Advanced Topics in Clustering

Manu Madhavan

Today

- Enhancing K-means clustering
- Spectral Clustering
- Biclustering

Issues with K-means

- Computational complexity of the original k-means algorithm is very high, especially for large data sets (*O(nkl)*)
- Results in different types of clusters depending on the random choice of initial centroids—> *Accuracy of the final clusters heavily depends on the selection of the initial centroids*

Enhanced K-means (K.A.A. Nazeer and M.P. Sebastian)

Algorithm 2 The Improved Clustering Algorithm

Input:

D = d1, d2,.....dn // set of n data items

k // Number of desired clusters

Output:

A set of k clusters

Steps:

1. Determine the initial centroids of the clusters by using Algorithm 3;
2. Assign the data points to the clusters by using Algorithm 4;

Algorithm 3 Finding the Initial Centroids

Input:

D = d1, d2,.....dn // set of n data items

k // Number of desired clusters

Output:

A set of k initial centroids

Steps:

1. Set m = 1;
2. Compute the distance between each data point and all other data points in the set D;
3. Find the closest pair of data points from the set D and form a data point set Am (1 <= m <= k) which contains these two data points. Delete these two data points from the set D;
4. Find the data point in D that is closest to the data point set Am. Add it to Am and delete it from D;
5. Repeat step 4 until the number of data points in Am reaches 0.75*(n/k);
6. If m < k, then m = m + 1. Find another pair of data points from D between which the distance is the shortest. Form another data point set Am and delete it from D. Go to Step 4;
7. For each data point set Am (1 <= m <= k) find the arithmetic mean of the vectors of data points in Am. These means will be the initial centroids.

Algorithm 4 Assigning data points to clusters

Input:

D = d1, d2,.....dn // set of n data items

C = c1, c2,.....ck // set of k centroids

Output:

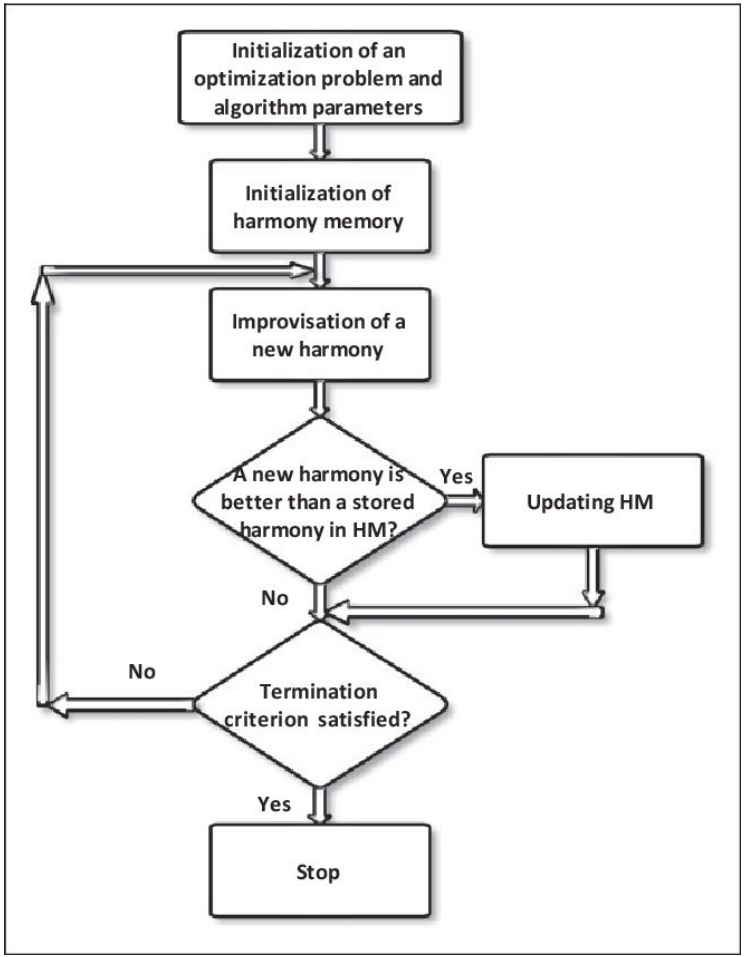
A set of k clusters

Steps:

