Introduction to Machine Learning

K-Means, K-Meoids, K-Centers and Variations

Andres Mendez-Vazquez

August 4, 2018

Outline

Outline

The Hardness of K-means clustering

Definition

- Given a multiset $S \subseteq \mathbb{R}^d$, an integer k and $L \in \mathbb{R}$, is there a subset $T \subset \mathbb{R}^d$ with $|T| = k$ such that

$$\sum_{\boldsymbol{x} \in S} \min_{\boldsymbol{t} \in T} \|\boldsymbol{x} - \boldsymbol{t}\|^2 \leq L?$$

Theorem

- The k-means clustering problem is NP-complete even for $d = 2$.

The Hardness of K-means clustering

Definition

- Given a multiset $S \subseteq \mathbb{R}^d$, an integer k and $L \in \mathbb{R}$, is there a subset $T \subset \mathbb{R}^d$ with $|T| = k$ such that

$$\sum_{\boldsymbol{x} \in S} \min_{\boldsymbol{t} \in T} \|\boldsymbol{x} - \boldsymbol{t}\|^2 \leq L?$$

Theorem

- The k-means clustering problem is NP-complete even for $d = 2$.

Reduction

The reduction to an NP-Complete problem

- Exact Cover by 3-Sets problem

Definition

- Given a finite set U containing exactly $3n$ elements and a collection $C = \{S_1, S_2, ..., S_l\}$ of subsets of U each of which contains exactly 3 elements, Are there n sets in C such that their union is U?

Reduction

The reduction to an NP-Complete problem

- Exact Cover by 3-Sets problem

Definition

- Given a finite set U containing exactly $3n$ elements and a collection $\mathcal{C} = \{S_1, S_2, ..., S_l\}$ of subsets of U each of which contains exactly 3 elements, Are there n sets in $\mathcal{C}$ such that their union is U?

However

There are efficient heuristic and approximation algorithms

- Which can solve this problem

Outline

K-Means - Stuart Lloyd(Circa 1957)

History

Invented by Stuart Loyd in Bell Labs to obtain the best quantization in a signal data set.

Something Notable

The paper was published until 1982

Basically given N vectors $x_1, ..., x_N \in \mathbb{R}^d$

It tries to find k points $\mu_1, ..., \mu_k \in \mathbb{R}^d$ that minimize the expression (i.e. a partition S of the vector points):

$$\sum_{k=1}^{N} \sum_{i:x_i \in C_k} \|x_i - \mu_k\|^2 = \sum_{k=1}^{N} \sum_{i:x_i \in C_k} (x_i - \mu_k)^T (x_i - \mu_k)$$

K-Means - Stuart Lloyd(Circa 1957)

History

Invented by Stuart Loyd in Bell Labs to obtain the best quantization in a signal data set.

Something Notable

The paper was published until 1982

Basically given N vectors $x_1, ..., x_N \in \mathbb{R}^d$

It tries to find k points $\mu_1, ..., \mu_k \in \mathbb{R}^d$ that minimize the expression (i.e. a partition S of the vector points):

$$\sum_{k=1}^{N} \sum_{i:x_i \in C_k} \|x_i - \mu_k\|^2 = \sum_{k=1}^{N} \sum_{i:x_i \in C_k} (x_i - \mu_k)^T (x_i - \mu_k)$$

K-Means - Stuart Lloyd(Circa 1957)

History

Invented by Stuart Loyd in Bell Labs to obtain the best quantization in a signal data set.

Something Notable

The paper was published until 1982

Basically given N vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_N \in \mathbb{R}^d$

It tries to find k points $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k \in \mathbb{R}^d$ that minimize the expression (i.e. a partition S of the vector points):

$$\sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|^2 = \sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k \right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k \right)$$

K-means clustering

K-means

It is a partitional clustering algorithm.

K-means clustering

K-means

It is a partitional clustering algorithm.

Definition

Let the set of data points (or instances) D be $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ir})^T$:

- The K-means algorithm partitions the given data into K clusters.
- Each cluster has a cluster center, called centroid.
- K is specified by the user.

K-means clustering

K-means

It is a partitional clustering algorithm.

Definition

Let the set of data points (or instances) D be $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ir})^T$:

- The K-means algorithm partitions the given data into K clusters.
- Each cluster has a cluster center, called centroid.
- K is specified by the user.

K-means clustering

K-means

It is a partitional clustering algorithm.

Definition

Let the set of data points (or instances) D be $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ir})^T$:

- The K-means algorithm partitions the given data into K clusters.
- Each cluster has a cluster center, called centroid.
- K is specified by the user.

K-means clustering

K-means

It is a partitional clustering algorithm.

Definition

Let the set of data points (or instances) D be $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ir})^T$:

- The K-means algorithm partitions the given data into K clusters.
- Each cluster has a cluster center, called centroid.
- K is specified by the user.

K-means algorithm

The K-means algorithm works as follows

Given k as the possible number of cluster:

1. Randomly choose K data points (seeds) to be the initial centroids, cluster centers,
   - $\{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$
2. Assign each data point to the closest centroid
   - $c_i = \arg\min_j \{dist(\mathbf{x}_i - \mathbf{v}_j)\}$
3. Re-compute the centroids using the current cluster memberships.
   - $\mathbf{v}_j = \dfrac{\sum_{i=1}^{n} I(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(c_i = j)}$
4. If a convergence criterion is not met, go to 2.

K-means algorithm

The K-means algorithm works as follows

Given k as the possible number of cluster:

1. Randomly choose K data points (seeds) to be the initial centroids, cluster centers,
   - $\{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$
2. Assign each data point to the closest centroid
   - $c_i = \arg\min_j \{dist(\mathbf{x}_i - \mathbf{v}_j)\}$
3. Re-compute the centroids using the current cluster memberships.
   - $\mathbf{v}_j = \dfrac{\sum_{i=1}^{n} I(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(c_i = j)}$
4. If a convergence criterion is not met, go to 2.

K-means algorithm

The K-means algorithm works as follows

Given k as the possible number of cluster:

1. Randomly choose K data points (seeds) to be the initial centroids, cluster centers,
   - $\{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$
2. Assign each data point to the closest centroid
   - $c_i = \arg\min_{j} \{dist(\mathbf{x}_i - \mathbf{v}_j)\}$
3. Re-compute the centroids using the current cluster memberships.
   - $$\mathbf{v}_j = \frac{\sum_{i=1}^{n} I(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(c_i = j)}$$
4. If a convergence criterion is not met, go to 2.

K-means algorithm

The K-means algorithm works as follows

Given k as the possible number of cluster:

1. Randomly choose K data points (seeds) to be the initial centroids, cluster centers,
   - $\{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$
2. Assign each data point to the closest centroid
   - $c_i = \arg\min_j \{dist(\mathbf{x}_i - \mathbf{v}_j)\}$
3. Re-compute the centroids using the current cluster memberships.
   - $\mathbf{v}_j = \dfrac{\sum_{i=1}^{n} I(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(c_i = j)}$
4. If a convergence criterion is not met, go to 2.

K-means algorithm

The K-means algorithm works as follows

Given k as the possible number of cluster:

1. Randomly choose K data points (seeds) to be the initial centroids, cluster centers,
   - $\{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$
2. Assign each data point to the closest centroid
   - $c_i = \arg\min_j \{dist(\mathbf{x}_i - \mathbf{v}_j)\}$
3. Re-compute the centroids using the current cluster memberships.
   - $\mathbf{v}_j = \dfrac{\sum_{i=1}^{n} I(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(c_i = j)}$
4. If a convergence criterion is not met, go to 2.

What is the code trying to do?

It is trying to find a partition S

K-means tries to find a partition S such that it minimizes the cost function:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{1}$$

Where μ_k is the centroid for cluster C_k

$$\mu_k = \frac{1}{N_k} \sum_{i:\boldsymbol{x}_i \in C_k} x_i \tag{2}$$

Where N_k is the number of samples in the cluster C_k.

What is the code trying to do?

It is trying to find a partition S

K-means tries to find a partition S such that it minimizes the cost function:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{1}$$

Where μ_k is the centroid for cluster C_k

$$\mu_k = \frac{1}{N_k} \sum_{i:\boldsymbol{x}_i \in C_k} x_i \tag{2}$$

Where N_k is the number of samples in the cluster C_k.

What is the code trying to do?

It is trying to find a partition S

K-means tries to find a partition S such that it minimizes the cost function:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{1}$$

Where μ_k is the centroid for cluster C_k

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i \tag{2}$$

Where N_k is the number of samples in the cluster C_k.

Outline

What Stopping/convergence criterion should we use?

First

No (or minimum) re-assignments of data points to different clusters.

What Stopping/convergence criterion should we use?

First

No (or minimum) re-assignments of data points to different clusters.

Second

No (or minimum) change of centroids.

- C_k is cluster k
- v_k is the centroid of cluster C_k

$$SSE = \sum_{k=1}^{K} \sum_{x \in C_k} dist(x, v_k)^2$$

What Stopping/convergence criterion should we use?

First

No (or minimum) re-assignments of data points to different clusters.

Second

No (or minimum) change of centroids.

Third

Minimum decrease in the sum of squared error (SSE),

- C_k is cluster k.
- $\mathbf{v}_k$ is the centroid of cluster C_k.

$$SSE = \sum_{k=1}^{K} \sum_{x \in C_k} dist\left(\mathbf{x}, \mathbf{v}_k\right)^2$$

What Stopping/convergence criterion should we use?

First

No (or minimum) re-assignments of data points to different clusters.

Second

No (or minimum) change of centroids.

Third

Minimum decrease in the sum of squared error (SSE),

- C_k is cluster k.
- $\mathbf{v}_k$ is the centroid of cluster C_k.

$$SSE = \sum_{k=1}^{K} \sum_{x \in C_k} dist\left(\mathbf{x}, \mathbf{v}_k\right)^2$$

What Stopping/convergence criterion should we use?

First

No (or minimum) re-assignments of data points to different clusters.

Second

No (or minimum) change of centroids.

Third

Minimum decrease in the sum of squared error (SSE),

- C_k is cluster k.
- $\mathbf{v}_k$ is the centroid of cluster C_k.

$$SSE = \sum_{k=1}^{K} \sum_{x \in c_k} dist\left(\mathbf{x}, \mathbf{v}_k\right)^2$$

Outline

The distance function

Actually, we have the following distance functions:

Euclidean

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}|| = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}$$

Manhattan

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Mahalanobis

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_A = \sqrt{(\mathbf{x}-\mathbf{y})^T A(\mathbf{x}-\mathbf{y})}$$

The distance function

Actually, we have the following distance functions:

Euclidean

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}|| = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}$$

Manhattan

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Mahalanobis

$$dist(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_A = \sqrt{(\mathbf{x}-\mathbf{y})^T A(\mathbf{x}-\mathbf{y})}$$

The distance function

Actually, we have the following distance functions:

Euclidean

$$dist(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}|| = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}$$

Manhattan

$$dist(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_1 = \sum_{i=1}^{n} |x_i - y_i|$$

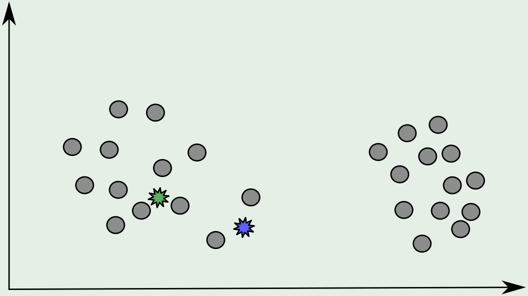
Mahalanobis

$$dist(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_A = \sqrt{(\mathbf{x} - \mathbf{y})^T A (\mathbf{x} - \mathbf{y})}$$

