MAC 0459 / 5865

Data Science and Engineering

R. Hirata Jr. (hirata@ime.usp.br)

Class 13 (2020)

Clustering

Clustering – definition

Notation:

$X = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_m\}$ (objects)
Number of clusters: c (usually unknown)
Clusters: C_1, C_2, ..., C_c

Standard definition

Partition

A **partition of a set** X is a collection of parts/subsets $C_1, C_2, \ldots, C_c$, $c > 0$, such that:

- $C_j \neq \emptyset, \quad j = 1, \ldots, c$
- $\cup_{j=1}^{c} C_j = X$
- $C_i \cap C_j = \emptyset, \quad i, j = 1, 2, \ldots, c$ e $i \neq j$

Clustering approaches

- **Sequential:** fast and straightforward because the objects (feature vectors) are presented at most six times to the algorithm. The final result is dependent of the order of the objects presented. The resulting clusters are compact and hyperspherical or hyperellipsoidal.
- **Hierarchical:** agglomerative or divisive.
  - agglomerative: decreasing sequence of the number of clusters.
  - divisive: increasing sequence of the number of clusters.
- **Optimization:** The number is usually fixed and a cost function is optimized.
- **Other:** Branch and bound, genetic, stochastic, etc.

Optimization Algorithms

(minimize a cost function)

k-means

Cost function

Two basic types:

- Functions that measures variance between objects of the same group: Sum-of-Squared Error Criterion or Minimum Variance criteria.
- Functions based on Scatter matrices.

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m **objects, how many** $k = 1$ **partitions?**

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m objects, how many $k = 1$ partitions?

$$N(m, 1) = m$$

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m objects, how many $k = 1$ partitions?

$$N(m, 1) = m$$

How many $k = m$ partitions?

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m objects, how many $k = 1$ partitions?

$$N(m, 1) = m$$

How many $k = m$ partitions?

$$N(m, m) = 1$$

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m objects, how many $k = 1$ partitions?

$$N(m, 1) = m$$

How many $k = m$ partitions?

$$N(m, m) = 1$$

How many $k = n, n > m$ partitions?

Clustering based on the minimization of a cost function

From all possible partitioning, choose the one that minimizes a cost function

Given m objects, how many $k = 1$ partitions?

$$N(m, 1) = m$$

How many $k = m$ partitions?

$$N(m, m) = 1$$

How many $k = n, n > m$ partitions?

$$N(m, n) = 0$$

Clustering based on the minimization of a cost function

Let L_{m-1}^{k} be the set of all possible partitions of $m - 1$ elements into k parts.

If we add a new element to this partition:

Clustering based on the minimization of a cost function

Let L^k_{m-1} be the set of all possible partitions of $m-1$ elements into k parts.

If we add a new element to this partition:

this element can be added to one of the parts of L^k_{m-1}

Clustering based on the minimization of a cost function

Let L^k_{m-1} be the set of all possible partitions of $m-1$ elements into k parts.

If we add a new element to this partition:

this element can be added to one of the parts of L^k_{m-1}

this element can form a new cluster to each member of L^{k-1}_{m-1}

Clustering based on the minimization of a cost function

Therefore the number of possible clusterings of m elements in k clusters is:

$$N(m,k) = kN(m-1,k) + N(m-1,k-1)$$

$$N(m,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^m$$

Some examples: $N(9,2) = 109584$, $N(100,5) \sim 10^{68}$

Clustering based on the minimization of a cost function

Therefore the number of possible clusterings of m elements in k clusters is:

$$N(m,k) = kN(m-1,k) + N(m-1,k-1)$$

$$N(m,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^m$$

Some examples: $N(9,2) = 109584$, $N(100,5) \sim 10^{68}$

It is not possible to try all possible partitions!

Iteractive algorithms

- Start with an arbitrary partition (random, for instance)

Iteractive algorithms

- Start with an arbitrary partition (random, for instance)
- Repete until a stop criterium is satisfied

The returned solution is not optimal.

Iteractive algorithms

- Start with an arbitrary partition (random, for instance)
- Repete until a stop criterium is satisfied
  - slightly modify the partition

The returned solution is not optimal.

Iteractive algorithms

- Start with an arbitrary partition (random, for instance)
- Repete until a stop criterium is satisfied
  - slightly modify the partition
  - verify if the new partition decreases the cost function and substitute of true.

The returned solution is not optimal.

