

Homework Report

Clustering with sklearn

姓名：黄巧

学号：201934715

实验目的

- 1、理解聚类算法的原理，学会应用。
- 2、调用 sklearn 模块中的聚类函数以及评估函数对数据集进行聚类分析和效果评价。

实验环境

windows10, python3.5, scikit-learn0.21.2

实验内容

- 1、测试 sklearn 中以下聚类算法在 sklearn.datasets.loaddigits, sklearn.datasets.fetch20newsgroups 两个数据集上的聚类效果。

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large <code>n_samples</code> , medium <code>n_clusters</code> 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 <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium <code>n_samples</code> , small <code>n_clusters</code>	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large <code>n_samples</code> , medium <code>n_clusters</code>	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

- 2、使用 Homogeneity,Completeness, NMI(Normalized Mutual Information)作为评价指标，评估以上聚类算法的聚类效果。

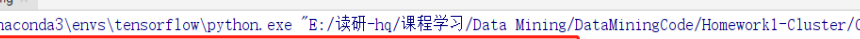
实验过程

（一）数据预处理

1、`fetch20newsgroups` 数据集包括 18846 篇新闻文章，一共涉及到 20 种话题。`sklearn` 提供了该数据的接口：`sklearn.datasets.fetch20newsgroups`，首先加载数据集，选取 `categories=['alt.atheism','talk.religion.misc','comp.graphics','sci.space','rec.autos']` 的类别进行聚类分析，使用 `TfidfVectorizer` 方法进行文本向量化与 TF-IDF 预处理，得到一个 `[nsamples,nfeatures]` 维度的稀疏矩阵。

```
vectorizer = TfidfVectorizer(max_df=0.5,          # 词汇表中过滤掉df>(0.5*doc_num)的单词
                             max_features=3000,   # 构建词汇表仅考虑max_features(按语料词频排序)
                             min_df=2,           # 词汇表中过滤掉df<2的单词
                             stop_words='english', # 词汇表中过滤掉英文停用词
                             use_idf=True)        # 启动inverse-document-frequency重新计算权重
X = vectorizer.fit_transform(dataset.data)        # 稀疏矩阵
```

执行结果如下图所示:



```
Run: Clustering x
D:\Anaconda3\envs\tensorflow\python.exe "E:/读研-hq/课程学习/Data Mining/DataMiningCode/Homework1-Cluster/Clustering.py"
Loading 20 newsgroups dataset for categories:
4377 documents
5 categories

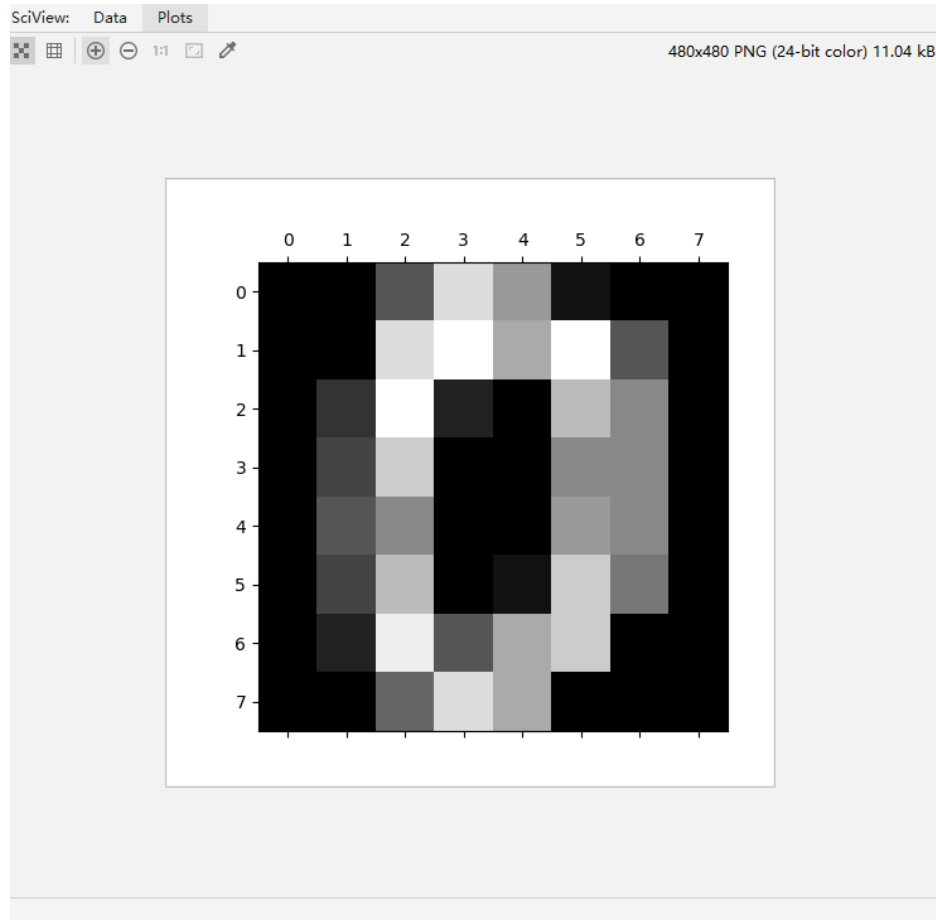
Extracting features from the training dataset using a sparse vectorizer
done in 0.772907s
n_samples: 4377, n_features: 3000
```

