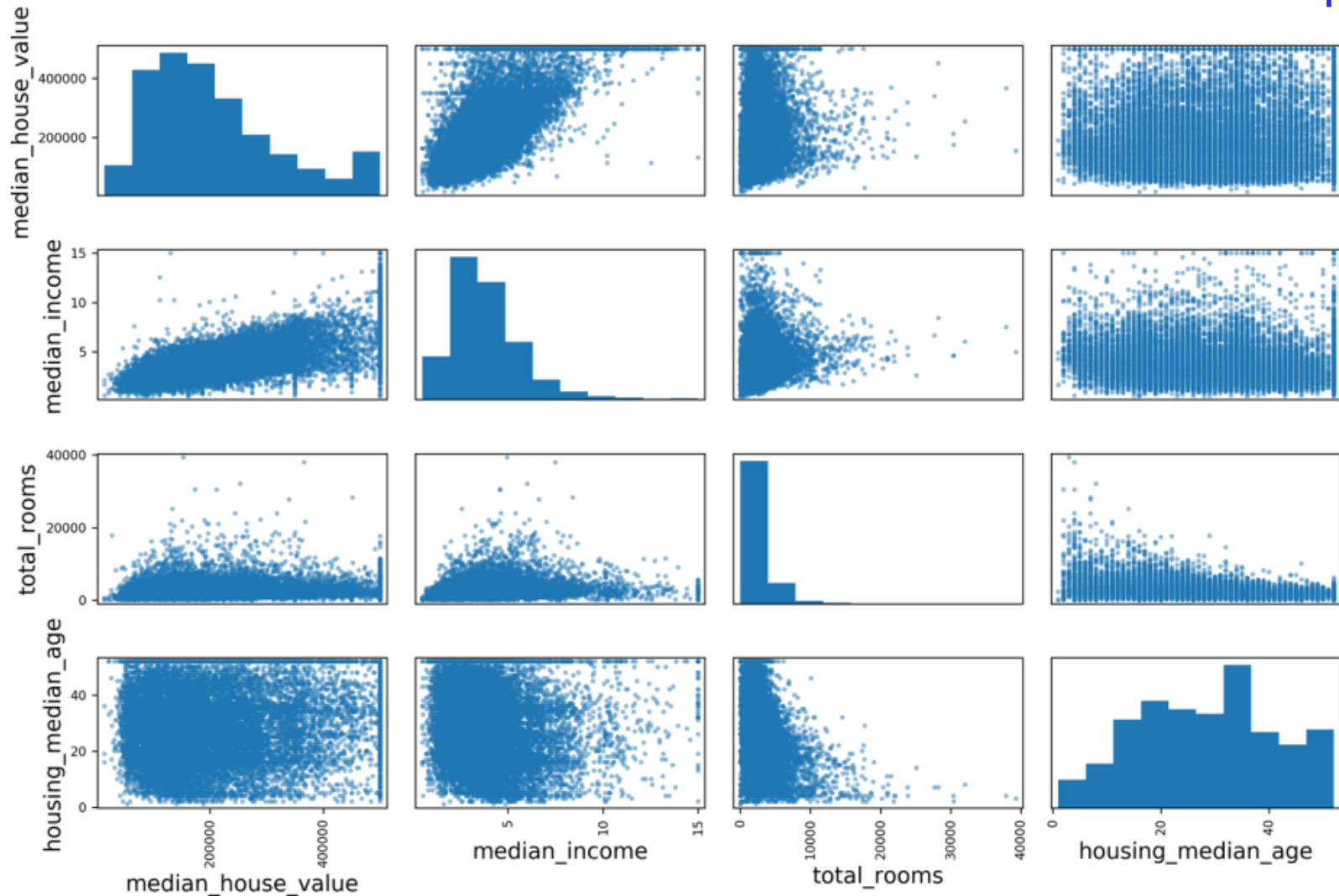
## Ví dụ 2

- B3: Khám phá và trực quan hóa dữ liệu



## Ví dụ 2

- B3: Khám phá và trực quan hóa dữ liệu **scatter matrix plot**



## Ví dụ 2

---

- B3: Khám phá và trực quan hóa dữ liệu

correlation matrix

```
>>> corr_matrix = housing.corr()
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value      1.000000
median_income           0.687160
rooms_per_household     0.146285
total_rooms             0.135097
housing_median_age      0.114110
households              0.064506
total_bedrooms          0.047689
population_per_household -0.021985
population              -0.026920
longitude               -0.047432
latitude                -0.142724
bedrooms_per_room       -0.259984
Name: median_house_value, dtype: float64
```

## Ví dụ 2

---

- B4: Chuẩn bị dữ liệu cho các thuật toán ML

### Data cleaning

```
housing.dropna(subset=["total_bedrooms"])    # option 1
housing.drop("total_bedrooms", axis=1)       # option 2
median = housing["total_bedrooms"].median()  # option 3
housing["total_bedrooms"].fillna(median, inplace=True)
```

## Ví dụ 2

---

- B4: Chuẩn bị dữ liệu cho các thuật toán ML

### Handling text and categorical attributes

```
>>> housing_cat = housing[["ocean_proximity"]]
```

```
>>> housing_cat.head(10)
```

	ocean_proximity
17606	<1H OCEAN
18632	<1H OCEAN
14650	NEAR OCEAN
3230	INLAND
3555	<1H OCEAN
19480	INLAND
8879	<1H OCEAN
13685	INLAND
4937	<1H OCEAN
4861	<1H OCEAN

```
>>> from sklearn.preprocessing import OneHotEncoder
```

```
>>> cat_encoder = OneHotEncoder()
```

```
>>> housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
```

```
>>> housing_cat_1hot
```

```
<16512x5 sparse matrix of type '<class 'numpy.float64'>'  
with 16512 stored elements in Compressed Sparse Row format>
```

## Ví dụ 2

---

- B5: Lựa chọn mô hình và huấn luyện

```
from sklearn.linear_model import LinearRegression
```

```
lin_reg = LinearRegression()  
lin_reg.fit(housing_prepared, housing_labels)
```

```
>>> some_data = housing.iloc[:5]  
>>> some_labels = housing_labels.iloc[:5]  
>>> some_data_prepared = full_pipeline.transform(some_data)  
>>> print("Predictions:", lin_reg.predict(some_data_prepared))  
Predictions: [ 210644.6045  317768.8069  210956.4333  59218.9888  189747.5584]  
>>> print("Labels:", list(some_labels))  
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

## Ví dụ 2

---

- B6: Tinh chỉnh mô hình

### Grid search

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```



## Ví dụ 2

---

- B7: Trình bày giải pháp
- B8: Phát hành, giám sát và duy trì hệ thống

# Tổng kết

---

- Hiểu và thực hiện được giải thuật gradient descent
- Nắm được cách thực hiện bài toán xấp xỉ hàm trong thực tế

# Hoạt động sau buổi học

---

- Làm BTVN
- Ôn lại xấp xỉ hàm

# Chuẩn bị cho buổi học tiếp theo

---

- Đọc, tìm hiểu thông qua các tài liệu về chủ đề phân lớp (classification)
- Tìm hiểu về K-nearest neighbor

# Tài liệu tham khảo

---

- <https://www.wolframalpha.com/input?i=plot+x%5E2+-+2x+%2B+2>
- <https://www.ibm.com/topics/gradient-descent>
- <https://www.benfrederickson.com/numerical-optimization/>