Iteractive algorithms

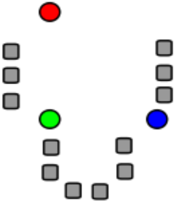
- Start with an arbitrary partition (random, for instance)
- Repete until a stop criterium is satisfied
  - slightly modify the partition
  - verify if the new partition decreases the cost function and substitute of true.
- Return the best partition

The returned solution is not optimal.

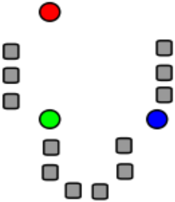
k-means algorithm

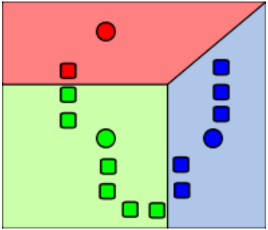
1. Choose k points in the feature space (initial centroids).
2. Put each object to be classified to the group whose centroid is nearer.
3. Recompute the centroids after distributing all the objects
4. Repete steps 2 and 3 until convergence of the centroids.

k-means algorithm: simulation

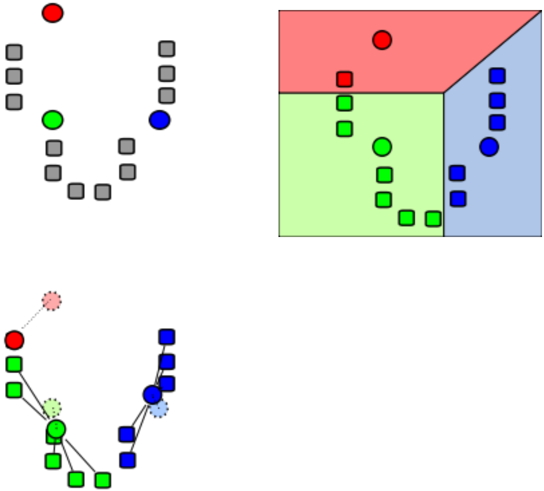


k-means algorithm: simulation

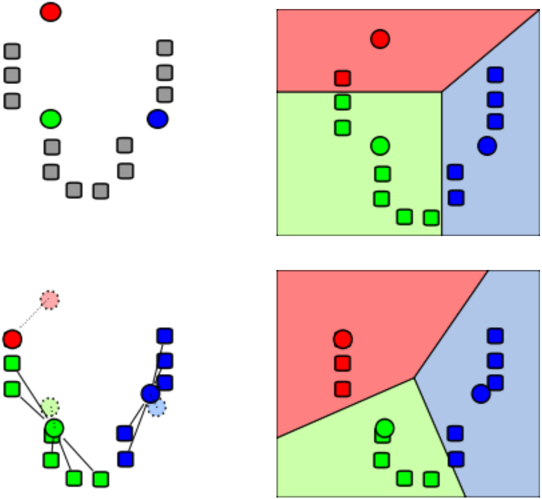




k-means algorithm: simulation



k-means algorithm: simulation



Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.

Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.
- Different initial partitions can generate different final results.

Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.
- Different initial partitions can generate different final results.
- Disadvantage: specify the number of classes.

Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.
- Different initial partitions can generate different final results.
- Disadvantage: specify the number of classes.
- Alternative: run the algorithm for different number of clusters $k = 1, 2, 3, \ldots$, and analyse some critera to choose k.

Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.
- Different initial partitions can generate different final results.
- Disadvantage: specify the number of classes.
- Alternative: run the algorithm for different number of clusters $k = 1, 2, 3, \ldots$, and analyse some critera to choose k.
- The algorithm can use other points than the centroid.

Algoritmo k-means

- There are several k-means alternatives, mainly in relation to the initial k prototypes.
- Different initial partitions can generate different final results.
- Disadvantage: specify the number of classes.
- Alternative: run the algorithm for different number of clusters $k = 1, 2, 3, \ldots$, and analyse some critera to choose k.
- The algorithm can use other points than the centroid.
- It is possible to show that k-means algorithms minimize the "mean square error" cost function.

