Data Mining Learning Report



姓名： 祁高翔

班级： 191141

学号： 20141004059

学院： 计算机学院

专业：计算机科学与技术

Lab 1 3

Project1:Numeric Data Analysis 3

Lab2 10

Project:Density Based Clustering 10

DENCLUE算法思想及步骤 10

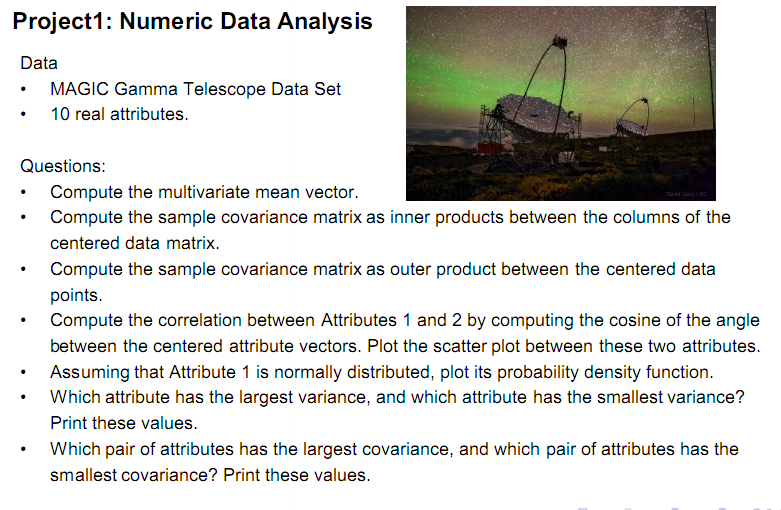
Python代码和运行结果 10

Lab3 12

Project:Decision Tree 12

算法思想 12

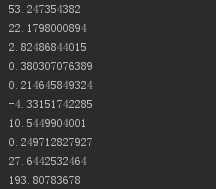
Python代码和运行结果 12



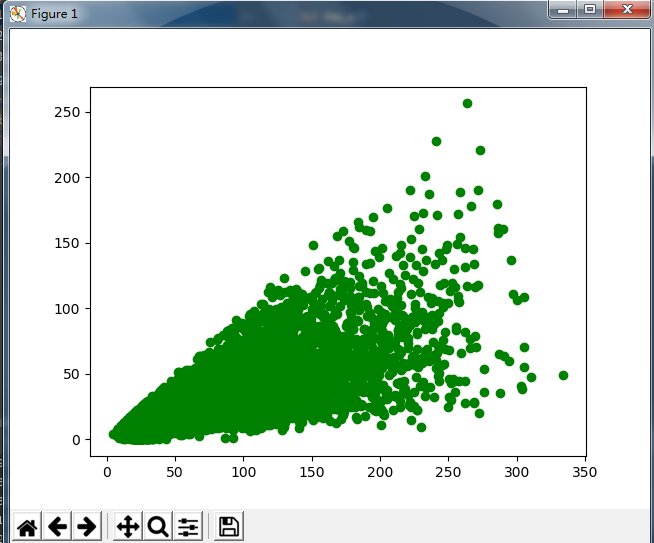
1. 实验1的python代码如下
2. # coding:utf-8  
   import numpy as np  
   from matplotlib import pyplot  
     
   attribute = [([] \* 10) for i in range(10)]  
   filename = 'magic04.txt'  
     
     
   def read\_file():  
    count = 0  
    with open(filename, 'r') as file\_to\_read:  
    while True:  
    lines = file\_to\_read.readline()  
    count += 1  
    if not lines:  
    break  
    pass  
    temp = lines.split(",")  
    for i in range(len(attribute)):  
    attribute[i].append(float(temp[i]))  
    pass  
    pass  
    for Attr in attribute:  
    print sum(Attr)/count  
    pass  
     
     
   def normal\_fun(x, mu, sigma):  
    pdf = np.exp(-((x - mu) \*\* 2) / (2 \* sigma \*\* 2)) / (sigma \* np.sqrt(2 \* np.pi))  
    return pdf  
    pass  
     
