

## Chương 4: Unsupervised Learning (Học tập không giám sát)

Nhóm 4:

19110311 - Nguyễn Ngô Trung Hậu.

19110257 - Trần Bửu Ân.

19110281 - Phùng Thị Diệp.

19110315 - Trịnh Ngọc Hiền.

19110327 - Nguyễn Thị Cẩm Hương.

2021

# Nội dung

**4.1 Giới thiệu.**

**4.2 Rủi ro và mất mát trong việc học không giám sát (Risk and Loss in Unsupervised Learning).**

**4.3 Thuật toán tối đa hóa kỳ vọng (EM).**

**4.4 Hàm phân phối thực nghiệm và ước lượng mật độ(Empirical Distribution and Density Estimation)**

**4.5 Phân Cụm Thông Qua Các Mô Hình Hỗn Hợp.**

4.5.1 Mô hình hỗn hợp.

4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp.

# Nội dung

## **4.6 Phân cụm thông qua lượng tử hóa vectơ.**

4.6.1 K- Phương tiện.

4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục.

## **4.7 Phân cụm phân cấp (Hierarchical Clustering)**

4.7.1 Chiến lược hợp nhất (agglomerative).

4.7.2 Khoảng cách giữa hai cụm.

4.7.3 Chiến lược phân chia (divisive).

4.7.4 Thuật toán Lance - Williams.

## **4.8 Phân tích thành phần chính (PCA).**

4.8.1 Động lực: Trục chính của một Ellipsoid.

4.8.2 PCA và Phân tích Giá trị Số ít (SVD).

## 4.1 Giới thiệu

Học không có giám sát là một phương pháp của ngành máy học nhằm tìm ra một mô hình mà phù hợp với các quan sát. Khác với học có giám sát, học không giám sát có đầu ra đúng tương ứng cho mỗi đầu vào là không biết trước.

Học không có giám sát thường xem các đối tượng đầu vào như là một tập các biến ngẫu nhiên. Sau đó, một mô hình mật độ kết hợp sẽ được xây dựng cho tập dữ liệu đó.

Học không có giám sát có thể được dùng kết hợp với suy luận Bayes để cho ra xác suất có điều kiện (nghĩa là học có giám sát) cho bất kì biến ngẫu nhiên nào khi biết trước các biến khác.

## 4.1 Giới thiệu

Học không có giám sát rất hữu ích cho việc nén dữ liệu. Về cơ bản, mọi giải thuật nén dữ liệu là dựa vào một phân bố xác suất trên một tập đầu vào một cách tường minh hoặc không tường minh.

Một dạng khác của học không có giám sát là phân nhóm dữ liệu, nó đôi khi không mang tính xác suất.

## 4.1 Giới thiệu

Trong **mục 4.2**, sẽ thiết lập một khuôn khổ cho việc học tập không giám sát, tương tự như khung được thiết lập cho học có giám sát (**mục 2.3**). Đó là, công thức học tập không giám sát trong điều kiện giảm thiểu rủi ro và tổn thất, nhưng sẽ **liên quan đến rủi ro chéo entropy, thay vì rủi ro sai số bình thường**.

Trong **mục 4.3**, sẽ giới thiệu về **Thuật toán Kỳ vọng - Tối đa hóa (EM)** như một thuật toán hữu ích để tối đa hóa các hàm khả năng khi không thể dễ dàng tìm thấy giải pháp của chúng trong dạng đóng.

## 4.1 Giới thiệu

Trong **mục 4.4**, sẽ **khái niệm hóa về phân phối thực nghiệm** và giải thích cách đưa ra ước tính về xác suất cơ bản của hàm mật độ của dữ liệu bằng cách sử dụng công cụ ước tính mật độ hạt nhân.

Trong **mục 4.5**, **hình thành vấn đề phân cụm** theo mô hình hỗn hợp.

Trong **mục 4.6**, nói về một cách **tiếp cận heuristic** hơn để **phân cụm**, nơi dữ liệu **được nhóm theo một số "trung tâm cụm"** nhất định, có vị trí được tìm thấy bằng cách **giải quyết một vấn đề tối ưu hóa**.

## 4.1 Giới thiệu

Trong **mục 4.7**, nói về **mô tả cách các cụm có thể được xây dựng theo cách phân cấp**.

Cuối cùng, trong **mục 4.8**, nói về kỹ thuật học tập không giám sát được gọi là **Phân tích Thành phần Chính (PCA)**, đây là một công cụ quan trọng để giảm bớt các chiều của dữ liệu.



## 4.2 Rủi ro và mất mát trong học không giám sát

Giả sử để là tìm hiểu hàm mật độ xác suất  $f$  chưa biết của  $X$  dựa trên một kết quả  $\tau := \{x_1, \dots, x_n\}$  của dữ liệu huấn luyện  $T$ . Chúng ta có thể dùng các dòng lý luận ở học có giám sát được thảo luận trong **mục 2.3-2.5**

Tương tự như học có giám sát, để tìm một hàm  $g$ , thứ mà bây giờ là một xác suất mật độ (liên tục hoặc rời rạc), gần đúng nhất với hàm mật độ xác suất  $f$  về mặt giảm thiểu rủi ro

$$\ell(g) := \mathbb{E} \text{Loss}(f(X), g(X)), \quad (4.1)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

trong đó Loss là một hàm mất mát. Ta gặp lại rủi ro Kullback – Leibler trong (2.27)

$$\ell(g) := \mathbb{E} \ln \frac{f(X)}{g(X)} = \mathbb{E} \ln f(X) - \mathbb{E} \ln g(X) \quad (4.2)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Nếu  $\mathcal{G}$  là một lớp hàm chứa  $f$ , thì việc giảm thiểu rủi ro Kullback – Leibler qua  $\mathcal{G}$  sẽ mang lại bộ thu nhỏ (đúng)  $f$ . Qua đó, việc giảm thiểu (4.2) phụ thuộc vào  $f$ , thường không được dùng đến.

Tuy nhiên,  $\mathbb{E} \ln f(X)$  không phụ thuộc vào  $g$ , nó không đóng vai trò gì trong việc giảm thiểu rủi ro Kullback-Leibler. Qua đó, việc loại bỏ số hạng này, chúng ta thu được rủi ro chéo entropy (đối với  $X$  rời rạc thay thế tích phân bằng một tổng):

$$\ell(g) := -\mathbb{E} \ln g(X) = - \int f(x) \ln g(x) dx \quad (4.3)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

|  |  |
|--|--|
| $\mathbf{x}$   | Fixed feature vector.  |
| $\mathbf{X}$   | Random feature vector.   |
| $f(\mathbf{x})$                                      | Pdf of $\mathbf{X}$ evaluated at the point $\mathbf{x}$ .  |
| $\tau$ or $\tau_n$                                   | Fixed training data $\{\mathbf{x}_i, i = 1, \dots, n\}$ .  |
| $\mathcal{T}$ or $\mathcal{T}_n$                     | Random training data $\{\mathbf{X}_i, i = 1, \dots, n\}$ .   |
| $g$  | Approximation of the pdf $f$ .   |
| $\text{Loss}(f(\mathbf{x}), g(\mathbf{x}))$          | Loss incurred when approximating $f(\mathbf{x})$ with $g(\mathbf{x})$ .  |
| $\ell(g)$  | Risk for approximation function $g$ ; that is, $\mathbb{E} \text{Loss}(f(\mathbf{X}), g(\mathbf{X}))$ .  |
| $g^{\mathcal{G}}$                                    | Optimal approximation function in function class $\mathcal{G}$ ; that is, $\text{argmin}_{g \in \mathcal{G}} \ell(g)$ .  |
| $\ell_{\tau}(g)$                                     | Training loss for approximation function (guess) $g$ ; that is, the sample average estimate of $\ell(g)$ based on a fixed training sample $\tau$ .   |
| $\ell_{\mathcal{T}}(g)$                              | The same as $\ell_{\tau}(g)$ , but now for a random training sample $\mathcal{T}$ .  |
| $g_{\tau}^{\mathcal{G}}$ or $g_{\tau}$               | The <i>learner</i> : $\text{argmin}_{g \in \mathcal{G}} \ell_{\tau}(g)$ . That is, the optimal approximation function based on a fixed training set $\tau$ and function class $\mathcal{G}$ . We suppress the superscript $\mathcal{G}$ if the function class is implicit. |
| $g_{\mathcal{T}}^{\mathcal{G}}$ or $g_{\mathcal{T}}$ | The learner for a random training set $\mathcal{T}$ .  |

Hình: Bảng 4.1: Tóm tắt định nghĩa của học không giám sát

## 4.2 Rủi ro và mất mát trong học không giám sát

Do đó, việc giảm thiểu rủi ro chéo entropy (4.3) trên tất cả  $g \in \mathcal{G}$ , một lần nữa cho bộ giảm thiểu  $f$ , với điều kiện là  $f \in \mathcal{G}$ .

Tuy nhiên, việc giải quyết (4.3) nói chung cũng không khả thi, vì nó vẫn phụ thuộc vào  $f$ . Thay vào đó, chúng tôi tìm cách giảm thiểu tổn thất do tạo entropy chéo:

$$\ell_r(g) := \frac{1}{n} \sum_{i=1}^n \text{Loss}(f(x_i), g(x_i)) = -\frac{1}{n} \sum_{i=1}^n \ln g(x_i) \quad (4.4)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

trên lớp của các hàm  $\mathcal{G}$ , trong đó,  $\tau := \{x_1, \dots, x_n\}$  là một mẫu biến ngẫu nhiên từ  $f$ . Việc tối ưu hóa này có thể thực hiện được mà không cần biết  $f$  và tương đương với việc giải quyết bài toán tối đa hóa

$$\max_{g \in \mathcal{G}} \sum_{i=1}^n \ln g(x_i)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Để thiết lập quy trình học, bước quan trọng là chọn một lớp chức năng  $\mathcal{G}$  phù hợp để tối ưu hóa. Tiếp theo là tham số hóa  $g$  với một tham số  $\theta$ . Gọi  $\mathcal{G}$  là lớp của các hàm  $\{g(\cdot|\theta), \theta \in \Theta\}$  đối với một tập tham số  $p$ -chiều  $\Theta$ .

Đối với phần còn lại của **mục 4.2**, chúng tôi sẽ sử dụng lớp hàm này, cũng như nguy cơ entropy chéo.

## 4.2 Rủi ro và mất mát trong học không giám sát

Hàm  $\theta \mapsto g(x|\theta)$  được gọi là *hàm khả năng xảy ra*. Nó cho khả năng xuất hiện của vectơ đặc trưng quan sát  $x$  dưới  $g(\cdot|\theta)$ , như một hàm của tham số  $\theta$ .

Lôgarit tự nhiên của hàm khả năng được gọi là hàm log - hàm khả năng xảy ra và gradient của nó đối với  $\theta$  được gọi là hàm điểm, ký hiệu là  $S(x|\theta)$ ; đó là,

$$S(x|\theta) := \frac{\partial \ln g(x|\theta)}{\partial \theta} = \frac{\frac{\partial g(x|\theta)}{\partial \theta}}{g(x|\theta)}. \quad (4.6)$$



## 4.2 Rủi ro và mất mát trong học không giám sát

Điểm ngẫu nhiên  $S(X|\theta)$ , với  $X \sim g(\cdot|\theta)$ , được quan tâm đặc biệt. Trong nhiều trường hợp, kỳ vọng của nó *bằng vectơ không*; cụ thể là

$$\begin{aligned}\mathbb{E}_{\theta}S(X|\theta) &= \int \frac{\frac{\partial g(\mathbf{x}|\theta)}{\partial \theta}}{g(\mathbf{x}|\theta)} g(\mathbf{x}|\theta) d\mathbf{x} \\ &= \int \frac{\partial g(\mathbf{x}|\theta)}{\partial \theta} d\mathbf{x} = \frac{\partial \int g(\mathbf{x}|\theta) d\mathbf{x}}{\partial \theta} = \frac{\partial 1}{\partial \theta} = \mathbf{0},\end{aligned}\tag{4.7}$$

## 4.2 Rủi ro và mất mát trong học không giám sát

### Lưu ý:

**Điều quan trọng là phải xem liệu các kỳ vọng được coi là  $X \sim g(\cdot|\theta)$  hay  $X \sim f$ . Chúng tôi sử dụng các ký hiệu kỳ vọng  $\mathbb{E}_\theta$  và  $\mathbb{E}$  để phân biệt hai trường hợp.**

Ma trận hiệp phương sai của điểm ngẫu nhiên  $S(X|\theta)$  được gọi là *ma trận thông tin Fisher*, chúng ta ký hiệu là  $F$  hoặc  $F(\theta)$  để thể hiện sự phụ thuộc của nó vào  $\theta$ . Vì điểm số dự kiến là 0, chúng tôi đã

$$F(\theta) = \mathbb{E}_\theta[S(X|\theta)S(X|\theta)^T] \quad (4.8)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Ma trận liên quan là ma trận Hessian kỳ vọng của  $-\ln g(X|\theta)$ :

$$\mathbf{H}(\theta) := \mathbb{E} \left[ -\frac{\partial \mathbf{S}(X; \theta)}{\partial \theta} \right] = -\mathbb{E} \begin{bmatrix} \frac{\partial^2 \ln g(X|\theta)}{\partial^2 \theta_1} & \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_1 \partial \theta_2} & \dots & \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_1 \partial \theta_p} \\ \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \ln g(X|\theta)}{\partial^2 \theta_2} & \dots & \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_2 \partial \theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_p \partial \theta_1} & \frac{\partial^2 \ln g(X|\theta)}{\partial \theta_p \partial \theta_2} & \dots & \frac{\partial^2 \ln g(X|\theta)}{\partial^2 \theta_p} \end{bmatrix}. \quad (4.9)$$

Lưu ý rằng kỳ vọng này là  $X \sim f$ . Nó chỉ ra rằng nếu  $f = g(\cdot|\theta)$ , thì hai ma trận giống nhau; đó là,

$$F(\theta) = H(\theta), \quad (4.10)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Với điều kiện là thứ tự của sự khác biệt và tích hợp (kỳ vọng) có thể hoán đổi. Kết quả này được gọi là *ma trận bình đẳng thông tin*.

Ma trận  $F(\theta)$  và  $H(\theta)$  đóng vai trò quan trọng trong việc tính gần đúng nguy cơ chéo entropy với  $n$  lớn. Để thiết lập nó, ta đặt  $g^{\mathcal{G}} = g(\cdot|\theta^*)$  là bộ giảm thiểu rủi ro chéo entropy

$$r(\theta) := -\mathbb{E} \ln g(X|\theta)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Ta giả định rằng  $r$ , là một hàm của  $\theta$ , được xử lý tốt; đặc biệt, trong vùng lân cận của  $\theta^*$ , là lồi nghiêm ngặt và có thể phân biệt hai lần liên tục (điều này đúng, ví dụ, nếu  $g$  là mật độ Gauss). Theo đó  $\theta^*$  là một gốc của  $\mathbb{E}S(X|\theta)$ , bởi vì

$$0 = \frac{\partial r(\theta^*)}{\partial \theta} = -\frac{\partial \mathbb{E} \ln g(X|\theta^*)}{\partial \theta} = -\mathbb{E} \frac{\partial \ln g(X|\theta^*)}{\partial \theta} = -\mathbb{E}S(X|\theta^*),$$

Do điều kiện có thể hoán đổi được, nhờ đó,  $H(\theta)$  là ma trận Hessian của  $r$ . Gọi  $g(\cdot|\hat{\theta}_n)$  là giá trị tối thiểu của tổn thất đào tạo

$$r_{\mathcal{T}_n} := -\frac{1}{n} \sum_{i=1}^n \ln g(X_i|\theta)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

Với  $\mathcal{T}_n = \{X_1, \dots, X_n\}$  là một tập hợp huấn luyện ngẫu nhiên. Gọi  $r^*$  là rủi ro chéo entropy nhỏ nhất có thể, được tính trên tất cả các hàm; rõ ràng,  $r^* = -\mathbb{E} \ln f(X)$ , trong đó  $X \sim f$ .

Tương tự như trường hợp học có giám sát, chúng ta có thể phân rã rủi ro tổng quát hóa,  $\ell(g(\cdot | \hat{\theta}_n)) = r(\hat{\theta}_n)$ , thành

$$r(\hat{\theta}_n) = r^* + \underbrace{r(\theta^*) - r^*}_{\text{approx. error}} + \underbrace{r(\hat{\theta}_n) - r(\theta^*)}_{\text{statistical error}} = r(\theta^*) - \mathbb{E} \ln \frac{g(X | \theta^*)}{g(X | \hat{\theta}_n)}.$$

## 4.2 Rủi ro và mất mát trong học không giám sát

### Định lý 4.1: Xấp xỉ Rủi ro chéo Entropy

Tiệm cận với ( $n \rightarrow \infty$ ) là

$$\mathbb{E}r(\hat{\theta}_n) - r(\theta^*) \simeq \text{tr}(F(\theta^*)H^{-1}(\theta^*)) / (2n), \quad (4.11)$$

trong đó

$$r(\theta^*) \simeq \mathbb{E}r_{\mathcal{T}_n}(\hat{\theta}_n) + \text{tr}(F(\theta^*)H^{-1}(\theta^*)) / (2n). \quad (4.12)$$

Định lý này chỉ rõ việc tiệm cận các thành phần của rủi ro tổng quát hóa.

