

k-Nearest Neighbors

☑ Trắc nghiệm 26 📄 Thực hành 1 ⬇ PDF bài

kNN (k-Nearest Neighbors) dự đoán cho một mẫu mới bằng cách tìm k mẫu gần nó nhất trong dữ liệu đã có rồi lấy nhãn chiếm đa số. Ví dụ: xếp thành viên mới vào nhóm luyện dựa trên những thành viên có điểm thuật toán và số giờ tự học gần giống. Vì dựa hoàn toàn vào khoảng cách, kNN rất nhạy với đơn vị đo: đổi giờ sang phút có thể làm đổi kết quả. Bài này học cách kNN dự đoán, cách chọn k và vì sao phải chuẩn hóa dữ liệu trước.

MỤC TIÊU

- Tìm k láng giềng gần nhất và tính dự đoán bằng tay.
- Biết ảnh hưởng của cách đo khoảng cách, đơn vị đo và k .
- Chọn k bằng Pipeline và cross-validation.

Thuật toán

Cho tập huấn luyện $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ và một điểm cần dự đoán $\mathbf{q}$:

1. Tính khoảng cách $d(\mathbf{q}, \mathbf{x}_i)$ đến mọi điểm.
2. Lấy tập $N_k(\mathbf{q})$ gồm k điểm gần nhất.
3. Phân loại: chọn nhãn xuất hiện nhiều nhất trong N_k . Hồi quy: lấy trung bình y của N_k .

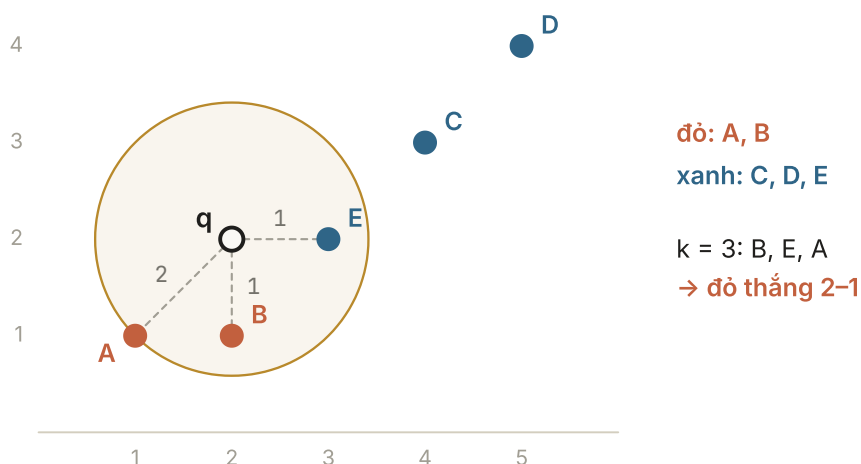
Ví dụ năm điểm $A(1, 1)$ nhãn đỏ, $B(2, 1)$ đỏ, $C(4, 3)$ xanh, $D(5, 4)$ xanh, $E(3, 2)$ xanh.

Truy vấn $\mathbf{q} = (2, 2)$, $k = 3$, khoảng cách Euclid bình phương:

Điểm	A	B	C	D	E
d^2	2	1	5	13	1

Ba láng giềng gần nhất là B, E, A : hai đỏ, một xanh, nên dự đoán đỏ. Với $k = 1$ thì B và E cùng khoảng cách, nên cần quy tắc phá hòa, chẳng hạn chọn điểm có chỉ số nhỏ hơn.

Đề chấm tự động thường ghi rõ quy tắc phá hòa; đọc sót nó là đủ để kết quả lệch đáp án.



Ba láng giềng gần q nhất là B, E, A; đỏ thắng 2-1.

Biến thể có trọng số cho láng giềng càng gần càng có tiếng nói lớn, ví dụ trọng số $1/d$. Một láng giềng lớp A cách 1 và hai láng giềng lớp B cách 2, 3: bỏ phiếu đều thì B thắng 2-1, còn theo $1/d$ thì A được 1, B chỉ được $1/2 + 1/3 \approx 0.83$. Hồi quy cũng vậy: giá trị 10, 20, 40 ở khoảng cách 1, 2, 4 cho $(10 + 10 + 10)/1.75 \approx 17.1$, gần láng giềng sát nhất hơn trung bình thường 23.3.

kNN là học lười (lazy learning): “huấn luyện” chỉ là cất dữ liệu đi, mọi công việc dồn vào lúc dự đoán. Ngược lại, mô hình tuyến tính là học chăm (eager): học xong chỉ giữ $\mathbf{w}$, dữ liệu gốc có thể bỏ. kNN vẫn là học có giám sát, vì các láng giềng phải có nhãn mới bỏ phiếu được. Chữ k ở đây là số láng giềng, khác hẳn K của K-means ở bài 2.10, là số cụm.

