哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称： 机器学习

课程类型： 专业选修

实验题目： k-means 聚类

学号：120L030501
姓名： 张明远

一、实验目的

目标：

- 理解 k-means 聚类过程
- 理解混合高斯模型(GMM)用 EM 估计参数的实现过程
- 掌握 k-means 和混合高斯模型的联系
- 学会 EM 估计参数的方法和代码实现
- 学会用 GMM 解决实际问题

二、实验要求及实验环境

要求：

目标：实现一个 k-means 算法和混合高斯模型，并且用 EM 算法估计模型中的参数。

测试：

用高斯分布产生 k 个高斯分布的数据（不同均值和方差）（其中参数自己设定）。

（1）用 k-means 聚类，测试效果；

（2）用混合高斯模型和你实现的 EM 算法估计参数，看看每次迭代后似然值变化情况，考察 EM 算法是否可以获得正确的结果（与你设定的结果比较）。

应用：可以 UCI 上找一个简单问题数据，用你实现的 GMM 进行聚类。

环境：

- Win10，x86-64
- Pycharm 2019.1
- python 3.7.1
- 使用 python 库：numpy（计算）, matplotlib.pyplot(用于可视化)

三、设计思想（本程序中的用到的主要算法及数据结构）

3.1 生成数据

本实验要求对多分类数据进行 k-means 聚类。利用多维高斯分布函数来生成数据，为便于画图展示，主要使用二维数据。首先生成朴素贝叶斯数据，在这里选取中心点为（-10，-10）（10，10）（-10，10）（10，-10）四个点。

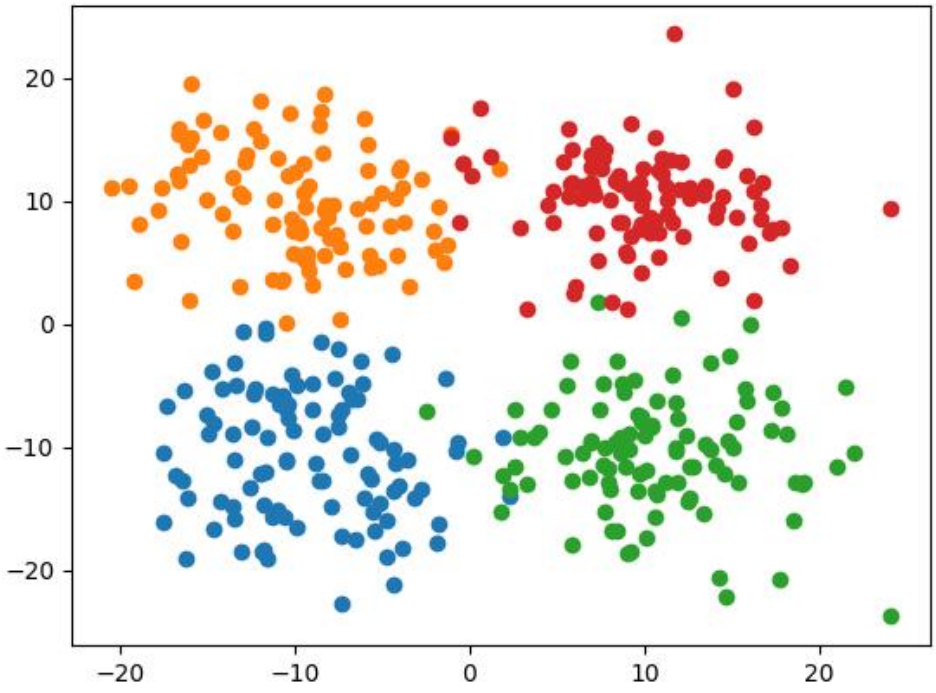
若满足朴素贝叶斯，则认为 X 的各维度数据关于 Y 条件独立，则协方差矩阵为

Cov = [[cov11,cov12][cov21,cov22]]

其中 cov12 = cov21 = 0

cov11 就是 X1 的方差， cov22 就是 X2 的方差，

根据协方差矩阵添加高斯噪声。令方差为 20 生成样本点生成数据如下，数据保存在 data.csv：



代码核心部分如下：

```
    def generatePoints(center, rand_normal_x, rand_normal_y, nums,
naive_rate):
    x = center[0]
    y = center[1]
    cov = [[rand_normal_x, naive_rate], [naive_rate, rand_normal_y]]
    xs = np.random.multivariate_normal((x, y), cov, nums)
    return xs
```

