International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

Int. J. Sci. Res. Arch.
IJSRA
International Journal of Science and Research Archive
Research Journal Archive, INDIA

(Research Article)

Ideal/balance point clustering algorithm

Sathya Narayanan *

Research enthusiast, Chennai, India.

International Journal of Science and Research Archive, 2023, 10(02), 618–625

Publication history: Received on 25 October 2023; revised on 01 December 2023; accepted on 04 December 2023

Article DOI: https://doi.org/10.30574/ijsra.2023.10.2.1006

Abstract

Generally clustering is one of the largest used ML techniques to group data points of similar type together. Mostly while grouping data points based upon few popular traditional clustering algorithms like KNN, K-means etc. only the magnitudes of those data points are considered. The deviation of those data points from a particular point said to be the center point or the balance point of the entire dataset is not considered that much, but Here in this Ideal Balance Point clustering algorithm the degree of deviation of all the data points from the balance point is primarily considered. By using such a methodology, we may be able to form clusters of data points based on their degrees of deviation from the Ideal Balance Point of the entire dataset. The Ideal Balance Point of the dataset is actually the centroid of the plane on which the datapoints are distributed. Here in the Ideal Balance Point clustering algorithm the user may also manually feed-in the Ideal Balance Point for the dataset based on the requirement or as per the practical idealness of the circumstance and nature of the dataset. Generally, this idea is derived from the theory that an object has all its' mass equally distributed on the centroid. Hence the plane formed by the distribution of the data points is considered as an object and the centroid of it is assumed to be the point where the entire data points are directed into, but the ideal balance point for the data may also differ as per real life scenario, thus the feature of feeding in the balance point is also given in this Ideal Balance Point clustering algorithm.

Keywords: Clustering Algorithm; Balance Point; Ideal Point; non-hierarchical clustering; Degree of Deviation from Ideal/Balance point

1. Introduction

Clustering algorithms are normally used in various sectors of machine learning. A few real-life use case applications are customer segregation, medical imaging, social media analysis, friend suggestions in social media etc...Mostly all the clustering algorithms consider to use a common methodology to achieve the output clusters that is by grouping data points of same category together. A base point is fixed as per various assumptions and logics and keeping that particular point as base the clusters are formed. Here in this Balance point clustering algorithm the base point (balance point) is calculated based on the provided data set itself. One more important feature of this algorithm is that the ideal number of clusters are decided by the computer itself. The end user need not even specify the no. of clusters he/she needs. The need for specifying the seed point is also mitigated since any changes in seed point does not affect the output.

2. Explanation of the methodology and logic

The basic logic behind the algorithm is at first we find the centroid of the given data points using the formula of centroid. The centroid is known to be the balance point as per the terminology of the Balance point algorithm. The formula to calculate it is as follows

For simplicity let us assume that there are only 2 coordinates in the set of data points given then

* Corresponding author: Vaishali Akshitha

Balance point = (a1 + a2 + ... + aN/N, b1 + b2 + ... + bN/N)

Where a1,a2, ..., aN and b1,b2, ..., bN => coordinates of particular features in the set of data points given. N => total no. of data points in the entire dataset.

The balance point which is assumed to be the ideal point or the coordinates under ideal condition may also be manually fed in by the user. If this is done then,

Balance point = the coordinates of idealness fed in by the user

After the calculation of the balance point we find the distance between the balance point and every single data point given in the dataset. As a next step all the data points are arranged in the ascending order based on their distance from the balance point (that is the data point which is least distant to the balance point is kept at first and the most distant data point from the balance point is kept at last).

As the next step the distance between the second data point and the first and third data points are calculated. If the distance between the first and second data point is less than the distance between the second and third data point then the second data point is left to be in the group of first data point. If the distance between the second and third data point is less than distance between first and second data points then the first data point (also the data points preceding it if any) is left to be a cluster and the second and third data points along with the other data points following them is formed to be a group or a cluster. In other words, the data points which are located more closer to each other are set to be in a cluster and the data points located far away are set to be in another cluster.

The above-mentioned process is started from the second data point which is obtained after arranging all data points in ascending order based on their distances from the balance point. The first data point is set to be in the first cluster by default so it is skipped since the cluster this data points is going to be present is decided by default. Now the above-mentioned step of cluster breaking is carried out from the second to the second last data point obtained after the ascending order arrangement. The last data point is added to the last cluster itself.

