Python编程与人工智能实践

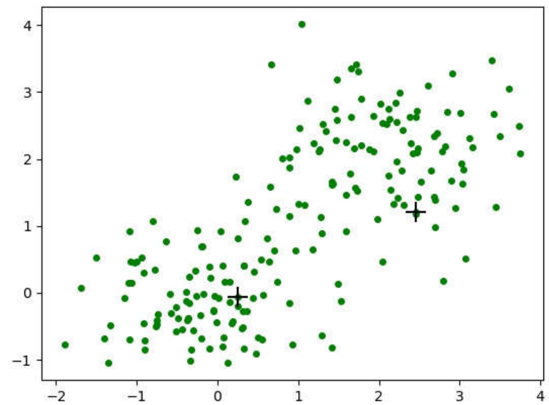
算法篇：无监督聚类
k-means 与 GMM （高斯混合模型）

于泓
鲁东大学
信息与电气工程学院
2021.3.20

K-means （k均值聚类）

- 与有监督学习相比，无监督学习的样本没有任何标记。无监督学习的算法需要自动找到这些没有标记的数据里面的数据结构和特征。
- **聚类**：把数据集分成一个个的簇（ cluster ）

自动把数据分堆?



K-mean 算法

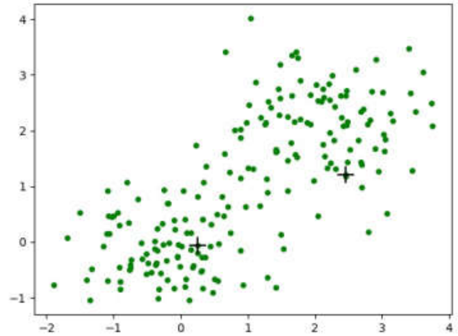
初始化
K个 聚类中心
C_k

按照距离最近原则
将数据分簇

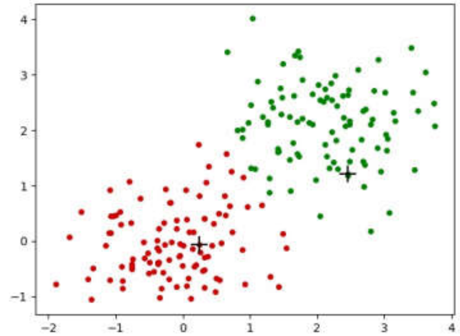
对每组数据取均值
计算新的聚类中心

$$C_k = \frac{1}{N_k} \sum_{x_i \in k\text{族}} x_i$$

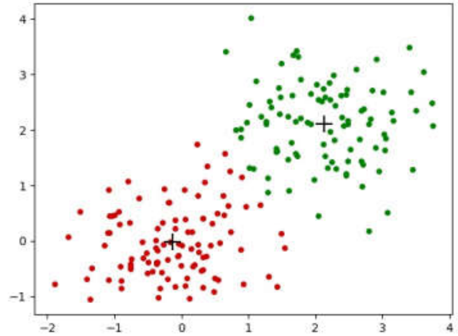
迭代次数完成
聚类结果不变



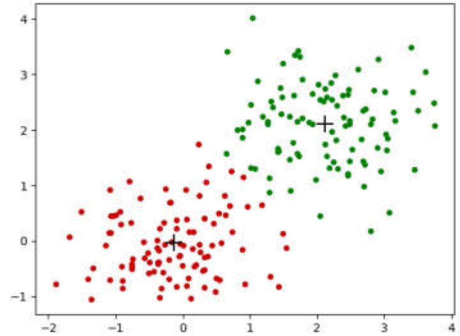
初始化



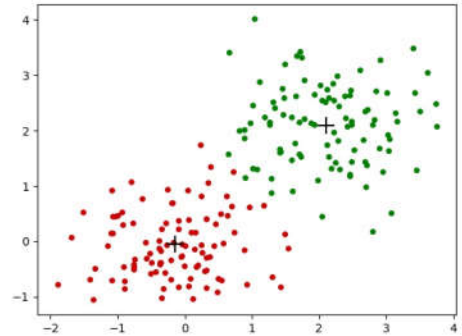
初次聚类



修正中心



再次聚类



迭代3次后最终结果

```python
# 构造聚类中心
# dataset [N,D]
# K 聚类中心的数目 [K,D]
def creat_centers(dataset,K):
    val_max = np.max(dataset,axis=0)
    val_min = np.min(dataset,axis=0)
    centers = np.linspace(val_min,val_max,num=K+2)
    return centers[1:-1,:]
```

在X的值域内均匀取值

```python
def creat_centers(dataset,K):
    N,D = np.shape(dataset)
    index = np.random.permutation(N)
    centers = dataset[index[:2],:]
    return centers
```

