k 均值算法

Pick $\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \dots, \boldsymbol{c}^{(k)}$ at random

For $t = 1,2, \dots, N$

1.Assign to centers：

$\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_k = \emptyset$

For $i = 1,2, \dots, m$:

$j^* = \text{argmin}_{1 \le j \le k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}^{(j)} \right\|$

$C_{j^*} \leftarrow C_{j^*} \cup \left\{ \boldsymbol{x}^{(i)} \right\}$

2. Adjust centers

For $j = 1,2, \dots, k$：

$$c^{(j)} \leftarrow \frac{1}{|C_j|} \sum_{\boldsymbol{x} \in C_j} \boldsymbol{x}$$

Return $\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \dots, \boldsymbol{c}^{(k)}$

图 11.1　k 均值算法描述

machine_learning.clustering.lib.kmeans

```
1  import numpy as np
2
3  class KMeans:
4      def __init__(self, n_clusters = 1, max_iter = 300, random_state =0):
5          self.k = n_clusters
6          self.N = max_iter
7          np.random.seed(random_state)
8
9      def assign_to_centers(self, centers, X):
10         assignments = []
11         for i in range(len(X)):
12             distances = [np.linalg.norm(X[i] - centers[j], 2) for j in
range(self.k)]
13             assignments.append(np.argmin(distances))
14         return assignments
15
16     def adjust_centers(self, assignments, X):
17         new_centers = []
18         for j in range(self.k):
19             cluster_j = [X[i] for i in range(len(X)) if assignments[i] == j]
20             new_centers.append(np.mean(cluster_j, axis = 0))
21         return new_centers
22
23     def fit_transform(self, X):
24         idx = np.random.randint(0, len(X), self.k)
25         centers = [X[i] for i in idx]
26         for t in range(self.N):
27             assignments = self.assign_to_centers(centers, X)
28             centers = self.adjust_centers(assignments, X)
29         return np.array(centers), np.array(assignments)
```

图 11.2　*k* 均值算法实现

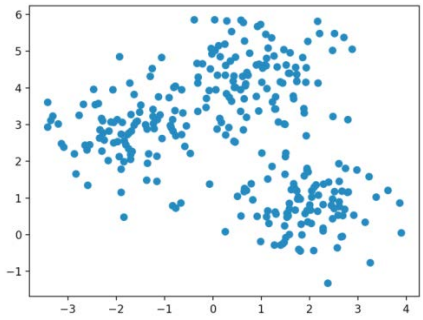


图 11.3　墨渍数据集中的 300 个不显示标签的数据样本

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from machine_learning.clustering.lib.kmeans import KMeans

X, y = make_blobs(n_samples = 300, centers = 3, cluster_std = 0.8)
model = KMeans(n_clusters = 3, max_iter = 10)
centers, assignments = model.fit_transform(X)

