实验二 实验报告

姓名：单宝迪 学号：201700210069 班级：17数据

实验环境和实验时间

实验环境：

- 硬件环境1: Intel(R) Core(TM) i7-8550U 16GRAM
- 硬件环境2: Intel(R) E5 128GRAM
- 软件环境: Windows 10 专业版　Python3.7
- IDE: Pycharm　Jupyter-Notebook

> 备注：基于digits数据的聚类实验在本地PC运行，基于fetch_20newsgroups的数据在服务器上运行。

实验目标

- 测试sklearn中部分聚类算法在digits,fetch_20newsgroups两个数据集上的聚类效果。

实现过程

数据一

1.数据的加载

通过下述代码加载数据集，并读取数据集的有关信息

```
from sklearn import metrics

digits = load_digits()
data = scale(digits.data)

n_samples, n_features = data.shape
n_digits = len(np.unique(digits.target))
labels = digits.target

sample_size = 1797
```

2.聚类操作

由于digits数据属于sklearn标准数据集，对于其聚类操作，仅需要简单的调用api，而不需要进行过多的处理。设置聚类对象的代码如下：

```
bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
              name="k-means++", data=data)

bench_k_means(KMeans(init='random', n_clusters=n_digits, n_init=10),
```

```
                name="random", data=data)

bench_AffinityPropagation(AffinityPropagation(convergence_iter=20),
                          name="AP", data=data)

bench_MeanShift(MeanShift(), name="MeanShift", data=data)


bench_SpectralClustering(SpectralClustering(
    n_clusters=n_digits), name="Spectral", data=data)

bench_AgglomerativeClustering(AgglomerativeClustering(n_clusters=n_digits,
linkage='ward', connectivity=None),
                          name="Ward-hier", data=data)

bench_AgglomerativeClustering(AgglomerativeClustering(n_clusters=n_digits,
linkage='complete', connectivity=None),
                          name="Agglomerative", data=data)

bench_DBSCAN(DBSCAN(eps=5, min_samples=3), name="DBSCAN", data=data)

bench_GaussianMixture(mixture.GaussianMixture(n_components=n_digits,
covariance_type='full'),
                      name="GaussMix", data=data)
```

3.聚类评估

经过运行对应代码，得到了如下的Evaluation：

init	time	inertia	NMI	Homogeneity	Completeness
k-means++	1.58s	69432	0.626	0.602	0.650
random	0.33s	69694	0.689	0.669	0.710
AP	13.30s	89	0.655	0.932	0.460
MeanShift	9.69s	unknown	0.063	0.014	0.281
Spectral	578.17s	unknown	0.012	0.001	0.271
Ward-hier	0.52s	unknown	0.797	0.758	0.836
Agglomerative	0.65s	unknown	0.065	0.017	0.249
DBSCAN	1.01s	unknown	1.000	1.000	1.000
GaussMix	0.89s	16	0.642	0.610	0.676
PCA-based	0.10s	70804	0.685	0.671	0.698

