Đề thi tuyển sinh
Trại hè Toán học và Ứng dụng 2021

—PHẦN TỰ NGHIÊN CỨU—

Cấu trúc đề thi và hướng dẫn

Đây là phần Đề thi tuyển sinh phần Câu hỏi tự nghiên cứu cho trại hè PiMA 2021. Đề thi gồm nhiều phần câu hỏi xoay quanh chủ đề "Phân cụm K-means" trong Khoa học Dữ liệu, và đòi hỏi từ bạn kỹ năng tự tìm kiếm thông tin và tiếp thu kiến thức. Bạn có thể trình bày câu trả lời của mình bằng nhiều hình thức tùy ý, nhưng hãy đảm bảo đủ ý và không quá dài dòng. Một cách tìm kiếm hiệu quả là gõ tên các khái niệm vào thanh tìm kiếm của trang web www.google.com. Ngoài ra, phần phụ lục cuối đề thi sẽ hỗ trợ bạn về nền tảng để lập trình Python.

Thuật ngữ tiếng Anh

Một số thuật ngữ chúng mình sẽ dùng từ tiếng Anh thay cho tiếng Việt ở một hoặc nhiều chỗ trong file đề bài này:

STT	Tiếng Việt	Tiếng Anh	STT	Tiếng Việt	Tiếng Anh
1	Phân cụm	Clustering	2	Cụm	Cluster
3	Tập dữ liệu	Dataset	4	Điểm dữ liệu	Data point
5	Trọng tâm	Centroid	6	Nhãn	Label
7	Đầu vào	Input	8	Đầu ra	Output
9	Cho ra (kết quả)	Return	10	Chú giải hoá (code)	Comment
11	Ngừng chú giải hoá	Uncomment			

Phần câu hỏi chính - Phân cụm K-means

Thuật toán phân cụm **K-means** là một thuật toán phổ biến đối với người nhập môn Khoa học Dữ liệu. Bạn hãy tìm hiểu thật nhiều về thuật toán này để trả lời các câu hỏi sau.

Câu hỏi lý thuyết

Trả lời các câu hỏi sau và giải thích ngắn gọn câu trả lời của bạn.

LT1 Mô tả bài toán mà thuật toán này muốn giải. **Gợi ý:** trả lời các câu hỏi nhỏ:

- Dữ liệu đầu vào (**input**), đầu ra (**output**) của thuật toán này là gì?
- Thuật toán này muốn tối đa / tối thiểu hoá đại lượng gì?

LT2 Mô tả thuật toán K-means bằng ngôn ngữ của bạn. **Gợi ý:** trả lời các câu hỏi nhỏ:

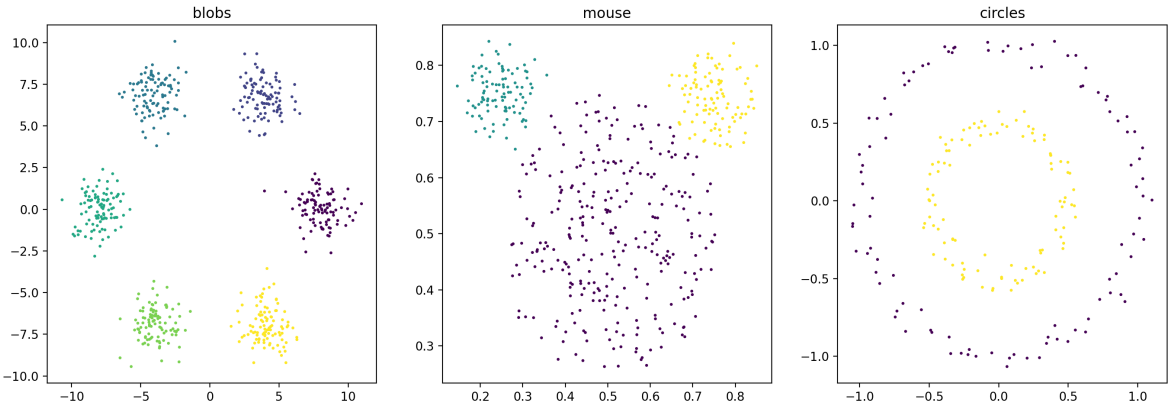
- Viết ra các bước của thuật toán theo trình tự, dưới dạng danh sách hoặc các sơ đồ như sơ đồ khối.
- Giải thích mục đích từng bước trong thuật toán một cách ngắn gọn.