Outline

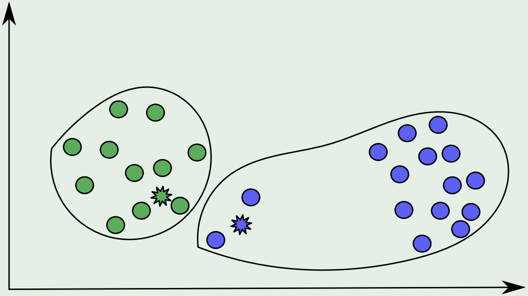
An example

Dropping two possible centroids



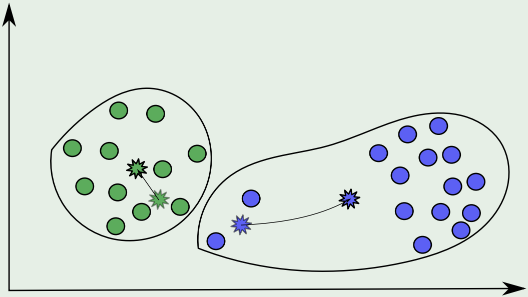
An example

Calculate the memberships



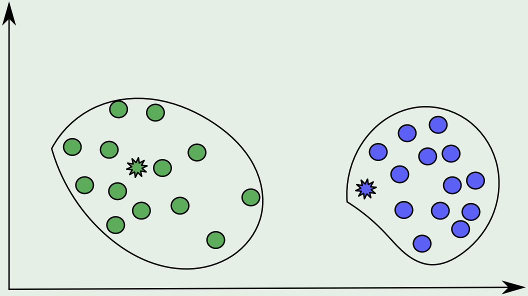
An example

We re-calculate centroids



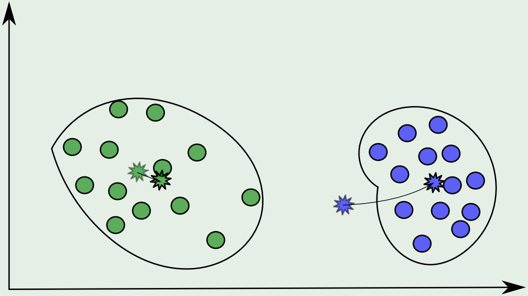
An example

We re-calculate memberships



An example

We re-calculate centroids and keep going



Outline

Strengths of K-means

Strengths

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tKN)$, where N is the number of data points, K is the number of clusters, and t is the number of iterations.
- Since both K and t are small. K-means is considered a linear algorithm.

Strengths of K-means

Strengths

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tKN)$, where N is the number of data points, K is the number of clusters, and t is the number of iterations.
- Since both K and t are small. K-means is considered a linear algorithm.

Popularity

K-means is the most popular clustering algorithm.

Strengths of K-means

Strengths

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tKN)$, where N is the number of data points, K is the number of clusters, and t is the number of iterations.
- Since both K and t are small. K-means is considered a linear algorithm.

Popularity

K-means is the most popular clustering algorithm.

Note that

It terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Strengths of K-means

Strengths

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tKN)$, where N is the number of data points, K is the number of clusters, and t is the number of iterations.
- Since both K and t are small. K-means is considered a linear algorithm.

Popularity

K-means is the most popular clustering algorithm.

Note that

It terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Strengths of K-means

Strengths

- Simple: easy to understand and to implement
- Efficient: Time complexity: $O(tKN)$, where N is the number of data points, K is the number of clusters, and t is the number of iterations.
- Since both K and t are small. K-means is considered a linear algorithm.

Popularity

K-means is the most popular clustering algorithm.

Note that

It terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

In addition

The user needs to specify K.

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

In addition

The user needs to specify K.

Outliers

The algorithm is sensitive to outliers.

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

In addition

The user needs to specify K.

Outliers

The algorithm is sensitive to outliers.

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

In addition

The user needs to specify K.

Outliers

The algorithm is sensitive to outliers.

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of K-means

Important

The algorithm is only applicable if the mean is defined.

- For categorical data, K-mode - the centroid is represented by most frequent values.

In addition

The user needs to specify K.

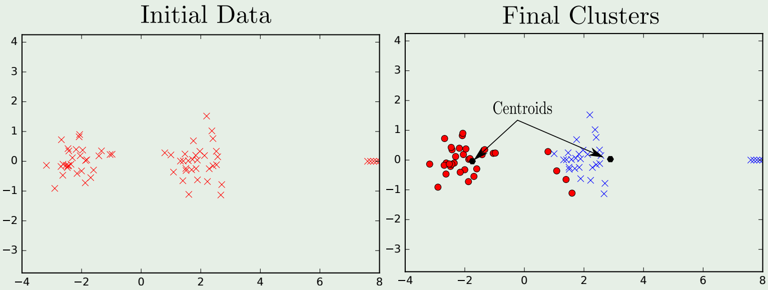
Outliers

The algorithm is sensitive to outliers.

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

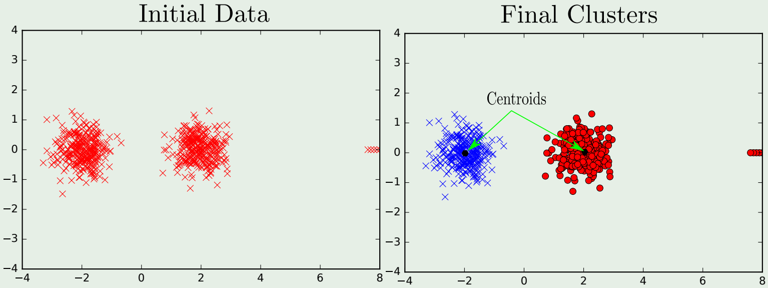
Weaknesses of K-means: Problems with outliers

A series of outliers



Weaknesses of K-means: Problems with outliers

Nevertheless, if you have more dense clusters



Weaknesses of K-means: How to deal with outliers

One method

To remove some data points in the clustering process that are much further away from the centroids than other data points.

- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

Weaknesses of K-means: How to deal with outliers

One method

To remove some data points in the clustering process that are much further away from the centroids than other data points.

- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

Another method

To perform random sampling.

- Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification.

Weaknesses of K-means: How to deal with outliers

One method

To remove some data points in the clustering process that are much further away from the centroids than other data points.

- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

Another method

To perform random sampling.

- Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification.

Weaknesses of K-means: How to deal with outliers

One method

To remove some data points in the clustering process that are much further away from the centroids than other data points.

- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

Another method

To perform random sampling.

- Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification.

Weaknesses of K-means: How to deal with outliers

One method

To remove some data points in the clustering process that are much further away from the centroids than other data points.

- To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

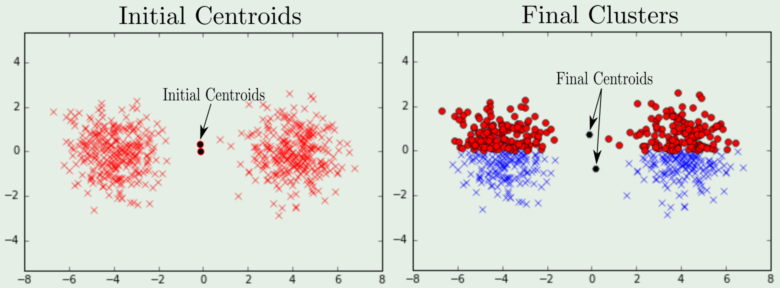
Another method

To perform random sampling.

- Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification.

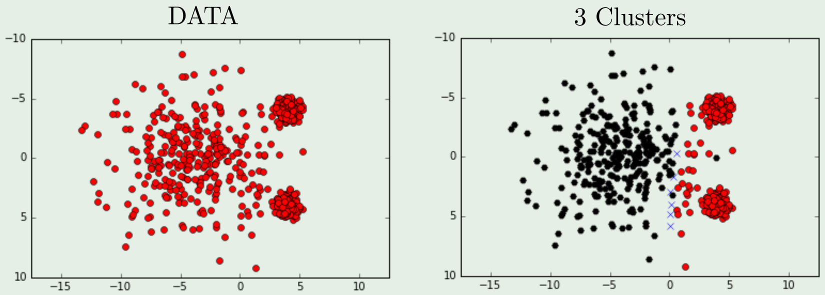
Weaknesses of K-means (cont...)

The algorithm is sensitive to initial seeds



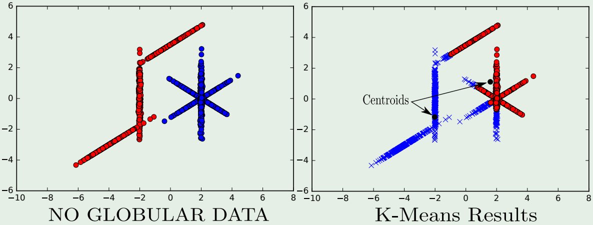
Weaknesses of K-means : Different Densities

We have three cluster nevertheless



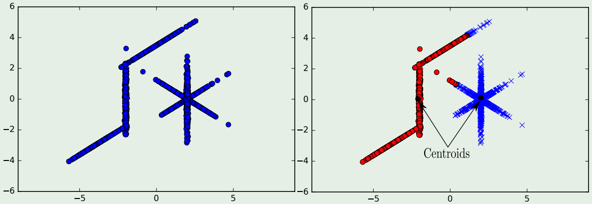
Weaknesses of K-means: Non-globular Shapes

Here, we notice that K-means may only detect globular shapes



Weaknesses of K-means: Non-globular Shapes

However, it sometimes work better than expected



NO GLOBULAR DATA K-Means Results

Outline

Consider the following

Theorem

- Every matrix $A \in R^{m \times n}$ has an SVD.

Frobenious Matrix Norm

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{\text{trace}\left(A^T A\right)}$$

Consider the following

Theorem

- Every matrix $A \in R^{m \times n}$ has an SVD.

Frobenious Matrix Norm

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} = \sqrt{\text{trace}\left(A^T A\right)}$$

Then, you have a the Eckhart-Young Theorem

Theorem

- Let A be a real $m \times n$ matrix. Then for any $k \in \mathbb{N}$ and any $m \times m$ orthogonal projection matrix P of rank k, we have

$$\|A - P_k A\|_F \leq \|A - PA\|_F$$

  - with $P_k = \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^T$

Thus

We have the Covariance matrix

$$S=\frac{1}{N-1}\sum_{i=1}^{N}\left(\boldsymbol{x}_i-\overline{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i-\overline{\boldsymbol{x}}\right)^T$$

Therefore, we have the following decomposition

$$S=U\Sigma U^T$$

- Where $UU^T=I$ and U is a $d\times d$ matrix

Thus

We have the Covariance matrix

$$S = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}) (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

Therefore, we have the following decomposition

$$S = U \Sigma U^T$$

- Where $UU^T = I$ and U is a $d \times d$ matrix

Orthogonal Projection

Therefore, we have that U is a orthogonal projection

- Given that $UU^T = I$ and $U\boldsymbol{x} = \boldsymbol{x}$

Now, we can re-write k-means

$$f_{k-\text{mean}} = \min_{\mu_1,\ldots\mu_k} \sum_{i\in[n]} \min_{j\in[k]} \left\|x_i - \mu_j\right\|^2$$

Orthogonal Projection

Therefore, we have that U is a orthogonal projection

- Given that $UU^T = I$ and $U\boldsymbol{x} = \boldsymbol{x}$

Now, we can re-write k-means

$$f_{k-\text{mean}} = \min_{\mu_1,\ldots\mu_k} \sum_{i\in[n]} \min_{j\in[k]} \left\|\boldsymbol{x}_i - \mu_j\right\|^2$$

Then

PCA can also re-write the cost function

$$f_{PCA} = \min_{P_k} \sum_{i \in [n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 = \min_{P_k} \sum_{i \in [n]} \min_{\boldsymbol{y}_i \in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2$$

Where

- Given that P_k is a projection into dimension k and $y \in P_k$ means that $P_k y = y$

Furthermore

$$\arg\min_{y \in P} \|x - y\| = Px$$

Then

PCA can also re-write the cost function

$$f_{PCA} = \min_{P_k} \sum_{i \in [n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 = \min_{P_k} \sum_{i \in [n]} \min_{\boldsymbol{y}_i \in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2$$