数据二

1.数据的加载

```
import logging
from optparse import OptionParser
import sys
from time import time

import numpy as np
name=[]
NMI=[]
Homogeneity=[]
Completeness=[]
# Display progress logs on stdout
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(levelname)s %(message)s')

op = OptionParser()
op.add_option("--lsa",
              dest="n_components", type="int",
              help="Preprocess documents with latent semantic analysis.")
op.add_option("--no-minibatch",
              action="store_false", dest="minibatch", default=False,
              help="Use ordinary k-means algorithm (in batch mode).")
op.add_option("--no-idf",
              action="store_false", dest="use_idf", default=True,
              help="Disable Inverse Document Frequency feature weighting.")
op.add_option("--use-hashing",
              action="store_true", default=False,
              help="Use a hashing feature vectorizer")
op.add_option("--n-features", type=int, default=10000,
              help="Maximum number of features (dimensions)"
                   " to extract from text.")
op.add_option("--verbose",
              action="store_true", dest="verbose", default=False,
              help="Print progress reports inside k-means algorithm.")


def is_interactive():
    return not hasattr(sys.modules['__main__'], '__file__')


argv = [] if is_interactive() else sys.argv[1:]
(opts, args) = op.parse_args(argv)
if len(args) > 0:
    op.error("this script takes no arguments.")
    sys.exit(1)



categories = [
    'alt.atheism',
    'talk.religion.misc',
    'comp.graphics',
    'sci.space',
```

```
]

print("Loading 20 newsgroups dataset for categories:")
print(categories)

dataset = fetch_20newsgroups(subset='all', categories=categories,
                             shuffle=True, random_state=42)

print("%d documents" % len(dataset.data))
print("%d categories" % len(dataset.target_names))
print()

labels = dataset.target
true_k = np.unique(labels).shape[0]

print("Extracting features from the training dataset "
      "using a sparse vectorizer")
t0 = time()
if opts.use_hashing:
    if opts.use_idf:
        hasher = HashingVectorizer(n_features=opts.n_features,
                                   stop_words='english', alternate_sign=False,
                                   norm=None, binary=False)
        vectorizer = make_pipeline(hasher, TfidfTransformer())
    else:
        vectorizer = HashingVectorizer(n_features=opts.n_features,
                                       stop_words='english',
                                       alternate_sign=False, norm='l2',
                                       binary=False)
else:
    vectorizer = TfidfVectorizer(max_df=0.5, max_features=opts.n_features,
                                 min_df=2, stop_words='english',
                                 use_idf=opts.use_idf)
X = vectorizer.fit_transform(dataset.data)

print("done in %fs" % (time() - t0))
print("n_samples: %d, n_features: %d" % X.shape)
print()

if opts.n_components:
    print("Performing dimensionality reduction using LSA")
    t0 = time()
    svd = TruncatedSVD(opts.n_components)
    normalizer = Normalizer(copy=False)
    lsa = make_pipeline(svd, normalizer)

    X = lsa.fit_transform(X)

    print("done in %fs" % (time() - t0))

    explained_variance = svd.explained_variance_ratio_.sum()
    print("Explained variance of the SVD step: {}%".format(
        int(explained_variance * 100)))
```

```
    print()
```

2.聚类操作

根据Scikit-Learn官方文档提供的操作，通过更改Api接口的调用，实现多个聚类方法的实现。

Kmeans

```
km = KMeans(n_clusters=true_k, init='k-means++', max_iter=100, n_init=1,
            verbose=opts.verbose)

print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

DBSCAN

```
km = DBSCAN(eps=5, min_samples=3)

print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

AffinityPropagation

```
km = AffinityPropagation(convergence_iter=20)

print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

Ward hierarchical clustering

```
km = AgglomerativeClustering(n_clusters=4, linkage='ward',connectivity=None)
```

```
print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

AgglomerativeClustering

```
km = AgglomerativeClustering(n_clusters=4, linkage='complete',connectivity=None)

print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

GaussianMixture

```
from sklearn import mixture
km = mixture.GaussianMixture(n_components=4, covariance_type='full')

print("Clustering sparse data with %s" % km)
t0 = time()
km.fit(X)
#print(km.cluster_centers_)
print("done in %0.3fs" % (time() - t0))
```

MeanShift

经实验，MeanShift不适用于文本聚类

3.聚类评估

init	time	NMI	Homogeneity	Completeness
k-means++	4.13s	0.426	0.426	0.529
AP	11.870s	0.885	0.885	0.191
Spectral	578.17s	0.012	0.001	0.271
Ward-hier	46.737s	0.797	0.758	0.836
Agglomerative	47.260s	0.066	0.064	0.068
DBSCAN	295.972s	0	0	1.000
GaussMix	181.503s	0.569	0.534	0.607

备注：Jupyter Notebook文件只是中间形式，实验结果以py文件为准。