Cost function

- **Set of objects to be grouped**: m itens

$$\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_m\}$$

- **Clustering**: c clusters

$$C_j = \{\mathrm{x}_{j1}, \mathrm{x}_{j2}, \ldots, \mathrm{x}_{jn_j}\}, \quad j = 1, 2, \ldots, c$$

- **Centroid** C_j: m_j (n_j itens)

$$\mathrm{m}_j = \frac{1}{n_j} \sum_{\mathrm{x} \in C_j} \mathrm{x}$$

- **Global mean** (m objects)

$$\mathrm{m} = \frac{1}{m} \sum_{\mathrm{x} \in \mathrm{X}} \mathrm{x} = \frac{1}{m} \sum_{j=1}^{c} n_j \mathrm{m}_j$$

Minimum variance cost

Object: $\mathrm{x} = (x_1, \ldots, x_d)$

Group mean C_j: $\mathrm{m}_j = (m_{j1}, \ldots, m_{jd})$

Difference between x **and** m_j:

$$||\mathrm{x} - \mathrm{m}_j||^2 = (x_1 - m_{j1})^2 + \ldots + (x_d - m_{jd})^2$$

Sum of the squared differences between all objects of the same group to its mean.

$$\sum_{\mathrm{x} \in C_j} ||\mathrm{x} - \mathrm{m}_j||^2$$

Cost based on minimum variance

Sum of the squared differences

$$J_e = \sum_{j=1}^{c} \sum_{\mathrm{x} \in C_j} ||\mathrm{x} - \mathrm{m}_j||^2$$

Can be rewritten as:

$$J_e = \frac{1}{2} \sum_{j=1}^{c} n_j \bar{s}_j$$

where

$$\bar{s}_j = \frac{1}{n_j^2} \sum_{\mathrm{x} \in C_j} \sum_{\mathrm{x}' \in C_j} ||\mathrm{x} - \mathrm{x}'||^2$$

In this formulation, it is clear that we are computing the Euclidean mean squared distance to all pair of points in the group.

Cost based on minimum variance

Sum of the squared differences cost function (or minimum variance)

$$J_e = \sum_{i=1}^{c} \sum_{\mathrm{x} \in \mathrm{X}_i} ||\mathrm{x} - \mathrm{m}_i||^2$$

Cost based on minimum variance

Sum of the squared differences cost function (or minimum variance)

$$J_e = \sum_{i=1}^{c} \sum_{\mathrm{x} \in \mathrm{X}_i} ||\mathrm{x} - \mathrm{m}_i||^2$$

- k means minimizes J_e cost function

Cost based on minimum variance

Sum of the squared differences cost function (or minimum variance)

$$J_e = \sum_{i=1}^{c} \sum_{\mathrm{x} \in \mathrm{X}_i} ||\mathrm{x} - \mathrm{m}_i||^2$$

- k means minimizes J_e cost function
- Compact groups

Cost based on minimum variance

Sum of the squared differences cost function (or minimum variance)

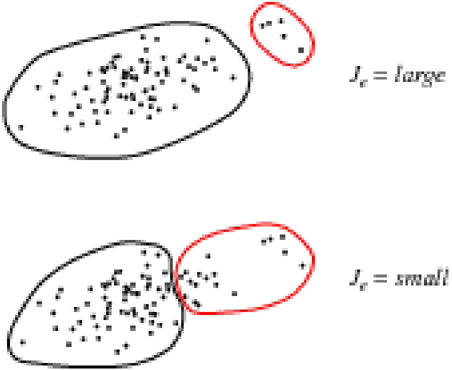
$$J_e = \sum_{i=1}^{c} \sum_{\mathsf{x} \in \mathsf{X}_i} ||\mathsf{x} - \mathsf{m}_i||^2$$

- k means minimizes J_e cost function
- Compact groups
- May not be the best option if group sizes are too different

Cost based on minimum variance

Sum of the squared differences cost function (or minimum variance)

$$J_e = \sum_{i=1}^{c} \sum_{\mathrm{x} \in \mathrm{X}_i} ||\mathrm{x} - \mathrm{m}_i||^2$$



Cost based on minimum variance

In the equation of J_e,

$$J_e = \sum_{j=1}^{c} \sum_{\mathrm{x} \in C_j} ||\mathrm{x} - \mathrm{m}_j||^2 = \frac{1}{2} \sum_{i=j}^{c} n_j \bar{s}_j$$

$\bar{s}_j$ can be substituted by any other measure.