Finally the clusters are acquired after the breaking of the initially formed single cluster (ascending order arrangement of data points based on their distances from the balance point).

These clusters are formed in such a way that data points having a similar range of deviation from the balance point and very close inter-distances between them are packed into a single cluster. By doing this it is ensured that the data points are grouped in an ideal manner and the ideal no. of clusters is obtained, so the no. of clusters also need not be specified by the user as mentioned before.

2.1. Explanation with example

To get more clarity on the above logic it will be better if a demonstration of the algorithm with an example is done. Consider 7 data points with 2 features as (15,45), (34,56), (7,12), (111,11), (9,999), (65,64), (1,1). Provided that there is no coordinates of ideal situation provided by the user for balance point then the balance point is calculated from the given dataset itself internally by the algorithm.

Step 1 --> Calculate the balance point using the above mentioned formula such as :-

Balance point = (15+34+7+111+9+65+1/7, 45+56+12+11+999+64+1/7).

Balance point = (34.57, 169.71).

Step 2 --> Calculate distances of each data point from the balance point.

Distances of the data points from the balance point are :-

i) for (15,45) => 126.23
ii) for (34,56) => 113.71
iii) for (7,12) => 160.10
iv) for (111,11) => 179.15

v) for (9,999) => 829.68
vi) for (65,64) => 110
vii) for (1,1) => 172.01

Step 3 --> Sort the data points in ascending order based on their distances.

P1) 110 ----> (65,64)
P2) 113.71 ----> (34,56)
P3) 126.23 ----> (15,45)
P4) 160.10 ----> (7,12)
P5) 172.01 ----> (1,1)
P6) 179.15 ----> (111,11)
P7) 829.68 ----> (9,999)

Step 4 --> Difference(P2,P1) < Difference(P2,P3), so P2 is allowed to be in same cluster along with P1.

Next Difference(P3,P2) < Difference(P3,P4). Hence P3 is also allowed to be in same cluster along with P2. (Note that P1 is already present in this cluster).

Now as next step Difference(P4,P3) > Difference(P4,P5) hence contradicting to the first 2 cases the points P4 and P5 which comes as the 2nd comparison set has a smaller difference, so here after point P3 a new cluster starting from P4 is formed.

Moving to next as Difference(P5,P4) > Difference(P5,P6) so here once again a new cluster starting from P5 is formed.

Similarly Difference(P6,P5) < Difference(P6,P7) so P6 is added into the cluster where P5 is already present.

Now since P7 is the last data point it is allowed to be present in the last cluster itself.

Note: The first data point obtained after the ascending order arrangement is to be present as the first point of the first cluster by default as per the algorithm itself.

The final clusters are

- Cluster 1) (65,64), (34,56), (15,45)
- Cluster 2) (7,12)
- Cluster 3) (1,1), (111,11), (9,999).

The ideal no. of clusters are grouped by the Balance point algorithm itself without any human intervention as mentioned above.

