fuzzy-c-means-clustering

May 31, 2024

0.0.1 Fuzzy C-Means Clustering on 'make_blobs' module

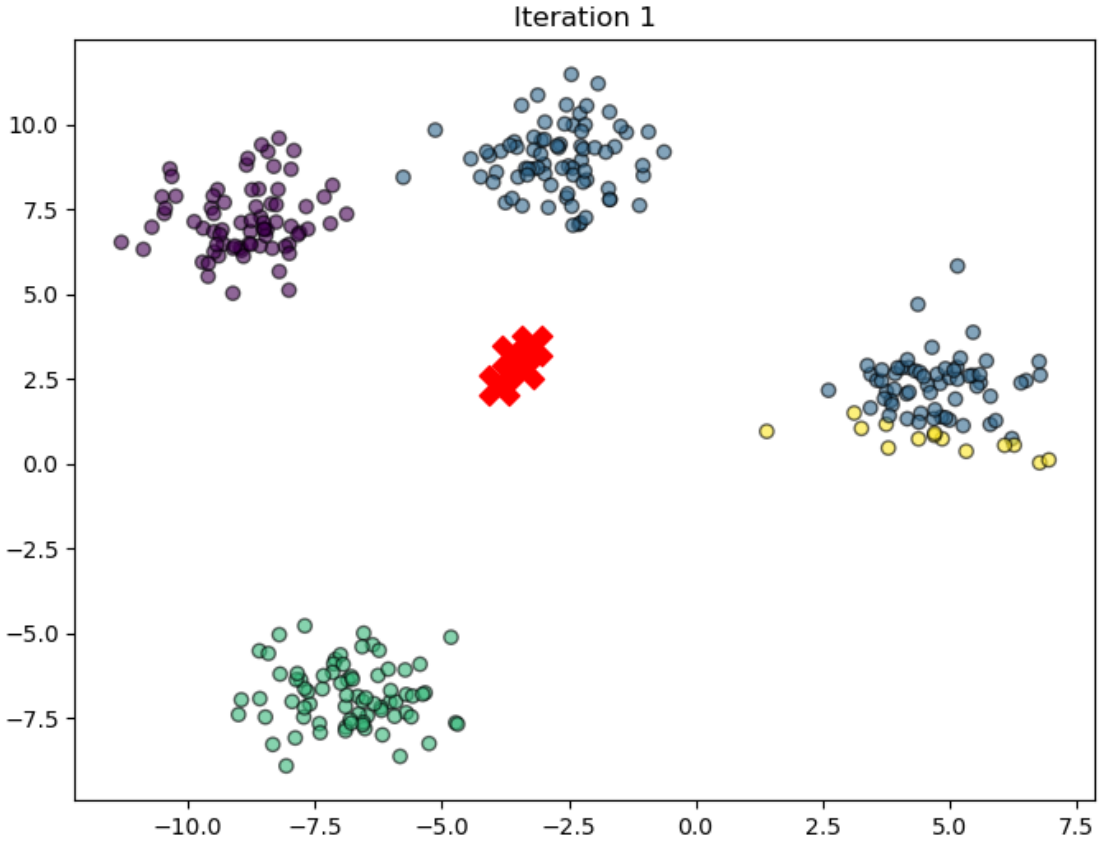
This code will 1. Generate synthetic data and initialize the membership matrix U.
2. Compute cluster centers & Update the membership matrix U.
3. Repeat until convergence or maximum iterations.
4. Plot and print membership matrix at each iterations.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

def initialize_membership_matrix(n_samples, n_clusters):
    U = np.random.dirichlet(np.ones(n_clusters), size=n_samples)
    return U

def calculate_cluster_centers(U, X, m):
    um = U ** m
    centers = (um.T @ X) / um.sum(axis=0)[:, None]
    return centers

def update_membership_matrix(U, X, centers, m):
    dist = np.linalg.norm(X[:, None, :] - centers, axis=2)
    dist = np.fmax(dist, np.finfo(np.float64).eps)  # Avoid division by zero
    power = 2 / (m - 1)
    U_new = 1 / np.sum((dist[:, :, None] / dist[:, None, :]) ** power, axis=2)
    return U_new

def plot_iteration(X, U, centers, iteration):
    cluster_labels = np.argmax(U, axis=1)
    plt.figure(figsize=(8, 6))
    plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, cmap='viridis', alpha=0.6,␣
 ↪edgecolor='k')
    plt.scatter(centers[:, 0], centers[:, 1], s=300, c='red', marker='X')
    plt.title(f'Iteration {iteration}')
    plt.show()