Mở trang web sau: K-means visualization by N. Harris. Bạn có thể nhấn vào từng nút bấm trên trang web để hiểu cách K-means được thể hiện bằng đồ hoạ. Sau khi đã hiểu cách trang web hoạt động, trả lời các câu hỏi sau:

LT3 Đối với tập dữ liệu (**dataset**) "Gaussian Mixture" và số cụm là 3, tìm 1 cách chọn 3 trọng tâm ban đầu sao cho output của thuật toán không phải là tối ưu. Giải thích tại sao lại có hiện tượng này.

LT4 Hãy thử đưa ra một thuật toán, gọi là "Centroids Smart Select", để chọn các trọng tâm ban đầu sao cho tăng khả năng tìm chính xác các cụm. Mô tả input, output và các trình tự các bước của thuật toán đó. **Gợi ý:** Bạn có thể tìm thấy vài thuật toán như vậy trên mạng.

LT5 Các biểu đồ sau thể hiện một số datasets gồm các điểm trong mặt phẳng $\mathbb{R}^2$. Các điểm cùng một cụm được tô cùng màu. Với mỗi datasets, dự đoán xem thuật toán K-means, với số cụm đúng được cho trước, có thể tìm chính xác các cụm được không. Giải thích tại sao. **Gợi ý:** Hãy thử thuật toán này với các dataset còn lại ngoài "Gaussian Mixture" trên trang web của N. Harris và quan sát kết quả.



Câu hỏi lập trình

Lưu ý: Nếu bạn chưa có kinh nghiệm lập trình, có thể làm phần này cuối cùng hoặc bỏ qua nếu thời gian không cho phép. Có rất nhiều trại sinh PiMA chưa có kinh nghiệm lập trình tại thời điểm tham gia trại.

Trong folder này, bạn sẽ tìm thấy các file sau:

1. File đề bài này.
2. `my_kmeans.py` - chứa một đoạn code bằng Python triển khai thuật toán K-means. Trong các câu hỏi TH1 và TH2 bạn sẽ phải hoàn thành file này.
3. `main.py` - chứa một đoạn code bằng Python để kiểm tra đoạn code bạn viết ở file `my_kmeans.py`. Chạy file này khi bạn muốn kiểm tra code của mình.
4. `blobs.csv` - dataset "blobs", `mouse.csv` - dataset "mouse', `circles.csv` - dataset "circles", `emails.csv` - dataset "emails", `senate.csv` - dataset "lịch sử bỏ phiếu của Thượng viện Mỹ khóa 114".

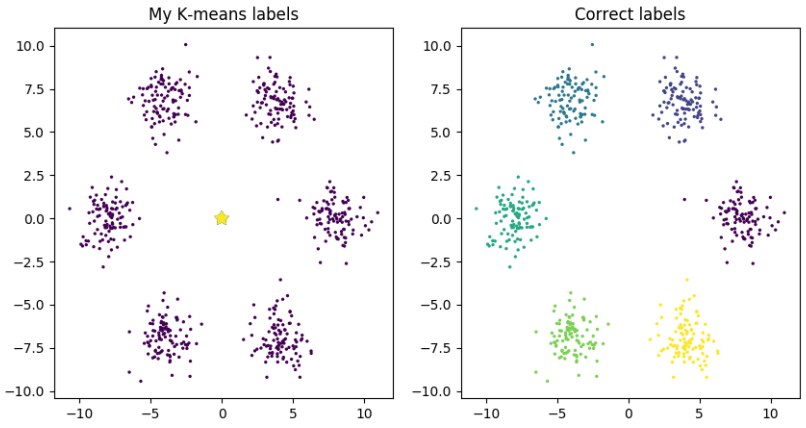
Hãy **uncomment** dòng số 72 ở `main.py` bằng cách xoá dấu # ở đầu dòng:

```
# kmeans_test('blobs', smart=False)   # Uncomment de test cau hoi TH1
```

thành

```
kmeans_test('blobs', smart=False)   # Uncomment de test cau hoi TH1
```

Chạy file `main.py`, bạn sẽ thấy kết quả như sau:



Biểu đồ bên trái thể hiện output của thuật toán K-means trong file `my_kmeans.py` với các điểm data cùng cluster được tô cùng màu. Biểu tượng ngôi sao thể hiện vị trí trọng tâm (centroid) của các cluster.

