Open in app

Following

565K Followers

Understanding DBSCAN Algorithm and Implementation from Scratch

DBSCAN Algorithm Step by Step, Python Implementation, and Visualization.

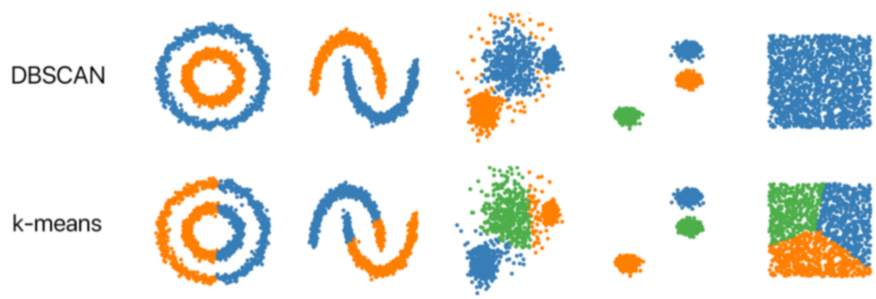
Andrewngai Jun 9, 2020 · 5 min read ★

What is DBSCAN

DBSCAN(Density-Based Spatial Clustering of Applications with Noise) is a commonly used unsupervised clustering algorithm proposed in 1996. Unlike the most well known K-mean, DBSCAN does not need to specify the number of clusters. It can automatically detect the number of clusters based on your input data and parameters. More importantly, DBSCAN can find arbitrary shape clusters that k-means are not able to find. For example, a cluster surrounded by a different cluster.

Click to add this story to a list.

Got it



DBSCAN vs K-means, credit

Also, DBSCAN can handle noise and outliers. All the outliers will be identified and marked without been classified into any cluster. Therefore, DBSCAN can also be used for Anomaly Detection (Outlier Detection)

Before we take a look at the preusdecode, we need to first understand some basic concepts and terms. Eps, Minpits, Directly density-reachable, density-reachable, density-connected, core point and border point

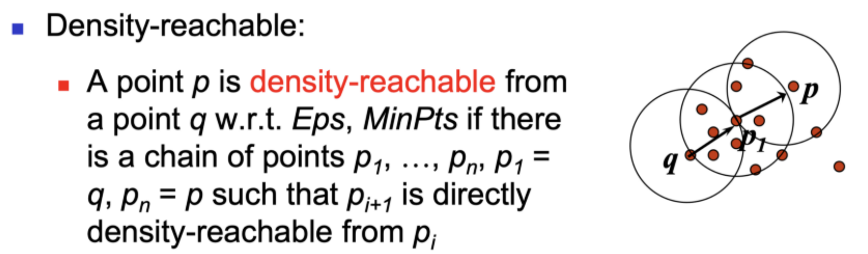
First of all, there are two parameters we need to set for DBSCAN, Eps, and MinPts.

Eps: Maximum radius of the neighborhood

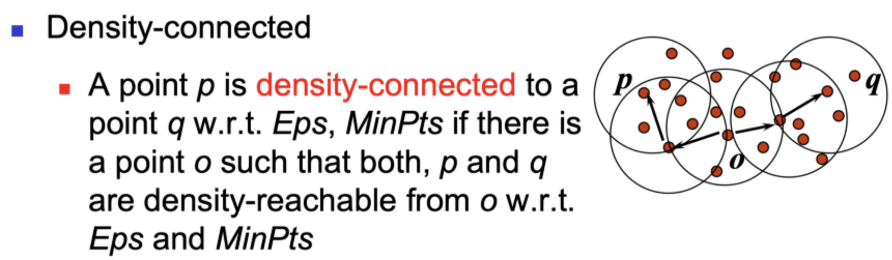
MinPts: Minimum number of points in an Eps-neighbourhood of that point

And there is the concept of **Directly density-reachable**: A point p is directly density reachable from a point q w.r.t. Eps, MinPts, if NEps (q): {p belongs to D | dist(p,q) ≤ Eps} and |N Eps (q)| ≥ MinPts. Let's take a look at an example with Minpts = 5, Eps =

Open in app

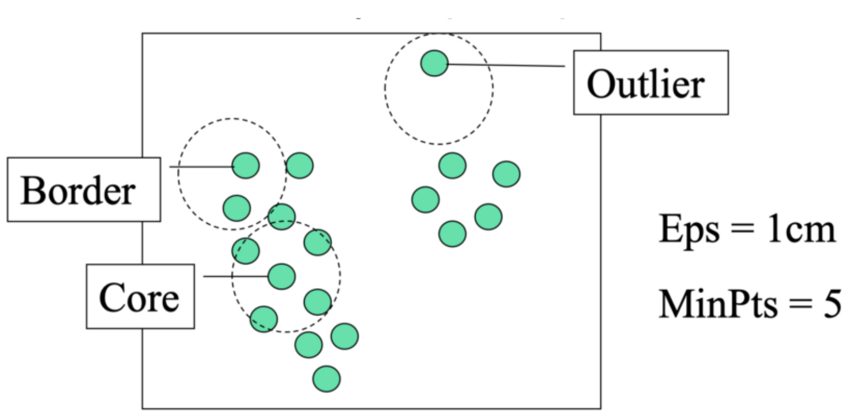


Density-reachable example



Density-connected example

Finally, a point is a core point if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster A. And a border point has fewer than MinPts within Eps, but is in the neighborhood of a core point. We can also define the outlier(noise) point, which is the points that are neither core nor border points.



