**西安电子科技大学人工智能学院**

**模式识别 课程实践报告**

成 绩

**实践课题名称 Kmeans和FCM算法性能比较**

班级 2020039 姓名 刘焕宇 学号 20009200770

实践日期 2022 年 11 月

|  |
| --- |
| 指导教师评语：  指导教师：  年 月 日 |

一、实验内容

聚类模型可以建立在无类标记的数据上，是一种无监督的学习算法。尽管全球每日新增数据量以指数级别增长，但是大部分数据属于无标注甚至非结构化。所以相对于监督学习，不需要标注的无监督学习蕴含了巨大的潜力与价值。动态聚类算法，根据数据自身的距离或相似度将他们划分为若干组，划分原则是组内样本最小化而组间距离最大化。直接给出一个样本所属簇的算法属于硬聚类，给出所属簇概率、隶属度等属于软聚类算法。

K均值算法在1967年由提出,是一种基于划分的动态聚类算法，同时也是一种具有较大影响力的无监督学习算法。该算法的优点是思想简单易行，其算法思想采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为簇是由距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。算法时间复杂性接近线性，对大规模数据的挖掘具有高效性和可伸缩性，在工程分类等领域中有着广泛的应用。

1973年，提出了里程碑式的模糊均值聚类算法，通过引入样本到聚类中心的隶属度，使准则函数不仅可微，且软化了模式的归属。它通过优化目标函数得到每个样本点对所有类中心的隶属度，从而决定样本点的类属以达到自动对样本数据进行分类的目的。

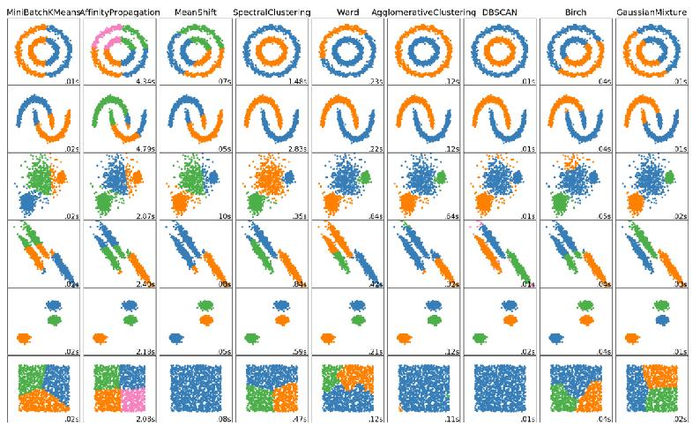
本实验聚焦于探究动态聚类算法中，和这二者分别代表硬聚类、软聚类的算法性能比较。

图1 聚类算法概述

1. 实验原理
2. Kmeans算法介绍

是一种典型的基于相似性度量的方法，目标时根据输入参数将数据集划分为个簇，根据初始值、相似度、聚类均值计算策略的不同，有很多-均值的变种。在数据分布接近球体的情况下，该算法有较好的聚类效果。

