4.6

Nguyễn Thị Cẩm Hương

Ngày 24 tháng 12 năm 2021

1 Phân cụm thông qua lượng tử hóa vectơ

Trong phần trước, chúng tôi đã giới thiệu phân cụm thông qua các mô hình hỗn hợp, như một hình thức ước tính mật độ tham số (trái ngược với ước tính mật độ phi tham số trong Phần 4.4).
Các cụm được mô hình hóa một cách tự nhiên thông qua các biến tiềm ẩn và thuật toán EM cung cấp một cách thuận tiện để gán các thành viên cụm. Trong phần này, chúng tôi xem xét một cách tiếp cận heuristic (khám phá 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.
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 $\tau := \{x_1, ..., x_n\}$ 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 dist $(\cdot, \cdot)$ trên không gian này. Một lựa chọn tiêu chuẩn là Euclidean (hoặc L_2) khoảng cách:

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

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

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

và khoảng cách tối đa

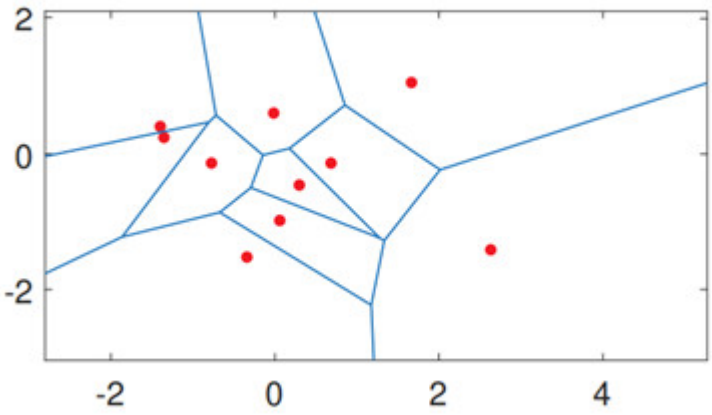
$$\max_{i=1,..,d} |x_i - x'_i|$$

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

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

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

và 011010 là 4.
Chúng ta có thể phân vùng không gian X thành các vùng như sau:
Đầu tiên, chúng ta chọn K điểm $c_1, ..., c_K$ được gọi là tâm cụm hoặc vectơ nguồn . Với mỗi k = 1, ..., K , sao cho:
$R_k = \{x \in X : dist(x, c_k) \leq dist(x, c_i)$ cho tất cả i ≠ 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 (biểu đồ Voronoi) hoặc Voronoi tessellatio.
Hình 4.8 cho thấy sự phân chia Voronoi của máy bay thành mười vùng, sử dụng Euclidean khoảng cách. Lưu ý rằng ở đây ranh giới giữa các ô Voronoi là đường thẳng. Đặc biệt, nếu ô R_i vàR_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 $(cj + ci)/2$ (nghĩa là trung điểm của đoạn thẳng giữa c_i và c_j) và vuông góc với $c_j - c_i$



Hình 4.8: Một điểm dừng Voronoi của máy bay thành mười ô, được xác định bởi (màu đỏ) các trung tâm. Khi các tâm (và do đó các ô R_k) được chọn, các điểm trong τ có thể được nhóm lại theo trung tâm gần nhất của chúng. Các điểm trên ranh giới phải được xử lý riêng biệt. Điều này là điểm tranh luận cho dữ liệu liên tục, vì nói chung không có điểm dữ liệu nào nằm chính xác trên ranh giới. Vấn đề còn lại chính là làm thế nào để chọn các trung tâm để phân cụm dữ liệu trong một số cách tối ưu. Về khung học tập (không giám sát), muốn ước tính một vectơ x qua một trong $c_1, ..., c_K$, sử dụng một hàm có giá trị vectơ hằng số

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

trong đó C là ma trận $d \times$ K $[c_1, ..., c_K]$. Do đó, $g(x|C) = c_k$ khi x nằm trong vùng R_k (chúng ta bỏ qua các ràng buộc). Trong lớp hàm G này, được tham số hóa bởi C, mục đích của chúng tôi là giảm thiểu sự mất mát trong đào tạo.

Đặc biệt, đối với tổn thất sai số bình phương. $Loss(x, x') = x - x'2$

$$l_{\tau n}(g(.|C)) = \frac{1}{n}\sum_{i=1}^{n}\|x_i - g(x_i|\mathbb{C})\|^2 = \frac{1}{n}\sum_{k=1}^{K}\sum_{x\in R_k\cap\tau n}\|x - c_k\|^2$$

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
[125]. Cụ thể, chúng tôi muốn "lượng tử hóa" hoặc "mã hóa" các vectơ trong τ theo cách mà mỗi vectơ được biểu diễn bởi một trong K vectơ nguồn $c_1, ..., c_K$, sao cho (4.40) đại diện được giảm thiểu. Hầu hết các phương pháp phân cụm và lượng tử hóa vectơ nổi tiếng đều cập nhật vectơ của các trung tâm, bắt đầu từ một số lựa chọn ban đầu và sử dụng thủ tục lặp lại (thường dựa trên gradient). 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 là tốt 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 gradient(dốc, điểm) có xu hướng hội tụ ở mức tối thiểu cục bộ hơn là mức tối thiểu toàn cầu.

