实验五 层次聚类

- 姓名：马永田
- 学号：2012911
- 专业：计算机科学与技术专业

实验要求

截止日期：12月2日实验课之前

- 以.ipynb形式的文件提交，输出运行结果，并确保自己的代码能够正确运行
- 发送到邮箱：2120220594@mail.nankai.edu.cn

基本要求

1. 实现single-linkage层次聚类算法；
2. 实现complete-linkage层次聚类算法；

中级要求

1. 实现average-linkage层次聚类算法；
2. 将上述三种算法的性能进行简要对比；

高级要求

1. 通过变换聚类簇的个数，测试上述三种算法的性能，并给出分析；

实验流程

基本要求&中级要求

实现实验要求中的三种算法

In [1]:
```
import numpy as np

MAX_NUM = 1e3

# method
def singleLinkage(dest,src,setList,Dist,X):
    # your code
    for i in range(len(X[0])):
        X[0][i] = min(X[0][i],X[1][i])
    return X[0]

def completeLinkage(dest,src,setList,Dist,X):
    # your code
    for i in range(len(X[0])):
        X[0][i] = max(X[0][i],X[1][i])
```

```
    return X[0]

def averageLinkage(dest,src,setList,Dist,X):
    # your code
    ListNum = len(setList[dest]) + len(setList[src])
    for i in range(len(X[0])):
        sumDist = 0
        for j in setList[i]:   #遍历第i类中的点
            for k in setList[dest]:# 遍历要合并的src类中的点
                sumDist += Dist[k][j]
            for k in setList[src]: # 遍历要合并的dest类中的点
                sumDist += Dist[k][j]
        X[0][i] = sumDist / (len(setList[i]) * ListNum)
    return X[0]

class AgglomerativeClustering:
    def __init__(self):
        # 对每次的合并进行记录
        self.steps=[]

    def fit(self,datas,method):
        self.dataCnt = datas.shape[0]
        # 预处理各点之间的距离
        allDist = np.zeros((self.dataCnt,self.dataCnt))
        for i in range(self.dataCnt):
            for j in range(i):
                allDist[i][j] = allDist[j][i] = np.sum((datas[i]-datas[j])**2)
        setList, clusterCount = [[i] for i in range(self.dataCnt)], self.dataCnt
        print("calculate distance finish!")

        # 聚类间距离矩阵
        clusterDist = np.zeros((self.dataCnt,self.dataCnt))+MAX_NUM
        for i in range(clusterCount):
            for j in range(i+1,clusterCount):
                clusterDist[i][j] = clusterDist[j][i] = allDist[i][j]
        print("calculate cluster distance finish!")

        while clusterCount != 1:
            # 最相似的两个聚类
            res = np.argmin(clusterDist)
            dest,src = int(res/clusterCount),res%clusterCount
            # steps进行一次记录
            self.steps.append((setList[dest][0],setList[src][0]))
            # 聚类间距离矩阵更新
            modify = method(dest,src,setList,allDist,clusterDist[[dest,src]])
            clusterDist[dest] = modify
            clusterDist[:,dest] = modify
            clusterDist = np.delete(clusterDist,src,axis=0)
            clusterDist = np.delete(clusterDist,src,axis=1)
            clusterDist[dest][dest] = MAX_NUM
            # 聚类更新
            setList[dest] = setList[dest] + setList[src]
            del setList[src]
            clusterCount -= 1
            #if (self.dataCnt - clusterCount) % (self.dataCnt / 20) == 0:
                #print(clusterCount, " clusters left.")

        print("cluster finish !")

    def label(self,k):
        root = list(range(self.dataCnt))
        def find_root(n):
            if root[root[n]] == root[n]:
                return root[n]
```

```python
                root[n]=find_root(root[n])
            return root[n]
        for i in range(self.dataCnt-k): # 根据steps记录产生非连通图
            src,dest = self.steps[i]
            root[find_root(dest)] = find_root(src)
        cluster, clusterNum = [0 for i in range(self.dataCnt)], 0
        for i in range(self.dataCnt): # 将根节点标注为新的cluster
            if i == root[i]: # i是根
                clusterNum += 1
                cluster[i] = clusterNum
        for i in range(self.dataCnt): # 将非根节点标注为根节点的cluster
            if i != root[i]: # i不是根
                cluster[i] = cluster[find_root(i)]
        return cluster
```