Where

- Given that P_k is a projection into dimension k and $\boldsymbol{y} \in P_k$ means that $P_k \boldsymbol{y} = \boldsymbol{y}$

Furthermore

$$\arg\min_{y \in P} \|x - y\| = Px$$

Then

PCA can also re-write the cost function

$$f_{PCA} = \min_{P_k} \sum_{i\in[n]} \left\|\boldsymbol{x}_i - P_k\boldsymbol{x}_i\right\|^2 = \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i \in P_k} \left\|\boldsymbol{x}_i - \boldsymbol{y}_i\right\|^2$$

Where

- Given that P_k is a projection into dimension k and $\boldsymbol{y} \in P_k$ means that $P_k\boldsymbol{y} = \boldsymbol{y}$

Furthermore

$$\arg\min_{y\in P} \|\boldsymbol{x} - \boldsymbol{y}\| = P\boldsymbol{x}$$

Thus, using the Eckhart-Young Theorem

Assume P_k^* which contains the k optimal centers

- Given that $\mu_j \in P_k^*$

$$
\begin{aligned}
f_{k-\text{mean}} &= \sum_{i\in[n]} \min_{j\in[k]} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 \\
&\geq \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k^*} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&\geq \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&= \min_{P_k} \sum_{i\in[n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 \\
&= f_{PCA}
\end{aligned}
$$

Thus, using the Eckhart-Young Theorem

Assume P_k^* which contains the k optimal centers

- Given that $\mu_j \in P_k^*$

$$
\begin{aligned}
f_{k-\text{mean}} &= \sum_{i\in[n]} \min_{j\in[k]} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 \\
&\geq \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k^*} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&\geq \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&= \min_{P_k} \sum_{i\in[n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 \\
&= f_{PCA}
\end{aligned}
$$

Thus, using the Eckhart-Young Theorem

Assume P_k^* which contains the k optimal centers

- Given that $\mu_j \in P_k^*$

$$
\begin{aligned}
f_{k-\text{mean}} &= \sum_{i\in[n]} \min_{j\in[k]} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 \\
&\geq \sum_{i\in[n]} \min_{\boldsymbol{y}_i \in P_k^*} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&\geq \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i \in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&= \min_{P_k} \sum_{i\in[n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 \\
&= f_{PCA}
\end{aligned}
$$

Thus, using the Eckhart-Young Theorem

Assume P_k^* which contains the k optimal centers

- Given that $\mu_j \in P_k^*$

$$
\begin{aligned}
f_{k-\text{mean}} &= \sum_{i\in[n]} \min_{j\in[k]} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 \\
&\geq \sum_{i\in[n]} \min_{\boldsymbol{y}_i \in P_k^*} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&\geq \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i \in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&= \min_{P_k} \sum_{i\in[n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 \\
&= f_{PCA}
\end{aligned}
$$

Thus, using the Eckhart-Young Theorem

Assume P_k^* which contains the k optimal centers

- Given that $\mu_j \in P_k^*$

$$
\begin{aligned}
f_{k-\text{mean}} &= \sum_{i\in[n]} \min_{j\in[k]} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 \\
&\geq \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k^*} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&\geq \min_{P_k} \sum_{i\in[n]} \min_{\boldsymbol{y}_i\in P_k} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 \\
&= \min_{P_k} \sum_{i\in[n]} \left\| \boldsymbol{x}_i - P_k \boldsymbol{x}_i \right\|^2 \\
&= f_{PCA}
\end{aligned}
$$

Therefore

Now, consider solving k-means on the points $\boldsymbol{y}_i$ instead

- They are embedded into dimension exactly k by projection P_k

Therefore

Now, consider solving k-means on the points $\boldsymbol{y}_i$ instead

- They are embedded into dimension exactly k by projection P_k

Therefore, given $P\boldsymbol{x}_i = \boldsymbol{y}_i$ and $\widehat{\mu}_j = P\mu_j$

- Where the $\widehat{S}$ and $\widehat{\mu}$ are the assignments and centers of the projected points $\boldsymbol{y}_i$:

$$
\begin{aligned}
\sum_{j\in[k]} \sum_{i\in S_j} \left\| \boldsymbol{x}_i - \mu_j \right\|^2 &\geq \sum_{j\in[k]} \sum_{i\in S_j} \left\| P\boldsymbol{x}_i - P\mu_j \right\|^2 \\
&= \sum_{j\in[k]} \sum_{i\in S_j} \left\| \boldsymbol{y}_i - \widehat{\mu}_j \right\|^2 \\
&\geq \sum_{j\in[k]} \sum_{i\in \widehat{S}_j} \left\| \boldsymbol{y}_i - \widehat{\mu}_j \right\|^2 = f^*_{k\text{-means}}
\end{aligned}
$$

Therefore

Now, consider solving k-means on the points $\boldsymbol{y}_i$ instead

- They are embedded into dimension exactly k by projection P_k

Therefore, given $P\boldsymbol{x}_i = \boldsymbol{y}_i$ and $\widehat{\mu}_j = P\mu_j$

- Where the $\widehat{S}$ and $\widehat{\mu}$ are the assignments and centers of the projected points $\boldsymbol{y}_i$:

$$
\begin{aligned}
\sum_{j\in[k]} \sum_{i\in S_j} \left\|\boldsymbol{x}_i - \mu_j\right\|^2 &\geq \sum_{j\in[k]} \sum_{i\in S_j} \left\|P\boldsymbol{x}_i - P\mu_j\right\|^2 \\
&= \sum_{j\in[k]} \sum_{i\in S_j} \left\|\boldsymbol{y}_i - \widehat{\mu}_j\right\|^2 \\
&\geq \sum_{j\in[k]} \sum_{i\in \widehat{S}_j} \left\|\boldsymbol{y}_i - \widehat{\mu}_j\right\|^2 = f^*_{k\text{-means}}
\end{aligned}
$$

Therefore

Now, consider solving k-means on the points $\boldsymbol{y}_i$ instead

- They are embedded into dimension exactly k by projection P_k

Therefore, given $P\boldsymbol{x}_i = \boldsymbol{y}_i$ and $\widehat{\mu}_j = P\mu_j$

- Where the $\widehat{S}$ and $\widehat{\mu}$ are the assignments and centers of the projected points $\boldsymbol{y}_i$:

$$
\begin{aligned}
\sum_{j\in[k]} \sum_{i\in S_j} \|\boldsymbol{x}_i - \mu_j\|^2 &\geq \sum_{j\in[k]} \sum_{i\in S_j} \|P\boldsymbol{x}_i - P\mu_j\|^2 \\
&= \sum_{j\in[k]} \sum_{i\in S_j} \|\boldsymbol{y}_i - \widehat{\mu}_j\|^2 \\
&\geq \sum_{j\in[k]} \sum_{i\in \widehat{S}_j} \|\boldsymbol{y}_i - \widehat{\mu}_j\|^2 = f^*_{k-\text{means}}
\end{aligned}
$$

Therefore, your best beat

Steps

1. Compute the PCA of the points x_i into dimension k.
2. Solve k-means on the points y_i in dimension k.
3. Output the resulting clusters and centers.

Therefore, your best beat

Steps

1. Compute the PCA of the points x_i into dimension k.
2. Solve k-means on the points y_i in dimension k.
3. Output the resulting clusters and centers.

Therefore, your best beat

Steps

1. Compute the PCA of the points x_i into dimension k.
2. Solve k-means on the points y_i in dimension k.
3. Output the resulting clusters and centers.

Given that

We have that

$$f_{new} = \sum_{j\in[k]} \sum_{i\in S_j^*} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 = *$$

Therefore by the fact that $\boldsymbol{x}_i - \boldsymbol{y}_i$ and $\boldsymbol{y}_i - \mu_j^*$ are perpendiculars

$$* = \sum_{j\in[k]} \sum_{i\in S_j^*} \left\{ \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 + \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2 \right\} = **$$

Finally

$$** = \sum_{i\in[n]} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 + \sum_{j\in[k]} \sum_{i\in S_j^*} \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2$$

Given that

We have that

$$f_{new} = \sum_{j\in[k]} \sum_{i\in S_j^*} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 = *$$

Therefore by the fact that $\boldsymbol{x}_i - \boldsymbol{y}_i$ and $\boldsymbol{y}_i - \mu_j^*$ are perpendiculars

$$* = \sum_{j\in[k]} \sum_{i\in S_j^*} \left\{ \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 + \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2 \right\} = **$$

Finally

$$** = \sum_{i\in[n]} \left\| \boldsymbol{x}_i - \boldsymbol{y}_i \right\|^2 + \sum_{j\in[k]} \sum_{i\in S_j^*} \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2$$

Given that

We have that

$$f_{new} = \sum_{j \in [k]} \sum_{i \in S_j^*} \left\| \boldsymbol{x}_i - \mu_j^* \right\|^2 = *$$

Therefore by the fact that $\boldsymbol{x}_i - \boldsymbol{y}_i$ and $\boldsymbol{y}_i - \mu_j^*$ are perpendiculars

$$* = \sum_{j \in [k]} \sum_{i \in S_j^*} \left\{ \| \boldsymbol{x}_i - \boldsymbol{y}_i \|^2 + \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2 \right\} = **$$

Finally

$$** = \sum_{i \in [n]} \| \boldsymbol{x}_i - \boldsymbol{y}_i \|^2 + \sum_{j \in [k]} \sum_{i \in S_j^*} \left\| \boldsymbol{y}_i - \mu_j^* \right\|^2$$

Therefore, we have

Something Notable

$$f_{PCA} + f^*_{k-means} \leq 2 f_{k-means}$$

Outline

Until now, we have assumed a Euclidean metric space

Important step

- The cluster representatives $m_1, ..., m_k$ in are taken to be the means of the currently assigned clusters.

We can generalize this by using a dissimilarity $D(x_i, x_{i'})$

- By using an explicit optimization with respect to $m_1, ..., m_k$

Until now, we have assumed a Euclidean metric space

Important step

- The cluster representatives $m_1, ..., m_k$ in are taken to be the means of the currently assigned clusters.

We can generalize this by using a dissimilarity $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$

- By using an explicit optimization with respect to $m_1, ..., m_k$

Outline

Algorithm K-meoids

Step 1

- For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_k^* = \arg \min_{\{i|C(i)=k\}} \sum_{C(i')=k} D\left(\boldsymbol{x}_i, \boldsymbol{x}_{i'}\right)$$

  - Then $m_k = \boldsymbol{x}_{i_k^*}\ k = 1, ..., K$ are the current estimates of the cluster centers.

Now

Step 2

- Given a current set of cluster centers $m_1, ..., m_k$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \arg \min_{1 \leq k \leq K} D(\boldsymbol{x}_i, m_k)$$

Iterate over steps 1 and 2

- Until the assignments do not change.

Now

Step 2

- Given a current set of cluster centers $m_1, ..., m_k$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \arg \min_{1 \leq k \leq K} D(\boldsymbol{x}_i, m_k)$$

Iterate over steps 1 and 2

- Until the assignments do not change.

Outline

Complexity

Problem, solving the first step has a complexity for $k = 1, ..., K$

$$O\left(N_k^2\right)$$

Given a set of cluster "centers," $\{i_1, i_2, ..., i_K\}$

- Given the new assignments

$$C(i) = \arg\min_{1 \leq k \leq K} D(x_i, m_k)$$

  - It requires a complexity of $O(KN)$ as before.

Complexity

Problem, solving the first step has a complexity for $k = 1, ..., K$

$$O\left(N_k^2\right)$$

Given a set of cluster "centers," $\{i_1, i_2, ..., i_K\}$

- Given the new assignments

$$C(i) = \arg\min_{1 \leq k \leq K} D(\boldsymbol{x}_i, m_k)$$

  - It requires a complexity of $O(KN)$ as before.

Therefore

We have that

- K-medoids is more computationally intensive than K-means.