2、load_digits 数据集包括有 1797 个样本，每个样本包括 8*8 像素的图像和一个[0,9]整数的标签。

```
Run: Clustering_digits x
D:\Anaconda3\envs\tensorflow\python.exe "E:/读研-hq/课程学习/Data Mining/DataMiningCode/Homework1-Cluster/digits.keys= dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
digits.target = [0 1 2 ..., 8 9 8]
n_digits: 10,      n_samples 1797,      n_features 64
digits.images.shape = (1797, 8, 8)
digits.images = [[ [ 0.  0.  5. ...,  1.  0.  0.]
[ 0.  0. 13. ..., 15.  5.  0.]
[ 0.  3. 15. ..., 11.  8.  0.]
...,
[ 0.  4. 11. ..., 12.  7.  0.]
[ 0.  2. 14. ..., 12.  0.  0.]
[ 0.  0.  6. ...,  0.  0.  0.]

[[ 0.  0.  0. ...,  5.  0.  0.]
[ 0.  0.  0. ...,  9.  0.  0.]
[ 0.  0.  3. ...,  6.  0.  0.]
```

images 是一个 179788 的三维矩阵，即有 1797 张 8*8 的数字图片组成。



（二）聚类算法调用与评估

1、分别调用 `sklearn` 中的八种聚类算法对 `fetch_20newsgroups` 数据进行聚类分析，算法的关键函数如下：

(1) KMeans

K 均值聚类是以样本间距离为基础，将所有的观测样本划分到 K 个群体，使得群体和群体之间的距离尽量大，同时群体内部的样本之间的“距离和”最小。K 均值是期望最大化算法的特殊情况，K 均值是在每次迭代中只计算聚类分布的质心。

优点：

- KMeans 算法具有实现简单，收敛速度快的优点。

缺点：

- 必须预先指定聚类的数目，在聚类数目未知时效果较差，而且有时候不稳定，陷入局部收敛。

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

(2) Affinity propagation

AP 聚类是通过在样本对之间发送消息直到收敛来创建聚类。然后使用少量示例样本作为聚类中心来描述数据集，聚类中心是数据集中最能代表一类数据的样本。在样本对之间发送的消息表示一个样本作为另一个样本的示例样本的适合程度，适合程度值在根据通信的反馈不断更新。更新迭代直到收敛，完成聚类中心的选取，因此也给出了最终聚类。

优点：

- AP 聚类不需要指定 K (经典的 K-Means)或者是其他描述聚类个数(SOM 中的网络结构和规模)的参数。
- 一个聚类中最具代表性的点在 AP 算法中叫做 **Exemplar**，与其他算法中的聚类中心不同，**exemplar** 是原始数据中确切存在的一个数据点，而不是由多个数据点求平均而得到的聚类中心(K-Means)。
- 多次执行 AP 聚类算法，得到的结果是完全一样的，即不需要进行随机选取初值步骤。