## 4.2 Rủi ro và mất mát trong học không giám sát

Một số hệ quả từ **Định lý 4.1**

1. Việc mất mát trong đào tạo  $\ell_{\mathcal{T}_n}(g_{\mathcal{T}_n}) = r_{\mathcal{T}_n}(\hat{\theta}_n)$  có xu hướng đánh giá thấp rủi ro  $\ell(g^{\mathcal{G}}) = r(\theta^*)$ , bởi tập huấn luyện được dùng để huấn luyện  $g \in \mathcal{G}$  (nghĩa là ước lượng  $\theta^*$ ) và ước tính rủi ro.

Nhờ (4.12), ta biết được trung bình tổn thất đào tạo đánh giá thấp rủi ro thực sự bằng  $\text{tr}(F(\theta^*)H^{-1}(\theta^*))/(2n)$ .



## 4.2 Rủi ro và mất mát trong học không giám sát

2. Gộp 2 phương trình (4.11) và (4.12), sẽ mang lại giá trị xấp xỉ tiệm cận sau cho rủi ro tổng quát hóa dự kiến:

$$\mathbb{E} r(\widehat{\theta}_n) \simeq \mathbb{E} r_{\mathcal{T}_n}(\widehat{\theta}_n) + \frac{1}{n} \text{tr}(\mathbf{F}(\theta^*) \mathbf{H}^{-1}(\theta^*)) \quad (4.16)$$

Số hạng đầu tiên ở phía bên phải của (4.16) có thể được ước lượng (không có độ lệch) thông qua tổn thất huấn luyện  $r_{\mathcal{T}_n}(\widehat{\theta}_n)$ .

Khi mô hình thực  $f \in \mathcal{G}$ , thì  $F(\theta^*) = H(\theta^*)$ . Khi đó, khi  $\mathcal{G}$  được coi là một tập đầy đủ được tham số hóa bởi vectơ  $p$ - chiều  $\theta$ , chúng ta có thể tính gần đúng số hạng thứ hai là  $\text{tr}(F(\theta^*)H^{-1}(\theta^*))/n \approx \text{tr}(I_p)/n = p/n$ .

## 4.2 Rủi ro và mất mát trong học không giám sát

Điều này xấp xỉ của rủi ro tổng quát hóa (dự kiến):

$$\mathbb{E}r(\hat{\theta}_n) \approx r_{\mathcal{T}_n}(\hat{\theta}_n) + \frac{p}{n} \quad (4.17)$$

## 4.2 Rủi ro và mất mát trong học không giám sát

3. Nhân cả hai vế của (4.16) với  $2n$  và thay  $\text{tr}(F(\theta^*)H^{-1}(\theta^*)) \approx p$ , ta được giá trị gần đúng:

$$2nr(\hat{\theta}_n) \approx -2 \sum_{i=1}^n \ln g(X_i | \hat{\theta}_n) + 2p. \quad (4.18)$$

Vế phải của (4.18) được gọi là tiêu chí thông tin Akaike (AIC). Cũng giống như (4.17), phép gần đúng AIC có thể được sử dụng để so sánh sự khác biệt về rủi ro tổng quát hóa.

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Thuật toán Tối đa hóa Kỳ vọng là một kỹ thuật được dùng rộng rãi trong thống kê và học máy để giải bài toán tìm hợp lý cực đại hoặc hậu nghiệm cực đại (MAP) của một mô hình xác suất có các biến ẩn.

Thuật toán Tối đa hóa Kỳ vọng – (EM) là một thuật toán chung để tối đa hóa các hàm khả năng phức tạp (log-), thông qua việc giới thiệu các biến phụ trợ.

Giải thuật EM cho ta một phương pháp giải quyết bài toán trên một lớp bài toán tương đối rộng.

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Nguyên lý của nó là tại mỗi bước (E: Kỳ vọng), ta giả thiết rằng tham số đã biết và cố gắng ước lượng giá trị của biến ẩn này và dùng giá trị tìm được này ở bước (M: Tối đa hóa) để tìm giá trị của các tham số.

Ta có thể chứng minh được rằng tại mỗi vòng lặp, ta luôn tìm được kết quả tốt hơn của vòng lặp trước đó, vì thế EM luôn hội tụ về giá trị tối ưu (địa phương).

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

### Thuật toán 4.3.1: Thuật toán EM chung

---

**input:** Data  $\tau$ , initial guess  $\theta^{(0)}$ .

**output:** Approximation of the maximum likelihood estimate.

```

1  $t \leftarrow 1$ 
2 while a stopping criterion is not met do
3   Expectation Step: Find  $p^{(t)}(\mathbf{z}) := g(\mathbf{z} | \tau, \theta^{(t-1)})$  and compute the expectation
      
$$Q^{(t)}(\theta) := \mathbb{E}_{p^{(t)}} \ln g(\tau, \mathbf{Z} | \theta). \quad (4.24)$$

4   Maximization Step: Let  $\theta^{(t)} \leftarrow \operatorname{argmax}_{\theta \in \Theta} Q^{(t)}(\theta)$ .
5    $t \leftarrow t + 1$ 
6 return  $\theta^{(t)}$ 
```

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

**Thuật toán 4.3.1:** Thuật toán EM chung

**input:** Dữ liệu  $\tau$ , phỏng đoán ban đầu  $\theta^{(0)}$ .

**output:** Tính gần đúng của ước lượng khả năng xảy ra tối đa.

- 1  $t \leftarrow 1$
- 2 Sau đó, chạy vòng lặp cho đến khi thỏa điều kiện dừng:
- 3 + **Bước E(Kỳ vọng):**  
 Tìm  $p^{(t)}(z) := g(z|\tau, \theta^{(t-1)})$  và  
 Tính toán kỳ vọng của  $Q^{(t)}(\theta) := \mathbb{E}_{p^{(t)}} \ln g(\tau, Z|\theta)$ .
- 4 + **Bước M(Tối đa hóa):**  
 Cho  $\theta^{(t)} \leftarrow \operatorname{argmax}_{\theta \in \Theta} Q^{(t)}(\theta)$ .
- 5  $t \leftarrow t + 1$  và tiếp tục chạy vòng lặp.
- 6 **Nếu thỏa điều kiện thì xuất ra giá trị của  $\theta^{(t)}$**

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Điều kiện dừng khả thi là khi

$$\left| \frac{\ln g(\tau | \theta^{(t)}) - \ln g(\tau | \theta^{(t-1)})}{\ln g(\tau | \theta^{(t)})} \right| \leq \epsilon$$

đối với một số  $\epsilon > 0$  đủ nhỏ.



## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Giả sử, ta xét một mô hình thống kê bao gồm 1 tập dữ liệu quan sát được  $X$ , 1 tập dữ liệu bị thiếu hoặc ẩn biến  $Z$  và 1 vector tham số  $\theta$ , cùng với hàm số hợp lý (likelihood)  $L(\theta; X, Z) = p(X, Z|\theta)$ .

Đầu tiên thuật toán EM sẽ gán  $\theta$  với 1 bộ giá trị khởi điểm. Sau đó, EM sẽ tuần tự thực hiện các vòng lặp bằng cách áp dụng tại mỗi vòng 2 bước sau: Tại vòng lặp thứ  $t + 1, t \geq 0$ :

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

+ (E): Tính kỳ vọng của log hàm hợp lý (log-likelihood) của phân phối có điều kiện của  $Z$  cho trước giá trị của  $X$  và ước lượng của  $\theta^{(t)}$  có được ở vòng sát trước:

$$Q(\theta|\theta^{(t)}) = E_{Z|X, \theta^{(t)}}[\log L(\theta; X, Z)]$$

+ (M): Ước lượng giá trị tham số để cực đại hoá đại lượng ở (E):

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta|\theta^{(t)})$$

Thuật toán lặp lại (E) và (M) liên tiếp cho đến khi điều kiện dừng được thoả mãn.

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

### VD 4.2 (Dữ liệu được kiểm duyệt):

Giả sử vòng đời (tính bằng năm) của một loại máy nhất định được mô hình hóa thông qua phân phối  $\mathcal{N}(\mu, \sigma^2)$ .

Để ước tính  $\mu$  và  $\sigma^2$ , tuổi thọ của  $n$  máy (độc lập) được ghi lại lên đến  $c$  năm. Ký hiệu các vòng đời được kiểm duyệt này bằng  $x_1, \dots, x_n$ . Do đó,  $\{x_i\}$  là các biến ngẫu nhiên  $\{X_i\}$ , được phân phối dưới dạng  $\min\{Y, c\}$ , trong đó  $Y \sim \mathcal{N}(\mu, \sigma^2)$ .

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Theo đó khả năng xảy ra của dữ liệu  $\tau = \{x_1, \dots, x_n\}$  dưới dạng một hàm của tham số  $\theta := [\mu, \sigma^2]^T$  là:

$$g(\tau|\theta) = \prod_{i:x_i < c} \frac{\exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}} \times \prod_{i:x_i = c} \bar{\Phi}((c - \mu)/\sigma).$$

Gọi  $n_c$  là tổng số  $x_i$  sao cho  $x_i = c$ . Sử dụng  $n_c$  biến tiềm ẩn  $z = [z_1, \dots, z_{n_c}]^T$ , chúng ta có thể viết thành:

$$g(\tau, z|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\sum_{i:x_i < c} (x_i - \mu)^2}{2\sigma^2} - \frac{\sum_{i=1}^{n_c} (z_i - \mu)^2}{2\sigma^2}\right) \mathbb{1}_{\{\min_i z_i \geq c\}}$$

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

sao cho  $\int g(\tau, z|\theta) dz = g(\tau|\theta)$ . nhờ đó, chúng ta có thể áp dụng thuật toán EM để tối đa hóa khả năng xảy ra, như sau.

Đối với bước "Kỳ vọng", chúng ta cố định  $\theta$  :

$$g(z|\tau, \theta) = \prod_{i=1}^{n_c} g(z_i|\tau, \theta),$$

trong đó,  $g(z|\tau, \theta) = \mathbb{1}\{z \geq c\} \varphi_{\sigma^2}(z - \mu) / \bar{\Phi}((c - \mu)/\sigma)$  chỉ đơn giản là hàm mật độ xác suất của phân phối  $N(\mu, \sigma^2)$ , được cắt bớt thành  $[c, \infty)$ .

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Đối với bước "Tối đa hóa", chúng ta tính toán kỳ vọng của khả năng xảy ra hoàn toàn liên quan đến một  $g(z|\tau, \theta)$  cố định và sử dụng  $Z_1, \dots, Z_{n_c}$  là biến ngẫu nhiên:

$$\mathbb{E} \ln g(\tau, Z|\theta) = -\frac{\sum_{i: x_i < c} (x_i - \mu)^2}{2\mu^2} - \frac{n_c \mathbb{E}(Z - \mu)^2}{2\sigma^2} - \frac{n}{2} \ln \sigma^2 - \frac{n}{2} \ln(2\pi),$$

trong đó  $Z$  có phân phối  $\mathcal{N}(\mu, \sigma^2)$ , được cắt ngắn thành  $[c, \infty)$ . Để tối đa hóa biểu thức cuối cùng đối với  $\mu$ , chúng ta đặt đạo hàm đối với  $\mu$  thành 0 và thu được:

$$\mu = \frac{n_c \mathbb{E} Z + \sum_{i: x_i < c} x_i}{n}.$$

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Tương tự, đặt đạo hàm đối với  $\sigma^2$  thành 0 sẽ thành:

$$\sigma^2 = \frac{n_c \mathbb{E}(Z - \mu)^2 + \sum_{i: x_i < c} (x_i - \mu)^2}{n}.$$

## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

Tóm lại, EM lặp lại cho  $t = 1, 2, \dots$  được thể hiện như sau.

**Bước E:** Với ước lượng hiện tại  $\theta_t := [\mu_t, \sigma_t^2]^T$ , tính các kỳ vọng  $v_t := \mathbb{E}Z$  và  $\zeta_t^2 := \mathbb{E}(Z - \mu_t)^2$ , trong đó  $Z \sim \mathcal{N}(\mu, \sigma_t^2)$ , với điều kiện  $Z \geq c$ ; đó là,

$$v_t := \mu_t + \sigma_t^2 \frac{\varphi_{\sigma_t^2}(c - \mu_t)}{\bar{\Phi}((c - \mu_t)/\sigma_t)}$$

$$\zeta_t^2 := \sigma_t^2 \left( 1 + (c - \mu_t) \frac{\varphi_{\sigma_t^2}(c - \mu_t)}{\bar{\Phi}((c - \mu_t)/\sigma_t)} \right).$$



## 4.3 Thuật toán tối đa hóa kỳ vọng (EM)

**Bước M:** Cập nhật ước tính thành  $\theta_{t+1} := [\mu_{t+1}, \sigma_{t+1}^2]^T$  thông qua công thức:

$$\mu_{t+1} = \frac{n_c v_t + \sum_{i: x_i < c} x_i}{n}$$

$$\sigma_{t+1}^2 = \frac{n_c \zeta_t^2 + \sum_{i: x_i < c} (x_i - \mu_{t+1})^2}{n}$$

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Trong mục 1.5.2.3 chúng ta thấy hàm phân phối tích lũy thực nghiệm  $\widehat{F}_n$ , thu được từ một tập huấn luyện độc lập và phân phối đồng nhất  $\tau = \{x_1, \dots, x_n\}$  từ một phân phối không xác định trên  $\mathbb{R}$ , đưa ra ước tính về hàm phân phối tích lũy  $F$  chưa biết của phân phối lấy mẫu này. Hàm  $\widehat{F}_n$  là một hàm phân phối tích lũy luôn đúng vì nó liên tục phải, tăng dần và nằm trong khoảng từ 0 đến 1. Phân phối xác suất rời rạc tương ứng được gọi là phân phối thực nghiệm của dữ liệu

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Một biến ngẫu nhiên  $X$  được phân phối theo phân phối thực nghiệm này nhận các giá trị  $x_1, \dots, x_n$  với xác suất bằng nhau  $\frac{1}{n}$ . Khái niệm phân phối thực nghiệm tổng quát một cách tự nhiên theo các kích thước lớn hơn: một vectơ ngẫu nhiên  $X$  được phân phối theo phân phối thực nghiệm của  $x_1, \dots, x_n$  có hàm mật độ xác suất rời rạc  $P[X = x_i] = \frac{1}{n}$  với  $i = 1, \dots, n$ .

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Theo một cách nào đó, phân phối thực nghiệm là câu trả lời tự nhiên cho câu hỏi học tập không giám sát: phân phối xác suất cơ bản của dữ liệu là gì? Tuy nhiên, theo định nghĩa, phân phối thực nghiệm là một phân phối rời rạc, trong khi phân phối lấy mẫu thực sự có thể liên tục. Đối với dữ liệu liên tục cũng hợp lý khi xem xét ước tính của hàm mật độ xác suất dữ liệu. Một cách tiếp cận phổ biến là ước tính mật độ thông qua ước tính mật độ hạt nhân (KDE)

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

**Định nghĩa:** Gaussian KDE

Cho  $x_1, \dots, x_n \in R^d$  là kết quả của một mẫu độc lập và phân phối đồng nhất từ một hàm mật độ xác suất liên tục  $f$ . Ước tính mật độ hạt nhân Gaussian của  $f$  là hỗn hợp của các hàm mật độ xác suất chuẩn, có dạng:

$$g_{\tau_n}(x|\sigma) = \frac{1}{n} \sum_{i=1}^n \left( \frac{1}{(2\pi)^{d/2} \sigma^d} e^{-\frac{\|x - x_i\|^2}{2\sigma^2}} \right), \text{ với } x \in R^d$$

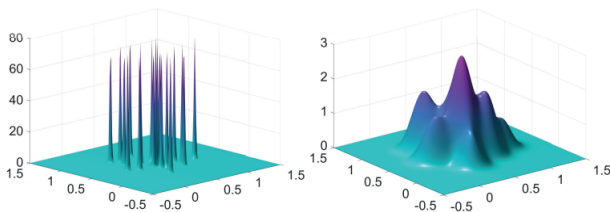
trong đó  $\sigma > 0$  được gọi là độ lệch chuẩn

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Chúng ta thấy rằng  $g_{\tau_n}$  trong định nghĩa Gaussian KDE là giá trị trung bình của một tập hợp hàm mật độ xác suất chuẩn  $n$ , trong đó mỗi phân phối chuẩn được căn giữa tại điểm dữ liệu  $x_i$  và có ma trận hiệp phương sai  $\sigma^2 I_d$ . Một câu hỏi chính là làm thế nào để chọn độ lệch chuẩn  $\sigma$  sao cho gần đúng nhất với hàm mật độ xác suất  $f$  chưa biết? Việc chọn  $\sigma$  rất nhỏ sẽ dẫn đến một ước tính "nhảy cảm", trong khi một  $\sigma$  lớn sẽ tạo ra một ước tính quá mượt mà có thể không xác định được các đỉnh quan trọng có trong hàm mật độ xác suất không xác định.