1. Compute the distance of each data point di (1 <= i <= n) to all the centroids cj (1 <= j <= k) as d(di, cj);
2. For each data point di, find the closest centroid cj and assign di to cluster j.
3. Set ClusterId[i] = j; // j:Id of the closest cluster
4. Set NearestDist[i] = d(di, cj);
5. For each cluster j (1 <= j <= k), recalculate the centroids;
6. *Repeat*
7. For each data point di,
   - 7.1 Compute its distance from the centroid of the present nearest cluster;
   - 7.2 If this distance is less than or equal to the present nearest distance, the data point stays in the cluster; Else
     - 7.2.1 For every centroid cj (1 <= j <= k) Compute the distance d(di, cj); Endfor;
     - 7.2.2 Assign the data point di to the cluster with the nearest centroid cj;
     - 7.2.3 Set ClusterId[i] = j;
     - 7.2.4 Set NearestDist[i]= d(di, cj); Endfor;
8. For each cluster j (1 <= j <= k), recalculate the centroids; *Until* the convergence criteria is met.

Harmony Search Optimization

- Harmony search (HS) is a meta-heuristic search algorithm which tries to mimic the improvisation process of musicians in finding a pleasing harmony
- Initialize HS Memory
- Improvise HM based on probabilities HMCR and PAR
- Update HM



Step 1. Initialize the HS Memory (HM). The initial HM consists of a certain number of randomly generated solutions to the optimization problems under consideration. For an n -dimension problem, an HM with the size of N can be represented as follows:

$$\mathrm{HM} = \begin{bmatrix} x_1^1, x_2^1, \ldots, x_n^1 \\ x_1^2, x_2^2, \ldots, x_n^2 \\ \vdots \\ x_1^{\mathrm{HMS}}, x_2^{\mathrm{HMS}}, \ldots, x_n^{\mathrm{HMS}} \end{bmatrix}, \tag{1}$$

where $[x_1^i, x_2^i, \ldots, x_n^i]$ $(i = 1, 2, \ldots, \mathrm{HMS})$ is a solution candidate. HMS is typically set to be between 50 and 100.

Step 2. Improvise a new solution $[x_1', x_2', \ldots, x_n']$ from the HM. Each component of this solution, x_j', is obtained based on the Harmony Memory Considering Rate (HMCR). The HMCR is defined as the probability of selecting a component from the HM members, and 1-HMCR is, therefore, the probability of generating it randomly. If x_j' comes from the HM, it is chosen from the jth dimension of a random HM member and is further mutated according to the Pitching Adjust Rate (PAR). The PAR determines the probability of a candidate from the HM to be mutated. As we can see, the improvisation

Step 3. Update the HM. The new solution from Step 2 is evaluated. If it yields a better fitness than that of the worst member in the HM, it will replace that one. Otherwise, it is eliminated.

Step 4. Repeat Step 2 to Step 3 until a preset termination criterion, for example, the maximal number of iterations, is met.

Bioinformation. 2013; 9(2): 84–88.
Published online 2013 Jan 18. doi: 10.6026/97320630009084
PMCID: PMC3563403
PMID: 23390351

A novel harmony search-K means hybrid algorithm for clustering gene expression data

KA Abdul Nazeer,[1,*] MP Sebastian,[2] and SD Madhu Kumar[1]

- The proposed HSKH clustering algorithm consists of two phases. In the first phase, the initial centroids of the clusters are determined using an improved Harmony Search optimization technique. The centroids thus determined are used in the second phase to form the final clusters by repetitively assigning the data points to the clusters with the nearest centroids.