缺点：

- 算法复杂度较高，为 $O(NM \log N)$ ，而 K-Means 只是 $O(N \cdot K)$ 的复杂度。因此当 N 比较大时 ($N > 3000$)，AP 聚类算法往往需要算很久。。
- 若以误差平方和来衡量算法间的优劣，AP 聚类比其他方法的误差平方和都要低。(无论 k-center clustering 重复多少次，都达不到 AP 那么低的误差平方和)

```
ap = AffinityPropagation(damping=0.5, preference=None).fit(X)
```

(3) Mean-shift

MeanShift 算法旨在发现一个样本密度平滑的 **blobs**。均值漂移算法是基于质心的算法，通过更新质心的候选位置为所选定区域的偏移均值。然后，这些候选者在后处理阶段被过滤以消除近似重复，从而形成最终质心集合。即聚类中心是通过在给定区域中的样本均值确定的，通过不断更新聚类中心，直到聚类中心不再改变为止

优点：

- 不同于 K-Means 算法，均值漂移聚类算法不需要我们知道有多少类/组。
- 基于密度的算法相比于 K-Means 受均值影响较小。

缺点：

- 窗口半径 r 的选择可能是不重要的。

```
ms = MeanShift(bandwidth=0.9, bin_seeding=True)
X_array = X.toarray()
ms.fit(X_array)
```

(4) Spectral clustering

SpectralClustering 是在样本之间进行亲和力矩阵的低维度嵌入，其实是低维空间中的 **KMeans**。如果亲和度矩阵稀疏，则这是非常有效的并且 **SpectralClustering** 需要指定聚类数。这个算法适用于聚类数少时，在聚类数多时不建议使用。

```
sc = SpectralClustering(n_clusters=true_k).fit(X)
```

(5) Ward hierarchical clustering

Hierarchical clustering 是一个常用的聚类算法，它通过不断的合并或者分割来构建聚类。聚类的层次被表示成树（或者 **dendrogram**（树形图））。树根是拥有所有样本的唯一聚类，叶子是仅有一个样本的聚类。

Ward 方法是一种质心算法。质心方法通过计算集群的质心之间的距离来计算两个簇的接近度。对于 **Ward** 方法来说，两个簇的接近度指的是当两个簇合并时产生的平方误差的增量。

优点：

- 能够展现数据层次结构，易于理解
- 可以基于层次事后再选择类的个数（根据数据选择类，但是数据量大，速度慢）

缺点：

- 计算量比较大，不适合样本量大的情形，较多用于宏观综合评价。

```
ward = AgglomerativeClustering(n_clusters=true_k, linkage='ward').fit(X_array)
```

(6) Agglomerative clustering

凝聚聚类对象使用自底向上的方法执行层次聚类：每个观察从其自己的集群中开始，集群依次合并在一起。链接标准决定了用于合并策略的度量：

```
ac = AgglomerativeClustering(n_clusters=true_k, linkage='complete').fit(X_array)
```

(7) DBSCAN

DBSCAN 算法将聚类视为被低密度区域分隔的高密度区域。**DBSCAN** 发现的聚类可以是任何形状的，与假设聚类是凸区域的 **K-means** 相反。**DBSCAN** 的核心概念是 **core samples**，是指位于高密度区域的样本。因此一个聚类是一组核心样本，每个核心样本彼此靠近（通过一定距离度量测量）和一组接近核心样本的非核心样本（但本身不是核心样本）。算法中的两个参数，**minsamples** 和 **eps**，正式的定义了 **dense**（稠密）。较高的 **minsamples** 或者较低的 **eps** 表示形成聚类所需的较高密度。

优点：不需要知道簇的数量；对噪声不敏感；能发现任意形状的聚类。

缺点：

- 需要确定距离 r 和 minPoints ;
- 聚类的结果与参数有很大的关系; DBSCAN 用固定参数识别聚类, 但当聚类的稀疏程度不同时, 相同的判定标准可能会破坏聚类的自然结构, 即较稀的聚类会被划分为多个类或密度较大且离得较近的类会被合并成一个聚类。

```
dbscan = DBSCAN(min_samples=1,metric='cosine').fit(X)
```

(8) Gaussian mixtures

高斯混合模型是一种概率模型, 它假定所有的数据点都是由有限个未知参数的高斯分布的混合产生的。可以把混合模型看作是对 KMeans 的一般化, 它包含了关于数据的协方差结构以及潜在高斯分布中心的信息。

```
gm = GaussianMixture(n_components=50).fit(X_array)
```