HỎI NHANH

Có ba láng giềng: một láng giềng nhãn 0 ở khoảng cách 1, hai láng giềng nhãn 1 ở khoảng cách 4 và 5. Bỏ phiếu đều thì ra nhãn nào? Bỏ phiếu theo trọng số $1/d$ thì ra nhãn nào?

A Bỏ phiếu đều ra 0, trọng số $1/d$ ra 1

B Bỏ phiếu đều ra 1, trọng số $1/d$ ra 0

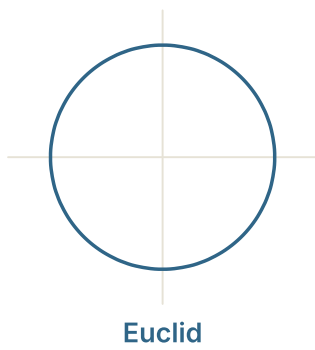
C Cả hai cách đều ra nhãn 1

D Cả hai cách đều ra nhãn 0

Khoảng cách

Tên	Công thức	Hợp với
Euclid (L2)	$\sqrt{\sum_j (q_j - x_j)^2}$	Đặc trưng số đã chuẩn hóa
Manhattan (L1)	$\sum_j q_j - x_j $	Ít nhạy với một chiều lệch lớn
Cosine	$1 - \frac{\mathbf{q}^\top \mathbf{x}}{\ \mathbf{q}\ \ \mathbf{x}\ }$	Embedding, văn bản: quan tâm hướng, không quan tâm độ dài

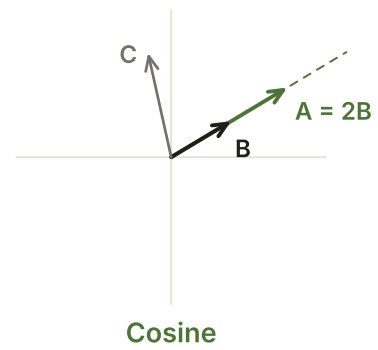
Với vector đã chuẩn hóa về độ dài 1, $\|\mathbf{q} - \mathbf{x}\|^2 = 2 - 2\cos(\mathbf{q}, \mathbf{x})$, nên xếp hạng theo Euclid và theo cosine là như nhau. Thư viện tìm kiếm embedding dùng đúng mẹo này. Đổi cách đo có thể đổi láng giềng: từ (0, 0), điểm (3, 0) cách 3 theo cả hai cách, còn (2, 2) cách $\sqrt{8} \approx 2.83$ theo Euclid nhưng cách 4 theo Manhattan. Muốn một cột nặng ký hơn thì đánh trọng số ngay trong khoảng cách, $\sqrt{\sum_j w_j (q_j - x_j)^2}$.



Euclid



Manhattan



Cosine

Các điểm cách gốc đúng 1 theo Euclid, Manhattan; cosine chỉ xét hướng.

HỎI NHANH

Văn bản A là văn bản B chép lại hai lần, nên vector đếm từ của A đúng bằng 2 lần vector của B. Theo cosine và theo Euclid, A và B có gần nhau không?

- A Cả hai khoảng cách đều bằng 0
- B Khoảng cách cosine lớn, Euclid bằng 0
- C Cả hai khoảng cách đều lớn
- D Khoảng cách cosine bằng 0, Euclid khá lớn

Thang đo quyết định tất cả. Hai đặc trưng “tuổi” (20 đến 60) và “thu nhập tháng” tính bằng đồng (5 triệu đến 50 triệu): khoảng cách gần như chỉ còn là chênh lệch thu nhập, tuổi chẳng còn tiếng nói nào. Khi đơn vị các cột không so sánh được, chuẩn hóa hoặc thiết kế trọng số khoảng cách là bước cần kiểm bằng validation.

Chọn k

- $k = 1$: lỗi trên train thường bằng 0 nếu không có hồ sơ trùng nhau mang nhãn mâu thuẫn, ranh giới lởm chởm theo từng điểm nhiễu. Variance cao.
- k lớn: ranh giới mượt, nhưng $k = n$ thì đoán mọi điểm là lớp đông nhất. Bias cao.