Phase 1: Improved Harmony Search Algorithm for finding the Initial Centroids

Input:

Gene Expression data matrix, D (Samples * Genes) // *m+1* rows * *n+1* columns

k // Number of desired clusters.

HMCR, PAR, MI // Harmony search optimization parameters

Output:

A set of *k* initial centroids

Steps:

1. For each column of the data matrix, *i = 2* to *n+1*
   - 1.1 Sort the first column of the data matrix based on column *i*;
   - 1.2 Divide the sorted column into *k* equal parts;
   - 1.3 For each part *j =1* to *k*, determine the index of the data item at the middle.
     - 1.3.1 Initialize the harmony memory *HM[i-1][j]* with the middle index.

   Endfor

   Endfor
2. Calculate the fitness value of all the candidate solutions in the harmony memory.
3. For *p = 1* to *MI*
   - 3.1 Improvise a new solution from the harmony memory.
   - 3.2 If the new solution is better than the worst solution in the harmony memory, replace the worst solution with the new solution.

   Endfor
4. Return the best solution in the harmony memory as the set of initial centroids.

with the nearest centroids using a variant of the method followed in [6].

Phase 2: Algorithm for assigning data-points to the clusters

Input:

D = {d_1, d_2,.....,d_n} // set of *n* data-points.

C = {c_1, c_2,.....,c_k} // set of *k* centroids

Output:

A set of *k* clusters

Steps:

1. Compute the Euclidean distance of each data-point *di* (1<=i<=n) to all the centroids *cj* (1<=j<=k) as *d(di, cj)*;
2. For each data-point *di*, find the closest centroid *cj* and assign *di* to cluster *j*.
3. Set ClusterId[i]=j; // j:Id of the closest cluster
4. Set Nearest_Dist[i]= *d(di, cj)*;
5. For each cluster *j* (1<=j<=k), recalculate the centroids;
6. **Repeat**
7. For each data-point *di*,

7.1 Compute its distance from the centroid of the present nearest cluster;

7.2 If this distance is less than or equal to the present nearest distance, the data-point stays in the cluster; Else

7.2.1 For every centroid *cj* (1<=j<=k), Compute the distance *d(di, cj)*;

Endfor;

7.2.2 Assign the data-point *di* to the cluster with the nearest centroid *cj;*

7.2.3 Set ClusterId[i]=j;

7.2.4 Set Nearest_Dist[i]= *d(di, cj)*;

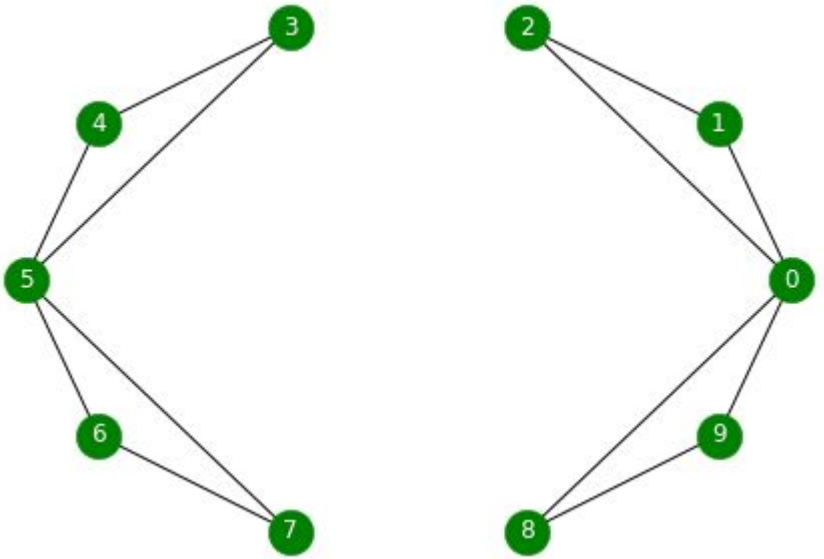
Endfor;

8. For each cluster *j* (1<=j<=k), recalculate the centroids;

Until convergence (i.e., no more data-points cross the cluster boundaries).