算法目标是优化各个数据与其对应的聚类中心点的误差平方和最小：

其中为第类簇的目标函数，是类的均值向量，为聚类个数。

算法流程如下：

**Step1.**初始化：随机选择个样本点，并将其视为各聚类的初始中心。

**Step2.**按照最小距离法则逐个将样本划分到以距离中心为代表的个类中。

**Step3.**计算聚类准则函数，重新计算各类的聚类中心。

**Step4.**重复Step2和3，直到聚类中心无改变或目标函数不减少（阈值）。

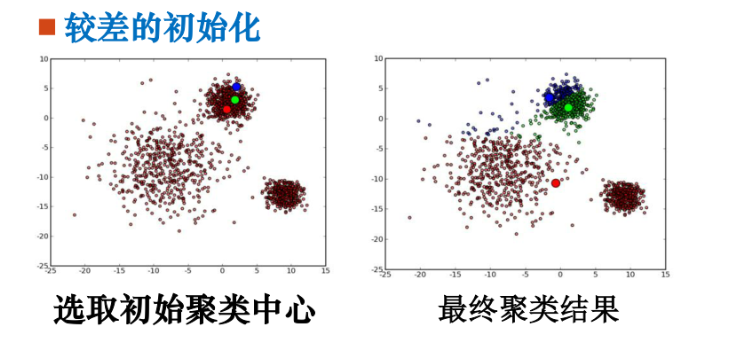
算法简单快速，对于大数据集，该算法是相对可伸缩和高效率的，当类内密集，类间区别明显时（球形聚类），效果较好。但不适合发现非凸面的聚类，对噪声与孤立点较敏感，且受初始值，聚类数影响较大。

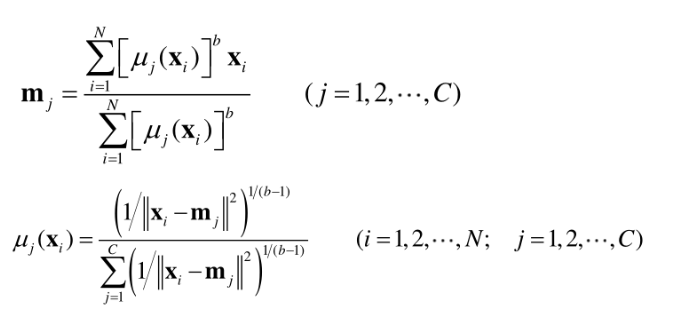
图2 Kmeans聚类易受影响

1. FCM算法介绍

属于硬聚类算法，它把数据点划分到确切的某一簇中。而在模糊聚类（也称软聚类）中，数据点则可能归属于不止一个聚类中，并且这些聚类与数据点通过一个成员水平（实际上类似于模糊集合中隶属度的概念）联系起来。成员水平显示了数据点与某一聚类之间的联系有多强。模糊聚类就是计算这些成员水平，以此为依据决定聚类的过程。

模糊均值算法的聚类准则函数增加了一个隶属度矩阵，各聚类的隶属度和为：

其中是一个可以控制聚类结果的模糊程度常数，约束条件为一个样本属于各个聚类的隶属度之和为。

利用拉格朗日乘数法求解，可以得到与的修改公式：

其余迭代步骤大体上与相似。

1. 聚类评价指标
   1. 准确率（纯度）：分类正确样本占总样本比例。
   2. 熵()：每个样本分类于每个类别的概率与其对数乘积之和，之后取反。再对每个样本的熵加权平均。
   3. 兰德指数()：

其中a、b分别表示在聚类前后是否同类别的对数，分母为组合对数。将其归一化至即为调整兰德指数()。

三、实验结果与分析

声呐数据集来源于，是初学机器学习常用的数据集之一。共有208行60列特征，数据分为两类，标签为R/M。表示208个观察对象，60个不同角度返回的力度值，二分类结果是岩石/金属。

