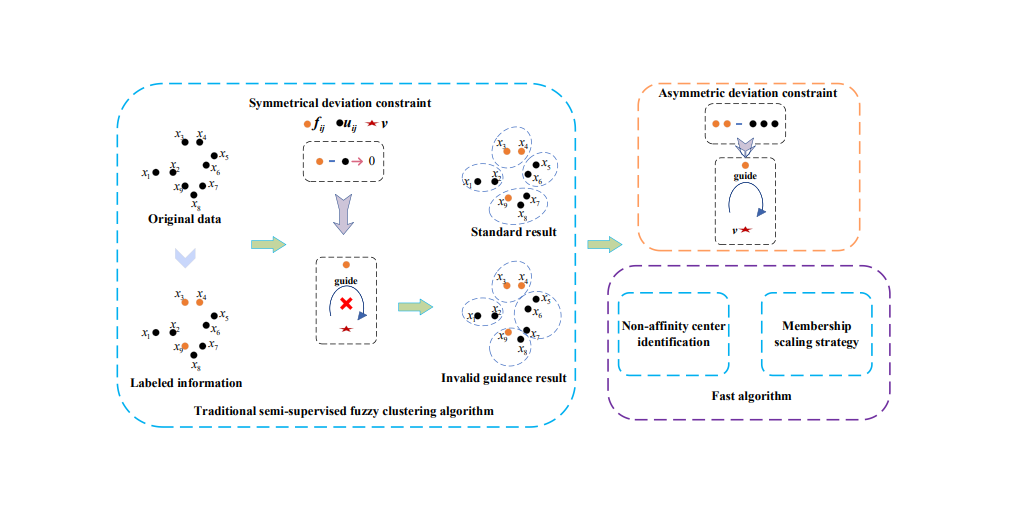
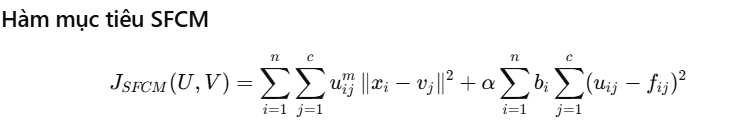
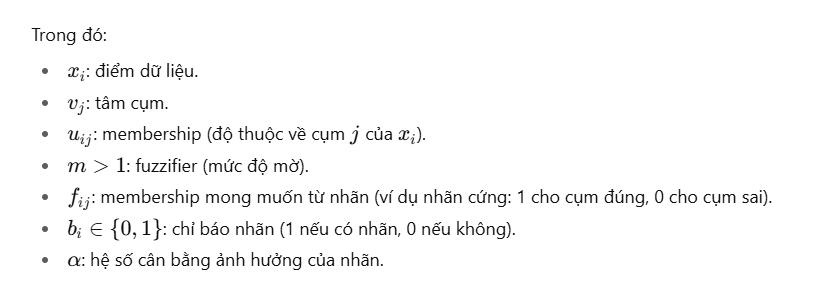
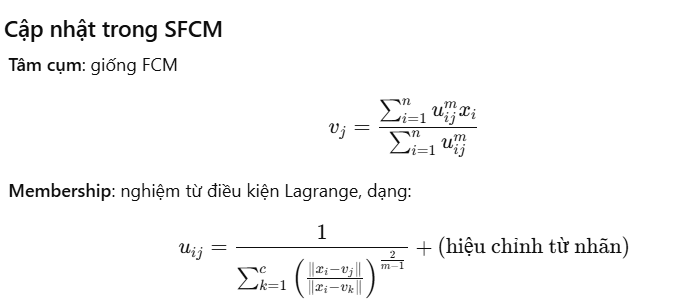
1. Cơ sở phát triển thuật toán

Bài báo này phân tích hàm mục tiêu C-means mờ bán giám sát (SFCM) truyền thống, lưu ý rằng khi mức độ thành viên của mẫu được gán nhãn phù hợp với thông tin trước đó, tác động của thông tin này lên ràng buộc độ lệch sẽ yếu đi. Điều này làm giảm hiệu quả giám sát của nó trong việc tối ưu hóa ma trận phân vùng thành viên, đặc biệt là với hệ số chính quy lớn.

Để khắc phục điều này, chúng tôi đề xuất một phương pháp C-means mờ bán giám sát mới dựa trên ràng buộc độ lệch bất đối xứng và phát triển một thuật toán tối ưu hóa lặp xen kẽ hai cấp, được hỗ trợ bởi phân tích hội tụ lý thuyết sử dụng định lý Zangwill và ma trận Hessian có biên.