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Trường hợp minh họa sau: dữ liệu bao gồm 20 điểm được vẽ thống nhất từ hình vuông đơn vị. Do đó, hàm mật độ xác suất thực sự là 1 trên  $[0, 1]^2$  và 0 ở những nơi khác.



**Hình:** Hai KDE Gaussian hai chiều, với  $\sigma = 0,01$  (trái) và  $\sigma = 0,1$  (phải)

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Chúng ta viết lại định nghĩa Gaussian KDE dưới dạng

$$g_{\tau_n}(x|\sigma) = \frac{1}{n} \sum_{i=1}^n \frac{1}{\sigma^d} \phi\left(\frac{x - x_i}{\sigma}\right), \quad (1)$$

trong đó

$$\phi(z) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\|z\|^2}{2}}, \quad z \in R^d \quad (2)$$

là hàm mật độ xác suất của phân phối chuẩn chuẩn d-chiều. Bằng cách chọn một mật độ xác suất khác  $\phi$  trong (1), thỏa mãn  $\phi(x) = \phi(-x)$  với mọi  $x$ , chúng ta có thể thu được nhiều ước tính mật độ hạt nhân



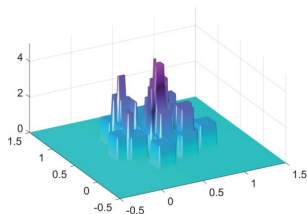
## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Ví dụ: một hàm mật độ xác suất đơn giản  $\phi$  là hàm mật độ xác suất đồng nhất trên  $[-1, 1]^d$

$$\phi(z) = \begin{cases} 2^{-d}, & \text{if } z \in [-1, 1]^d. \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Hình 2 cho thấy đồ thị của KDE tương ứng, sử dụng dữ liệu tương tự như trong Hình 1 và với độ lệch chuẩn  $\sigma = 0,1$ . Chúng tôi quan sát hành vi tương tự về mặt chất lượng đối với các KDE Gaussian và đồng nhất. Theo quy luật, sự lựa chọn của hàm  $\phi$  ít quan trọng hơn sự lựa chọn của độ lệch chuẩn trong việc xác định chất lượng của ước lượng

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ



Hình: một KDE đồng nhất hai chiều, với  $\sigma = 0,1$

Vấn đề quan trọng của việc lựa chọn độ lệch chuẩn đã được nghiên cứu rộng rãi đối với dữ liệu một chiều. Để giải thích các ý tưởng, chúng tôi sử dụng thiết lập thông thường của chúng tôi và đặt  $\tau = \{x_1, \dots, x_n\}$  là dữ liệu quan sát (một chiều) từ hàm mật độ xác suất  $f$  chưa xác định.

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Đầu tiên, chúng tôi định nghĩa hàm mất mát là

$$\text{Loss}(f(x), g(x)) = \frac{(f(x) - g(x))^2}{f(x)} \quad (3)$$

Do đó, rủi ro cần giảm thiểu là  $\ell(g) := \mathbb{E}_f \text{Loss}(f(x), g(x)) = \int (f(x) - g(x))^2 dx$

Chúng ta bỏ qua việc chọn một lớp hàm xấp xỉ bằng cách chọn người học được chỉ định bởi định nghĩa Gaussian KDE cho một  $\sigma$  cố định. Mục tiêu bây giờ là tìm một  $\sigma$  giảm thiểu rủi ro tổng quát hóa  $\ell(g_\tau(\cdot|\sigma))$  hoặc rủi ro tổng quát hóa dự kiến  $\mathbb{E}\ell(g_\tau(\cdot|\sigma))$

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Rủi ro tổng quát hóa trong trường hợp này là:

$$\int (f(x) - g_{\tau}(x|\sigma))^2 dx = \int f^2(x) dx - 2 \int f(x) g_{\tau}(x|\sigma) dx + \int g_{\tau}^2(x|\sigma) dx$$

Việc giảm thiểu biểu thức này đối với  $\sigma$  tương đương với việc giảm thiểu hai số hạng cuối cùng, có thể được viết là

$$-2\mathbb{E}_f(g_{\tau}(X|\sigma)) + \int \left[ \frac{1}{n} \sum_{i=1}^n \frac{1}{\sigma} \phi\left(\frac{x - x_i}{\sigma}\right) \right]^2 dx$$

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Biểu thức này có thể được ước lượng bằng cách sử dụng một mẫu thử nghiệm  $\{x'_1, \dots, x'_{n'}\}$  từ  $f$ , đưa ra bài toán tối thiểu hóa sau:

$$\min_{\sigma} - \frac{2}{n'} \sum_{i=1}^{n'} g_{\tau}(x'_i | \sigma) + \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \int \frac{1}{\sigma^2} \phi\left(\frac{x - x_i}{\sigma}\right) \phi\left(\frac{x - x_j}{\sigma}\right) dx,$$

Trong đó  $\int \frac{1}{\sigma^2} \phi\left(\frac{x - x_i}{\sigma}\right) \phi\left(\frac{x - x_j}{\sigma}\right) dx = \frac{1}{\sqrt{2}\sigma} \phi\left(\frac{x_i - x_j}{\sqrt{2}\sigma}\right)$  trong trường hợp của hạt nhân Gaussian (4.4.2) với  $d = 1$

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Để ước lượng  $\sigma$  theo cách này rõ ràng cần phải có một mẫu thử nghiệm, hoặc ít nhất là một ứng dụng xác nhận chéo. Một cách tiếp cận khác là giảm thiểu rủi ro tổng quát hóa dự kiến, (nghĩa là, tính trung bình trên tất cả các tập huấn luyện):

$$\mathbb{E} \int (f(x) - g_{\tau}(x|\sigma))^2 dx$$

Đây được gọi là lỗi bình phương tích hợp trung bình (MISE). Nó có thể được phân tách thành thành phần thiên vị bình phương tích hợp và thành phần phương sai tích hợp:

$$\int (f(x) - \mathbb{E}g_{\tau}(x|\sigma))^2 dx + \int \text{Var}(g_{\tau}(x|\sigma)) dx$$

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Một phân tích điển hình hiện đang được tiến hành bằng cách điều tra MISE hoạt động như thế nào đối với  $n$  lớn, dưới nhiều giả thiết khác nhau về  $f$ . Đối với  $\sigma \rightarrow 0$  và  $n\sigma \rightarrow \infty$ , xấp xỉ tiệm cận với MISE của công cụ ước lượng mật độ hạt nhân Gaussian (đối với  $d = 1$ ) được cho bởi

$$\frac{1}{4} \|f''\|^2 + \frac{1}{2n\sqrt{\pi\sigma^2}}$$

trong đó  $\|f''\|^2 := \int (f''(x))^2 dx$  Giá trị tiệm cận tối ưu của  $\sigma$  là giá trị tối thiểu

## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

$$\sigma^* := \left( \frac{1}{2n\sqrt{\pi}\|f\|^2} \right)^{\frac{1}{5}}$$

Để tính  $\sigma^*$  tối ưu trong,, người ta cần ước lượng hàm  $\|f\|^2$ . Quy tắc ngón tay cái của Gaussian là giả định rằng  $f$  là mật độ của phân phối  $N(\bar{x}, s^2)$  trong đó  $\bar{x}$  và  $s^2$  lần lượt là trung bình mẫu và

phương sai của dữ liệu. Trong trường hợp này  $\|f\|^2 = s^{-5} \pi^{-\frac{1}{2}} \frac{3}{8}$

và quy tắc ngón tay cái của Gaussian trở thành:

$$\sigma_{rot} = \left( \frac{4s^5}{3n} \right)^{1/5} \approx 1.06sn^{-1/5}$$



## 4.4 Hàm phân phối thực nghiệm và ước lượng mật độ

Tuy nhiên, chúng tôi đề xuất theta KDE nhanh và đáng tin cậy, chọn băng thông theo cách tối ưu thông qua quy trình điểm cố định. Hai hình trên minh họa một vấn đề phổ biến với KDE truyền thống: đối với các phân phối trên miền bị giới hạn, chẳng hạn như phân phối đồng đều trên  $[0, 1]$ , KDE ấn định khối lượng xác suất dương bên ngoài miền này. Một lợi thế bổ sung của theta KDE là nó phần lớn tránh được hiệu ứng ranh giới này.

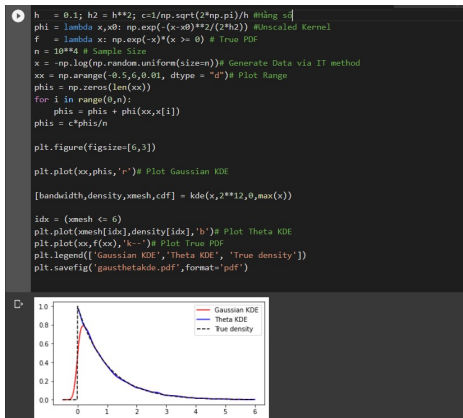
## Ví dụ 4.3 : So sánh Gaussian và theta KDE

Chương trình Python sau đây lấy một mẫu iid từ phân phối Exp (1) và xây dựng ước tính mật độ hạt nhân Gaussian. Chúng ta thấy trong Hình 4.3 rằng với sự lựa chọn thích hợp của băng thông, có thể đạt được sự phù hợp tốt với pdf thực, ngoại trừ ở ranh giới  $x = 0$ . Theta KDE không thể hiện hiệu ứng ranh giới này. Hơn nữa, nó chọn băng thông tự động, để đạt được sự phù hợp vượt trội

## Ví dụ 4.3 : So sánh Gaussian và theta KDE

Chương trình Python sau đây lấy một mẫu iid từ phân phối Exp (1) và xây dựng ước tính mật độ hạt nhân Gaussian. Chúng ta thấy trong Hình 4.3 rằng với sự lựa chọn thích hợp của băng thông, có thể đạt được sự phù hợp tốt với pdf thực, ngoại trừ ở ranh giới  $x = 0$ . Theta KDE không thể hiện hiệu ứng ranh giới này. Hơn nữa, nó chọn băng thông tự động, để đạt được sự phù hợp vượt trội

## Ví dụ 4.3 : So sánh Gaussian và theta KDE



Hình: Biểu đồ so sánh Gaussian và theta KDE

## 4.5 Phân Cụm Thông Qua Các Mô Hình Hỗn Hợp

Phân cụm liên quan đến việc nhóm các vectơ đặc trưng không được gán nhãn thành các cụm, sao cho các mẫu trong một cụm giống với nhau hơn các mẫu thuộc các cụm khác nhau. Thông thường, người ta cho rằng số lượng cụm được biết trước, nhưng nếu không thì không có thông tin trước nào được đưa ra về dữ liệu. Các ứng dụng của phân cụm có thể được tìm thấy trong các lĩnh vực truyền thông, nén và lưu trữ dữ liệu, tìm kiếm cơ sở dữ liệu, đối sánh mẫu và nhận dạng đối tượng.

## 4.5 Phân Cụm Thông Qua Các Mô Hình Hỗn Hợp

Một cách tiếp cận phổ biến để phân tích phân cụm là giả định rằng dữ liệu đến từ một tổ hợp các phân phối (thường là Gaussian), và do đó mục tiêu là ước tính các tham số của mô hình hỗn hợp bằng cách tối đa hóa hàm khả năng cho dữ liệu. Tối ưu hóa trực tiếp hàm khả năng trong trường hợp này không phải là một nhiệm vụ đơn giản, do những ràng buộc cần thiết đối với các tham số (sẽ nói thêm về điều này sau) và tính chất phức tạp của hàm khả năng, nói chung có rất nhiều địa phương cực đại và điểm yên ngựa.

## 4.5 Phân Cụm Thông Qua Các Mô Hình Hỗn Hợp

Một phương pháp phổ biến để ước lượng các tham số của mô hình hỗn hợp là thuật toán EM, đã được thảo luận trong một cài đặt tổng quát hơn trong Phần 4.3. Trong phần này, chúng tôi giải thích những điều cơ bản về mô hình hỗn hợp(trong active 128 )và giải thích hoạt động của phương pháp EM trong bối cảnh này. Ngoài ra, chúng ta chỉ ra cách các phương pháp tối ưu hóa trực tiếp có thể được sử dụng để tối đa hóa khả năng xảy ra.

## 4.5 Phân Cụm Thông Qua Các Mô Hình Hỗn Hợp

### 4.5.1 Mô hình hỗn hợp.

Cho  $\tau = \{(X_1, X_2, \dots, X_n)\}$  là các vectơ ngẫu nhiên nhận các giá trị trong một số tập  $\chi \subseteq R^d$ , mỗi  $X_i$  được phân phối theo mật độ hỗn hợp.

$$g(x|\theta) = w_1\phi_1(x) + \dots + w_n\phi_n(x) \quad (4.31)$$

trong đó  $\phi_1, \dots, \phi_K$  là mật độ xác suất (rời rạc hoặc liên tục) trên  $\chi$  và trọng số dương  $w_1, \dots, w_K$  tổng lên đến 1. Bản pdf hỗn hợp này có thể được diễn giải theo cách sau. Gọi  $Z$  là một biến ngẫu nhiên rời rạc nhận các giá trị  $1, 2, \dots, K$  với các xác suất  $w_1, \dots, w_K$  và cho  $X$  là vectơ ngẫu nhiên có pdf có điều kiện, cho trước  $Z = z$ , là  $\phi_z$ .



## 4.5.1 Mô hình hỗn hợp

Theo quy tắc tích số (C.17), joint pdf của  $Z$  và  $X$  được đưa ra bởi  $\phi_{Z,X}(z, x) = \phi_Z(z)\phi_{X|Z}(x|z) = w_z\phi_z(x)$  và pdf cận biên của  $X$  được tìm thấy bằng cách tính tổng joint pdf với các giá trị của  $z$ , cho (4.31). Do đó, một vectơ ngẫu nhiên  $X \sim g$  có thể được mô phỏng theo hai bước:

1. Đầu tiên, vẽ  $Z$  theo các xác suất

$$P[Z = z] = w_z, z = 1, \dots, K.$$

2. Sau đó vẽ  $X$  theo pdf  $\phi_z$

Vì  $\tau$  chỉ chứa các biến  $\{X_i\}$ , nên  $\{Z_i\}$  được xem như là các biến tiềm ẩn. Chúng ta có thể gọi  $Z_i$  là nhãn ẩn của cụm mà  $X_i$  thuộc về

## 4.5.1 Mô hình hỗn hợp

Thông thường, mỗi  $\phi_k$  trong (4.31) được giả định là đã biết đến một số vector tham số  $\eta_k$ . Thông thường 1 trong phân tích phân cụm là làm việc với các hỗn hợp Gaussian; nghĩa là, mỗi mật độ  $\phi_k$  là Gaussian với một số vectơ kỳ vọng chưa biết  $\mu_k$  và ma trận hiệp phương sai  $\Sigma_k$ . Chúng ta tập hợp tất cả các tham số chưa biết, bao gồm cả trọng số  $w_k$ , vào một vectơ tham số  $\theta$ . Như thường lệ,  $\tau = x_1, \dots, x_n$  biểu thị kết quả của  $\tau$ . Vì các thành phần của  $\tau$  là iid, joint pdf của chúng được đưa ra bởi

$$g(\tau|\theta) = \prod_{i=1}^n g(x_i|\theta) = \prod_{i=1}^n \sum_{k=1}^K w_k \phi_k(x_i|\mu_k, \Sigma_k) \quad (4.32)$$

## 4.5.1 Mô hình hỗn hợp

Theo lý luận tương tự như đối với (4.5), chúng ta có thể ước tính  $\theta$  từ một kết quả  $\tau$  bằng cách tối ưu hóa hàm log-khả năng

$$l(\theta|\tau) = \sum_{i=1}^n \ln g(x_i|\theta) = \sum_{i=1}^n \ln\left(\sum_{k=1}^K w_k \phi_k(x_i|\mu_k, \Sigma_k)\right) \quad (4.33)$$

Tuy nhiên, việc tìm giá trị cực đại của  $L(\theta|\tau)$  nói chung là không dễ dàng, vì hàm thường là đa chiều

## 4.5.1 Mô hình hỗn hợp

Ví dụ 4.4 (Phân cụm thông qua các mô hình hỗn hợp) Dữ liệu được mô tả trong Hình 4.4 bao gồm 300 điểm dữ liệu được tạo độc lập từ ba phân phối chuẩn hai biến , có các tham số được đưa ra trong cùng một hình. Đối với mỗi trong số ba giải thưởng này , chính xác 100 điểm đã được tạo ra. Tốt nhất, chúng tôi muốn phân cụm dữ liệu thành ba cụm tương ứng với ba trường hợp.

## 4.5.1 Mô hình hỗn hợp

Để phân cụm dữ liệu thành ba nhóm, một mô hình khả thi cho dữ liệu là giả định rằng các điểm được lấy từ hỗn hợp (chưa biết) của ba phân biệt Gaussian 2 chiều. Đây là một cách tiếp cận hợp lý, mặc dù trên thực tế, dữ liệu không được mô phỏng theo cách này. Đó là hướng dẫn để hiểu sự khác biệt giữa hai mô hình..