def print_membership_matrix(U, iteration):
    print(f"Membership matrix at iteration {iteration}:\n", U.T)
```

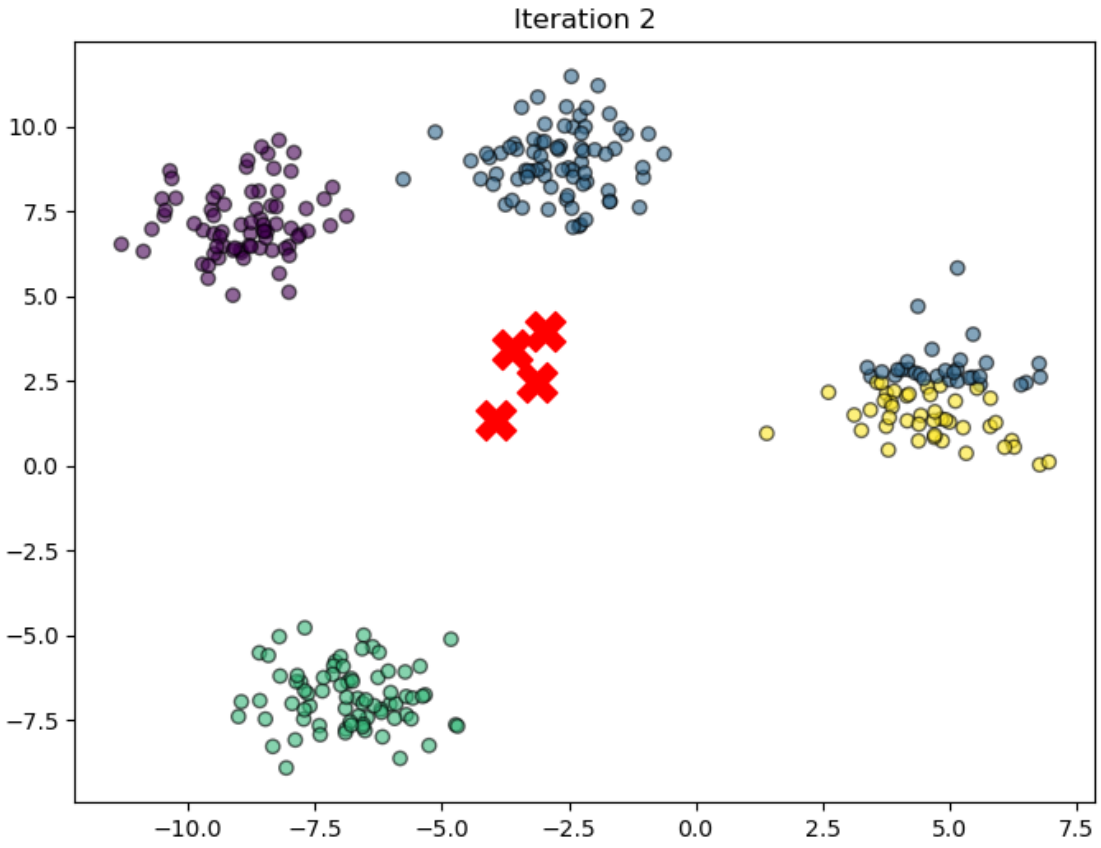
```
def fuzzy_c_means(X, n_clusters, m=2.0, max_iter=300, tol=1e-5):
    n_samples = X.shape[0]
    U = initialize_membership_matrix(n_samples, n_clusters)

    # Plot the first 10 iterations separately
    for i in range(min(10, max_iter)):
        U_old = U.copy()
        centers = calculate_cluster_centers(U, X, m)
        U = update_membership_matrix(U, X, centers, m)

        # Print the membership matrix
        print_membership_matrix(U, i + 1)

        # Plot the current state
        plot_iteration(X, U, centers, i + 1)

        if np.linalg.norm(U - U_old) < tol:
            break

    # Continue with the remaining iterations
    for i in range(10, max_iter):
        U_old = U.copy()
        centers = calculate_cluster_centers(U, X, m)
        U = update_membership_matrix(U, X, centers, m)

        if np.linalg.norm(U - U_old) < tol:
            break

    # Print final membership matrix
    print_membership_matrix(U, i + 1)

    # Plot final iteration
    plot_iteration(X, U, centers, i + 1)

    # Return final iteration number
    return centers, U, i + 1


# Example usage
if __name__ == "__main__":
    # Generate synthetic data
    X, _ = make_blobs(n_samples=300, centers=4, random_state=42)

    # Apply Fuzzy C-Means
    n_clusters = 4
    centers, U, final_iteration = fuzzy_c_means(X, n_clusters)
```

```
    # Print final membership matrix
    print_membership_matrix(U, final_iteration)

