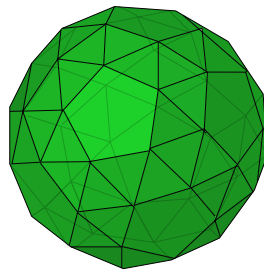


Projects in Mathematics and Applications

# MEAN SHIFT CLUSTERING

Ngày 15 tháng 8 năm 2021

Huỳnh Quỳnh Anh \*      †Nguyễn Nhật Nam  
Nguyễn Lương Thắng ‡      §Trần Hoài An



---

\*Trường Phổ thông Năng khiếu - Đại học Quốc gia Thành phố Hồ Chí Minh

†Trường Trung học phổ thông chuyên Hùng Vương - tỉnh Bình Dương

‡Trường Phổ thông Năng khiếu - Đại học Quốc gia Thành phố Hồ Chí Minh

§Trường Trung học phổ thông chuyên Hoàng Lê Kha - tỉnh Tây Ninh

## Lời cảm ơn

Chúng tôi xin chân thành cảm ơn Ban tổ chức trại hè PiMA đã mang tới cho chúng tôi một trại hè vô cùng bổ ích và thú vị, đã tạo điều kiện cho mọi người có cơ hội học tập và nghiên cứu những phần thật sự thú vị trong toán học cũng như các ứng dụng trong Data Science. Ngoài việc học toán, chúng tôi còn được gặp và tiếp xúc với nhiều người có cùng đam mê với Toán trên mọi miền đất nước.

Chúng tôi xin cảm ơn các diễn giả đã tới và có những buổi nói chuyện thân mật, nhưng cũng rất chi tiết và đem lại cho chúng tôi những kiến thức bổ ích, qua đó đã giúp chúng tôi hiểu hơn cũng như được truyền lửa đam mê trong việc học sau này. Cuối cùng, xin cảm ơn anh Trần Thành Trung, anh Nguyễn Hồ Thành Long và anh Vũ Lê Thế Anh đã theo sát nhóm chúng tôi trong quá trình hoàn thành dự án này.

Trại hè PiMA 2021 tuy kết thúc nhưng chúng tôi tin rằng tinh thần chia sẻ từ những con người nơi đây sẽ thúc đẩy PiMA phát triển hơn nữa trong tương lai. Những bài học từ đây sẽ giúp chúng tôi đón đầu các thử thách mới một cách chủ động và nhiệt tình.

Bài viết này được đã hoàn thành trong 2 tuần của trại hè, do đó sẽ không thể tránh khỏi những thiếu sót. Nếu bạn đọc có ý kiến hay góp ý đến bài báo cáo thì có thể liên hệ với các tác giả ở email [nguyenluongthang33@gmail.com](mailto:nguyenluongthang33@gmail.com).

## Tóm tắt nội dung

Bài báo cáo mô tả về thuật toán mean shift clustering, bao gồm việc trình bày lí thuyết, chứng minh toán học về tính đúng đắn và tính dừng của thuật toán, cũng như mô tả thuật toán và trình bày kết quả trên một số ví dụ cụ thể.

# Mục lục

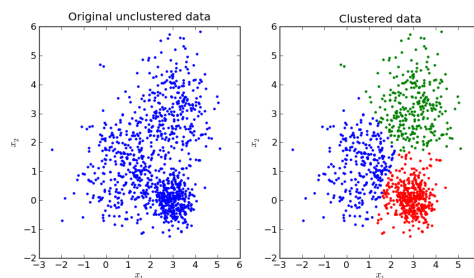
<b>1</b>	<b>Đặt vấn đề</b>	<b>1</b>
1.1	Giới thiệu về bài toán phân cụm . . . . .	1
1.2	Ứng dụng của bài toán phân cụm . . . . .	1
<b>2</b>	<b>Thuật toán mean shift clustering</b>	<b>2</b>
2.1	Một số ký hiệu . . . . .	2
2.2	Kernel density estimator (KDE) . . . . .	2
2.3	Phương pháp Gradient Descent . . . . .	4
2.4	Thuật toán mean shift clustering . . . . .	4
2.5	Tham số bandwidth . . . . .	6
2.6	Sự hội tụ và tính dừng của thuật toán . . . . .	8
<b>3</b>	<b>Áp dụng mô hình</b>	<b>9</b>
3.1	Thuật toán mean shift clustering cơ bản trên bộ dữ liệu mouse.csv . . . . .	9
3.2	Áp dụng các kernel và bandwidth khác nhau . . . . .	11
3.3	Phân mảng hình ảnh (Image Segmentation) . . . . .	13
3.4	Tối ưu hoá Minibatch cho thuật toán mean shift clustering . . . . .	14
<b>4</b>	<b>Kết luận</b>	<b>17</b>
4.1	Kết luận đánh giá . . . . .	17
4.2	Hướng phát triển trong tương lai . . . . .	17

# 1 Đặt vấn đề

## 1.1 Giới thiệu về bài toán phân cụm

Phân cụm là bài toán rất quan trọng trong khai phá dữ liệu, nó thuộc lớp các phương pháp unsupervised learning (học không giám sát) trong machine learning. Có rất nhiều định nghĩa khác nhau về bài toán này, nhưng về bản chất ta có thể hiểu phân cụm là đưa đối tượng về các cụm (clusters), sao cho các đối tượng trong cùng một cụm thì có đặc điểm tương tự nhau. Tuy nhiên, không có tiêu chí nào được xem là tốt nhất để đánh giá hiệu suất của kết quả phân cụm, điều này phụ thuộc vào mục đích của phân cụm như: giảm dữ liệu, phát hiện điểm ngoại lai (outlier).

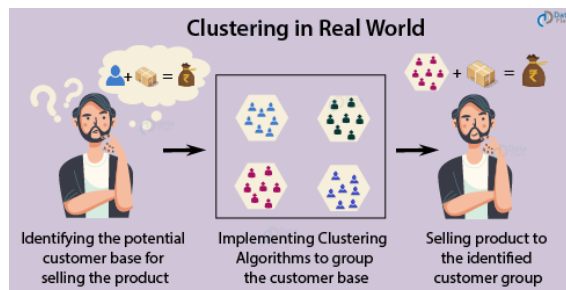
Một số thuật toán phân cụm: k-mean, mean shift, k-nearest neighbors



Hình 1: Dữ liệu khi sử dụng bài toán phân cụm

## 1.2 Ứng dụng của bài toán phân cụm

- Phân mảng hình ảnh (xem thêm [5][6]).
- Marketing: Xác định các nhóm khách hàng tiềm năng, phân loại và dự đoán hành vi khách hàng.
- Bảo hiểm, tài chính: Phân nhóm các đối tượng sử dụng bảo hiểm và các dịch vụ tài chính, dự đoán xu hướng của khách hàng, phát hiện gian lận tài chính.
- Sinh học, y học: Phân nhóm động vật và thực vật dựa vào các thuộc tính của chúng.



Hình 2: Ứng dụng của bài toán phân cụm trong marketing

## 2 Thuật toán mean shift clustering

### 2.1 Một số ký hiệu

- Với  $x$  là một vector thuộc không gian  $d$  chiều ( $x \in \mathbb{R}^d$ ) thì:  $\|x\| = \sqrt{\sum_{i=1}^d x_i^2}$ .
- $S$  là miền xác định của tập dữ liệu ( $S \subset \mathbb{R}^d$  với  $d$  là số chiều của dữ liệu).

### 2.2 Kernel density estimator (KDE)

#### 2.2.1 Động lực

Giả định rằng có một tập dữ liệu rời rạc hữu hạn được sinh ra theo một phân bố nào đó. kernel density estimator (KDE) là một phương pháp không có tham số (non-parametric) được sử dụng để ước lượng hàm phân bố xác suất (probability density function) đã được dùng để sinh ra tập dữ liệu này.