## 4.5.1 Mô hình hỗn hợp

Trong mô hình hỗn hợp, mỗi nhãn cụm  $Z$  nhận giá trị 1, 2, 3 với xác suất bằng nhau và do đó, vẽ các nhãn một cách độc lập, tổng số điểm trong mỗi cụm sẽ được  $B(300, 1/3)$  (phân phối nhị thức). Tuy nhiên, trong mô phỏng thực tế, số điểm trong mỗi cụm chính xác là 100. Tuy nhiên, mô hình hỗn hợp sẽ là một mô hình chính xác (mặc dù không chính xác) cho những dữ liệu này.

Hình 4.5 hiển thị mật độ hỗn hợp Gaussian “mục tiêu” cho dữ liệu trong Hình 4.4; nghĩa là hỗn hợp có khối lượng bằng nhau và với các thông số chính xác như quy định trong Hình 4.4

## 4.5.1 Mô hình hỗn hợp

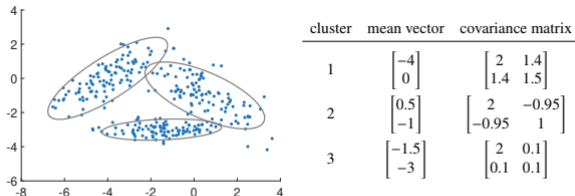


Figure 4.4: Cluster the 300 data points (left) into three clusters, without making any assumptions about the probability distribution of the data. In fact, the data were generated from three bivariate normal distributions, whose parameters are listed on the right.

## 4.5.1 Mô hình hỗn hợp

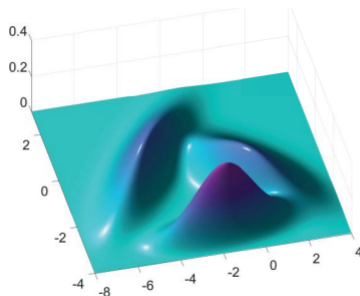


Figure 4.5: The target mixture density for the data in [Figure 4.4](#).

Trong phần tiếp theo, chúng ta sẽ thực hiện phân cụm bằng thuật toán EM



## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

### 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp.

Như chúng ta đã thấy trong Phần 4.3, thay vì tối đa hóa hàm khả năng log trong (4.33) trực tiếp từ dữ liệu  $\tau = \{x_1, \dots, x_n\}$ , thuật toán EM trước tiên tăng cường dữ liệu bằng vectơ của các biến tiềm ẩn - trong trường hợp này là các nhãn cụm ẩn  $z = \{z_1, \dots, z_n\}$ .

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Ý tưởng là  $\tau$  là chỉ phân quan sát được của dữ liệu ngẫu nhiên hoàn chỉnh  $(\tau, Z)$ , được tạo ra thông qua quy trình hai bước được mô tả ở trên. Nghĩa là, đối với mỗi điểm dữ liệu  $X$ , trước tiên hãy vẽ nhãn cụm  $Z \in \{1, \dots, K\}$  theo xác suất  $\{w_1, \dots, w_K\}$  và sau đó, cho  $Z = z$ , vẽ  $X$  từ  $\phi_z$ . Bản joint pdf của  $\tau$  và  $Z$  là

$$g(\tau, z|\theta) = \prod_{i=1}^n w_{z_i} \phi_{z_i}(x_i)$$

có dạng đơn giản hơn nhiều so với (4.32).

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Tiếp theo là complete-data log-likelihood function

$$\tilde{l}(\theta|\tau, z) = \sum_{i=1}^n \ln(w_{zi}\phi_{zi}(x_i)) \quad (4.34)$$

thường dễ tối đa hóa hơn khả năng log ban đầu (4.33), đối với bất kỳ  $(\tau, z)$  đã cho. Tuy nhiên, tất nhiên các biến tiềm ẩn  $z$  không được quan sát và  $\tilde{l}(\theta|\tau, z)$  không thể được đánh giá.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Trong bước E của thuật toán EM, the complete-data log-likelihood được thay thế bằng kỳ vọng  $E_p \tilde{l}(\theta|\tau, z)$  trong đó chỉ số con  $p$  trong kỳ vọng chỉ ra rằng  $Z$  được phân phối theo pdf có điều kiện của  $Z$  cho trước  $T = \tau$ ; nghĩa là, với pdf

$$p(z) = g(z|\tau, \theta) \propto g(\tau, z|\theta) \quad (4.35)$$

Chú ý rằng  $p(z)$  có dạng  $p_1(z_1)....p_n(z_n)$  sao cho  $T = \tau$  cho trước, các thành phần của  $Z$  là độc lập với nhau. Thuật toán EM cho các mô hình hỗn hợp hiện có thể được xây dựng như sau.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

---

### Algorithm 4.5.1: EM Algorithm for Mixture Models

---

**input:** Data  $\tau$ , initial guess  $\theta^{(0)}$ .

**output:** Approximation of the maximum likelihood estimate.

1  $t \leftarrow 1$

2 **while** a stopping criterion is not met **do**

3     **Expectation Step:** Find  $p^{(t)}(z) := g(z | \tau, \theta^{(t-1)})$  and  $Q^{(t)}(\theta) := \mathbb{E}_{p^{(t)}} \tilde{l}(\theta | \tau, Z)$ .

4     **Maximization Step:** Let  $\theta^{(t)} \leftarrow \operatorname{argmax}_{\theta} Q^{(t)}(\theta)$ .

5      $t \leftarrow t + 1$

6 **return**  $\theta^{(t)}$

---

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

**Thuật toán 4.5.1:** Thuật toán EM cho các mô hình hỗn hợp :

input :Dữ liệu  $\tau$ , dự đoán ban đầu

output: Sự gần đúng của ước tính khả năng xảy ra tối đa

1  $t \leftarrow 1$

2 Trong khi một điều kiện dừng lại chưa được đáp ứng

3 Bước kỳ vọng: Tìm  $p^{(t)}(z) := g(z|\tau, \theta^{(t-1)})$  and

$Q^t(\theta) := E_{p^t} \tilde{l}(\theta|\tau, z)$ .

4 Bước tối đa hóa: cho  $\theta \leftarrow \operatorname{argmax}_{\theta} Q^t(\theta)$

5  $t \leftarrow t + 1$

6 trở lại  $\theta$

Điều kiện kết thúc có thể xảy ra là dừng khi

$$|l(\theta^{(t)}|\tau) - l(\theta^{(t-1)}|\tau)|/|l(\theta^t|\tau)| < \epsilon$$

đối với một số dung sai nhỏ  $\epsilon > 0$

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Như đã được đề cập trong Phần 4.3, chuỗi các giá trị khả năng xảy ra log không giảm theo mỗi lần lặp. Theo những điều kiện liên tục nhất định , chuỗi  $\{\theta^{(t)}\}$  được đảm bảo hội tụ tới một cực đại cục bộ của khả năng  $l$ . Sự hội tụ thành công cụ tối đa hóa toàn cầu (nếu nó tồn tại) phụ thuộc vào lựa chọn thích hợp cho giá trị bắt đầu. Thông thường, thuật toán được chạy từ các điểm bắt đầu ngẫu nhiên khác nhau .

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Đối với trường hợp hỗn hợp Gauss, mỗi  $\phi_k = \phi(\cdot | \mu_k, \Sigma_k)$ ,  $k=1, \dots, K$  là mật độ của phân bố Gaussian d-chiều. Gọi  $\theta^{(t-1)}$  là dự đoán hiện tại cho vectơ tham số tối ưu, bao gồm trọng số  $\{w_k^{(t-1)}\}$ , vectơ trung bình  $\{\mu_k^{(t-1)}\}$  và ma trận hiệp phương sai  $\{\Sigma_k^{(t-1)}\}$ .



## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Đầu tiên chúng ta xác định  $p^{(t)}$  - pdf của  $Z$  với điều kiện  $T = \tau$  - đối với dự đoán  $\theta^{(t-1)}$  đã cho. Như đã đề cập trước đây, các thành phần của  $Z$  cho trước  $T = \tau$  là độc lập, vì vậy nó đủ để chỉ định pdf rời rạc,  $p_i^{(t)}$  mà tôi nói, của mỗi  $Z_i$  cho điểm quan sát  $X_i = x_i$ . Cái sau có thể được tìm thấy từ công thức của Bayes:

$$p_i^t(k) \propto w_k^{(t-1)} \phi_k(x_i | \mu_k^{(t-1)}, \Sigma_k^{(t-1)}), k = 1, \dots, K \quad (4.36)$$

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Tiếp theo, theo quan điểm của (4.34), hàm  $Q^{(t)}(\theta)$  có thể được viết dưới dạng

$$\begin{aligned} Q^{(t)}(\theta) &= E_{p^{(t)}} \sum_{i=1}^n (\ln w_{Z_i} + (\ln \phi_{Z_i}(x_i | \mu_{Z_i}, \Sigma_{Z_i}))) \\ &= \sum_{i=1}^n (E_{p^{(t)}} (\ln w_{Z_i} + \ln(\phi_{Z_i}(x_i | \mu_{Z_i}, \Sigma_{Z_i})))) \end{aligned}$$

trong đó  $\{Z_i\}$  là độc lập và  $Z_i$  được phân phối theo  $p_i^{(t)}$  trong (4.36). Điều này làm giảm tốc độ của E-step.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Trong bước M, chúng ta cực đại  $Q^{(t)}$  đối với tham số  $\theta$ ; nghĩa là đối với  $\{w_k\}$ ,  $\{\mu_k\}$  và  $\{\Sigma_k\}$ . Đặc biệt, chúng ta tối đa hóa

$$\sum_{i=1}^n \sum_{k=1}^K p_i^k(t) (\ln w_k + \ln \phi_k(x_i | \mu_k, \Sigma_k))$$

với điều kiện  $\sum_k w_k = 1$ . Sử dụng nhân Lagrange và thực tế là  $\sum_{k=1}^K p_i^{(t)}(k) = 1$  đưa ra nghiệm cho  $\{w_k\}$

$$w_k = \frac{1}{n} \sum_{i=1}^n p_i^{(t)}(k), \quad k = 1, 2, \dots, K \quad (4.37)$$

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Các giải pháp cho  $\mu_k$  và  $\Sigma_k$  bây giờ tuân theo từ việc tối đa hóa  $\sum_{i=1}^n p_i^{(t)}(k) \ln \phi_k(x_i | \mu_k, \Sigma_k)$

$$\mu_k = \frac{\sum_{i=1}^n p_i^{(t)}(k) x_i}{\sum_{i=1}^n p_i^{(t)}(k)}, k = 1, \dots, K \quad (4.38)$$

and

$$\Sigma_k = \frac{\sum_{i=1}^n p_i^{(t)}(k) (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^n p_i^{(t)}(k)}, k = 1, \dots, K \quad (4.39)$$

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

những công thức rất giống với các công thức nổi tiếng cho MLE của các tham số của phân phối Gauss. Sau khi gán các tham số nghiệm cho  $\theta^{(t)}$  và tăng bộ đếm lặp  $t$  lên 1, các bước (4.36), (4.37), (4.38) và (4.39) được lặp lại cho đến khi đạt được sự kết hợp. Sự hội tụ của thuật toán EM rất nhạy cảm với việc lựa chọn các tham số ban đầu. Do đó, chúng tôi đề nghị thử nhiều điều kiện bắt đầu khác nhau. Để thảo luận thêm về các khía cạnh lý thuyết và thực tiễn của thuật toán EM, chúng tôi đề cập đến [85]. .

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Ví dụ 4.5 (Phân cụm qua EM) Chúng tôi quay lại dữ liệu trong Ví dụ 4.4, được mô tả trong Hình 4.4 và áp dụng mô hình mà dữ liệu đến từ hỗn hợp của ba phân phối Gaussian hai biến.

Mã Python bên dưới thực hiện thủ tục EM được mô tả trong Thuật toán 4.5.1.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Các vectơ trung bình ban đầu  $\{\mu_k\}$  của phân bố Gaussian hai biến được chọn (từ việc kiểm tra bằng mắt) để nằm gần đúng ở giữa mỗi cụm, trong trường hợp này là  $[-2, -3]^T$ ,  $[-4, 1]^T$ ,  $[0, -1]^T$ . Các ma trận hiệp phương sai tương ứng ban đầu được chọn làm ma trận nhận dạng, điều này thích hợp với sự trải rộng dữ liệu quan sát được trong Hình 4.4.

Cuối cùng, các trọng lượng ban đầu là  $1/3, 1/3, 1/3$ . Để đơn giản, thuật toán dừng sau 100 lần lặp, trong trường hợp này là quá đủ để đảm bảo sự hội tụ.

Mã và dữ liệu có sẵn từ trang web của cuốn sách trong thư mục GitHub Chương 4

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

```
import numpy as np
from scipy.stats import multivariate_normal

Xmat = np.genfromtxt('clusterdata.csv', delimiter=',')
K = 3
n, D = Xmat.shape

W = np.array([[1/3, 1/3, 1/3]])
M = np.array([[-2.0, -4.0], [-3.1, -1]], dtype=np.float32)
# Note that if above *all* entries were written as integers, M would
# be defined to be of integer type, which will give the wrong answer

C = np.zeros((3, 2, 2))

C[:, 0, 0] = 1
C[:, 1, 1] = 1

p = np.zeros((3, 300))

for i in range(0, 100):

    #E-step
    for k in range(0, K):
        mvn = multivariate_normal( M[:, k].T, C[k, :, :] )
        p[k, :] = W[0, k]*mvn.pdf(Xmat)

    # M-Step
    p = (p/sum(p, 0)) #normalize
    W = np.mean(p, 1).reshape(1, 3)

    for k in range(0, K):
        M[:, k] = (Xmat.T @ p[k, :].T)/sum(p[k, :])
        xm = Xmat.T - M[:, k].reshape(2, 1)
        C[k, :, :] = xm @ (xm*p[k, :].T)/sum(p[k, :])
```



## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

```

import numpy as np
from scipy.stats import multivariate_normal
Xmat = np.genfromtxt('clusterdata.txt', delimiter=',')
K = 3
n, D = Xmat.shape
W = np.array([[1/3, 1/3, 1/3]])
M = np.array([[-2.0, -4.0], [-3, 1, -1]], dtype=np.float32)
# Note that if above *all* entries were written as integers, M would
# be defined to be of integer type, which will give the wrong answer
C = np.zeros((3, 2, 2))
C[:, 0, 0] = 1
C[:, 1, 1] = 1
p = np.zeros((3, 300))
for i in range(0, 100):
    #E-step
    for k in range(0, K):
        mvn = multivariate_normal(M[:, k].T, C[k, :, :])
        p[k, :] = W[0, k] * mvn.pdf(Xmat)
    # M-Step
    p = (p / sum(p, 0)) # normalize
    W = np.mean(p, 1).reshape(1, 3)
    for k in range(0, K):

```

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

```

C = np.zeros ((3 ,2 ,2))
C[:,0 ,0] = 1
C[:,1 ,1] = 1
p = np.zeros ((3 ,300))
for i in range(0 ,100):
    #E-step
    for k in range(0,K):
        mvn = multivariate_normal ( M[:,k].T, C[k, : ,:] )
        p[k ,:] = W[0,k]* mvn.pdf(Xmat)
    # M-Step
    p = (p/sum(p ,0)) # normalize
    W = np.mean(p ,1).reshape (1 ,3)
    for k in range(0,K):
        M[:,k] = (Xmat.T @ p[k ,:].T)/sum(p[k ,:])
        xm = Xmat.T - M[:,k]. reshape (2 ,1)
        C[k, : ,:] = xm @ (xm*p[k ,:].T)/sum(p[k ,:])
print(C)

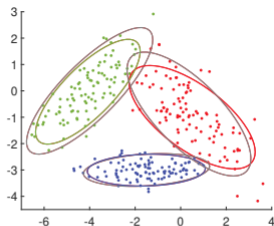
[[[ 1.74763402  0.03378977]
  [ 0.03378977  0.09495477]]

```

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Các thông số ước lượng của sự phân bố hỗn hợp được cho ở bên phải của Hình 4.6. Sau khi gán nhãn lại cho các cụm, chúng ta có thể quan sát thấy sự trùng khớp chặt chẽ với các thông số trong Hình 4.4. Các hình elip ở phía bên trái của Hình 4.6 cho thấy sự phù hợp chặt chẽ giữa các hình elip xác suất 95% của các phân phối Gaussian ban đầu (màu xám) và các phân bố ước tính. Một cách tự nhiên để phân cụm từng điểm  $x_i$  là gán nó vào cụm  $k$  mà xác suất có điều kiện  $p_i(k)$  là cực đại (với các mối quan hệ được giải quyết tùy ý). Điều này cho phép nhóm các điểm thành các cụm màu đỏ, xanh lục và xanh lam trong hình.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp



| weight | mean vector                                     | covariance matrix  |
|--------|---|--|
| 0.33   | $\begin{bmatrix} -1.51 \\ -3.01 \end{bmatrix}$  | $\begin{bmatrix} 1.75 & 0.03 \\ 0.03 & 0.095 \end{bmatrix}$  |
| 0.32   | $\begin{bmatrix} -4.08 \\ -0.033 \end{bmatrix}$ | $\begin{bmatrix} 1.37 & 0.92 \\ 0.92 & 1.03 \end{bmatrix}$   |
| 0.35   | $\begin{bmatrix} 0.36 \\ -0.88 \end{bmatrix}$   | $\begin{bmatrix} 1.93 & -1.20 \\ -1.20 & 1.44 \end{bmatrix}$ |

Figure 4.6: The results of the EM clustering algorithm applied to the data depicted in Figure 4.4.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Để thay thế cho thuật toán EM, tất nhiên người ta có thể sử dụng các thuật toán tối ưu hóa đa văn bản liên tục để tối ưu hóa trực tiếp hàm khả năng đăng nhập  $l(\theta|\tau) = \ln g(\tau|\theta)$  trong (4.33) trên tập  $\Theta$  tất cả những gì có thể  $\theta$ . Điều này được thực hiện chẳng hạn trong [15], chứng tỏ kết quả vượt trội so với EM khi có ít điểm dữ liệu .

