实 验 报 告

学生姓名	王科翔	学号	16061040	指导老师	李辉勇
实验地点	F332	实验时间	19.03.19	班级	160612

一、 实验名称

聚类算法实验

二、 实验学时

4

三、 实验原理

K-means 算法的思想是将数据分为 k 个簇，通过使得簇内样本方差较降低，最小化簇内样本差距，达到聚类目标。算法初始需要指定目标聚类数量—k，由于算法简单，可以很好的适配大样本分类任务，在不同领域应用广泛。K-means 主要思路是将数据集 N 中的 X 个样本分为 K 个不相连的簇 C 中，用簇内样本距离均值 uj 来进行度量。C 的中心为“簇心（centroid）”，簇心由算法迭代计算获得，不一定属于 X。计算公式

$$\sum_{i=0}^{n} \min_{u_j \in C}(||x_i - u_j||^2)$$

K-means 主要依赖该公式对簇心进行迭代计算，获取最优值。

1. 算法输入：数据集 X；
2. 随机从 X 中选取 k 个初始簇心；
3. 遍历其余样本到 k 个簇心的距离（可以采用不同的距离度量方式），对每个样本，选择最近的簇心加入该簇；
4. 根据本轮形成的 k 个簇重新计算簇心；
5. 重复步骤 3，4，直到算法满足停止（如迭代次数、距离阈值等）条件。

层次聚类（Hierarchical），一般情况下可以分为“自顶向下”和“自底向上”两种。在自顶向下的层次聚类算法中，初始所有节点均为一个簇。通过计算簇之

间的关联度（linkage），依次选择最小关联度的两个簇合并，直到达到停止条件。不同于 K-means 计算点到点的距离，层次聚类中的簇间关联（簇与簇之间的距离度量）可以通过选择不同的关联度计算方式获得，如：Ward，Maximum ，Average linkage 和 Single linkage 计算获得。自底向上算法逻辑相反，初始状态所有样本为一个分类，依次选择能够生成簇间关联度最大的组合子簇进行分裂，直到达到停止条件。层次聚类也可以将目标类别作为聚类停止条件。

四、 实验目的

学习掌握 k 均值算法（K-means）和层次聚类（Hierarchical）基本原理；
并理解两种方法之间的联系和区别。

五、 实验内容

通过 PC 上位机连接服务器，登陆 SimpleAI 平台，利用 python 语言调用 K-means 和 Hierarchical 代码。分别利用 k-means 算法和 Hierarchical 对 HCI 数据集下的鸢尾花的数据进行聚类分析和对比，并分析算法性能。

六、 实验步骤

任务 1：

采用 DBSCAN 进行聚类， 并尝试不同的参数组合，提高三类算法的聚类精度
Kmeans 调参

```
def eva_kmeans(x, y):
    parameters = [
        {
            'n_clusters': [3],
            'init': ['k-means++', 'random'],
            'algorithm': ['auto']
        }
    ]
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=percentage,
                                                        random_state=np.random.random_integers(1, 100))
    kmeans = KMeans()
    clf = GridSearchCV(kmeans, parameters, cv=5, n_jobs=-1)
    clf.fit(x_train, y_train)
    kmeans_pred = clf.predict(x_test)
    print(clf.best_params_)
    print(classification_report(y_test, kmeans_pred, digits=3))
```

Hierarchical 调参

```python
def eva_hierarchical(x, y):
    X, labels_true = x, y
    nums = range(1, 50)
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    # 链接方式的影响
    linkages = ['ward', 'complete', 'average']
    markers = "+o*"
    for i, linkage in enumerate(linkages):
        ARIs = []
        for num in nums:
            clst = AgglomerativeClustering(n_clusters=num, linkage=linkage)
            # 预测
            predicted_labels = clst.fit_predict(X)
            # ARI指数
            ARIs.append(adjusted_rand_score(labels_true, predicted_labels))
        ax.plot(nums, ARIs, marker=markers[i], label="linkage:%s" % linkage)

    ax.set_xlabel("n_clusters")
    ax.set_ylabel("ARI")
    ax.legend(loc="best")
    fig.suptitle("AgglomerativeClustering")
    plt.show()
```