Outline

The K-center Problem

The input

It is a set of points with distances represented by a weighted graph $G = (V, V \times V)$.

- Such that minimize maximum distance of every point to a centroid.

The K-center Problem

The input

It is a set of points with distances represented by a weighted graph $G = (V, V \times V)$.

Output

- Select K centroids in G.
- Such that minimize maximum distance of every point to a centroid.

The K-center Problem

The input

It is a set of points with distances represented by a weighted graph $G = (V, V \times V)$.

Output

- Select K centroids in G.
- Such that minimize maximum distance of every point to a centroid.

Theorem (In the general case for any distance)

It is NP-hard to approximate the general K-center problem within any factor α.

The K-center Problem

The input

It is a set of points with distances represented by a weighted graph $G = (V, V \times V)$.

Output

- Select K centroids in G.
- Such that minimize maximum distance of every point to a centroid.

Theorem (In the general case for any distance)

It is NP-hard to approximate the general K-center problem within any factor α.

Therefore

We change the distance to be constrained by

The Triangle Inequality

Given x, y and z

$$L(x,z) \leq L(x,y) + L(y,z)$$

Therefore

We change the distance to be constrained by

The Triangle Inequality

Given $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$

$$L(\boldsymbol{x}, \boldsymbol{z}) \leq L(\boldsymbol{x}, \boldsymbol{y}) + L(\boldsymbol{y}, \boldsymbol{z})$$

Outline

We have a new criterion

Instead of using the K-mean criterion

We use the K-center criterion under the triangle inequality.

New Criterion

The K-center criterion partitions the points into K clusters so as to minimize the maximum distance of any point to its cluster center.

We have a new criterion

Instead of using the K-mean criterion

We use the K-center criterion under the triangle inequality.

New Criterion

The K-center criterion partitions the points into K clusters so as to minimize the maximum distance of any point to its cluster center.

Explanation

Setup

Suppose we have a data set A that contains N objects.

Explanation

Setup

Suppose we have a data set A that contains N objects.

We want the following

We want to partition A into K sets labeled $C_1, C_2, ..., C_K$.

- The smallest value D for which all points in this cluster C_k are:
  - Within distance D of each other
  - Or within distance $\frac{D}{2}$ of some point called the cluster center

Explanation

Setup

Suppose we have a data set A that contains N objects.

We want the following

We want to partition A into K sets labeled $C_1, C_2, ..., C_K$.

Now

Define a cluster size for any cluster C_k as follows.

- The smallest value D for which all points in this cluster C_k are:
  - Within distance D of each other.
  - Or within distance $\frac{D}{2}$ of some point called the cluster center.

Explanation

Setup

Suppose we have a data set A that contains N objects.

We want the following

We want to partition A into K sets labeled $C_1, C_2, ..., C_K$.

Now

Define a cluster size for any cluster C_k as follows.

- The smallest value D for which all points in this cluster C_k are:
  - Within distance D of each other.
  - Or within distance $\frac{D}{2}$ of some point called the cluster center.

Explanation

Setup

Suppose we have a data set A that contains N objects.

We want the following

We want to partition A into K sets labeled $C_1, C_2, ..., C_K$.

Now

Define a cluster size for any cluster C_k as follows.

- The smallest value D for which all points in this cluster C_k are:
  - Within distance D of each other.
  - Or within distance $\frac{D}{2}$ of some point called the cluster center.

Explanation

Setup

Suppose we have a data set A that contains N objects.

We want the following

We want to partition A into K sets labeled $C_1, C_2, ..., C_K$.

Now

Define a cluster size for any cluster C_k as follows.

- The smallest value D for which all points in this cluster C_k are:
  - Within distance D of each other.
  - Or within distance $\frac{D}{2}$ of some point called the cluster center.

Another Way

We have

Another way to define the cluster size:

- D is the maximum pairwise distance between an arbitrary pair of points in the cluster.
- Or twice the maximum distance between a data point and a chosen centroid.

Thus

Denoting the cluster size of C_k by D_k, we have that the cluster size of partition (the way the points are grouped) S by:

$$D = \max_{k=1,\ldots,K} D_k \tag{3}$$

In another words

The cluster size of a partition composed of multiple clusters is the maximum size of these clusters.

Another Way

We have

Another way to define the cluster size:

- D is the maximum pairwise distance between an arbitrary pair of points in the cluster.
- Or twice the maximum distance between a data point and a chosen centroid.

Thus

Denoting the cluster size of C_k by D_k, we have that the cluster size of partition (the way the points are grouped) S by:

$$D = \max_{k=1,\ldots,K} D_k \tag{3}$$

In another words

The cluster size of a partition composed of multiple clusters is the maximum size of these clusters.

Another Way

We have

Another way to define the cluster size:

- D is the maximum pairwise distance between an arbitrary pair of points in the cluster.
- Or twice the maximum distance between a data point and a chosen centroid.

Thus

Denoting the cluster size of C_k by D_k, we have that the cluster size of partition (the way the points are grouped) S by:

$$D = \max_{k=1,\ldots,K} D_k \tag{3}$$

In another words

The cluster size of a partition composed of multiple clusters is the maximum size of these clusters.

Outline

Comparison with K-means

We use the following distance for comparison

In order to compare these methods, we use Euclidean distance.

In K-means we assume the distance between vectors is the squared Euclidean distance

K-means tries to find a partition S that minimizes:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:\boldsymbol{x}_i \in C_k} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \quad (4)$$

Where μ_k is the centroid for cluster C_k

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i \quad (5)$$

Comparison with K-means

We use the following distance for comparison

In order to compare these methods, we use Euclidean distance.

In K-means we assume the distance between vectors is the squared Euclidean distance

K-means tries to find a partition S that minimizes:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:x_i \in C_k} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \tag{4}$$

Where μ_k is the centroid for cluster C_k

$$\mu_k = \frac{1}{N_k} \sum_{i:x_i \in C_k} x_i \tag{5}$$

Comparison with K-means

We use the following distance for comparison

In order to compare these methods, we use Euclidean distance.

In K-means we assume the distance between vectors is the squared Euclidean distance

K-means tries to find a partition S that minimizes:

$$\min_{S} \sum_{k=1}^{N} \sum_{i:x_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{4}$$

Where μ_k is the centroid for cluster C_k

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:x_i \in C_k} \boldsymbol{x}_i \tag{5}$$

Now, what about the K-centers

K-center, on the other hand, minimizes the worst case distance to these centroids

$$\min_{S} \max_{k=1,\ldots,K} \max_{i:x_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \quad (6)$$

Now, what about the K-centers

K-center, on the other hand, minimizes the worst case distance to these centroids

$$\min_{S} \max_{k=1,\ldots,K} \max_{i:x_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{6}$$

Where

$\boldsymbol{\mu}_k$ is called the "centroid", but may not be the mean vector.

- Moreover, among the clusters, only the worst cluster matters, whose farthest data point yields the maximum distance to the centroid comparing with the farthest data points of the other clusters.

Now, what about the K-centers

K-center, on the other hand, minimizes the worst case distance to these centroids

$$\min_{S} \max_{k=1,\dots,K} \max_{i:x_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{6}$$

Where

$\boldsymbol{\mu}_k$ is called the "centroid", but may not be the mean vector.

Properties

- The above objective function shows that for each cluster, only the worst scenario matters, that is, the farthest data point to the centroid.
- Moreover, among the clusters, only the worst cluster matters, whose farthest data point yields the maximum distance to the centroid comparing with the farthest data points of the other clusters.

Now, what about the K-centers

K-center, on the other hand, minimizes the worst case distance to these centroids

$$\min_{S} \max_{k=1,\ldots,K} \max_{i:x_i \in C_k} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k\right) \tag{6}$$

Where

$\boldsymbol{\mu}_k$ is called the "centroid", but may not be the mean vector.

Properties

- The above objective function shows that for each cluster, only the worst scenario matters, that is, the farthest data point to the centroid.
- Moreover, among the clusters, only the worst cluster matters, whose farthest data point yields the maximum distance to the centroid comparing with the farthest data points of the other clusters.

In other words

First

This minimax type of problem is much harder to solve than solving the objective function of k-means.

Another formulation of K-center minimizes the worst case pairwise distance instead of using centroids

$$\min_{S} \max_{k=1,\ldots,K} \max_{i,j:x_i,x_j \in C_k} L\left(x_i, x_j\right) \tag{7}$$

With

$L\left(x_i, x_j\right)$ denotes any distance between a pair of objects in the same cluster.

In other words

First

This minimax type of problem is much harder to solve than solving the objective function of k-means.

Another formulation of K-center minimizes the worst case pairwise distance instead of using centroids

$$\min_{S} \max_{k=1,\ldots,K} \max_{i,j:x_i,x_j \in C_k} L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{7}$$

With

$L\left(x_i, x_j\right)$ denotes any distance between a pair of objects in the same cluster.

In other words

First

This minimax type of problem is much harder to solve than solving the objective function of k-means.

Another formulation of K-center minimizes the worst case pairwise distance instead of using centroids

$$\min_{S} \max_{k=1,\dots,K} \max_{i,j:x_i,x_j \in C_k} L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{7}$$

With

$L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ denotes any distance between a pair of objects in the same cluster.

Outline

Greedy Algorithm for K-Center

Main Idea

The idea behind the Greedy Algorithm is to choose a subset H from the original dataset S consisting of K points that are farthest apart from each other.

Intuition

Since the points in set H are far apart then the worst-case scenario has been taken care of and hopefully the cluster size for the partition is small.

Thus

Each point $h_k \in H$ represents one cluster or subset of points C_k.

Greedy Algorithm for K-Center

Main Idea

The idea behind the Greedy Algorithm is to choose a subset H from the original dataset S consisting of K points that are farthest apart from each other.

Intuition

Since the points in set H are far apart then the worst-case scenario has been taken care of and hopefully the cluster size for the partition is small.

Thus

Each point $h_k \in H$ represents one cluster or subset of points C_k.

Greedy Algorithm for K-Center

Main Idea

The idea behind the Greedy Algorithm is to choose a subset H from the original dataset S consisting of K points that are farthest apart from each other.

Intuition

Since the points in set H are far apart then the worst-case scenario has been taken care of and hopefully the cluster size for the partition is small.

Thus

Each point $\boldsymbol{h}_k \in H$ represents one cluster or subset of points C_k.

Then

Something Notable

We can think of it as a centroid.

However

Technically it is not a centroid because it tends to be at the boundary of a cluster, but conceptually we can think of it as a centroid.

Important

The way that we partition these points given the centroids is the same as in K-means, that is, the nearest-neighbor rule.

Then

Something Notable

We can think of it as a centroid.

However

Technically it is not a centroid because it tends to be at the boundary of a cluster, but conceptually we can think of it as a centroid.

Important

The way that we partition these points given the centroids is the same as in K-means, that is, the nearest-neighbor rule.

Then

Something Notable

We can think of it as a centroid.

However

Technically it is not a centroid because it tends to be at the boundary of a cluster, but conceptually we can think of it as a centroid.

Important

The way that we partition these points given the centroids is the same as in K-means, that is, the nearest-neighbor rule.

Specifically

We do the following

For every point x_i, in order to see which cluster C_k it is partitioned into, we compute its distance to each cluster centroid as follows, and find out which centroid is the closest:

$$L(\boldsymbol{x}_i, \boldsymbol{h}_k) = \min_{k'=1,\ldots,K} L(\boldsymbol{x}_i, \boldsymbol{h}_{k'}) \tag{8}$$

Thus

Whichever centroid with the minimum distance is selected as the cluster for x_i.

Specifically

We do the following

For every point x_i, in order to see which cluster C_k it is partitioned into, we compute its distance to each cluster centroid as follows, and find out which centroid is the closest:

$$L\left(\boldsymbol{x}_i, \boldsymbol{h}_k\right) = \min_{k'=1,\ldots,K} L\left(\boldsymbol{x}_i, \boldsymbol{h}_{k'}\right) \tag{8}$$

Thus

Whichever centroid with the minimum distance is selected as the cluster for x_i.