In [2]:

```python
from matplotlib import pyplot as plt
import numpy as np
from sklearn.datasets import make_blobs
import itertools

def create_data(centers,num=100,std=0.7):
    '''
    生成用于聚类的数据集
    :param centers: 聚类的中心点组成的数组。如果中心点是二维的，则产生的每个样本都是
    :param num: 样本数
    :param std: 每个簇中样本的标准差
    :return: 用于聚类的数据集。是一个元组，第一个元素为样本集，第二个元素为样本集的真
    '''
    X, labels_true = make_blobs(n_samples=num, centers=centers, cluster_std=std)
    return  X,labels_true


def plot_data(*data):
    '''
    绘制用于聚类的数据集
    :param data: 可变参数。它是一个元组。元组元素依次为：第一个元素为样本集，第二个元
    :return: None
    '''
    X,labels_true,labels_predict,cnt=data
    fig=plt.figure()
    ax=fig.add_subplot(1,1,1)
    colors='rgbyckm' # 每个簇的样本标记不同的颜色
    markers='o^sP*DX'
    for i in range(len(labels_true)):
        predict=labels_predict[i]
        ax.scatter(X[i,0],X[i,1],label="cluster %d"%labels_true[i],
        color=colors[predict%len(colors)],marker=markers[labels_true[i]%len(markers)

centers=[[1,1,1],[1,3,3],[3,6,5],[2,6,8]]# 用于产生聚类的中心点, 聚类中心的维度代表产
X,labels_true= create_data(centers,2000,0.5) # 产生用于聚类的数据集，聚类中心点的个数
np.savetxt('D:/大三上课程/机器学习及其应用/Machine-Learning/Lab5/data.dat',X)
np.savetxt('D:/大三上课程/机器学习及其应用/Machine-Learning/Lab5/label.dat',labels_t
print("generate data finish!")

METHOD_APPLY = [singleLinkage,completeLinkage,averageLinkage]
```

```
generate data finish!
```

计算准确率 对性能进行简要分析

In [3]:

```python
def Permutations(arrs):
    if len(arrs)==1:
        return [arrs]
```

```python
    result = []  # 结果
    for i in range(len(arrs)):
        rest_arrs = arrs[:i]+arrs[i+1:]  # 取出第i个元素后剩余的元素
        rest_lists = Permutations(rest_arrs)   # 剩余的元素完成全排列
        lists = []
        for term in rest_lists:
            lists.append(arrs[i:i+1]+term)  # 将取出的第i个元素加到剩余全排列的前面
        result += lists
    return result
```

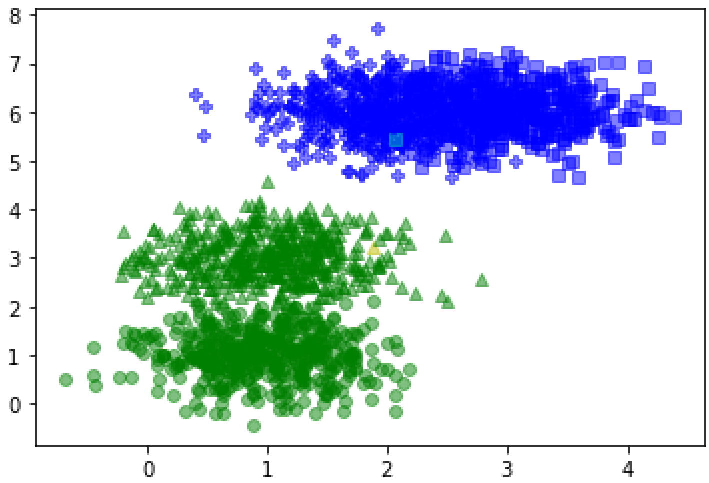
In [4]:

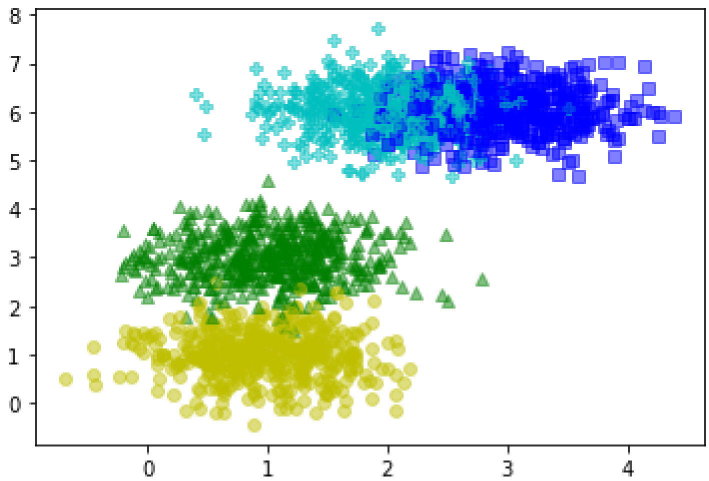
```python
def accCalculate(labels_true,predicts,k):
    res_acc = 0
    labels = [i for i in range(k)]
    for trans_label in Permutations(labels):
        acc_count = 0
        for i in range(len(predicts)):
            if labels_true[i] == trans_label[predicts[i]-1] :
                acc_count += 1
        acc = acc_count/len(labels_true)
        if acc > res_acc:
             res_acc = acc
    return res_acc
```

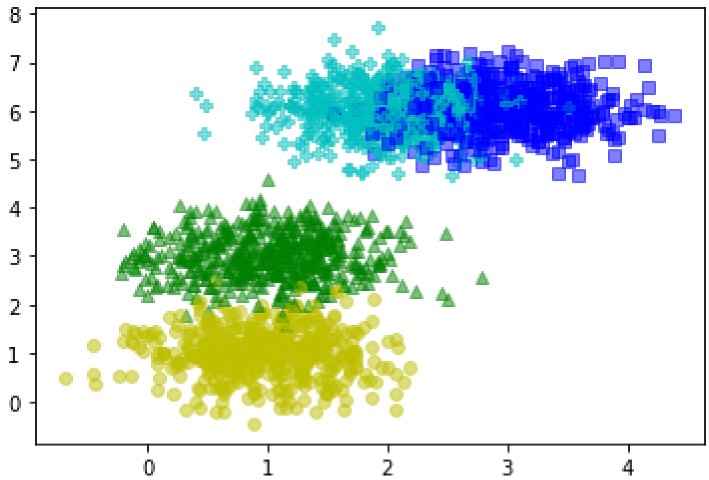
In [5]:

```python
cnt = 0
for method in METHOD_APPLY:
    model = AgglomerativeClustering()
    model.fit(X,method)
    k = 4
    print("Cluster Number = ",k)
    predict = model.label(k)
    plot_data(X,labels_true,predict,cnt)
    acc = accCalculate(labels_true,predict,k)
    cnt += 1
    print("Accuracy rate of clustering : ",acc,"%")
    print("-------Segmentation-------")
```

```
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Cluster Number =  4
Accuracy rate of clustering :  0.501 %
-------Segmentation-------
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Cluster Number =  4
Accuracy rate of clustering :  0.9995 %
-------Segmentation-------
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Cluster Number =  4
Accuracy rate of clustering :  0.9965 %
-------Segmentation-------
```







如上图可见,最小连接距离法性能最差,聚类正确率仅有50%左右,而最大连接距离法与平均连接距离法则性能比较好,且两者相差不大,聚类成功率接近100%.