2、利用 sklearn 自带的 Homogeneity、Completeness、NMI 评估函数对每个聚类算法进行效果评价, 得到评估分数。以 KMeans 为例, 评估函数调用如下所示, 分别得到运行时间、同质性 Homogeneity、完整性 Completeness、归一化 NMI 的评估分数:

```
print("done in %0.3fs" % (time() - t0))
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels, km.labels_))
print("Completeness: %0.3f" % metrics.completeness_score(labels, km.labels_))
print("Normalized Mutual Information (NMI): %0.3f" % metrics.normalized_mutual_info_score(labels, km.labels_))
```

(1) 同质性 Homogeneity、完整性 Completeness

同质性 Homogeneity: 每个群集只包含单个类的成员;

完整性 Completeness: 给定类的所有成员都分配给同一个群集;

数学公式如下图所示:

K 代表簇, C 代表类;

homogeneity(同质性): 每个簇只包含一个类的成员;

$$h = 1 - \frac{H(C|K)}{H(C)}$$

completeness(完整性): 给定类的所有成员都分配给同一个簇;

$$c = 1 - \frac{H(K|C)}{H(K)}$$

其中,

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n}\right)$$

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$$

n 代表样本总数, n_c 和 n_k 代表分别属于 c 类和簇 k 的样本数, 最后 $n_{c,k}$ 代表分配给簇 k 的类 c 的样本数;

$H(K|C)$ 和 $H(K)$ 以类似方式定义;

v_measure: 同质性和完整性的调和平均;

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

优点:

- 分数明确: 从 0 到 1 反应出最差到最优的表现;
- 解释直观: 具有不良 **v-measure** 的聚类可以在同质性和完整性方面进行定性分析, 以更好地感知到聚类的错误类型;
- 对簇结构不作假设: 可以比较两种聚类算法如 **k** 均值算法和谱聚类算法的结果。

缺点:

- 随着样本数量、簇的数量以及标定过的真实标签的不同, 完全随机的标签并不总是产生相同数值的同质性、完整性和 **v-measure** (随机标记不会产生零分, 特别是当簇的数量大时); 当样本数量超过 1000, 簇的数量小于 10 时, 可以忽略上述缺点, 但对于较小的样本数量或者较大数量的簇, 还是建议使用调整过的度量标准, 比如 **Adjusted Rand Index (ARI)**;
- 基于互信息的度量方式, 由于需要正确聚类标签, 在实践中几乎不可用;

(2)**NMI(Normalized Mutual Information)** 数学公式如下图所示:

假设两列标签分配 (数据集中有 N 个对象), U 和 V ;

$$\text{mutual_info_score: } MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log\left(\frac{P(i, j)}{P(i)P'(j)}\right)$$

$$P(i) = \frac{|U_i|}{N} \text{ 是从 } U \text{ 中随机选取的对象属于类 } U_i \text{ 的概率;}$$

$$P'(j) = \frac{|V_j|}{N} \text{ 是从 } V \text{ 中随机选取的对象属于类 } V_j \text{ 的概率;}$$

$$P(i, j) = \frac{|U_i \cap V_j|}{N} \text{ 是随机选择的对象属于两个类 } U_i \text{ 和 } V_j \text{ 的概率;}$$

$$\text{normalized_mutual_info_score: } NMI(U, V) = \frac{MI(U, V)}{\sqrt{H(U)H(V)}}$$

$$U \text{ 的熵: } H(U) = - \sum_{i=1}^{|U|} P(i) \log(P(i))$$

$$V \text{ 的熵: } H(V) = - \sum_{j=1}^{|V|} P'(j) \log(P'(j))$$

$$\text{adjusted_mutual_info_score: } AMI = \frac{MI - E[MI]}{\max(H(U), H(V)) - E[MI]}$$

优点:

- 对于随机的标签分配, **AMI** 趋近于 0 (而 **MI** 就不能保证获得接近 0 的值);
- **MI** 和 **NMI** 的取值范围是[0,1], 值越大意味着聚类结果与真实情况越吻合, 接近 0 的值代表两列聚类标签相对独立, 接近 1 的值代表两列聚类标签具有一致性, 0 代表两列聚类标签完全独立, 1 代表两列聚类标签完全相同 (**AMI** 的取值范围为[-1,1]);