plt.scatter(X[:, 0], X[:, 1], c = assignments)
plt.scatter(centers[:, 0], centers[:, 1], c = 'b')
plt.show()
```

图 11.4　墨渍数据聚类的k 均值算法

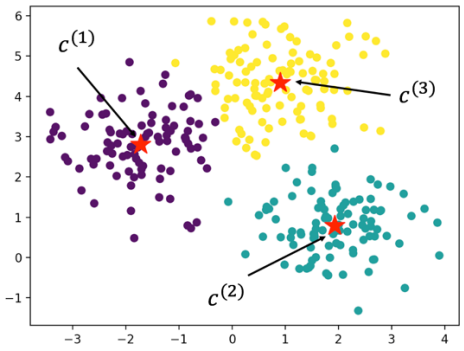


图 11.5　k 均值算法对墨渍数据的聚类结果

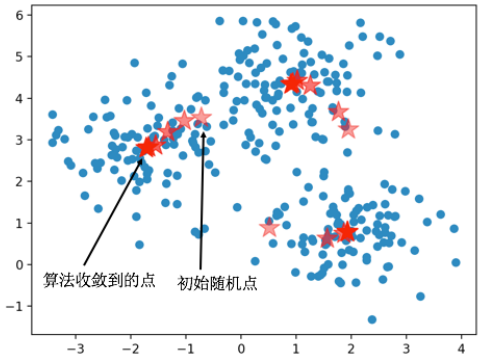


图 11.6　k 均值算法对墨渍数据聚类的收敛轨迹图

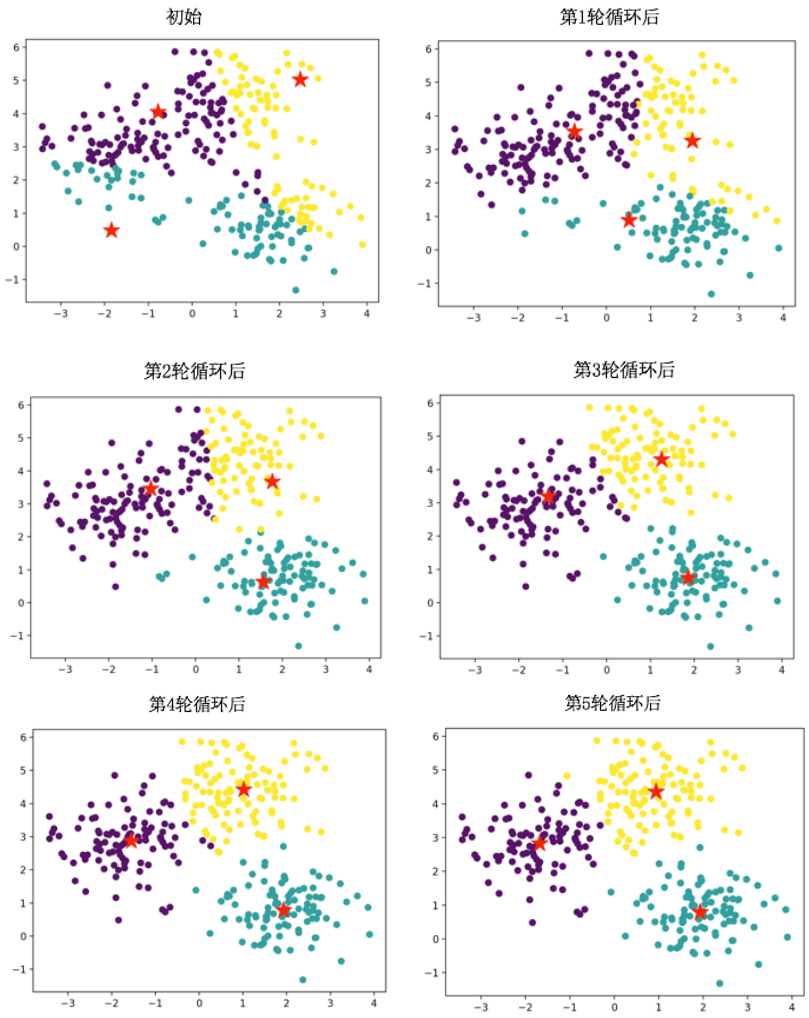


图 11.7　k 均值算法对墨渍数据聚类的过程图

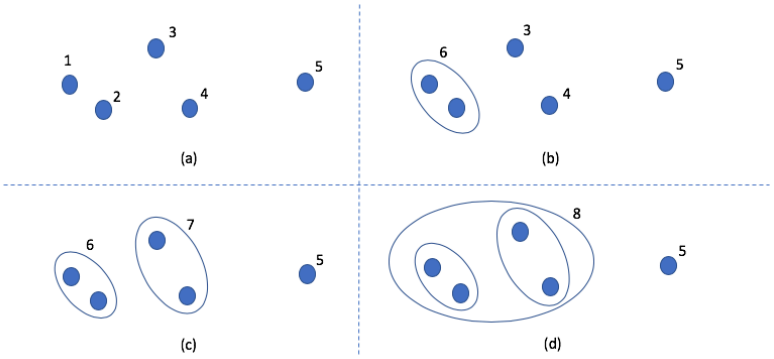


图 11.8　合并聚类算法将 5 个数据样本聚为 2 类的过程

合并聚类算法

For $i = 1,2,\ldots,m$: $C_i = \{\boldsymbol{x}^{(i)}\}$

$\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$

new_id $= |C| + 1$

While $|\mathcal{C}| > k$:

Pick the closest two clusters C_{i_1}, $C_{i_2} \in \mathcal{C}$

$C_{\text{new\_id}} = C_{i_1} \cup C_{i_2}$

$\mathcal{C} \leftarrow \mathcal{C} - C_{i_1} - C_{i_2} + C_{\text{new\_id}}$

new_id $\leftarrow$ new_id $+ 1$

Return $\mathcal{C}$

图 11.9　合并聚类算法描述

machine_learning.clustering.lib.agglomerative_clustering