#### 2.2.2 Kernel

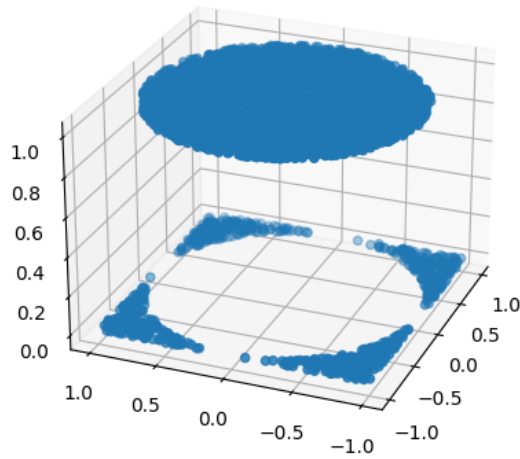
Hàm  $K(\vec{x}) : S \rightarrow \mathbb{R}$  được gọi là một kernel khi và chỉ khi tồn tại một hàm  $k : [0, +\infty) \rightarrow \mathbb{R}$  thoả mãn:

- $K(\vec{x}) = k(\|\vec{x}\|^2)$
- $k(x)$  không âm
- $k(x)$  không tăng, tức là  $\forall a < b : k(a) \geq k(b)$
- $k(x)$  khả vi với mọi  $x \in [0, +\infty)$
- $\int_0^{+\infty} k(x)dx < +\infty$  (thông thường để chuẩn hoá, tích phân này có giá trị bằng 1)

Một số kernel thường gặp trong khoa học dữ liệu: flat, triangle, Epanechnikov, quartic (biweight), tricube, triweight, Gaussian, quadratic. Cụ thể trong thuật toán mean shift, flat và Gaussian kernel là 2 kernel phổ biến mà ta thường thấy khi tìm hiểu về thuật toán này:

#### Flat kernel

$$k(x) = \begin{cases} 1 & \text{if } x \leq \lambda \\ 0 & \text{if } x > \lambda. \end{cases}$$

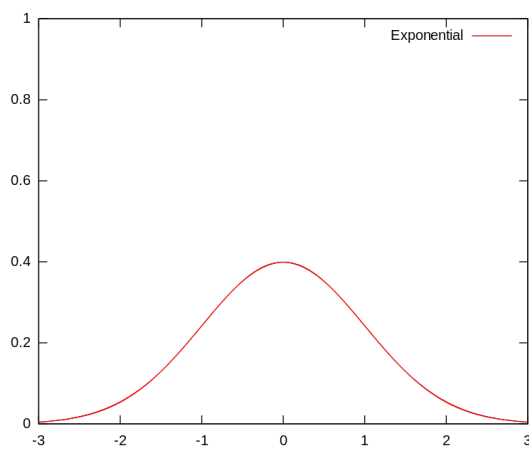


Hình 3: Unit flat kernel cho dữ liệu 2 chiều

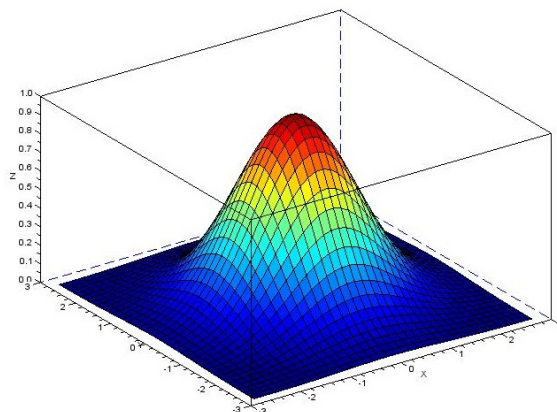
### Gaussian kernel

$$k(x) = e^{-\frac{x^2}{2\sigma^2}},$$

trong đó tham số độ lệch chuẩn  $\sigma$  được coi như là tham số bandwidth của thuật toán mean shift (sẽ được giải thích rõ hơn sau).



Hình 4: Unit Gaussian kernel cho dữ liệu 1 chiều



Hình 5: Unit Gaussian kernel cho dữ liệu 2 chiều

### 2.2.3 Hàm kernel density estimator

Gọi  $(x_1, x_2, \dots, x_n)$  là  $n$  điểm dữ liệu được lấy mẫu độc lập với nhau từ một phân bố xác suất nào đó có hàm mật độ  $f$ . Chúng ta cần ước lượng hình dáng hàm  $f$  này. Hàm  $f$  có thể được ước lượng bởi hàm số  $\hat{f}$ , gọi là kernel density estimator (KDE) và tính bởi công thức [6]:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right). \quad (1)$$

Trong đó  $K$  là hàm kernel còn  $h$  là tham số bán kính (bandwidth), ảnh hưởng đến độ trơn trong ước lượng của phân phối (sẽ được giải thích rõ hơn sau).

## 2.3 Phương pháp Gradient Descent

Trong các bài toán tối ưu hóa, chúng ta thường xuyên phải tìm cực tiểu (hoặc cực đại) của các hàm số nhiều biến và khả vi. Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm  $x_0$ , sau đó dùng một phép toán lặp đi lặp lại để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0. Gradient descent là một trong những phương pháp được dùng nhiều nhất. Tư tưởng của thuật toán này là đi ngược chiều đạo hàm để tìm cực tiểu (hoặc đi cùng chiều để tìm cực đại).

### Thuật toán gradient descent

Dữ liệu đầu vào: hàm  $f(u)$  cần đi tìm cực tiểu theo biến  $u$ , tốc độ học  $\alpha > 0$  và điều kiện dừng  $\epsilon > 0$ .

1. Chọn  $u_0, \alpha > 0, \epsilon > 0$ .
2. Với  $i = 1, 2, \dots$ 
  - If  $\|\nabla f(u_i)\| < \epsilon$ : trả về  $u_i$  và dừng thuật toán;
  - Else: cập nhật  $u_{i+1} = u_i + \alpha \nabla f(u_i)$ .

Trong đó,  $\alpha$  là tốc độ học của thuật toán, quyết định độ dài bước đi theo hướng gradient. Việc chọn  $\alpha$  có vai trò quan trọng vì nếu  $\alpha$  quá nhỏ thì ta sẽ mất rất nhiều thời gian để chạy thuật toán. Ngược lại,  $\alpha$  quá lớn thì có thể sẽ không hội tụ đến cực trị cần tìm. Chính vì vậy, những phiên bản cải tiến hơn của gradient descent cho phép điều chỉnh tốc độ học theo từng bước.

## 2.4 Thuật toán mean shift clustering

### 2.4.1 Lịch sử hình thành

Thuật toán mean shift clustering ra đời vào năm 1975 bởi Fukunaga và Hostetler. Thuật toán này và các phiên bản hoàn thiện hơn của nó thường được ứng dụng vào các bài toán: phân cụm dữ liệu (clustering), phân mảng hình ảnh (image segmentation), dò theo đối tượng hình ảnh (object tracking).

### 2.4.2 Tổng quan thuật toán

Mean shift clustering thuộc loại thuật toán phân cụm bằng cách sử dụng phương pháp *multiple restart gradient ascent* để đưa các điểm dữ liệu về các cực đại địa phương của hàm

KDE (những cụm có mật độ cao). Qua từng bước mean shift, hàm KDE sẽ được cập nhật lại nên ở đây gradient ascent sẽ dùng multiple restart để tính được vị trí mới của mỗi điểm. Khi thuật toán dừng, mỗi điểm được gán cho một cụm.

Không giống như thuật toán phân cụm k-means phổ biến, mean shift clustering không yêu cầu chỉ định trước số lượng cụm. Số lượng các cụm được xác định bởi thuật toán đối với dữ liệu.

