Generalized Fuzzy k-Means Clustering Using m nearest Cluster Centers

Franklin J. C. Lai , Eric Y. T. Juan, and Jim Z. C. Lai

Department of Computer Science and Engineering

National Taiwan Ocean University

Keelung, Taiwan 202

R. O. C.

**PHƯƠNG PHÁP ĐỀ XUẤT**

Sử dụng fuzzy *k*-means clustering để phân chia một data set, một data point thuộc về đồng thời k cụm. Tuy nhiên, trong một vài các data set, một data point thuộc cụm với trung tâm không ở tập hợp của M các hàng xóm gần nhất của nó là rất thấp. Trong trường hợp này, chúng ta có thể sử dụng M trung tâm cụm gần nhất của data point của bộ phận data set. Trong fuzzy *k*-means clustering tổng quát, một thành viên ****∈[0, 1] còn được sử dụng để thể hiện mối liên hệ giữa một data point **** và đại diện cụm **C***j*. Đặt *NNTi* bao gồm *M* trọng tâm cụm gần nhất cho data point ****. Thuật toán phân cụm GFKM (generalized FKM) dựa vào sự giảm thiểu độ sai lệch sau đây:

*J* =  (5)

với thể hiện cho các đại diện cụm **C***j* và các thành viên **,** với *N* là số lượng các data point; *q* là fuzzifier; *M* =  là số lượng các cụm gần nhất cho một data point; và  là khoảng cách Ơ-le bình phương giữa các data point **** và đại diện cụm gần nhất thứ *j* của nó. Trong GFKM, **** phải đáp ứng ràng buộc sau:

= 1, cho *i* = 1 tới *N*. (6)

Quá trình chính của GFKM còn mapping một tập hợp các vector đại diện vào một cải thiện thông qua việc phân vùng các data point. Nó bắt đầu với một tập hợp khởi tạo các trung tâm cụm và lặp lại quá trình mapping cho đến khi điều kiện dừng xuất hiện. It begins with an initial set of cluster centers and repeats this mapping process until a stopping criterion is met. Có hàm Lagrangian *Λ* được định nghĩa như sau

*Λ* = -  (7)

với  hệ số nhân Lagrangian và **** là thành viên giữa data point **X***i* và trung tâm cụm gần nhất thứ *j* của nó. Lấy đạo hàm *Λ* đốivới**** nhận được:

 = ** (8)

Giải quyết  = 0 đối với biến **** được

 = , cho *s* = 1 tới *M*. (9)

Thay thế biểu thức (9) vào (6) và làm một ít biến đổi chúng ta có được

 =  (10)

Kết hợp các biểu thức (9) và (10) có

**** = , cho *r* = 1 tới *N* và *s* = 1 tới *M* (11)

Nếu **C***j*∈*NNTi* là hàng xóm gần nhất thứ *l* của **X***i*, gán  = **;** ngược lại gán  = 0, với  là các thành viên giữa các điểm **X***i* và các trung tâm cụm **C***j*. Ở giai đoạn này, chúng ta có thể viết lại biểu thức (5) như sau:

*J* =  (12)

với  là khoảng cách Ơ-le bình phương giữa **X***i* và **C***j*. Cần lưu ý rằng

=  (13)

với *t* thể hiện cho sự chuyển vị. Đạo hàm biển thức (12) đối với  có

= 0 (14)

Giải quyết biểu thức (14) cho **C***j* có

**C***j* =  cho *j* = 1 tới *k* (15)

Cho S *j* = {**X** *i*: **X** *i*∈ *NNTj*, *i* = 1 tới *N*}, sau đó biểu thức (15) có thể được viết lại như sau

**C***j* =  (16)

Bây giờ, chúng ta có thể biểu diễn thuật toán GFKM như sau.

