

Blog Tuần 2 – Module 3

K-Means, KNN và PySpark: Từ Thuật Toán Đến Ứng Dụng Dữ Liệu Lớn

Hiểu sâu, triển khai và khai thác sức mạnh của các công cụ phân tích dữ liệu

Tác giả: GRID034

Tuần thứ hai của Module 3 tiếp tục hành trình khám phá các công cụ và thuật toán quan trọng trong phân tích và xử lý dữ liệu. Bài Blog tuần này tập trung vào ba mảng lớn: **K-Means** – thuật toán phân cụm đơn giản nhưng mạnh mẽ, **KNN** – phương pháp “hồi hàng xóm” trực quan và dễ hiểu trong học máy, và **PySpark** – công cụ toàn diện để xử lý dữ liệu lớn. Nội dung không chỉ giải thích nguyên lý, công thức và quy trình hoạt động, mà còn kèm theo cài đặt chi tiết bằng Python và các tình huống ứng dụng thực tế, giúp bạn có thể áp dụng ngay vào công việc.

Các chủ đề nổi bật bao gồm:

1. Thuật toán K-Means

- Nguyên lý hoạt động, công thức tổng quát và tiêu chí tối ưu WCSS.
- Cài đặt từ đầu bằng NumPy, từ khởi tạo centroid, tính khoảng cách, gán nhãn đến hội tụ.
- Ứng dụng với dữ liệu số, dữ liệu ảnh và văn bản (BoW, TF-IDF).
- Phân tích kết quả thực nghiệm, minh họa quá trình hội tụ và đánh giá chất lượng cụm.

2. Thuật toán k-Nearest Neighbors (KNN)

- Ba đặc trưng quan trọng: Supervised Learning, Non-parametric, Instance-based.
- Bốn bước hoạt động: chọn k , tính khoảng cách, tìm láng giềng gần nhất, ra quyết định.
- Thực hành dự đoán kết quả học tập sinh viên (Classification và Regression).
- Ứng dụng trong hệ thống gợi ý của Netflix và cài đặt KNN đơn giản bằng Python.

3. PySpark – Xử lý dữ liệu lớn

- Giới thiệu Big Data và các thách thức 5V: Volume, Velocity, Variety, Veracity, Value.
- So sánh Spark và Hadoop; cấu trúc RDD, DataFrame, Spark SQL.
- Các thao tác cơ bản với RDD, DataFrame, UDF, và Pipeline MLlib.
- Ứng dụng PySpark trong phân tích dữ liệu lớn và mô hình hóa học máy.

Giá trị nhận được sau khi đọc Blog

- Nắm vững nguyên lý và cách triển khai K-Means và KNN bằng Python.
- Biết cách áp dụng hai thuật toán này trong các bài toán thực tế từ phân cụm đến gợi ý.
- Thành thạo các thao tác cơ bản với PySpark để xử lý dữ liệu lớn.
- Hiểu rõ sự khác biệt và ứng dụng của Spark so với các công cụ xử lý dữ liệu khác.

Hiểu và Úng dụng Thuật Toán K-Means

Dàm Nguyễn Khánh

1. Giới thiệu

K-Means là một thuật toán **phân cụm** (*clustering*) thuộc nhóm **học máy không giám sát** (*unsupervised learning*). Mục tiêu là nhóm các điểm dữ liệu thành K cụm sao cho các điểm trong cùng cụm *tương tự nhau nhất* và *khác biệt nhất* với *các cụm khác*.

Úng dụng thực tế:

- Phân nhóm khách hàng (marketing)
- Nén ảnh (image compression)
- Phát hiện bất thường (anomaly detection)
- Giảm thiểu và tiền xử lý dữ liệu

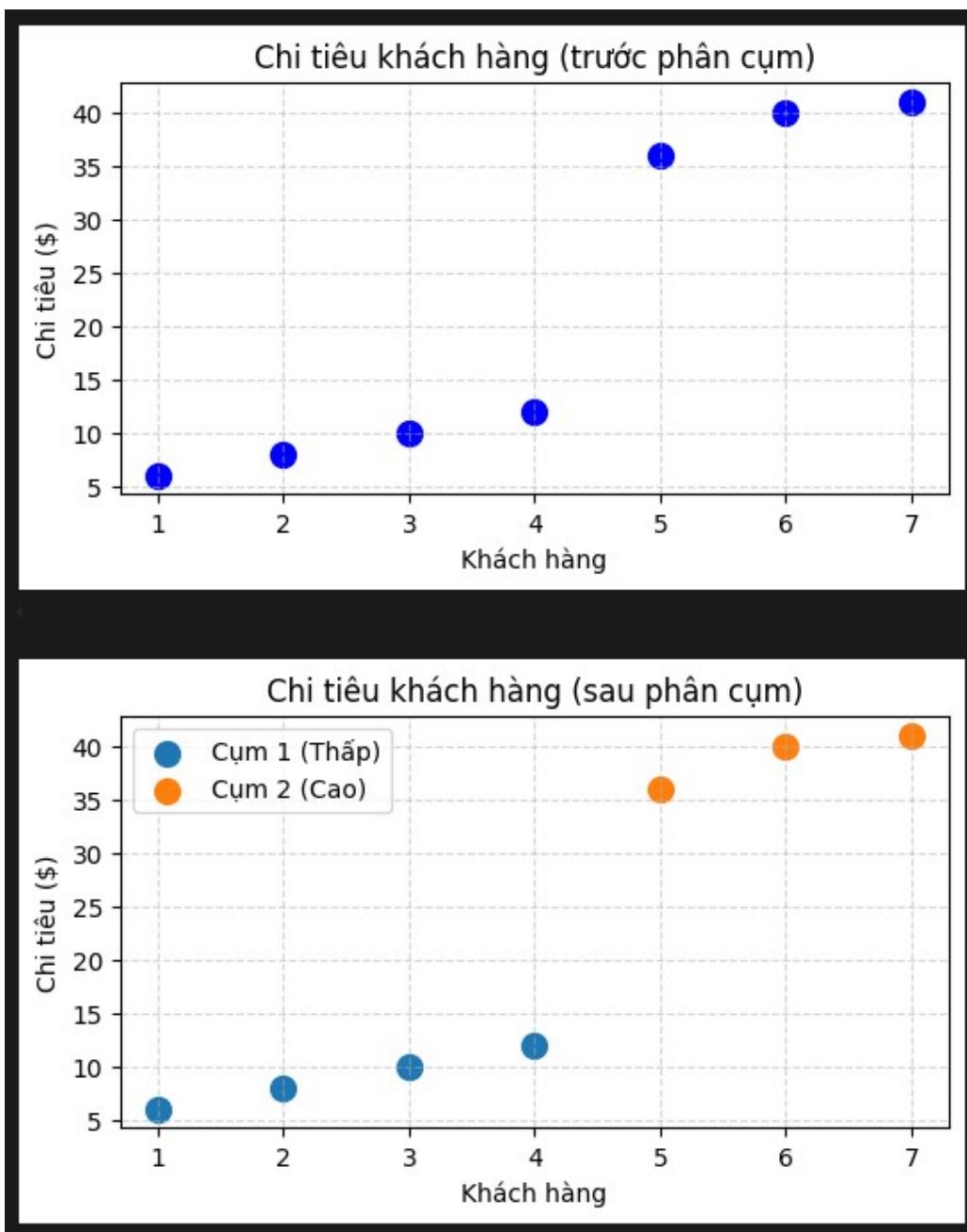
2. Bối cảnh và Động lực

Trong nhiều bài toán, dữ liệu **không được gắn nhãn**. Ví dụ: danh sách chi tiêu khách hàng nhưng không có thông tin nhóm.

Câu hỏi đặt ra:

- Làm sao phân nhóm khách hàng VIP và khách hàng thường?
- Làm sao tìm ra cấu trúc ẩn trong dữ liệu?

Ví dụ: Khách hàng có mức chi tiêu: 6, 8, 10, 12, 36, 40, 41 (\$).