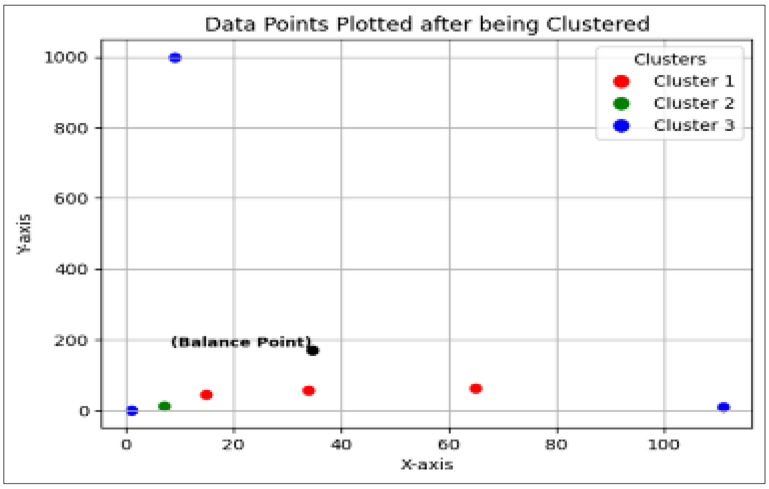


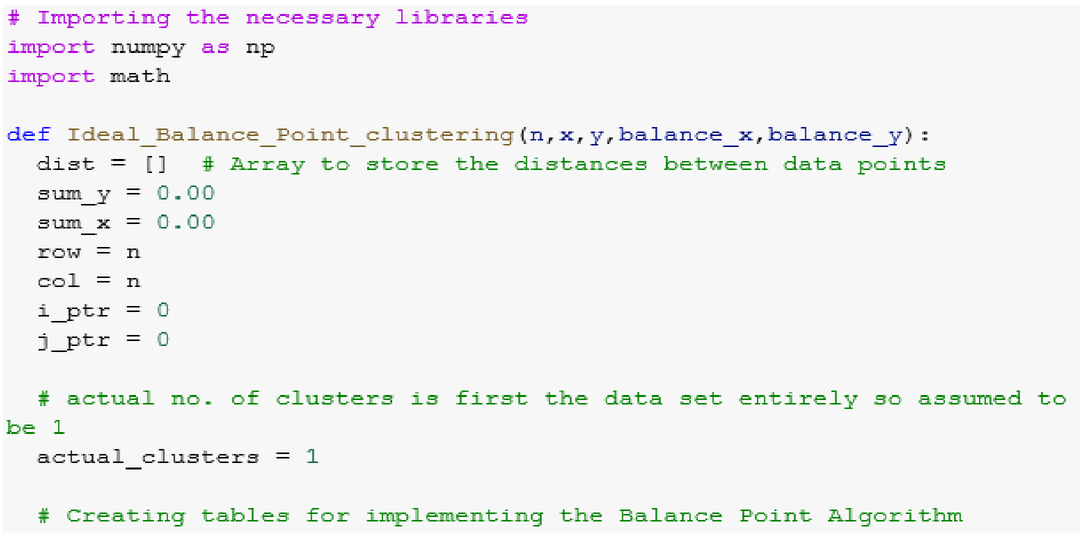
Figure 1 Graph representing the data points of the above example and their respective clusters

The above given figure shows the graph of the data points that we considered initially. It also shows the clusters obtained after clustering those points based on the Balance Point Algorithm.

The same process which was demonstrated above will be used in finding the clusters of very large sized datasets too. The no. of features may also be either increased or decreased as per the dataset but still the process would be the same as above. The balance point will have same no. of coordinates as no. of features in the dataset. Similar steps will be followed to find the distances of each data point from the balance point too.

2.2. Source Code

The Source code for the above-mentioned Balance Point clustering Algorithm is written in Python language. The code shown in this section is a simpler version for the purpose of understanding the flow and working of the balance point Algorithm, the same code can be written in backend with a way to accept any no. of arguments and operate on them by using the *args and *kwargs logic of Python. The implementation of the code is as follows: -