Thuật toán phân cụm GFKM

1. Đưa vào một tập hợp khởi tạo các trung tâm cụm *SC*0 = {**C***j*(0)} và các giá trị ε và *M*. Gán *p* = 0.
2. Nhận một tập hợp các trung tâm cụm *SCp*, tính  cho *i* = 1 tới *N* và *j* = 1 tới *M*. Cập nhật thành viên **** sử dụng biểu thức (11). Nếu **C***j*∈*NNTi* hàng xóm gần nhất thứ *l* của **X***i*, gán = ****; ngược lại cho  = 0.
3. Tính trung tâm của mỗi cụm sử dụng biểu thức (16) để xác định một tập hợp mới của các đại diện cụm *SCp+*1 = {**C***j*(*p*+1)}.
4. Cập nhật *NNTi* cho *i* = 1 tới *N*.
5. Nếu < ε cho *j* = 1 tới *k*, sau đó dừng, với ε > 0 là một số nguyên dương rất nhở. Ngược lại gán *p = p + 1* và quay lại bước (2).

Độ phức tạp tính toán của GFKM cũng là O(*Nkt*), với *t* là số lần lặp lại.

**KẾT QUẢ NGHIÊN CỨU**

Để xác định giá trị của *M* được sử dụng trong phương thức đề xuất, một điều kiện tới hạn cụm cho phân cụm mờ được sử dụng. Điều kiện tới hạn cụm này là hàm Xie-Beni V*XB* [12], nó được định nghĩa như sau:

V*XB* =  (17)

với *N* là số lượng các điểm dữ liệu; *k* là số lượng các cụm; *SCD*minlà khoảng cách bình phương Ơ-clit nhỏ nhất giữa các trọng tâm cụm; và **X***i* là điểm dữ liệu thứ *i*; và **C***j* là trọng tâm cụm thứ *j.* Chú ý ở đây một kết quả phân cụm tốt hơn là với giá trị V*XB* nhỏ hơn.

Năm bộ dữ liệu là được sử dụng đẻ kiểm tra hiệu suất của thuật toán đề xuất. Trong bộ dữ liệu đầu tiên, nó có 49,152 điểm dữ liệu với dimension *d* = 16. Bộ dữ liệu thứ hai có 16,384 điểm dữ liệu với *d* = 16. Bộ dữ liệu cuối cùng không thẻ tìm thấy trong tham khảo [14]. Bộ dữ liệu đầu tiên là được sinh ra từ ba ảnh thực: “Lena,” “Baboon,” và “Peppers.” Với bộ dữ liệu thứ hai, nó là được sinh ra từ ảnh “Lena.” Trong ví dụ thứ ba, bộ dữ liệu “Landsat Satellite” bao gồm 6,435 điểm dữ liệu với 36 thuộc tính. Các điểm dữ liệu là được phân chia vào 6 lớp. Các giá trị thuộc tính tương ứng là số nguyên và trong phạm vi của 0 đến 255. Trong ví dụ thứ tư, bộ dữ liệu “Image Segmentation” có 2310 điểm dữ liệu với *d* = 19, với *d* là chiều của dữ liệu. Tất cả điểm dữ liệu trong bộ dữ liệu là được phân chia vào 7 lớp và các giá trị thuộc tính là thực. Trong ví dụ thứ năm, bộ dữ liệu “Yeast” được sử dụng. Bộ dữ liệu này bao gồm 1484 điểm dữ liệu với 8 thuc tính thực. Các điểm dữ liệu trong ví dụ thứ năm là được phân chia vào 10 lớp. Thêm nhiều chi tiếp mô tả về ba bộ dữ liệu trên có thể tìm thấy trong tham khảo [13].

Nó là được chỉ ra trong tham khảo [14], giá trị tối ưu của fuzzifier *q* có thể được xác định bằng biểu thức (18).

*q*= 1 + (+22.05)*d*-2 + (+0.243)*d*-0.0406In(*N*)-0.1134 (18)

với *N* là số lượng các điểm dữ liệu trong bộ dữ liệu và *d* là chiều dữ liệu. Trong các thí nghiệm sau, giá trị của *q* cho phương thức đề xuất và FKM được xác định sử dụng biểu thức (18). Cho mỗi kiểm tra, phương thức đề xuất and FKM được xử lý 100 lần. Bộ giống nhau của các trung tâm cụm khởi tạo là được sử dụng bởi cả FKM và phương thức đề xuất cho mỗi xử lý. Tất cả các tính toán là được thực hiện trên Intel i7 2.80 GHz PC với 12GB bộ nhớ. Tất cả các chương trình là được thực hiện như các ứng dụng console của Microsoft Visual Studio 2010 và được thực thi dưới môi trường Window 7.