Hình 1: Biểu đồ scatter/cột thể hiện dữ liệu chi tiêu khách hàng trước và sau khi phân cụm.

Điễn giải:

- **Biểu đồ trên:** Chi tiêu của khách hàng trước khi phân cụm, tất cả điểm có cùng màu vì chưa phân loại.
- **Biểu đồ dưới:** Kết quả sau khi áp dụng K-Means với 2 cụm:
 - **Cụm 1 (Thấp):** nhóm khách hàng chi tiêu thấp.
 - **Cụm 2 (Cao):** nhóm khách hàng chi tiêu cao.
- Phân cụm giúp nhận diện nhóm khách hàng dựa trên hành vi chi tiêu, hỗ trợ phân tích và ra quyết định kinh doanh.

3. Nguyên lý hoạt động của K-Means

K-Means hoạt động theo **chu trình lặp**:

1. Chọn số cụm K .
2. Khởi tạo ngẫu nhiên K tâm cụm (centroids).
3. Gán mỗi điểm vào cụm có tâm gần nhất (khoảng cách Euclidean).
4. Tính lại tâm cụm mới.
5. Lặp lại cho đến khi hội tụ.

Khoảng cách Euclidean:

$$D(x, c) = \sqrt{\sum_{i=1}^n (x_i - c_i)^2}$$

4. Công thức tổng quát

Bước gán cụm:

$$S_i^{(t)} = \{x_j : \|x_j - c_i^{(t)}\|^2 \leq \|x_j - c_k^{(t)}\|^2\}$$

Bước cập nhật tâm cụm:

$$c_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Tiêu chí tối ưu - WCSS:

$$WCSS = \sum_{i=1}^K \sum_{x \in S_i} \|x - c_i\|^2$$

5. Cài đặt K-Means với NumPy

1. Khởi tạo centroid.
2. Tính khoảng cách với broadcasting.
3. Gán nhãn bằng np.argmin().
4. Cập nhật centroid bằng np.mean().
5. Điều kiện dừng: np.all(C == C_new).

```

1 import numpy as np
2
3 def pairwise_l2(X, C):
4     """
5         Tính ma trận khoảng cách Euclidean giữa mỗi điểm dữ liệu và mỗi centroid.
6         X: (n_samples, n_features)
7         C: (k, n_features)
8         Trả về: D có shape (n_samples, k) với D[i, j] = ||X[i] - C[j]||_2
9         """
10        # Broadcasting: (n, 1, d) - (1, k, d) -> (n, k, d) rồi norm theo trục đặc trưng
11        return np.linalg.norm(X[:, None, :] - C[None, :, :], axis=2)

```

```
12
13 def wcss(X, labels, C):
14     """
15     Within-Cluster Sum of Squares (WCSS): tổng bình phương khoảng cách
16     của điểm đến centroid của cụm mà điểm thuộc về.
17     """
18     total = 0.0
19     for j in range(C.shape[0]):
20         Xj = X[labels == j]
21         if len(Xj) > 0:
22             total += np.sum((Xj - C[j])**2)
23     return float(total)
24
25 def init_kmeans_plus_plus(X, k, rng):
26     """
27     Khởi tạo k-means++: chọn ngẫu nhiên 1 tâm đầu,
28     sau đó chọn các tâm còn lại theo phân phối tỉ lệ với khoảng cách bình phương lớn nhất.
29     """
30     n = X.shape[0]
31     idx0 = rng.integers(0, n)
32     centers = [X[idx0]]
33     for _ in range(1, k):
34         D = pairwise_l2(X, np.vstack(centers)) # (n, m)
35         # Khoảng cách bình phương tới centroid gần nhất
36         d2 = np.min(D**2, axis=1)
37         probs = d2 / d2.sum()
38         next_idx = rng.choice(n, p=probs)
39         centers.append(X[next_idx])
40     return np.vstack(centers)
41
42 def kmeans(X, k, max_iter=100, tol=1e-4, init="k-means++", random_state=42):
43     """
44     K-Means chuẩn với:
45     - init in {"random", "k-means+"}
46     - điều kiện dừng: ||C_new - C_old||_F <= tol * ||C_old||_F
47     - xử lý cụm rỗng bằng cách re-seed centroid từ điểm xa nhất
48     Trả về:
49     labels: (n,), centers: (k,d), history: dict chứa WCSS theo vòng lặp
50     """
51     rng = np.random.default_rng(random_state)
52     n, d = X.shape
53
54     # Khởi tạo centroids
55     if init == "k-means++":
56         C = init_kmeans_plus_plus(X, k, rng)
57     elif init == "random":
58         C = X[rng.choice(n, size=k, replace=False)]
59     else:
60         raise ValueError("init must be 'random' or 'k-means++'")
61
```

```
62 hist_wcss = []
63 for it in range(max_iter):
64     # Bước gán cụm
65     D = pairwise_l2(X, C)          # (n, k)
66     labels = np.argmin(D, axis=1)    # (n, )
67
68     # Bước cập nhật centroid
69     C_new = C.copy()
70     for j in range(k):
71         Xj = X[labels == j]
72         if len(Xj) > 0:
73             C_new[j] = Xj.mean(axis=0)
74         else:
75             # Cum rỗng: chọn lại centroid là điểm xa nhất so với mọi centroid hiện tại
76             far_idx = np.argmax(np.min(D**2, axis=1))
77             C_new[j] = X[far_idx]
78
79     # Theo dõi hội tụ bằng WCSS
80     current_wcss = wcss(X, labels, C_new)
81     hist_wcss.append(current_wcss)
82
83     # Điều kiện dừng theo chuẩn Frobenius tương đối
84     denom = max(np.linalg.norm(C), 1e-12)
85     delta = np.linalg.norm(C_new - C) / denom
86     C = C_new
87     if delta <= tol:
88         break
89
90 return labels, C, {"wcss": np.array(hist_wcss), "iters": it + 1}
91
92 # ===== Ví dụ chạy nhanh =====
93 if __name__ == "__main__":
94     # Dữ liệu 2 chiều (Age, Expenditure) từ ví dụ trong bài
95     X = np.array([
96         [18, 80],
97         [20, 90],
98         [22, 85],
99         [30, 50],
100        [34, 64],
101        [40, 60],
102        [60, 30],
103        [66, 40],
104        [70, 25]
105    ], dtype=float)
106
107 labels, centers, hist = kmeans(X, k=3, init="k-means++", random_state=7)
108 print("Centers:\n", centers)
109 print("Labels:", labels)
110 print("WCSS history:", hist["wcss"])
111 print("Iterations:", hist["iters"])
```

Kết quả thực nghiệm

Sau khi chạy thuật toán K-Means với $k = 3$ trên tập dữ liệu khách hàng (Age, Expenditure), thu được kết quả như sau:

Bảng 1: Tâm cụm (Centers) sau khi hội tụ

Cụm	Tuổi (Age)	Chi tiêu (Expenditure)
0	65.33	31.67
1	20.00	85.00
2	34.67	58.00

Giải thích:

- Cụm 0** (màu giả định: xanh): nhóm khách hàng lớn tuổi (khoảng 65 tuổi) với mức chi tiêu trung bình thấp (≈ 32).
- Cụm 1** (màu giả định: đỏ): nhóm khách hàng trẻ (20 tuổi) với mức chi tiêu cao (≈ 85).
- Cụm 2** (màu giả định: vàng): nhóm khách hàng trung niên (≈ 35 tuổi) với chi tiêu trung bình khá (≈ 58).

Bảng 2: Nhãn cụm (Labels) của từng khách hàng

Khách hàng (Index)	Cụm
0	1
1	1
2	1
3	2
4	2
5	2
6	0
7	0
8	0

Giải thích: Các khách hàng $\{0, 1, 2\}$ thuộc nhóm trẻ – chi tiêu cao; $\{3, 4, 5\}$ thuộc nhóm trung niên – chi tiêu trung bình; $\{6, 7, 8\}$ thuộc nhóm cao tuổi – chi tiêu thấp.