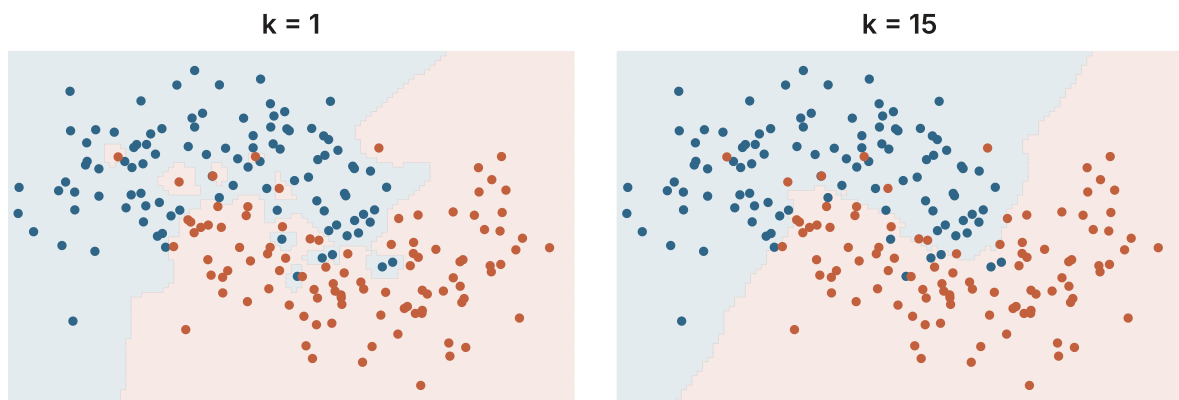
Chọn k bằng cross-validation. Với phân loại nhị phân, k lẻ tránh hòa phiếu. Leave-one-out (bài 1.3) rất rẻ với kNN: bỏ từng điểm ra rồi hỏi láng giềng trong phần còn lại, không phải học lại gì. Trên một trục số, 1-NN chỉ đổi dự đoán ở trung điểm giữa hai điểm liền kề khác nhãn, nên nó chính là một dãy ngưỡng.

HỎI NHANH

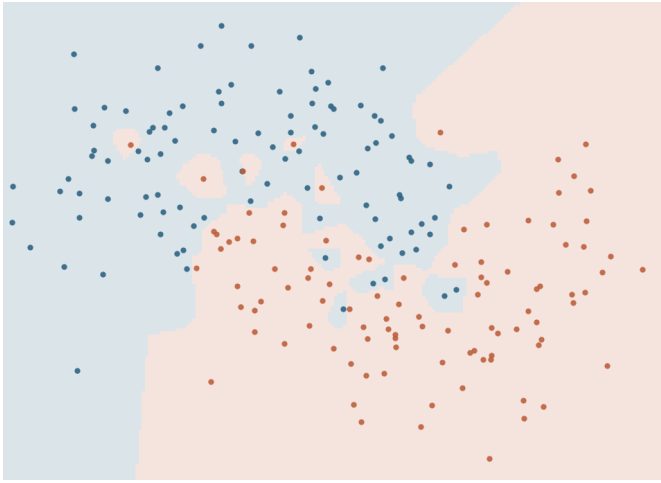
Một bạn chọn $k = 1$, vì trong các k đã thử, $k = 1$ có lỗi train thấp nhất, bằng 0. Chọn như vậy có ổn không?

- A Ổn, vì lỗi train bằng 0 là tốt nhất có thể
- B Ổn, nếu trong train không có điểm nào trùng nhau
- C Không ổn, phải chọn k lớn nhất đã thử cho an toàn
- D Không ổn, vì 1-NN luôn khớp train do mỗi điểm tự là láng giềng của nó

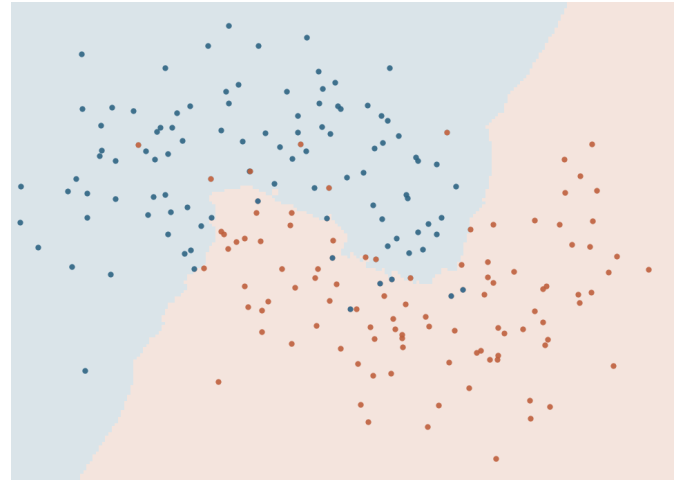
PHÒNG THEO CMU 10-601 · Exam 1 Practice · Q2.3(f)