    # Plot final iteration
    plot_iteration(X, U, centers, final_iteration)
```

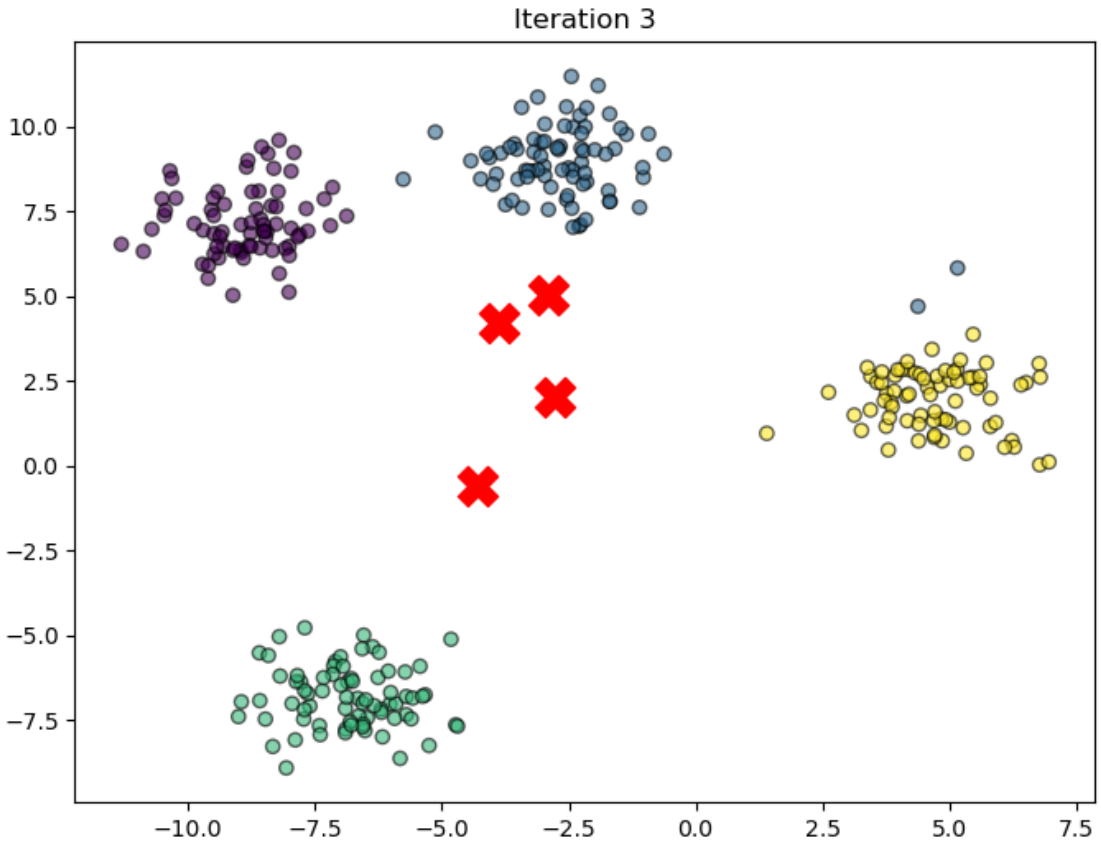
```
Membership matrix at iteration 1:
 [[0.26788489 0.26663761 0.26440438 … 0.26972252 0.26718769 0.26940154]
 [0.24995232 0.2511008  0.31346616 … 0.29127929 0.26164592 0.26525268]
 [0.24526793 0.24454469 0.18788699 … 0.20490616 0.233468   0.22908905]
 [0.23689487 0.2377169  0.23424248 … 0.23409203 0.23769838 0.23625674]]
```



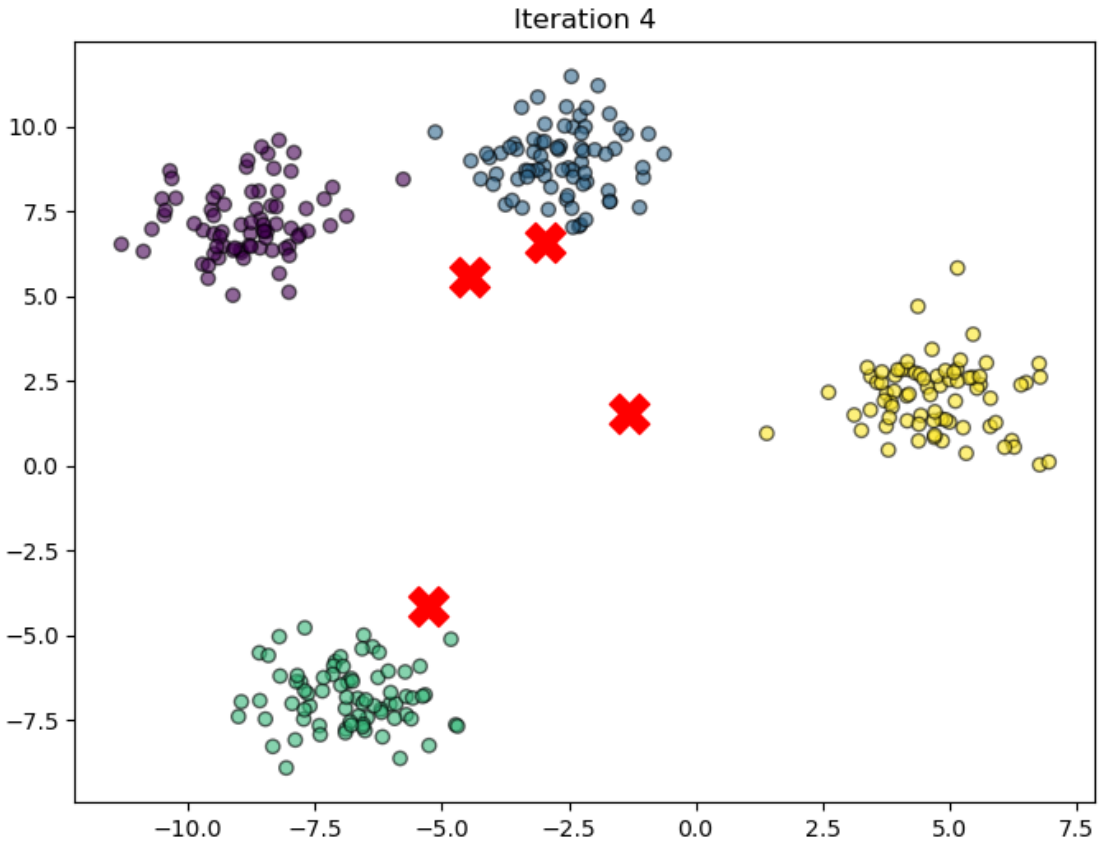

