Density based clustering algorithm with Python

A Report

Submitted for the Requirements of the Course

Big Data Analytics

by

Palavarapu Srikar
(1600290C203)

Department of Computer Science and Engineering
BML Munjal University
April 2019

INDEX

Introduction:

Clustering is the group of a particular set of objects based on their characteristics and aggregates them according to their similarities. Clustering is considered the most important unsupervised learning. It works with finding a structure in a collection of unlabeled data. A cluster is a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. For example, we can be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). Possible applications for clustering algorithms can be use to find groups of customers with similar behavior, classify plants based on the given features, and document classification. Clustering algorithms are scalable, can deal with different types of attributes, discovery clusters with arbitrary shape, deal with with noise and outliers and are high dimensional.

Dataset Information:

In this experiment we use well known Iris dataset,
This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2,This is an exceedingly simple dataset.

Attribute Information of Iris database:
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Density based cluster algorithm and code:

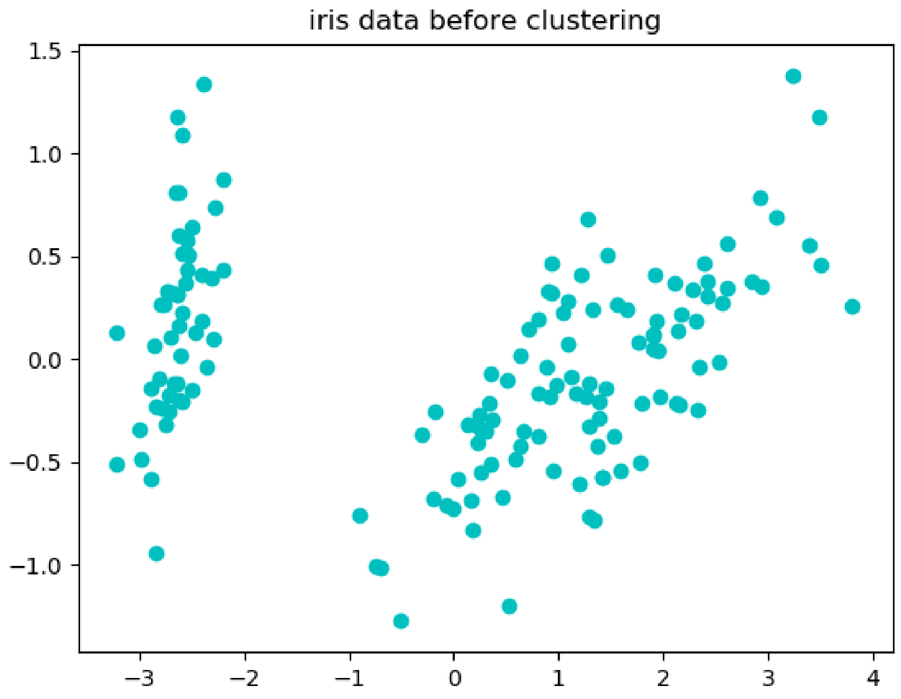
Density based clustering which typically for each data point in a cluster, at least a minimum number of points must exist within a given radius. The method was designed for spatial databases but can be used in other applications. It requires two input parameters: the size of the neighborhood(R) and the minimum points in the neighborhood (N). Essentially these two parameters determine the density within points in the clusters the user is willing to accept since they specify how many points must be in a region.density-based definition of a cluster, it is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes. Thus, Density based clustering can find many clusters that could not be found using K-means. Density based clustering has some challenges when clusters have widely varying densities, high-dimensional data because density if more difficult to define and can be expensive to compute the nearest neighbors.