* 1. Lý Thuyết SFCM
* FCM chỉ dùng dữ liệu không nhãn → nhiều khi phân cụm không đúng, nhất là dữ liệu phức tạp.
* SFCM tận dụng **một phần dữ liệu có nhãn** (semi-supervised).
* Nguyên tắc: thêm một ràng buộc vào hàm mục tiêu để ép membership của mẫu có nhãn tiến gần đến nhãn thật.



1. Lý thuyết của Asymmetric deviation-based semi-supervised fuzzy clustering algorithm (ADSFCM) (Thuật toán SFCM mới với ràng buộc bất đối xứng).

* ADSFCM thay thế ràng buộc đối xứng bằng ràng buộc bất đối xứng (Asymmetric Deviation Constraint).
* Công thức mới giữ cho ảnh hưởng của thông tin nhãn không mất đi ngay cả khi membership gần bằng nhãn.
* Điều này giúp đảm bảo vai trò giám sát luôn tồn tại. Thuật toán được xây dựng trên cơ sở tối ưu 2 tầng:   
   (1) cập nhật ma trận membership;

(2) cập nhật vị trí tâm cụm. Tính hội tụ được chứng minh

bằng định lý Zangwill và ma trận Hessian có biên.

* ADSFCM là thuật toán kế thùa từ SFCM nhưng thay vì với ràng buộc đối xứng (u−f)2 (của SFCM) đổi thành bất đối xứng (u−f)3.Thêm vào đó là cơ chế **membership scaling** và **non-affinity filtering**.
  1. **Membership scaling**

**Vấn đề:**

* Trong FCM/SFCM, sau khi cập nhật uij, các membership thường **mờ nhạt**, ví dụ:

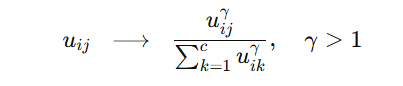
ui​=[0.55,0.40,0.05]

→ Điểm xi​ hơi thiên về cụm 1, nhưng vẫn có trọng số đáng kể ở cụm 2.

* Điều này làm tâm cụm dịch chuyển ít, dẫn đến **hội tụ chậm** và phân cụm chưa rõ rang.

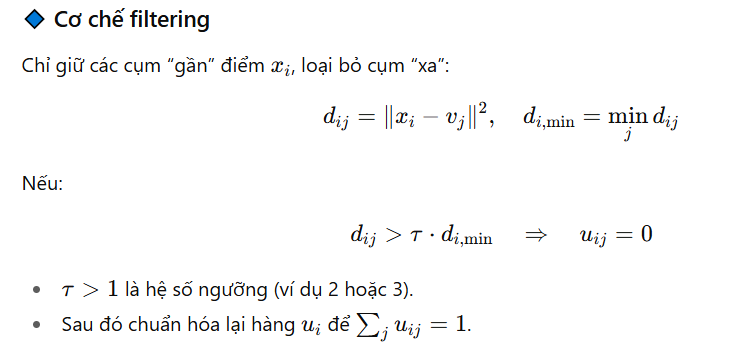
**Cơ chế scaling**

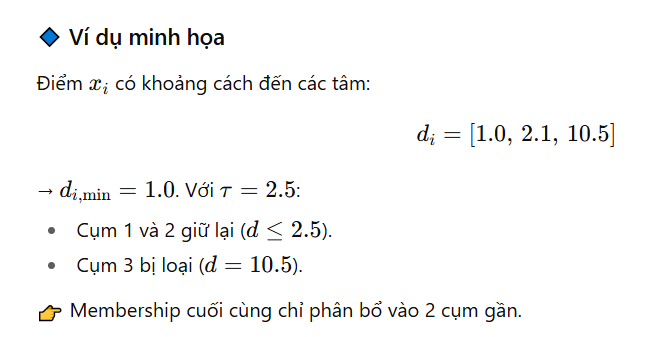
Membership scaling áp dụng phép **co giãn phi tuyến** để khuếch đại membership lớn và thu nhỏ membership nhỏ:



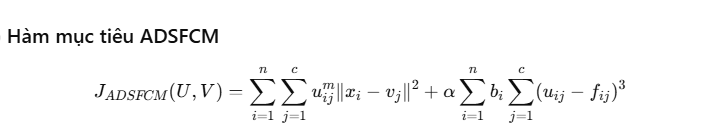
* Nếu uij​ đã lớn → sau scaling còn lớn hơn.
* Nếu uij → sau scaling càng nhỏ hơn.
* Hội tụ nhanh: vì tâm cụm được kéo mạnh mẽ hơn về phía dữ liệu gần
* Tăng tính phân biệt: giảm nhầm lẫn giữa các cụm
  1. **Non-affinity center filtering**

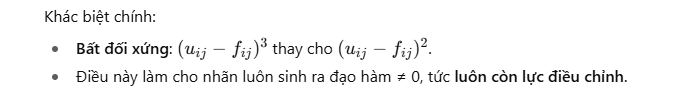
**Vấn đề**

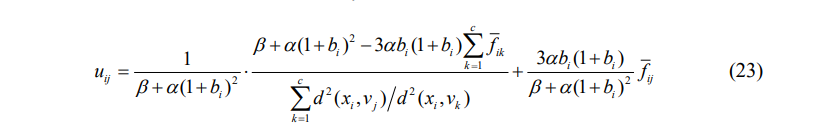
* Trong FCM/SFCM, mọi điểm dữ liệu đều có membership > 0 với mọi cụm.
* Nhưng trên thực tế, nhiều cụm **rất xa** điểm xi, gần như không liên quan.
* Việc duy trì membership này → tốn tính toán và gây nhiễu.



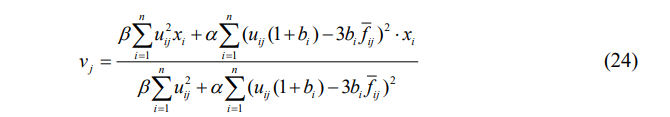
* 1. Các công thức liên quan.



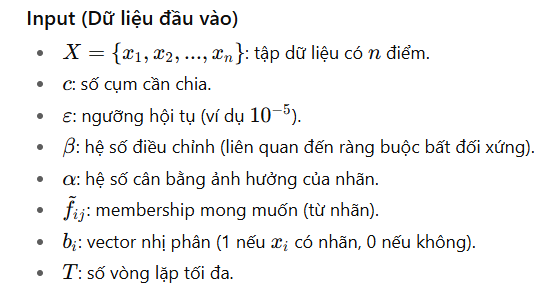
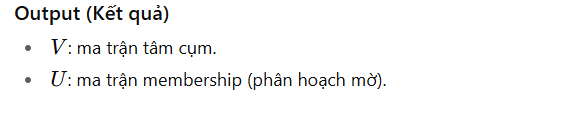
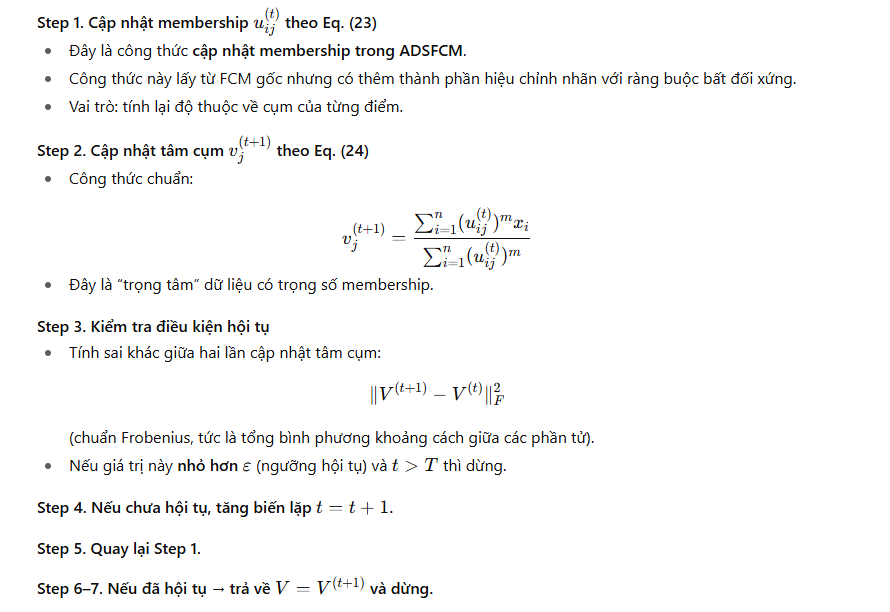