Spectral Clustering

- Spectral clustering is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them.
- Spectral clustering uses information from the **eigenvalues** (spectrum) of special matrices built from the graph or the data set



Important Concepts

- Adjacency Matrix
- Degree Matrix
- Graph Laplacian (L= D-A)
- Eigenvalues and Eigenvectors

1. Form a distance matrix

2. Transform the distance matrix into an affinity matrix A

3. Compute the degree matrix D and the Laplacian matrix L = D − A.

4. Find the eigenvalues and eigenvectors of L.

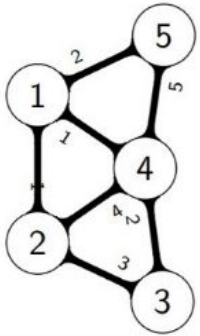
5. With the eigenvectors of k largest eigenvalues computed from the previous step form a matrix.

6. Normalize the vectors.

7. Cluster the data points in k-dimensional space.

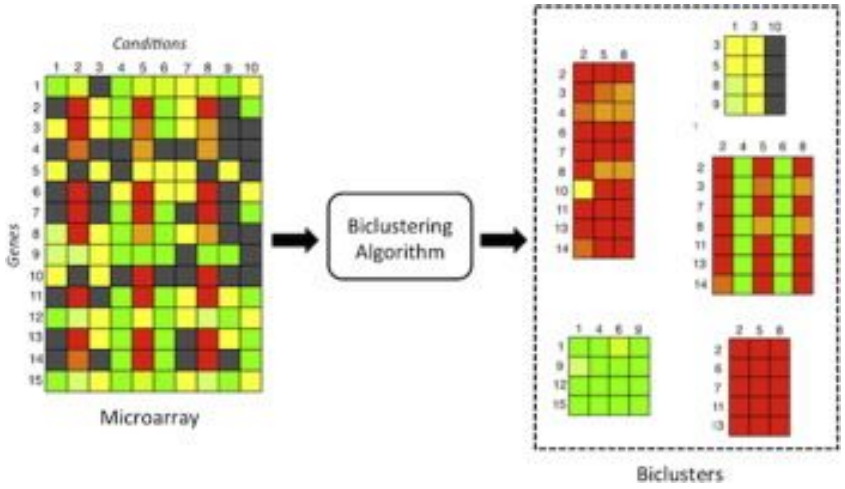
$$\begin{aligned} \mathbf{A} &\quad \text{adjacency matrix} \\ \mathbf{W} &\quad \text{weight matrix} \\ \mathbf{D} &\quad \text{(diagonal) degree matrix} \\ \mathbf{L} = \mathbf{D} - \mathbf{W} &\quad \text{graph Laplacian matrix} \end{aligned}$$

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & 0 & -1 & -2 \\ -1 & 8 & -3 & -4 & 0 \\ 0 & -3 & 5 & -2 & 0 \\ -1 & -4 & -2 & 12 & -5 \\ -2 & 0 & 0 & -5 & 7 \end{pmatrix}$$



Biclustering

- Biclustering is a powerful data mining technique that allows clustering of rows and columns, simultaneously, in a matrix-format data set
- Ex: identify co-expressed genes under a subset of all the conditions/samples



a) Bicluster with constant values

2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0

b) Bicluster with constant values on rows

1.0	1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0	4.0
5.0	5.0	5.0	5.0	5.0

c) Bicluster with constant values on columns

1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0

d) Bicluster with coherent values (additive)

1.0	4.0	5.0	0.0	1.5
4.0	7.0	8.0	3.0	4.5
3.0	6.0	7.0	2.0	3.5
5.0	8.0	9.0	4.0	5.5
2.0	5.0	6.0	1.0	2.5