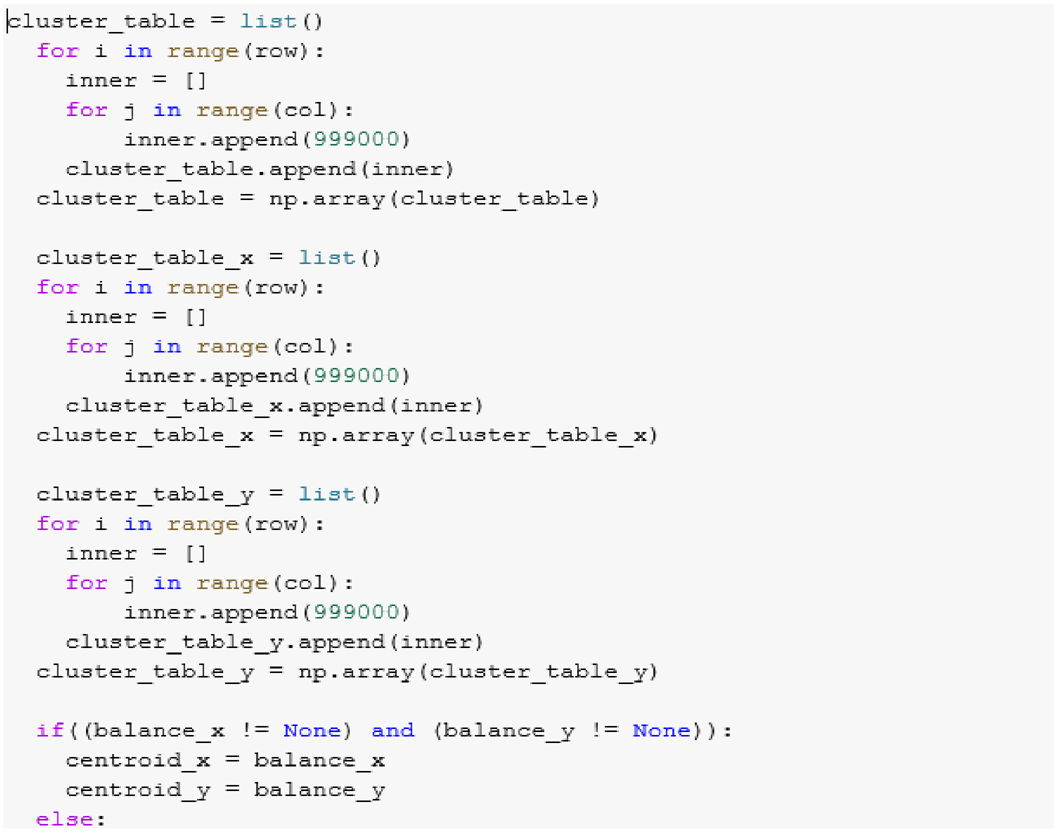


```
# Importing the necessary libraries
import numpy as np
import math

def Ideal_Balance_Point_clustering(n,x,y,balance_x,balance_y):
  dist = []  # Array to store the distances between data points
  sum_y = 0.00
  sum_x = 0.00
  row = n
  col = n
  i_ptr = 0
  j_ptr = 0

  # actual no. of clusters is first the data set entirely so assumed to
be 1
  actual_clusters = 1

  # Creating tables for implementing the Balance Point Algorithm
```



```
cluster_table = list()
  for i in range(row):
    inner = []
    for j in range(col):
        inner.append(999000)
    cluster_table.append(inner)
  cluster_table = np.array(cluster_table)

  cluster_table_x = list()
  for i in range(row):
    inner = []
    for j in range(col):
        inner.append(999000)
    cluster_table_x.append(inner)
  cluster_table_x = np.array(cluster_table_x)

  cluster_table_y = list()
  for i in range(row):
    inner = []
    for j in range(col):
        inner.append(999000)
    cluster_table_y.append(inner)
  cluster_table_y = np.array(cluster_table_y)

  if((balance_x != None) and (balance_y != None)):
    centroid_x = balance_x
    centroid_y = balance_y
  else:
    sum_x = 0
    sum_y = 0

    for i in range(n):
      sum_x = sum_x + x[i]
      sum_y = sum_y + y[i]

    centroid_x = sum_x / n
    centroid_y = sum_y / n

  for i in range(n):
    dist1 = abs(math.pow(abs(centroid_x-x[i]),2) +
math.pow(abs(centroid_y-y[i]),2))
    dist1 = math.sqrt(dist1)
    dist.append(dist1)

  # Arranging all data point’s Distances and Features in ascending
order
  for i in range(n-1):
     for j in range(n-i-1):
```

```
       if dist[j] > dist[j+1]:

         t = dist[j]
         dist[j] = dist[j+1]
         dist[j+1] = t

         t_x = x[j]
         x[j] = x[j+1]
         x[j+1] = t_x

         t_y = y[j]
         y[j] = y[j+1]
         y[j+1] = t_y