Ví dụ 1: Bộ dữ liệu được sinh ra từ ba ảnh thực: “Lena,” “Baboon,” and “Peppers.”

Trong ví dụ này, bộ dữ liệu bao gồm 49,152 điểm dữ liệu với *d* = 16. Bộ dữ liệu này được sử dụng để xác định giá trị của *M* để nó có thể xác định kết quả phân cụm tốt hơn cho phương thức đề xuất. Bảng 1 thể hiện các giá trị của V*XB* cho nhiều giá trị khác nhau của *M*. Từ bảng 1, chúng ta có thể kết luận rằng giá trị tối ưu của *M* là 2. Trong các ví dụ sau đây, *M* = 2 là được sử dụng cho phương thức đề xuất.

Bảng1: Giá trị trung bình của V*XB* cho GFKM với 100 lần chạy sử dụng bộ dữ liệu được sinh từ ba ảnh thực: “Lena,” “Baboon,” and “Peppers”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phương thức | *k* | | | |
|  | 8 | 16 | 32 | 64 |
| GFKM (*M* = 2) | 0.5034 | 0.8288 | 1.260 | 2.106 |
| GFKM (*M* = 3) | 0.5225 | 0.8511 | 1.349 | 2.240 |
| GFKM (*M* = 4) | 0.5264 | 0.8654 | 1.4177 | 2.328 |

Trong các ví dụ sau, thêm hai chỉ số giá trị: hệ số phân chia của Bezdek (V*pc*) [15] và chỉ số Davies-Bouldin (V*DB*) [16] cũng được sử dụng để đo chất lượng phân đoạn. V*pc* được định nghĩa như sau. Chú ý rằng V*pc* càng lớn và V*DB* càng nhỏ sẽ cho kết quả phân đoạn tốt hơn.

V*pc* =  (19)

Ví dụ 2: Bộ dữ liệu được sinh từ một ảnh thực: “Lena.”

This data set having 16,384 data points with *d* = 16 is used to test statistically the performances of the proposed method and fuzzy *k*-means clustering. The sign test is a popular way of comparing the performances of algorithms [17] and will be used in this example. The test consists of *P*×*Q* cases, where *P* and *Q*, respectively, are the number of sets of initial cluster centers and number of distinct values of *k* used in the test. *P* = 20 and *Q* = 20 are used here. Table 2 presents the number of wins for the proposed method using Bezdek’s partition coefficient, Xie-Beni, and Davies-Bouldin indexes. The *z*-test says that if the number of wins for the proposed method is larger than (*P*×*Q*/2 + 1.96) = 201.96 for this test, then the proposed method is significantly better. From table 2, we can find that the numbers of obtaining better Bezdek’s partition coefficient, Xie-Beni, and Davies-Bouldin indexes for the proposed method than FKM are larger than 201.96. That is, the proposed method is significantly better than FKM in terms of clustering quality.

Table 2: Numbers of obtaining better V*pc*,V*XB* and V*DB* for the proposed method than FKM with 400 cases using the data set obtained from “Lena”

|  |  |  |
| --- | --- | --- |
| V*pc* | V*XB* | V*DB* |
| 352 | 304 | 381 |

Tables 3 and 4 present the average values of V*pc*/V*XB*/V*DB* and computing time with 100 runs for various values of *k*. From tables 3 and 4, we can find that the proposed method can reduce the computing time of FKM by 19.5% to 44.3% and obtain a better clustering quality in terms of V*pc*, V*XB*, and V*DB*. Note here that the larger V*pc* and smaller V*XB* and V*DB* imply the better clustering quality. From table 4, we can also find that the proposed method can reduce more computing time of FKM for a larger number of clusters. Note here that some approaches for fast *k*-means clustering [18, 19] can be used to further reduce the computing time of the proposed method.

Table 3: The average values of V*pc*/V*XB*/V*DB* for various *k* using the data set obtained from “Lena”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | *k* | | | |
| 4 | 8 | 16 | 32 |
| Proposed | 0.9633/0.1551/0.7847 | 0.9416/0.3209/1.196 | 0.9173/0.6174/1.364 | 0.8891/0.9927/1.507 |
| FKM | 0.9631/0.1552/0.7849 | 0.9409/0.3212/1.202 | 0.9148/0.6756/1.392 | 0.8861/1.030/1.547 |

Table 4: The average computing time (seconds) of the proposed method and FKM for various *k* using the data set obtained from “Lena”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | *k* | | | |
| 4 | 8 | 16 | 32 |
| Proposed | 1.550 | 4.667 | 1.251 | 6.421 |
| FKM | 1.936 | 6.898 | 2.316 | 11.601 |

Example 3: The “Landsat Satellite” data set.