```
Membership matrix at iteration 2:
 [[0.29281913 0.28966211 0.27690149 … 0.29223162 0.29003367 0.29487857]
 [0.2621959  0.26353852 0.38263733 … 0.3388978  0.28208489 0.28944349]
 [0.22057984 0.22107546 0.13509008 … 0.1585334  0.20406389 0.19538112]
 [0.22440513 0.22572391 0.20537111 … 0.21033718 0.22381755 0.22029682]]
```



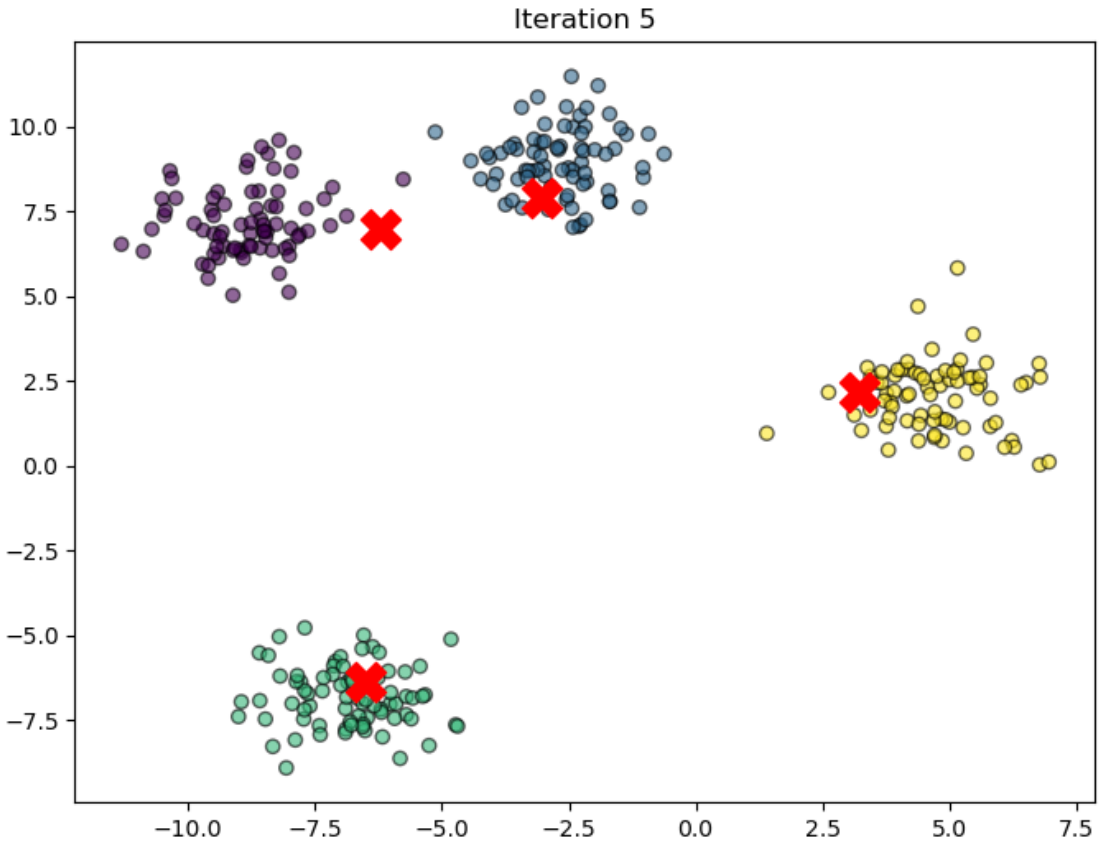

```
Membership matrix at iteration 3:
 [[0.35378057 0.34559642 0.27241496 … 0.32678162 0.34252174 0.35254941]
 [0.28626422 0.28877369 0.52628777 … 0.4346769  0.32173357 0.33456228]
 [0.1632588  0.16664625 0.06226674 … 0.08522748 0.14461644 0.13053577]
 [0.19669641 0.19898364 0.13903053 … 0.153314   0.19112825 0.18235254]]
```



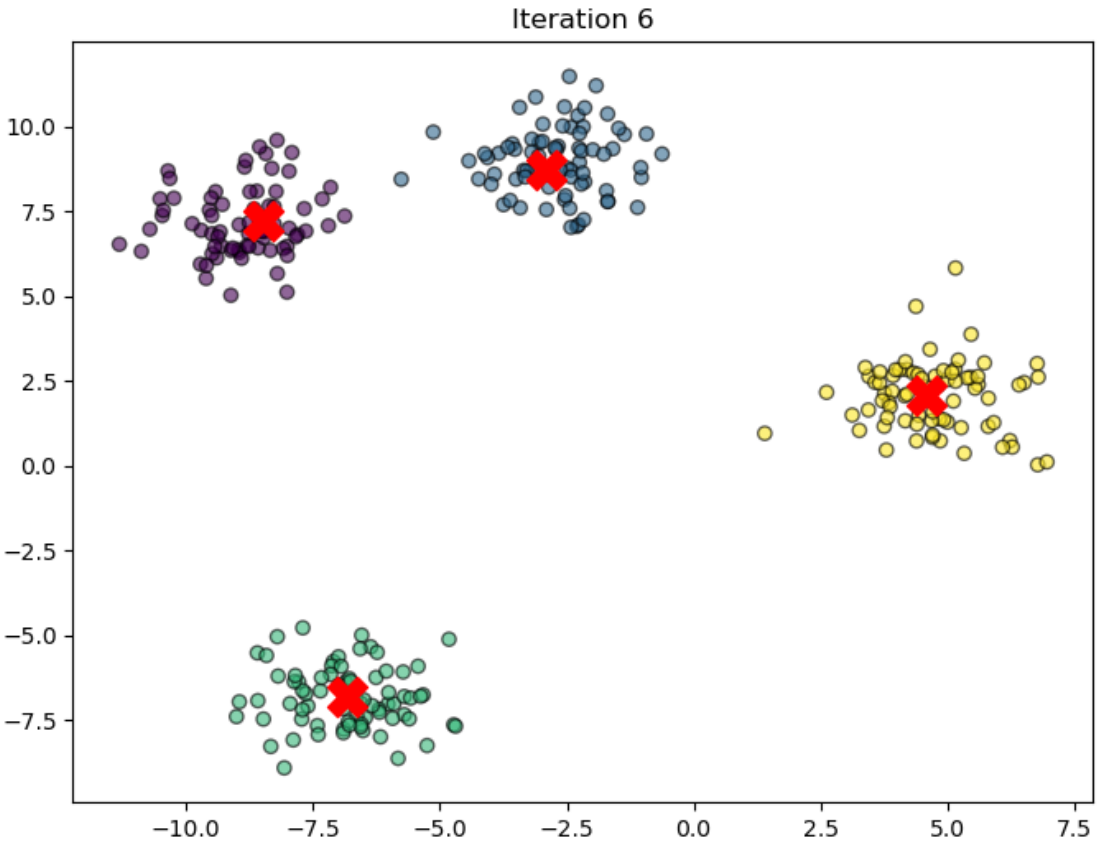