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

Algorithm 4.6.1: K-Means

input: Collection of points $\tau = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, number of clusters K, initial centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$.
output: Cluster centers and cells (regions).

while a stopping criterion is not met **do**
  $\mathcal{R}_1, \ldots, \mathcal{R}_K \leftarrow \emptyset$ (empty sets).
  for $i = 1$ **to** n **do**
    $\boldsymbol{d} \leftarrow [\text{dist}(\boldsymbol{x}_i, \boldsymbol{c}_1), \ldots, \text{dist}(\boldsymbol{x}_i, \boldsymbol{c}_K)]$ // distances to centers
    $k \leftarrow \text{argmin}_j\, d_j$
    $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \{\boldsymbol{x}_i\}$ // assign $\boldsymbol{x}_i$ to cluster k
  for $k = 1$ **to** K **do**
    $\boldsymbol{c}_k \leftarrow \frac{\sum_{\boldsymbol{x}\in\mathcal{R}_k}\boldsymbol{x}}{|\mathcal{R}_k|}$ // compute the new center as a centroid of points
return $\{\boldsymbol{c}_k\}, \{\mathcal{R}_k\}$

Do đó, tại mỗi lần lặp, đối với một lựa chọn trung tâm nhất định, mỗi điểm trong τ 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.
Ví dụ: đã chọn ngẫu nhiên từ hộp giới hạn của các điểm dữ liệu. Chúng ta có thể xem phương pháp K-mean như một phiên bản xác định (hoặc "cứng") của xác suất thuật toán EM ilistic (hoặc "mềm") như sau. Giả sử trong thuật toán EM, chúng ta có Gausian hỗn hợp với ma trận hiệp phương sai cố định $\sum_k \sigma^2 \mathbb{I}_d, k = 1, ..., K$, trong đó σ^2 phải là được coi là rất nhỏ. Xem xét lần lặp t của thuật toán EM. Dang có thu được vectơ kỳ vọng${\mu_k}^{t-}$) và trọng lượng ${w_k}^{t-1}$, k = 1, ..., K, mỗi điểm x_i là đã ký một nhãn cụm Z_i theo các xác suất $p_i^{(t)}(k)$, k = 1, ..., K cho trong (4.36). Trang 4 Chương 4. Học tập không giám sát 145
Nhưng đối với $\sigma^2 \to 0$ thì phân phối xác suất $p_i^{(t)}(k)$ trở nên thoái hóa, đặt tất cả khối lượng xác suất trên $argmin_k \|x_i - \mu_k\|^2$. 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. Do đó chúng tôi có được quy tắc cập nhật xác định tương tự như trong K-mean. Ví dụ 4.6 (K-means Clustering) Chúng tôi phân cụm dữ liệu từ Hình 4.8 qua K-means, bằng cách sử dụng triển khai Python bên dưới. Lưu ý rằng các điểm dữ liệu được lưu trữ dưới dạng 300 $\times$ 2 ma trận Xmat. 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.

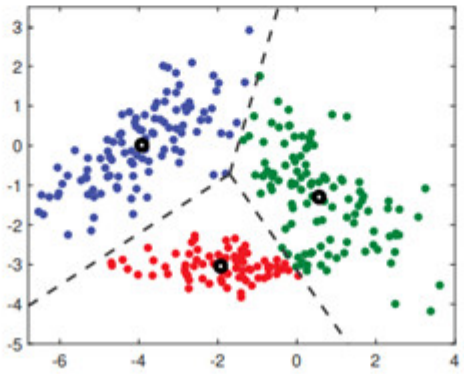
Kmeans.py

```
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 = {:3.3f}'.format(minvals.mean()))
```

```
Loss = 2.288
```



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.

1.2 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 rất đa phương thức. Tuy nhiên, không có gì ngăn cản chúng tôi 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 Ví dụ 4.7 (Phân cụm qua CE) Chúng tôi 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. Con trăn mã dưới đây rất giống với mã trong Ví dụ 3.16, ngoại trừ việc bây giờ chúng ta đang xử lý 101 với một bài toán tối ưu hóa sáu chiều. Hàm mất mát được thực hiện trong function Scluster , 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 lỗ 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.