Important

For K-center clustering

We only need pairwise distance $L(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in S$.

Where

x_i can be a non-vector representation of the objects.

As long we can calculate

$L(x_i, x_j)$ which makes the K-center more general than the K-means.

Important

For K-center clustering

We only need pairwise distance $L(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in S$.

Where

$\boldsymbol{x}_i$ can be a non-vector representation of the objects.

As long we can calculate

$L(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which makes the K-center more general than the K-means.

Important

For K-center clustering

We only need pairwise distance $L(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in S$.

Where

$\boldsymbol{x}_i$ can be a non-vector representation of the objects.

As long we can calculate

$L(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which makes the K-center more general than the K-means.

Properties

Something Notable

The greedy algorithm achieves an approximation factor of 2 as the distance measure L satisfies the triangle inequality.

Thus, we have that

$$D^* = \min_S \max_{k=1,\dots,K} \max_{i,j:x_i,x_j \in C_k} L(x_i, x_j) \tag{9}$$

Then, we have the following guarantee for the greedy algorithm

$$D \leq 2D^* \tag{10}$$

Properties

Something Notable

The greedy algorithm achieves an approximation factor of 2 as the distance measure L satisfies the triangle inequality.

Thus, we have that

$$D^* = \min_{S} \max_{k=1,\dots,K} \max_{i,j:x_i,x_j \in C_k} L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{9}$$

Then, we have the following guarantee for the greedy algorithm

$$D \leq 2D^* \tag{10}$$

Properties

Something Notable

The greedy algorithm achieves an approximation factor of 2 as the distance measure L satisfies the triangle inequality.

Thus, we have that

$$D^* = \min_{S} \max_{k=1,\dots,K} \max_{i,j:x_i,x_j \in C_k} L\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{9}$$

Then, we have the following guarantee for the greedy algorithm

$$D \leq 2D^* \tag{10}$$

Nevertheless

We have that

K-center does not provide a locally optimal solution.

We can get only a solution

Guaranteeing to be within a certain performance range of the theoretical optimal solution.

Nevertheless

We have that

K-center does not provide a locally optimal solution.

We can get only a solution

Guaranteeing to be within a certain performance range of the theoretical optimal solution.

Outline

Setup

First

Set H denotes the set of cluster centroids or cluster of representative objects $\{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\} \subset S$.

Second

Let $cluster\left(\boldsymbol{x}_i\right)$ be the identity of the cluster $\boldsymbol{x}_i \in S$ belongs to.

Third

The distance $dist\left(x_i\right)$ is the distance between x_i and its closest cluster representative object (centroid):

$$dist\left(\boldsymbol{x}_i\right) = \min_{h_j \in H} L\left(\boldsymbol{x}_i, \boldsymbol{h}_j\right) \tag{11}$$

Setup

First

Set H denotes the set of cluster centroids or cluster of representative objects $\{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\} \subset S$.

Second

Let $cluster\left(\boldsymbol{x}_i\right)$ be the identity of the cluster $\boldsymbol{x}_i \in S$ belongs to.

Third

The distance $dist\left(x_i\right)$ is the distance between x_i and its closest cluster representative object (centroid):

$$dist\left(\boldsymbol{x}_i\right) = \min_{\boldsymbol{h}_j \in H} L\left(\boldsymbol{x}_i, \boldsymbol{h}_j\right) \tag{11}$$

Setup

First

Set H denotes the set of cluster centroids or cluster of representative objects $\{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\} \subset S$.

Second

Let $cluster\left(\boldsymbol{x}_i\right)$ be the identity of the cluster $\boldsymbol{x}_i \in S$ belongs to.

Third

The distance $dist\left(x_i\right)$ is the distance between x_i and its closest cluster representative object (centroid):

$$dist\left(\boldsymbol{x}_i\right) = \min_{h_j \in H} L\left(\boldsymbol{x}_i, \boldsymbol{h}_j\right) \tag{11}$$

The Main Idea

Something Notable

We always assign $\boldsymbol{x}_i$ to the closest centroid. Therefore $dist\left(\boldsymbol{x}_i\right)$ is the minimum distance between $\boldsymbol{x}_i$ and any centroid.

The Main Idea

Something Notable

We always assign $\boldsymbol{x}_i$ to the closest centroid. Therefore $dist\left(\boldsymbol{x}_i\right)$ is the minimum distance between $\boldsymbol{x}_i$ and any centroid.

The Algorithm is Iterative

- We generate one cluster centroid first and then add others one by one until we get K clusters.
- The set of centroids H starts with only a single centroid, h_1.

The Main Idea

Something Notable

We always assign $\boldsymbol{x}_i$ to the closest centroid. Therefore $dist\left(\boldsymbol{x}_i\right)$ is the minimum distance between $\boldsymbol{x}_i$ and any centroid.

The Algorithm is Iterative

- We generate one cluster centroid first and then add others one by one until we get K clusters.
- The set of centroids H starts with only a single centroid, $\boldsymbol{h}_1$.

Outline

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j,\ H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j, \ H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\,(\boldsymbol{x}_i) = L\,(\boldsymbol{x}_i, \boldsymbol{h}_1)$.
2. $cluster\,(\boldsymbol{x}_i)$.

- Because so far we only have one cluster, we will assign a cluster label 1 to every x_i.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j,\ H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\left(\boldsymbol{x}_i\right) = L\left(\boldsymbol{x}_i, \boldsymbol{h}_1\right)$.
2. $cluster\left(\boldsymbol{x}_i\right)$.

In other words

- The $dist\left(x_i\right)$ is the computed distance between x_j and h_1.
- Because so far we only have one cluster, we will assign a cluster label 1 to every x_j.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j,\ H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\left(\boldsymbol{x}_i\right) = L\left(\boldsymbol{x}_i, \boldsymbol{h}_1\right)$.
2. $cluster\left(x_i\right)$.

In other words

- The $dist\left(x_i\right)$ is the computed distance between x_j and h_1.
- Because so far we only have one cluster, we will assign a cluster label 1 to every x_j.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j$, $H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\left(\boldsymbol{x}_i\right) = L\left(\boldsymbol{x}_i, \boldsymbol{h}_1\right)$.
2. $cluster\left(\boldsymbol{x}_i\right)$.

In other words

- The $dist\left(x_i\right)$ is the computed distance between x_j and h_1.
- Because so far we only have one cluster, we will assign a cluster label 1 to every x_j.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j, \; H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\left(\boldsymbol{x}_i\right) = L\left(\boldsymbol{x}_i, \boldsymbol{h}_1\right)$.
2. $cluster\left(\boldsymbol{x}_i\right)$.

In other words

- The $dist\left(x_i\right)$ is the computed distance between $\boldsymbol{x}_j$ and $\boldsymbol{h}_1$.
- Because so far we only have one cluster, we will assign a cluster label 1 to every x_j.

Algorithm

Step 1

- Randomly select an object $\boldsymbol{x}_j$ from S, let $\boldsymbol{h}_1 = \boldsymbol{x}_j, \ H = \{\boldsymbol{h}_1\}$.
- It does not matter how x_j is selected.

Step 2

For $j = 1$ to n:

1. $dist\left(\boldsymbol{x}_i\right) = L\left(\boldsymbol{x}_i, \boldsymbol{h}_1\right)$.
2. $cluster\left(\boldsymbol{x}_i\right)$.

In other words

- The $dist\left(x_i\right)$ is the computed distance between $\boldsymbol{x}_j$ and $\boldsymbol{h}_1$.
- Because so far we only have one cluster, we will assign a cluster label 1 to every $\boldsymbol{x}_j$.

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max\limits_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max\limits_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Algorithm

Step 3

For $i = 2$ to K

1. $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$
2. Choose $\boldsymbol{h}_i \in S - H$ such that $dist\left(\boldsymbol{h}_i\right) == D$
3. $H = H \cup \{\boldsymbol{h}_i\}$
4. for $j = 1$ to N
5. if $L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right) \leq dist\left(\boldsymbol{x}_j\right)$
6. $dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, \boldsymbol{h}_i\right)$
7. $cluster\left(\boldsymbol{x}_j\right) = i$

Thus

As the algorithm progresses

- We gradually add more and more cluster centroids, beginning with 2 until we get to K.
- At each iteration, we find among all of the points which are not yet included in the set, a worst point:
  - Worst in the sense that this point has maximum distance to its corresponding centroid.

Thus

As the algorithm progresses

- We gradually add more and more cluster centroids, beginning with 2 until we get to K.
- At each iteration, we find among all of the points which are not yet included in the set, a worst point:
  - Worst in the sense that this point has maximum distance to its corresponding centroid.

This worst point

- It is added to the set H.
- To stress gain, points already included in H are not among the consideration.

Thus

As the algorithm progresses

- We gradually add more and more cluster centroids, beginning with 2 until we get to K.
- At each iteration, we find among all of the points which are not yet included in the set, a worst point:
  - Worst in the sense that this point has maximum distance to its corresponding centroid.

This worst point

- It is added to the set H.
- To stress gain, points already included in H are not among the consideration.

Thus

As the algorithm progresses

- We gradually add more and more cluster centroids, beginning with 2 until we get to K.
- At each iteration, we find among all of the points which are not yet included in the set, a worst point:
  - Worst in the sense that this point has maximum distance to its corresponding centroid.

This worst point

- It is added to the set H.
- To stress gain, points already included in H are not among the consideration.

Outline

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Of course it is necessary to have extra fields for the necessary distances

Thus, each node-element for the sets must have a field $dist(x_j)$ such that

$$dist(x_j) = L(x_j, Find(x_j)) \tag{12}$$

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Of course it is necessary to have extra fields for the necessary distances

Thus, each node-element for the sets must have a field $dist(x_j)$ such that

$$dist(x_j) = L(x_j, Find(x_j)) \quad (12)$$

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Of course it is necessary to have extra fields for the necessary distances

Thus, each node-element for the sets must have a field $dist(x_j)$ such that

$$dist(x_j) = L(x_j, Find(x_j)) \quad (12)$$

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Of course it is necessary to have extra fields for the necessary distances

Thus, each node-element for the sets must have a field $dist(x_j)$ such that

$$dist(x_j) = L(x_j, Find(x_j)) \quad (12)$$

Implementation

We can use the following for implementation

- The disjoint-set data structure with the following operations:
  - MakeSet
  - Find
  - Union
  - Remove Iteratively through the disjoint trees.