高级要求

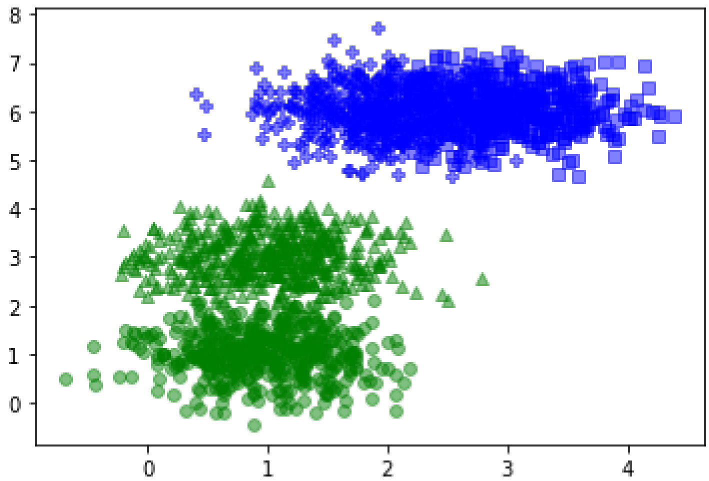
变换聚类簇个数，测试算法性能并进行分析

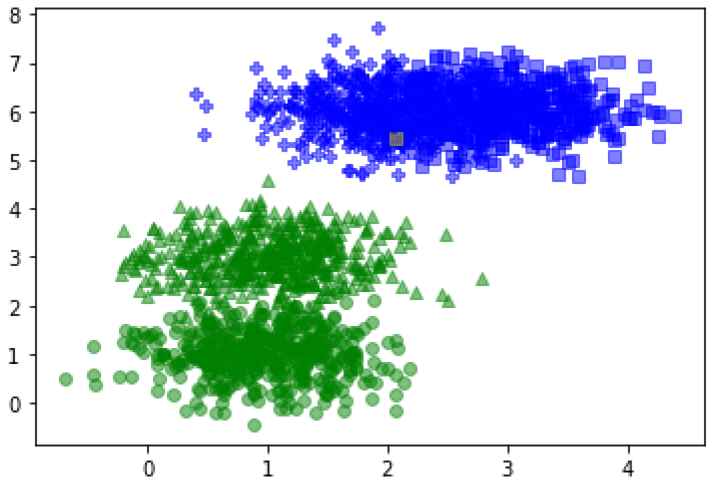
In [7]:

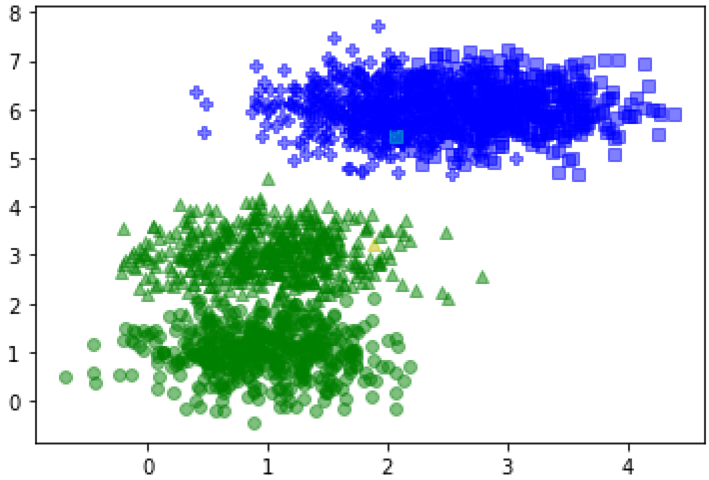
```
print("single-linkage hierarchical clustering:")
method = METHOD_APPLY[0]
cnt = 0
for k in range(2,6):
    print("Cluster Number = ",k)
    model = AgglomerativeClustering()
    model.fit(X,method)
```

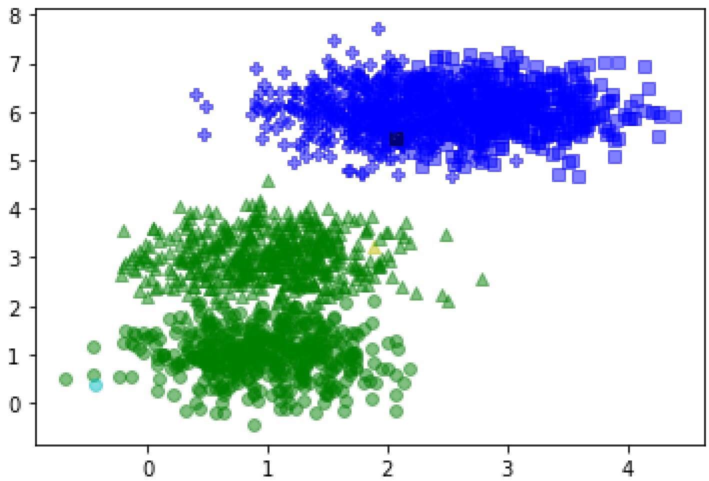
```
    predict = model.label(k)
    plot_data(X,labels_true,predict,cnt)
    acc = accCalculate(labels_true,predict,k)
    cnt += 1
    print("Accuracy rate of clustering : ",acc,"%")
    print("-------Segmentation-------")
```

```
single-linkage hierarchical clustering:
Cluster Number =  2
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.25 %
-------Segmentation-------
Cluster Number =  3
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.4995 %
-------Segmentation-------
Cluster Number =  4
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.501 %
-------Segmentation-------
Cluster Number =  5
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.5005 %
-------Segmentation-------
```