随机取值

随机取K个点

```python
# keams 绘图
# dataset (N,D)
# lab (N,)
# dic_colors K 种颜色
# centers (K,D)
def draw_kmeans(dataset,lab,centers,dic_colors=None,name="0.jpg"):
    plt.cla()

    vals_lab = set(lab.tolist())

    for i,val in enumerate(vals_lab):
        index = np.where(lab==val)[0]
        sub_dataset = dataset[index,:]
        plt.scatter(sub_dataset[:,0],sub_dataset[:,1],s=16., color=dic_colors[i])

    for i in range(np.shape(centers)[0]):
        plt.scatter(centers[i,0],centers[i,1],color="k",marker="+",s = 200.)

    plt.savefig(name)
```

绘图函数
显示中间结果

```python
def run_kmeans(dataset,K, m = 20,dic_colors=None, b_draw=False):
    N,D = np.shape(dataset)
    # print(N,D)
    # 确定初始化聚类中心
    centers = creat_centers(dataset,K)

    lab = np.zeros(N)
    if b_draw:
        draw_kmeans(dataset,lab,centers,dic_colors,name="int.jpg")


    # 进行m轮迭代
    labs = np.zeros(N) # 初始聚类结果
    for it in range(m):
        # 计算每个点距离中心的距离
        distance = np.zeros([N,K])
        for k in range(K):
            center = centers[k,:]

            # 计算欧式距离
            diff = np.tile(center, (N, 1)) - dataset
            sqrDiff = diff ** 2
            sqrDiffSum = sqrDiff.sum(axis=1)
            distance[:,k] = sqrDiffSum

        # 距离排序，进行聚类
        labs_new = np.argmin(distance,axis=1)
        error = np.sum(np.min(distance,axis=1))/N
        print("第 %d 次聚类 距离误差 %.2f"%(it,error))
```

按照聚类中心进行聚类

对聚类中心进行更新

```python
        # 绘图
        if b_draw:
            draw_kmeans(dataset,labs_new,centers,
                        dic_colors,name=str(it)+"_oldcenter.jpg")

        # 计算新的聚类中心
        for k in range(K):
            index = np.where(labs_new==k)[0]
            centers[k,:] = np.mean(dataset[index,:],axis=0)

        # 绘图
        if b_draw:
            draw_kmeans(dataset,labs_new,centers,
                        dic_colors,name=str(it)+"_newcenter.jpg")

        # 如果聚类结果和上次相同，退出
        if np.sum(labs_new-labs)==0:
            return labs_new
        else:
            labs = labs_new

    return labs
```

```python
if __name__=="__main__":
    a = np.random.multivariate_normal([2,2], [[.5,0],[0,.5]], 100)
    b = np.random.multivariate_normal([0,0], [[0.5,0],[0,0.5]], 100)
    dataset = np.r_[a,b]
    lab_ture = np.r_[np.zeros(100),np.ones(100)].astype(int)
    dic_colors={0:(0.,0.5,0.),1:(0.8,0,0)}
    labs = run_kmeans(dataset,K=2, m = 20,dic_colors=dic_colors,b_draw=True)
```

```
yuhong@admin2:/home/sdo/machinelearning/k-mean$ python test_kmeans.py
200 2
第 0 次聚类 距离误差 5.47
第 1 次聚类 距离误差 1.42
第 2 次聚类 距离误差 0.97
第 3 次聚类 距离误差 0.97
```

初始化
K个 聚类中心
C_k

按照距离最近原则
将数据分簇

对每组数据取均值
计算新的聚类中心

$$C_k = \frac{1}{N_k} \sum_{x_i \in k\text{族}} x_i$$

迭代次数完成
聚类结果不变

聚类本质：寻找聚类中心，根据到中心距离进行分簇

参数

函数的形式

$$P(Y=yes|x_1,x_2,x_3,x_4) = \frac{P(x_1,x_2,x_3,x_4|Y=yes)*P(Y=yes)}{P(x_1,x_2,x_3,x_4)}$$

谁大？

$$P(Y=no|x_1,x_2,x_3,x_4) = \frac{P(x_1,x_2,x_3,x_4|Y=no)*P(Y=no)}{P(x_1,x_2,x_3,x_4)}$$

利用函数

$$f(x;C_k) = e^{-(x-C_k)(x-C_k)^T}$$

拟合第C簇数据的分布

高斯分布

高斯分布（正态分布）是一个常见的连续概率分布，在统计领域中有非常重要的作用

（1维）

$$N(x;\mu,\delta)=\frac{1}{\delta\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\delta^2}}$$

（多维）

$$N(\mathrm{x};\mathrm{m},\Sigma)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(\mathrm{x}-\mathrm{m})\Sigma^{-1}(\mathrm{x}-\mathrm{m})^T}$$

x: [1, D]

m: [1, D]

Σ: [D, D]

如何利用高斯函数来拟合一组数据的分布?