Of course it is necessary to have extra fields for the necessary distances

Thus, each node-element for the sets must have a field $dist\left(\boldsymbol{x}_j\right)$ such that

$$dist\left(\boldsymbol{x}_j\right) = L\left(\boldsymbol{x}_j, Find\left(\boldsymbol{x}_j\right)\right) \quad (12)$$

Although

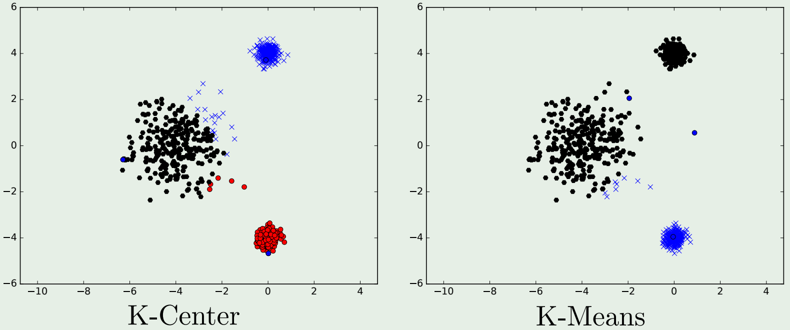
Other things need to be taken in consideration

I will allow to you to think about them

Outline

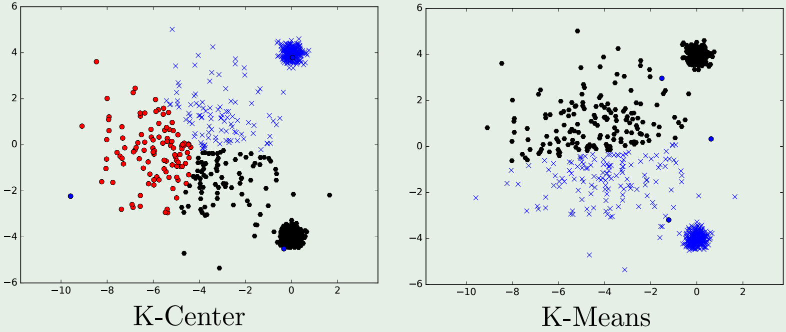
Example

Running the k-center and k-means algorithms allows to see that for different densities k-center is more robust



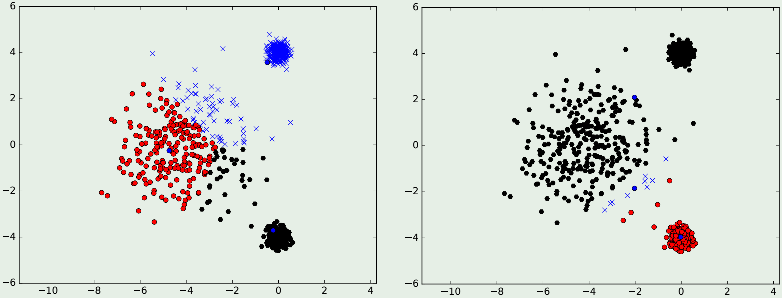
Example

Decreasing the density of one of the clusters, we see a degradation on the clusters



Using Centroids of the K-center to initialize K-mean

Thus, we can use the centroids of K-center to try to improve upon K-means to a certain degree



K-Means using the K-Center Centroids

K-Means using random centroids

Outline

The Running Time

We have that

The running time of the algorithm is $O(KN)$, where K is the number of clusters generated and N is the size of the data set.

Why?

Because K-center only requires pairwise distance between any point and the centroids.

The Running Time

We have that

The running time of the algorithm is $O(KN)$, where K is the number of clusters generated and N is the size of the data set.

Why?

Because K-center only requires pairwise distance between any point and the centroids.

Main Bound of the K-center algorithm

Lemma

Given the distance measure L satisfying the **triangle inequality**.

- If the partition obtained by the greedy algorithm is $\tilde{S}$ and the optimal partition be S^*, such that the cluster size of $\tilde{S}$ be $\tilde{D}$ and the one for S^* is D^*, then

$$\tilde{D} \leq 2D^* \tag{13}$$

Main Bound of the K-center algorithm

Lemma

Given the distance measure L satisfying the **triangle inequality**.

- If the partition obtained by the greedy algorithm is $\widetilde{S}$ and the optimal partition be S^*, such that the cluster size of $\widetilde{S}$ be $\widetilde{D}$ and the one for S^* is D^*, then

$$\widetilde{D} \leq 2D^* \quad (13)$$

Main Bound of the K-center algorithm

Lemma

Given the distance measure L satisfying the **triangle inequality**.

- If the partition obtained by the greedy algorithm is $\widetilde{S}$ and the optimal partition be S^*, such that the cluster size of $\widetilde{S}$ be $\widetilde{D}$ and the one for S^* is D^*, then

$$\widetilde{D} \leq 2D^* \tag{13}$$

Outline

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Thus

$$D_1 \geq D_2 \geq D_3 \ldots \quad (14)$$

- You can prove that part by yourselves.

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Thus

$$D_1 \geq D_2 \geq D_3 \dots \tag{14}$$

Why?

- Because $D = \max_{x_j : x_j \in S-H} dist(x_j)$ and lines 5-7 in the step 3 and using induction!!!
- You can prove that part by yourselves...

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Thus

$$D_1 \geq D_2 \geq D_3 ... \tag{14}$$

Why?

- Because $D = \max_{x_j : x_j \in S-H} dist(x_j)$ and lines 5-7 in the step 3 and using induction!!!
- You can prove that part by yourselves..

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Thus

$$D_1 \geq D_2 \geq D_3 ... \tag{14}$$

Why?

- Because $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S - H} dist\left(\boldsymbol{x}_j\right)$ and lines 5-7 in the step 3 and using induction!!!
- You can prove that part by yourselves..

Proof

If we look only at the first j centroid

It generates a partition j with size D_j and also:

- The cluster size is the size of the biggest cluster in the current partition.
- The size of every cluster is defined as the maximum distance between a point and the cluster centroid

Thus

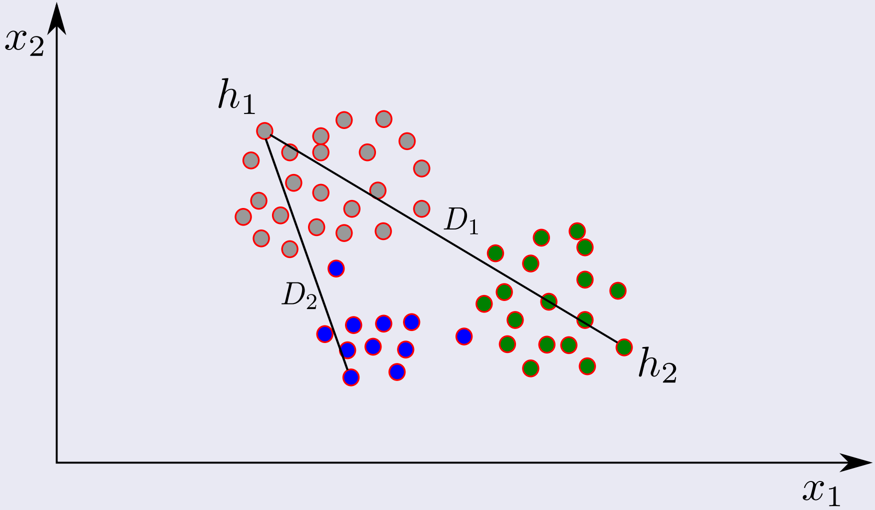
$$D_1 \geq D_2 \geq D_3 ... \tag{14}$$

Why?

- Because $D = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in S-H} dist\left(\boldsymbol{x}_j\right)$ and lines 5-7 in the step 3 and using induction!!!
- You can prove that part by yourselves..

Graphically

It is easy to see that when the inner loop changes distances



Now

It is necessary to prove

$$\forall i < j,\ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \geq D_{j-1} \tag{15}$$

Thus

D_{j-1} is a lower bound for the distance between h_i and h_j.

Now

It is necessary to prove

$$\forall i < j,\ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \geq D_{j-1} \tag{15}$$

Thus

D_{j-1} is a lower bound for the distance between h_i and h_j.

Now

It is necessary to prove

$$\forall i < j,\ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \geq D_{j-1} \tag{15}$$

Thus

D_{j-1} is a lower bound for the distance between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$.

Proof of the Previous Statement

It is possible to see that

$$L\left(\boldsymbol{h}_{j-2}, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right) \quad (16)$$

How?

Assume it is not true. Then, $L\left(\boldsymbol{h}_{j-2}, \boldsymbol{h}_j\right) < L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right)$

Proof of the Previous Statement

It is possible to see that

$$L\left(\boldsymbol{h}_{j-2}, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right) \tag{16}$$

How?

Assume it is not true. Then, $L\left(\boldsymbol{h}_{j-2}, \boldsymbol{h}_j\right) < L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right)$

Then

We have that given the partition $\widetilde{S}$ generated by the greedy algorithm

- $\boldsymbol{h}_{j-2} \in \widetilde{C}_{j-2}$, $\boldsymbol{h}_{j-2} \in \widetilde{C}_{j}$ for $\widetilde{C}_{j-2}, \widetilde{C}_{j} \in \widetilde{S}$
- It is a contradiction because $\boldsymbol{h}_{j-2}$ is generated by the algorithm such that cannot be in any other cluster!!!

Thus iteratively

$$L\left(\boldsymbol{h}_1, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_2, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_3, \boldsymbol{h}_j\right) \geq \ldots \geq L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right) = D_{j-1} \quad (17)$$

Then

We have that given the partition $\widetilde{S}$ generated by the greedy algorithm

- $\boldsymbol{h}_{j-2} \in \widetilde{C}_{j-2}$, $\boldsymbol{h}_{j-2} \in \widetilde{C}_{j}$ for $\widetilde{C}_{j-2}, \widetilde{C}_{j} \in \widetilde{S}$
- It is a contradiction because $\boldsymbol{h}_{j-2}$ is generated by the algorithm such that cannot be in any other cluster!!!

Thus iteratively

$$L\left(\boldsymbol{h}_1, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_2, \boldsymbol{h}_j\right) \geq L\left(\boldsymbol{h}_3, \boldsymbol{h}_j\right) \geq ... \geq L\left(\boldsymbol{h}_{j-1}, \boldsymbol{h}_j\right) = D_{j-1} \quad (17)$$

Not only that

Not only that

$$\forall j, \exists i < j, \ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = D_{j-1} \quad (18)$$

Therefore

Therefore, D_{j-1} is not only the lower bound for the distance between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, it is also the exact boundary for a specific i.

Not only that

Not only that

$$\forall j, \exists i < j,\ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = D_{j-1} \tag{18}$$

Therefore

Therefore, D_{j-1} is not only the lower bound for the distance between h_i and h_j, it is also the exact boundary for a specific i.

Not only that

Not only that

$$\forall j, \exists i < j,\ L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = D_{j-1} \tag{18}$$

Therefore

Therefore, D_{j-1} is not only the lower bound for the distance between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, it is also the exact boundary for a specific i.

If, we continue with the proof

Now

- Let us consider the optimal partition S^* with K clusters and its size D^*.
  - **Suppose the greedy algorithm generates** the centroids $\widetilde{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K\}$.
  - For the proof, we are adding one more, $\boldsymbol{h}_{K+1}$.
  - This can be done without losing generality.

If, we continue with the proof

Now

- Let us consider the optimal partition S^* with K clusters and its size D^*.
  - **Suppose the greedy algorithm generates** the centroids $\widetilde{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K\}$.
  - For the proof, we are adding one more, $\boldsymbol{h}_{K+1}$.
  - This can be done without losing generality.

According to the pigeonhole principle, at least two of the centroids among $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K, \boldsymbol{h}_{K+1}\}$ will fall into one cluster k of the partition S^*.

If, we continue with the proof

Now

- Let us consider the optimal partition S^* with K clusters and its size D^*.
  - **Suppose the greedy algorithm generates** the centroids $\widetilde{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K\}$.
  - For the proof, we are adding one more, $\boldsymbol{h}_{K+1}$.
  - This can be done without losing generality.

According to the pigeonhole principle, at least two of the centroids among

$\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K, \boldsymbol{h}_{K+1}\}$will fall into one cluster k of the partition S^*.

Thus assume

$1 \leq i < k < j \leq K+1 \Rightarrow$ Using the triangle inequality:

$$L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq L(\boldsymbol{h}_i, \boldsymbol{h}_k) + L(\boldsymbol{h}_k, \boldsymbol{h}_j) \leq D^* + D^* = 2D^* \quad (19)$$

If, we continue with the proof

Now

- Let us consider the optimal partition S^* with K clusters and its size D^*.
    - **Suppose the greedy algorithm generates** the centroids $\widetilde{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K\}$.
    - For the proof, we are adding one more, $\boldsymbol{h}_{K+1}$.
    - This can be done without losing generality.

According to the pigeonhole principle, at least two of the centroids among

$\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K, \boldsymbol{h}_{K+1}\}$will fall into one cluster k of the partition S^*.

Thus assume

$1 \leq i < k < j \leq K+1 \Rightarrow$ Using the triangle inequality:

$$L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq L(\boldsymbol{h}_i, \boldsymbol{h}_k) + L(\boldsymbol{h}_k, \boldsymbol{h}_j) \leq D^* + D^* = 2D^* \quad (19)$$

If, we continue with the proof

Now

- Let us consider the optimal partition S^* with K clusters and its size D^*.
  - **Suppose the greedy algorithm generates** the centroids $\widetilde{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K\}$.
  - For the proof, we are adding one more, $\boldsymbol{h}_{K+1}$.
  - This can be done without losing generality.