Lịch sử WCSS và số vòng lặp

WCSS history = [1232.93, 803.08, 380.00, 380.00]

Số vòng lặp (Iterations) = 4

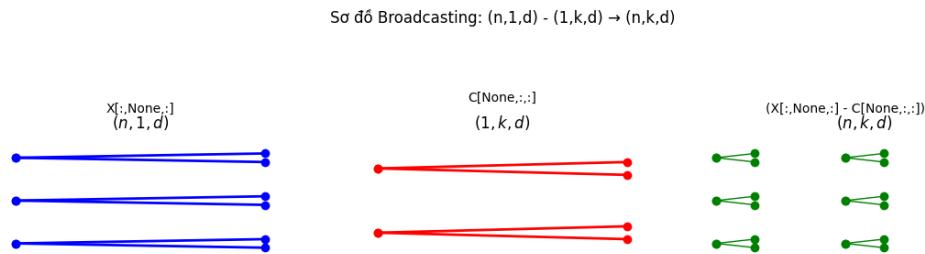
Giải thích:

- WCSS (Within-Cluster Sum of Squares)** giảm dần qua từng vòng lặp:

$$1232.93 \rightarrow 803.08 \rightarrow 380.00$$

Điều này chứng tỏ thuật toán đang cải thiện việc gom nhóm — các điểm ngày càng gần tâm cụm hơn.

- Ở vòng lặp thứ 3 và 4, WCSS không thay đổi (380.00), nghĩa là thuật toán đã **hội tụ**.
- Tổng số vòng lặp cần để hội tụ: 4 vòng.



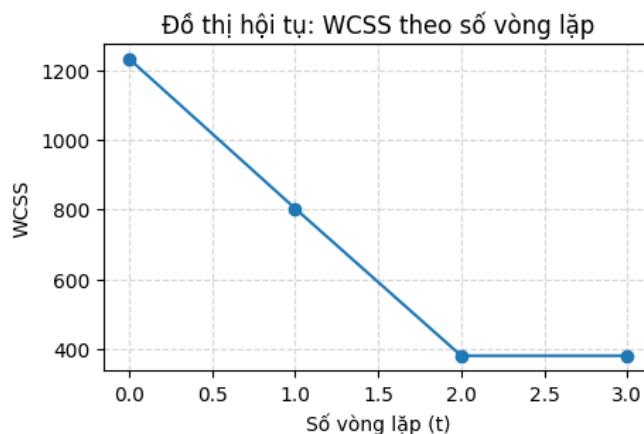
Hình 2: Minh họa quá trình **broadcasting** trong NumPy khi tính khoảng cách giữa dữ liệu và tâm cụm.

Chú thích:

$$\mathbf{X}[:, \text{None}, :] \rightarrow (n, 1, d), \quad \mathbf{C}[\text{None}, :, :] \rightarrow (1, k, d)$$

Khi trừ: $\mathbf{X}[:, \text{None}, :] - \mathbf{C}[\text{None}, :, :] \Rightarrow$ tensor (n, k, d) , trong đó n là số điểm dữ liệu, k là số centroid, d là số chiều đặc trưng.

Mã trận khoảng cách thu được bằng `np.linalg.norm(., axis=2)` $\Rightarrow (n, k)$.



Hình 3: Đồ thị biểu diễn quá trình hội tụ của thuật toán K-Means qua các vòng lặp.

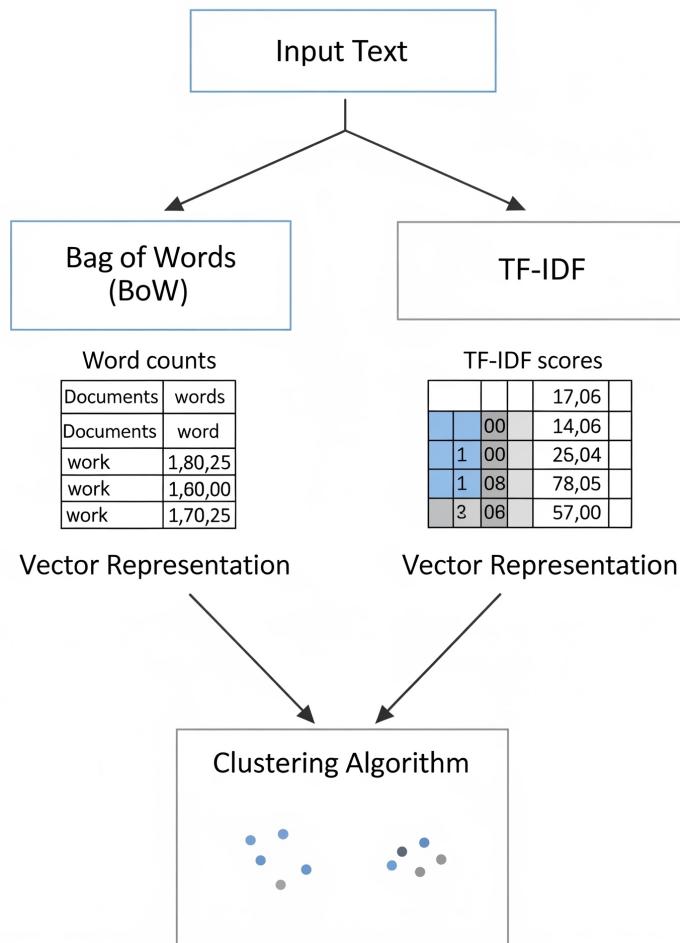
Chú thích:

- Trục hoành (x): Số vòng lặp t .
- Trục tung (y): Giá trị WCSS (*Within-Cluster Sum of Squares*).
- WCSS giảm mạnh từ 1232.93 \rightarrow 803.08 \rightarrow 380.00 trong 3 vòng đầu, cho thấy các điểm dữ liệu dần được gán vào cụm tối ưu hơn.
- Từ vòng 2 đến vòng 3, WCSS giữ nguyên ở 380.00, chứng tỏ thuật toán đã hội tụ và không còn cải thiện.

6. K-Means với dữ liệu ảnh và văn bản

6.1 Ảnh

Mỗi ảnh được biểu diễn thành vector đặc trưng, K-Means nhóm pixel/embedding để nén hoặc phân loại.



Hình 4: Quy trình sử dụng *Bag of Words* hoặc *TF-IDF* để biến văn bản thành vector và phân cụm.

Điều giải:

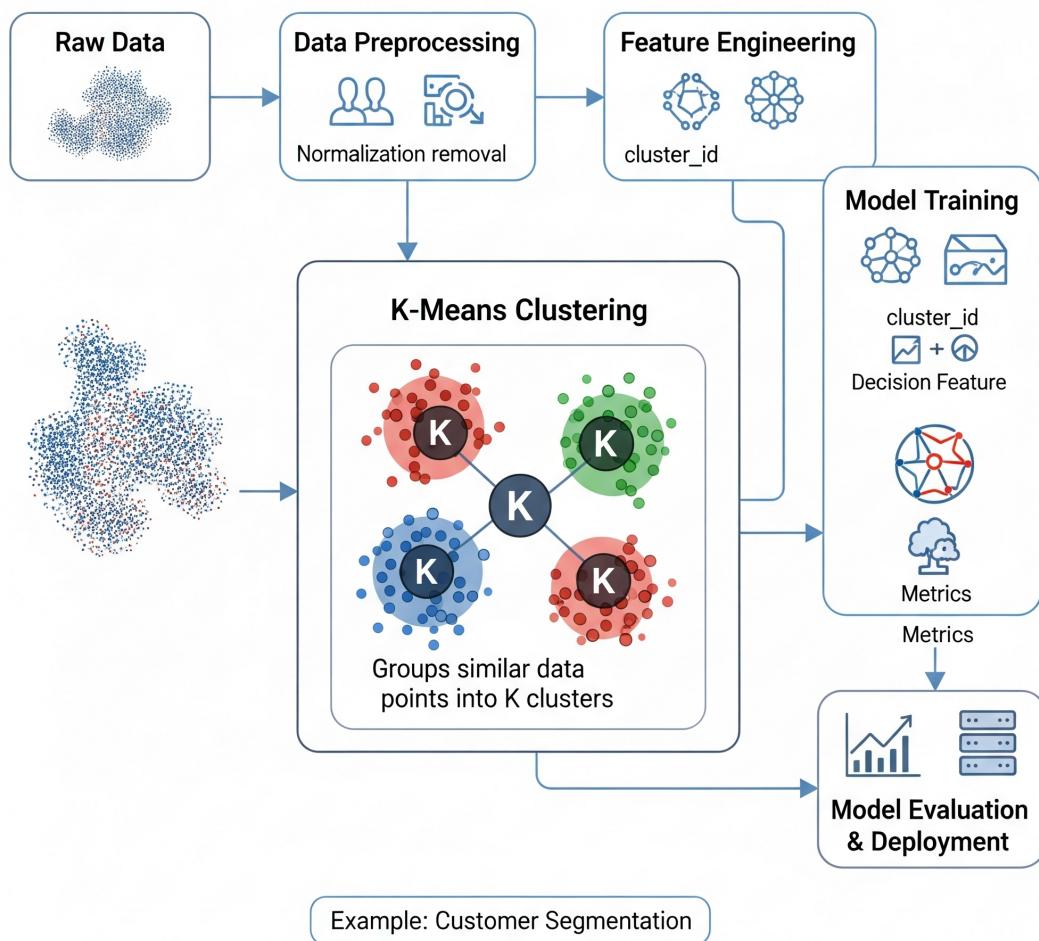
- **Input Text:** dữ liệu văn bản gốc.
- **Bag of Words (BoW):** biểu diễn văn bản dựa trên tần suất xuất hiện của từ (word counts) mà không xét thứ tự từ.
- **TF-IDF:** biểu diễn văn bản dựa trên tần suất xuất hiện và mức độ đặc trưng của từ trong tập tài liệu.
- Cả hai phương pháp chuyển đổi văn bản thành **vector representation**.
- Các vector này được đưa vào thuật toán **Clustering** để nhóm các văn bản tương tự nhau.

Chú thích:

- Ma trận hiển thị sự xuất hiện (1) hoặc không xuất hiện (0) của các từ phổ biến trong mỗi câu.
- Trục hoành: Các từ phổ biến (Frequent Words).
- Trục tung: Các câu trong tập dữ liệu.
- Ô màu xanh: từ xuất hiện trong câu, ô màu đỏ: từ không xuất hiện.

7. Ứng dụng nâng cao

- Phát hiện bất thường (viễn thông, IoT).
- Tiền xử lý dữ liệu hình ảnh để giảm nhiễu.
- Chọn siêu tham số.
- Giảm số lớp để tăng mẫu huấn luyện.



Hình 5: Sơ đồ ứng dụng K-Means trong pipeline AI.

Diễn giải nhanh:

- **Raw Data → Data Preprocessing:** làm sạch, chuẩn hoá/khử chuẩn hoá.
- **Feature Engineering:** sinh `cluster_id` từ K-Means để dùng như đặc trưng.
- **K-Means Clustering:** gom các điểm tương tự vào K cụm.
- **Model Training:** dùng `cluster_id` (và feature khác) huấn luyện mô hình dự đoán.
- **Evaluation & Deployment:** đánh giá bằng các *metrics*, triển khai suy luận.
- **Ví dụ:** phân khúc khách hàng (*Customer Segmentation*).

8. Kết luận

K-Means là thuật toán **mạnh mẽ, đơn giản, dễ triển khai** nhưng có hạn chế:

- Phải chọn K trước.
- Nhạy cảm với giá trị khởi tạo.
- Không phù hợp cho cụm không lồi hoặc kích thước khác nhau.

Trong thực tế, K-Means thường được kết hợp với K-Means++, PCA hoặc làm bước tiền xử lý cho deep learning.

Bảng 3: Ưu và nhược điểm của thuật toán K-Means

Ưu điểm	Nhược điểm
<ul style="list-style-type: none">• Thuật toán đơn giản, dễ hiểu và dễ triển khai.• Tốc độ nhanh trên dữ liệu lớn với số chiều vừa phải.• Hiệu quả với cụm dạng cầu (spherical) và kích thước tương tự.• Có thể mở rộng cho nhiều ứng dụng như phân cụm khách hàng, nén ảnh, tiền xử lý cho mô hình học sâu.	<ul style="list-style-type: none">• Phải chọn số cụm K trước khi chạy.• Nhạy cảm với vị trí khởi tạo centroid (dễ rơi vào cực tiểu cục bộ).• Không phù hợp cho cụm có hình dạng phức tạp, không lồi hoặc kích thước chênh lệch.• Ảnh hưởng lớn bởi nhiều và điểm ngoại lai (outlier).

Khám phá thuật toán k-Nearest Neighbors (KNN): Từ lý thuyết đến ứng dụng thực tế

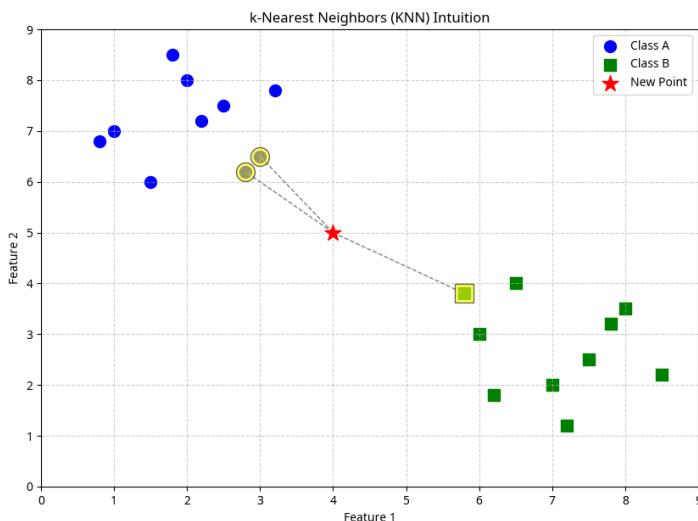
Vũ Thái Sơn

Hành trình tìm hiểu thuật toán "hồi hàng xóm" trong học máy

1. Giới thiệu: "Ngưu tầm ngưu, mã tầm mã" - Triết lý của KNN

Ngạn ngữ có câu ”**Ngưu tầm ngưu, mã tầm mã**” (Birds of a feather flock together), hay một câu nói quen thuộc khác: ”Hãy cho tôi biết bạn của bạn là ai, tôi sẽ cho bạn biết bạn là người như thế nào”. Thật vậy, trong cuộc sống, chúng ta thường có xu hướng đưa ra đánh giá hoặc nhận định dựa trên một nguyên tắc rất tự nhiên: **những gì giống nhau thường có xu hướng ở gần nhau**.

Chính cách suy luận quen thuộc này, tưởng chừng chỉ thuộc về đời sống, lại vô tình phản ánh triết lý cốt lõi của một trong những thuật toán học máy trực quan và dễ hiểu nhất: **k-Nearest Neighbors (KNN)** [1].



Hình 6: Trực quan triết lý “ngưu tầm ngưu” trong KNN: điểm mới (màu đỏ) được phân loại dựa vào các hàng xóm gần nhất.

Ý tưởng của KNN vô cùng đơn giản nhưng lại rất trực quan: để xác định bản chất của một đối tượng mới, chúng ta chỉ cần quan sát những ”người hàng xóm” gần nhất của nó trong một không gian đã biết và đưa ra dự đoán dựa trên thông tin từ họ.