如何求解高斯分布的参数?

假设有一组数据 x_i，令下式最大：

$$\sum_{i=1}^{N}\log\left(N(\mathrm{x}_i;\mathrm{m},\Sigma)\right)$$

求最大似然

上式的解为：

$$\mathrm{m}=\frac{1}{N}\sum_{i=1}^{N}\mathrm{x}_i$$

$$\Sigma=\frac{1}{\mathrm{N}}\sum_{i=1}^{N}(\mathrm{x}_i-\mathrm{m})(\mathrm{x}_i-\mathrm{m})^T$$

初始化
K个 聚类中心
C_k

按照距离最近原则
将数据分簇

对每组数据取均值
计算新的聚类中心

$$C_k = \frac{1}{N_k} \sum_{x_i \in k\text{族}} x_i$$

迭代次数完成
聚类结果不变

利用**加权**的高斯函数拟合**第k簇**数据的分布

$$w_k N(x; m_k, \Sigma_k) \quad \sum_{k=1}^{K} w_k = 1$$

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

$$r_{ik} = \frac{w_k N(x_i; m_k, \Sigma_k)}{\sum_{k=1}^{K} w_k N(x_i; m_k, \Sigma_k)}$$

$$m_k = \frac{\sum_{i=1}^{N} r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}}, \quad \Sigma_k = \frac{\sum_{i=1}^{N} r_{ik} (x_i - m_k)(x_i - m_k)^T}{\sum_{i=1}^{N} r_{ik}}$$

$$w_k = \frac{\sum_{i=1}^{N} r_{ik}}{N},$$

初始化
K个 分布函数
的参数
w_k、m_k、Σ_k

按照不同分布函数
计算的概率
将数据“**分簇**”

根据最大似然
原则对参数更新

迭代次数
完成

```python
import numpy as np
import matplotlib.pyplot as plt

# GMM 参数初始化
# dataset: [N,D] 训练数据
# K : 高斯成分的个数
def inti_GMM(dataset,K):
    N,D = np.shape(dataset)
    val_max = np.max(dataset,axis=0)
    val_min = np.min(dataset,axis=0)
    centers = np.linspace(val_min,val_max,num=K+2)

    mus = centers[1:-1,:]
    sigmas = np.array([0.5*np.eye(D) for i in range(K)])
    ws = 1.0/K * np.ones(K)

    return mus,sigmas,ws
```

```python
# 计算一个高斯的pdf
# x: 数据 [N,D]
# sigma 方差 [D,D]
# mu 均值 [1,D]
def getPdf(x,mu,sigma,eps = 1e-12):
    N,D = np.shape(x)

    if D==1:
        sigma = sigma+eps
        A = 1.0 / (sigma)
        det = np.fabs(sigma[0])
    else:
        sigma = sigma + eps*np.eye(D)
        A = np.linalg.inv(sigma)
        det = np.fabs(np.linalg.det(sigma))

    # 计算系数
    factor = (2.0 * np.pi)**(D / 2.0) * (det)**(0.5)

    # 计算 pdf
    dx = x - mu
    pdf =[ (np.exp(-0.5*np.dot(np.dot(dx[i],A),dx[i]))+eps)/ factor for i in range(N)]
    return pdf
```

$$N(\mathrm{x};\mathrm{m},\Sigma)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(\mathrm{x}-\mathrm{m})\Sigma^{-1}(\mathrm{x}-\mathrm{m})^{T}}$$

```python
def train_GMM_step(dataset,mus,sigmas,ws):
    N,D = np.shape(dataset)
    K,D = np.shape(mus)
    # 计算样本在每个成分上的pdf
    pdfs = np.zeros([N,K])
    for k in range(K):
        pdfs[:,k] = getPdf(dataset,mus[k],sigmas[k])

    # 获取r
    r = pdfs*np.tile(ws, (N, 1))
    r_sum = np.tile(np.sum(r,axis=1,keepdims=True),(1,K))
    r = r/r_sum

    # 进行参数的更新
    for k in range(K):
        r_k = r[:,k]
        N_k = np.sum(r_k)
        r_k = r_k[:,np.newaxis] #[N,1]

        # 更新mu
        mu = np.sum(dataset*r_k,axis=0)/ N_k #[D,1]

        # 更新sigma
        dx = dataset - mu
        sigma = np.zeros([D,D])
        for i in range(N):
            sigma = sigma + r_k[i,0]*np.outer(dx[i],dx[i])
        sigma = sigma/N_k

        # 更新w
        w = N_k/N
        mus[k]= mu
        sigmas[k]=sigma
        ws[k]=w
    return mus,sigmas,ws
```