Specifically, in this equation

$$\bar{s}_j = \frac{1}{n_j^2} \sum_{\mathrm{x} \in C_j} \sum_{\mathrm{x}' \in C_j} ||\mathrm{x} - \mathrm{x}'||^2$$

the term in red can be substituted by any other similarity measures $s(\mathrm{x}, \mathrm{x}')$

In special

$$\bar{s}_j = min_{\mathrm{x},\mathrm{x}' \in C_j} s(\mathrm{x}, \mathrm{x}')$$

Cost based on scatter matrix

- **Scatter matrix for cluster j:**

$$S_j = \sum_{\mathrm{x} \in C_j} (\mathrm{x} - \mathrm{m}_j)(\mathrm{x} - \mathrm{m}_j)^t$$

- **Within class scatter matrix:**

$$S_W = \sum_{i=j}^{c} S_j$$

- **Between class scatter matrix:**

$$S_B = \sum_{j=1}^{c} n_j (\mathrm{m}_j - \mathrm{m})(\mathrm{m}_j - \mathrm{m})^t$$

- **Total scatter matrix:** it does not depend on the partioning

$$S_T = \sum_{\mathrm{x} \in \mathrm{X}} (\mathrm{x} - \mathrm{m})(\mathrm{x} - \mathrm{m})^t$$

Pause for an example (J&W, page 57)

Some facts to think on

Scatter matrix S_j of a class j is proportinal to the sample convariance matrix of that same class

The eigenvalues and eigenvectors of S_j tell the orthogonal directions of the higher variance of clusters S_j

The sum of variances (diagonal of matrix S_j) is equal to the sum of the eigenvalues of S_j

The Within class scatter matrix S_W "summarizes" the internal variance of the classes

Cost based on scatter matrix

Trace cost:

$$tr[S_W] = \sum_{j=1}^{c} tr[S_j] = \sum_{j=1}^{c} \sum_{\mathrm{x} \in C_j} ||\mathrm{x} - \mathrm{m}_j||^2 = J_e$$

trace: sum of the diagonal elements

Diagnonal elements: represent variances in each direction of the feature space $\mathbb{R}^d$

Minimize the trace of S_W means minimize the Within class spreading

$tr[S_W]$ is equivalent to the squared sum cost function

Cost based on scatter matrix

Instead of minimizing the Within class, we can **maximize the between class matrix**

Because $S_T = S_W + S_B$, maximize $tr[S_B]$ is equivalent to minimize $tr[S_W]$.

Cost based on scatter matrix

Determinant cost:

$$J_d = |S_W| = |\sum_{j=1}^{c} S_j|$$

The determinant of the Within class scatter matrix represents the volume of the scattering of a cluster.

In many situations it results in clusters that are similar to the trace cost function.

However, it is not sensible to scaling.

Cost based on scatter matrix

The eigenvalues $\lambda_1, \ldots, \lambda_d$ of $S_W^{-1} S_B$ are invariant to non null linear transforms.

We can show that:

$$tr[S_W^{-1} S_B] = \sum_{i=1}^{d} \lambda_i$$

Therefore, a good criterium is to maximize $tr[S_W^{-1} S_B]$

Invariant cost function

$$J_d = tr[S_T^{-1} S_B] = \sum_{i=1}^{d} \frac{1}{1 + \lambda_i}$$

How to validate the clustering result

- Run the algorithm several times, using different parameters
- Run different cluster algorithms
- Check with area specialists

Some other thoughs and algorithms

Clustering based on density

Clusters are regions of high density.

Basic idea: estimate the density of the points in the space and group based on density significance

Good for complex shaped clusters

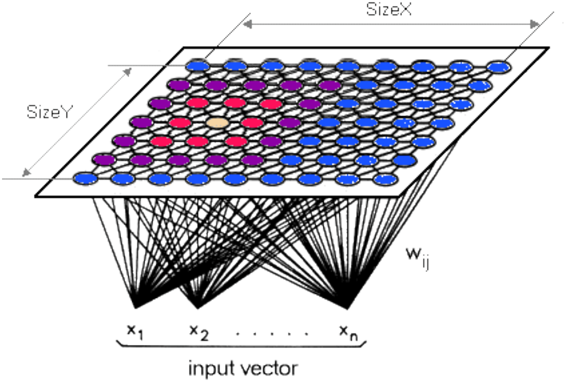
SOM - Self Organized Maps

SOM - Self Organized Maps

Basic idea: map objects in a high dimensional space to a low dimensional space making that objects that are near in high dimension remain near in low dimension.

- **The low dimension space corresponds to a set of notes organized in a grid in the plane (map)**
- **Each node of the map has a coordinate (in the plane) and a vector of weights of dimension d**
- **OBS.: The literature usually presents as SOM as a kind of neural network**

SOM - Self Organized Maps - Architecture



Orange nodes: BMU (best matching unit), node that has a vector of weights similar to a given input $x \in X$.