  # Initializing the first data point at first by default
  cluster_table_x[0][0] = x[0]
  cluster_table_y[0][0] = y[0]
  cluster_table[0][0] = dist[0]

  # Breaking the single cluster of whole dataset into parts based upon
the logic explained
  for i in range(1,n-1):
   pre = abs(dist[i] - dist[i-1])
   post = abs(dist[i] - dist[i+1])
  if pre < post:

   if j_ptr < col:
     j_ptr += 1
     cluster_table[i_ptr][j_ptr] = dist[i]
     cluster_table_x[i_ptr][j_ptr] = x[i]
     cluster_table_y[i_ptr][j_ptr] = y[i]

   else:
     actual_clusters += 1
     i_ptr += 1
     j_ptr = 0
     cluster_table[i_ptr][j_ptr] = dist[i]
     cluster_table_x[i_ptr][j_ptr] = x[i]
     cluster_table_y[i_ptr][j_ptr] = y[i]


  if post <= pre:
     actual_clusters += 1
     i_ptr += 1
     j_ptr = 0
     cluster_table[i_ptr][j_ptr] = dist[i]
     cluster_table_x[i_ptr][j_ptr] = x[i]
     cluster_table_y[i_ptr][j_ptr] = y[i]
```

```
  if j_ptr < col:
    j_ptr = j_ptr + 1
    cluster_table[i_ptr][j_ptr] = dist[n-1]
    cluster_table_x[i_ptr][j_ptr] = x[n-1]
    cluster_table_y[i_ptr][j_ptr] = y[n-1]
  else:
    i_ptr = i_ptr + 1
    cluster_table[i_ptr][j_ptr] = dist[n-1]
    cluster_table_x[i_ptr][j_ptr] = x[n-1]
    cluster_table_y[i_ptr][j_ptr] = y[n-1]

  # This code may be used to view the Output Clusters
  print("\tThe Clusters are:")
  for i in range(0,actual_clusters):
    print("\t\t\tCluster",i+1,": ")
    for j in range(0,col):
       if(cluster_table[i][j] != 999000):
          print("\t\t\t\t(",cluster_table_x[i][j],",",cluster_table_y[i
][j],")")
```

```
def main():
 balance_x = None
 balance_y = None

 # Feature decalration as per the above example
 x = [15,34,7,111,9,65,1]
 y = [45,56,12,11,999,64,1]

 no_of_samples = len(x)    # Total no. of data points (samples)
 n = no_of_samples

 # Ideal/Balance Point Clustering Algorithm implementation
 Ideal_Balance_Point_clustering(n,x,y,balance_x,balance_y)

if __name__ == "__main__":
    main()
```