Điều tra kỹ hơn về hàm khả năng cho thấy rằng có một vấn đề tiềm ẩn với bất kỳ cách tiếp cận khả năng tối đa nào để phân cụm nếu  $\Theta$  được chọn càng lớn càng tốt – tức là, bất kỳ phân phối hỗn hợp nào cũng có thể xảy ra.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Để chứng minh vấn đề này, hãy xem xét Hình 4.7, mô tả hàm mật độ xác suất,  $g(\cdot|\theta)$  của hỗn hợp hai phân bố Gaussian, trong đó  $\theta = [w, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2]^T$  là véc tơ tham số của phân bố hỗn hợp. Hàm log-khả năng xảy ra được cho bởi

$$l(\theta|\tau) = \sum_{i=1}^4 \ln g(x_i|\theta)$$

trong đó  $x_1, \dots, x_4$  là dữ liệu (được biểu thị bằng các dấu chấm trong hình)

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

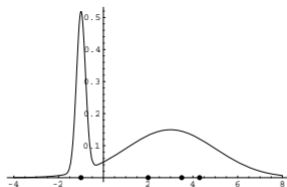


Figure 4.7: Mixture of two Gaussian distributions.

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Rõ ràng là bằng cách cố định hằng số trộn  $w$  ở 0,25 (giả sử) và căn giữa cụm đầu tiên ở  $x_1$ , người ta có thể thu được giá trị khả năng lớn tùy ý bằng cách lấy phương sai của cụm đầu tiên nhỏ tùy ý.

Tương tự, đối với dữ liệu có chiều cao hơn, bằng cách chọn các cụm “điểm” hoặc “dòng” hoặc các cụm “suy giảm” nói chung, người ta có thể làm cho giá trị của khả năng là vô hạn.



## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Đây là biểu hiện của vấn đề trang bị quá mức quen thuộc đối với mất đào tạo mà chúng ta đã gặp trong Chương 2.

Do đó, cực đại không bị giới hạn của hàm khả năng log-khả năng là một bài toán khó, bất kể việc lựa chọn thuật toán tối ưu hóa là gì!

## 4.5.2 Thuật Toán EM Cho Các Mô Hình Hỗn Hợp

Hai giải pháp khả thi cho vấn đề "overfitting" này là:

1. Hạn chế tập tham số  $\Theta$  theo cách không cho phép các cụm suy biến (đôi khi được gọi là cụm giả).
2. Chạy thuật toán đã cho và nếu giải pháp bị suy biến, hãy loại bỏ nó và chạy thuật toán mới. Tiếp tục khởi động lại thuật toán cho đến khi thu được giải pháp không suy biến .

Cách tiếp cận đầu tiên thường được áp dụng cho các thuật toán tối ưu hóa đa chiều và cách tiếp cận thứ hai được sử dụng cho thuật toán EM.

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Trong phần này, ta sẽ xem xét một cách tiếp cận sâu hơn để phân cụm bằng cách bỏ qua các thuộc tính phân phối của dữ liệu. Các thuật toán kết quả có xu hướng mở rộng tốt hơn với số lượng mẫu  $n$  và số chiều  $d$ .

Cho tập hợp  $\tau := \{x_1, \dots, x_n\}$  các điểm dữ liệu trong một số  $d$  chiều không gian  $X$ , chia tập dữ liệu này thành  $K$  nhóm sao cho một số hàm mất được tối thiểu hóa.

Đầu tiên, ta phân chia toàn bộ không gian  $X$ , sử dụng một số hàm khoảng cách  $\text{dist}(\cdot, \cdot)$  trên không gian này. Một lựa chọn tiêu chuẩn là Euclidean (hoặc  $L_2$ ) khoảng cách:

$$\text{dist}(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Các thước đo khoảng cách thường được sử dụng khác trên  $\mathbb{R}^d$  bao gồm:

Khoảng cách Manhattan

$$\sum_{i=1}^d |x_i - x'_i|$$

Khoảng cách tối đa

$$\max_{i=1, \dots, d} |x_i - x'_i|$$

Trên tập hợp các chuỗi có độ dài  $d$ , một thước đo khoảng cách thường được sử dụng là khoảng cách Hamming

$$\sum_{i=1}^d 1_{x_i \neq x'_i}$$

Nghĩa là số lượng ký tự không khớp. Ví dụ, khoảng cách Hamming giữa 010101 và 011010 là 4.

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Chúng ta có thể phân vùng không gian  $X$  thành các vùng như sau:

Đầu tiên, chúng ta chọn  $K$  điểm  $c_1, \dots, c_K$  được gọi là tâm cụm hoặc vectơ nguồn. Với mỗi  $k = 1, \dots, K$ , sao cho:

$$R_k = \{x \in X : \text{dist}(x, c_k) \leq \text{dist}(x, c_i) \text{ cho tất cả } i \neq j\}$$

là tập hợp các điểm trong  $X$  nằm gần  $c_k$  hơn bất kỳ tâm nào khác.

Các vùng hoặc ô  $R_k$  chia không gian  $X$  thành cái được gọi là Voronoi diagram

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

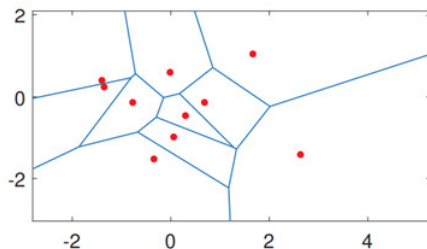
Hình 4.8 cho thấy sự phân chia Voronoi của máy bay thành mười vùng, sử dụng Euclidean khoảng cách.

Ở đây ranh giới giữa các ô Voronoi là đường thẳng. Đặc biệt, nếu ô  $R_i$  và  $R_j$  có chung một đường viền thì một điểm trên đường viền này phải thỏa mãn

$$\|x - c_i\| = \|x - c_j\|$$

nghĩa là nó phải nằm trên đường thẳng đi qua điểm  $(c_j + c_i)/2$  và vuông góc với  $c_j - c_i$

## 4.6 Phân cụm thông qua lượng tử hóa vectơ



Hình 4.8: Một điểm dừng Voronoi của máy bay gồm mười ô, được xác định bởi (màu đỏ) các trung tâm.

Khi các tâm được chọn, các điểm trong  $\tau$  có thể được nhóm lại theo trung tâm gần nhất của chúng.

Các điểm trên ranh giới phải được xử lý riêng biệt, chọn các trung tâm để phân cụm dữ liệu trong một số cách tối ưu.

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Về khung học tập (không giám sát), muốn ước tính một vectơ  $x$  qua một trong  $c_1, \dots, c_K$ , sử dụng một hàm có giá trị vectơ hằng số

$$g(x|C) := \sum_{k=1}^K c_k \mathbf{1}\{x \in R_k\}$$

trong đó  $C$  là ma trận  $d \times K$   $[c_1, \dots, c_K]$ .

Do đó,  $g(x|C) = c_k$  khi  $x$  nằm trong vùng  $R_k$ . Trong lớp hàm  $G$  này, được tham số hóa bởi  $C$ , mục đích là giảm thiểu sự mất mát trong đào tạo.



## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Đặc biệt, đối với tổn thất sai số bình phương.

$$Loss(x, x') = \|x - x'\|^2$$

$$l_{\tau n}(g(\cdot|C)) = \frac{1}{n} \sum_{i=1}^n \|x_i - g(x_i|C)\|^2$$

$$= \frac{1}{n} \sum_{k=1}^K \sum_{x \in R_k \cap \tau n} \|x - c_k\|^2$$

Do đó, sự mất mát trong đào tạo giảm thiểu khoảng cách bình phương trung bình giữa các trung tâm. Điều này cũng kết hợp cả các bước mã hóa và giải mã trong lượng tử hóa vectơ

## 4.6 Phân cụm thông qua lượng tử hóa vectơ

Cụ thể, muốn "lượng tử hóa" hoặc "mã hóa" các vectơ trong  $\tau$  theo cách mà mỗi vectơ được biểu diễn bởi một trong  $K$  vectơ nguồn  $c_1, \dots, c_K$ , sao cho (4.40) đại diện được giảm thiểu.

Điều quan trọng là nhận ra rằng trong trường hợp này (4.40) được coi là một chức năng của các trung tâm, trong đó mỗi điểm  $x$  được gán cho tâm gần nhất, do đó xác định các cụm. Nó được biết rằng loại vấn đề này tối ưu hóa đối với các trung tâm rất đa dạng, tùy thuộc vào các cụm ban đầu, các thủ tục dựa trên các điểm có xu hướng hội tụ ở mức tối thiểu cục bộ hơn là mức tối thiểu toàn cầu.

## 4.6.1 K- Phương tiện

Một trong những phương pháp đơn giản nhất để phân cụm là phương pháp K-mean. Nó là một phương pháp lặp lại trong đó, bắt đầu từ phỏng đoán ban đầu cho các trung tâm, các trung tâm mới được hình thành bằng cách lấy trung tâm phương tiện mẫu của các điểm hiện tại trong mỗi cụm. Do đó, các trung tâm mới là trung tâm của các điểm trong mỗi ô.

## 4.6.1 K- Phương tiện

Mặc dù tồn tại nhiều loại K-means khác nhau thuật toán, chúng về cơ bản đều có dạng sau:

**Thuật toán 4.6.1 : K-Means**

input: Tập hợp các điểm  $\tau = \{x_1, \dots, x_n\}$ , số cụm  $K$ , tâm ban đầu

$c_1, \dots, c_K$ .

đầu ra: Trung tâm cụm và ô (vùng).

1 trong khi tiêu chí dừng không được đáp ứng

2      $R_1, \dots, R_K \leftarrow \emptyset$  (tập trống).

3     for  $i = 1$  to  $n$  do

4          $d \leftarrow [\text{dist}(x_i, c_1), \dots, \text{dist}(x_i, c_K)]$                      // khoảng cách đến các tâm

5          $k \leftarrow \text{argmin}_j d_j$

6          $R_k \leftarrow R_k \cup \{x_i\}$                                      // gán  $x_i$  cho cụm  $k$

7     for  $k = 1$  to  $K$  do

số 8          $c_k \leftarrow \sum \frac{x \in R_k X}{|R_k|}$      // tính tâm mới dưới dạng tâm của các điểm

9 trả về  $\{c_k\}, \{R_k\}$

## 4.6.1 K- Phương tiện

Tại mỗi lần lặp, đối với một lựa chọn trung tâm nhất định, mỗi điểm trong  $\tau$  được gán cho trung tâm gần nhất của nó. Sau khi tất cả các điểm đã được chỉ định, các trung tâm được tính lại thành trọng tâm của tất cả các điểm trong cụm hiện tại (Dòng 8). Một tiêu chí dừng điển hình là dừng lại khi các trung tâm không còn thay đổi nhiều. Vì thuật toán khá nhạy cảm với sự lựa chọn của các trung tâm ban đầu, nên thận trọng khi thử nhiều giá trị bắt đầu.

## 4.6.1 K- Phương tiện

Ví dụ: Đã chọn ngẫu nhiên từ hộp giới hạn của các điểm dữ liệu. Giả sử trong thuật toán EM, chúng ta có Gaussian hỗn hợp với ma trận hiệp phương sai cố định  $\sum_k \sigma^2 \mathbb{I}_d$ ,  $k = 1, \dots, K$ , trong đó  $\sigma^2$  phải là được coi là rất nhỏ. Xem xét lần lượt  $t$  của thuật toán EM. Đang có thu được vectơ kỳ vọng  $\mu_k^{t-}$  và trọng lượng  $w_k^{t-1}$ ,  $k = 1, \dots, K$ , mỗi điểm  $x_i$  là đã ký một nhãn cụm  $Z_i$  theo các xác suất  $p_i^{(t)}(k)$ ,  $k = 1, \dots, K$  (4.36).

## 4.6.1 K- Phương tiện

Nhưng đối với  $\sigma^2 \rightarrow 0$  thì phân phối xác suất  $p_i^{(t)}(k)$  trở nên thoái hóa, đặt tất cả khối lượng xác suất trên  $\operatorname{argmin}_k \|x_i - \mu_k\|^2$ . Điều này tương ứng với quy tắc K-mean chỉ định  $x_i$  đến trung tâm cụm gần nhất của nó.

Hơn nữa, ở bước M (4.38) mỗi trung tâm cụm  $\mu_k^{(t)}$  hiện tại là được cập nhật theo giá trị trung bình của  $x_i$  đã được gán cho cụm  $k$ . Lưu ý rằng các điểm dữ liệu được lưu trữ dưới dạng  $300 \times 2$  ma trận  $X_{\text{mat}}$ . Chúng ta lấy các tâm bắt đầu tương tự như trong ví dụ EM:  $c_1 = [2, 3]^T$ ,  $c_2 = [4, 1]^T$  và  $c_3 = [0, 1]^T$ . Cũng lưu ý rằng khoảng cách Euclid bình phương được sử dụng trong tính toán, vì chúng được tính toán nhanh hơn một chút so với khoảng cách Euclide (vì không có hình vuông tính toán gốc là bắt buộc) trong khi mang lại chính xác các đánh giá trung tâm cụm.

## 4.6.1 K- Phương tiện

Kmeans.py

```

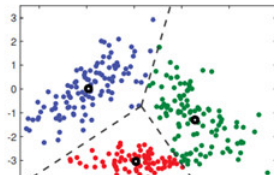
import numpy as np
Xmat = np.genfromtxt('clusterdata.csv', delimiter=',')
K = 3
n, D = Xmat.shape
c = np.array([[ -2.0, -4.0], [-3.1, -1]]) #initialize centers
cold = np.zeros(c.shape)
dist2 = np.zeros((K,n))
while np.abs(c - cold).sum() > 0.001:
    cold = c.copy()
    for i in range(0,K): #compute the squared distances
        dist2[i,:] = np.sum((Xmat - c[:,i].T)**2, 1)

    label = np.argmin(dist2,0) #assign the points to nearest centroid
    minvals = np.amin(dist2,0)
    for i in range(0,K): # recompute the centroids
        c[:,i] = np.mean(Xmat[np.where(label == i),:],1).reshape(1,2)

print('Loss = {:.3f}'.format(minvals.mean()))

```

Loss = 2.288





## 4.6.1 K- Phương tiện

```

▶ import numpy as np
Xmat = np.genfromtxt('clusterdata.csv', delimiter=',')
K = 3
n, D = Xmat.shape
c = np.array([[ -2.0 , -4 ,0] , [ -3 ,1 , -1]]) # initialize centers
cold = np.zeros(c.shape)
dist2 = np.zeros((K,n))
while np.abs(c - cold).sum() > 0.001:
    cold = c.copy()
    for i in range(0,K): #compute the squared distances
        dist2[i,:] = np.sum((Xmat - c[:,i].T)**2, 1)

    label = np.argmin(dist2,0) #assign the points to nearest centroid
    minvals = np.amin(dist2,0)
    for i in range(0,K): # recompute the centroids
        c[:,i] = np.mean(Xmat[np.where(label == i),:],1).reshape(1,2)

print('Loss = {:.3f}'.format(minvals.mean()))

```

↳ Loss = 2.288

## 4.6.1 K- Phương tiện

Hình 4.9: Kết quả của thuật toán K-mean được áp dụng cho dữ liệu trong Hình 4.4 . Dày vòng tròn đen là tâm và các đường chấm xác định ranh giới tế bào

Chúng tôi tìm thấy các trung tâm cụm

$c_1 = [1.9286, 3.0416]^T, c_2 = [-3.9237, 0.0131]^T$  và

$c_3 = [0, 5611, 1, 2980]^T$  , cho phép phân nhóm được mô tả trong Hình 4.9 . Tương ứng tổn thất (4,40) được tìm thấy là 2,288.

## 4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục

Như đã đề cập, việc tối thiểu hóa chính xác hàm mất mát (4.40) là khó hoàn thành thông qua các phương pháp tìm kiếm cục bộ tiêu chuẩn, chẳng hạn như gradient descent, như hàm đa phương thức. Tuy nhiên, ta có thể sử dụng tính năng tối ưu hóa toàn cầu các phương pháp như phương pháp CE hoặc SCO được thảo luận trong Phần 3.4.2 và 3.4.3

## 4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục

### Ví dụ 4.7 (Phân cụm qua CE)

Lấy cùng một tập dữ liệu như trong Ví dụ 4.6 và phân nhóm các điểm thông qua giảm thiểu tổn thất (4.40) bằng phương pháp CE. Mã Python dưới đây rất giống với mã trong Ví dụ 3.16, ngoại trừ việc bây giờ chúng ta đang xử lý với một bài toán tối ưu hóa sáu chiều. Hàm mất mát được thực hiện trong cụm hàm, về cơ bản sử dụng lại phép tính khoảng cách bình phương của K- mean mã trong Ví dụ 4.6. Chương trình CE thường quy về mức lỗi 2,287, tương ứng đi vào bộ thu nhỏ (toàn cục)  $c_1 = [1.9286, 3.0416]^T$ ,  $c_2 = [3.8681, 0.0456]^T$  và  $c_3 = [0, 5880, 1, 3526]^T$ , hơi khác so với các bộ giảm thiểu cục bộ cho K- mean thuật toán.