```
Membership matrix at iteration 4:
 [[0.48232841 0.46476318 0.1848494  … 0.34607878 0.45224407 0.4742335 ]
 [0.29315299 0.30139923 0.74162941 … 0.55990925 0.34802338 0.35537887]
 [0.09129002 0.0963181  0.01491474 … 0.02552977 0.07538747 0.06093984]
 [0.13322858 0.13751948 0.05860645 … 0.06848219 0.12434508 0.10944779]]
```



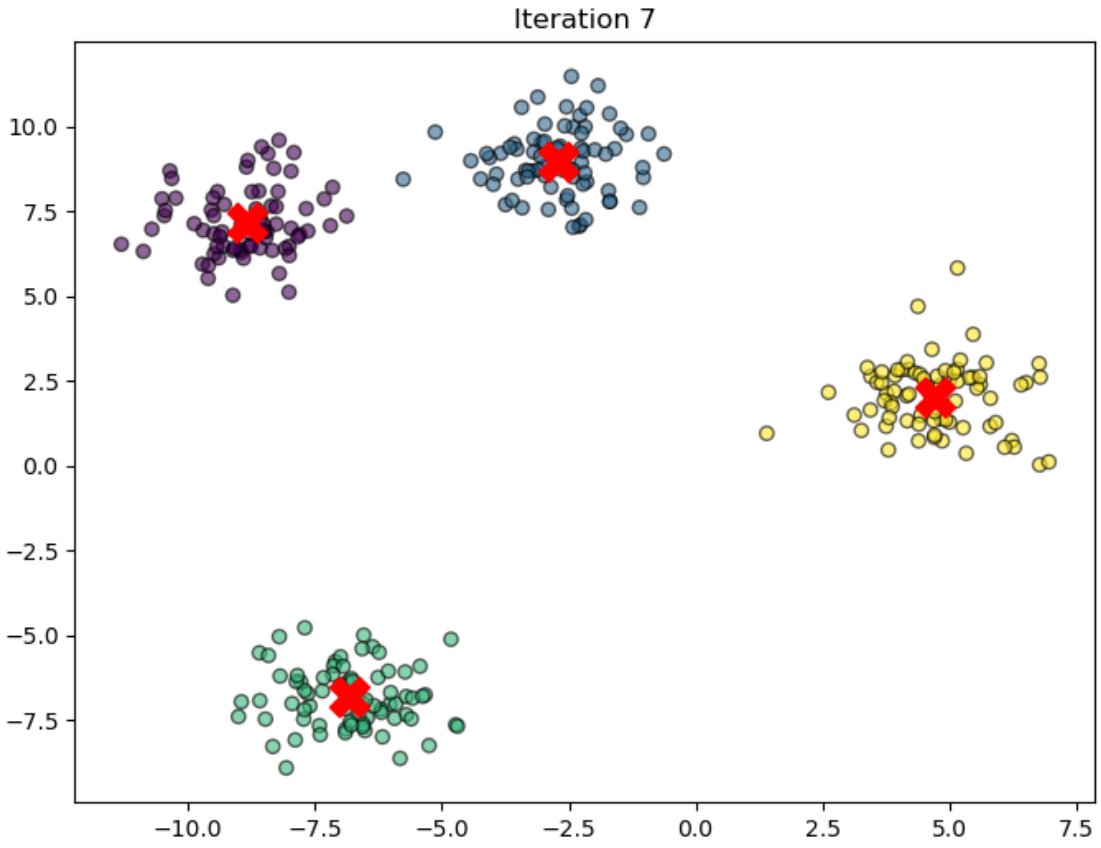

```
Membership matrix at iteration 5:
 [[0.74208686 0.71654674 0.0774042  … 0.22740146 0.70652056 0.76844241]
 [0.17524097 0.19168542 0.88571733 … 0.74328843 0.22175803 0.18421597]
 [0.04155349 0.04608717 0.0073393  … 0.00860013 0.03167465 0.01999003]
 [0.04111869 0.04568067 0.02953917 … 0.02070998 0.04004676 0.02735159]]
```



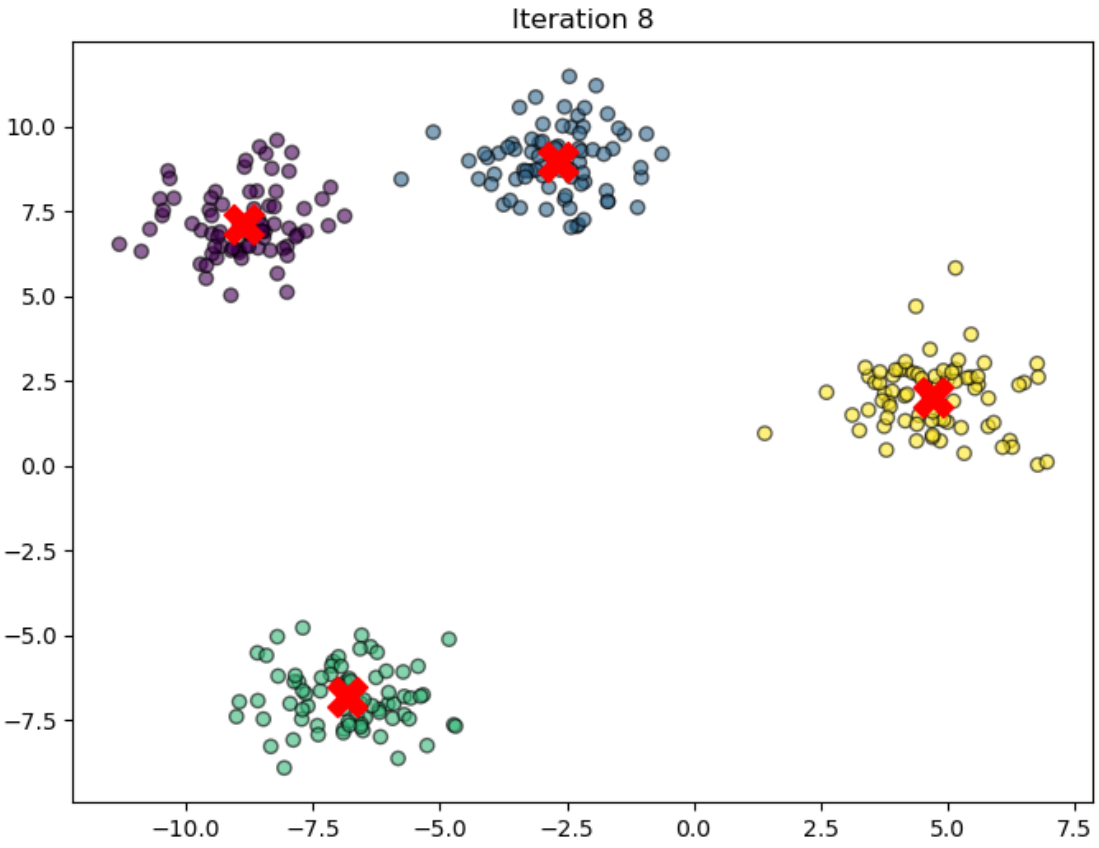