```
1  import numpy as np
2  import heapq
3
4  class AgglomerativeClustering:
5      def __init__(self, n_clusters = 1):
6          self.k = n_clusters
7
8      def fit_transform(self, X):
9          m, n = X.shape
10         C, centers = {}, {}
11         assignments = np.zeros(m)
12         for id in range(m):
13             C[id] = [id]
14             centers[id] = X[id]
15             assignments[id] = id
16         H = []
17         for i in range(m):
18             for j in range(i+1, m):
19                 d = np.linalg.norm(X[i] - X[j], 2)
20                 heapq.heappush(H, (d, [i, j]))
21         new_id = m
22         while len(C) > self.k:
23             distance, [id1, id2] = heapq.heappop(H)
24             if id1 not in C or id2 not in C:
25                 continue
26             C[new_id] = C[id1] + C[id2]
27             for i in C[new_id]:
28                 assignments[i] = new_id
29             del C[id1], C[id2], centers[id1], centers[id2]
30             new_center = sum(X[C[new_id]]) / len(C[new_id])
31             for id in centers:
32                 center = centers[id]
33                 d = np.linalg.norm(new_center - center, 2)
34                 heapq.heappush(H, (d, [id, new_id]))
35             centers[new_id] = new_center
36             new_id += 1
37         return np.array(list(centers.values())), assignments
```

图 11.10　合并聚类的优先队列算法

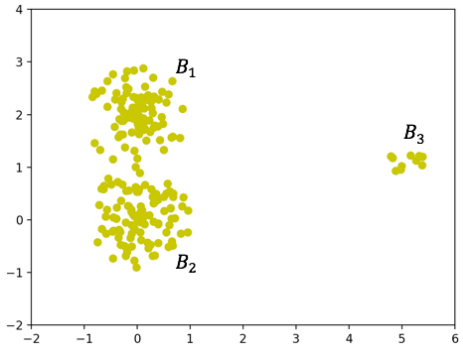


图 11.11　例 11.2 的数据样本分布

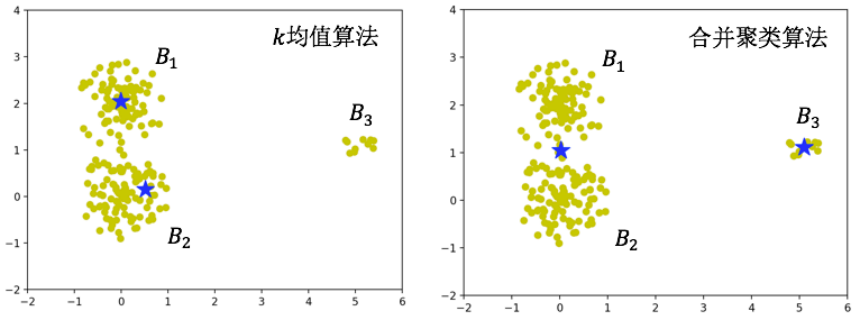
```
import numpy as np
import machine_learning.lib.kmeans as km
import machine_learning.clustering.lib.agglomerative_clustering as ac

def generate_ball(x, radius, m):
    r = radius * np.random.rand(m)
    pi = 3.14
    theta = 2 * pi * np.random.rand(m)
    B = np.zeros((m,2))
    for i in range(m):
        B[i][0] = x[0] + r[i] * np.cos(theta[i])
        B[i][1] = x[1] + r[i] * np.sin(theta[i])
    return B