- 基于互信息的度量方式对于簇的结构没有作出任何假设，例如，可以用于比较 K-Means 与谱聚类的结果；

缺点：

- 基于互信息的度量方式，由于需要正确聚类标签，在实践中几乎不可用，但是可以用来在无监督的环境下，比较各种聚类算法结果的一致性（基于互信息的三种度量方式是对称的）；

实验结果

1、在 fetch20newsgroups 数据集上八种聚类算法的效果评估如下表所示：

Run: 20newsgroups x

```

D:\Anaconda3\envs\tensorflow\python.exe "E:/读研-hq/课程学习/Data Mining/DataMiningCode/Homework1-Cluster/"
Loading 20 newsgroups dataset for categories:
4377 documents
5 categories

Extracting features from the training dataset using a sparse vectorizer
done in 0.753662s
n_digits: 5,      n_samples 4377,      n_features 3000

```

init	time	Homogeneity	Completeness	NMI
k-means++	2.57s	0.550	0.621	0.585
AffinityPropagation	25.52s	0.829	0.207	0.414
MeanShift	93.16s	0.534	0.200	0.327
SpectralClustering	5.53s	0.089	0.303	0.164
Ward hierarchical clustering	73.20s	0.350	0.470	0.405
AgglomerativeClustering	71.28s	0.014	0.016	0.015
DBSCAN	1.37s	0.855	0.224	0.437
GaussianMixture	465.87s	0.573	0.265	0.390

Process finished with exit code 0

效果评估图中用红色标注了效果最好的指标值，橙色次之；用蓝色标注了效果最差的指标值，绿色次之。

- 从运行时间指标看，DBSCAN 算法运行时间仅 1.37s 最快，K-means 算法 2.57s 以及 Spectral clustering 算法 5.53s 略慢一些，而 Gaussian mixtures 高达 465.87s 非常慢，效率较低。
- 从 Homogeneity 指标看，DBSCAN 算法 0.855 表现最好，AffinityPropagation 算法 0.829 也较好；而 Agglomerative clustering 算法 0.014、Spectral clustering 算法 0.089 均未超过 0.02，表现最差；其他几个算法集中在 0.5 左右较为一般。
- 从 Completeness 指标看，K-means 算法 0.621 表现最好，其次是 Ward hierarchical clustering 算法 0.470 次之；而 Agglomerative clustering 算法 0.016 表现最差；其他几个算法集中在 0.2/0.3 左右，较为一般。

- 从 NMI 指标看，K-means 算法 0.585 表现最好，DBSCAN 算法 0.437 次之；Agglomerative clustering 算法 0.015 表现最差，Spectral clustering 算法也相对较差；其他几个算法集中在 0.32~0.41 之间，表现较为一般。

三项指标综合来看：

Agglomerative clustering 算法三项评估指标均低于 0.02，运行时间也在 70s 以上，整体效果最差；

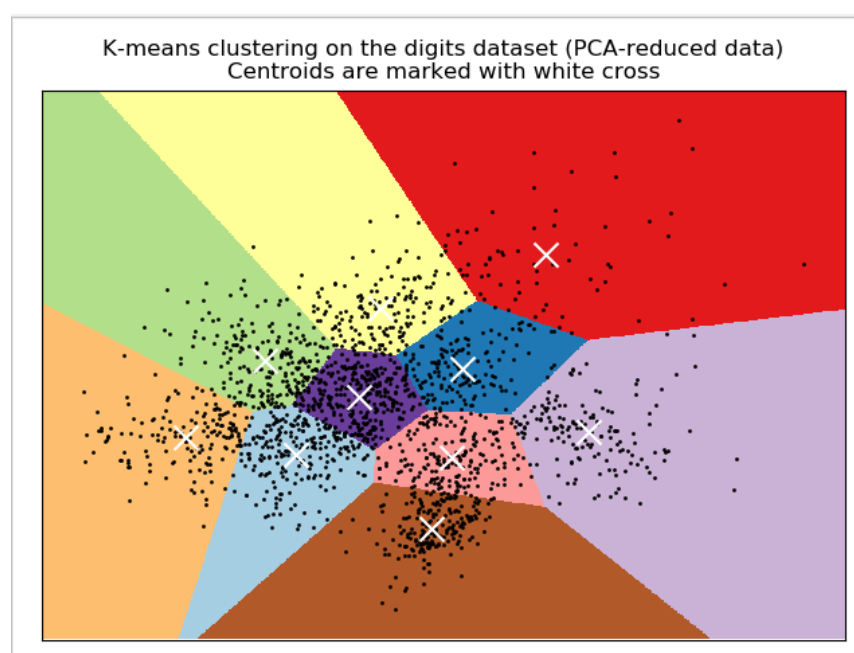
Gaussian mixtures 的运行时间高达 465s，各个指标值也相对一般，整体运行效果较差；