According to the pigeonhole principle, at least two of the centroids among

$\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K, \boldsymbol{h}_{K+1}\}$will fall into one cluster k of the partition S^*.

Thus assume

$1 \leq i < k < j \leq K + 1 \Rightarrow$ Using the triangle inequality:

$$L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq L(\boldsymbol{h}_i, \boldsymbol{h}_k) + L(\boldsymbol{h}_k, \boldsymbol{h}_j) \leq D^* + D^* = 2D^* \qquad (19)$$

Then

In addition

Also $L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \geq D_{j-1} \geq D_k$ then $D_k \leq 2D^*$

Given $\widetilde{S}$, the partition generated by the greedy algorithm

We define Δ as

$$\Delta = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in \widetilde{S} - \widetilde{H}} \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{x}_j, \boldsymbol{h}_k\right) \tag{20}$$

Basically

The maximum of all points that are not centroids that minimize the distance to some centroid for the partition generated by the greedy algorithm.

Then

In addition

Also $L\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \geq D_{j-1} \geq D_k$ then $D_k \leq 2D^*$

Given $\widetilde{S}$, the partition generated by the greedy algorithm

We define Δ as

$$\Delta = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in \widetilde{S} - \widetilde{H}} \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{x}_j, \boldsymbol{h}_k\right) \tag{20}$$

Basically

The maximum of all points that are not centroids that minimize the distance to some centroid for the partition generated by the greedy algorithm.

Then

In addition

Also $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \geq D_{j-1} \geq D_k$ then $D_k \leq 2D^*$

Given $\widetilde{S}$, the partition generated by the greedy algorithm

We define Δ as

$$\Delta = \max_{\boldsymbol{x}_j : \boldsymbol{x}_j \in \widetilde{S} - \widetilde{H}} \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L(\boldsymbol{x}_j, \boldsymbol{h}_k) \tag{20}$$

Basically

The maximum of all points that are not centroids that minimize the distance to some centroid for the partition generated by the greedy algorithm.

Now, we are ready for the final part

Let $\boldsymbol{h}_{K+1}$ be an element in $\widetilde{S} - \widetilde{H}$

Such that

$$\min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) = \Delta \tag{21}$$

By definition

$$L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \geq \Delta, \ \forall k = 1, ..., K \tag{22}$$

Thus, we have the following sets

Let $H_k = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\}$ with $k = 1, 2, ..., K$.

Now, we are ready for the final part

Let $\boldsymbol{h}_{K+1}$ be an element in $\widetilde{S} - \widetilde{H}$

Such that

$$\min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) = \Delta \tag{21}$$

By definition

$$L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \geq \Delta, \ \forall k = 1, ..., K \tag{22}$$

Thus, we have the following sets

Let $H_k = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\}$ with $k = 1, 2, ..., K$.

Now, we are ready for the final part

Let $\boldsymbol{h}_{K+1}$ be an element in $\widetilde{S} - \widetilde{H}$

Such that

$$\min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) = \Delta \tag{21}$$

By definition

$$L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \geq \Delta, \ \forall k = 1, ..., K \tag{22}$$

Thus, we have the following sets

Let $H_k = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_k\}$ with $k = 1, 2, ..., K$.

Now

Consider the distance between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ for $i < j \leq K$

- According the greedy algorithm
$$\min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_j, \boldsymbol{h}_k\right) \geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{x}_l, \boldsymbol{h}_k\right) \text{ for any } x_l \in \widetilde{S} - H_j$$
- Basically remember that the $\boldsymbol{h}_i$ are obtained by finding the farthest points.

Now, we have

Since $\boldsymbol{h}_{K+1} \in \widetilde{S} - \widetilde{H}$ and $\widetilde{S} - \widetilde{H} \subset \widetilde{S} - H_j$

$$
\begin{aligned}
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) &\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_j, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&= \Delta
\end{aligned}
$$

Now, we have

Since $\boldsymbol{h}_{K+1} \in \widetilde{S} - \widetilde{H}$ and $\widetilde{S} - \widetilde{H} \subset \widetilde{S} - H_j$

$$
\begin{aligned}
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) &\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_j, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&= \Delta
\end{aligned}
$$

We have shown that for any for $i < j \leq K + 1$

$$
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) \geq \Delta \tag{23}
$$

Now, we have

Since $\boldsymbol{h}_{K+1} \in \widetilde{S} - \widetilde{H}$ and $\widetilde{S} - \widetilde{H} \subset \widetilde{S} - H_j$

$$
\begin{aligned}
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) &\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_j, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
&= \Delta
\end{aligned}
$$

We have shown that for any for $i < j \leq K+1$

$$
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) \geq \Delta \tag{23}
$$

Now, we have

Since $\boldsymbol{h}_{K+1} \in \widetilde{S} - \widetilde{H}$ and $\widetilde{S} - \widetilde{H} \subset \widetilde{S} - H_j$

$$
\begin{aligned}
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) \geq & \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_j, \boldsymbol{h}_k\right) \\
\geq & \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
\geq & \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L\left(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k\right) \\
= & \Delta
\end{aligned}
$$

We have shown that for any for $i < j \leq K + 1$

$$
L\left(\boldsymbol{h}_j, \boldsymbol{h}_i\right) \geq \Delta \tag{23}
$$

Now, we have

Since $\boldsymbol{h}_{K+1} \in \widetilde{S} - \widetilde{H}$ and $\widetilde{S} - \widetilde{H} \subset \widetilde{S} - H_j$

$$
\begin{aligned}
L(\boldsymbol{h}_j, \boldsymbol{h}_i) &\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L(\boldsymbol{h}_j, \boldsymbol{h}_k) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in H_{j-1}} L(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k) \\
&\geq \min_{\boldsymbol{h}_k : \boldsymbol{h}_k \in \widetilde{H}} L(\boldsymbol{h}_{K+1}, \boldsymbol{h}_k) \\
&= \Delta
\end{aligned}
$$

We have shown that for any for $i < j \leq K+1$

$$
L(\boldsymbol{h}_j, \boldsymbol{h}_i) \geq \Delta \tag{23}
$$

Now

Consider the optimal partition $S^* = \{C_1^*, C_2^*, ..., C_K^*\}$

Thus at least 2 of the $K+1$ elements $h_1, h_2, ..., h_{K+1}$ will be covered by one cluster.

Assume that

h_i and h_j belong to the same cluster in S^* . Then $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq D^*$.

In addition

We have that since $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \geq \Delta$ then $\Delta \leq D^*$

Now

Consider the optimal partition $S^* = \{C_1^*, C_2^*, ..., C_K^*\}$

Thus at least 2 of the $K+1$ elements $h_1, h_2, ..., h_{K+1}$ will be covered by one cluster.

Assume that

$\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ belong to the same cluster in S^* . Then $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq D^*$.

In addition

We have that since $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \geq \Delta$ then $\Delta \leq D^*$

Now

Consider the optimal partition $S^* = \{C_1^*, C_2^*, ..., C_K^*\}$

Thus at least 2 of the $K+1$ elements $h_1, h_2, ..., h_{K+1}$ will be covered by one cluster.

Assume that

$\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ belong to the same cluster in S^* . Then $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \leq D^*$.

In addition

We have that since $L(\boldsymbol{h}_i, \boldsymbol{h}_j) \geq \Delta$ then $\Delta \leq D^*$

In addition

Consider elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in any cluster represented by $\boldsymbol{h}_k$

$$L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) \leq \Delta \text{ and } L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq \Delta \tag{24}$$

By Triangle Inequality

$$L\left(\boldsymbol{x}_m, \boldsymbol{x}_k\right) \leq L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) + L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq 2\Delta \tag{25}$$

Finally, there are two elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in a cluster such that $\widetilde{D} = L\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$

$$\widetilde{D} = \max_k D_k \leq 2\Delta \leq 2D^* \tag{26}$$

In addition

Consider elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in any cluster represented by $\boldsymbol{h}_k$

$$L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) \leq \Delta \text{ and } L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq \Delta \tag{24}$$

By Triangle Inequality

$$L\left(\boldsymbol{x}_m, \boldsymbol{x}_k\right) \leq L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) + L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq 2\Delta \tag{25}$$

Finally, there are two elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in a cluster such that $\widetilde{D} = L\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$

$$\widetilde{D} = \max_k D_k \leq 2\Delta \leq 2D^* \tag{26}$$

In addition

Consider elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in any cluster represented by $\boldsymbol{h}_k$

$$L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) \leq \Delta \text{ and } L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq \Delta \tag{24}$$

By Triangle Inequality

$$L\left(\boldsymbol{x}_m, \boldsymbol{x}_k\right) \leq L\left(\boldsymbol{x}_m, \boldsymbol{h}_k\right) + L\left(\boldsymbol{x}_n, \boldsymbol{h}_k\right) \leq 2\Delta \tag{25}$$

Finally, there are two elements $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ in a cluster such that $\widetilde{D} = L\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$

$$\widetilde{D} = \max_k \; D_k \leq 2\Delta \leq 2D^* \tag{26}$$

Outline

Some of the Fuzzy Clustering Models

Fuzzy Clustering Model

Bezdek, 1981

Possibilistic Clustering Model

Krishnapuram - Keller, 1993

Fuzzy Possibilistic Clustering Model

N. Pal - K. Pal - Bezdek, 1997

Some of the Fuzzy Clustering Models

Fuzzy Clustering Model

Bezdek, 1981

Possibilistic Clustering Model

Krishnapuram - Keller, 1993

Fuzzy Possibilistic Clustering Model

N. Pal - K. Pal - Bezdek, 1997

Some of the Fuzzy Clustering Models

Fuzzy Clustering Model

Bezdek, 1981

Possibilistic Clustering Model

Krishnapuram - Keller, 1993

Fuzzy Possibilistic Clustering Model

N. Pal - K. Pal - Bezdek, 1997

Fuzzy C-Means Clustering

The input an unlabeled data set

- $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_N\}$.
- $\boldsymbol{x}_k \in \mathbb{R}^p$

Output

- A partition S of the X as a matrix U of $C \times N$.
- Set of cluster centers $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_C\} \subset \mathbb{R}^p$

Fuzzy C-Means Clustering

The input an unlabeled data set

- $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_N\}$.
- $\boldsymbol{x}_k \in \mathbb{R}^p$

Output

- A partition $\mathcal{S}$ of the X as a matrix U of $C \times N$.
- Set of cluster centers $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_C\} \subset \mathbb{R}^p$

What we want

Creation of the Cost Function

First:

- We can use a distance defined as:

$$\|\boldsymbol{x}_k - \boldsymbol{v}_i\| = \sqrt{(\boldsymbol{x}_k - \boldsymbol{v}_i)^T (\boldsymbol{x}_k - \boldsymbol{v}_i)} \tag{27}$$

The euclidean distance from a point k to a centroid i.
NOTE other distances based in Mahalonobis can be taken in consideration.

What we want

Creation of the Cost Function

First:

- We can use a distance defined as:

$$\|\boldsymbol{x}_k - \boldsymbol{v}_i\| = \sqrt{(\boldsymbol{x}_k - \boldsymbol{v}_i)^T (\boldsymbol{x}_k - \boldsymbol{v}_i)} \tag{27}$$

The euclidean distance from a point k to a centroid i.

NOTE other distances based in Mahalonobis can be taken in consideration.

What we want

Creation of the Cost Function

First:

- We can use a distance defined as:

$$\|\boldsymbol{x}_k - \boldsymbol{v}_i\| = \sqrt{(\boldsymbol{x}_k - \boldsymbol{v}_i)^T (\boldsymbol{x}_k - \boldsymbol{v}_i)} \tag{27}$$

The euclidean distance from a point k to a centroid i.
NOTE other distances based in Mahalonobis can be taken in consideration.