### 2.4.3 Các bước trong thuật toán mean shift clustering

Cho  $Q \subset S$  là một tập hữu hạn dữ liệu xung quanh điểm  $x$  và hàm kernel  $K$ . Giá trị trung bình có trọng số với kernel  $K$  tại điểm  $x$  được định nghĩa như sau:

$$m(x) = \frac{\sum_{x_i \in Q} K(x_i - x)x_i}{\sum_{x_i \in Q} K(x_i - x)}. \quad (2)$$

Khi đó,  $m(x) - x$ , được gọi là mean shift vector  $v$ , sẽ chỉ mỗi điểm về hướng của cụm có mật độ cao. Thuật toán mean shift sẽ gán  $x \leftarrow m(x)$  và lặp đi lặp lại cho đến khi hội tụ.

Như vậy, thuật toán mean shift gồm 2 quá trình sau:

#### 1. Mean shift

- (a) Tính toán và dịch chuyển điểm  $x_i^t$  tới vị trí mới  $x_i^{t+1}$ .
- (b) Tính toán mean shift vector  $v(x_i^t)$ .
- (c) Lặp lại 2 bước trên cho tới khi  $x_i^{t+1}$  gần như hội tụ (tương ứng với  $v$  nhỏ).

#### 2. Clustering: Xếp các điểm dữ liệu vào các cụm thích hợp.

### 2.4.4 Ý nghĩa toán học của bước mean shift

Về bản chất, quá trình mean shift các điểm chính là quá trình dùng gradient ascent tìm ra các cực trị địa phương của hàm KDE. Theo 1.2.3, hàm KDE được định nghĩa như sau (tham khảo [6]):

$$f_K(x, Q) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right). \quad (3)$$

Trong đó,  $h$  là tham số bandwidth chỉ bán kính của kernel và hàm  $K(x)$  được định nghĩa như sau:

$$K(x) = ck(\|x\|^2), \quad (4)$$

với  $c$  là hằng số chuẩn hóa và  $k$  là hàm kernel được định nghĩa ở mục 2.2.2. Hằng số chuẩn hóa được sử dụng để giảm bất kỳ hàm xác suất nào thành hàm mật độ xác suất với tổng xác suất là 1. Thay (4) vào (3), hàm dự đoán mật độ (3) sẽ trở thành:

$$f_K(x, Q) = \frac{c}{n} \sum_{i=1}^n k\left(\left\|\frac{x - x_i}{h}\right\|^2\right). \quad (5)$$

Hàm dự đoán mật độ gradient có được bằng cách lấy gradient của hàm (5):

$$\nabla f_K(x, Q) = \frac{2c}{nh^2} \sum_{i=1}^n (x - x_i) k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right). \quad (6)$$



Đặt

$$g(x) = -k'(x). \quad (7)$$

Thay (7) vào (6), ta được:

$$\begin{aligned} \nabla f_K(x, Q) &= \frac{2c}{nh^2} \sum_{i=1}^n (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \\ &= \frac{2c}{nh^2} \left[ \sum_{i=1}^{nh^2} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \right] \left[ \frac{\sum_{i=1}^n g(\|\frac{x-x_i}{h}\|^2) x_i}{\sum_{i=1}^n g(\|\frac{x-x_i}{h}\|^2)} - x \right]. \end{aligned}$$

Suy ra,

$$\frac{\sum_{i=1}^n g(\|\frac{x-x_i}{h}\|^2) x_i}{\sum_{i=1}^n g(\|\frac{x-x_i}{h}\|^2)} = x + \frac{nh^2}{2c \sum_{i=1}^n g(\|\frac{x-x_i}{h}\|^2)} \nabla f_K(x, S). \quad (8)$$

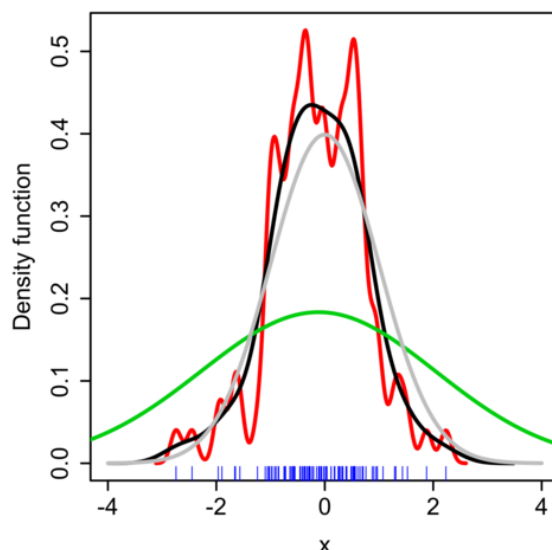
Giờ đây, về trái của (8) chính là vị trí tiếp theo mà điểm  $x$  được dịch chuyển tới, giống như trong thuật toán gradient ascent. Có thể thấy đây là bước gradient với tốc độ học tại điểm  $x$  phụ thuộc vào vị trí mà nó đang đứng. Cụ thể, khi điểm  $x$  ở các vùng có giá trị mật độ thấp, tức là càng ở xa trung tâm các cụm, thì vector mean shift có giá trị càng lớn để điểm  $x$  có thể đến được các cực đại địa phương nhanh hơn. Ngược lại, khi đến gần cực đại địa phương, các vector mean shift sẽ nhỏ lại để không bị văng ra khỏi vùng đó. Do đó, quá trình mean shift đã sử dụng thuật toán multiple restart gradient ascent để đưa mỗi điểm về đúng cụm.

## 2.5 Tham số bandwidth

### 2.5.1 Giới thiệu

Bandwidth của kernel là một tham số tự do có ảnh hưởng mạnh mẽ đến kết quả ước tính. Tùy thuộc vào bandwidth được sử dụng, bề mặt KDE (và kết quả phân cụm) sẽ khác nhau. Nếu ta sử dụng một bandwidth cực kỳ nhỏ, bề mặt KDE sẽ có một đỉnh cho mỗi điểm. Điều này sẽ dẫn đến việc mỗi điểm được đặt vào một cụm riêng của nó. Ngược lại, nếu ta sử dụng bandwidth cực kỳ lớn sẽ dẫn đến tất cả các điểm thuộc vào cùng một cụm. Vì vậy các bandwidth ở giữa hai điểm cực này sẽ tạo ra các phân nhóm đẹp hơn.

Để minh họa (xem hình 6), ta lấy một mẫu ngẫu nhiên từ phân phối chuẩn trên không gian 1 chiều (được vẽ tại các gai màu xanh dương trên trục hoành). Đường cong màu xám là hàm mật độ thực (có giá trị trung bình là 0 và phương sai 1). Khi đó, đường cong màu đỏ quá gồ ghề (hiện tượng undersmoothed) vì nó chứa quá nhiều đỉnh được tạo ra từ việc sử dụng bandwidth quá nhỏ  $h = 0.05$ . Đường cong màu xanh lá cây lại quá bằng phẳng (hiện tượng oversmoothed) vì sử dụng bandwidth lớn  $h = 2$ . Đường cong màu đen có bandwidth  $h = 0.337$  được coi là tối ưu vì ước tính mật độ của nó gần với mật độ thực.



Hình 6: KDE với các bandwidth khác nhau của một phép thử ngẫu nhiên 100 điểm từ phân phối chuẩn.