3.2 k-means 聚类

k-means 聚类就是根据某种度量方式(常用欧氏距离，如欧氏距离越小，相关性越大)，将相关性较大的一些样本点聚集在一起，一共聚成 k 个堆，每一个堆

我们称为一“类”。k-means 的过程为：先在样本点中选取 k 个点作为暂时的聚类中心，然后依次计算每一个样本点与这 k 个点的距离，将每一个与距离这个点最近的中心点聚在一起，这样形成 k 个类“堆”，求每一个类的期望，将求得的期望作为这个类的新的中心点。一直不停地将所有样本点分为 k 类，直至中心点不再改变停止。

伪代码：

```
获取数据 n 个 m 维的数据
随机生成 K 个 m 维的点
while(t)
    for(int i=0;i < n;i++)
        for(int j=0;j < k;j++)
            计算点 i 到类 j 的距离
    for(int i=0;i < k;i++)
        1. 找出所有属于自己这一类的所有数据点
        2. 把自己的坐标修改为这些数据点的中心点坐标
end
```

3.3k-means 聚类实现

3.3.1 读取数据

使用 np.loadtxt 从‘data.csv’中将数据读取出来

python 实现

```
def get_data_set():
    matrix = np.loadtxt(data_file)
    return matrix
```

3.3.2k-means 聚类

随机选取 k 个点作为中心点

利用 random 随机生成正整数，结合 set()的互异性得到 k 个互不相等的正整数，将这 k 个样本点作为初始的 k 个类的中心。

python 实现

```
def get_k_center():
    xs = set()
    while len(xs) < k_num:
        xs.add((2*random.randint(-15, 15),2*random.randint(-15, 15)))
    k_num_list = []
```

```
    for x in xs:
        k_num_list.append(x)
    return k_num_list
```

计算所有样本点到中心的距离，并对样本点进行重新标记

python 实现

```
for i in range(times):
    for key, value in data_dict.items():
        for k_seg in range(k_num):
            data_dict[key][k_seg] = get_length(key, k_num_list[k_seg])
```

根据更新后的样本点重新计算中心点的位置

python 实现

```
for k_seg in range(k_num):
    k_points_dict[k_seg] = []
    for key, value in data_dict.items():
        minflag = True
        min_value = value[k_seg]
        for i in range(k_num):
            if min_value > value[i]:
                minflag = False
        if minflag:
            k_points_dict[k_seg].append(key)
    k_num_list[k_seg] = mean(k_points_dict[k_seg])
```

画图展示最终结果

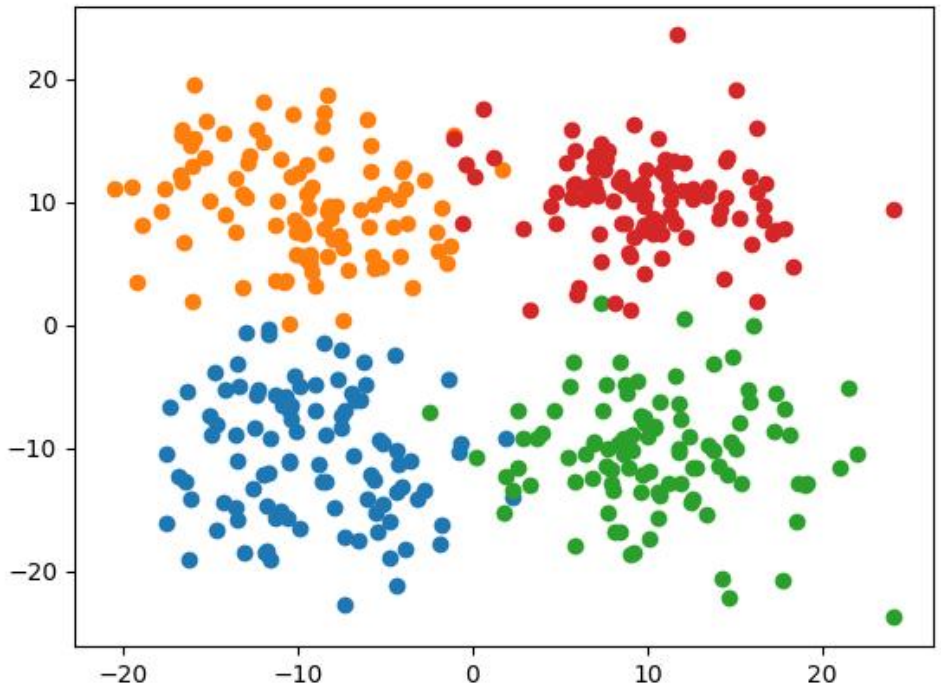
python 实现

```
for k_seg in range(k_num):
    x = [k_points_dict[k_seg][i][0] for i in
range(len(k_points_dict[k_seg]))]
    y = [k_points_dict[k_seg][i][1] for i in
range(len(k_points_dict[k_seg]))]
    plt.scatter(x, y)
plt.show()
```