Pink and dark blue nodes: neighbors defined by a window function.

SOM - Self Organized Maps - Algorithm

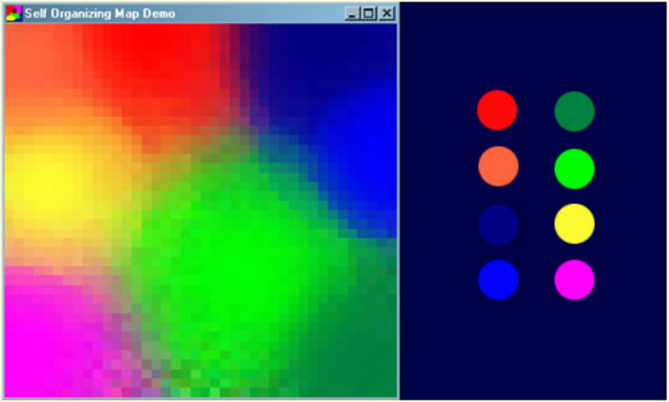
```
Initialize the weight of the nodes of the map
Repeat
    For each $\mathrm{x} \in X$
        Let $\mathrm{p}_k$ be a BMU
        Update the BMU and its neighbors $\mathrm{p}$

            $\mathrm{w}_{ki}(t+1) = \mathrm{w}_{ki}(t) + \eta(t)\phi(\mathrm{p} - \mathrm{p}_k)(\mathrm{x}_i - \mathrm{w}_{ki}(t))$

 until convergence
```

ϕ is a window function (kernel function) and $\eta(t)$ is a learning rate.

w_{ki} is the ith component of the weight vector w_k associated to node p_k in the map

SOM - Self Organized Maps

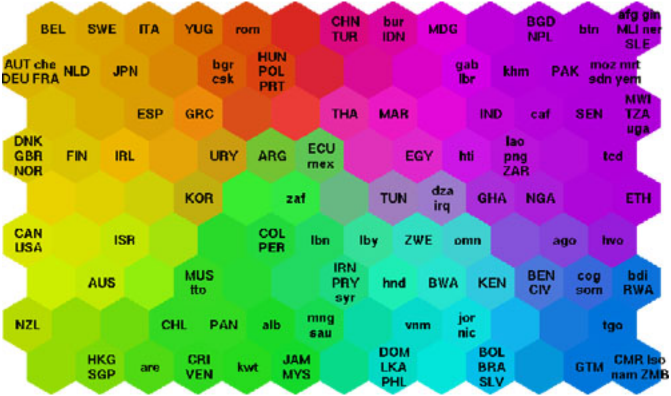
Example: if the weight vector has 3 components, they can be thought as the R, G, B channes and the map can be "painted" by the corresponding RGB color.



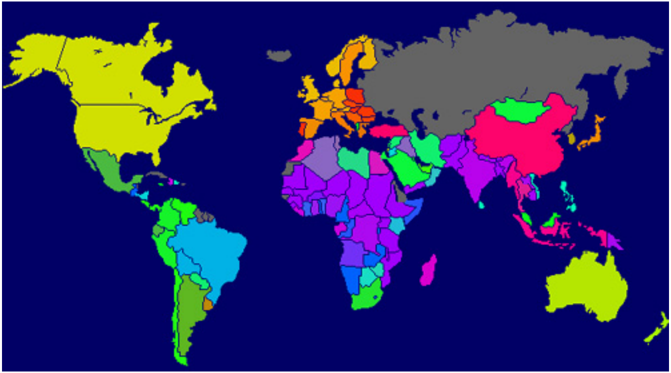
It is not easy to divide the map in regions: how may colors (groups)? To which group the nodes in the border are inside (for instance, between green and blue?)

SOM - Self Organized Maps

Example: The original space are several statistics of a country (education, health, etc)

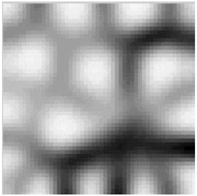


SOM - Self Organized Maps



SOM - Self Organized Maps

Interpretation of a map



Color of the nodes: the intensity represents the difference between nodes (neighbors), for instance, the mean difference between the weight vectors.

Dark lines corresponds to discontinuities and light color regions to similar weight nodes

Each region can be interpreted as a group

We still can apply clustering.