Ví dụ thực tế:

- Nếu đa số hàng xóm của một email mới là ”Spam”, khả năng cao email đó cũng sẽ là ”Spam”
- Nếu những ngôi nhà có cùng diện tích, vị trí và số phòng ngủ đều có mức giá nhất định, thì một ngôi nhà mới với các đặc điểm tương tự cũng sẽ có giá xấp xỉ như vậy

2. Khái niệm cơ bản: Ba đặc trưng quan trọng của KNN

k-Nearest Neighbors (KNN) là một trong những thuật toán cơ bản trong học máy [2], được đặc trưng bởi ba tính chất quan trọng:

2.1 1. Supervised Learning (Học có giám sát)

KNN yêu cầu một tập dữ liệu huấn luyện mà trong đó mỗi điểm dữ liệu (đầu vào) đã được gán sẵn một "nhân" hoặc "giá trị" (đầu ra) chính xác. Mục tiêu của thuật toán là học cách dự đoán đầu ra cho các điểm dữ liệu mới chưa từng thấy dựa trên mối liên hệ đã học được từ dữ liệu huấn luyện.

2.2 2. Non-parametric (Phi tham số)

Thuật ngữ này không có nghĩa là mô hình không có tham số nào, mà là nó **không đưa ra bất kỳ giả định nào về dạng phân phối của dữ liệu**. Mô hình không bị giới hạn bởi một số lượng tham số cố định. Thay vào đó, độ phức tạp của mô hình (chính là toàn bộ tập dữ liệu) sẽ tăng lên khi chúng ta cung cấp thêm dữ liệu.

Điều này giúp KNN rất linh hoạt và có thể học được các ranh giới quyết định phức tạp.

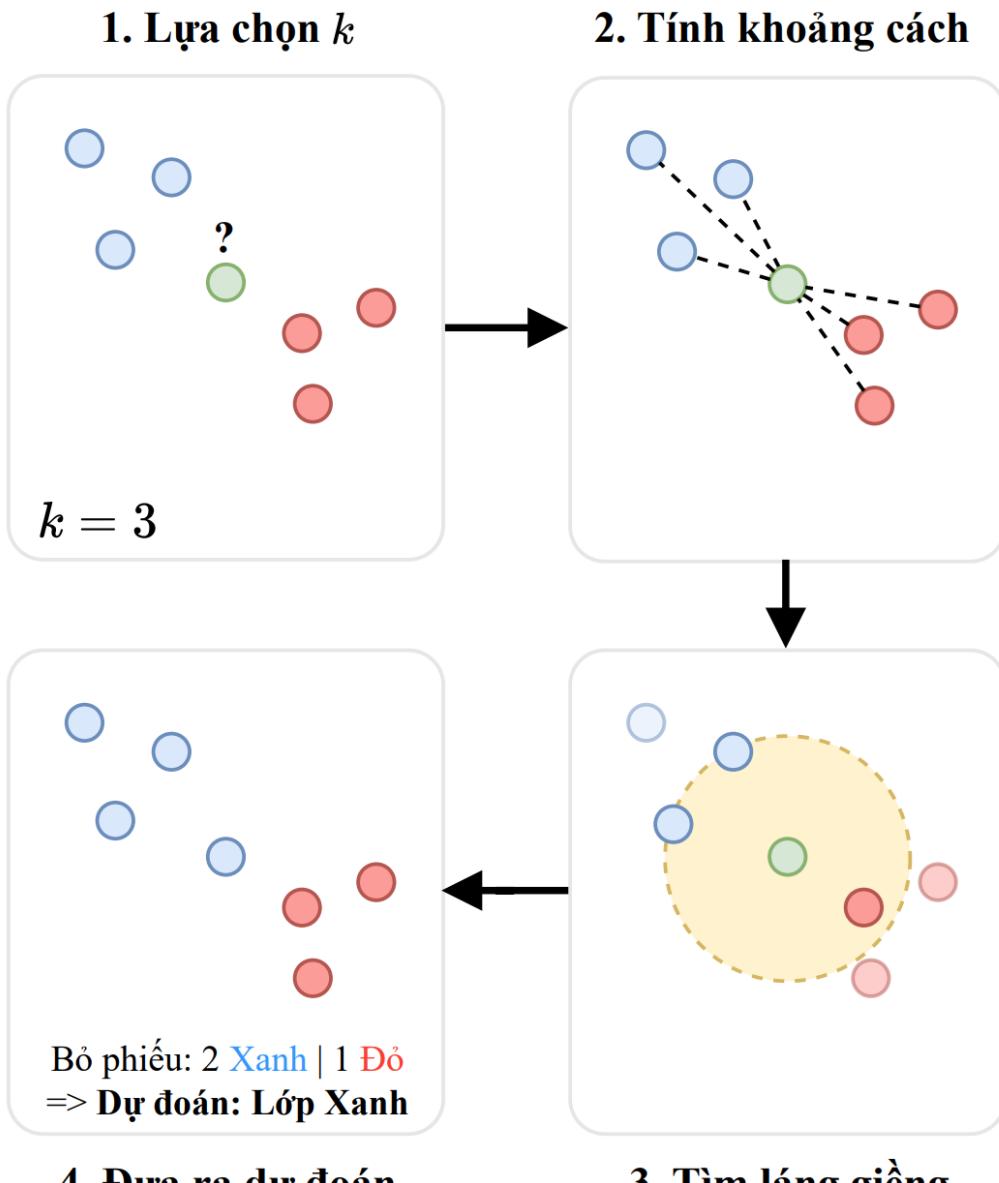
2.3 3. Instance-based / Lazy Learning (Dựa trên thực thể / Học lười)

Đây là đặc điểm độc đáo nhất của KNN. Không giống như các thuật toán khác (như hồi quy tuyến tính, mạng nơ-ron) cố gắng xây dựng một hàm tổng quát trong giai đoạn huấn luyện, KNN **không có giai đoạn huấn luyện thực sự**.

Nó chỉ đơn giản là lưu trữ toàn bộ tập dữ liệu huấn luyện. Quá trình tính toán chính, bao gồm việc tìm kiếm láng giềng và dự đoán, bị trì hoãn cho đến khi có một yêu cầu dự đoán mới.

3. Bốn bước hoạt động của KNN

KNN hoạt động theo một quy trình đơn giản gồm 4 bước chính [3]:



Hình 7: Bốn bước chính của thuật toán KNN: Chọn k , tính khoảng cách, tìm láng giềng, và ra quyết định.

3.1 Bước 1: Lựa chọn siêu tham số k

Bước đầu tiên và mang tính định hướng cho toàn bộ thuật toán là việc lựa chọn siêu tham số k - số lượng "hàng xóm" mà chúng ta sẽ tham khảo ý kiến để quyết định tính chất của điểm dữ liệu mới.

Tác động của việc chọn k :

- **k quá nhỏ ($k = 1$):** Mô hình trở nên cực kỳ linh hoạt, rất nhạy cảm với nhiễu → có thể dẫn đến overfitting
- **k quá lớn ($k = N$):** Mô hình trở nên quá "bảo thủ", bỏ qua các chi tiết quan trọng → có thể dẫn đến underfitting

Phương pháp chọn k tối ưu:

1. **Cross-Validation:** Phương pháp tiêu chuẩn và đáng tin cậy nhất
2. **Rule of Thumb:** $k = \sqrt{N}$, trong đó N là tổng số điểm trong tập dữ liệu huấn luyện
3. **Chọn k lẻ:** Để tránh trường hợp "hòa phiếu" trong bài toán phân loại

3.2 Bước 2: Tính khoảng cách

Để xác định ai là "hàng xóm" gần nhất, KNN cần một phương pháp đo "sự gần gũi".

Lưu ý quan trọng - Chuẩn hóa dữ liệu: Trước khi tính khoảng cách, có một bước tiền xử lý bắt buộc là chuẩn hóa dữ liệu. KNN cực kỳ nhạy cảm với thang đo của các đặc trưng.

Ví dụ minh họa: Khi dự đoán giá nhà dựa trên diện tích ($30-200 m^2$) và số phòng ngủ (1-5 phòng). Chênh lệch diện tích ($150-80=70$) sẽ hoàn toàn lấn át chênh lệch số phòng ngủ ($3-2=1$), làm cho đặc trưng số phòng ngủ gần như vô dụng.