```
Membership matrix at iteration 6:
 [[0.96173722 0.95543296 0.04684745 … 0.06717002 0.89364433 0.89857317]
 [0.02603426 0.03035757 0.9139343  … 0.91485016 0.08229979 0.08219168]
 [0.0065798  0.00762226 0.00910421 … 0.00584707 0.01138944 0.00876331]
 [0.00564872 0.00658722 0.03011404 … 0.01213275 0.01266645 0.01047184]]
```



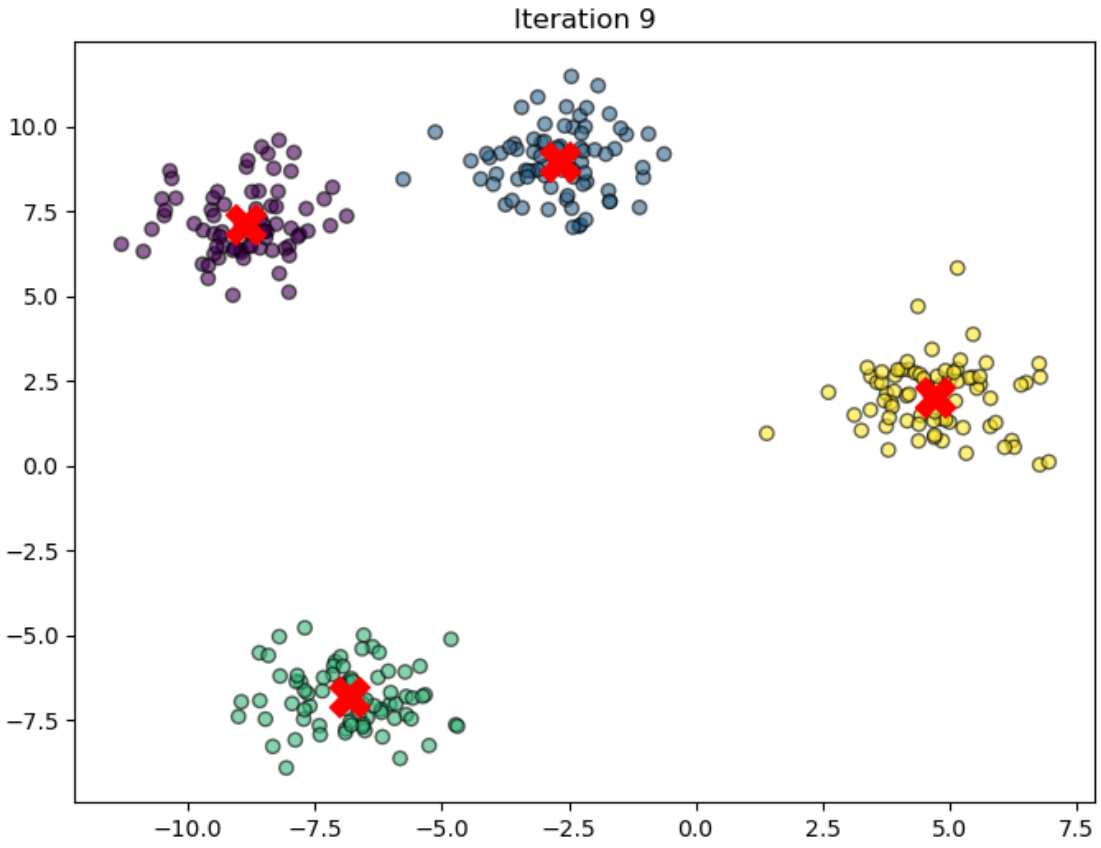