$$r_{ik} = \frac{w_k N(x_i; m_k, \Sigma_k)}{\sum_{k=1}^{K} w_k N(x_i; m_k, \Sigma_k)}$$

$$m_k = \frac{\sum_{i=1}^{N} r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}},$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} r_{ik}(x_i - m_k)(x_i - m_k)^T}{\sum_{i=1}^{N} r_{ik}}$$

$$w_k = \frac{\sum_{i=1}^{N} r_{ik}}{N},$$

GMM 训练

```
def train_GMM(dataset,K=2,m=10):
    mus,sigmas,ws = inti_GMM(dataset,K)

    for i in range(m):
        print("Step ",i)
        mus,sigms,ws,r =train_GMM_step(dataset,mus,sigmas,ws)
    return mus,sigms,ws
```

计算输入数据在每个高斯成分上的似然值

```
def getlogPdfFromeGMM(datas,mus,sigmas,ws):
    N,D = np.shape(datas)
    K,D = np.shape(mus)

    weightedlogPdf = np.zeros([N,K])

    for k in range(K):
        temp = getPdf(datas,mus[k],sigmas[k],eps = 1e-12)
        weightedlogPdf[:,k] = np.log(temp) + np.log(ws[k])

    return weightedlogPdf, np.sum(weightedlogPdf,axis=1)
```

利用GMM进行聚类（比较似然值）

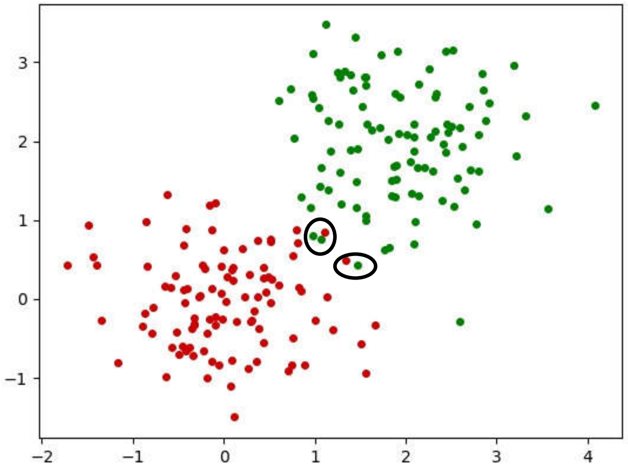
```
def clusterByGMM(datas,mus,sigmas,ws):
    weightedlogPdf,_ = getlogPdfFromeGMM(datas,mus,sigmas,ws)
    labs = np.argmax(weightedlogPdf,axis=1)
    return labs
```

```python
def draw_cluster(dataset,lab,dic_colors,name="0.jpg"):
    plt.cla()
    vals_lab = set(lab.tolist())

    for i,val in enumerate(vals_lab):
        index = np.where(lab==val)[0]
        sub_dataset = dataset[index,:]
        plt.scatter(sub_dataset[:,0],sub_dataset[:,1],s=16., color=dic_colors[i])

    plt.savefig(name)
```

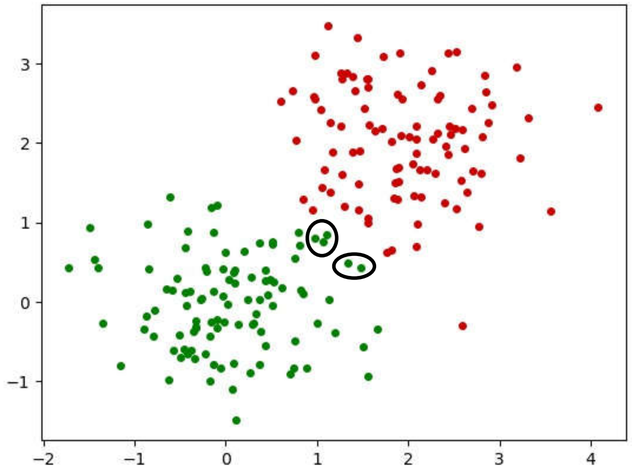
测试程序
比较 GMM
与k-means

```
yuhong@admin2:/home/sdo/machinelearning/GMM$ python (
[[ 0.01103085 -0.0445471 ]
 [ 2.19003523  1.86395921]]
[[[0.52244123 0.11045438]
  [0.11045438 0.46578895]]

 [[0.43072345 0.00740219]
  [0.00740219 0.43643166]]]
[0.50825447 0.49174553]
第 0 次聚类 距离误差 0.97
第 1 次聚类 距离误差 0.89
第 2 次聚类 距离误差 0.89
```