### 2.5.2 Đánh giá độ tốt của bandwidth

Có nhiều cách đánh giá độ tốt của bandwidth, cách phổ biến nhất là sử dụng hàm mất mát  $L_2$  (còn gọi là MISE - mean integrated squared error).

$$MISE(h) = E \left[ \int (\hat{f}_h(x) - f(x))^2 dx \right], \quad (9)$$

với  $\hat{f}_h$  là hàm KDE mà ta dự đoán còn  $f$  là hàm mật độ xác suất thật sự. Bằng chứng minh toán học [9], giá trị  $h$  làm cho hàm MISE đạt giá trị nhỏ nhất là:

$$h = \frac{\int K(x)^2 dx}{\left[ \int x^2 K(x) dx \right] \left[ \int f''(x)^2 dx \right]^{1/5}}. \quad (10)$$

Lưu ý rằng, công thức ở trên không thể áp dụng được trong thực hành, bởi lẽ trên bộ dữ liệu thực tế, do ta chưa biết được  $f$  nên cũng không thể tính được  $f''$ .

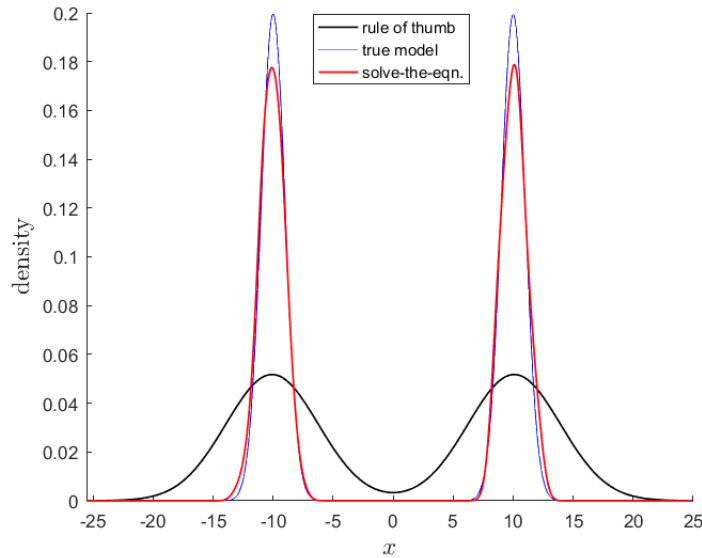
Nếu giả định rằng dữ liệu được sinh bằng phân phối chuẩn, người ta có thể chứng minh được hàm mất mát  $L_2$  đạt giá trị cực tiểu tại (tham khảo [11]):

$$h = \left( \frac{4\sigma^5}{3n} \right)^{1/5}. \quad (11)$$

Tuy nhiên, trên thực tế giả định như vậy là chưa chuẩn, bởi lẽ ta chưa biết hàm mật độ xác suất của dữ liệu đầu vào nên mới phải đi ước lượng nó bằng phương pháp KDE. Trong trường hợp hàm mật độ thật sự không giống như giả định, việc đạt giá trị cực tiểu của hàm mất mát  $L_2$  không còn tương đương với việc ước lượng KDE là gần đúng với hàm mật độ thật. Hình dưới đây minh họa cho ví dụ vừa nêu, trong đó người ta tiến hành sinh 200 điểm theo phân phối hỗn tạp với hàm mật độ

$$f(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-10)^2}{2}} + \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x+10)^2}{2}}$$

thể hiện bằng đường màu xanh dương, đường màu đen là ước lượng KDE tìm được tại bandwidth theo công thức (11), còn đường màu đỏ có  $h$  được tính toán theo (10) và (12). Ta thấy rằng với một giả định sai thì hàm KDE mà ta ước lượng được không chính xác, cụ thể trong trường hợp này ước lượng tìm được là quá bằng phẳng (oversmoothed).



Hình 7: Việc đánh giá độ tốt của bandwidth liên quan đến giả định của ta về kernel

Tới hiện tại, bài toán chọn bandwidth cho hợp lý về mặt toán học vẫn còn là một bài toán mở. Phương pháp chọn bandwidth được dùng phổ biến hiện nay là những phương pháp phụ thuộc vào chính bộ dữ liệu (*adaptive kernel density estimation.*, xem [12], [8])

## 2.6 Sự hội tụ và tính dừng của thuật toán

Như ta đã chứng minh ở trên thì thuật toán mean shift đã sử dụng gradient ascent để đưa mỗi điểm đến cực đại địa phương trong tập data một cách độc lập. Vì vậy, nếu tập data hoặc mật độ của nó không thay đổi trong quá trình thực hiện thuật toán thì sự hội tụ của thuật toán mean shift là hệ quả của việc sử dụng gradient ascent đối với các điểm  $x$  riêng lẻ.

Đối với dữ liệu rời rạc, số bước hội tụ phụ thuộc vào kernel được sử dụng. Khi  $G$  là flat kernel, sự hội tụ đạt được trong một số bước hữu hạn vì số lượng vị trí tạo ra các giá trị trung bình khác biệt là hữu hạn. Tuy nhiên, khi ta sử dụng một kernel khác, ví dụ như Gaussian kernel, thì sự hội tụ sau hữu hạn bước là không chắc chắn hoặc sẽ tốn rất nhiều bước. Trên thực tế, để tránh lặp vô hạn, người ta đặt giới hạn dưới cho độ lớn của vectơ dịch chuyển trung bình, tức là khi nào độ dài *mean shift vector* đủ nhỏ ( $\|v(x_i)\| < \epsilon$ ) thì dừng.

Mặc dù thuật toán mean shift được sử dụng rộng rãi với nhiều ứng dụng khác nhau nhưng sự hội tụ của nó ở trường hợp tổng quát vẫn còn là một bài toán mở. Trong một số điều kiện nhất định về kernel và số chiều, chúng ta có một số kết quả sau:

Định lý về sự hội tụ của thuật toán mean shift trong trường hợp 1 chiều (tham khảo phần chứng minh ở [1]):

**Định lý 1.** Cho  $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}$  là tập dữ liệu đầu vào.  $\hat{f}_{h,k}(x)$  là hàm ước tính mật độ xác suất sử dụng kernel  $K$  với  $k$  lõi, khả vi, tăng ngặt và bandwidth  $h$ . Giả sử  $g(x) = -k'(x)$  là một hàm giảm ngặt, đặt  $y_{i,m}$  là điểm mean shift của  $x_i$  ở bước thứ  $m$  thì ta có dãy  $(y_{i,m})$  sẽ hội tụ  $\forall i$ .

Trong không gian nhiều chiều với Gaussian kernel, ta có bổ đề định lý sau đây (tham khảo phần chứng minh ở [2]):

**Bổ đề 1.** Cho  $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$  là tập dữ liệu đầu vào. Giả sử  $\hat{f}(x)$  là hàm ước tính mật độ xác suất với ma trận hiệp phương sai  $\Sigma = h^2 \mathbf{I}$ . Đặt  $\|x_{\max}\| = \max_{1 \leq i \leq n} \|x_i\|$ . Nếu  $\|x_{\max}\| < h$ , thì ma trận Hessian của hàm ước tính mật độ xác suất tại những điểm dừng khả nghịch và những điểm dừng cũng bị cô lập.

Lưu ý rằng số lượng điểm dừng của Gaussian kernel sẽ bất biến trong quá trình thực hiện mean shift. Chính vì vậy, nếu điều kiện  $\max_{1 \leq i \leq n} \|x_i - x_0\|$  đúng cho một số điểm trung tâm thì bổ đề 1 sẽ đúng và điều kiện đủ sẽ thực tế hơn. Trong trường hợp các điểm dừng cô lập và có số lượng hữu hạn thì ta có định lý sau:

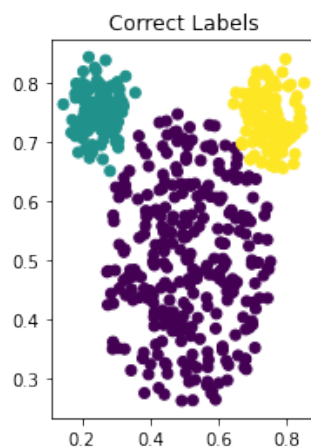
**Định lý 2.** Cho  $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$  là tập dữ liệu đầu vào. Giả sử rằng các điểm dừng bị cô lập thì dãy  $(y_{i,m})$  với  $m \rightarrow \infty$  sẽ hội tụ  $\forall i$  với  $y_{i,m}$  là điểm mean shift của  $x_i$  ở bước thứ  $m$ .