3. Discussion

The basic idea behind the balance point clustering algorithm is that instead of taking the closeness or proximity in-between the data points, the level of deviation of the data points from a specific point said to be the balance point is taken. Traditional clustering algorithms work on the principal of clustering data points within close proximity but here the balance point clustering algorithm works on the principle of clustering data points having the similar deviation together. In other words the data points having a very close range of deviation from the balance point is clustered into a group. The practical use case of this algorithm which considers the degree of deviation can be done in the places where degrees of deviations of data points from a particular point play a major role. A real life example is food recommendations, where the degree of taste deviation from a particular point matters the most. Foods having similar tastes may be suggested to the users for purchasing since there is a high chance that the user might purchase it.

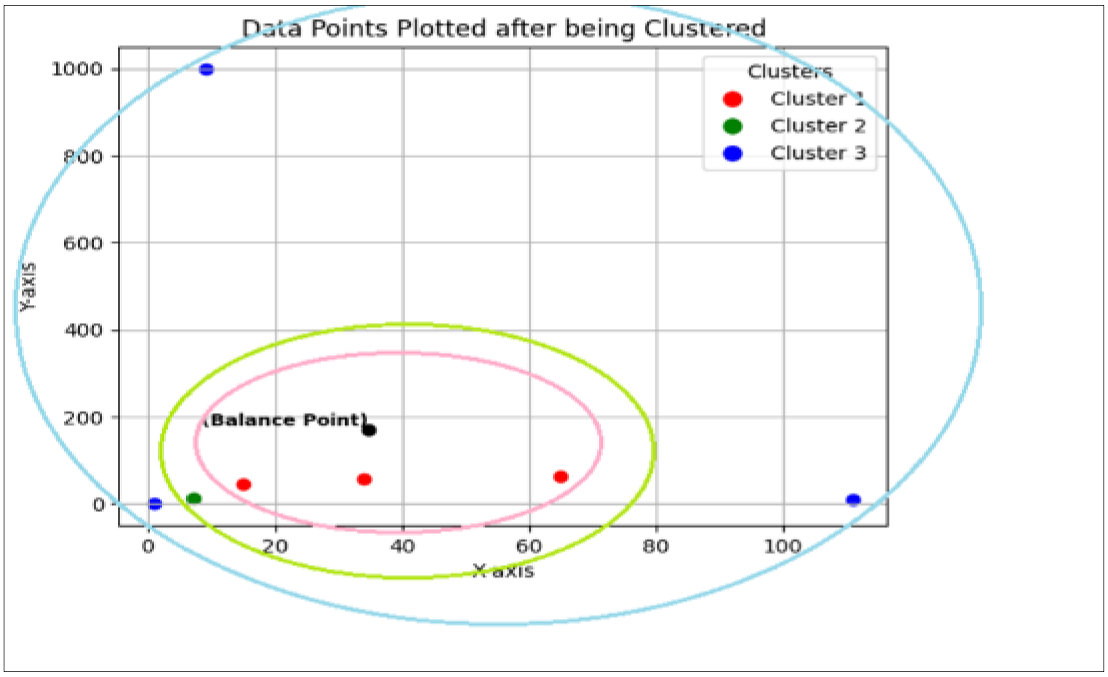


Figure 2 Graph representing the ranges of degrees of deviation(imaginary) of each clusters of the above eg.

The above graph represents the data points that was clustered in the example above and the circles in the graph indicate the level of deviation (imaginary). The data points falling under those level of deviation (if there were any) would be considered to be under the same cluster of those points already present in the cluster. To be more precise the pink circle denotes the range of cluster 1, green circle the range of cluster 2 and the blue circle the range of cluster 3. Since range of the 3rd cluster is too large if there were any data points between the range of deviation of cluster 2 and cluster 3 there might be a separate cluster formed only for those data points. Hence the data points having the similar level of deviation from the Balance point within the deviation range are clustered together. In other words, the data points with a same range of distance from the calculated or manually fed in balance point irrespective of their presence of direction is considered to be under the same cluster.

Thus the same logic implemented in this algorithm can be used for clustering large to very large datasets. The additional activity needs to be done is to create N no. of arrays and N dimensioned tables for performing the same activity upon datasets containing N no. of Features.

4. Conclusion

The Machine learning is a very rapidly used branch in the nowadays technology domains. There are a lot of new creativity and ideologies emerging in to this field and as a result of it there has been a lot of astonishing developments in this branch of computer technology. The balance point clustering algorithm which considers the degree of deviation from the balance point and also gives the user the convenience to either fix his own balance point or this algorithm also doesn't fail to calculate its' own balance point based on the dataset enabling the user to be free from the tediousness of fixing the perfect balance point. Hence this balance point algorithm also adds a feather to the crown of creativity in the ML domain. The Algorithm specified in this paper either uses the Ideal data point coordinates for clustering (Ideal Point clustering algorithm) or it also uses the balance point which is calculated on its' own (Balance Point clustering algorithm). The names may be referred as per the way of use it is used.

Compliance with ethical standard

Acknowledgement

The Author of this Paper, Sathya Narayanan is a research enthusiast who is very fond of contributing to the world of Technology by creating a lot of new innovative ideas and publishing them. He has a strong passion for Research and Technology. This Paper is also one such a publication by him and this is solely done for the purpose of adding one more ideology to the world of innovation. All these Techniques and methodologies are designed by his own also with reference to various documentations and blogs. Sathya loves to do research and create entirely new techniques to solve a problem on his own. The foremost thing taken care of while creating a new Algorithm is to be Novel at any cost and under any circumstance. Thus Sathya Narayanan the Author of this Research Paper is the only owner of this idea and methodology discussed in this paper. If anyone going to implement this into their innovation in any business point of view or in anyway that impacts the society on a large scale, there should be a credit for the Author, Sathya Narayanan who is the actual and lone creator of this above mentioned (Ideal/Balance Point Clustering Algorithm) Technique.

References

[1] https://www.geeksforgeeks.org/clustering-in-machine-learning/

[2] https://www.geeksforgeeks.org/k-means-clustering-introduction/

[3] https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

[4] https://medium.com/delvify/introduction-to-clustering-algorithms-and-its-use-cases-35c1655c91e7