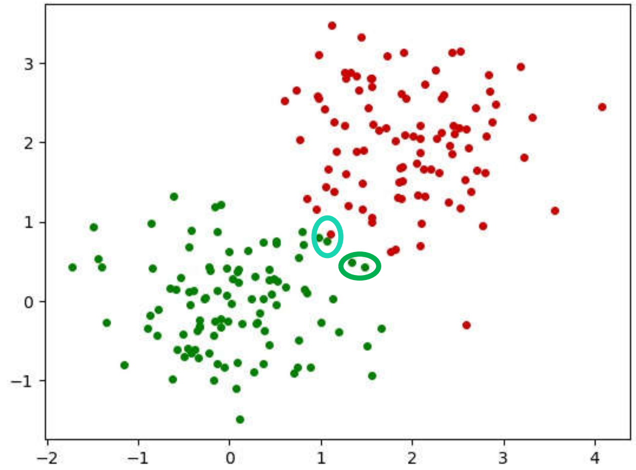
```python
if __name__=="__main__":
    '''
        聚类测试 1
    '''
    dic_colors={0:(0.,0.5,0.),1:(0.8,0,0)}
    a = np.random.multivariate_normal([2,2], [[.5,0],[0,.5]], 100)
    b = np.random.multivariate_normal([0,0], [[0.5,0],[0,0.5]], 100)
    dataset = np.r_[a,b]
    lab_ture = np.r_[np.zeros(100),np.ones(100)].astype(int)

    # 训练GMM
    mus,sigmas,ws = train_GMM(dataset,K=2,m=10)
    print(mus)
    print(sigmas)
    print(ws)
    # 进行聚类
    labs_GMM = clusterByGMM(dataset,mus,sigmas,ws)
    # k-menas 比较
    labs_kmeans = run_kmeans(dataset,K=2, m = 20)
    # 画结果
    draw_cluster(dataset,lab_ture,dic_colors,name="c_ture1.jpg")
    draw_cluster(dataset,labs_GMM,dic_colors,name="c_GMM1.jpg")
    draw_cluster(dataset,labs_kmeans,dic_colors,name="c_kmeans1.jpg")
```



真实标签



K-means 聚类



GMM-聚类

$$f(\mathrm{x};C_k)=e^{-(\mathrm{x}\text{-}\mathrm{C}_\mathrm{k})(x-C_k)^T}$$

$$N(\mathrm{x};\mathrm{m},\Sigma)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(\mathrm{x}-\mathrm{m})\Sigma^{-1}(\mathrm{x}-\mathrm{m})^T}$$

都默认特征维度之间弱相关

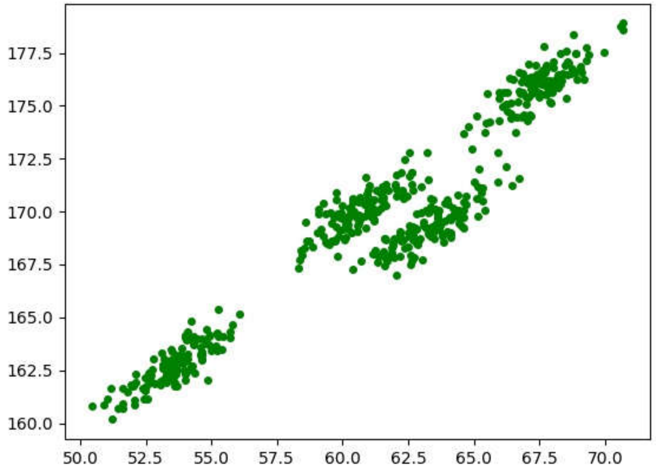
```
a = np.random.multivariate_normal([2,2], [[.5,0],[0,.5]], 100)
b = np.random.multivariate_normal([0,0], [[0.5,0],[0,0.5]], 100)
dataset = np.r_[a,b]
```

```python
'''
    聚类测试 2
'''
with open('Clustering_gmm.csv','r') as f:
    lines = f.read().splitlines()[1:]
lines = [ line.split(",") for line in lines]
dataset = np.array(lines).astype(np.float)
lab_ture = np.ones(np.shape(dataset)[0])
dic_colors={0:(0.,0.5,0.),1:(0.8,0,0),2:(0.5,0.5,0),3:(0,0.5,0.5)}

# 训练GMM
mus,sigmas,ws = train_GMM(dataset,K=4,m=100)

# 进行聚类
labs_GMM = clusterByGMM(dataset,mus,sigmas,ws)

# k-menas 比较
labs_kmeans = run_kmeans(dataset,dic_colors,K=4, m = 20)

# 画结果
draw_cluster(dataset,lab_ture,dic_colors,name="c_ture2.jpg")
draw_cluster(dataset,labs_GMM,dic_colors,name="c_GMM2.jpg")
draw_cluster(dataset,labs_kmeans,dic_colors,name="c_kmeans2.jpg")
```