Các thước đo khoảng cách phổ biến:

- **Khoảng cách Euclidean (L2):** $d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$ - Phổ biến nhất
- **Khoảng cách Manhattan (L1):** $d(p, q) = \sum_{i=1}^n |q_i - p_i|$ - Khoảng cách "đi theo đường phố"

3.3 Bước 3: Tìm k láng giềng gần nhất

Sau khi tính khoảng cách từ điểm dữ liệu mới đến mọi điểm trong tập huấn luyện:

1. Sắp xếp các khoảng cách theo thứ tự từ nhỏ đến lớn
2. Chọn ra k điểm dữ liệu ứng với k khoảng cách nhỏ nhất

3.4 Bước 4: Ra quyết định

Cách "hội đồng" k láng giềng đưa ra quyết định phụ thuộc vào bản chất của bài toán:

Đối với bài toán Phân loại - Majority Vote (Lấy ý kiến số đông):

- Điểm dữ liệu mới sẽ được gán cho lớp nào chiếm đa số trong số k láng giềng gần nhất
- **Ví dụ:** Với $k=5$, nếu có 3 phiếu "Spam" và 2 phiếu "Not Spam" \rightarrow kết quả là "Spam"

Đối với bài toán Hồi quy - Averaging (Lấy giá trị trung bình):

- Giá trị dự đoán = trung bình các giá trị mục tiêu của k láng giềng gần nhất
- **Ví dụ:** Ba ngôi nhà có giá 2 tỷ, 2.2 tỷ, 2.4 tỷ \rightarrow giá dự đoán = $(2+2.2+2.4)/3 = 2.2$ tỷ

4. Thực hành Step-by-Step: Dự đoán kết quả học tập sinh viên

Để hiểu rõ hơn về cơ chế hoạt động của thuật toán, chúng ta sẽ thực hiện tính toán từng bước trên một ví dụ cụ thể.

4.1 Bối cảnh bài toán

Giả sử chúng ta có một tập dữ liệu nhỏ về kết quả học tập của 5 sinh viên dựa trên số giờ học và số giờ ngủ mỗi ngày:

Sinh viên	Giờ học (X1)	Giờ ngủ (X2)	Kết quả	Điểm thi
p1	2	8	Trượt	4.5
p2	3	7	Trượt	5.5
p3	5	6	Đậu	7.5
p4	6	5	Đậu	8.5
p5	4	6	Đậu	7.0

Bảng 4: Dữ liệu huấn luyện về kết quả học tập của sinh viên

Bài toán: Có một sinh viên mới với Giờ học = 4, Giờ ngủ = 7. Hãy dự đoán: 1. Sinh viên này sẽ ”Đậu” hay ”Trượt”? (bài toán phân loại) 2. Điểm thi của sinh viên này sẽ là bao nhiêu? (bài toán hồi quy)

4.2 Thực hiện 4 bước KNN

Bước 1: Chọn k = 3 (số lẻ, phù hợp với tập dữ liệu nhỏ)

Bước 2: Tính khoảng cách Euclidean từ $p_{\text{new}}(4, 7)$ đến các điểm:

- Đến $p_1(2, 8)$: $d = \sqrt{(4 - 2)^2 + (7 - 8)^2} = \sqrt{4 + 1} = \sqrt{5} \approx 2.24$
- Đến $p_2(3, 7)$: $d = \sqrt{(4 - 3)^2 + (7 - 7)^2} = \sqrt{1 + 0} = \sqrt{1} = 1.00$
- Đến $p_3(5, 6)$: $d = \sqrt{(4 - 5)^2 + (7 - 6)^2} = \sqrt{1 + 1} = \sqrt{2} \approx 1.41$
- Đến $p_4(6, 5)$: $d = \sqrt{(4 - 6)^2 + (7 - 5)^2} = \sqrt{4 + 4} = \sqrt{8} \approx 2.83$
- Đến $p_5(4, 6)$: $d = \sqrt{(4 - 4)^2 + (7 - 6)^2} = \sqrt{0 + 1} = \sqrt{1} = 1.00$

Bước 3: Sắp xếp và chọn 3 láng giềng gần nhất:

Điểm	Khoảng cách	Kết quả	Điểm thi
p2	1.00	Trượt	5.5
p5	1.00	Đậu	7.0
p3	1.41	Đậu	7.5

Bảng 5: 3 láng giềng gần nhất

Bước 4: Ra quyết định

Cho bài toán phân loại:

- Số phiếu ”Đậu”: 2 (p5, p3)
- Số phiếu ”Trượt”: 1 (p2)
- Kết luận:** Sinh viên mới sẽ **Đậu**

Cho bài toán hồi quy:

- Điểm trung bình = $(5.5 + 7.0 + 7.5)/3 = 20.0/3 \approx 6.67$
- Kết luận:** Điểm thi dự đoán là **6.67**

5. Netflix và KNN: Ứng dụng thực tế trong hệ thống gợi ý

Một trong những ứng dụng nổi bật nhất của KNN trong thực tế là hệ thống gợi ý (recommendation system) của Netflix [4].

Cách Netflix áp dụng KNN:

- Thu thập dữ liệu:** Netflix ghi nhận hành vi xem phim của từng người dùng (phim nào đã xem, đánh giá, thời gian xem)
- Tìm người dùng tương tự:** Sử dụng KNN để tìm những người dùng có sở thích tương tự (những "hàng xóm" gần nhất)
- Gợi ý phim:** Recommend những bộ phim mà các "làng giềng" đã xem và đánh giá cao

Ví dụ cụ thể: Nếu bạn và 5 người dùng khác có sở thích phim tương tự đều thích phim hành động Marvel, và 4/5 người đó đã xem và đánh giá cao "Spider-Man: No Way Home", Netflix sẽ gợi ý bộ phim này cho bạn.

Cách tiếp cận này giúp Netflix tạo ra trải nghiệm cá nhân hóa, tăng thời gian xem và sự hài lòng của người dùng.

6. Code Implementation đơn giản

Để hiểu rõ hơn cách KNN hoạt động trong thực tế, đây là một implementation đơn giản bằng Python [3]:

```

1 import numpy as np
2 from collections import Counter
3
4 class SimpleKNN:
5     def __init__(self, k=3):
6         self.k = k
7
8     def fit(self, X, y):
9         """ Store training data (Lazy Learning approach) """
10        self.X_train = X
11        self.y_train = y
12
13    def euclidean_distance(self, x1, x2):
14        """ Calculate Euclidean distance between two points """
15        return np.sqrt(np.sum((x1 - x2) ** 2))
16
17    def predict(self, X):
18        """ Make predictions for multiple data points """
19        predictions = [self._predict(x) for x in X]
20        return np.array(predictions)
21
22    def _predict(self, x):
23        """ Make prediction for a single data point """

```

```
24     # Step 2: Calculate distances to all training points
25     distances = [self.euclidean_distance(x, x_train)
26                   for x_train in self.X_train]
27
28     # Step 3: Find k nearest neighbors
29     k_indices = np.argsort(distances)[:self.k]
30     k_nearest_labels = [self.y_train[i] for i in k_indices]
31
32     # Step 4: Majority voting for classification
33     most_common = Counter(k_nearest_labels).most_common(1)
34     return most_common[0][0]
35
36 # Example usage with student data
37 X_train = np.array([[2,8], [3,7], [5,6], [6,5], [4,6]])
38 y_train = np.array(['Fail', 'Fail', 'Pass', 'Pass', 'Pass'])
39
40 knn = SimpleKNN(k=3)
41 knn.fit(X_train, y_train)
42
43 # Predict for new student
44 X_new = np.array([[4, 7]])
45 prediction = knn.predict(X_new)
46 print(f"Prediction result: {prediction[0]}") # Output: Pass
```

Listing 1: Simple implementation of KNN algorithm

7. Kết luận: KNN - Đơn giản nhưng mạnh mẽ

KNN là một thuật toán học máy đặc biệt bởi sự đơn giản và trực quan của nó [2]. Không giống như các thuật toán phức tạp khác, KNN hoạt động dựa trên một nguyên lý cơ bản mà con người chúng ta vẫn thường áp dụng trong đời sống: **”đánh giá một thứ gì đó dựa trên những gì giống nó nhất”**.

