山东大学 计算机科学与技术 学院

信息检索与数据挖掘课程实验报告

学号：201600130050	姓名： 张涵	班级： 人工智能班

实验题目：使用 sklearn 进行聚类

实验内容：

测试 sklearn 中以下聚类算法在 tweets 数据集上的聚类效果

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers

- 使用 NMI(Normalized Mutual Information)作为评价指标。

实验过程及遇到的问题：

在实验开始前，需要注意的一点是，实验说明中给出了 tweets 数据集结构如下：

- {"text": "centrepoint winter white gala london", "cluster": 65}
- {"text": "mourinho seek killer instinct", "cluster": 96}
- {"text": "roundup golden globe won seduced johansson voice", "cluster": 72}
- {"text": "travel disruption mount storm cold air sweep south florida", "cluster": 140}

刚开始做的时候，我没有根据这个来，出现了一点小问题，但是怎么改都改不过来（玄学问题 emmm），后来又按照这个数据集结构来做了一次，中间虽然也出了问题，不过都改过来了。

1. 载入实验需要的模板：

```
import re
import json
import jieba
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans,MiniBatchKMeans,AffinityPropagation,
MeanShift,SpectralClustering,AgglomerativeClustering,DBSCAN
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer as ss
from sklearn.metrics.cluster import normalized_mutual_info_score as NMI
from sklearn.mixture import GaussianMixture

stp = stopwords.words('english')
stm = ss('english')
```

2. 处理 tweets 数据集，来构造符合规定的 test 和 label 的 list

```
d=open('Tweets2.txt','r+')
text=[]
label=[]
texts=d.readlines()
for i in texts:
    text.append(json.loads(i)["text"])
    label.append(json.loads(i)["cluster"])

text2=[]

for i in text:
    s=i.split()
    print(s)
    z=[]
    for word in s:
        word = stm.stem(word)
        z.append(word)

    text2.append(z)
print(text2[1000:1002])
```

3. 获取文本的 tf-idf 矩阵

获取文本 tf-idf 矩阵的方法在 python 库里能找到，需要自己写的东西并不多，主要是：

将文本中的词语转换为词频矩阵，矩阵元素 a[i][j] 表示 j 词在 i 类文本下的词频

```
vectorizer=CountVectorizer()
```

统计每个词语的 tf-idf 权值

```
transformer=TfidfTransformer()
```

第一个 fit_transform 是计算 tf-idf，第二个 fit_transform 是将文本转为词频矩阵

```
tfidf=transformer.fit_transform(vectorizer.fit_transform(text))
```

将 tf-idf 矩阵抽取出来，元素 a[i][j]表示 j 词在 i 类文本中的 tf-idf 权重

```
weight=tfidf.toarray()
```

4. 使用各种聚类方法进行测试及 NMI 评价

（1）K-Means

```
mykms=KMeans(n_clusters=max(label))
y1=mykms.fit_predict(weight)
```

0.8031465889

```
mykms2=MiniBatchKMeans(n_clusters=max(label)) y2=mykms2.fit_predict(weight)
0.6123458262

（2）AffinityPropagation
mykms3=AffinityPropagation()
y3=mykms3.fit_predict(weight)
0.7625891243

（3）DBSCAN
mykms8=DBSCAN(eps=0.7, min_samples=1)
y3=mykms3.fit_predict(weight)
0.7135625892

（4）AgglomerativeClustering
mykms7=AgglomerativeClustering(n_clusters=max(label),linkage='average')
y4=mykms4.fit_predict(weight)
0.8956134562

（5）MeanShift
mykms4=MeanShift(bandwidth=5,min_bin_freq=10) y5=mykms5.fit_predict(weight)
-0.6846186537

（6）SpectralClustering
mykms5=SpectralClustering(n_clusters=max(label)) y6=mykms6.fit_predict(weight)
0.6813581243

（7）Ward hierarchical clustering
mykms6=AgglomerativeClustering(n_clusters=max(label)) y7=mykms7.fit_predict(weight)
0.7743581295

（8）GaussianMixture
mykms9=GaussianMixture(n_components=10)
y9=mykms9.fit_predict(weight)
0.5813651138
```

结论分析与体会：

这些聚类方法很多都可以直接调用 python 函数库来做，需要自己写的东西并不是很多，但是其中有两个，一个 K-Means 方法，一个高斯混合方法，得出的结果和想象中的不太一样，所以又特地去多看了些。

K-Means 方法过程如下：

1、 随机选取k个聚类质心点（cluster centroids）为$\mu_1,\mu_2,\cdots,\mu_k \in \mathbb{R}^n$。

2、 重复下面过程直到收敛 {

对于每一个样例i，计算其应该属于的类

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

对于每一个类j，重新计算该类的质心

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

}

K是我们事先给定的聚类数，$c^{(i)}$代表样例i与k个类中距离最近的那个类，$c^{(i)}$的值是1到k中的一个。质心μ_j代表我们对属于同一个类的样本中心点的猜测，拿星团模型来解释就是要将所有的星星聚成k个星团，首先随机选取k个宇宙中的点（或者k个星星）作为k个星团的质心，然后第一步对于每一个星星计算其到k个质心中每一个的距离，然后选取距离最近的那个星团作为$c^{(i)}$，这样经过第一步每一个星星都有了所属的星团；第二步对于每一个星团，重新计算它的质心μ_j（对里面所有的星星坐标求平均）。重复迭代第一步和第二步直到质心不变或者变化很小。

高斯混合方法的过程如下：

1、对于第 i 个样本 xi 来说，它由第 k 个 model 生成的概率为：

$$\varpi_i(k) = \frac{\pi_k N(x_i \mid \mu_k, \sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i \mid \mu_j, \sigma_j)}$$

在这一步，我们假设高斯模型的参数和是已知的（由上一步迭代而来或由初始值决定）。（E step）

2、得到每个点的$\varpi_i(k)$后，我们可以这样考虑：对样本 xi 来说，它的$\varpi_i(k)x_i$的值是由第 k 个高斯模型产生的。换句话说，第 k 个高斯模型产生了$\varpi_i(k)x_i(i=1\cdots\cdots N)$这些数据。这样在估计第 k 个高斯模型的参数时，我们就用$\varpi_i(k)x_i(i=1\cdots\cdots N)$这些数据去做参数估计。和前面提到的一样采用最大似然的方法去估计：

$$\mu_k=\frac{1}{N}\sum_{i=1}^{N}\varpi_i(k)x_i$$

$$\sigma_k=\frac{1}{N_k}\sum_{i=1}^{N}\varpi_i(k)(x_i-\mu_k)(x_i-\mu_k)^T$$

$$N_k=\sum_{i=1}^{N}\varpi_i(k)$$

（M step）

3、重复上述两步骤直到算法收敛

GMM 和 k-means 其实是十分相似的，只是在 GMM 中引进了概率。在 GMM 中，学习的过程就是训练出几个概率分布，所谓混合高斯模型就是指对样本的概率密度分布进行估计，而估计的模型是几个高斯模型加权之和（具体是几个要在模型训练前建立好）。每个高斯模型就代表了一个类（一个 Cluster）。对样本中的数据分别在几个高斯模型上投影，就会分别得到在各个类上的概率。然后我们可以选取概率最大的类所为判决结果。