Core point, Border point, Outlier Point examples

Now, let's take a look at how DBSCAN algorithm actually works. Here is the preusdecode.

1. Arbitrary select a point p
2. Retrieve all points density-reachable from p based on Eps and MinPts
3. If p is a core point, a cluster is formed
4. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database

Open in app

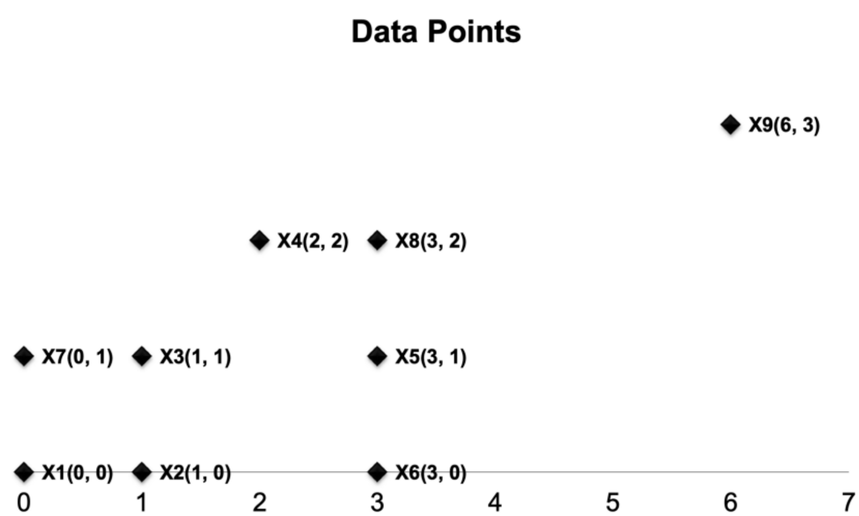
If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects. Otherwise, the complexity is $O(n^2)$

Example

Consider the following 9 two-dimensional data points:

x1(0,0), x2(1,0), x3(1,1), x4(2,2), x5(3,1), x6(3,0), x7(0,1), x8(3,2), x9(6,3)

Use the Euclidean Distance with Eps =1 and MinPts = 3. Find all core points, border points and noise points, and show the final clusters using DBCSAN algorithm. Let's show the result step by step.



Example Data Visuilization

First, Calculate the N(p), Eps-neighborhood of point p

N(x1) = {x1, x2, x7}

N(x2) = {x2, x1, x3}

N(x3) = {x3, x2, x7}

N(x4) = {x4, x8}

N(x5) = {x5, x6, x8}

N(x6) = {x6, x5}

N(x7) = {x7, x1, x3}

N(x8) = {x8, x4, x5}

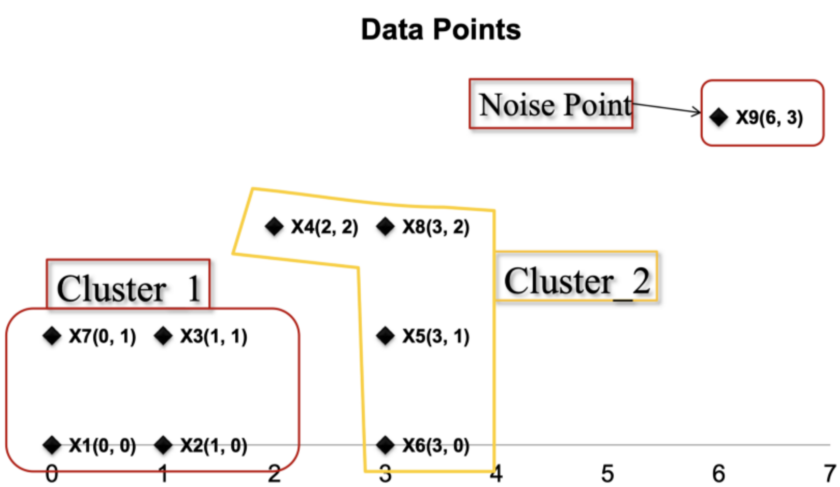
N(x9) = {x9}

If the size of N(p) is at least MinPts, then p is said to be a core point. Here the given MinPts is 3, thus the size of N(p) is at least 3. **Thus core points are:{x1, x2, x3, x5, x7, x8}**

N(x6) = {x6, x5}. here x8 and x5 are core points, So both **x4 and x6 are border points**. Obviously, the point left, **x9 is a noise point.**

Now, let's follow the preusdecode to produce the clusters.