Biểu đồ bên phải thể hiện các điểm data với màu tương ứng với cluster chính xác.

Vì thuật toán K-means **chưa được triển khai đúng**, cụ thể là các hàm `find_nearest_centroid`, `find_new_centroids` (và `centroids_smart_select` cho phiên bản cải tiến) trong file `my_kmeans.py`, output chỉ có một cluster, so với 6 cluster chính xác.

Để **comment** dòng này lại, chỉ cần thêm dấu # lại vào đầu dòng. Khi chạy file, các dòng được comment (tức là có dấu # đầu dòng) sẽ không được tính là một phần của code.
Lưu ý: Chỉ xoá dấu # đầu dòng, hãy giữ nguyên dấu # thứ hai.

Qua các câu hỏi sau, bạn sẽ điền code của mình vào để hoàn thiện thuật toán K-means.

Cảnh báo: Bạn không được phép thay đổi bất kỳ chỗ nào trong các file, trừ ba hàm `find_nearest_centroid`, `find_new_centroids` và `centroids_smart_select` trong `my_kmeans.py`

TH1 Hoàn thành đoạn code triển khai K-means bằng cách điền nội dung vào các hàm `find_nearest_centroid` và `find_new_centroids` trong `my_kmeans.py`.
Uncomment dòng `kmeans_test('blobs', smart=False)` và chạy file `main.py` để kiểm tra lỗi và thử output của đoạn code K-means của bạn.

Sau khi hoàn thành câu TH1 hãy comment dòng này lại.

Uncomment dòng `kmeans_testblobs(100, smart=False)` và chạy `main.py`. Dòng này sẽ áp dụng thuật toán K-means mà bạn triển khai lên dataset "blobs" 100 lần, mỗi lần tính một số điểm (score) bằng số điểm dữ liệu được phân đúng cụm. Dòng cuối sẽ hiển thị score trung bình của 100 lần chạy.

TH2 Hoàn thành việc triển khai thuật toán "Centroid Smart Select" mà bạn nghĩ ra trong câu hỏi LT4 bằng cách điền nội dung vào hàm `centroids_smart_select` trong file `my_kmeans.py`. Đổi dòng `kmeans_testblobs(100, smart=False)` thành `kmeans_testblobs(100, smart=True)` và chạy `main.py` để thử output của thuật toán `centroids_smart_select` của bạn. So sánh output khi `smart=True` so với khi `smart=False` và giải thích nguyên nhân cho sự khác nhau.

Sau khi hoàn thành TH2 hãy comment dòng này lại.

TH3 Uncomment dòng `kmeans_test('mouse', smart=False)`, chạy file `main.py` và mô tả output. Đổi dòng này thành `kmeans_test('mouse', smart=True)`, chạy và so sánh output với khi `smart=False`. Lặp lại 2 bước trên với dataset `'circles'`. Thuật toán `centroid_smart_select` có cải thiện output của K-means trên các dataset "mouse" và "circles" không?

Câu hỏi nâng cao

Với mỗi câu hỏi sau, bạn sẽ được hỏi liệu thuật toán phân cụm K-means có thể chia dataset tương ứng thành các cụm theo yêu cầu không. Giải thích ngắn gọn liệu bạn nghĩ K-means có thể được áp dụng vào mỗi trường hợp. Nếu không áp dụng được, hãy đề xuất một biến thể / mở rộng của K-means mà bạn nghĩ có thể áp dụng được cho bài toán. Bạn có thể nộp code của mình nếu có (**không bắt buộc**). Ở phần này, bạn được phép dùng các phiên bản K-means có trong các thư viện online, ví dụ như `sklearn.cluster.KMeans`.

NC4 File `senate.csv`[1] chứa lịch sử bỏ phiếu của các thượng nghị sĩ Mỹ khoá 114 đối với các dự luật. Ví dụ 3 dòng đầu của dataset:

[1] Bạn cũng có thể download file này tại đây

```
name,party,state,00001,00004,00005,00006,00007,00008,00009,00010,00020,00026,00032,00038,00039,00044,00047
Alexander,R,TN,0,1,1,1,1,0,0,1,1,1,0,0,0,0,0
Ayotte,R,NH,0,1,1,1,1,0,0,1,0,1,0,1,0,1,0
```

Dòng đầu chứa tên các biến `name` (tên), `party` (đảng), `state` (bang) và lịch sử bỏ phiếu cho các dự luật có mã từ `00001` đến `00047`. Từ dòng thứ hai trở đi, mỗi dòng chứa tên, đảng, bang và lịch sử bỏ phiếu của từng thượng nghĩ sĩ. Số `1` là Thuận, `0` là Chống và `0.5` là phiếu Trắng. Ví dụ, dòng thứ ba cho biết: Bà Ayotte là thượng nghị sĩ đảng Republican (R) từ bang New Hampshire (NH), bà bỏ phiếu Thuận cho các dự luật có mã 4, 5, 6, 7, 10, 26, 38, 44 và bỏ phiếu Chống cho các dự luật còn lại.