7.1 Những điểm mạnh của KNN:

- **Dễ hiểu và triển khai:** Không cần kiến thức toán học phức tạp
- **Không có giả định về dữ liệu:** Có thể xử lý các ranh giới quyết định phức tạp
- **Hiệu quả với dữ liệu nhỏ:** Không cần lượng dữ liệu lớn để hoạt động tốt
- **Ứng dụng đa dạng:** Cả Classification và Regression

7.2 Những hạn chế cần lưu ý:

- **Nhạy cảm với thang đo:** Cần chuẩn hóa dữ liệu
- **Chậm với dữ liệu lớn:** Phải tính khoảng cách đến tất cả điểm
- **Nhạy cảm với nhiễu:** Đặc biệt khi k nhỏ

7.3 Hướng phát triển:

Để trở thành thành thạo với KNN và học máy, hãy:

1. **Thực hành:** Triển khai KNN trên các bộ dữ liệu thực tế
2. **Tìm hiểu sâu hơn:** Các biến thể như Weighted KNN, Distance metrics khác
3. **Ứng dụng:** Xây dựng hệ thống gợi ý đơn giản cho bản thân

KNN có thể không phải là thuật toán mạnh nhất, nhưng nó là một nền tảng tuyệt vời để hiểu về học máy. Một khi bạn thành thạo KNN, bạn đã có được tư duy cơ bản để chinh phục những thuật toán phức tạp hơn trong hành trình khám phá AI!

Tài liệu

- [1] E. Fix and J. Hodges, “Discriminatory analysis: Nonparametric discrimination,” *USAF School of Aviation Medicine*, 1951.
- [2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
- [3] S. learn developers. (2024) k-nearest neighbors algorithm. [Online]. Available: <https://scikit-learn.org/stable/modules/neighbors.html>
- [4] N. T. Blog, “How netflix uses machine learning for personalized recommendations,” *Netflix Tech Blog*, 2023. [Online]. Available: <https://netflixtechblog.com/>

PySpark: Công Cụ Xử Lý Dữ Liệu Lớn Toàn Năng

Bùi Đức Xuân

Tóm tắt nội dung

Trong thế giới số ngày nay, dữ liệu đang bùng nổ với tốc độ chóng mặt. Từ sự phát triển của ô cứng, mạng di động (từ 1G đến 5G), cho đến sự gia tăng hàng tỷ người dùng Internet và mạng xã hội, chúng ta đang sống trong kỷ nguyên của **Dữ Liệu Lớn (Big Data)**. Việc thu thập và truyền tải dữ liệu trở nên dễ dàng, nhưng xử lý và lưu trữ những bộ dữ liệu khổng lồ này lại là một thách thức lớn.

Đây là lúc PySpark tỏa sáng. Hãy cùng nhau khám phá PySpark là gì và tại sao nó lại là một công cụ mạnh mẽ và cần thiết cho bất kỳ ai làm việc với dữ liệu.



Hình 8: Big Data

1. Big Data và những thách thức

Big Data không chỉ đơn thuần là dữ liệu có kích thước lớn. Nó được định nghĩa bởi "5 V's", những đặc tính tạo nên sự phức tạp của nó:

- **Volume (Khối lượng):** Lượng dữ liệu cực lớn, có thể lên đến Terabytes hoặc Petabytes. Ví dụ, Facebook từng xử lý 25 Terabytes dữ liệu log mỗi ngày.
- **Velocity (Tốc độ):** Dữ liệu được tạo ra và cần được xử lý với tốc độ rất cao, gần như theo thời gian thực. Hãy nghĩ đến hàng triệu lượt tweet, tìm kiếm trên Google, hay giao dịch diễn ra mỗi phút.
- **Variety (Sự đa dạng):** Dữ liệu đến từ nhiều nguồn và có nhiều định dạng khác nhau: có cấu trúc (structured), bán cấu trúc (semi-structured), và phi cấu trúc (unstructured).
- **Veracity (Tính xác thực):** Chất lượng và độ tin cậy của dữ liệu. Dữ liệu không chính xác có thể dẫn đến những phân tích sai lệch và quyết định kinh doanh không hiệu quả.
- **Value (Giá trị):** Khả năng khai thác thông tin hữu ích từ dữ liệu để tạo ra giá trị, như cải thiện trải nghiệm khách hàng, tối ưu hóa chi phí, hay phát hiện gian lận.

Để giải quyết những thách thức này, các nền tảng (framework) xử lý dữ liệu lớn đã ra đời.

2. Giới thiệu Apache Spark

Trong số các framework về Big Data, **Apache Spark** nổi lên như một giải pháp hàng đầu, vượt trội hơn so với người tiền nhiệm là Hadoop ở nhiều khía cạnh.

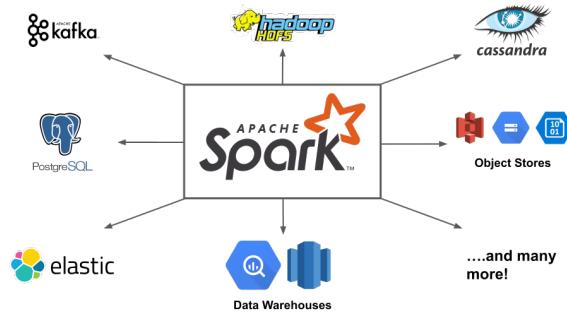
2.1 Apache Hadoop là gì?

Hadoop là một nền tảng mã nguồn mở dùng để lưu trữ và xử lý dữ liệu lớn. Hai thành phần chính của nó là:

- **HDFS (Hadoop Distributed File System)**: Một hệ thống tệp phân tán để lưu trữ dữ liệu trên nhiều máy tính.
- **MapReduce**: Một mô hình lập trình để xử lý song song các bộ dữ liệu lớn. MapReduce chia tác vụ lớn thành các nhiệm vụ nhỏ hơn (Map), sau đó tổng hợp kết quả (Reduce).

2.2 Apache Spark: Nhanh hơn và linh hoạt hơn

Spark là một công cụ xử lý dữ liệu nhanh và linh hoạt hơn Hadoop. Điểm khác biệt lớn nhất của Spark là khả năng **xử lý dữ liệu trong bộ nhớ (in-memory processing)**, giúp tăng tốc độ tính toán lên đáng kể so với cơ chế đọc/ghi trên đĩa của Hadoop MapReduce.



Hình 9: Apache Spark Logo

2.2.1 Spark RDD (Resilient Distributed Dataset)

Cốt lõi của Spark là **RDD (Resilient Distributed Dataset)**. Đây là một cấu trúc dữ liệu cơ bản của Spark, đại diện cho một tập hợp các phần tử không thể thay đổi, được phân tán trên các node của một cụm máy tính.

- **Resilient (Bền bỉ):** Có khả năng tự phục hồi khi có lỗi xảy ra.
- **Distributed (Phân tán):** Dữ liệu được chia thành các phân vùng (partitions) và lưu trữ trên nhiều máy trong cụm.
- **Dataset (Tập dữ liệu):** Là một bộ sưu tập dữ liệu (ví dụ: mảng, bảng).

Các hoạt động trên RDD được chia làm hai loại:

1. **Transformations (Phép biến đổi):** Tạo ra một RDD mới từ RDD hiện có (ví dụ: map(), filter()). Các phép biến đổi này có tính "lười biếng" (lazy), tức là chúng không thực thi ngay lập tức.

2. **Actions (Hành động):** Thực hiện tính toán trên RDD và trả về một kết quả cho driver hoặc ghi ra bộ nhớ ngoài (ví dụ: collect(), count()).