Công thức cập nhật ma trận thành viên:

Công thức cập nhật tâm cụm:

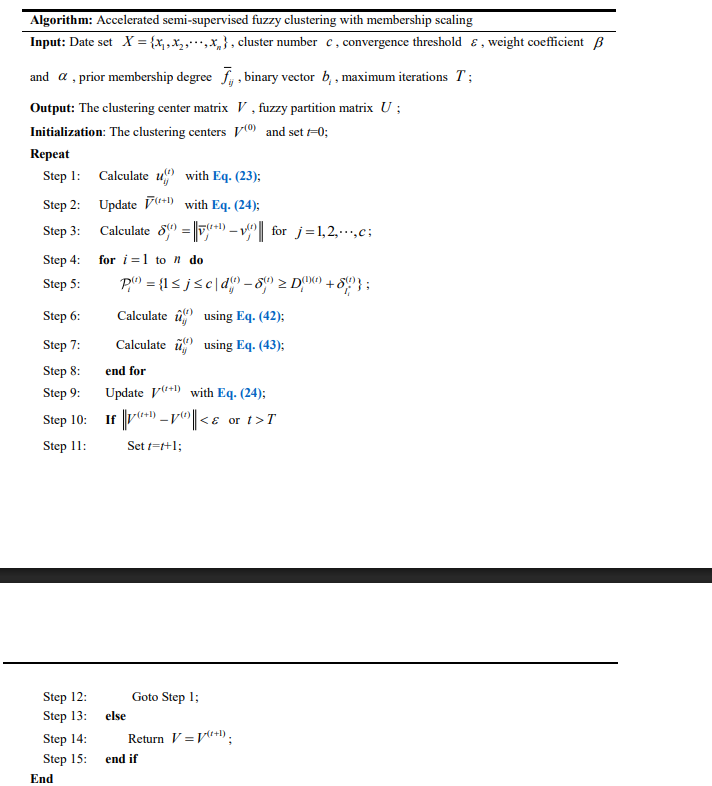


Các bước của thuật toán ADFCM core (chưa áp dụng 2 có chế tăng tốc)

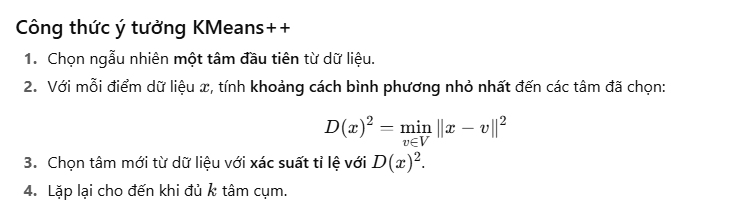


Các bước cài đặt thuật toán Fast ADSFCM

(Áp dụng cơ chế membership **Membership scaling** và Non-affinity center filtering)

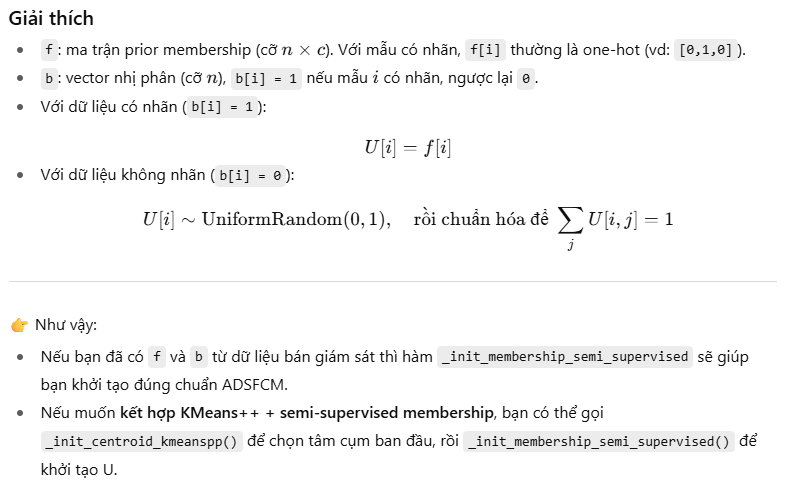


B1: Khởi tạo tâm cụm bằng Kmeans ++



B2: Khơi tạo ma trận thành viên

* Với mẫu **có nhãn**: membership được gán theo f[i] (thường là one-hot hoặc phân bố prior).
* Với mẫu **không nhãn**: membership khởi tạo ngẫu nhiên nhưng vẫn chuẩn hóa (tổng bằng 1).