Định lý 2 đảm bảo sự hội tụ của vị trí các điểm mean shift khi các điểm dừng (cực đại địa phương) bị cô lập (với những điều kiện đủ được chỉ ra ở bổ đề 1). Tuy nhiên, việc tìm và sử dụng một ma trận hiệp phương sai  $\Sigma$  thỏa mãn điều kiện đủ trong bổ đề 1 là một nhiệm vụ khó khăn và làm tăng độ phức tạp, đặc biệt khi kích thước của tập dữ liệu đầu vào lớn. Do đó, trong thực tế để giảm chi phí tính toán, ma trận hiệp phương sai được chọn tỷ lệ với ma trận  $\Sigma = h^2 \mathbf{I}$ . Khi ma trận hiệp phương sai được chọn tỷ lệ với ma trận trên, các điểm dừng sẽ cố định nếu  $\|x_{\max}\| < h$ . Tuy nhiên trên thực tế chúng ta sẽ không chọn một giá trị bandwidth  $h$  lớn để đảm bảo điều này, vì độ lệch lớn sẽ dẫn đến ước tính hàm mật độ xác suất kém, dẫn đến vị trí các điểm dừng không chính xác. Tóm lại, các điều kiện lý thuyết được cung cấp bởi bổ đề 1 để ước lượng hàm mật độ xác suất Gaussian có các điểm dừng cô lập chỉ được sử dụng hạn chế trong thực tế.

## 3 Áp dụng mô hình

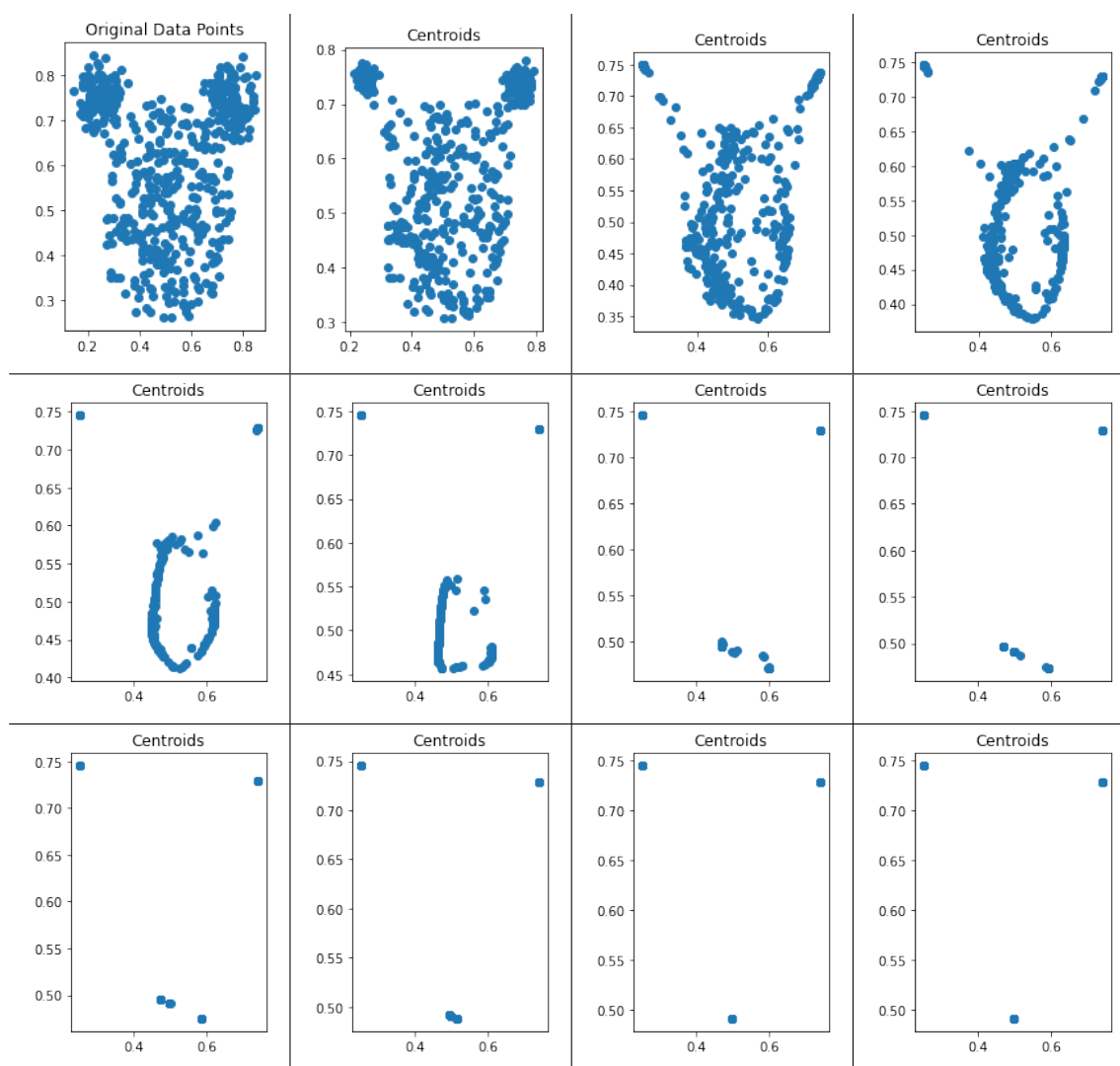
### 3.1 Thuật toán mean shift clustering cơ bản trên bộ dữ liệu mouse.csv

Bộ dữ liệu này có hình dạng như sau:

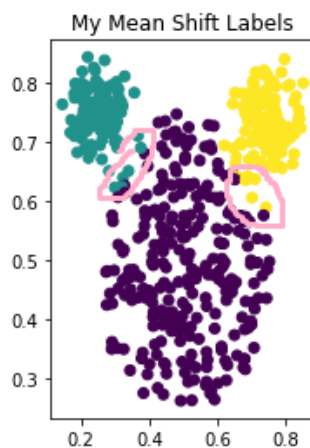


Hình 8: Bộ dữ liệu mouse.csv với kết quả phân cụm dự tính

Ta có thể chạy thuật toán mean shift clustering với tham số `kernel=Gaussiankernel`, `bandwidth=0.116` và quan sát sự thay đổi của các điểm trọng tâm sau mỗi iteration như sau:



Bảng 1: Các điểm trọng tâm mỗi cụm sau các iteration từ thứ 0 đến thứ 11 theo thứ tự từ trái sang phải, từ trên xuống dưới