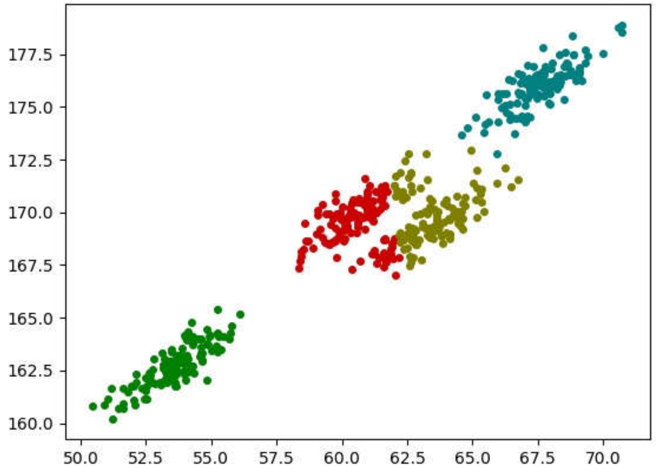
Clustering_gmm.csv | test_knn.py | test_kmeans.py | sompak_gui.

```
Weight,Height
67.06292382432797,176.08635470037433
68.80409404055906,178.38866853397775
60.93086316752808,170.28449576512674
59.73384301263917,168.69199180312273
65.43123003070372,173.7636790317747
61.577160332549624,168.0917512363048
63.34186626427547,170.64251602924492
61.04164336121978,170.09668165680276
62.63362334719166,171.8629715737515
53.4078596166848,162.756843114429
62.93820030525223,168.71007855359954
68.55485709616576,176.4737467967919
65.16304259531378,171.17658203536783
53.44601710597588,162.91516701320765
60.65937348586088,170.6476590778724
59.1765543543734,169.19080975295822
67.16384091157683,176.33706853049466
60.62703390449537,169.8481191146251
53.96476394988538,162.49055420493335
60.35999202109157,169.89078752485602
59.856871023856,168.85468537336587
```

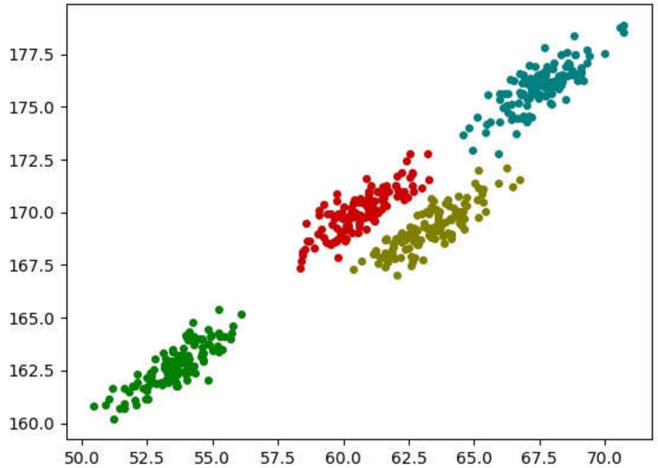
```
第 0 次聚类 距离误差 4.70
第 1 次聚类 距离误差 2.48
第 2 次聚类 距离误差 2.34
第 3 次聚类 距离误差 2.28
第 4 次聚类 距离误差 2.27
第 5 次聚类 距离误差 2.27
第 6 次聚类 距离误差 2.27
```