DBSCAN 调参：

```python
def eva_DBSCAN(x, y):
    ...
    X, labels_true = x, y
    nums = [0.125, 0.25, 0.5, 1,1.5, 2]
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    metric_ = ['euclidean', 'chebyshev']
    markers = "+o"
    for i, linkage in enumerate(metric_):
        ARIs = []
        for num in nums:
            print(num, "::", linkage)
            clst = DBSCAN(eps=num, metric=linkage)
            predicted_labels = clst.fit_predict(X)
            ARIs.append(adjusted_rand_score(labels_true, predicted_labels))
        ax.plot(nums, ARIs, marker=markers[i], label="metric:%s" % linkage)
    ax.set_xlabel("eps")
    ax.set_ylabel("ARI")
    ax.legend(loc="best")
    fig.suptitle("DBSCAN")
    plt.show()
```

任务 2：

将结果可视化，方便观察结果。

原代码中，对多次计算取平均值，但是当进行可视化的时候应该只选取一个计算结果，故我共用同一组数据，对不同方法和正确结构进行可视化，代码如下。

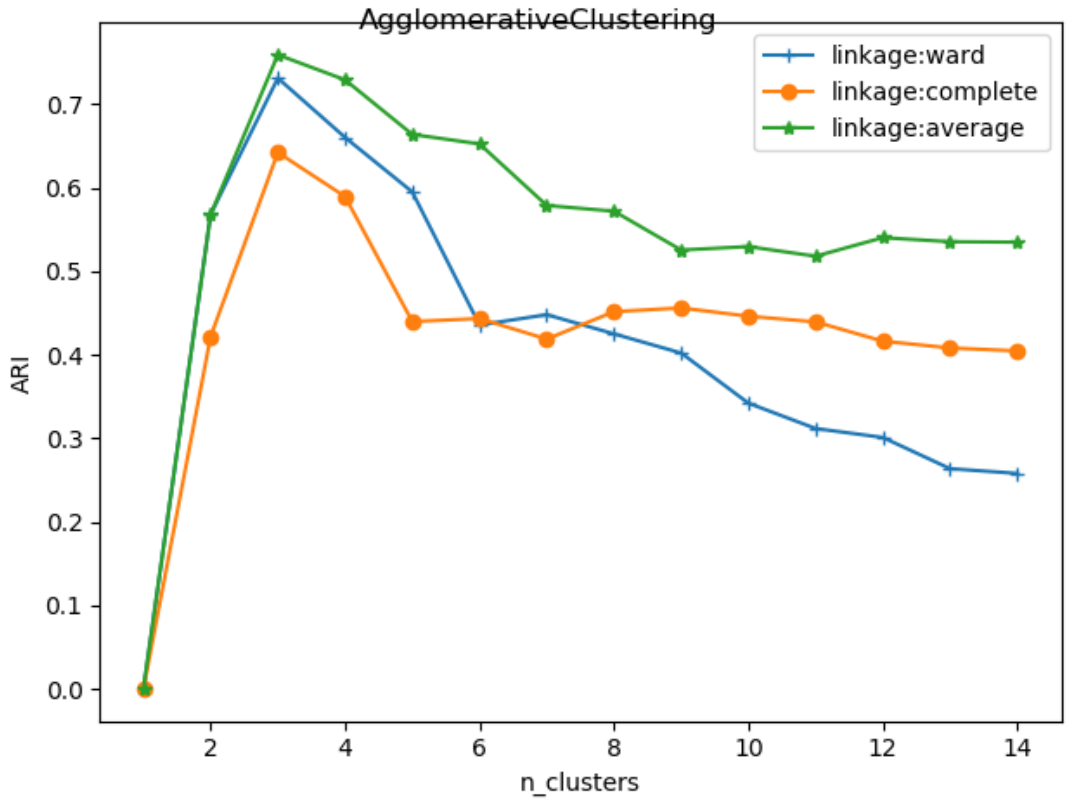
```
def visualize(x, y):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=percentage,
                                                        random_state=np.random.random_integers(1, 100))
    dbscan = DBSCAN().fit(x_train)
    dbscan_pred = dbscan.fit_predict(x_test)
    plt.scatter(x_test[:, 0], x_test[:, 1], c=dbscan_pred)
    plt.title("DBSCAN")
    plt.show()
    hier = AgglomerativeClustering(n_clusters=3, affinity="euclidean", linkage="average").fit(x_train, y_train)
    hier_pred = hier.fit_predict(x_test)
    plt.scatter(x_test[:, 0], x_test[:, 1], c=hier_pred)
    plt.title("hierarchical")
    plt.show()
    kmeans = KMeans(n_clusters=3, random_state=0).fit(x_train, y_train)
    kmeans_pred = kmeans.predict(x_test)
    plt.scatter(x_test[:, 0], x_test[:, 1], c=kmeans_pred)
    plt.title("kmeans")
    plt.show()
    plt.scatter(x_test[:, 0], x_test[:, 1], c=y_test)
    plt.title("y_test")
    plt.show()
```