四、实验结果与分析

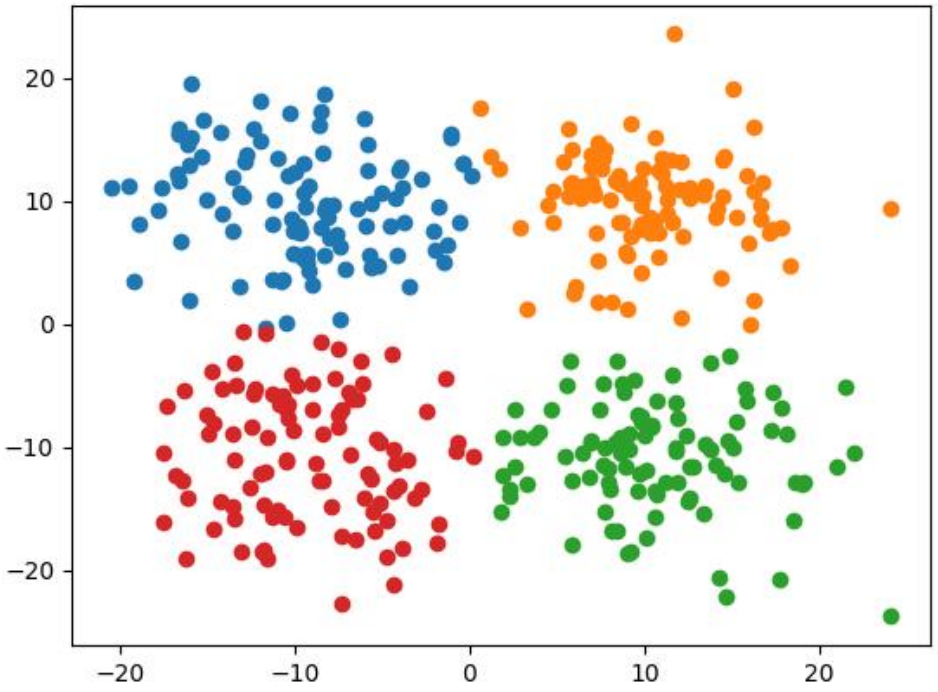
4.1 使用数据说明

第一类：中心点（10，10），噪声均值皆为 0，标准差 cov11=cov22=20 的 100 个点
第二类：中心点（10，-10），噪声均值皆为 0，标准差 cov11=cov22=20 的 100 个点
第一类：中心点（-10，10），噪声均值皆为 0，标准差 cov11=cov22=20 的 100 个点
第二类：中心点（-10，-10），噪声均值皆为 0，标准差 cov11=cov22=20 的 100 个点



