Chương 4:
K-Nearest Neighbors Algorithm

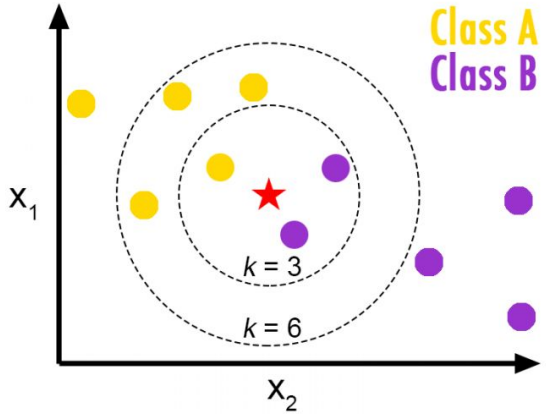
TS. Phạm Tuấn

ptuan@ute.udn.vn

Giới thiệu

Thuật toán Nearest neighbor là một trong những thuật toán học máy **có giám sát** "đơn giản nhất" và đã được nghiên cứu kỹ trong lĩnh vực nhận dạng mẫu trong thế kỷ trước.

Mặc dù các thuật toán Nearest neighbor không còn phổ biến như trước đây nhưng chúng vẫn được sử dụng rộng rãi trong thực tế bởi vì độ đơn giản cũng như hiệu suất của thuật toán Nearest neighbor.



Ý tưởng

Trong khi *k*-NN là một hàm xấp xỉ phổ quát trong những điều kiện nhất định, và ý tưởng đằng sau của *k*-NN là tương đối đơn giản. *k*-NN là một thuật toán cho việc học có giám sát, đơn giản là lưu các mẫu dữ liệu đã được huấn luyện.

$$\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n),$$

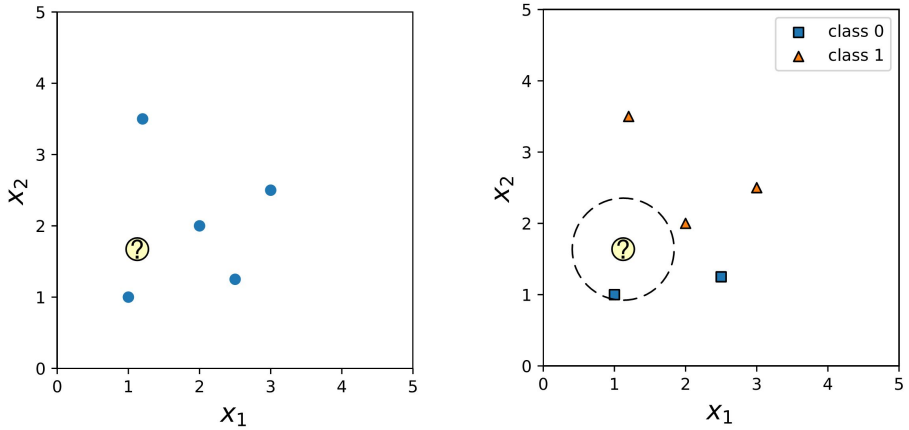
Bởi vì thuật toán này không cần huấn luyện nên *k*-NN được gọi là thuật toán lười.

Lười ở đây nghĩa là *k*-NN không xử lý gì ở giai đoạn training mà chỉ lưu trữ những dữ liệu đã training từ trước. *k*-NN sẽ tập trung xử lý ở giai đoạn dự đoán.

Ý tưởng

Để đưa ra dự đoán (nhãn lớp hoặc giá trị liên tục), các thuật toán *k*-NN tìm **k** lân cận gần nhất của một điểm đầu vào và tính toán nhãn (phân loại) hoặc giá trị liên tục (hồi quy) dựa trên **k** điểm gần nhất (tương tự).

Cơ chế chính xác sẽ được giải thích trong các phần tiếp theo. Tuy nhiên, ý tưởng tổng quát là thay vì xấp xỉ hàm mục tiêu y = f(x) trên toàn cục, trong mỗi lần dự đoán, *k*-NN xấp xỉ hàm mục tiêu cục bộ.



Thuật toán *k*-NN

Slide này giới thiệu về thuật toán với 1-nearest neighbor:

Huấn luyện: Lưu các mẫu dữ liệu đã training(x[i], *f*(x[i])).