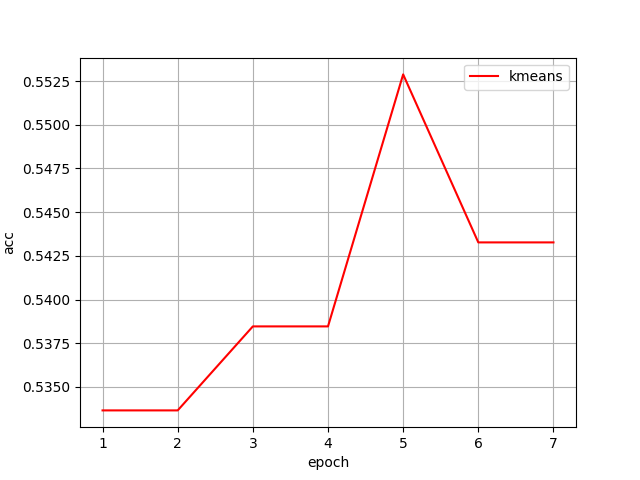
 采用算法，迭代最大步数设置为50，可得如下实验结果：

图3 K-means实验结果图

准确率只比随机猜高了四个百分点，可见对于高维非凸、非球形分布数据、分类效果并不是很理想。

采用算法，分别采用归一化指标（正确率），以及隶属度概率指标（熵），可以得到如下实验结果：

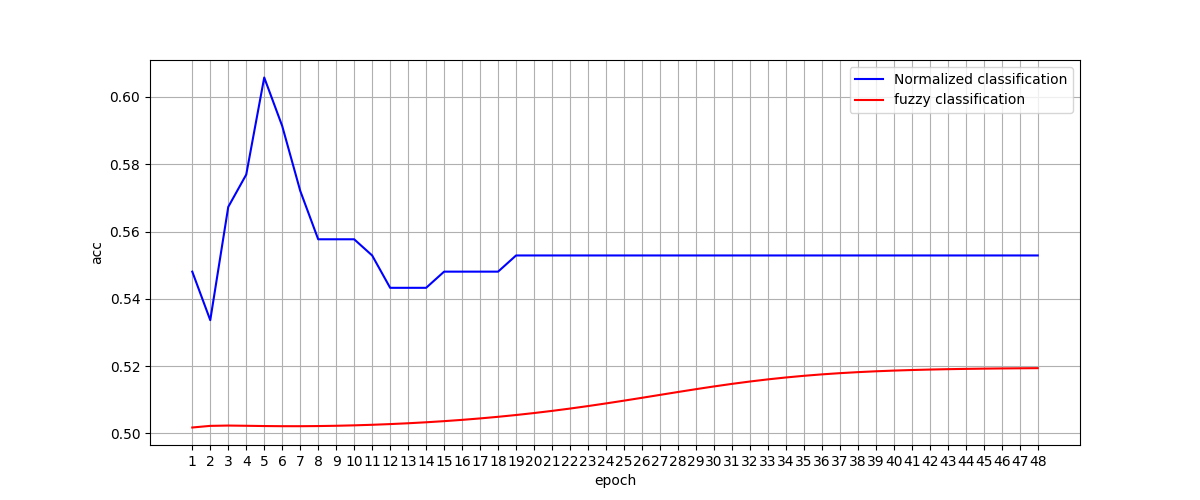


图4 FCM实验结果图

可见模糊聚类方法与硬聚类算法相比有比较好的鲁棒性与适用性，可以选取不同的软聚类评价指标，提高聚类的正确率，更加灵活通用。

四、源程序代码

*# kmeans*

import **pandas** **as** **pd**

import **numpy** **as** **np**

import **matplotlib.pyplot** **as** **plt**

import **copy**

**if** \_\_name\_\_ == "\_\_main\_\_":

df = pd.read\_csv("sonar.all-data", header=**None**)

df.replace('R', 0, inplace=**True**)

df.replace('M', 1, inplace=**True**)

data = np.array(df.values, dtype='float')

x = data[:, :-1]

y = data[:, -1]

K = 2

z = [] *# K个聚类中心*

idx = np.arange(0, len(x))

np.random.shuffle(idx)

**for** i **in** range(K):

z.append(x[idx[i], :])

epoch = 0

ACC = list()

**while** **True**:

pre = copy.copy(z)

clusters = [[] **for** i **in** range(K)] *# K个聚类蔟中包含的点*

**for** i **in** range(len(x)):

dis = [] *# 该点到K个中心的距离表*

**for** j **in** range(K):

dis.append(np.linalg.norm(x[i] - z[j]))

nearest = dis.index(min(dis)) *# 找出距离其最近的聚类中心*

clusters[nearest].append(i)

flag = **True**

**for** i **in** range(K):

cluster\_mean = np.zeros(len(z[i]))

**for** j **in** range(len(clusters[i])):

cluster\_mean += x[clusters[i][j]] / len(clusters[i])

z[i] = cluster\_mean

**if** (z[i] != pre[i]).all():

flag = **False**

epoch += 1

*# 计算分类准确率*

acc = 0

**for** i **in** range(K):

label\_list = []