```
Membership matrix at iteration 7:
 [[0.97971719 0.97774807 0.04231114 … 0.07081549 0.89350335 0.87883974]
 [0.01351881 0.01487333 0.91928052 … 0.90766839 0.08139626 0.09689493]
 [0.00365934 0.00397898 0.00904682 … 0.00706358 0.01195373 0.01112634]
 [0.00310466 0.00339962 0.02936152 … 0.01445254 0.01314666 0.013139  ]]
```



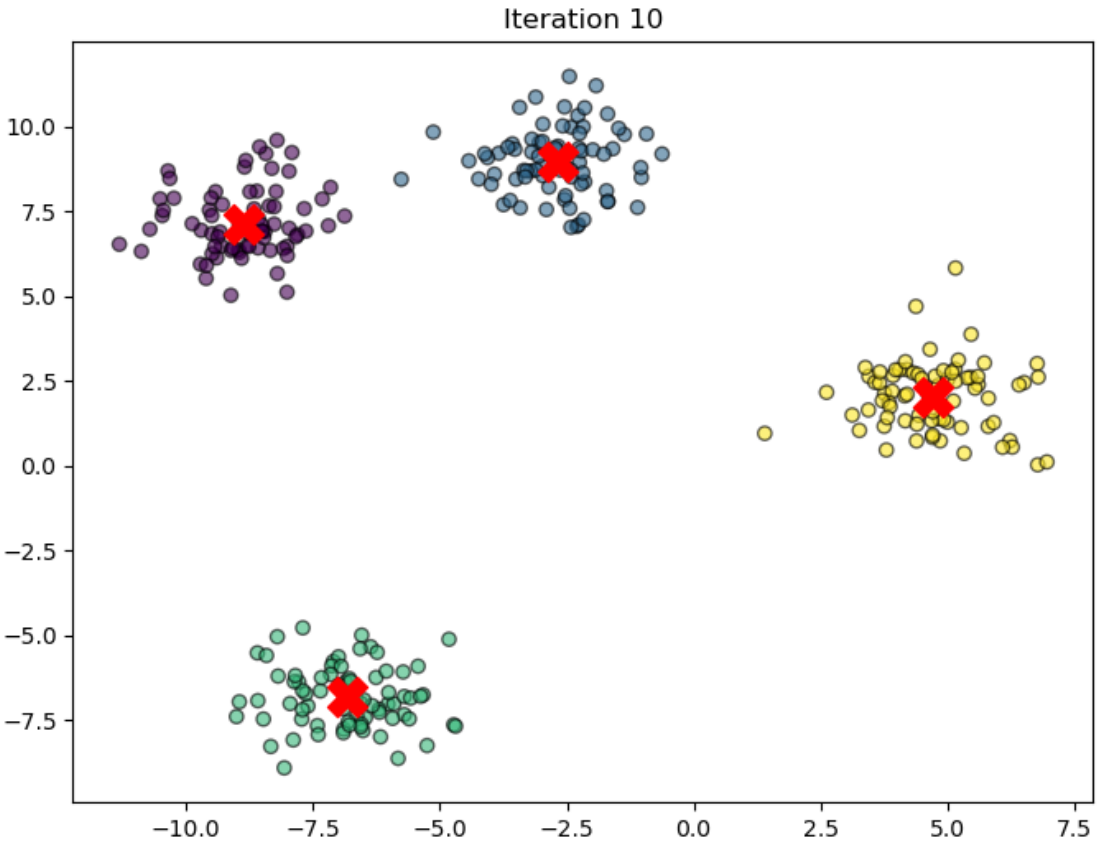

```
Membership matrix at iteration 8:
 [[0.98132193 0.97969686 0.04217113 … 0.07116036 0.89249731 0.87583905]
 [0.01242074 0.01354251 0.91920089 … 0.9069107  0.08205746 0.0991545 ]
 [0.00338678 0.00364736 0.00911011 … 0.00720514 0.01212407 0.01147253]
 [0.00287054 0.00311327 0.02951788 … 0.01472381 0.01332116 0.01353392]]
```



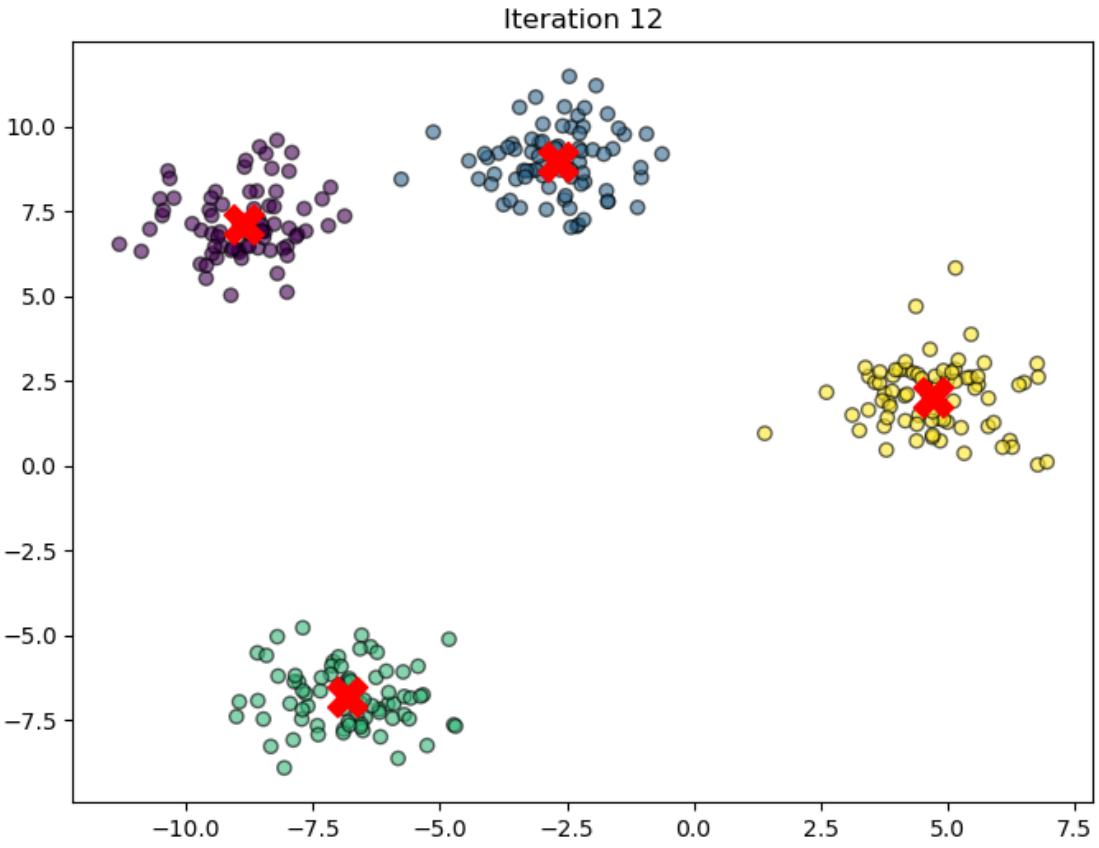