     
   def fun\_a():  
    print np.cov(attribute)  
    a = np.array(attribute[0])  
    b = np.array(attribute[1])  
    pyplot.plot(a, b, 'bo')  
    pyplot.show()  
     
    data = a  
    mean = data.mean()  
    std = data.std()  
    x = np.arange(-200, 200, 0.1)  
    y = normal\_fun(x, mean, std)  
    pyplot.plot(x, y)  
    pyplot.hist(data, bins=10, rwidth=0.9, normed=True)  
    pyplot.title('Data distribution')  
    pyplot.xlabel('Data')  
    pyplot.ylabel('Probability')  
    pyplot.show()  
    pass  
     
     
   def fun\_variance(a):  
    array = np.array(a)  
    return array.var()  
    pass  
     
     
   def fun\_covariance(a):  
    array = np.array(a)  
    covariance = np.cov(array)  
    return covariance  
     
     
   def fun\_print\_variance():  
    for j in range(len(attribute)):  
    print fun\_variance(attribute[j])  
    print "---------------------------------"  
    pass  
     
     
   def fun\_print\_covariance():  
    for j in range(len(attribute)):  
    print fun\_covariance(attribute[j])  
    pass  
     
     
   if \_\_name\_\_ == '\_\_main\_\_':  
    read\_file()  
    fun\_a()  
    fun\_print\_variance()  
    fun\_print\_covariance()

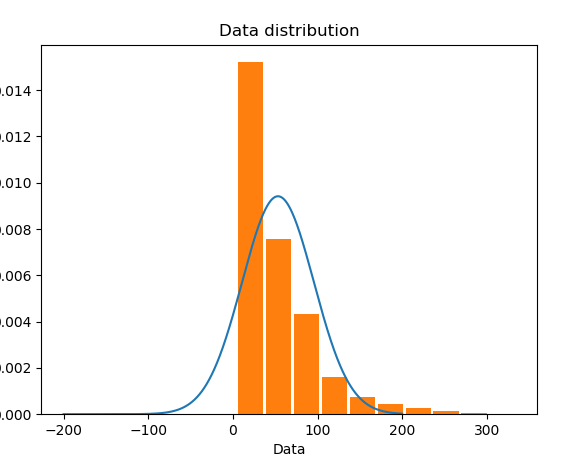
read\_file函数将文件magic04.txt读到二维矩阵arrtibute里面

并打印多元平均向量的值：结果为：



Fun\_a分别画出了散点图和正太分布图形，结果如图：

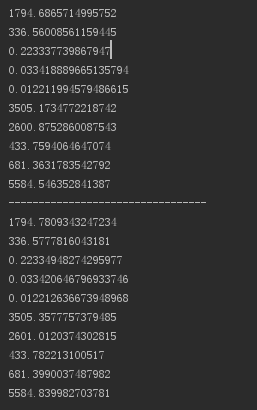


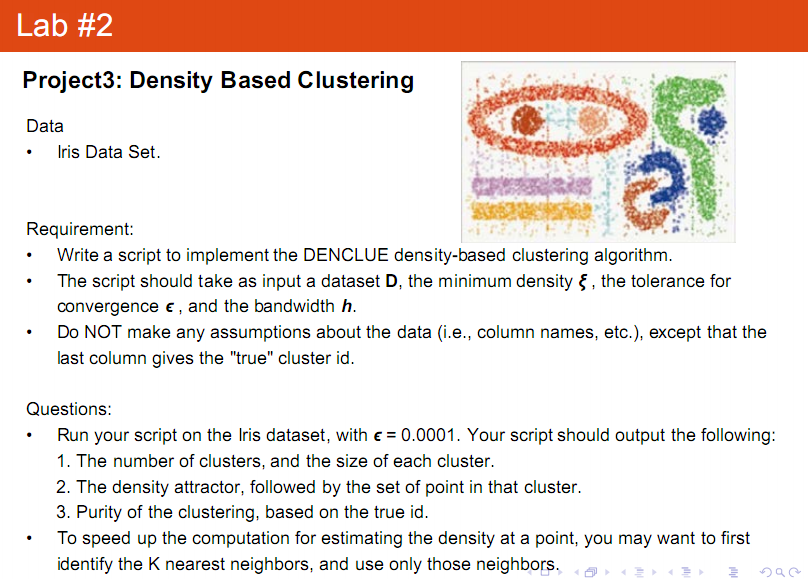


fun\_print\_variance()

fun\_print\_covariance()