This data set has 6,435 data points with 36 attributes. The attribute values are integer and in the range of 0 to 255. The data points are classified into 6 classes. Table 5 gives the average computing time of the proposed method and FKM, respectively, with 100 runs. From table 5, we can find that the proposed method can reduce the computing time of FKM significantly due to the proposed method converges much faster than FKM. The average values of V*pc*, V*XB* and V*DB* are given in table 6. From table 6, we can find that compared with FKM, the proposed method can obtain the same clustering quality in terms of these cluster validity indexes. Compared with FKM, the proposed method can reduce the computing time by 17.8%. That is, compared with FKM, the proposed method can reduce the computing time without sacrificing the quality of clustering quality.

Table 5: The average computing time (seconds) of the proposed method and FKM using using the “Landsat Satellite” data set

|  |  |
| --- | --- |
| Method | Computing time |
| Proposed | 2.442 |
| FKM | 2.974 |

Table 6: The average values of V*pc*,V*XB* and V*DB* for the proposed method and FKM using the “Landsat Satellite” data set

|  |  |  |  |
| --- | --- | --- | --- |
| Method | V*pc* | V*XB* | V*DB* |
| Proposed | 0.9780 | 0.3479 | 1.105 |
| FKM | 0.9780 | 0.3479 | 1.105 |

Example 4: The “Image Segmentation” data set.

In this example, the “Image Segmentation” data set has 2310 data points with *d* = 19. All the data points in this data set are divided into 7 classes. Table 7 shows the computing time of the proposed method “GFKM” and FKM. Table 8 presents the average values of V*pc*,V*XB* and V*DB* for the proposed method and FKM, each with 100 runs. From tables 7 and 8, we can conclude that compared with FKM, the proposed method has the better performance in terms of computing time; while maintains the same clustering quality. From table 8, we can also find that the proposed method can obtain a little better V*XB* value.

Table 7: The average computing time (seconds) of the proposed method and FKM using the “Image Segmentation” data set

|  |  |
| --- | --- |
| Method | Computing time |
| Proposed | 1.168 |
| FKM | 1.228 |

Table 8: The average values of V*pc*,V*XB* and V*DB* for the proposed method and FKM using the “Image Segmentation” data set

|  |  |  |  |
| --- | --- | --- | --- |
| Method | V*pc* | V*XB* | V*DB* |
| Proposed | 0.9476 | 0.5800 | 1.350 |
| FKM | 0.9472 | 0.5790 | 1.349 |

Example 5: The “Yeast” data set.

In this example, the “Yeast” data set is used to test the performances of FKM and GFKM. This data set consists of 1484 data points with *d* = 8. These data points are classified into 10 classes and the corresponding attribute values are real. Table 9 shows the computing time of the proposed method “GFKM” and FKM. From table 9, we can find that the proposed method can reduce the computing time of FKM by a factor of 1.41. Table 8 presents the average values of V*pc*,V*XB* and V*DB* for the proposed method and FKM, each with 100 runs. Compared with FKM, the proposed method can increase the value of V*pc* by 8.24% and reduce the values of V*XB* and V*DB* by 41.5% and 14.71%, respectively. From tables 9 and 10, we can also conclude that compared with FKM, the proposed method has the better performance in terms of computing time and any of the cluster validity indexes V*pc*,V*XB* and V*DB*.

Table 9: The average computing time (seconds) of the proposed method and FKM using the “Yeast” data set

|  |  |
| --- | --- |
| Method | Computing time |
| Proposed | 1.650 |
| FKM | 2.330 |

Table 10: The average values of V*pc*,V*XB*, andV*DB* for the proposed method and FKM using the “Yeast” data set

|  |  |  |  |
| --- | --- | --- | --- |
| Method | V*pc* | V*XB* | V*DB* |
| Proposed | 0.4845 | 0.5470 | 2.202 |
| FKM | 0.4476 | 0.9374 | 2.526 |