## 4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục



+ Code + Text



```
import numpy as np
np.set_printoptions ( precision =4)

Xmat = np. genfromtxt ('clusterdata.csv', delimiter =',')
K = 3
n, D = Xmat.shape

def Scluster(c):
    n, D = Xmat.shape
    dist2 = np.zeros ((K,n))
    cc = c.reshape(D,K)
    for i in range(0,K):
        dist2[i ,:] = np.sum(( Xmat - cc[:,i].T)**2, 1)
    minvals = np.amin(dist2 ,0)
    return minvals.mean ()

numvar = K*D
mu = np.zeros(numvar) # initialize centers
sigma = np.ones(numvar)*2
rho = 0.1
N = 500; Nel = int(N*rho); eps = 0.001
```

## 4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục

```

func = Scluster
best_trj = np.array(numvar)
best_perf = np.Inf
trj = np.zeros(shape =(N,numvar))

while(np.max(sigma)>eps):
    for i in range(0, numvar):
        trj[:,i] = (np.random.randn(N ,1)*sigma[i]+ mu[i]).reshape(N,)
    S = np.zeros(N)
    for i in range(0,N):
        S[i] = func(trj[i])

    sortedids = np.argsort(S) # from smallest to largest
    S_sorted = S[ sortedids ]
    best_trj = np.array(n)
    best_perf = np.Inf
    eliteids = sortedids [range(0,Nel)]
    eliteTrj = trj[eliteids ,:]
    mu = np.mean(eliteTrj ,axis =0)
    sigma = np.std(eliteTrj ,axis =0)

    if(best_perf >S_sorted [0]):
        best_perf = S_sorted [0]
        best_trj = trj[ sortedids [0]]

print( best_perf )
print(best_trj.reshape (2 ,3))

```

## 4.6.2 Phân cụm thông qua tối ưu hóa đa văn bản liên tục

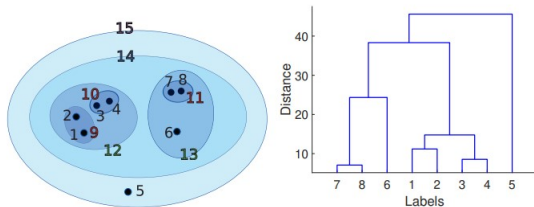
```
2.2871307773713108  
[[ 0.5878 -1.9286 -3.8683]  
 [-1.3526 -3.0414  0.0461]]
```

## 4.7 Phân cụm phân cấp (Hierarchical Clustering)

Thuật toán phân cụm K-means cho thấy cần phải cấu hình trước số lượng cụm cần phân chia. Ngược lại, phương pháp phân cụm phân cấp (Hierarchical Clustering) không yêu cầu khai báo trước số lượng cụm. Thay vào đó, thuật toán chỉ yêu cầu xác định trước thước đo về sự khác biệt giữa các cụm (không giao nhau), dựa trên sự khác biệt từng cặp giữa các quan sát trong hai cụm. Theo phương pháp này, chúng tạo ra những biểu diễn phân cấp trong đó các cụm ở mỗi cấp của hệ thống phân cấp được tạo bằng cách hợp nhất các cụm ở cấp độ thấp hơn bên dưới. Ở cấp thấp nhất, mỗi cụm chứa một quan sát. Ở cấp cao nhất, chỉ có một cụm chứa tất cả dữ liệu



## 4.7 Hàm phân phối thực nghiệm và ước lượng mật độ



**Hình:** Bên trái: một hệ thống phân cấp cụm gồm 15 cụm. Phải: biểu đồ dendrogram tương ứng

Thuật toán phân cụm phân cấp được xây dựng trên bộ dữ liệu có kích thước  $N$  thì sẽ trải qua tổng cộng  $N$  bước phân chia. Có hai chiến lược phân chia chính phụ thuộc vào chiều di chuyển trên biểu đồ dendrogram mà chúng ta sẽ tìm hiểu bên dưới: Chiến lược hợp nhất và chiến lược phân chia

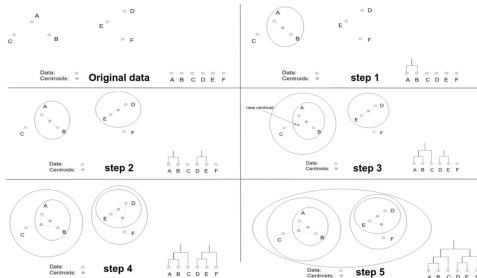
## 4.7.1 Chiến lược hợp nhất(agglomerative)

Chiến lược này sẽ đi theo chiều bottum-up (từ dưới lên trên). Quá trình phân cụm bắt đầu ở dưới cùng tại các node lá (còn gọi là leaf node hoặc termial node). Ban đầu mỗi quan sát sẽ được xem là một cụm tách biệt được thể hiện bởi một node lá. Ở mỗi level chúng ta sẽ tìm cách hợp một cặp cụm thành một cụm duy nhất nhằm tạo ra một cụm mới ở level cao hơn tiếp theo. Cụm mới này tương ứng với các node quyết định (non-leaf node). Như vậy sau khi hợp cụm thì số lượng cụm ít hơn. Một cặp được chọn để hợp nhất sẽ là những cụm trung gian không giao nhau.

## 4.7.1 Chiến lược hợp nhất(agglomerative)

Chiến lược hợp nhất sẽ bắt đầu biểu diễn mỗi quan sát là một cụm đơn lẻ. Giả định chúng ta có  $N$  quan sát, thuật toán cần thực hiện  $N-1$  bước để hợp nhất hai nhóm có khoảng cách gần nhất lại với nhau và đồng thời giảm số lượng cụm trước khi chúng đạt được tới node gốc gồm toàn bộ các quan sát. Ta có ví dụ minh họa về chiến lược hợp nhất

## 4.7.1 Chiến lược hợp nhất(agglomerative)



**Hình:** Hình minh họa các bước được thực hiện trên thuật toán phân cụm phân cấp sử dụng chiến lược hợp nhất đối với 6 điểm dữ liệu (A,B,C,D,E,F) . Chấm tròn thể hiện cho các điểm dữ liệu, chấm tròn có dấu x ở giữa là tâm của các cụm. Các đường elipse bao ngoài thể hiện cho các điểm được phân về cùng một cụm. Ở bên phải dưới cùng của mỗi hình là đồ thị dendrogram thể hiện sự gộp nhóm.

## 4.7.1 Chiến lược hợp nhất (agglomerative)

Bộ dữ liệu ở hình trên bao gồm 6 điểm nên sẽ trải qua 5 bước dữ liệu để nhóm dữ liệu. Thứ tự nhóm sẽ như sau:

Step 1: Dựa trên khoảng cách gần nhất giữa các điểm chúng ta sẽ nhóm 2 điểm A và B thành 1 cụm. Khi đó điểm đại diện cho một cụm (A,B) sẽ là trung bình cộng giữa hai điểm A và B, được thể hiện bằng dấu  $\oplus$  giữa A và B trên hình.

Step 2: Lựa chọn ngẫu nhiên một điểm chưa được gộp cụm, chẳng hạn điểm D. Đo khoảng cách tới các điểm còn lại và với tâm cụm (A,B) ta sẽ thu được khoảng cách  $d(D,E)$  là nhỏ nhất. Như vậy ta sẽ thu được một cụm (D,E).

Step 3: Xuất phát từ điểm C, ta đo khoảng cách tới các tâm cụm (A,B) và (D,E) và tới điểm F. Khoảng cách gần nhất là  $d(C,(A,B))$  nên ta nhóm C vào cụm (A,B) để thu được cụm mới

## 4.7.1 Chiến lược hợp nhất(agglomerative)

Step 4: Xuất phát từ  $F$  ta đo khoảng cách tới các tâm cụm  $(A,B,C)$  và  $(D,E)$ . Điểm  $F$  gần cụm  $(D,E)$  hơn nên sẽ được gộp vào thành cụm  $(D,E,F)$ . .

Step 5: Gộp cả 2 cụm  $(A,B,C)$  và  $(D,E,F)$  lại xuất phát từ node lá, thuật toán gộp dần thành các cụm theo chiều từ dưới lên trên. Sau đó sẽ thực hiện truy hồi việc gộp cụm (cụm ở đây có thể gồm một điểm hoặc nhiều điểm). Khoảng cách giữa hai cụm được đo lường thông qua một thước đo sẽ được làm rõ hơn ở bên dưới, trong ví dụ này chính là khoảng cách trong không gian euclidean giữa tâm của mỗi cụm. Trong đó tâm cụm được xác định bằng trung bình cộng của các quan sát bên trong cụm.  $(D,E,F)$  ta thu được cụm cuối cùng là node gốc bao trùm toàn bộ dữ liệu.

## 4.7.1 Chiến lược hợp nhất(agglomerative)

Chúng qui lại xuất phát từ node lá, thuật toán gộp dần thành các cụm theo chiều từ dưới lên trên. Sau đó sẽ thực hiện truy hồi việc gộp cụm (cụm ở đây có thể gồm một điểm hoặc nhiều điểm).

Khoảng cách giữa hai cụm được đo lường thông qua một thước đo sẽ được làm rõ hơn ở bên dưới, trong ví dụ này chính là khoảng cách trong không gian euclidean giữa tâm của mỗi cụm. Trong đó tâm cụm được xác định bằng trung bình cộng của các quan sát bên trong cụm.

## 4.7.2 Khoảng cách giữa hai cụm

Trước hết ta cùng ôn lại khoảng cách euclidean, chính là độ dài đoạn thẳng nối trực tiếp hai điểm trong không gian euclidean

$$\text{dist}(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}$$

Giả định tại một level cụ thể trong biểu đồ dendrogram chúng ta có hai cụm trung gian không trùng nhau là  $S_1 = \{x_i, i = I\}$  và  $S_2 = \{x_j, j = J\}$ . Khoảng cách giữa hai cụm chính là sự khác biệt giữa chúng. Có những phương pháp giúp xác định khoảng cách giữa hai cụm như sau:



## 4.7.2 Khoảng cách giữa hai cụm

Single linkage : Phương pháp này đo lường sự khác biệt giữa hai cụm bằng cách lấy ra cặp điểm gần nhất giữa hai cụm. Độ đo sự khác biệt được tính theo công thức:

$$d_{min}(I, J) := \min_{i \in I, j \in J} \text{dist}(x_i, x_j)$$

Complete linkage : Phương pháp này đo lường sự khác biệt giữa hai cụm bằng cách lấy ra hai cặp điểm xa nhau nhất giữa hai cụm:

$$d_{max}(I, J) := \max_{i \in I, j \in J} \text{dist}(x_i, x_j)$$

## 4.7.2 Khoảng cách giữa hai cụm

Group average. : Khoảng cách trung bình giữa các cụm. Lưu ý rằng điều này phụ thuộc vào kích thước cụm:

$$d_{avg}(I, J) := \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} \text{dist}(x_i, x_j)$$

Ward' linkage. : Khoảng cách trung bình giữa các cụm. Lưu ý rằng điều này phụ thuộc vào kích thước cụm:

$$d_{Ward}(I, J) := \sum_{k \in I \cup J} \|x_k - \bar{x}_{I \cup J}\|^2 - \left( \sum_{i \in I} \|x_i - \bar{x}_I\|^2 + \sum_{j \in J} \|x_j - \bar{x}_J\|^2 \right)$$

## 4.7.2 Khoảng cách giữa hai cụm

Cả bốn phương pháp ward linkage, single linkage, complete linkage, group average đều giúp tạo ra một thước đo về sự không tương đồng hay chính là khoảng cách giữa hai cụm. Khi giữa các cụm có sự tách biệt thể hiện qua phân phối dữ liệu và đường biên phân chia rõ rệt thì kết quả trả về  $d(S_1, S_2)$  về đều thu được lớn và trái lại. Tuy nhiên phương pháp single linkage và complete linkage thường bị ảnh hưởng bởi những điểm dữ liệu outliers. Chẳng hạn hai cụm rất cách xa nhau nhưng do hai điểm outliers của chúng lại rất gần nhau có thể trả về một khoảng cách theo single linkage rất bé. Một tình huống khác, khi hai cụm rất gần nhau nhưng do hai điểm outliers của chúng rất xa nên khoảng cách được đo theo complete linkage lại rất lớn. Trong khi đó ward linkage và group average ít bị ảnh hưởng bởi outliers hơn. Tuy nhiên ward linkage lại chỉ có thể hoạt động khi các điểm dữ liệu tồn tại trong không gian euclidean.

## 4.7.2 Khoảng cách giữa hai cụm

Chú ý: Trong triển khai phần mềm, hàm Ward' linkage thường được thay đổi tỷ lệ bằng cách nhân nó với hệ số 2. Bằng cách này, khoảng cách giữa các cụm một điểm  $\{x_i\}$  và  $\{x_j\}$  là khoảng cách Euclid bình phương  $\|x_i - x_j\|^2$ .

Sau khi chọn một khoảng cách trên X và một tiêu chí liên kết, một thuật toán phân cụm tích hợp chung sẽ tiến hành theo cách "greedy" sau đây.

Thuật toán: Greedy Agglomerative Clustering.

Đầu vào: Hàm khoảng cách dist, hàm liên kết d, số lượng cụm K.

Đầu ra: Bộ nhãn cho cây

## 4.7.2 Khoảng cách giữa hai cụm

1. Khởi tạo bộ nhận dạng cụm:  $I = \{1, \dots, n\}$
2. Khởi tạo các bộ nhãn tương ứng:  $L_i = \{i\}, i \in I$ .
3. Khởi tạo ma trận khoảng cách  $D = [d_{ij}]$  với  $d_{ij} = d(\{i\}, \{j\})$ .
4. for  $k = n + 1$  to  $2n - K$  do
5. Tìm  $i$  và  $j > i$  trong  $I$  sao cho  $d_{ij}$  là nhỏ nhất.
6. Tạo bộ nhãn mới  $L_k := L_i \cup L_j$
7. Thêm số nhận dạng mới  $k$  vào  $I$  và xóa các số nhận dạng cũ  $i$  và  $j$  khỏi  $I$
8. Cập nhật ma trận khoảng cách  $D$  đối với các định danh  $i, j$  và  $k$ .
9. return  $L_i, i = 1, \dots, 2n - K$

### 4.7.3 Chiến lược phân chia (divisive)

Chiến lược này sẽ thực hiện theo chiều top-down. Tức là phân chia bắt đầu từ node gốc của đồ thị. Node gốc bao gồm toàn bộ các quan sát, tại mỗi level chúng ta phân chia một cách đệ qui các cụm đang tồn tại tại level đó thành hai cụm mới. Phép phân chia được tiến hành sao cho tạo thành hai cụm mới mà sự tách biệt giữa chúng là lớn nhất. Sự tách biệt này sẽ được đo lường thông qua một thước đo khoảng cách mà ta sẽ tìm hiểu kĩ hơn bên dưới. Đầu tiên thuật toán sẽ chọn ra một điểm từ toàn bộ tập dữ liệu  $S$  sao cho điểm này thoả mãn điều kiện trung bình khoảng cách từ điểm đó tới toàn bộ những điểm còn lại là nhỏ nhất. Chúng ta đưa điểm này vào tập  $S_1$ , tập còn lại gồm  $N - 1$  điểm là tập  $S_2$ .

### 4.7.3 Chiến lược phân chia (divisive)

Tiếp theo ta sẽ thực hiện các lượt phân chia sao cho mỗi một lượt lựa chọn ra một điểm  $x_i$  từ tập  $S_2$  đưa sang  $S_1$ . Điểm này cần thoả mãn hai điều kiện:

- Trung bình khoảng cách từ điểm đó tới toàn bộ các điểm còn lại trong  $S_1$  phải là nhỏ nhất. Điều đó có nghĩa là  $x_i$  là điểm tách biệt nhất so với phần còn lại của  $S_1$

$$x_i = \operatorname{argmax}_{x_i} \frac{1}{|S_1| - 1} \sum_{j=1, j \neq i}^{|S_1|} d(x_i, x_j)$$

### 4.7.3 Chiến lược phân chia (divisive)

- Khoảng cách tối thiểu từ  $x_i$  tới các điểm trong  $S_2$  phải lớn hơn khoảng cách tối thiểu tới các điểm trong  $S_1$ . Điều này nhằm mục đích khiến cho điểm  $x_i$  phải gần với cụm  $S_2$  hơn cụm  $S_1$ .

$$d(x_i, S_1) \geq d(x_i, S_2)$$

Trong đó:

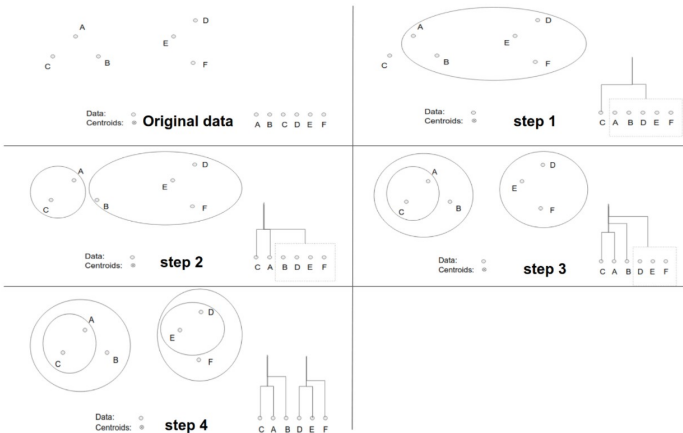
$$(x_i, S_k) = \min_{x_j \in S_k} d(x_i, x_j)$$

Quá trình chuyển cụm sẽ kết thúc khi không còn điểm nào thoả mãn hai điều kiện trên. Khi đó chúng ta lại thực hiện đệ quy lại quá trình trên trên từng tập  $S_1$  và  $S_2$ .