```
Membership matrix at iteration 9:
 [[0.98149745 0.97989173 0.04218686 … 0.07114164 0.89232223 0.87545974]
 [0.01230152 0.01341011 0.91913342 … 0.90690149 0.08218301 0.09944647]
 [0.00335643 0.00361378 0.00912318 … 0.00721477 0.01214812 0.01151307]
 [0.00284461 0.00308439 0.02955655 … 0.0147421  0.01334664 0.01358072]]
```



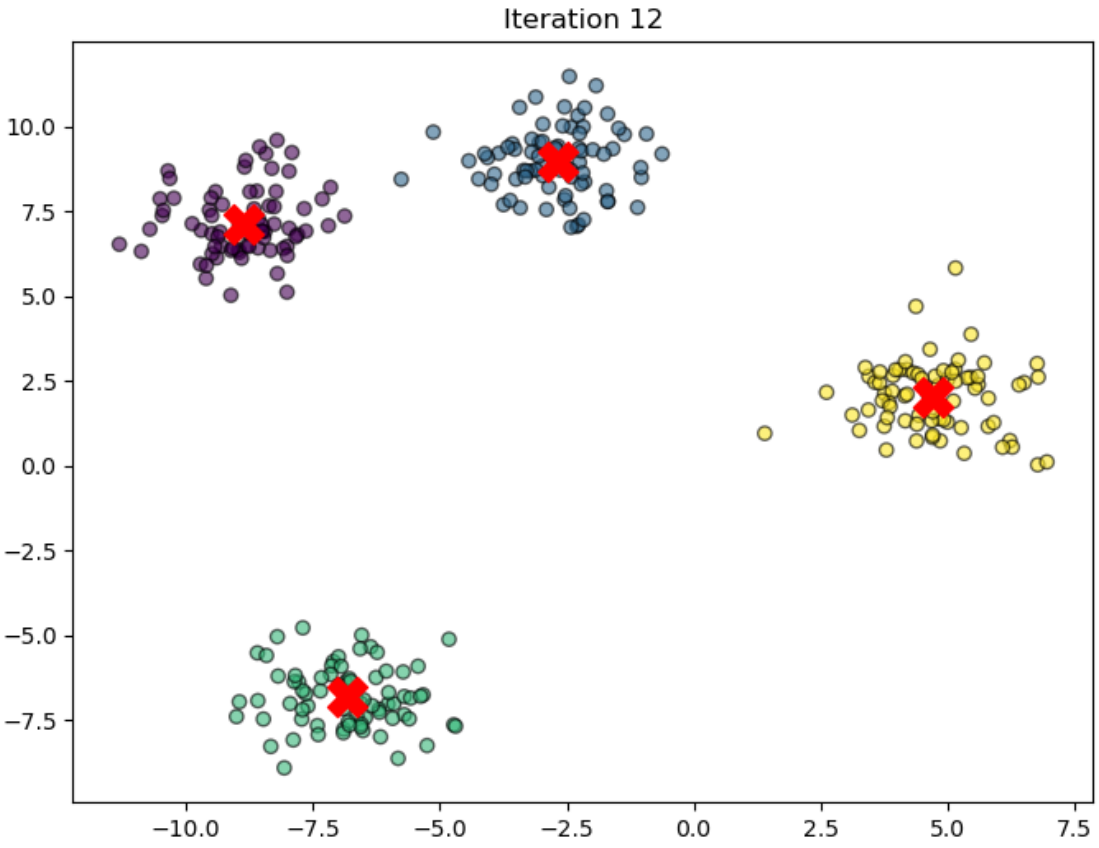

```
Membership matrix at iteration 10:
 [[0.98151798 0.97991249 0.0421921  … 0.07113139 0.89229522 0.8754105 ]
 [0.01228767 0.01339609 0.91911923 … 0.90691109 0.08220327 0.09948521]
 [0.00335282 0.00361015 0.00912534 … 0.00721501 0.01215137 0.01151792]
 [0.00284153 0.00308127 0.02956334 … 0.01474251 0.01335015 0.01358637]]
```




```
Membership matrix at iteration 12:
 [[0.98152084 0.97991514 0.04219326 … 0.07112869 0.89229058 0.87540292]
 [0.01228577 0.01339432 0.91911645 … 0.90691411 0.08220685 0.0994913 ]
 [0.0033523  0.00360967 0.00912573 … 0.00721491 0.01215187 0.0115186 ]
 [0.00284109 0.00308087 0.02956457 … 0.0147423  0.0133507  0.01358718]]
```




```
Membership matrix at iteration 12:
 [[0.98152084 0.97991514 0.04219326 … 0.07112869 0.89229058 0.87540292]
 [0.01228577 0.01339432 0.91911645 … 0.90691411 0.08220685 0.0994913 ]
 [0.0033523  0.00360967 0.00912573 … 0.00721491 0.01215187 0.0115186 ]
 [0.00284109 0.00308087 0.02956457 … 0.0147423  0.0133507  0.01358718]]
```


Iteration 12
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5