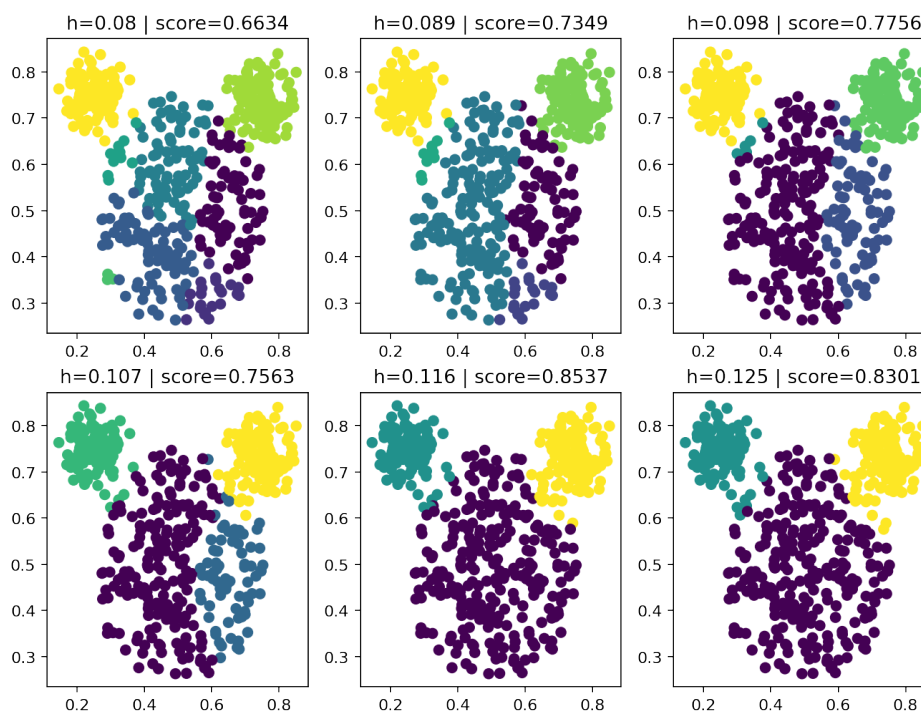


Hình 9: Bộ dữ liệu mouse.csv sau khi đã chạy thuật toán với Gaussian kernel, bandwidth=0.116. Một số điểm không được cluster chính xác được khoanh màu hồng.

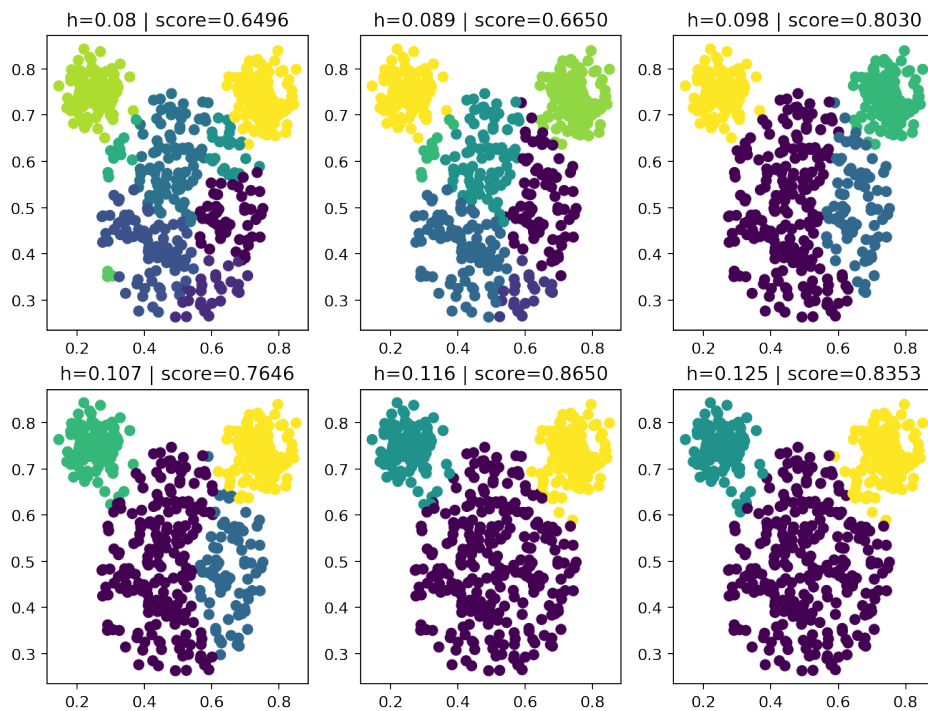
Thuật toán khi kết thúc cho ta 3 trọng tâm. Điều này là chính xác, tuy nhiên có một số ít điểm vẫn chưa được phân vào đúng cluster. Điều này có thể được giải thích bằng hàm KDE: tại những điểm bị cluster sai, gradient của hàm KDE đều hướng về cluster màu xanh (vàng) thay vì cluster màu tím do mật độ điểm tại window bán kính bandwidth tại các điểm này có nhiều điểm màu xanh (vàng) hơn so với mật độ các điểm màu tím.

## 3.2 Áp dụng các kernel và bandwidth khác nhau

### 3.2.1 Dataset mouse.csv



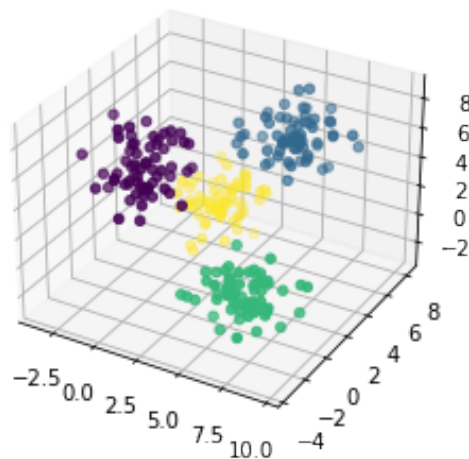
Hình 10: flat kernel trên dataset mouse.csv trên một số bandwidth



Hình 11: Gaussian kernel trên dataset mouse.csv trên một số bandwidth

Chỉ xét đến bộ dữ liệu này, có thể thấy rằng với giá trị bandwidth thích hợp, kết quả cho Gaussian kernel tốt hơn một chút so với flat kernel. Điều này có thể được giải thích do cách mà bộ dữ liệu mouse.csv này được xây dựng.

### 3.2.2 Dataset blobs

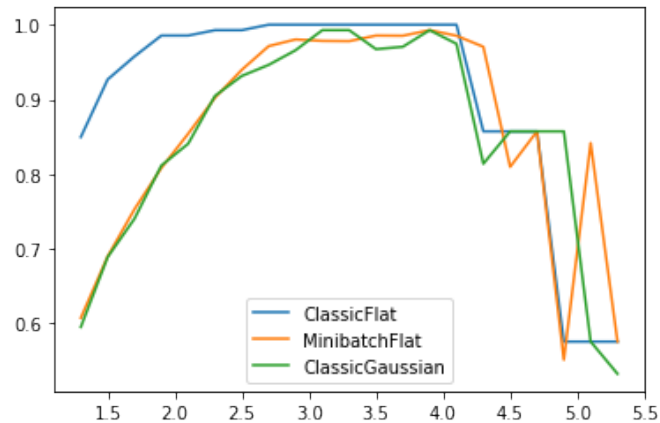


Hình 12: Một bộ dữ liệu blobs với cách phân cụm chính xác

Với dataset này, chúng tôi sinh ra nhiều bộ dữ liệu khác nhau với cùng tham số như sau:

```
1 X, Y = make_blobs(250, 3, centers = [[0,0.1,5.3],
    [5.6,5.4,5.8], [6.5,-1.1,-0.2], [1.5,5.1,0.1]],
    cluster_std=1.13)
```

rồi sau đó tiến hành chạy nhiều lần trên các bandwidth dao động từ 1.4 đến 5.3 và xem độ chính xác (tính theo [10]), trên các kernel flat, Gaussian, thu được biểu đồ sau đây:



Hình 13: Độ chính xác (V-measure) trên dataset blobs, trên các thuật toán Classic (kernel: flat, Gaussian), Minibatch (kernel: flat), với bandwidth chạy từ 1.4 đến 5.3

Trong trường hợp này, khi bandwidth thích hợp, flat kernel có vẻ cho kết quả tốt hơn. Điều này là trái ngược khi so sánh với dataset mouse.csv. Khi bandwidth quá lớn hoặc quá nhỏ, kết quả phân cụm trở nên hỗn loạn. Cụ thể khi bandwidth lớn, các điểm khác cụm sẽ được đưa vào cùng cụm, nói cách khác số cụm phân được là nhỏ hơn 3, còn khi bandwidth nhỏ, số cụm phân được là rất lớn. Từ đây, ta có thể kết luận rằng việc chọn kernel và bandwidth phụ thuộc nhiều vào dữ liệu, và để chắc chắn ta cần phải thử nhiều lựa chọn khác nhau để chọn ra được siêu tham số cho kết quả tốt nhất.