4.2k-means 聚类分析

用 k-means 聚类，发现效果不错

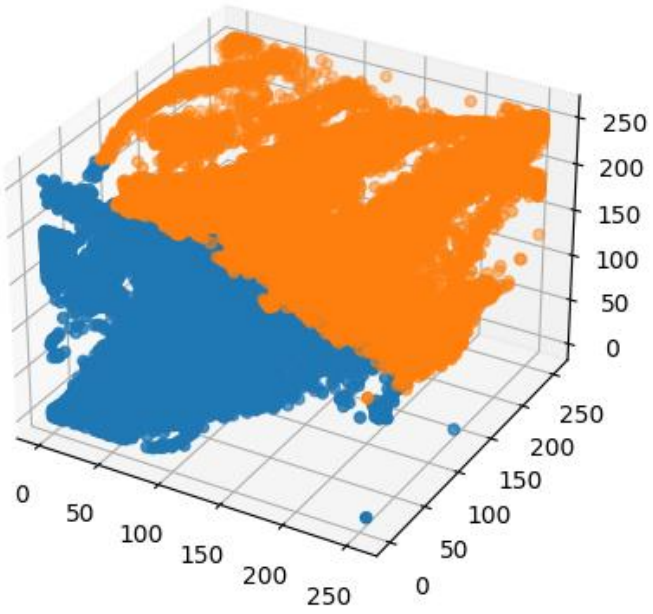


4.3UCI 数据集测试

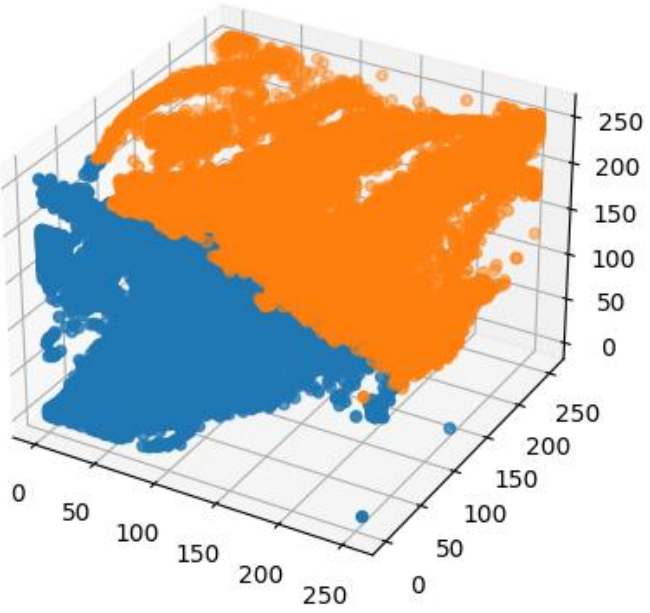
UCI 数据集是一个常用的机器学习标准测试数据集，是加州大学欧文分校(University of CalifornialrvineJ提出的用于机器学习的数据库。机器学习算法的测试大多采用的便是 UCI 数据集了，其重要之处在于“标准”二字，新编的机器学习程序可以采用 UCI 数据集进行测试,类似的机器学习算法也可以一较高下。

本次实验选用 UCI 中的 Skin Segmentation Data Set，它是一个二分类数据集，且 x 只含有三个维度，适合可视化处理。数据集中一共有 245057 条数据。

数据如图所示：



改造一下先前用于平面数据的程序即可进行聚类。
得到结果如下：



可见 k-means 很好地分类了数据集。

五、结论

由于 k-means 选取的初始点是随机选取的，大多数情况下表现都还不错，但是有时候选择的点过于奇怪会导致 k-means 分类效果不太好 ，甚至出现和初始的分类完全不像的分类图。K-Means 的分类结果受初始中心点的影响，我们可以通过一些求均值或者别的的方式寻找初始点来尽量减少这种影响 K-Means 其实就是一种特殊的高斯混合模型。

六、参考文献

1）Pattern Recognition and Machine Learning.
2）Gradient descent wiki
3）Conjugate gradient method wiki
4）Shewchuk J R. An introduction to the conjugate gradient method without the agonizing pain[J]. 1994.

七、附录：源代码（带注释）

```
   #
import math
import random

import numpy as np
from matplotlib import pyplot as plt

from config import *


# 读取数据
def get_data_set():
    matrix = np.loadtxt(data_file)
    return matrix


# 获得数据标记
def get_data_dict(data_set):
    data_dict = {}
    for data in data_set:
        data_dict[(data[0], data[1])] = {}
    return data_dict

#随机生成 k 个中心点
def get_k_center():
    xs = set()
```

```
    while len(xs) < k_num:
        xs.add((2*random.randint(-15, 15),2*random.randint(-15, 15)))
    k_num_list = []
    for x in xs:
        k_num_list.append(x)
    return k_num_list


# 计算距离
def get_length(point1, point2):
    return    math.sqrt(math.pow(point1[0]    -    point2[0],2)    +
math.pow(point1[1] - point2[1],2))


# 计算中心点
def mean(points_list):
    x_mean = 0
    y_mean = 0
    sum = 0
    for point in points_list:
        x_mean+=point[0]
        y_mean+=point[1]
        sum+=1
    if sum == 0:
        return (0,0)
    return (x_mean/sum,y_mean/sum)


# 主程序
data_set = get_data_set()
data_dict = get_data_dict(data_set)
k_num_list = get_k_center()
k_points_dict = {}
# 聚类
for i in range(times):
    for key, value in data_dict.items():
        for k_seg in range(k_num):
            data_dict[key][k_seg] = get_length(key, k_num_list[k_seg])
    for k_seg in range(k_num):
        k_points_dict[k_seg] = []
        for key, value in data_dict.items():
            minflag = True
            min_value = value[k_seg]
            for i in range(k_num):
```

```
                if min_value > value[i]:
                    minflag = False
            if minflag:
                k_points_dict[k_seg].append(key)
        k_num_list[k_seg] = mean(k_points_dict[k_seg])
# 画图
for k_seg in range(k_num):
    x = [k_points_dict[k_seg][i][0] for i in
range(len(k_points_dict[k_seg]))]
    y = [k_points_dict[k_seg][i][1] for i in
range(len(k_points_dict[k_seg]))]
    plt.scatter(x, y)
plt.show()
```