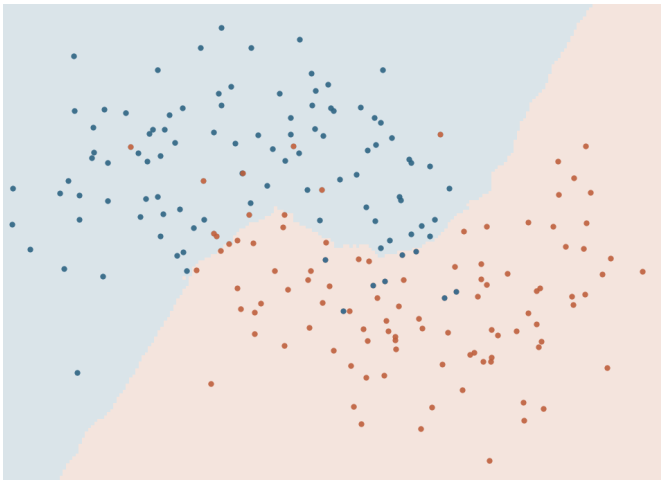
$k = 1$ bám từng điểm nhiều; $k = 15$ cho ranh giới mượt.



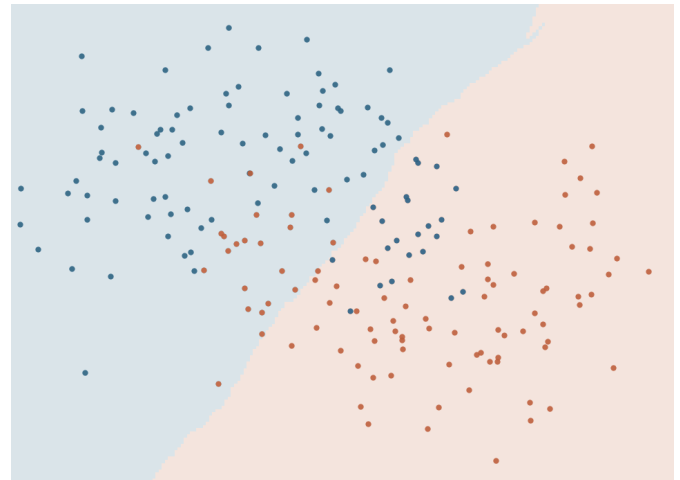
k = 1: accuracy 5-fold CV 0.880



k = 15: accuracy 5-fold CV 0.905



k = 50: accuracy 5-fold CV 0.875



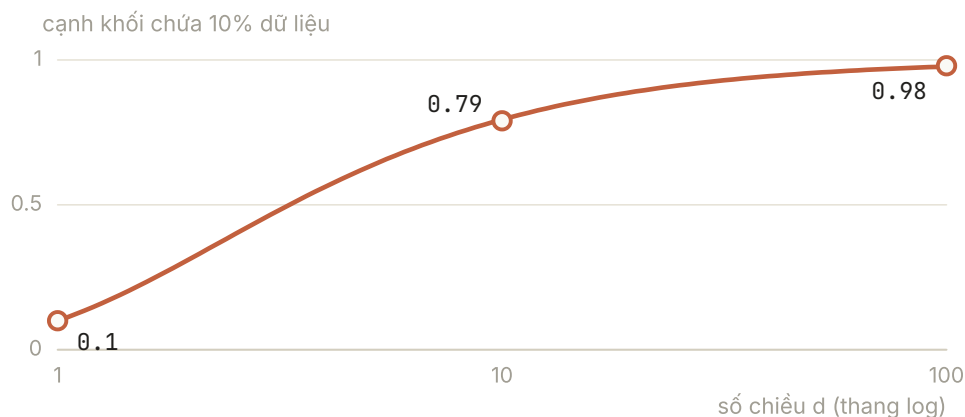
k = 120: accuracy 5-fold CV 0.750

Cùng 200 điểm của hình trên, tăng k từ 1 tới 120: ranh giới đi từ bám từng điểm nhiều tới quá phẳng.

Lời nguyên số chiều

Khi số chiều tăng, với dữ liệu trải đều, điểm gần nhất và điểm xa nhất cách truy vấn gần bằng nhau, nên “láng giềng” mất ý nghĩa. Dữ liệu thật thường nằm gần một cấu trúc ít chiều hơn nhiều, nên mức độ tẻ còn tùy số chiều thực chất của nó. Một phép tính nhỏ cho thấy hiện tượng này. Dữ liệu rải đều trong khối lập phương đơn vị d chiều. Muốn một khối lập phương nhỏ chứa 10% dữ liệu thì cạnh của nó phải dài $0.1^{1/d}$:

- $d = 1$: cạnh 0.1, đúng là “lân cận”.
- $d = 10$: cạnh 0.79, gần hết cả khoảng.
- $d = 100$: cạnh 0.98.