1. Arbitrary select a point p, now we choose x1
2. Retrieve all points density-reachable from x1: {x2, x3, x7}
3. Here x1 is a core point, a cluster is formed. So we have Cluster_1: {x1, x2, x3, x7}
4. Next, we choose x5, Retrieve all points density-reachable from x5: {x8, x4, x6}
5. Here x5 is a core point, a cluster is formed.So we have Cluster_2: {x5, x4, x8, x6}
6. Next, we choose x9, x9 is a noise point, noise points do NOT belong to any clusters.
7. Thus the algorithm stops here.



Final DBSCAN Cluster Result

Python Implementation

Here is some sample code to build FP-tree from scratch and find all frequency itemsets in Python 3. I have also added visualization of the points and marked all outliers in blue.

```
import numpy as np
import collections
import matplotlib.pyplot as plt
import queue
import scipy.io as spio

#Define label for differnt point group
NOISE = 0
UNASSIGNED = 0
core=-1
edge=-2



#function to find all neigbor points in radius
def neighbor_points(data, pointId, radius):
    points = []
    for i in range(len(data)):
```

Open in app

```
    return points

#DB Scan algorithom
def dbscan(data, Eps, MinPt):
    #initilize all pointlable to unassign
    pointlabel  = [UNASSIGNED] * len(data)
    pointcount = []
    #initilize list for core/noncore point
    corepoint=[]
    noncore=[]

    #Find all neigbor for all point
    for i in range(len(data)):
        pointcount.append(neighbor_points(train,i,Eps))

    #Find all core point, edgepoint and noise
    for i in range(len(pointcount)):
        if (len(pointcount[i])>=MinPt):
            pointlabel[i]=core
            corepoint.append(i)
        else:
            noncore.append(i)

    for i in noncore:
        for j in pointcount[i]:
            if j in corepoint:
                pointlabel[i]=edge

                break

    #start assigning point to luster
    cl = 1
    #Using a Queue to put all neigbor core point in queue and find neigboir's neigbor
    for i in range(len(pointlabel)):
        q = queue.Queue()
        if (pointlabel[i] == core):
            pointlabel[i] = cl
            for x in pointcount[i]:
                if(pointlabel[x]==core):
                    q.put(x)
                    pointlabel[x]=cl
                elif(pointlabel[x]==edge):
                    pointlabel[x]=cl
            #Stop when all point in Queue has been checked
            while not q.empty():
                neighbors = pointcount[q.get()]
                for y in neighbors:
                    if (pointlabel[y]==core):
                        pointlabel[y]=cl
                        q.put(y)
                    if (pointlabel[y]==edge):
                        pointlabel[y]=cl
            cl=cl+1 #move to next cluster

    return pointlabel,cl

#Function to plot final result
def plotRes(data, clusterRes, clusterNum):
    nPoints = len(data)
    scatterColors = ['black', 'green', 'brown', 'red', 'purple', 'orange', 'yellow']
    for i in range(clusterNum):
        if (i==0):
            #Plot all noise point as blue
            color='blue'
```

Open in app

```
        for j in range(nPoints):
            if clusterRes[j] == i:
                x1.append(data[j, 0])
                y1.append(data[j, 1])
        plt.scatter(x1, y1, c=color, alpha=1, marker='.')


#Load Data
raw = spio.loadmat('DBSCAN.mat')
train = raw['Points']

#Set EPS and Minpoint
epss = [5,10]
minptss = [5,10]
# Find ALl cluster, outliers in different setting and print resultsw
for eps in epss:
    for minpts in minptss:
        print('Set eps = ' +str(eps)+ ', Minpoints = '+str(minpts))
        pointlabel,cl = dbscan(train,eps,minpts)
        plotRes(train, pointlabel, cl)
        plt.show()
        print('number of cluster found: ' + str(cl-1))
        counter=collections.Counter(pointlabel)
        print(counter)
        outliers  = pointlabel.count(0)
        print('numbrer of outliers found: '+str(outliers) +'\n')
```

DBSCAN.py hosted with ♡ by GitHub view raw

Thanks for reading and I am looking forward to hearing your questions and thoughts. If you want to learn more about Data Science and Cloud Computing, you can find me on Linkedin.

Photo by Alfons Morales on Unsplash

Reference

https://github.com/NSHipster/DBSCAN

Open in app

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter

Emails will be sent to xili9@mail.yu.edu.
Not you?

Dbscan Data Mining Clustering Python Outlier Detection

About Write Help Legal

Get the Medium app

Download on the App Store GET IT ON Google Play