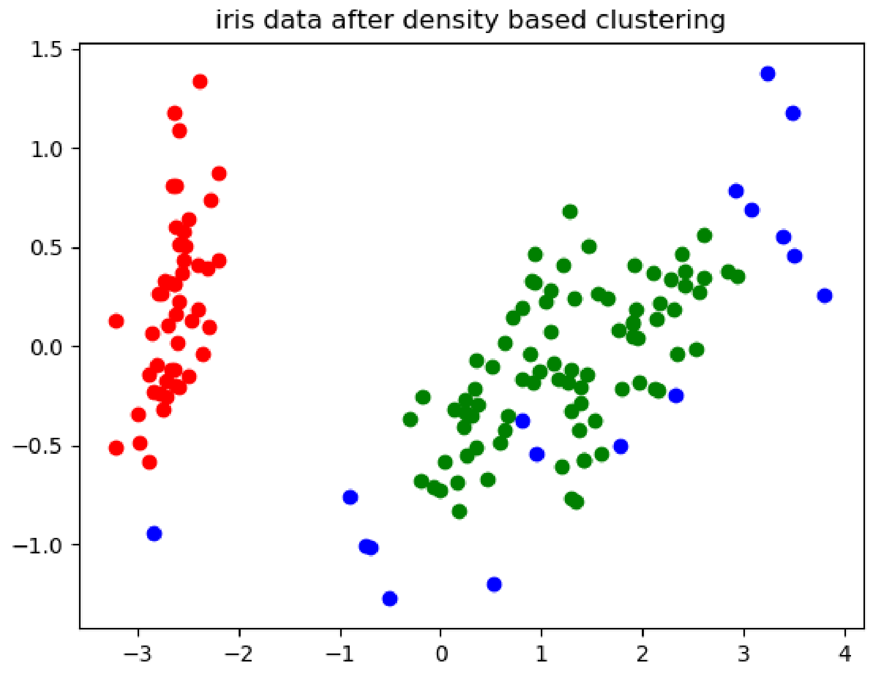
Python Code:

```
import math
import numpy
from pylab import *


def findneighbors(point, r):
    neighborpoints = []
    for pt in data:
        if math.sqrt(math.pow((point[0] - pt[0]), 2) + math.pow((point[1] - pt[1]), 2)) < r:
            neighborpoints.append(pt)
    return neighborpoints


def expandcluster(point, neighborpoints, c, r, n):
    point[3] = c
    for pt in neighborpoints:
        if pt[2] == 0:
            pt[2] = 1
            neighborpointsnext = findneighbors(pt, r)
            if len(neighborpointsnext) >= n:
                neighborpoints.extend(neighborpointsnext)
            if pt[3] == 0:
                pt[3] = c
    return neighborpoints
```

```
def DBSCAN(r, n):
    c = 0;
    for point in data:
        if point[2] == 0:
            point[2] = 1
            neighborpoints = findneighbors(point, r)
            if len(neighborpoints) < n:
                point[3] = -1
            else:
                c = c + 1;
                expandcluster(point, neighborpoints, c, r, n)
                clusters.insert(c - 1, [x[0:2] for x in neighborpoints])


point = [[], [], []]
clusterColor = [{1: 'bo'}, {2: 'co'}, {3: 'ko'}, {4: 'ro'}]
with open('iris.csv', 'rb') as csvfile:
    csvdata = numpy.genfromtxt(csvfile)
data = [x[0:2].tolist() for x in csvdata]

clusterX = [x[0] for x in data]
clusterY = [y[1] for y in data]
print(numpy.random.rand(3, 1))

plot(clusterX, clusterY,c=numpy.random.rand(3, 1))
show()

for d in data:
    d.append(0)
    d.append(0)
clusters = [[]]
DBSCAN(3, 2)

for cluster in clusters:
    clusterX = [x[0] for x in cluster]
    clusterY = [y[1] for y in cluster]
    clr = numpy.random.rand(3, 1)
    plot(clusterX, clusterY, marker='o', lw=0, c=clr)
show()

#from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA
iris = load_iris()
dbscan = DBSCAN()
dbscan.fit(iris.data)
pca = PCA(n_components=2).fit(iris.data)
pca_2d = pca.transform(iris.data)
for i in range(0, pca_2d.shape[0]):
    if dbscan.labels_[i] == 0:
        c1 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='r')
    elif dbscan.labels_[i] == 1:
        c2 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='g')
    else:
        c2 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='b')
plt.title('Density based cluster plot')
plt.show()
```

Results:


iris data before clustering
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4


iris data after density based clustering
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4

Conclusion:

In this report, we talk about clustering algorithm with the help of density and their implementation on python from scratch with the help of sklearn dictionary and we can also use sklearn.cluster and DBSCAN which is a simple method among both.
Even though we used both the algorithms to create density based clustering graph, scratch method gave us results in less time where as dbscan library method took a bit more time.

References:

Introduction Machine learning-ethem alpaydin

Contributions:

S.No	**Name**	**Contributions**
1	Palavarapu Srikar	Coding and report