Gom 10% dữ liệu: cạnh khối dài 0.1 ở 1 chiều, 0.98 ở 100 chiều.

Láng giềng gần nhất trong 100 chiều thực chất chẳng gần chút nào. Nhìn theo chiều ngược lại: khối lập phương đồng tâm cạnh 0.9 chỉ chứa $0.9^{100} \approx 0.003\%$ thể tích, gần như mọi điểm nằm sát biên. Muốn láng giềng gần nhất gần gấp đôi thì mỗi điểm phải phủ một quả cầu bán kính bằng nửa, thể tích nhỏ đi 2^d lần, nên cần khoảng 2^d lần số điểm. Vì vậy kNN chạy tốt trên đặc trưng ít chiều, hoặc trên embedding mà mô hình đã học sẵn để những thứ giống nhau nằm gần nhau, thường kèm giảm chiều bằng PCA (bài 2.9).

HỎI NHANH

Dữ liệu rải đều trong một khối lập phương đơn vị 50 chiều. Một khối lập phương nhỏ chứa 1% dữ liệu thì có cạnh dài khoảng bao nhiêu?

A 0.02

B 0.2

C 0.91

D 0.99

Chi phí

Dự đoán một điểm tốn $O(nd)$. Với n lớn, có ba cách tăng tốc:

- Cây không gian (KD-tree, Ball tree): nhanh khi d nhỏ, dưới khoảng 20.
- Vector hóa bằng phép nhân ma trận trên GPU: tính một lần khoảng cách giữa cả lô truy vấn và cả tập dữ liệu.
- Tìm gần đúng (approximate nearest neighbor) như FAISS, HNSW: chấp nhận đôi khi sót láng giềng thật để nhanh hơn hàng trăm lần. Đây là nền của các hệ thống tìm kiếm ảnh và văn bản. Chúng có núp đổi tốc độ lấy độ đúng, như `ef_search` của HNSW: giữ nhiều ứng viên hơn khi duyệt thì ít sót láng giềng thật nhưng chậm hơn.

Vài lưu ý khi tự cài:

- Chọn k phần tử nhỏ nhất: sort toàn bộ tốn $O(n \log n)$; `np.argsort` chọn trước rồi chỉ sort k phần tử thì rẻ hơn.
- Với m truy vấn một lúc, ma trận khoảng cách $m \times n$ có thể hết RAM dù code chỉ một dòng; chia truy vấn thành từng batch.
- KD-tree không phải lúc nào cũng $O(\log n)$: ở nhiều chiều nó có thể phải duyệt gần hết cây.

HỎI NHANH

Cần tìm ảnh giống nhau trong 1 triệu ảnh, mỗi ảnh là một embedding 512 chiều, và mỗi truy vấn phải trả lời trong vài mili giây. Cách nào hợp lý?

- A KD-tree, vì nó luôn bảo đảm $O(\log n)$ mỗi truy vấn
- B Giảm k xuống 1 để mỗi truy vấn tính ít khoảng cách hơn
- C Duyệt hết bằng vòng `for` Python qua từng ảnh
- D Tìm gần đúng (HNSW, FAISS), chấp nhận đôi khi sót

Cài kNN bằng NumPy

Cài lại ví dụ năm điểm ở mục Thuật toán, chưa cần thư viện. Mỗi hàng là một hồ sơ, hai cột là hai đặc trưng. Trừ truy vấn khỏi từng hàng bằng broadcasting, bình phương từng độ lệch, rồi cộng theo `axis=1` để được một khoảng cách cho mỗi hàng. Không lấy căn vì căn bậc hai tăng theo đối số: xếp hạng khoảng cách bình phương cho cùng thứ tự, tránh thêm một phép tính.