这两个函数实现了方差和协方差：





### DENCLUE算法思想及步骤

基本思想：使用核密度函数用个体数据对象影响之和对点集总密度建模，尽管DENCLUE本质上不是基于网格的技术，但是它使用基于网格的方法提高性能。DENCLUE也是一种基于密度的方法。结果总密度函数具有局部尖峰（称作局部吸引点），并且这些局部尖峰用来以自然的方式定义簇。集体的说，对于每个数据点，一个爬山过程找出与该点相关联的最近的尖峰，并且与一个特定的尖峰相关联的所有数据称为一个簇。

DENCLUE算法步骤：

（1）对数据点占据的空间推导密度函数；

（2）识别局部最大点（这是局部吸引点）；

（3）通过沿密度增长最大的方向移动，将每个点关联到一个密度吸引点；

（4）定义与特定的密度吸引点相关联的点构成的簇；

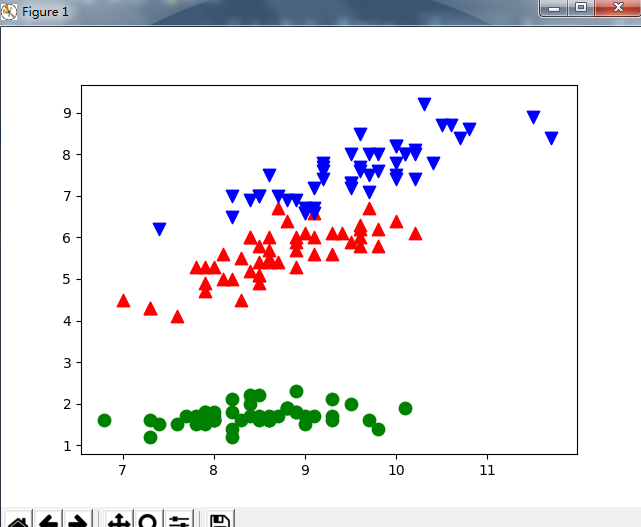
（5）丢弃密度吸引点的密度小于用户指定阈值的簇；

（6）合并通过密度大于或等于的点路径连接的簇。

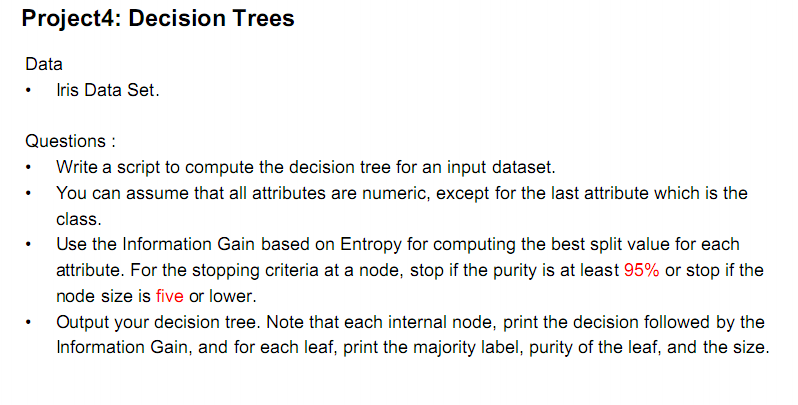
### Python代码和运行结果

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import DBSCAN  
from sklearn import datasets  
iris = datasets.load\_iris()  
data = iris.data  
  