Ghi chú rằng, cách đánh giá thuật toán clustering bằng V-measure không phải là cách đánh giá duy nhất. Mỗi cách đánh giá (evaluate) có một số ưu nhược điểm khác nhau, nằm ngoài phạm vi nghiên cứu của bài này. Có thể tham khảo thêm về điều này tại tài liệu tham khảo của thư viện scikit-learn.

### 3.3 Phân mảng hình ảnh (Image Segmentation)

Thử nghiệm trên data Silhouette of Mountains (nguồn tham khảo: [4]) (trong phần này gọi tắt là dataset scenery.png):

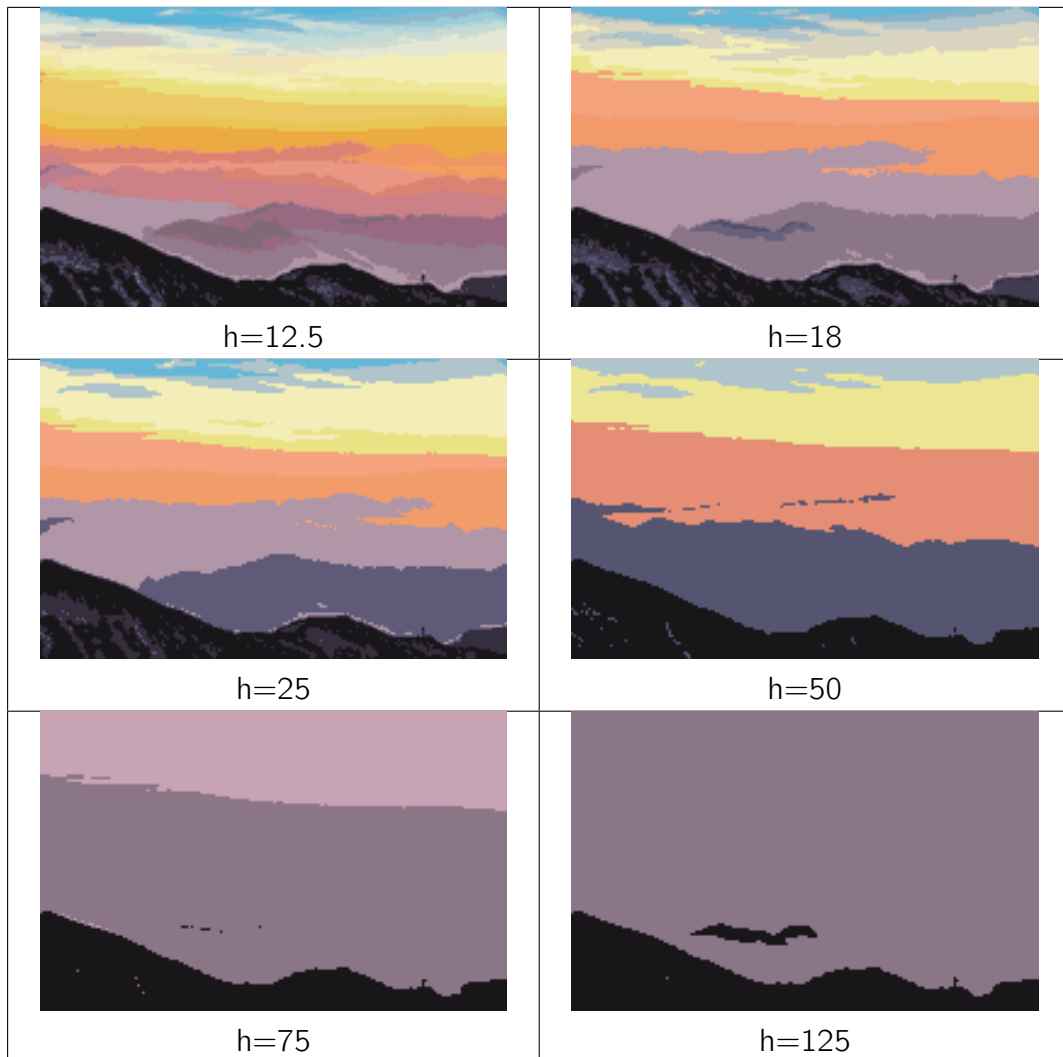


Hình 14: Silhouette of Mountains, Simon Berger

Mục tiêu: phân mảng ảnh thành các mảng màu.



Ứng dụng phân mảng hình ảnh có thể minh hoạ rõ nhất cách mà tham số bandwidth ảnh hưởng sâu sắc đến kết quả phân cụm. Như hình minh hoạ bên dưới, ảnh gốc được phân cụm dựa trên các bandwidth khác nhau (cách biểu diễn kết quả phân cụm: các pixel cùng một cụm thì được tô cùng một màu).



Bảng 2: Bộ dữ liệu scenery.png khi chạy trên các bandwidth từ 12.5 đến 125

### 3.4 Tối ưu hoá Minibatch cho thuật toán mean shift clustering

#### 3.4.1 Động lực

Khi đánh giá độ phức tạp thuật toán mean shift clustering, ta có thể thấy tại mỗi lần duyệt (iteration) tồn tại chi phí độ phức tạp thời gian vào cỡ  $O(n^2)$  ( $n$  điểm  $x$  cần tính lại  $m(x)$ , mỗi lần tính lại mất  $O(n)$ ), khiến độ phức tạp chung của thuật toán rơi vào khoảng  $O(iter \times n^2)$  với  $iter$  là số lần lặp.

Xem lại công thức tính  $m(x)$  (công thức (2)):

$$m(x) = \frac{\sum_{x_i \in Q} K(x_i - x)x_i}{\sum_{x_i \in Q} K(x_i - x)}.$$

Bước này có độ phức tạp  $O(n)$  do ta duyệt hết toàn bộ mọi điểm, lần lượt kiểm tra từng điểm xem điểm nào nằm trong bán kính  $h$  rồi tính trọng số tương ứng. Tuy nhiên trong trường hợp ta bị hạn chế về mặt thời gian, ta có thể dùng một ước lượng kém chính xác hơn để tính  $m(x)$  theo tư tưởng sau: chọn một tập hợp con ngẫu nhiên của những điểm dữ liệu (gọi là một *batch*), với đủ nhiều điểm để đại diện phân bố xác suất  $f$  ban đầu, rồi chỉ tính trung bình cộng có trọng số theo những điểm đã chọn. Điều này là tư tưởng chính của phép tối ưu minibatch: tính đáng kể về mặt thống kê (statistically significant).

Trong thực tế cài đặt, thay vì sinh cho mỗi điểm một batch ngẫu nhiên khác nhau, ta thực hiện phân hoạch ngẫu nhiên tập dữ liệu ra các tập con, mỗi tập có kích thước xấp xỉ `batch_size`, khi đó điểm dữ liệu nằm trong tập con nào thì batch tương ứng là tập hợp đó.