Python

```
import numpy as np

points = np.array([[1, 1], [2, 1], [4, 3], [5, 4], [3, 2]], dtype=float)
labels = np.array([0, 0, 1, 1, 1])
query = np.array([2, 2], dtype=float)
distance2 = np.sum((points - query) ** 2, axis=1)
order = np.argsort(distance2, kind="stable")
neighbors = order[:3]
counts = np.bincount(labels[neighbors], minlength=2)
print(distance2.astype(int).tolist())
print(neighbors.tolist(), counts.tolist(), int(np.argmax(counts)))
assert neighbors.tolist() == [1, 4, 0]
```

KẾT QUẢ

```
[2, 1, 5, 13, 1]
[1, 4, 0] [2, 1] 0
```

Chỉ số trong code bắt đầu từ 0, nên `[1, 4, 0]` chính là B, E, A. `stable` giữ thứ tự ban đầu khi khoảng cách bằng nhau; `argmax` chọn nhãn có chỉ số nhỏ nhất khi số phiếu hòa. Đây là hai quy tắc của **bản cài này**, không phải quy tắc chung của mọi thư viện. Với `k=1`, đổi thứ tự dữ liệu có thể đổi dự đoán ở điểm hòa khoảng cách. Trong đề chấm chính xác, phải cài quy ước đề yêu cầu.

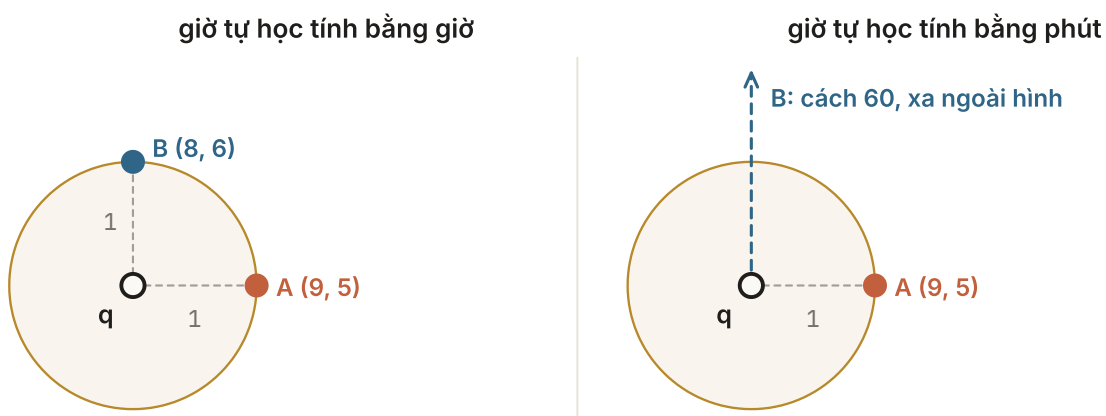
HỎI NHANH

Vì sao không cần lấy căn của `distance2` trước khi sắp xếp?

- A Vì khoảng cách Euclid vốn không có căn
- B Vì căn bậc hai giữ nguyên thứ tự của các số không âm
- C Vì khoảng cách bình phương bằng khoảng cách Euclid
- D Vì `argsort` không chạy được trên số thực

Thang đo thay đổi người hàng xóm

Xét hồ sơ truy vấn (8 điểm, 5 giờ), hai hồ sơ A (9, 5) và B (8, 6). Trong đơn vị hiện tại cả hai cách truy vấn 1. Nếu đổi giờ thành phút, B cách 60 còn A vẫn cách 1; thông tin thực tế không đổi mà thuật toán đã thiên hẫn về A. Hình dưới cho thấy đổi đơn vị kéo một trục dài ra. `StandardScaler` đưa các cột về cùng thang đo; còn nếu biết một cột quan trọng hơn, có thể nhân cột đó với một hệ số lớn hơn, rồi kiểm bằng validation.



Đổi giờ sang phút: B từ cách q 1 thành cách 60.

Cần phân biệt chuẩn hóa **theo cột** với chuẩn hóa **độ dài từng vector**. `StandardScaler` thường dùng cho các đặc trưng số khác đơn vị. Với embedding dùng cosine, ta thường

chia mỗi hàng cho norm của chính hàng đó. Vector toàn 0 không có hướng, nên cosine chưa xác định; cần quy tắc xử lý riêng trước khi tính.

HỎI NHANH

Mỗi ảnh là một embedding 512 chiều, và ta tìm ảnh giống nhau bằng cosine. Muốn chạy KNN theo Euclid cho nhanh, ta chuẩn hóa thế nào?