## 4.7.3 Chiến lược phân chia (divisive)

Chúng ta cùng xem ví dụ bên dưới để hiểu rõ hơn vấn đề này:



### 4.7.3 Chiến lược phân chia (divisive)

Bước 1: chúng ta sẽ lựa chọn ra điểm là điểm đầu tiên thuộc cụm mới dựa trên khoảng cách so với các điểm còn lại là xa nhất. Khi đó ta thu được tập  $S_1 = \{C\}$  và  $S_2 = \{A, B, D, E, F\}$ .

Bước 2: lựa chọn trong số các điểm thuộc  $S_2$  ra điểm mà có khoảng cách xa nhất so với những điểm còn lại sao cho điểm này gần với C hơn so với các điểm thuộc tập  $S_2$ , đó chính là điểm A. Di chuyển điểm này sang  $S_1$ .

Bước 3: lại tiếp tục thực hiện như vậy và lựa chọn được điểm để đưa sang  $S_1$ .

Bước 4: dừng quá trình chuyển cụm cho các điểm thuộc  $S_2$  vì thuật toán đã đạt sự hội tụ về hai cụm. Khi đó ta lại tiếp tục tiến hành đệ qui thuật toán trên từng cụm con.

## 4.7.4 Thuật toán Lance-Williams

Ban đầu, ma trận khoảng cách  $D$  chứa khoảng cách (liên kết) giữa các cụm một điểm chứa một trong các điểm dữ liệu  $x_1, \dots, x_n$ , và do đó với các số nhận dạng  $1, \dots, n$  làm khoảng cách ngắn nhất tương ứng với tra cứu bảng trong  $D$ . Khi các cụm gần nhất được tìm thấy, chúng được hợp nhất thành một cụm mới và số nhận dạng mới  $k$  (số nguyên dương nhỏ nhất chưa được sử dụng làm định danh) được gán cho cụm này. Các số nhận dạng cũ  $i$  và  $j$  bị xóa khỏi tập nhận dạng cụm  $I$ . Sau đó, ma trận  $D$  được cập nhật bằng cách thêm cột và hàng thứ  $k$  chứa khoảng cách giữa  $k$  và  $m \in I$ . Bước cập nhật này có thể khá tốn kém về mặt tính toán, nếu kích thước cụm lớn và khoảng cách liên kết giữa các cụm phụ thuộc vào tất cả các điểm trong cụm. May mắn thay, đối với nhiều hàm liên kết, ma trận  $D$  có thể được cập nhật một cách hiệu quả.

## 4.7.4 Thuật toán Lance-Williams

Giả sử rằng tại một số giai đoạn trong thuật toán, các cụm  $I$  và  $J$ , với số nhận dạng  $i$  và  $j$ , được hợp nhất thành một cụm  $K = I \cup J$  với số nhận dạng  $k$ . Cho  $M$ , với số nhận dạng  $m$ , là một cụm đã được gán trước đó. Quy tắc cập nhật về khoảng cách liên kết  $d_{km}$  giữa  $K$  và  $M$  được gọi là cập nhật Lance – Williams nếu nó có thể được viết dưới dạng

$$d_{km} = \alpha d_{im} + \beta d_{jm} + \gamma d_{ij} + \delta |d_{im} - d_{jm}|,$$

trong đó  $\alpha, \dots, \delta$  chỉ phụ thuộc vào các đặc điểm đơn giản của các cụm liên quan, chẳng hạn như số lượng phần tử trong các cụm.

## 4.7.4 Thuật toán Lance-Williams

| Linkage    | $\alpha$                            | $\beta$                             | $\gamma$                       | $\delta$ |
|------------|-------------------------------------|-------------------------------------|--------------------------------|----------|
| Single     | $1/2$                               | $1/2$                               | $0$                            | $-1/2$   |
| Complete   | $1/2$                               | $1/2$                               | $0$                            | $1/2$    |
| Group avg. | $\frac{n_i}{n_i + n_j}$             | $\frac{n_j}{n_i + n_j}$             | $0$                            | $0$      |
| Ward       | $\frac{n_i + n_j}{n_i + n_j + n_m}$ | $\frac{n_j + n_m}{n_i + n_j + n_m}$ | $\frac{-n_m}{n_i + n_j + n_m}$ | $0$      |

**Hình:** Các hằng số cho quy tắc cập nhật Lance-Williams cho các hàm liên kết khác nhau với  $n_i, n_j, n_m$  biểu thị số phần tử trong các cụm tương ứng

## Ví dụ 4.8 (Chiến lược hợp nhất)

Cách 1:

```
import numpy as np
from scipy.spatial.distance import cdist

def update_distances(D,i,j, sizes): # tính toán khoảng cách cho cụm đã hợp nhất
    n = D.shape[0]
    d = np.inf * np.ones(n+1)
    for k in range(n): # cập nhật khoảng cách
        d[k] = ((sizes[i]+sizes[k])*D[i,k] +
                (sizes[j]+sizes[k])*D[j,k] -
                sizes[k]*D[i,j])/(sizes[i] + sizes[j] + sizes[k]))

    infs = np.inf * np.ones(n) # mảng vô cực
    D[i,:],D[:,i],D[j,:],D[:,j] = infs,infs,infs,infs #
    new_D = np.inf * np.ones((n+1,n+1))
    new_D[0:n,0:n] = D # sao chép ma trận cũ vào new_D
    new_D[-1,:], new_D[:,-1] = d,d # thêm hàng và cột mới
    return new_D

def agg_cluster(X):
    n = X.shape[0]
    sizes = np.ones(n)
    D = cdist(X, X, metric = 'sqeuclidean') # khởi tạo ma trận khoảng cách
    np.fill_diagonal(D, np.inf * np.ones(D.shape[0]))
    Z = np.zeros((n-1,4)) #ma trận liên kết mã hóa cây phân cấp
    for t in range(n-1):
        i,j = np.unravel_index(D.argmin(), D.shape) # cập giảm thiểu
        sizes = np.append(sizes, sizes[i] + sizes[j])
        Z[t,:]=np.array([i, j, np.sqrt(D[i,j]), sizes[-1]])
        D = update_distances(D, i,j, sizes) # cập nhận lại ma trận khoảng cách
    return Z
```


Hình: Xây dựng thuật toán Ward linkage

## Ví dụ 4.8(Chiến lược hợp nhất)

```
import scipy.cluster.hierarchy as h
#vẽ biểu đồ dendrogram
X = np.genfromtxt('clusterdata.csv',delimiter=',') # đọc dữ liệu
Z = agg_cluster(X) # Hình thành ma trận liên kết
plt.figure(figsize=(20, 7))
h.dendrogram(Z) #tạo ra biểu đồ dendrogram(Z)
```

Hình: Vẽ biểu đồ dendrogram

## Ví dụ 4.8 (Chiến lược hợp nhất)

```
 #Biểu đồ hóa dữ liệu
cl = h.fcluster(Z, criterion = 'maxclust', t=3)
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 7)), plt.clf()
cols = ['red', 'green', 'blue']
colors = [cols[i-1] for i in cl]
plt.scatter(X[:,0], X[:,1],c=colors)
plt.show()
```

Hình: Vẽ biểu đồ hóa dữ liệu



## Ví dụ 4.8 (Chiến lược hợp nhất)

### Cách 2

```
[21] data = pd.read_csv("clusterdata.csv")
      print(data.shape)

(299, 2)

[23] #Chuẩn hóa dữ liệu để loại bỏ sự khác biệt về mặt đơn vị giữa các chiều.
      std = MinMaxScaler()
      X_std = std.fit_transform(data)

      plt.figure(figsize=(20, 7))
      dend = shc.dendrogram(shc.linkage(data, method = 'ward'))
```

Hình:

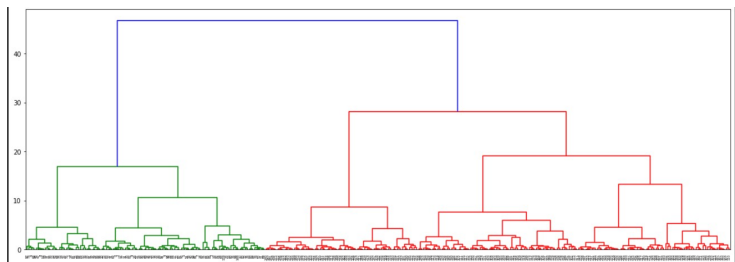
## Ví dụ 4.8 (Chiến lược hợp nhất)

```
▶ AgglomerativeClustering(  
    n_clusters=2,  
    affinity='euclidean',  
    compute_full_tree='auto',  
    linkage='ward',  
    distance_threshold=None,  
    compute_distances=False)  
  
📄 AgglomerativeClustering()  
  
[35] from sklearn.cluster import AgglomerativeClustering  
  
    cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')  
    labels = cluster.fit_predict(X_std)  
  
[53] def _plot_kmean_scatter(X, labels):  
    # lựa chọn màu sắc  
    num_classes = len(np.unique(labels))  
    palette = np.array(sns.color_palette("hls", num_classes))  
    # vẽ biểu đồ scatter  
    fig = plt.figure(figsize=(20, 7))  
    sc = plt.scatter(X[:,0], X[:,1], c=palette[labels.astype(np.int)])  
    _plot_kmean_scatter(X_std, labels)
```

Hình:

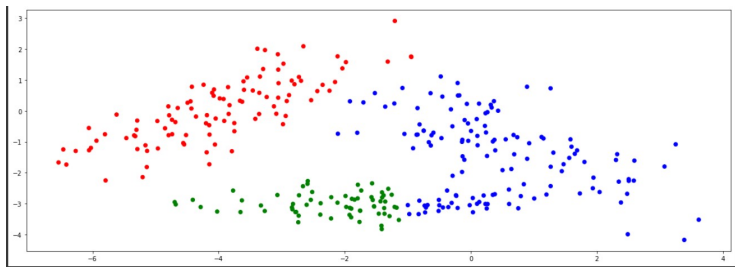
## Ví dụ 4.8 (Chiến lược hợp nhất)

Xuất kết quả:



Hình: Biểu đồ dendrogram

## Ví dụ 4.8 (Chiến lược hợp nhất)



Hình: Biểu đồ hóa dữ liệu

## Ví dụ 4.9 (Chiến lược phân chia)

Mã Python sau được sử dụng để chuyển một tập dữ liệu nhỏ (có kích thước 300) thành hai phần theo liên kết trung bình nhóm tối đa. Nó sử dụng một thuật toán entropy chéo gần tương tự như thuật toán được trình bày trong Ví dụ 3.19. Cho vectơ xác suất  $\{\pi, i = 1, \dots, n\}$ , thuật toán tạo ra một ma trận  $n \times n$  gồm các biến ngẫu nhiên Bernoulli với xác suất thành công  $\pi$  cho cột  $i$ . Đối với mỗi hàng, các số 0 và 1 chia tập hợp chỉ mục thành hai cụm và mức độ liên kết trung bình tương ứng được tính toán. Sau đó, ma trận được sắp xếp theo hàng theo các khoảng cách này. Cuối cùng, xác suất  $\pi$  được cập nhật theo giá trị trung bình của 10% hàng tốt nhất. Quá trình được lặp lại cho đến khi  $\pi$  suy biến thành vectơ nhị phân. Điều này sau đó trình bày (gần đúng) giải pháp.

## Ví dụ 4.9 (Chiến lược phân chia)

```
def S(x,D):
    V1 = np.where(x==0)[0] # {V1,V2} là phân vùng
    V2 = np.where(x==1)[0]
    tmp = D[V1]
    tmp = tmp[:,V2]
    return np.mean(tmp)

def maxcut(D,N,eps,rho,alpha):
    n = D.shape[1]
    Ne = int(rho*N)
    p = 1/2*np.ones(n)
    p[0] = 1.0
    while (np.max(np.minimum(p,np.subtract(1,p))) > eps):
        x = np.array(np.random.uniform(0,1, (N,n))<=p, dtype=np.int64)
        sx = np.zeros(N)
        for i in range(N):
            sx[i] = S(x[i],D)

        sortSX = np.flip(np.argsort(sx))
        elIds = sortSX[0:Ne]
        elites = x[elIds]
        pnew = np.mean(elites, axis=0)
        p = alpha*pnew + (1.0-alpha)*p

    return np.round(p)
```

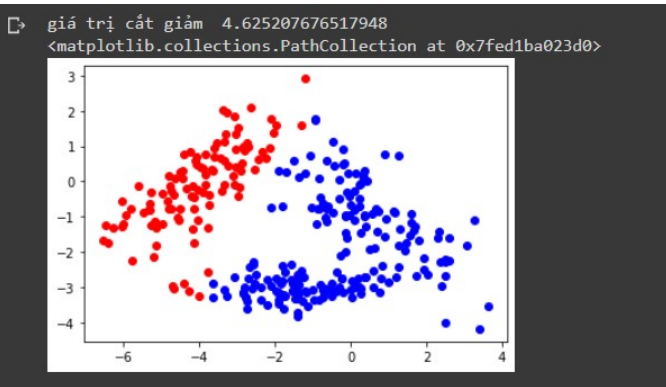
Hình: Tạo phân vùng và phân chia dữ liệu

## Ví dụ 4.9 (Chiến lược phân chia)

```
▶ Xmat = genfromtxt('clusterdata.csv', delimiter=',')  
n = Xmat.shape[0]  
D = squareform(pdist(Xmat))  
# CE  
pout = maxcut(D,1000,10**2,0.1, 0.9);  
cutval = S(pout,D)  
print("giá trị cắt giảm ",cutval)  
#plot  
V1 = np.where(pout==0)[0]  
xblue = Xmat[V1]  
V2 = np.where(pout==1)[0]  
xred = Xmat[V2]  
plt.scatter(xblue[:,0],xblue[:,1], c="blue")  
plt.scatter(xred[:,0],xred[:,1], c="red")
```

Hình: Tạo phân vùng và xây dựng hàm tính giá trị cắt giảm

## Ví dụ 4.9 (Gán các giá trị và gọi hàm)



Hình: Kết quả



## 4.8 Phân tích thành phần chính (PCA)

Ý tưởng chính của phân tích thành phần chính (PCA) là làm giảm kích thước của một tập dữ liệu bao gồm nhiều biến. PCA là một cơ chế giảm tính năng (hoặc trích xuất tính năng), giúp chúng ta xử lý dữ liệu chiều cao với nhiều các tính năng thông minh hơn để giải thích.

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Xem xét một phân phối chuẩn  $d$ -chiều với vectơ trung bình 0 và ma trận hiệp phương sai  $\Sigma$ . Hàm mật độ xác suất tương ứng (xem (2.33)) là

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} x^T \Sigma^{-1} x}, \quad x \in R^d.$$

Nếu chúng ta vẽ nhiều mẫu iid từ hàm mật độ xác suất này, các điểm gần như sẽ có dạng ellipsoid, như minh họa trong Hình 3.1, và tương ứng với các đường bao của  $f$ : tập hợp các điểm  $x$  sao cho  $x^T \Sigma^{-1} x = c$ ,  $c \geq 0$ .

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Đặc biệt, xem xét Ellipsoid

$$x^T \Sigma^{-1} x = 1, \quad x \in R^d \quad (4.42)$$

Giả sử  $\Sigma = BB^T$ , ví dụ B là ma trận Cholesky (thấp hơn). Sau đó, như được giải thích trong Ví dụ A.5, ellipsoid (4.42) cũng có thể được xem như là phép biến đổi tuyến tính của hình cầu đơn vị d-chiều qua ma trận B. Hơn nữa, các trục chính của ellipsoid có thể được tìm thấy thông qua một phép phân rã giá trị đơn lẻ (SVD) của B (hoặc  $\Sigma$ ).

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Đặc biệt, giả sử rằng SVD của  $B$  là  $B = UDV^T$  (lưu ý là SVD của  $\Sigma$  sau đó được  $UD^2U^T$ ).