**1. Non-Affinity Center Filtering**

* **Mô tả ngắn gọn**: Cơ chế này lọc bỏ các tâm cụm "không affinity" (quá xa) bằng cách kiểm tra khoảng cách giữa điểm dữ liệu và tâm cũ/mới. Nếu một tâm cụm j quá xa so với ngưỡng (dựa trên delta thay đổi vị trí tâm), thì set độ thuộc tạm thời (u\_hat) về 0 (hoặc giữ f nếu labeled). Điều này giảm tính toán không cần thiết, giúp tăng tốc.

**Cách hoạt động chi tiết**:

1. Tính ma trận khoảng cách dists từ mỗi điểm X[i] đến tất cả tâm cũ (old\_centroids), dùng norm bình phương (axis=2).
2. Tìm khoảng cách nhỏ nhất min\_dists[i] và chỉ số tâm gần nhất nearest[i] cho mỗi i.
3. Tính delta thay đổi vị trí tâm: deltas[j] = norm(new\_centroids[j] - old\_centroids[j]).
4. Copy U hiện tại vào u\_hat.
5. Loop qua mỗi điểm i: Tìm các j mà dists[i,j] - min\_dists[i] >= deltas[j] + deltas[nearest[i]] → Đây là ngưỡng lọc non-affinity.
6. Với mỗi j non-affinity:
   * Nếu labeled (b[i]==1): Giữ prior u\_hat[i,j] = f[i,j] (để duy trì supervisory effect).
   * Nếu unlabeled: Set u\_hat[i,j] = 0 (loại bỏ ảnh hưởng).

* Logging: In ra non\_affinity cho từng i, và debug u\_hat.

**2. Membership Scaling**

* **Mô tả ngắn gọn**: Sau filtering, cơ chế này scale (điều chỉnh tỷ lệ) lại ma trận u\_hat để đảm bảo tổng độ thuộc mỗi hàng (mỗi điểm) bằng 1 (ràng buộc fuzzy). Nếu tổng =0, set đều 1/c. Sau đó, re-update centroids dựa trên U mới. Điều này giữ fuzziness và cân bằng prior info.

**Cách hoạt động chi tiết**:

1. Sau khi u\_hat đã được filter (có thể có nhiều 0), tính tổng từng hàng: sums[i] = sum\_j u\_hat[i,j] (axis=1).
2. Sử dụng np.where để scale:
   * Nếu sums[i] > 0: self.membership[i,j] = u\_hat[i,j] / sums[i] (renormalize để sum=1).
   * Nếu sums[i] == 0 (tất cả non-affinity, hiếm): Set đều self.membership[i,j] = 1 / c.
3. Logging: In sums trước/sau scale để debug.
4. Re-update centroids dùng update\_centroids\_adsfcm với U mới, đảm bảo hội tụ dựa trên U đã scale.

* Công thức: U\_new = u\_hat / sums[:, None] (broadcasting để chia từng hàng).

**Các Bước Tổng Hợp Của Thuật Toán**

Dưới đây là quy trình ADSFCM (tóm tắt từ bài báo):

1. **Khởi tạo**: Set U U U (semi-supervised với f,b f, b f,b), V V V (K-means++), params (m,α,β,ϵ m, \alpha, \beta, \epsilon m,α,β,ϵ).
2. **Tính Asymmetric Deviation**: Điều chỉnh lệch lạc giữa U U U và f f f với ràng buộc bất đối xứng.
3. **Tối ưu Hóa Lặp Hai Cấp**:
   * Level 1: Update U U U (cố định V V V).
   * Level 2: Update V V V (cố định U U U).
4. **Nếu Accelerated**: Áp dụng Affinity Filtering (loại tâm xa), Membership Scaling (scale U U U), re-update V V V.
5. **Kiểm Tra Hội Tụ**: Nếu thay đổi V<ϵ V < \epsilon V<ϵ hoặc max\_iter, dừng; else lặp lại.
6. **Kết Quả**: Trả U U U (membership), V V V (centroids).

ADSFCM vượt trội SFCM ở accuracy (SI+ 0.1-0.2) và tốc độ. Nếu cần ví dụ code hoặc chi tiết toán học thêm, hãy hỏi!