七、 实验结果及分析：

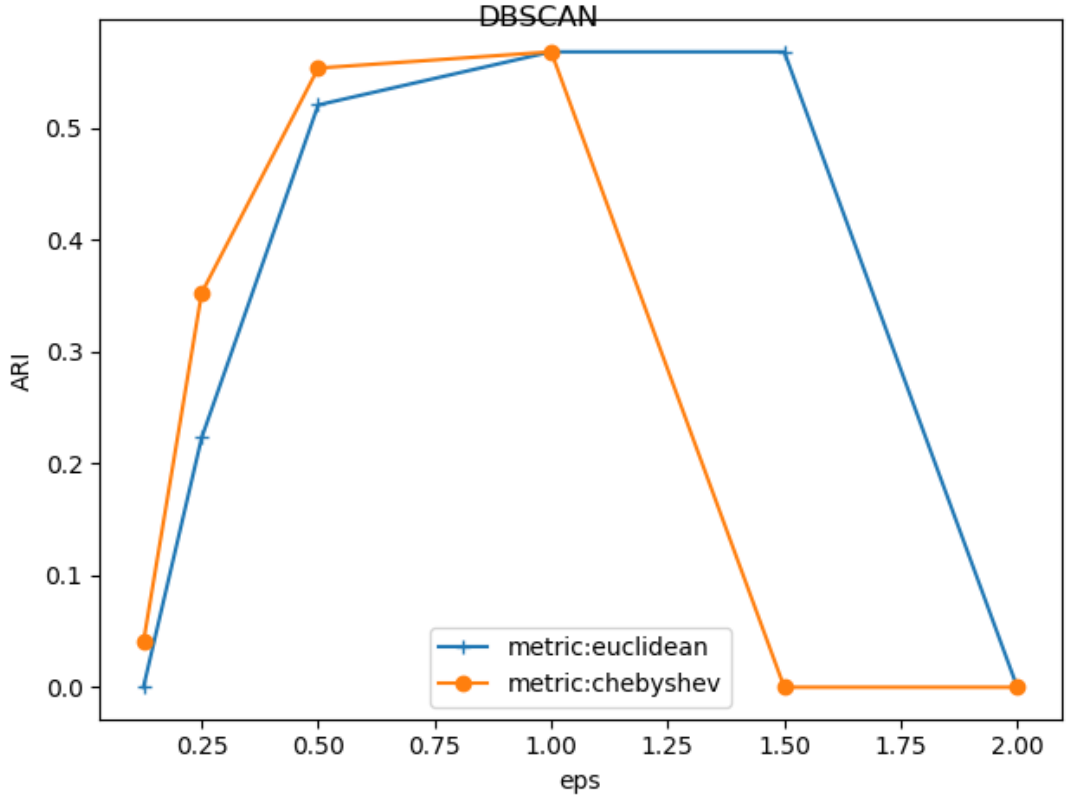
任务 1：

```
{'algorithm': 'auto', 'init': 'k-means++', 'n_clusters': 3}
              precision    recall  f1-score   support

           0      0.000     0.000     0.000        10
           1      0.000     0.000     0.000        10
           2      0.182     0.200     0.190        10

   micro avg      0.067     0.067     0.067        30
   macro avg      0.061     0.067     0.063        30
weighted avg      0.061     0.067     0.063        30
```

Kmeans 可进行调整的参数不多，由图可见，{'algorithm': 'auto', 'init': 'k-means++', 'n_clusters': 3}是比较好的参数组合。