Dự đoán:

- closest point := None
- closest distance := ∞
- for i = 1, ..., n:
  - current distance := d(x[i], x[q])
  - if current distance < closest distance:
    - closest distance := current distance
    - closest point := x[i]

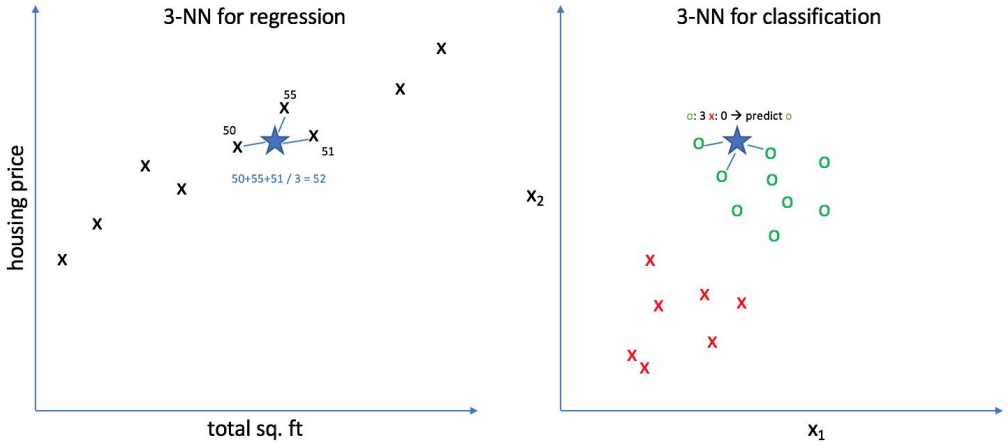
q = query point

prediction *h*(x[q]) is the target value of closest point

k-NN cho bài toán phân loại và hồi quy

Ở slide trước *k*-NN đưa ra dự đoán bằng việc gán nhãn hoặc giá trị nhất định dựa trên những điểm gần với điểm đầu vào nhất. Khoảng cách gần xa được dựa trên Euclidean distance.

k-NN xem xét **k điểm gần nhất** với điểm đầu vào để đưa ra các giá trị nhãn hoặc giá trị liên tục.



k-NN cho bài toán phân loại

k-NN cho bài toán phân loại hoạt động dựa trên số lượng voting nhiều nhất k điểm gần nhất cho bài toán phân loại.

Giả sử chúng ta có {1, . . . , t } nhãn dữ liệu.

$$h(\mathbf{x}^{[q]}) = \arg \max_{y \in \{1,\ldots,t\}} \sum_{i=1}^{k} \delta(y, f(\mathbf{x}^{[i]})).$$

Ở đây, δ định nghĩa là Kronecker Delta function:

$$\delta(a, b) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{if } a \neq b. \end{cases}$$

k-NN cho bài toán hồi quy

Ý tưởng của *k*-NN cho hồi quy khá giống với phân loại.

1. Đầu tiên, chúng ta tìm k điểm gần nhất của điểm đầu vào.
2. Sau đó, chúng ta đưa ra dự đoán dựa vào giá trị nhãn của k điểm gần nhất

Thông thường giá trị dự đoán cho điểm đầu vào thường được tính bằng cách lấy trung bình giá trị của k điểm gần nhất

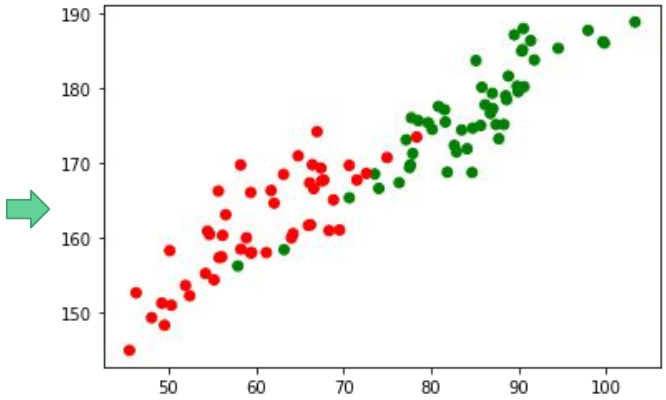
$$h\left(\mathbf{x}^{[t]}\right) = \frac{1}{k}\sum_{i=1}^{k} f\left(\mathbf{x}^{[i]}\right). \qquad f : \mathbb{R}^{>} \rightarrow \mathbb{R}.$$