原始数据



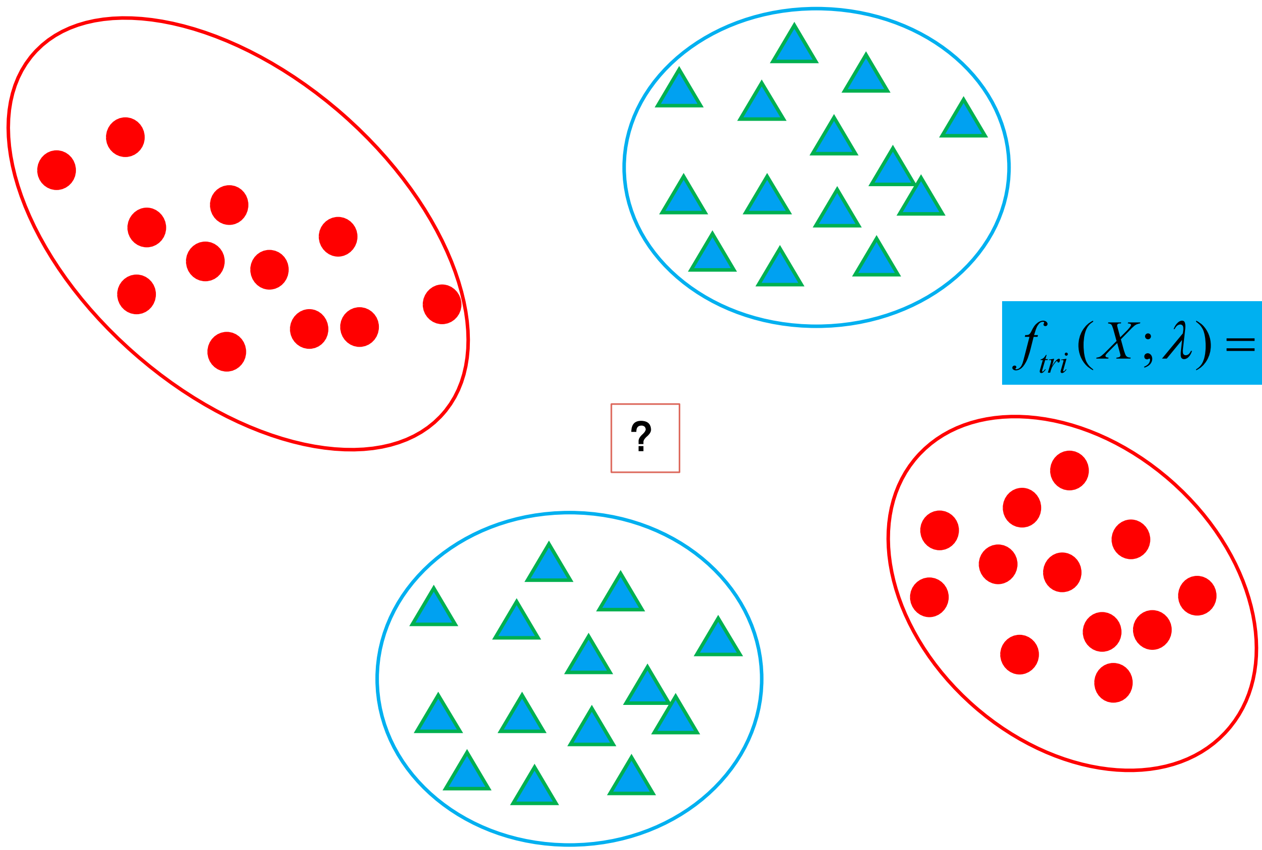
K-means聚类



GMM聚类

$$f_{cir}(X;\lambda)=w_1 N_{cir1}(X;\lambda_{cir1})+w_2 N_{cir2}(X;\lambda_{cir1})$$

GMM 用于分类任务



$$f_{tri}(X;\lambda)=w_1 N_{tri1}(X;\lambda_{tri1})+w_2 N_{tri2}(X;\lambda_{tri1})$$

鸢尾花数据集（Iris）

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
```

数据集分割

```
'''
    分类测试
'''
file_data = 'iris.data'
# 数据读取
data = np.loadtxt(file_data,dtype = np.float, delimiter = ',',usecols=(0,1,2,3))
lab = np.loadtxt(file_data,dtype = str, delimiter = ',',usecols=(4))

# 分为训练集和测试集
N = 150
N_train = 100
N_test = 50
perm = np.random.permutation(N)

index_train = perm[:N_train]
index_test = perm[N_train:]

data_train = data[index_train,:]
lab_train = lab[index_train]

data_test = data[index_test,:]
lab_test = lab[index_test]
```

为每一类花训练一个GMM

```python
# 获取 训练标签类型
unique_labs = np.unique(lab_train).tolist()
models ={}

# 进行GMM 训练，为每类数据训练一个GMM
for lab in unique_labs:

    # 进行数据筛选
    index = np.where(lab_train==lab)[0]
    dataset = data_train[index,:]

    # 利用训练的数据训练 GMM
    mus,sigmas,ws = train_GMM(dataset,K=2,m=20)
    models[lab]={}
    models[lab]["ws"]=ws
    models[lab]["mus"]=mus
    models[lab]["sigmas"]=sigmas
```

计算测试数据在每个模型上的似然值