Affinity propagation 和 DBSCAN 算法运行效果相对较好，其中 DBSCAN 算法运行时间也相对较少，表现最好。

2、在 digits 数据集上八种聚类算法的效果评估如下图所示：

	time	Homogeneity	Completeness	NMI
GaussianMixture	0' 482	0' 800	0' 240	0' 080
DBSCAN	0' 022	0' 001	0' 515	0' 011
AgglomerativeClustering	0' 142	0' 011	0' 540	0' 002
Mark hierarchical clustering	0' 102	0' 128	0' 830	0' 101
SpectralClustering	1' 102	0' 010	0' 500	0' 010
MeanShift	1' 382	1' 000	0' 301	0' 224
AffinityPropagation	4' 112	0' 035	0' 400	0' 022
K-means++	0' 112	0' 005	0' 020	0' 030
Init				

其中 K-means 算法使用 PCA 降维后绘制的聚类图示如下：



效果评估图中用红色标注了效果最好的指标值，橙色次之；用蓝色标注了效果最差的指标值，绿色次之。

- 从运行时间指标看，DBSCAN 算法运行时间仅 0.05s 最快，Spectral clustering 算法 418s 最慢，AffinityPropagation 算法 4.71s 相对较慢，其他算法都在 0.1-1s 之间，从运行时间角度相对较好。
- 从 Homogeneity 指标看，MeanShift 算法可以达到 1.0 表现最好，AffinityPropagation 算法 0.932 也较好；而 DBSCAN 算法仅有 0.001 表现最差，Agglomerative clustering 算法 0.017、Spectral clustering 算法 0.019 均未超过 0.02，表现也较差；其他几个算法集中在 0.7 左右相对较好。
- 从 Completeness 指标看，Ward hierarchical clustering 算法 0.836 表现最好，K-means 算法 0.650 表现次之；而 Agglomerative clustering 算法 0.249 表现最差，DBSCAN 以及 Spectral clustering 算法也都处于 0.3 以下整体较差；其他几个算法集中在 0.5 左右，相对一般。
- 从 NMI 指标看，Ward hierarchical clustering 算法 0.797 表现最好，GaussianMixture 算法 0.689 次之；DBSCAN 算法仅有 0.017 表现最差，Agglomerative clustering 算法 0.065 以及 Spectral clustering 算法 0.076 也相对较差；其他几个算法集中在 0.6 左右，相对较好。

三项指标综合来看：

SpectralClustering 算法三项评估指标均较差且运行时间极长，整体效果最差；

DBSCAN 算法和 Agglomerative clustering 算法三项评估指标也均较差，效果也很不好；

Gaussian mixtures 算法、Ward hierarchical clustering 算法以及 AffinityPropagation 算法在三项指标的综合表现最好；

MeanShift 算法在 Homogeneity 表现极佳但在 Completeness 和 NMI 指标上不是很出色；

K-means 算法三项指标都在 0.6 左右比较平均。

3、综合两个数据集运行结果来看：

在 Homogeneity 方面：

AffinityPropagation 算法在两个数据集上表现均较好，都能达到 0.9 左右；而 Agglomerative clustering 算法以及 Spectral clustering 算法在两个数据集上表现都较差，均未超过 0.02，这两个算法在同质性方面效果较差。

在 Completeness 方面：

K-means 算法以及 Ward hierarchical clustering 算法在两个数据集上表现均较好；而 Agglomerative clustering 算法一直表现较差。

在 NMI 方面：

Agglomerative clustering 算法一直表现较差，其他算法各有波动。