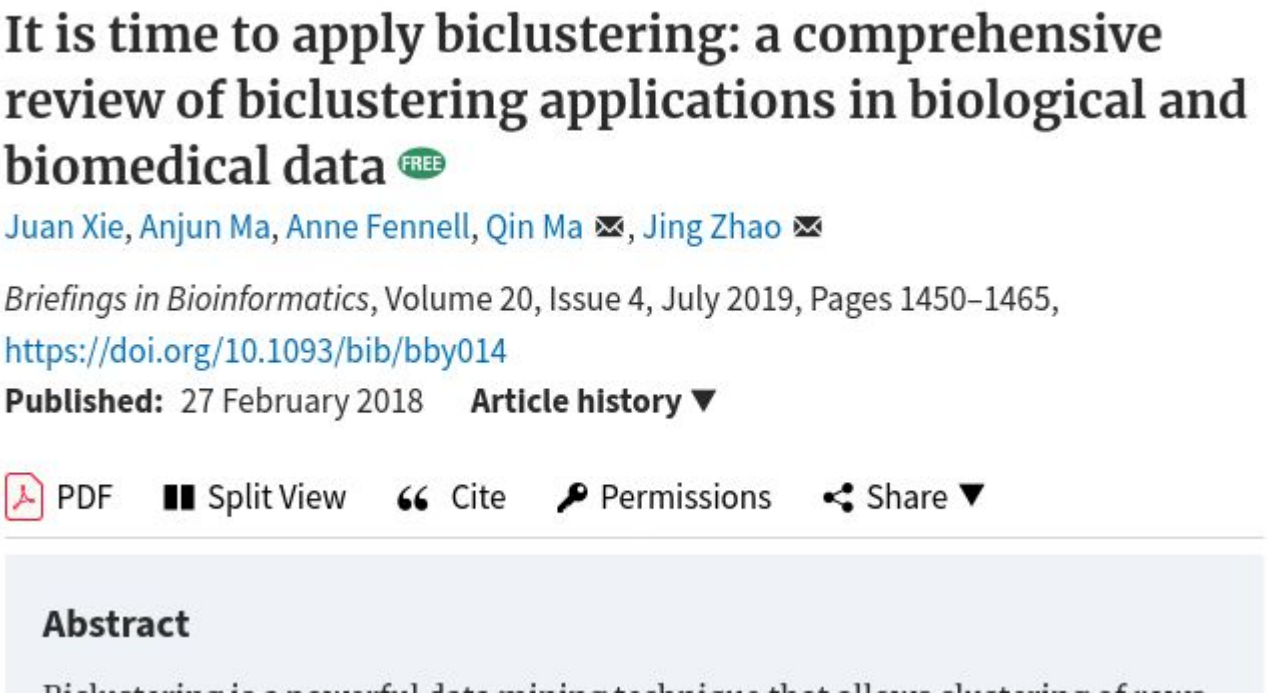
e) Bicluster with coherent values (multiplicative)

1.0	0.5	2.0	0.2	0.8
2.0	1.0	4.0	0.4	1.6
3.0	1.5	6.0	0.6	2.4
4.0	2.0	8.0	0.8	3.2
5.0	2.5	10.0	1.0	4.0

Applications of Biclustering

- Functional annotation of unclassified genes
- **Modularity analysis**: a group of physically or functionally linked molecules that work together to achieve distinct functions
- **Biological networks elucidation:** interactions can be conceptualized as networks, Analyzing these networks provides systematic views and novel insights for understanding the underlying mechanisms controlling cellular processes
- **Disease subtype identification**
- **Biomarker and gene signature identification**

Read more



It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data FREE

Juan Xie, Anjun Ma, Anne Fennell, Qin Ma ✉, Jing Zhao ✉

Briefings in Bioinformatics, Volume 20, Issue 4, July 2019, Pages 1450–1465,
https://doi.org/10.1093/bib/bby014
Published: 27 February 2018 **Article history** ▼

PDF Split View Cite Permissions Share ▼

Abstract

https://academic.oup.com/bib/article/20/4/1450/4911545