Ví dụ Python

Trong ví dụ này, chúng ta sẽ dự đoán giới tính dựa vào chiều cao và cân nặng.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('weight-height.csv')
data['Weight'] = data['Weight']*0.45
data['Height'] = data['Height']*2.54
data.head()

colors = {'Male':'g', 'Female':'r'}
plt.scatter(data['Weight'], data['Height'], c = data['Gender'].apply(lambda x:colors[x]))
```



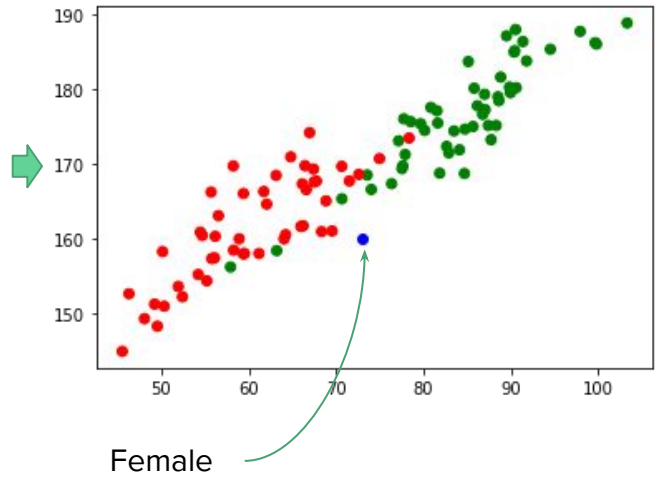
Ví dụ Python

Xây dựng mô hình *k*-NN đối với tập dữ liệu, với k=5.

```
from sklearn.neighbors import KNeighborsClassifier
number_neighbor = 5
neigh = KNeighborsClassifier(n_neighbors=number_neighbor)
neigh.fit(data[['Weight','Height']], data['Gender'])
```

Nhập chiều cao và cân nặng và dự đoán giới tính.

```
height = 160
weight = 73
plt.scatter(data['Weight'], data['Height'], c = data['Gender'].apply(lambda x:colors[x]))
plt.scatter(weight, height, c = 'b')
plt.show()
gender_predict = neigh.predict([[weight,height]])
print(gender_predict)
```



Ví dụ Python

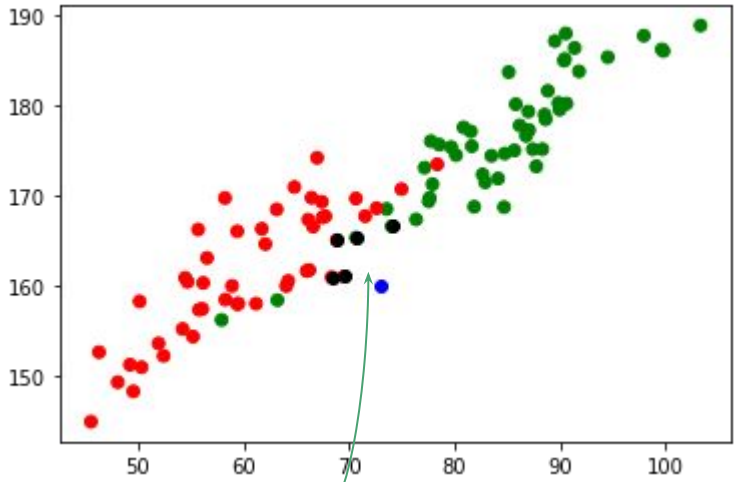
Lấy index của k điểm gần nhất.

```
neighbors_index = neigh.kneighbors([[weight,height]], number_neighbor,
return_distance=False)[0]
neighbors = data.iloc[neighbors_index]
neighbors
```

	Gender	Height	Weight
79	Female	160.949469	69.555092
63	Female	160.870319	68.351228
13	Male	165.255550	70.663676
66	Female	164.975827	68.849772
32	Male	166.506644	74.032043

Vẽ k điểm gần nhất lên trục tọa độ.

```
plt.scatter(data['Weight'], data['Height'], c = data['Gender'].apply(lambda
x:colors[x]))
plt.scatter(weight, height, c = 'b')
plt.scatter(neighbors['Weight'], neighbors['Height'], c = 'k')
plt.show()
```



Neighbors are in black

Thank you !