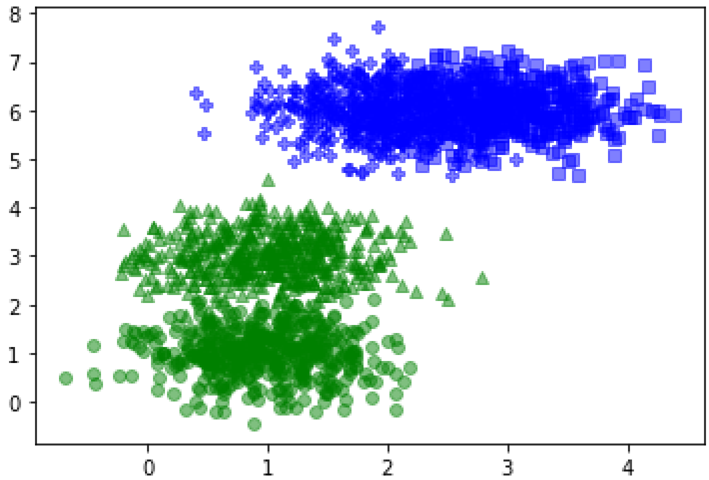


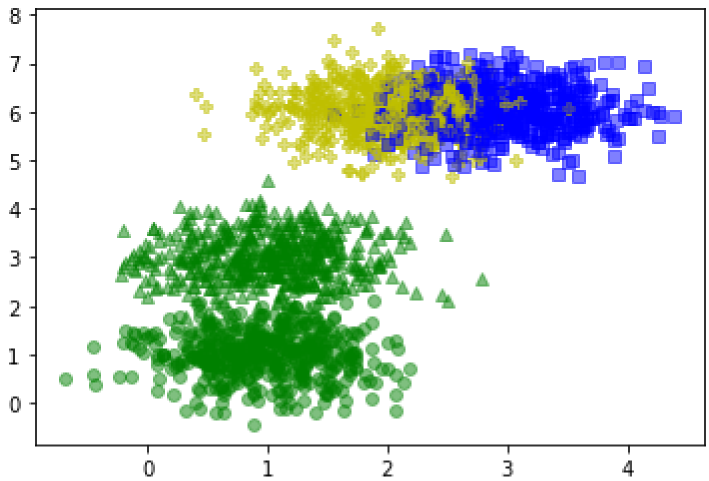


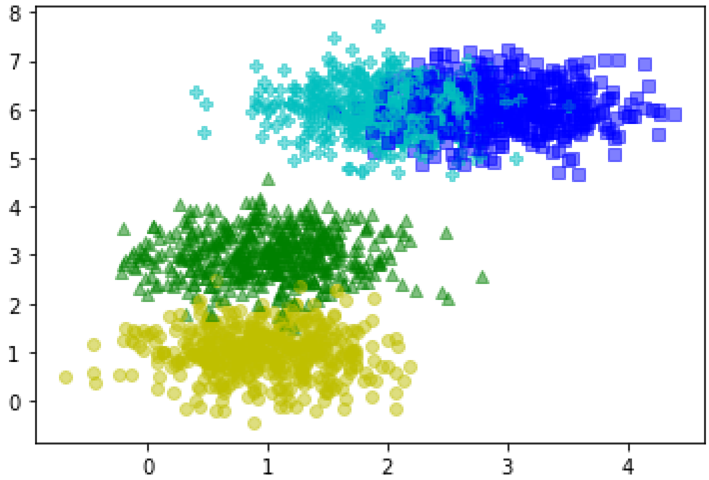


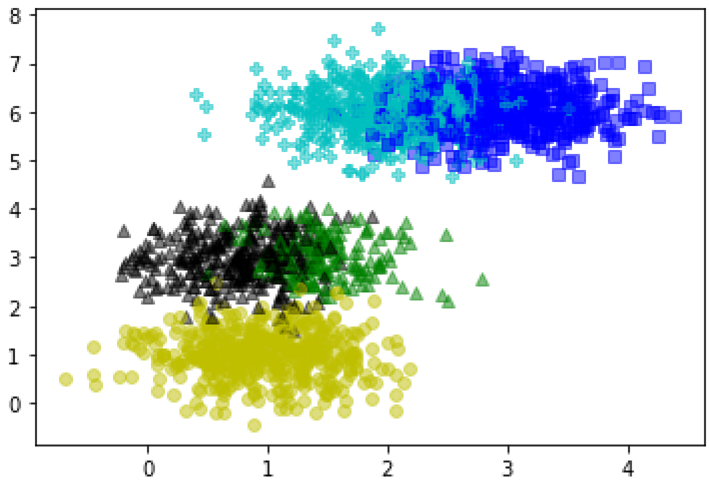
In [8]:
```
print("complete-linkage hierarchical clustering:")
method = METHOD_APPLY[1]
cnt = 0
for k in range(2,6):
    print("Cluster Number = ",k)
    model = AgglomerativeClustering()
    model.fit(X,method)
    predict = model.label(k)
    plot_data(X,labels_true,predict,cnt)
    acc = accCalculate(labels_true,predict,k)
    cnt += 1
    print("Accuracy rate of clustering : ",acc,"%")
    print("-------Segmentation-------")
```

```
complete-linkage hierarchical clustering:
Cluster Number =  2
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.25 %
-------Segmentation-------
Cluster Number =  3
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.4995 %
-------Segmentation-------
Cluster Number =  4
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.9995 %
-------Segmentation-------
Cluster Number =  5
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.9095 %
-------Segmentation-------
```