Các cột của ma trận  $UD$  tương ứng với các trục chính của ellipsoid và độ lớn tương đối của các trục được cho bởi các phần tử của ma trận đường chéo  $D$ . Nếu một số độ lớn này nhỏ so với các độ lớn khác, thì kích thước sẽ giảm của không gian có thể đạt được bằng cách chiếu mỗi điểm  $x \in R^d$  lên không gian con được kéo dài bởi các cột chính (giả sử  $k \ll d$ ) của  $U$  - gọi là các thành phần chính. Giả sử không mất tính tổng quát rằng  $k$  thành phần chính đầu tiên được cho bởi  $k$  cột đầu tiên của  $U$  và đặt  $U_k$  là ma trận  $d \times k$  tương ứng.

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Với cơ sở tiêu chuẩn  $\{e_i\}$ , vectơ  $x = x_1e_1 + \dots + x_de_d$  được biểu diễn bằng vectơ d-chiều  $[x_1, \dots, x_d]^T$ . Đối với cơ sở trục chuẩn  $\{u_i\}$  được tạo thành bởi các cột của ma trận  $U$ , biểu diễn của  $x$  là  $U^T x$ . Tương tự, hình chiếu của bất kỳ điểm  $x$  lên không gian con bao trùm bởi  $k$  vectơ chính đầu tiên được biểu diễn bằng vectơ  $k$ -chiều  $U^T x$ , đối với cơ sở trục chuẩn được tạo thành bởi các cột của  $U_k$ . Vì vậy, ý tưởng là nếu một điểm  $x$  nằm gần với hình chiếu  $U_k U_k^T x$  của nó, chúng ta có thể biểu diễn nó qua  $k$  số thay vì  $d$ , sử dụng các đặc trưng kết hợp được cho bởi  $k$  thành phần chính.

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Ví dụ 4.10 (thành phần chính), xét ma trận  $\Sigma = \begin{bmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{bmatrix}$ , có

thể được viết thành  $\Sigma = BB^T$ ,

với  $B = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$ ,

Hình 4.12 mô tả ellipsoid  $x^T \Sigma x = 1$ , có thể thu được bằng cách biến đổi tuyến tính các điểm trên hình cầu đơn vị nhờ ma trận B. Các trục chính và kích thước của ellipsoid được tìm thấy thông qua sự phân rã giá trị đơn lẻ  $B = UDV^T$ ,

## 4.8.1 Động lực: Trục chính của một Ellipsoid

trong đó  $U$  và  $D$  là

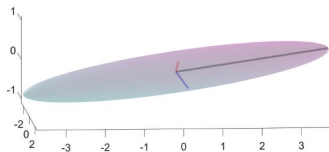
$$U = \begin{bmatrix} 0.8460 & 0.4828 & 0.2261 \\ 0.4973 & -0.5618 & -0.6611 \\ 0.1922 & -0.6718 & 0.7154 \end{bmatrix}$$

$$\text{và } D = \begin{bmatrix} 4.4027 & 0 & 0 \\ 0 & 0.7187 & 0 \\ 0 & 0 & 0.3160 \end{bmatrix}. \text{ Các cột của } U \text{ cho biết hướng}$$

của các trục chính của ellipsoid, và các phần tử khác nhau của  $D$  chỉ ra độ lớn tương đối của các trục chính. Chúng ta thấy rằng thành phần chính đầu tiên được cho bởi cột đầu tiên của  $U$  và thành phần chính thứ hai được cho bởi cột thứ hai của  $U$ .

## 4.8.1 Động lực: Trục chính của một Ellipsoid

Hình chiếu của điểm  $x = [1.052, 0.6648, 0.2271]^T$  lên không gian 1 chiều được bao bởi thành phần chính thứ nhất  $u_1 = [0.8460, 0.4972, 0.1922]^T$  là  $z = u_1^T x = [1.0696, 0.6287, 0.2429]^T$ . Đối với vectơ cơ sở  $u_1$ ,  $z$  được biểu diễn bằng số  $u_1^T z = 1.2643$ . Tức là,  $z = 1.2643u_1$ .



**Hình:** 4.12. Một ellipsoid "ván lướt sóng" trong đó một trục chính lớn hơn đáng kể so với hai trục còn lại.



## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Trong cài đặt trên, chúng ta không xem xét bất kỳ tập dữ liệu nào được rút ra từ hàm phân phối xác suất đa biến  $f$ . Toàn bộ phân tích dựa trên đại số tuyến tính. Trong phân tích thành phần chính (PCA), chúng ta bắt đầu với dữ liệu  $x_1, \dots, x_n$ , trong đó mỗi  $x$  là  $d$ -chiều. PCA không yêu cầu giả định về cách thu thập dữ liệu, nhưng để tạo liên kết với phần trước, chúng ta có thể nghĩ về dữ liệu khi iid lấy từ một hàm phân phối xác suất thông thường đa biến. Hãy để chúng ta thu thập dữ liệu trong ma trận  $X$  theo cách thông thường; đó là,

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}. \text{ Ma trận } X \text{ sẽ là đầu vào của}$$

PCA. Theo cài đặt này, dữ liệu bao gồm các điểm trong không gian  $d$ -chiều và mục tiêu của chúng ta là trình bày dữ liệu bằng cách sử dụng  $n$  vectơ đặc trưng của kích thước  $k < d$ . Theo phần trước, chúng ta giả định rằng phân phối cơ bản của dữ liệu có vectơ kỳ vọng 0. Trong thực tế, điều này có nghĩa là trước khi áp dụng PCA, dữ liệu cần được tập trung bằng cách trừ đi giá trị trung bình của cột trong mỗi cột:

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

$x'_{ij} = x_{ij} - \bar{x}_j$ , trong đó  $\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$ . Từ bây giờ, chúng ta giả

định rằng dữ liệu đến từ phân phối d-chiều tổng quát với vectơ trung bình 0 và một số ma trận hiệp phương sai  $\Sigma$ . Theo định nghĩa, ma trận hiệp phương sai  $\Sigma$  bằng với kỳ vọng của ma trận ngẫu nhiên  $XX^T$  và có thể được ước tính từ dữ liệu  $x_1, \dots, x_n$  qua

giá trị trung bình mẫu  $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n x_i x_i^T = \frac{1}{n} X^T X$ .

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Khi  $\hat{\Sigma}$  là một ma trận hiệp phương sai, chúng ta có thể tiến hành phân tích  $\hat{\Sigma}$  tương tự như chúng ta đã làm cho  $\Sigma$  trong phần trước. Cụ thể, giả sử  $\hat{\Sigma} = UD^2U^T$  là SVD của  $\hat{\Sigma}$  và đặt  $U_k$  là ma trận có các cột là  $k$  thành phần chính; nghĩa là,  $k$  cột của  $U$  tương ứng với các phần tử đường chéo lớn nhất trong  $D^2$ . Lưu ý rằng chúng ta đã sử dụng  $D^2$  thay vì  $D$  để tương đồng với phần trước. Phép biến đổi  $z = U_k U_k^T x_i$  ánh xạ mỗi vectơ  $x_i \in R^d$  (do đó, với  $d$  đặc điểm) thành một vectơ  $z_i \in R^d$  nằm trong không gian con được kéo dài bởi các cột của  $U_k$ .

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Theo cơ sở này, điểm  $z_i$  có biểu diễn  $z_i = U_k^T (U_k U_k^T x_i) = U_k^T x_i \in R^k$  (do đó có  $k$  đặc điểm). Ma trận hiệp phương sai tương ứng của  $z_i, i = 1, \dots, n$  là đường chéo. Các phần tử đường chéo  $d_{ll}$  của  $D$  có thể được hiểu là độ lệch chuẩn của dữ liệu theo hướng của các thành phần chính. Đại lượng  $v = \sum_{l=1} d_{ll}^2$  (nghĩa là dấu vết của  $D^2$ ) do đó là một thước đo cho lượng phương sai trong dữ liệu. Tỷ lệ  $d_{ll}^2/v$  cho biết mức độ phương sai trong dữ liệu được giải thích bằng thành phần chính thứ  $l$ .

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Một cách khác để xem xét PCA là xem xét câu hỏi: Làm thế nào chúng ta có thể chiếu dữ liệu lên không gian con  $k$ -chiều một cách tốt nhất theo cách mà tổng bình phương khoảng cách giữa các điểm được chiếu và các điểm gốc là nhỏ nhất? Từ Phần A.4, chúng ta biết rằng bất kỳ phép chiếu trực giao nào lên không gian con  $k$ -chiều  $V_k$  có thể được biểu diễn bằng một ma trận  $U_k U_k^T$ , trong đó  $U_k = [u_1, \dots, u_k]$  và  $u_\ell, \ell = 1, \dots, k$  là các vectơ trực giao có độ dài 1 kéo dài  $V_k$ . Do đó, câu hỏi trên có thể được xây dựng dưới

dạng phương trình giảm thiểu: 
$$\min_{u_1, \dots, u_k} \sum_{i=1}^n \|x_i - U_k U_k^T x_i\|^2 \quad (4.43)$$

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Bây giờ quan sát

$$\begin{aligned}
 \frac{1}{n} \sum_{i=1}^n \|x_i - U_k U_k^T x_i\|^2 &= \frac{1}{n} \sum_{i=1}^n (x_i^T - x_i^T U_k U_k^T)(x_i - x_i U_k U_k^T) \\
 &= \frac{1}{n} \sum_{i=1}^n \|x_i\|^2 - \frac{1}{n} \sum_{i=1}^n (x_i^T U_k U_k^T x_i) = c - \frac{1}{n} \sum_{i=1}^n \sum_{l=1}^k \text{tr}(x_i^T u_l u_l^T x_i) , \\
 &= c - \frac{1}{n} \sum_{l=1}^k \sum_{i=1}^n u_l^T x_i x_i^T u_l = c - \sum_{l=1}^k u_l^T \hat{\Sigma} u_l,
 \end{aligned}$$

trong đó chúng ta đã sử dụng tính chất tuần hoàn của một vết (Định lý A.1) và thực tế là  $U_k U_k^T$  có thể được viết dưới

dạng  $\sum_{l=1}^k u_l u_l^T$  theo đó bài toán tối thiểu hóa (4.43)

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

tương đương với bài toán tối đa hóa  $\max_{u_1, \dots, u_k} \sum_{l=1}^k u_l^T \hat{\Sigma} u_l$  (4.44). Cực

đại này có thể lớn nhất là  $\sum_{l=1}^k d_{ll}^2$  và đạt được chính xác

khi  $u_1, \dots, u_k$  là k thành phần chính đầu tiên của  $\hat{\Sigma}$ .

Ví dụ 4.11 (Phân tích giá trị đơn lẻ) Tập dữ liệu sau đây bao gồm các mẫu phụ thuộc từ phân bố Gaussian ba chiều với vectơ trung bình 0 và ma trận hiệp phương sai  $\hat{\Sigma}$  được cho trong ví dụ 4.10 :



## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

$$X = \begin{bmatrix} 3.1209 & 1.7438 & 0.5479 \\ -2.6628 & -1.5310 & -0.2763 \\ 3.7284 & 3.0648 & 1.8451 \\ 0.4203 & 0.3553 & 0.4268 \\ -0.7155 & -0.6871 & -0.1414 \\ 5.8728 & 4.0180 & 1.4541 \\ 4.8163 & 2.4799 & 0.5637 \\ 2.6948 & 1.2384 & 0.1533 \\ -1.1376 & -0.4677 & -0.2219 \\ -1.2452 & -0.9942 & -0.4449 \end{bmatrix} . \text{ Sau khi thay thế } X \text{ bằng}$$

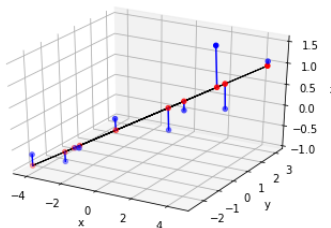
phiên bản ở giữa, SVD  $UD^2U^T$  của  $\hat{\Sigma} = X^T X / n$  tạo ra ma trận thành phần chính  $U$  và ma trận đường chéo  $D$ :

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

$$U = \begin{bmatrix} -0.8277 & 0.4613 & 0.3195 \\ -0.5300 & -0.4556 & -0.7152 \\ -0.1843 & -0.7613 & 0.6216 \end{bmatrix}, \text{ và } D = \begin{bmatrix} 3.3424 & 0 & 0 \\ 0 & 0.4778 & 0 \\ 0 & 0 & 0.1038 \end{bmatrix}.$$

Chúng ta cũng nhận thấy rằng, ngoài dấu của cột đầu tiên, ma trận thành phần chính  $U$  tương tự như trong Ví dụ 4.10. Tương tự như vậy đối với ma trận  $D$ . Chúng ta thấy rằng 97.90% tổng phương sai được giải thích bởi thành phần chính đầu tiên. Hình 4.13 cho thấy phép chiếu của dữ liệu được căn giữa lên không gian con được bao trùm bởi thành phần chính này.

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)



**Hình:** 4.13. Dữ liệu từ hàm mật độ phân phối xác suất "ván lướt sóng" được chiếu lên không gian con được kéo dài bởi thành phần chính lớn nhất.

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

```
import numpy as np
X = np.genfromtxt('pcadat.csv', delimiter=',')
n = X.shape[0]
X = X - X.mean(axis=0)
G = X.T @ X
U, _, _ = np.linalg.svd(G/n)
# Diem du kien
Y = X @ np.outer(U[:,0],U[:,0])
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.w_xaxis.set_pane_color((0, 0, 0, 0))
ax.plot(Y[:,0], Y[:,1], Y[:,2], c='k', linewidth=1)
ax.scatter(X[:,0], X[:,1], X[:,2], c='b')
ax.scatter(Y[:,0], Y[:,1], Y[:,2], c='r')
for i in range(n):
    ax.plot([X[i,0], Y[i,0]], [X[i,1],Y[i,1]], [X[i,2],Y[i,2]], 'b')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Tiếp theo là một ứng dụng của PCA cho bộ dữ liệu iris nổi tiếng của Fisher, đã được đề cập trong Phần 1.1 và Bài tập 1.5. Ví dụ 4.12 (PCA cho Tập dữ liệu Iris) Bộ dữ liệu iris chứa các phép đo về bốn đặc điểm của cây iris: chiều dài và chiều rộng của lá đài, chiều dài và chiều rộng của cánh hoa, với tổng số 150 mẫu vật. Tập dữ liệu đầy đủ cũng chứa tên loài, nhưng với mục đích của ví dụ này, chúng ta bỏ qua nó.

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Hình 1.9 cho thấy có mối tương quan đáng kể giữa các tính năng khác nhau. Có lẽ chúng ta có thể mô tả dữ liệu bằng cách sử dụng ít tính năng hơn bằng cách lấy một số tổ hợp tuyến tính nhất định của các đối tượng địa lý ban đầu không? Để điều tra điều này, hãy để chúng tôi thực hiện PCA, trước tiên hãy căn giữa dữ liệu. Mã Python sau đây thực hiện PCA. Giả định rằng tệp CSV irisX.csv đã được tạo có chứa tập dữ liệu iris (không có thông tin loài).

## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

```
import seaborn as sns, numpy as np
import scipy.linalg
np.set_printoptions(precision=4)
X = np.genfromtxt('IrisX.csv', delimiter=',')
n = X.shape[0]
X = X - np.mean(X, axis=0)
[U,D2,UT]= np.linalg.svd((X.T @ X)/n)
print('U = \n', U)
print('\n diag(D^2) = ', D2)
z = U[:,0].T @ X.T
sns.kdeplot(z, bw=0.15)
```

U =

```
[[ -0.3614 -0.6566  0.582   0.3155]
 [  0.0845 -0.7302 -0.5979 -0.3197]
 [-0.8567  0.1734 -0.0762 -0.4798]
 [-0.3583  0.0755 -0.5458  0.7537]]
```

```
diag(D^2) =  [4.2001  0.2411  0.0777  0.0237]
```

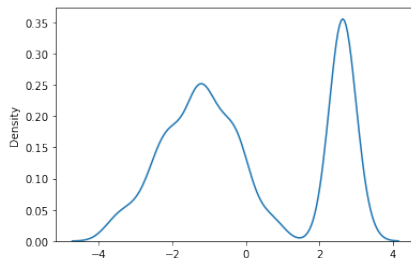
## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)

Kết quả ở trên hiển thị ma trận thành phần chính (mà chúng ta gọi là  $U$ ) cũng như đường chéo của ma trận  $D^2$ . Chúng ta thấy rằng một tỷ lệ lớn của phương

sai,  $4.2001/(4.2001 + 0,2411 + 0,0777 + 0,0237) = 92.46\%$  được giải thích bởi thành phần chính đầu tiên. Do đó, việc biến đổi mỗi điểm dữ liệu  $x \in R^4$  thành  $u_1^T x \in R$ . Hình 4.14 cho thấy ước tính mật độ hạt nhân của dữ liệu đã biến đổi. Điều thú vị là chúng ta thấy có hai chế độ, chỉ ra ít nhất hai cụm trong dữ liệu.



## 4.8.2 Động lực: PCA và Phân tích Giá trị Số ít (SVD)



Hình: 4.14. Ước tính mật độ nhân của dữ liệu iris kết hợp PCA.

Bài thuyết trình đã hết.  
Cảm ơn thầy và các bạn đã lắng nghe.