B1 = generate_ball([0,0], 1, 100)
B2 = generate_ball([0,2], 1, 100)
B3 = generate_ball([5,1], 0.5, 10)
X = np.concatenate((B1, B2, B3), axis=0)

kmeans = km.KMeans(n_clusters = 2)
print("k means centers: {}".format(kmeans.fit_transform(X)))
agg = ac.AgglomerativeClustering(n_clusters = 2)
print("agglomerative centers: {}".format(agg.fit_transform(X)))
```

图 11.12　例 11.2 中数据样本的聚类分析



(a) (b)

图 11.13　两种聚类算法对例 11.2 的数据的聚类结果

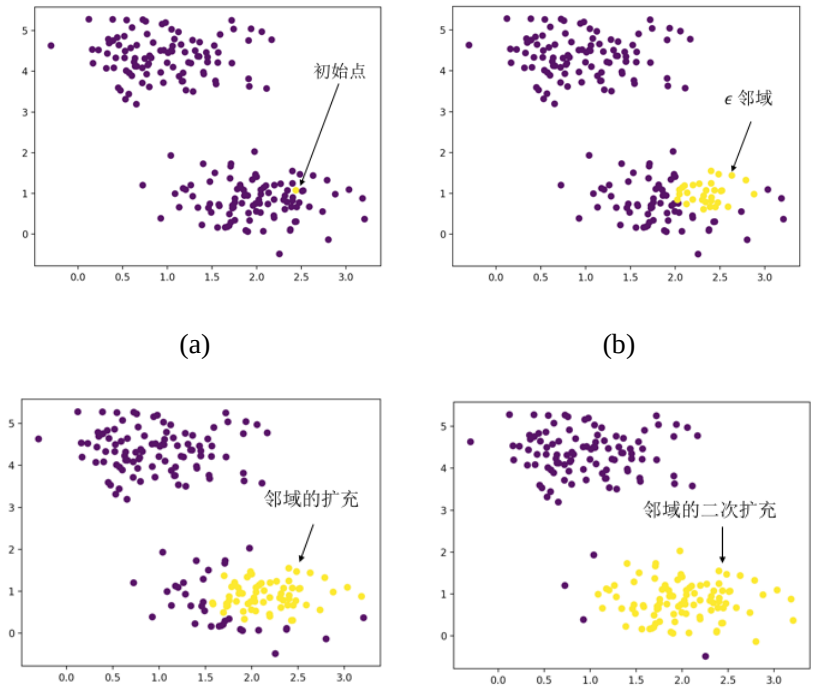


图 11.14　DBSCAN 算法生成一个类的过程

DBSCAN 算法

For $i = 1,2,\ldots,m$: assignments$(i) = 0$

$id = 1$

For $i = 1,2,\ldots,m$:

    If assignments$(i) = 0$ and $\left|N\left(\boldsymbol{x}^{(i)}, \epsilon\right)\right| >$ min_sample :

        GrowCluster$(i,\ id,$ assignments$)$

        $id \leftarrow id + 1$

Return assignments

图 11.15　DBSCAN 算法描述

GrowCluster($i,\ id,$ assignments)

assignments$(i) \leftarrow id$

$Q = N\left(\boldsymbol{x}^{(i)}, \epsilon\right)$

While $|Q| > 0$:

$j = Q.pop()$

If assignments$(j) = 0$:

assignments$(j) \leftarrow id$

If $\left|N\left(\boldsymbol{x}^{(j)}, \epsilon\right)\right| >$ min_sample:

$Q.push\left(N\left(\boldsymbol{x}^{(j)}, \epsilon\right)\right)$

Return

图 11.16　类的生成算法描述

machine_learning.clustering.lib.dbscan