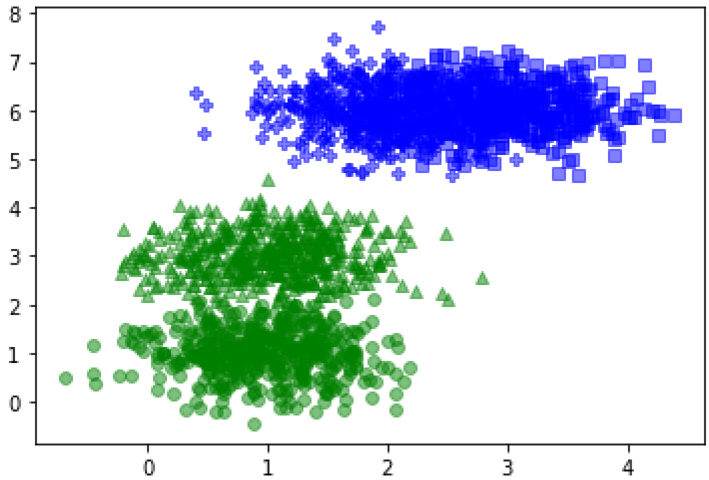


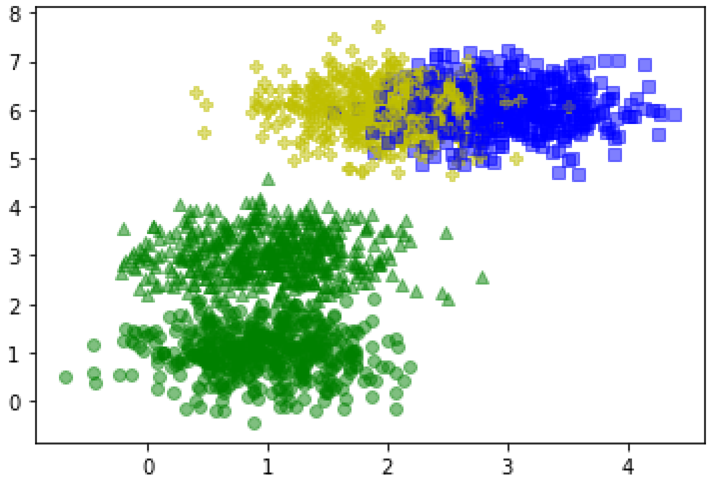


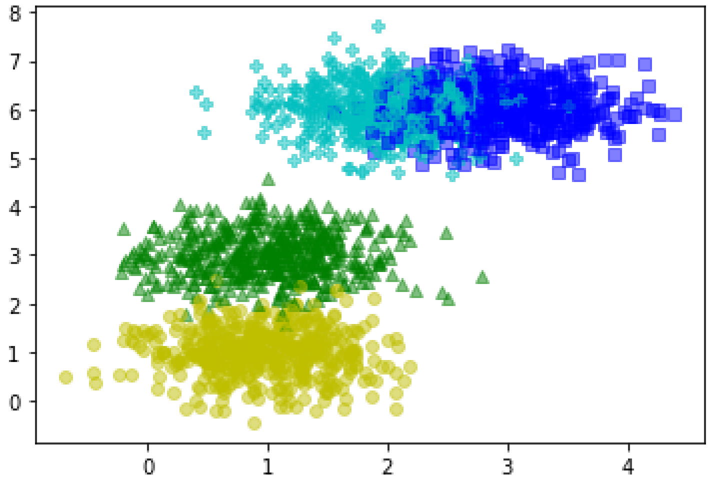
In [9]:

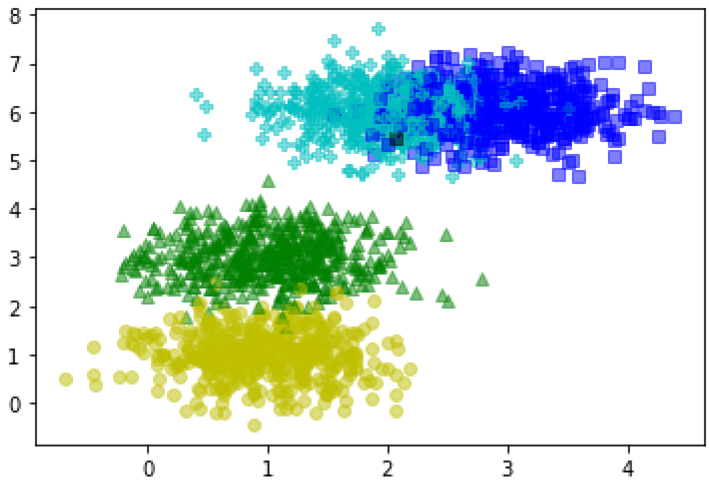
```
print("average-linkage hierarchical clustering:")
method = METHOD_APPLY[2]
cnt = 0
for k in range(2,6):
    print("Cluster Number = ",k)
    model = AgglomerativeClustering()
    model.fit(X,method)
    predict = model.label(k)
    plot_data(X,labels_true,predict,cnt)
    acc = accCalculate(labels_true,predict,k)
    cnt += 1
    print("Accuracy rate of clustering : ",acc,"%")
    print("-------Segmentation-------")
```

```
average-linkage hierarchical clustering:
Cluster Number =  2
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.25 %
-------Segmentation-------
Cluster Number =  3
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.4985 %
-------Segmentation-------
Cluster Number =  4
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.9965 %
-------Segmentation-------
Cluster Number =  5
calculate distance finish!
calculate cluster distance finish!
cluster finish !
Accuracy rate of clustering :  0.996 %
-------Segmentation-------
```









变换聚类簇的个数后重新进行聚类,得到如上十二张图,可以看到当聚类数目较小时,三种算法的正确率均较低,分析原因应当是由于聚类能力限制,分出的聚类个数有限,无法分出全部类别,因此性能较低.

而当算法的聚类簇个数变大时,最小连接距离法的性能变化不大,依旧很差,仅有50%左右,但最大连接距离法与平均连接距离法性能则会逐渐提高,直到设置的聚类簇的个数达到实际聚类个数时,正确率已经接近100%;但当个数超过实际聚类簇时,正确率又会开始降低,分析原因可能是由于聚类簇个数设置过高后,原本属于同一聚类的点可能会被划分成更细致的多个聚类簇,因此打上了不同的标签,检测到的正确率降低.