2.3 So sánh Spark và Hadoop

Tham số	Hadoop	Spark
Mục đích	Xử lý dữ liệu	Phân tích dữ liệu
Quy trình	Phân tích các lô dữ liệu lớn	Phân tích dữ liệu thời gian thực
Kiểu xử lý	Theo lô (Batch)	Thời gian thực (Real-time)
Độ trễ	Cao	Thấp
Chi phí	Ít tốn kém hơn	Tốn kém hơn (do trong bộ nhớ)
Dễ sử dụng	Phức tạp	Dễ sử dụng hơn
Ngôn ngữ	Java	Scala

Bảng 6: Bảng so sánh giữa Spark và Hadoop.

2.4 Hệ sinh thái Apache Spark

Spark không chỉ là một công cụ, mà là cả một hệ sinh thái mạnh mẽ bao gồm:

- Spark SQL:** Dành cho việc truy vấn dữ liệu có cấu trúc bằng SQL.
- Spark Streaming:** Xử lý các luồng dữ liệu theo thời gian thực.
- MLlib (Machine Learning Library):** Cung cấp các thuật toán và công cụ học máy.
- GraphX:** Dành cho việc xử lý và phân tích đồ thị.
- Spark Connect:** Một giao thức mới cho phép các ứng dụng client giao tiếp với máy chủ Spark.

3. Bắt đầu với PySpark

PySpark chính là Python API cho Apache Spark. Nó cho phép chúng ta tận dụng sự đơn giản của Python và sức mạnh của Apache Spark để xử lý dữ liệu lớn.

3.1 Thao tác với RDD trong PySpark

3.1.1 Khởi tạo SparkSession

Mọi thứ bắt đầu với SparkSession, điểm khởi đầu để lập trình với Spark.

```

1 from pyspark.sql import SparkSession
2
3 spark = SparkSession.builder \
4     .appName("PySparkExample.com") \
5     .getOrCreate()

```

Listing 2: Initialize SparkSession

3.1.2 Tạo RDD và thực hiện Actions

```

1 # Numeric data
2 num_data = [1, 2, 3, 4, 5]
3 num_rdd = spark.sparkContext.parallelize(num_data)
4
5 # Actions
6 print(num_rdd.collect())
7 # Output: [1, 2, 3, 4, 5]
8
9 print(num_rdd.take(3))
10 # Output: [1, 2, 3]
11
12 print(num_rdd.first())
13 # Output: 1
14
15 print(num_rdd.count())
16 # Output: 5

```

Listing 3: Perform Actions on an RDD

3.1.3 Transformations: map(), flatMap(), filter()

```

1 text_data = [
2     "Hello this is an example of Spark WordCount Example",
3     "we are using PySpark",
4     "PySpark is a great tool for Big Data Analysis"
5 ]
6 text_rdd = spark.sparkContext.parallelize(text_data)
7
8 # Use flatMap to split into words
9 words_rdd = text_rdd.flatMap(lambda line: line.split(" "))
10 print(words_rdd.collect())
11 # Output: ['Hello', 'this', 'is', 'an', 'example', 'of', 'Spark', ...]
12
13 # Use filter to get even numbers
14 num_rdd_filter = num_rdd.filter(lambda x: x % 2 == 0)
15 print(num_rdd_filter.collect())
16 # Output: [2, 4]

```

Listing 4: Perform Transformations on an RDD

3.2 Làm việc với DataFrame

DataFrame là một cấu trúc dữ liệu phân tán được tổ chức thành các cột có tên.

3.2.1 Đọc dữ liệu (ví dụ: file CSV)

```

1 # Read a CSV file into a DataFrame
2 # Assuming the file is named large_dataset.csv in the ./data directory
3 df = spark.read.csv("./data/large_dataset.csv", header=True, inferSchema=True)
4
5 # Show the first 20 rows
6 df.show()

```

Listing 5: Read a CSV file into a DataFrame

3.2.2 Khám phá DataFrame

```

1 # Describe the DataFrame
2 df.describe().show()

```

Listing 6: Get summary statistics of the DataFrame

3.2.3 Chọn và Lọc dữ liệu

```

1 from pyspark.sql.functions import col
2
3 # Select specific columns
4 df.select("name", "age").show()
5
6 # Filter rows based on a condition (e.g., age > 30)
7 df.filter(col("age") > 30).show()

```

Listing 7: Select and Filter a DataFrame

3.2.4 Thêm cột mới và nhóm dữ liệu

```

1 from pyspark.sql.functions import avg
2
3 # Add a new column based on a calculation
4 df_with_tax = df.withColumn("salary_plus_tax", col("salary") * 1.1)
5 df_with_tax.show()
6
7 # Group by a column and calculate an aggregate function
8 df.groupBy("occupation").agg(avg("salary")).show()

```

Listing 8: Add a new column and group the DataFrame

3.2.5 Sử dụng UDF (User-Defined Function)

```

1 from pyspark.sql.functions import udf
2 from pyspark.sql.types import StringType
3
4 # Define a UDF to categorize age
5 def categorize_age(age):
6     if age < 20:
7         return "teenager"
8     elif 20 <= age < 60:
9         return "adult"
10    else:
11        return "senior"
12
13 # Register the UDF
14 categorize_age_udf = udf(categorize_age, StringType())
15
16 # Apply the UDF to create a new column
17 df.withColumn("age_category", categorize_age_udf(col("age"))) \
18     .select("age", "age_category") \
19     .show()

```

Listing 9: Using a UDF to categorize age

4. Sức mạnh của Spark SQL

Spark SQL cho phép bạn chạy các truy vấn SQL trực tiếp trên DataFrame. Đầu tiên, hãy tạo một "temporary view" từ DataFrame.

```
1 df.createOrReplaceTempView("people")
```

Listing 10: Create a Temporary View

Bây giờ bạn có thể dùng spark.sql() để truy vấn.

```
1 -- Equivalent to df.select("*")
2 SELECT * FROM people
```

Listing 11: Basic SQL Query

```
1 -- Equivalent to df.filter(col("age") > 30)
2 SELECT * FROM people WHERE age > 30
```

Listing 12: Filter with WHERE clause

```
1 -- Equivalent to df.groupBy("occupation").agg(avg("salary"))
2 SELECT occupation, AVG(salary) as avg_salary
3 FROM people
4 GROUP BY occupation
```

Listing 13: Group with GROUP BY

5. Giới thiệu PySpark MLlib

PySpark MLlib là thư viện học máy của Spark, được thiết kế để chạy trên các bộ dữ liệu lớn một cách phân tán.

5.1 So sánh MLlib và Scikit-learn

Tác vụ	Scikit-learn	PySpark MLlib
Chia Train/Test	train_test_split	<DataFrame>.randomSplit
Mã hóa	LabelEncoder	StringIndexer
Chuẩn hóa	StandardScaler	StandardScaler
Mô hình hóa	LinearRegression	LinearRegressionWithSGD
Đánh giá	accuracy_score	MulticlassMetrics

Bảng 7: So sánh quy trình làm việc giữa MLlib và Scikit-learn.

Dòng chảy công việc trong MLlib thường bao gồm việc xây dựng một Pipeline, kết hợp các bước xử lý đặc trưng (feature engineering) và huấn luyện mô hình thành một quy trình duy nhất.

6. Kết luận

PySpark là một cầu nối mạnh mẽ, kết hợp sự linh hoạt của Python với khả năng xử lý phân tán hiệu năng cao của Apache Spark. Cho dù bạn đang xử lý các lô dữ liệu khổng lồ, phân tích luồng

dữ liệu thời gian thực, hay xây dựng các mô hình học máy phức tạp, PySpark đều cung cấp một bộ công cụ toàn diện và mạnh mẽ. Việc nắm vững PySpark sẽ mở ra cho bạn cánh cửa để giải quyết những bài toán dữ liệu lớn nhất và khai thác những giá trị tiềm ẩn bên trong chúng.