  
def fun\_denclude(data, eps=0.3, min\_samples=10):  
 db = DBSCAN(eps=eps, min\_samples=min\_samples).fit(data)  
 core\_sample\_mask = np.zeros\_like(db.labels\_, dtype=bool)  
 core\_sample\_mask[db.core\_sample\_indices\_] = True  
 cluster\_labels = iris.target  
 unique\_cluster\_labels = set(cluster\_labels)  
 colors = ['green', 'red', 'blue', 'black', 'gray', '#ff00ff', '#ffff00']  
 markers = ['o', '^', 'v', '\*', 'x', 'h', 'd']  
 for i, cluster in enumerate(unique\_cluster\_labels):  
 cluster\_index = (cluster\_labels == cluster)  
 core\_samples = data[cluster\_index & core\_sample\_mask]  
 plt.scatter(core\_samples[:, 0] + core\_samples[:, 1], core\_samples[:, 2] + core\_samples[:, 3],c=colors[i],  
 marker=markers[i], s=80)  
 noise\_samples = data[cluster\_index & ~core\_sample\_mask]  
 plt.scatter(noise\_samples[:, 0] + noise\_samples[:, 1],noise\_samples[:, 2] + noise\_samples[:, 3], c=colors[i],  
 marker=markers[i], s=26)  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 fun\_denclude(data, 10, 10)

fun\_denclude函数实现该算法,最后展示的图为:



Lab 3



### 算法思想

决策树(Decision Tree)是在已知各种情况发生概率的[基础](https://baike.baidu.com/item/%E5%9F%BA%E7%A1%80/32794)上，通过构成决策树来求取净现值的[期望](https://baike.baidu.com/item/%E6%9C%9F%E6%9C%9B/35704)值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。

### Python代码和运行结果

import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn import tree  
from sklearn.externals.six import StringIO  
import pydotplus  
  
  
def fun\_draw\_tree():  
 iris = load\_iris()  
 test\_idx = [0, 50, 100]  
 train\_target = np.delete(iris.target, test\_idx)  
 train\_data = np.delete(iris.data, test\_idx, axis=0)  
  
 test\_target = iris.target[test\_idx]  
 test\_data = iris.data[test\_idx]  
  
 clf = tree.DecisionTreeClassifier()  
 clf = clf.fit(train\_data, train\_target)  
 tree.export\_graphviz(clf, out\_file='tree.dot')  
 dot\_data = StringIO()  
 tree.export\_graphviz(clf, out\_file=dot\_data, feature\_names=iris.feature\_names, class\_names=iris.target\_names,  
 filled=True, rounded=True,  
 special\_characters=True)  
 graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
  
 print(graph)  
 graph.write\_pdf("iris.pdf")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # fun\_make\_point()  
 fun\_draw\_tree()

形成的数据点为：

digraph Tree {  
node [shape=box] ;  
0 [label="X[3] <= 0.8\ngini = 0.667\nsamples = 147\nvalue = [49, 49, 49]"] ;  
1 [label="gini = 0.0\nsamples = 49\nvalue = [49, 0, 0]"] ;  
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;  
2 [label="X[3] <= 1.75\ngini = 0.5\nsamples = 98\nvalue = [0, 49, 49]"] ;  
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;  
3 [label="X[2] <= 4.95\ngini = 0.171\nsamples = 53\nvalue = [0, 48, 5]"] ;  
2 -> 3 ;  
4 [label="X[3] <= 1.65\ngini = 0.042\nsamples = 47\nvalue = [0, 46, 1]"] ;  
3 -> 4 ;  
5 [label="gini = 0.0\nsamples = 46\nvalue = [0, 46, 0]"] ;  
4 -> 5 ;  
6 [label="gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]"] ;  
4 -> 6 ;  
7 [label="X[3] <= 1.55\ngini = 0.444\nsamples = 6\nvalue = [0, 2, 4]"] ;  
3 -> 7 ;  
8 [label="gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]"] ;  
7 -> 8 ;  
9 [label="X[0] <= 6.95\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]"] ;  
7 -> 9 ;  
10 [label="gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]"] ;  
9 -> 10 ;  
11 [label="gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]"] ;  
9 -> 11 ;  
12 [label="X[2] <= 4.85\ngini = 0.043\nsamples = 45\nvalue = [0, 1, 44]"] ;  
2 -> 12 ;  
13 [label="X[0] <= 5.95\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]"] ;  
12 -> 13 ;  
14 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]"] ;  
13 -> 14 ;  
15 [label="gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]"] ;  
13 -> 15 ;  
16 [label="gini = 0.0\nsamples = 42\nvalue = [0, 0, 42]"] ;  
12 -> 16 ;  
}

github代码地址: <https://github.com/Gaoxiangqi/Dataming/>