- A Dùng StandardScaler theo từng cột của bảng embedding
- B Chia mỗi hàng cho độ dài (norm) của chính nó
- C Đưa từng cột về đoạn $[0, 1]$ theo min và max
- D Không cần, vì Euclid và cosine luôn xếp hạng như nhau

Dùng thư viện

Python

```
from sklearn.datasets import load_wine
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

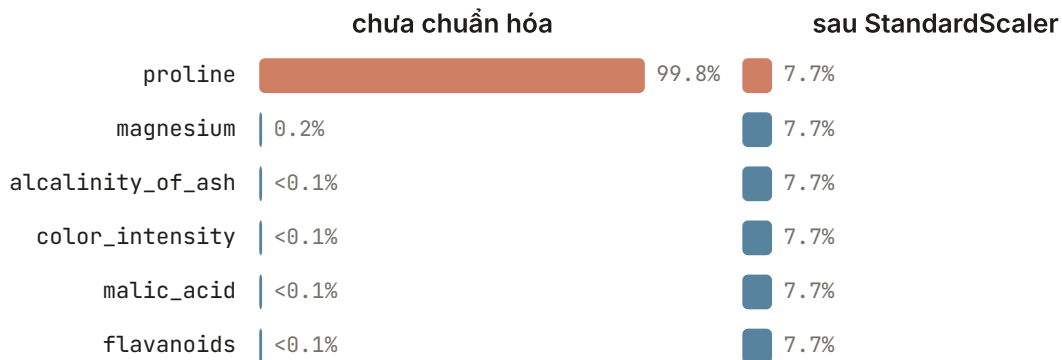
X, y = load_wine(return_X_y=True)
for scale in [False, True]:
    steps = [StandardScaler()] if scale else []
    model = make_pipeline(*steps, KNeighborsClassifier())
    grid = GridSearchCV(model, {"kneighborsclassifier__n_neighbors": [1, 3, 5, 9, 15]},
cv=5)
    grid.fit(X, y)
    print("chuẩn hóa" if scale else "không chuẩn hóa", round(grid.best_score_, 3),
grid.best_params_)
```

KẾT QUẢ

không chuẩn hóa 0.725 {'kneighborsclassifier__n_neighbors': 1}

chuẩn hóa 0.966 {'kneighborsclassifier__n_neighbors': 9}

Chênh lệch lớn có lý do cụ thể: cột proline chạy từ 278 đến 1680, trong khi phần lớn các cột khác chỉ cỡ 0.1 đến 30. Chưa chuẩn hóa, proline chiếm khoảng 99.8% bình phương khoảng cách, nên kNN gần như chỉ so một cột và phải dùng $k = 1$ để vớt vát.



phần của mỗi cột trong bình phương khoảng cách (6 cột lớn nhất trên 13)

Chưa chuẩn hóa, cột proline chiếm 99.8% khoảng cách.

HỎI NHANH

Trong bộ wine chưa chuẩn hóa, riêng cột proline chiếm khoảng 99.8% bình phương khoảng cách. Thêm một feature hữu ích nằm trong khoảng $[0, 1]$, thì kNN với khoảng cách thô sẽ ra sao?

- A Gần như không đổi, vì feature mới gần như vô hình
- B Tốt hơn hẳn, vì có thêm thông tin
- C Tệ hơn hẳn, vì thêm một chiều nữa
- D Không chạy được, vì các cột khác thang đo

Khi đi thi

- Đọc kỹ kiểu khoảng cách và quy tắc phá hòa trong đề.
- Chuẩn hóa các cột trước khi tính khoảng cách; đặt scaler và kNN trong cùng Pipeline khi dò k .
- Có embedding thì chuẩn hóa độ dài rồi dùng kNN theo cosine làm baseline.

Tóm tắt

- kNN dự đoán theo k điểm gần nhất; không có giai đoạn học tham số.
- Chuẩn hóa đặc trưng và chọn khoảng cách đúng quan trọng hơn chọn k .
- k nhỏ thì variance cao, k lớn thì bias cao; chọn bằng cross-validation.
- Nhiều chiều thì “gần” mất nghĩa; kNN mạnh nhất trên embedding tốt.

Kiểm tra cuối bài

Mười câu kiểm cả cách hiểu, phép tính và quy trình. Câu sai chỉ ra phần cần đọc lại; hãy làm trước khi mở lời giải.

CÂU 1

Năm láng giềng có nhãn $[1, 0, 0, 1, 0]$, bỏ phiếu đều. Nhãn dự đoán là gì, và xác suất lớp 1 tính theo tỉ lệ láng giềng là bao nhiêu?

A Nhãn 0, xác suất 0.4

B Nhãn 1, xác suất 0.4

C Nhãn 0, xác suất 0.6

D Nhãn 1, xác suất 0.6

CÂU 2

Câu nào về kNN **luôn** đúng?

A Giảm k làm ranh giới quyết định đơn giản hơn

B kNN chỉ dùng cho phân loại, không dùng cho hồi quy

C Tăng k làm mô hình ít nhạy với điểm ngoại lai

D kNN chỉ dùng được với feature là số thực

PHÒNG THEO CMU 10-601 · Exam 1 Practice · Q2.4

CÂU 3

Hai láng giềng có khoảng cách $d_1 = 1$ và $d_2 = 2$, target lần lượt là 0 và 6. Nếu trọng số là $1/d$, dự đoán hồi quy bằng bao nhiêu?