**for** j **in** range(len(clusters[i])):

label\_list.append(y[clusters[i][j]])

true\_label = []

**for** j **in** range(K):

true\_label.append(label\_list.count(j))

acc += max(true\_label) *# 选取数量最大的标签作为其标签*

acc /= len(y)

ACC.append(acc)

**if** flag:

print('已找到聚类结果')

**break**

px = np.arange(1, epoch + 1).astype(dtype='str')

plt.plot(px, ACC, c='r')

plt.grid()

plt.xlabel('epoch')

plt.ylabel('acc')

labels = ['kmeans']

plt.legend(labels, loc='best', fancybox=**True**)

*# plt.savefig('KMEANS-SONAR.png')*

plt.show()

*# FCM*

import **pandas** **as** **pd**

import **numpy** **as** **np**

import **matplotlib.pyplot** **as** **plt**

import **copy**

**if** \_\_name\_\_ == "\_\_main\_\_":

df = pd.read\_csv("sonar.all-data", header=**None**)

df.replace('R', 0, inplace=**True**)

df.replace('M', 1, inplace=**True**)

data = np.array(df.values, dtype='float')

x = data[:, :-1]

y = np.array(data[:, -1], dtype='int32')

K = 2

z = [0, 0] *# K个聚类中心*

U = np.random.rand(len(x), K)

**for** i **in** range(len(x)):

U[i] = U[i] / sum(U[i])

J = 0

a = 2 *# 柔性参数*

epoch = 0

ACC2 = list()

ACC1 = list()

**while** **True**:

z\_old = copy.copy(z)

U\_old = copy.copy(U)

J\_old = J

*# 计算新聚类中心*

**for** j **in** range(K):

sum\_ux = 0

sum\_u = 0

**for** i **in** range(len(x)):

sum\_ux += (U[i][j] \*\* a) \* x[i]

sum\_u += U[i][j] \*\* a

z[j] = sum\_ux / sum\_u

epoch += 1

*# 计算代价函数*

J = 0

**for** j **in** range(K):

**for** i **in** range(len(x)):

J += (U[i][j] \*\* a) \* (np.linalg.norm(z[j] - x[i]) \*\* 2)

**if** abs(J - J\_old) < 0.0001:

**break**

*# 计算新矩阵U*

**for** i **in** range(len(x)):

**for** j **in** range(K):

sum\_ud = 0

**for** k **in** range(K):

sum\_ud += ((np.linalg.norm(z[j] - x[i])) / (np.linalg.norm(z[k] - x[i])))\

\*\* (2 / (a - 1))

U[i][j] = 1 / sum\_ud

*# 计算第几蔟的实际标签是什么*

label\_order = []

**for** i **in** range(K):

K\_list = [0] \* K

**for** j **in** range(len(x)):

**if** np.argmax(U[j]) == i:

K\_list[y[j]] += 1

label\_order.append(K\_list.index(max(K\_list)))

**assert** len(set(label\_order)) == K, '出现了两类相同蔟！'

un\_label\_order = [0] \* K

**for** i **in** range(K):

un\_label\_order[label\_order[i]] = i

acc1 = 0

**for** i **in** range(len(x)):

**if** U[i][un\_label\_order[y[i]]] == max(U[i]):

acc1 += 1

acc1 /= len(x)

ACC1.append(acc1) *# 归1分类*

acc2 = 0

**for** i **in** range(len(x)):

acc2 += U[i][un\_label\_order[y[i]]]

acc2 /= len(x)

ACC2.append(acc2) *# 模糊分类准确率*

plt.figure(figsize=(12, 5))

px = np.arange(1, epoch).astype(dtype='str')

plt.plot(px, ACC1, c='b')

plt.plot(px, ACC2, c='r')

plt.grid()

plt.xlabel('epoch')

plt.ylabel('acc')

labels = ['Normalized classification', 'fuzzy classification']

plt.legend(labels, loc='best', fancybox=**True**)

*# plt.savefig('FCM-SONAR.png')*

plt.show()