### 3.4.2 Thuật toán

Phần mean shift của thuật toán được mô tả cụ thể như sau:

1. Tại mỗi iteration, phân hoạch ngẫu nhiên bộ dữ liệu  $X$  ra thành một số phần (batch)  $P_1, P_2 \dots P_b$  có kích thước xấp xỉ `batch_size`.
2. Tại mỗi bước tính  $m(x)$ , nếu  $x$  thuộc phần  $P_j$  thì ta ước lượng  $m(x)$  bằng giá trị  $\hat{m}(x)$  như sau:

$$\hat{m}(x) = \frac{\sum_{x_i \in (P_j \cap Q)} K(x_i - x) x_i}{\sum_{x_i \in (P_j \cap Q)} K(x_i - x)}.$$

3. Hợp các điểm dữ liệu sau khi đã gán  $x \leftarrow \hat{m}(x), \forall x \in X$ , thành tập  $X'$  mới rồi thực hiện lại hai bước trên đến điều kiện dừng.

### 3.4.3 Tính chất

Nếu sau mỗi iteration của thuật toán, tại bước 1 ta lại chia dữ liệu thành các batch ngẫu nhiên và `batch_size` đủ lớn để đại diện cho toàn bộ tập dữ liệu, ta có thể xem như ước lượng  $\hat{m}(x)$  là đủ chính xác cho  $m(x)$ , qua đó thuật toán có tính đúng đắn và tính dừng được chứng minh tương tự như thuật toán mean shift cơ bản.

Độ phức tạp thời gian cho một iteration của cách làm này là  $O(n \times \text{batchsize})$ . Với cách tối ưu này ta có thêm một siêu tham số `batch_size`, nếu `batch_size` quá lớn (gần bằng  $n$ ), thuật toán chạy chậm, không khác gì thuật toán mean shift clustering cơ bản. Nếu `batch_size` quá nhỏ, tập batch quá nhỏ không đại diện được cho hàm mật độ của phân bố lúc đầu, dẫn đến kết quả phân cụm sai.




Bằng thực nghiệm, con số `batch_size` hiệu quả rơi vào khoảng  $[\log_2(n), 10 \times \log_2(n)]$ .

### 3.4.4 Kiểm thử

Ta so sánh Minibatch so với thuật toán meanshift cơ bản và của thư viện sklearn, được kết quả như sau:

#### Dataset scenery.png

Ta nhận xét trên dataset này, thuật toán cơ bản không thể chạy ra kết quả trong 2 giờ đồng hồ. Thuật toán Minibatch meanshift clustering có kết quả sau khoảng hơn 30 phút, tuy nhiên kết quả có phần bị nhiễu. Điều này có thể được giải thích do batchsize nhỏ, tương ứng với

Thuật toán	Ảnh	Thời gian chạy
Ảnh gốc		-
sklearn.meanshift		222.1s
My meanshift	Không có kết quả	> 2 hours
Minibatch meanshift batchsize=10		1923.5s

Bảng 3: Bộ dữ liệu scenery.png khi chạy bằng các thuật toán: mean shift clustering, Minibatch clustering, Sklearn.meanshift, độ phân giải hình ảnh 192x123, bandwidth = 25

phân phối được lấy tại mỗi iteration có thể không trùng khớp với hàm mật độ chuẩn. Thuật toán sklearn.meanshift do sử dụng Ball Tree để tìm các điểm thuộc flat kernel (chi tiết xem tại triển khai cài đặt thư viện của sklearn), nên có độ phức tạp rơi vào khoảng  $O(T \times n \log(n))$  cho trường hợp số chiều nhỏ (2 chiều), vì vậy nên chạy nhanh vượt trội.

## 4 Kết luận

### 4.1 Kết luận đánh giá

Phương pháp mean shift clustering được xây dựng dựa trên cơ sở toán học về kernel density estimator và gradient ascent, bộc lộ nhiều ưu, nhược điểm trong một số trường hợp demo đã được thể hiện phía trên.

#### 4.1.1 Ưu điểm

- Không dựa trên một số giả định như thuật toán K-means: số lượng điểm trong 1 cluster phải xấp xỉ nhau, biết trước số lượng cụm cần tìm hay hình dạng của cụm.
- Chỉ phụ thuộc một siêu tham số là bandwidth.
- Xử lý các điểm dữ liệu ngoại lai tốt hơn khá nhiều so với thuật toán K-means (các cụm có số điểm ít có thể được xem là các điểm ngoại lai).

#### 4.1.2 Nhược điểm

- Trong trường hợp phải xử lý một lượng lớn các điểm dữ liệu từ không gian nhiều chiều, ta áp dụng Gradient Ascent cho toàn bộ dữ liệu trong một lúc dẫn đến độ phức tạp tính toán cao (xem thêm ở Bảng 3).
- Ngoài ra, trong một số trường hợp cần phải xác định số phân cụm cụ thể, ta cần phải tìm ra một giá trị bandwidth phù hợp và kernel phù hợp cho thuật toán mean shift. Tuy nhiên, như đã giới thiệu về cách xác định giá trị bandwidth tốt nhất (mục 2.4.2) là một việc gần như không thể áp dụng vào thực tế, do đó việc tìm ra giá trị này cho một trường hợp cụ thể cũng là một bài toán không hề đơn giản.

### 4.2 Hướng phát triển trong tương lai

Theo như kết quả chạy được biểu diễn qua Bảng 3, chúng tôi nhận thấy có một số vấn đề cần phải khắc phục như là cải thiện độ chính xác của Minibatch meanshift và cải thiện thời gian chạy của My meanshift trong việc xử lý hình ảnh.

Về Minibatch meanshift, chúng tôi đề xuất phương án sử dụng thêm một loại trọng số phụ thuộc vào độ quan trọng của pixel. Cụ thể, đối với các pixel nằm ở ngoài biên không có màu, chúng không đóng góp nhiều đến kết quả phân cụm sau cùng, do đó ta gán cho chúng trọng số thấp (như 0.1 chẳng hạn). Còn đối với các điểm không nằm ở ngoài biên thì ta có thể gán cho chúng trọng số cao hơn (như 1.0 chẳng hạn). Phương án này chỉ là một đề xuất để giảm bớt độ nhiễu của hình ảnh đã qua xử lý bằng Minibatch meanshift, nên vẫn chưa có chứng minh toán học chặt chẽ cho ý tưởng này.

Về My meanshift, đặc biệt là trong phân vùng hình ảnh hay phân cụm dữ liệu 2 chiều, theo [13] thì chúng tôi nhận thấy nếu tính mean shift Vector (MSV) theo hàm flat kernel thì chỉ cần qua một số bước xử lý ban đầu, ta có thể tính MSV trong  $O(1)$  bằng thuật toán Prefix Sum of Matrix (Array 2D). Qua đó, chúng tôi có thể giảm thiểu độ phức tạp tính toán trong quá trình chạy mean shift.

## Tài liệu

- [1] Youness Aliyari Ghassabeh. On the convergence of the mean shift algorithm in the one-dimensional space. *Pattern Recognition Letters*, 34(12):1423–1427, 2013.
- [2] Youness Aliyari Ghassabeh. A sufficient condition for the convergence of the mean shift algorithm with gaussian kernel. *Journal of Multivariate Analysis*, 135:1–10, 2015.
- [3] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.
- [4] Simon Berger. Silhouette of mountains.
- [5] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.
- [6] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
- [7] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.
- [8] Peter Mills. Efficient statistical classification of satellite measurements. *International Journal of Remote Sensing*, 32(21):6109–6132, 2011.
- [9] Mats Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9(2):65–78, 1982.
- [10] scikit-learn library. V-measure cluster labeling given a ground truth.
- [11] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall/CRC, 1986.
- [12] George R. Terrell and David W. Scott. Variable Kernel Density Estimation. *The Annals of Statistics*, 20(3):1236 – 1265, 1992.
- [13] Geming Wu, Xinyan Zhao, Shuqian Luo, and Hongli Shi. Histological image segmentation using fast mean shift clustering method. *Biomedical engineering online*, 14(1):1–12, 2015.