```python
# 进测试
pdfs = np.zeros([N_test,len(unique_labs)])
index2lab = {}
# 计算每条测试数据在不同GMM上的logpdf
for i, lab in enumerate(unique_labs):
    index2lab[i]= lab
    ws = models[lab]["ws"]
    mus = models[lab]["mus"]
    sigmas=models[lab]["sigmas"]
    # 计算每条测试数据在这个GMM上的pdf
    _,pdf = getlogPdfFromeGMM(data_test,mus,sigmas,ws)
    pdfs[:,i] = pdf

# 选取最大似然值 实现分类
det_labs_index = np.argmax(pdfs,axis = 1).tolist()
```

结果分析

```python
# 将分类结果转为字符串
det_labs_str = [index2lab[i] for i in det_labs_index]
# 进行测试结果输出并统计准确率
N_right =0
for i, lab_str in enumerate(det_labs_str):
    print("测试数据 %d 真实标签 %s 检测标签 %s"%(i,lab_test[i],lab_str))

    if lab_str == lab_test[i]:
        N_right = N_right+1

print("准确率为 %.2f%%"%(N_right*100/N_test))
```

```
测试数据 18 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 19 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 20 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 21 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 22 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 23 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 24 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 25 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 26 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 27 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 28 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 29 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 30 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 31 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 32 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 33 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 34 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 35 真实标签 Iris-virginica 检测标签 Iris-versicolor
测试数据 36 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 37 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 38 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 39 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 40 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 41 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 42 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 43 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 44 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 45 真实标签 Iris-versicolor 检测标签 Iris-versicolor
测试数据 46 真实标签 Iris-setosa 检测标签 Iris-setosa
测试数据 47 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 48 真实标签 Iris-virginica 检测标签 Iris-virginica
测试数据 49 真实标签 Iris-virginica 检测标签 Iris-virginica
准确率为 98.00%
```

检测结果

```
{'Iris-setosa': {'ws': array([0.39402107, 0.60597893]), 'mus': array(
[[4.81564238, 3.29694636, 1.38701516, 0.22828414],
       [5.03736233, 3.49279251, 1.51673888, 0.25380625]]), 'sigmas':
array([[[ 0.13375879,  0.10682633,  0.03618834,  0.00809864],
        [ 0.10682633,  0.18285167,  0.03188169, -0.00863784],
        [ 0.03618834,  0.03188169,  0.01343078,  0.00346548],
        [ 0.00809864, -0.00863784,  0.00346548,  0.0069349 ]],

       [[ 0.04630264,  0.05756617, -0.00830517,  0.00900663],
        [ 0.05756617,  0.09713673,  0.00572862,  0.01644271],
        [-0.00830517,  0.00572862,  0.03167733,  0.00408086],
        [ 0.00900663,  0.01644271,  0.00408086,  0.01373466]]])}, 'Ir
is-versicolor': {'ws': array([0.46999148, 0.53000852]), 'mus': array(
[[5.64017004, 2.71789978, 4.01803327, 1.22334826],
       [6.24832999, 2.82563431, 4.51466062, 1.42693318]]), 'sigmas':
array([[[0.06239092, 0.02193021, 0.07514929, 0.00607901],
        [0.02193021, 0.0311503 , 0.03886546, 0.01669104],
        [0.07514929, 0.03886546, 0.18662625, 0.02757355],
        [0.00607901, 0.01669104, 0.02757355, 0.01447079]],

       [[0.16558975, 0.04527669, 0.08135174, 0.00879371],
        [0.04527669, 0.09309925, 0.04504553, 0.03768444],
        [0.08135174, 0.04504553, 0.13301317, 0.05059577],
        [0.00879371, 0.03768444, 0.05059577, 0.03542002]]])}, 'Iris-v
irginica': {'ws': array([0.66787162, 0.33212838]), 'mus': array([[6.3
1165038, 2.98856619, 5.34603353, 2.07506742],
       [7.23001901, 2.95608353, 6.12742422, 1.88250241]]), 'sigmas':
array([[[0.22493145, 0.06224243, 0.12642455, 0.0500324 ],
        [0.06224243, 0.05823747, 0.05723793, 0.04267045],
        [0.12642455, 0.05723793, 0.14490017, 0.05790027],
        [0.0500324 , 0.04267045, 0.05790027, 0.06889828]],

       [[0.32173969, 0.16267393, 0.30498326, 0.1131243 ],
        [0.16267393, 0.20520604, 0.11800457, 0.04541396],
        [0.30498326, 0.11800457, 0.32946502, 0.12940313],
        [0.1131243 , 0.04541396, 0.12940313, 0.05878717]]])}}
```

训练得到的GMM模型