Câu hỏi: Liệu có thể dùng K-means để chia các thượng nghị sĩ thành 2 cụm sao cho các thành viên cùng đảng vào chung cụm nhiều nhất có thể? Hơn nữa, liệu có thể dùng K-means để tìm ra các phe phái nhỏ trong từng đảng với lịch sử bỏ phiếu tương tự nhau? Nếu không, bạn có thể đề xuất một cải biến của K-means để giải hai bài toán trên không?

NC5 Công ty PiMA Corp gồm 100 nhân viên được chia làm 4 phòng ban: Nghiên cứu, Kỹ sư, Marketing và Nhân sự. Mỗi nhân viên có xu hướng trao đổi nhiều emails với người cùng phòng mình hơn người khác phòng. File `emails.csv` chứa một ma trận $M \in \mathbb{R}^{100\times 100}$, trong đó M_{ij} là số emails nhân viên i gửi tới nhân viên j trong vòng 1 tháng. **Lưu ý:** M_{ij} có thể khác M_{ji}.

Câu hỏi: Liệu có thể dùng K-means để tìm ra ai thuộc phòng ban nào? Nếu không, bạn có thể đề xuất một cải biến của K-means để giải bài toán trên không?

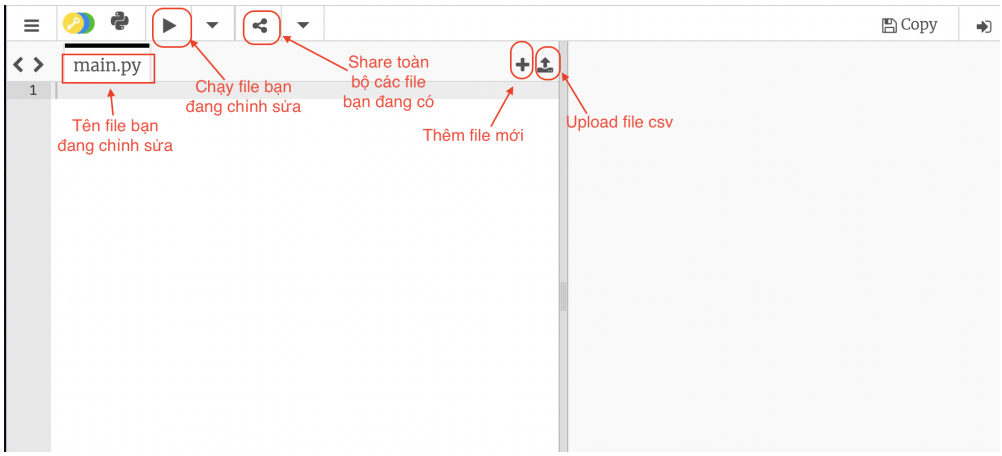
A. Hướng dẫn thêm về phần Lập trình

Bạn có thể tìm hiểu thêm về Python tại Laptrinh.io, Tutorialspoint, hoặc sử dụng Google.

Nếu bạn không có Python trong máy tính, hoặc bản Python của bạn không có các thư viện `numpy`, `scipy`, `sklearn`, v.v. thì có thể dùng trang web Trinket.io. Bạn có thể dùng bất cứ nền tảng chạy code Python nào tuỳ thích.

Cách sử dụng Trinket.io

Vào link Trinket.io, bạn sẽ thấy các nút thao tác sau:



Copy và paste toàn bộ code trong file `main.py` vào, sau đó tạo thêm một file mới với tên `my_kmeans.py`. Copy và paste toàn bộ code trong file `my_kmeans.py` trong folder này vào file vừa tạo. Upload các file `blobs.csv`, `mouse.csv`, `circles.csv` bằng nút upload. Bạn sẽ thấy như sau:

Bấm vào nút chạy code (hình ▶) để quan sát kết quả.

—HẾT—