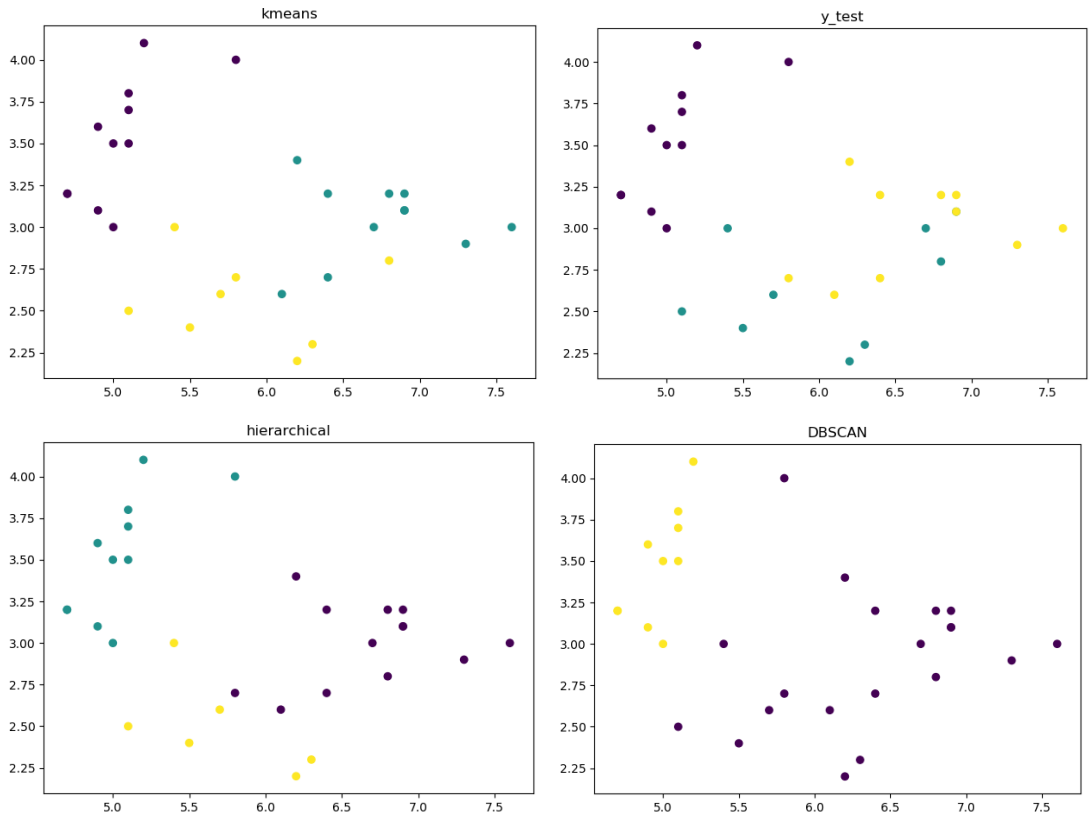


可见当聚类为 3 类，且 linkage 参数为 average 时，效果比较好。



可见，eps 参数为 1.0，同时，metric 参数为 Euclidean 的参数组合效果最好。

任务 2：



数据可视化效果如图，可见在 iris 的同一组数据集上，相对而言 kmeans 和 hierarchical 比较接近真实分类情况。

八、 实验结论：

任务 1：

聚类不是监督学习，不能使用 GridSearchCV 进行快速调参，所以只好对参数遍历进行分析，由于数据比较少，最终的分类的效果不是很好。

Kmeans：{'algorithm': 'auto', 'init': 'k-means++', 'n_clusters': 3}是比较好的参数组合。

AgglomerativeClustering：可见当聚类为 3 类，且 linkage 参数为 average 时，效果比较好。

DBSCAN：可见，eps 参数为 1.0，同时，metric 参数为 Euclidean 的参数组合效果最好。

任务 2：

由于 DBSCAN 不能指定分类的聚类个数，容易把两个靠近的 cluster 认为成一个团，故无法产生较好的效果，观察图像，可以发现，右下方两个团因为距离不够远，已经被分为一个团，这降低了效果。相对而言，拥有指定的分类数目的 kmeans 和 hierarchial 有比较好的效果。

九、 总结及心得体会

实验总结

学到了关于 kmeans 的理论知识和实践能力，提高了对于机器学习的实践能力。

存在问题

可视化只能可视化两层，不能对第三第四维的特征进行可视化。

对本实验意见或建议

提供的代码是进行了十次的实验然后求平均这种方式有点像摇色子，可视化画出来的图只能是其中一次摇出来的数字，不具有代表性。
应该是使用数量更大，更具有代表性的数据集进行这个实验，这样的一来可视化一目了然，二来，准确率高。