```
1  import numpy as np
2
3  class DBSCAN:
4      def __init__(self, eps = 0.5, min_sample = 5):
5          self.eps = eps
6          self.min_sample = min_sample
7
8      def get_neighbors(self, X, i):
9          m = len(X)
10         distances = [np.linalg.norm(X[i] - X[j], 2) for j in range(m)]
11         neighbors_i = [j for j in range(m) if distances[j] < self.eps]
12         return neighbors_i
13
14     def grow_cluster(self, X, i, neighbors_i, id):
15         self.assignments[i] = id
16         Q = neighbors_i
17         t = 0
18         while t < len(Q):
19             j = Q[t]
20             t += 1
21             if self.assignments[j] == 0:
22                 self.assignments[j] = id
23                 neighbors_j = self.get_neighbors(X, j)
24                 if len(neighbors_j) > self.min_sample:
25                     Q += neighbors_j
26
27     def fit_transform(self, X):
28         self.assignments = np.zeros(len(X))
29         id = 1
30         for i in range(len(X)):
31             if self.assignments[i] != 0:
32                 continue
33             neighbors_i = self.get_neighbors(X, i)
34             if len(neighbors_i) > self.min_sample:
35                 self.grow_cluster(X, i, neighbors_i, id)
36                 id += 1
37         return self.assignments
```

图 11.17　DBSCAN 算法

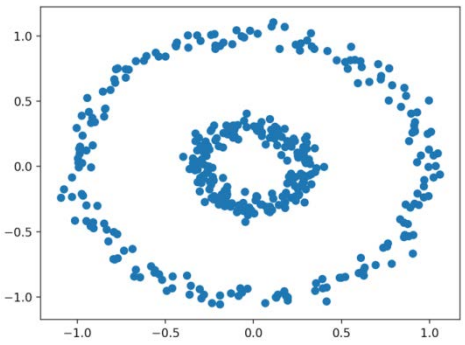


图 11.18　同心圆数据集中的 400 个数据样本

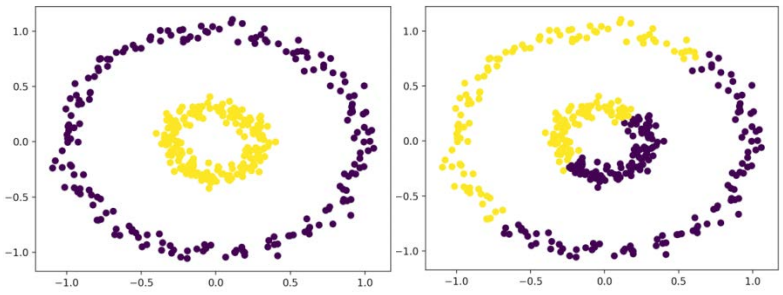
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from machine_learning.clustering.lib.dbscan import DBSCAN
from machine_learning.clustering.lib.kmeans import KMeans

np.random.seed(0)
X, y = make_circles(n_samples=400, factor=.3, noise=.05)

dbscan = DBSCAN(eps = 0.5, min_sample = 5)
db_assignments = dbscan.fit_transform(X)
kmeans = KMeans(n_clusters = 2)
km_centers, km_assignments = kmeans.fit_transform(X)

plt.figure(1)
plt.scatter(X[:, 0], X[:,1], c = db_assignments)
plt.figure(2)
plt.scatter(X[:, 0], X[:,1], c = km_assignments)
plt.show()
```

图 11.19　同心圆聚类的 DBSCAN 算法和 k 均值算法



(a) (b)

图 11.20　两种算法的聚类结果