A 2

B 3

C 1.2

D 6

CÂU 4

Dữ liệu một chiều, n điểm đã sắp xếp, phân loại hai lớp bằng 1-NN với khoảng cách Euclid. Có dựng được một quy tắc chỉ gồm các phép so sánh dạng $x \geq t$, nối tiếp nhau, mà cho dự đoán giống hệt 1-NN trên mọi đầu vào không?

A Được, đặt ngưỡng ở trung điểm của hai điểm kề nhau khác nhãn

B Không được, vì so sánh từng ngưỡng không bắt chước được khoảng cách

C Chỉ được khi các điểm train cách đều nhau trên trục

D Chỉ được nếu dùng k lẻ thay vì $k = 1$

PHÒNG THEO CMU 10-601 · Exam 1 Practice · Q2.5

CÂU 5

Một kNN dùng StandardScaler fit trên train. Đổi cột thu nhập từ triệu đồng sang đồng ở cả train lẫn validation, rồi fit lại scaler đúng trên train. Bỏ qua sai số làm tròn, khoảng cách sau chuẩn hóa thay đổi thế nào?

- A Không đổi, vì mean và std cũng đổi theo đơn vị
- B Cột thu nhập vẫn lớn hơn 1 000 000 lần
- C Khoảng cách ở mọi cột giảm 1 000 000 lần
- D Chỉ không đổi khi scaler fit cả trên validation

CÂU 6

Đề phân loại hai lớp, 30 feature số, chấm bằng macro-F1. Đề **bắt buộc** kNN là mô hình học có giám sát duy nhất, còn tiền xử lý thì tùy ý. Hướng làm nào hợp lý nhất?

- A Chỉ dò thật kỹ k , vì k là tham số duy nhất của kNN
- B Học một mô hình khác như boosting trước, rồi lấy dự đoán của nó làm feature cho kNN
- C Chuẩn hóa, chọn feature trong Pipeline, rồi dùng cross-validation theo macro-F1
- D Fit StandardScaler trên cả train lẫn test cho cùng thang

PHÒNG THEO [USA-NA-AIO 2026 · Vòng 1 · Problem 9](#)

CÂU 7

$k = 3$, ba láng giềng gần nhất mang ba nhãn khác nhau A, B, C, bỏ phiếu đều. Trước khi áp quy tắc xử lý hòa của thư viện, kết quả là gì?

- A A thắng, vì là láng giềng gần nhất
- B C thắng, vì đứng cuối danh sách
- C Không thể hòa, vì k là số lẻ
- D Hòa 1-1-1, k lẻ cũng không tránh được

CÂU 8

Hai vector đơn vị $\mathbf{u}$, $\mathbf{v}$ có cosine bằng 0.5. Bình phương khoảng cách Euclid giữa chúng bằng bao nhiêu?

- A 0.5
- B 1
- C 2
- D 4

CÂU 9

Có 10 000 truy vấn và 10 000 điểm tham chiếu. Ma trận khoảng cách kiểu float64 tốn bao nhiêu byte?

A 8×10^6

B 8×10^7

C 8×10^8

D 8×10^{10}

CÂU 10

Trên cùng các fold, $k = 1, 5, 21$ cho accuracy trên validation lần lượt là 0.76, 0.84 và 0.79. Chỉ chọn trong ba giá trị này thì chọn gì?

A $k = 21$, vì lấy trung bình nhiều mẫu luôn tốt hơn

B $k = 1$, vì không làm mờ cấu trúc cục bộ

C $k = 5$, vì cao nhất trong những giá trị đã thử

D Nội suy accuracy rồi chọn $k = 13$, không cần chấm

Đọc thêm

- James, Witten, Hastie, Tibshirani — [An Introduction to Statistical Learning](#), chương 2 và 4. kNN đặt cạnh các bộ phân loại khác, với lập luận bias-variance khi chọn k .
- Cover, Hart (1967) — *Nearest Neighbor Pattern Classification*, IEEE Transactions on Information Theory. Bài báo gốc của cận hai lần lỗi Bayes.
- [scikit-learn — Nearest Neighbors](#). So sánh metric, brute force, KD-tree và Ball tree; chú ý phần chi phí theo số chiều.
- [ISLP — Classification lab](#). Đối chiếu kNN với mô hình xác suất và thực hành lựa chọn k .