Do you remember the cost function for K-means?

Finding a partition S that minimizes the following function

$$\min_{S} \sum_{k=1}^{N} \sum_{k:\boldsymbol{x}_k \in C_i} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{28}$$

Where $\boldsymbol{v}_i = \frac{1}{N_i} \sum_{\boldsymbol{x}_k \in C_i} \boldsymbol{x}_k$

We can rewrite the previous equation as

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} I\left(\boldsymbol{x}_k \in C_i\right) \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{29}$$

Do you remember the cost function for K-means?

Finding a partition S that minimizes the following function

$$\min_{S} \sum_{k=1}^{N} \sum_{k:\boldsymbol{x}_k \in C_i} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{28}$$

Where $\boldsymbol{v}_i = \frac{1}{N_i} \sum_{\boldsymbol{x}_k \in C_i} \boldsymbol{x}_k$

We can rewrite the previous equation as

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} I\left(x_k \in C_i\right) \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{29}$$

In addition

Did you notice that the membership is always one or zero?

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} \overbrace{I\left(x_k \in C_i\right)}^{\text{Membership}} \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{30}$$

Outline

Thus, we can rethink the membership using something "Fuzzy"

What if we modify the cost function to something like this

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} \overbrace{\text{Fuzzy Value}}^{\text{Membership}} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{31}$$

This means that we think that each cluster C_i is "Fuzzy"

We can assume a fuzzy set for the cluster C_i with membership function:

Thus, we can rethink the membership using something "Fuzzy"

What if we modify the cost function to something like this

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} \overbrace{\text{Fuzzy Value}}^{\text{Membership}} \left\Vert \boldsymbol{x}_k - \boldsymbol{v}_i \right\Vert^2 \tag{31}$$

This means that we think that each cluster C_i is "Fuzzy"

We can assume a fuzzy set for the cluster C_i with membership function:

$$A_i : \mathbb{R}^p \rightarrow [0,1] \tag{32}$$

Such that we can tune it by using a power i.e. decreasing it by a m power.

Thus, we can rethink the membership using something "Fuzzy"

What if we modify the cost function to something like this

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} \overbrace{\text{Fuzzy Value}}^{\text{Membership}} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \quad (31)$$

This means that we think that each cluster C_i is "Fuzzy"

We can assume a fuzzy set for the cluster C_i with membership function:

$$A_i : \mathbb{R}^p \to [0, 1] \quad (32)$$

Such that we can tune it by using a power i.e. decreasing it by a m power.

Thus, we can rethink the membership using something "Fuzzy"

What if we modify the cost function to something like this

$$\min_{S} \sum_{k=1}^{N} \sum_{i=1}^{C} \overbrace{\text{Fuzzy Value}}^{\text{Membership}} \left\Vert \boldsymbol{x}_k - \boldsymbol{v}_i \right\Vert^2 \tag{31}$$

This means that we think that each cluster C_i is "Fuzzy"

We can assume a fuzzy set for the cluster C_i with membership function:

$$A_i : \mathbb{R}^p \to [0, 1] \tag{32}$$

Such that we can tune it by using a power i.e. decreasing it by a m power.

Under the following constraints

First

$$A_i\left(\boldsymbol{x}_k\right) \in [0,1] \;\; \forall i, k \tag{33}$$

Second

$$0 < \sum_{k=1}^{N} A_i\left(\boldsymbol{x}_k\right) < N \;\; \forall i \tag{34}$$

Third

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \;\; \forall k \tag{35}$$

Under the following constraints

First

$$A_i\left(\boldsymbol{x}_k\right) \in [0,1] \;\; \forall i, k \tag{33}$$

Second

$$0 < \sum_{k=1}^{N} A_i\left(\boldsymbol{x}_k\right) < N \;\forall i \tag{34}$$

Third

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \;\forall k \tag{35}$$

Under the following constraints

First

$$A_i\left(\boldsymbol{x}_k\right) \in [0, 1] \;\; \forall i, k \tag{33}$$

Second

$$0 < \sum_{k=1}^{N} A_i\left(\boldsymbol{x}_k\right) < N \;\forall i \tag{34}$$

Third

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \;\forall k \tag{35}$$

Final Cost Function

Properties

$$J_m(\mathcal{S}) = \sum_{k=1}^{N} \sum_{i=1}^{C} \left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{36}$$

- $\sum_{i=1}^{C} A_i(\boldsymbol{x}_k) = 1$, for $1 \leq k \leq N$
- $0 < \sum_{k=1}^{N} A_i(\boldsymbol{x}_k) < n$, for $1 \leq i \leq C$.
- $m > 1$.

Final Cost Function

Properties

$$J_m\left(\mathcal{S}\right)=\sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i\left(\boldsymbol{x}_k\right)\right]^m\left\Vert \boldsymbol{x}_k-\boldsymbol{v}_i\right\Vert^2 \tag{36}$$

Under the constraints

- $A_i\left(\boldsymbol{x}_k\right)\in[0,1]$, for $1\leq k\leq N$ and $1\leq i\leq C$.
- $\sum_{i=1}^{C}A_i\left(\boldsymbol{x}_k\right)=1$, for $1\leq k\leq N$.
- $0<\sum_{k=1}^{N}A_i\left(\boldsymbol{x}_k\right)<n$, for $1\leq i\leq C$.
- $m>1$.

Final Cost Function

Properties

$$J_m(\mathcal{S}) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \quad (36)$$

Under the constraints

- $A_i(\boldsymbol{x}_k) \in [0,1]$, for $1 \leq k \leq N$ and $1 \leq i \leq C$.
- $\sum_{i=1}^{C} A_i(\boldsymbol{x}_k) = 1$, for $1 \leq k \leq N$.
- $0 < \sum_{k=1}^{N} A_i(\boldsymbol{x}_k) < n$, for $1 \leq i \leq C$.
- $m > 1$.

Final Cost Function

Properties

$$J_m(\mathcal{S}) = \sum_{k=1}^{N} \sum_{i=1}^{C} \left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{36}$$

Under the constraints

- $A_i(\boldsymbol{x}_k) \in [0,1]$, for $1 \leq k \leq N$ and $1 \leq i \leq C$.
- $\sum_{i=1}^{C} A_i(\boldsymbol{x}_k) = 1$, for $1 \leq k \leq N$.
- $0 < \sum_{k=1}^{N} A_i(\boldsymbol{x}_k) < n$, for $1 \leq i \leq C$.
- $m > 1$.

Final Cost Function

Properties

$$J_m\left(\mathcal{S}\right) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \quad (36)$$

Under the constraints

- $A_i\left(\boldsymbol{x}_k\right) \in [0,1]$, for $1 \leq k \leq N$ and $1 \leq i \leq C$.
- $\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1$, for $1 \leq k \leq N$.
- $0 < \sum_{k=1}^{N} A_i\left(\boldsymbol{x}_k\right) < n$, for $1 \leq i \leq C$.
- $m > 1$.

Outline

Using the Lagrange Multipliers

New cost function

$$\bar{J}_m(\mathcal{S}) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \sum_{k=1}^{N}\lambda_k\left[\sum_{i=1}^{C}A_i(\boldsymbol{x}_k) - 1\right] \quad (37)$$

Derive with respect to $A_i(\boldsymbol{x}_k)$

$$\frac{\partial \bar{J}_m(\mathcal{S})}{\partial A_i(\boldsymbol{x}_k)} = mA_i(\boldsymbol{x}_k)^{m-1}\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \lambda_k = 0 \quad (38)$$

Thus

$$A_i(\boldsymbol{x}_k) = \left[\frac{\lambda_k}{m\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}\right]^{\frac{1}{m-1}} \quad (39)$$

Using the Lagrange Multipliers

New cost function

$$\bar{J}_m(\mathcal{S}) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \sum_{k=1}^{N}\lambda_k\left[\sum_{i=1}^{C}A_i(\boldsymbol{x}_k) - 1\right] \quad (37)$$

Derive with respect to $A_i(\boldsymbol{x}_k)$

$$\frac{\partial \bar{J}_m(\mathcal{S})}{\partial A_i(x_k)} = mA_i(\boldsymbol{x}_k)^{m-1}\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \lambda_k = 0 \quad (38)$$

Thus

$$A_i(\boldsymbol{x}_k) = \left[\frac{\lambda_k}{m\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}\right]^{\frac{1}{m-1}} \quad (39)$$

Using the Lagrange Multipliers

New cost function

$$\bar{J}_m(\mathcal{S}) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[A_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \sum_{k=1}^{N}\lambda_k\left[\sum_{i=1}^{C}A_i(\boldsymbol{x}_k) - 1\right] \quad (37)$$

Derive with respect to $A_i(\boldsymbol{x}_k)$

$$\frac{\partial \bar{J}_m(\mathcal{S})}{\partial A_i(x_k)} = mA_i(\boldsymbol{x}_k)^{m-1}\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 - \lambda_k = 0 \quad (38)$$

Thus

$$A_i(\boldsymbol{x}_k) = \left[\frac{\lambda_k}{m\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}\right]^{\frac{1}{m-1}} \quad (39)$$

Using the Lagrange Multipliers

Sum over all i's

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = \frac{\lambda_k^{\frac{1}{m-1}}}{m^{\frac{1}{m-1}}\left\Vert\boldsymbol{x}_k-\boldsymbol{v}_i\right\Vert^{\frac{2}{m-1}}} \tag{40}$$

Thus

$$\lambda_k = \frac{m}{\left[\sum_{i=1}^{C}\frac{1}{\left\Vert\boldsymbol{x}_k-\boldsymbol{v}_i\right\Vert^{\frac{2}{m-1}}}\right]^{m-1}} \tag{41}$$

Plug Back on equation 38 using j instead of i

$$\frac{m}{\left[\sum_{j=1}^{C}\frac{1}{\left\Vert\boldsymbol{x}_k-\boldsymbol{v}_j\right\Vert^{\frac{2}{m-1}}}\right]^{m-1}} = mA_i\left(\boldsymbol{x}_k\right)^{m-1}\left\Vert\boldsymbol{x}_k-\boldsymbol{v}_i\right\Vert^{2} \tag{42}$$

Using the Lagrange Multipliers

Sum over all i's

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = \frac{\lambda_k^{\frac{1}{m-1}}}{m^{\frac{1}{m-1}} \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^{\frac{2}{m-1}}} \tag{40}$$

Thus

$$\lambda_k = \frac{m}{\left[\sum_{i=1}^{C} \frac{1}{\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^{\frac{2}{m-1}}}\right]^{m-1}} \tag{41}$$

Plug Back on equation 38 using j instead of i

$$\frac{m}{\left[\sum_{j=1}^{C} \frac{1}{\left\|\boldsymbol{x}_k - \boldsymbol{v}_j\right\|^{\frac{2}{m-1}}}\right]^{m-1}} = mA_i\left(\boldsymbol{x}_k\right)^{m-1} \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{42}$$

Using the Lagrange Multipliers

Sum over all i's

$$\sum_{i=1}^{C} A_i(\boldsymbol{x}_k) = \frac{\lambda_k^{\frac{1}{m-1}}}{m^{\frac{1}{m-1}} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^{\frac{2}{m-1}}} \tag{40}$$

Thus

$$\lambda_k = \frac{m}{\left[\sum_{i=1}^{C} \frac{1}{\left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^{\frac{2}{m-1}}} \right]^{m-1}} \tag{41}$$

Plug Back on equation 38 using j instead of i

$$\frac{m}{\left[\sum_{j=1}^{C} \frac{1}{\left\| \boldsymbol{x}_k - \boldsymbol{v}_j \right\|^{\frac{2}{m-1}}} \right]^{m-1}} = m A_i(\boldsymbol{x}_k)^{m-1} \left\| \boldsymbol{x}_k - \boldsymbol{v}_i \right\|^2 \tag{42}$$

Finally

We have that

$$A_i\left(\boldsymbol{x}_k\right) = \frac{1}{\left[\sum_{j=1}^{C}\left\{\frac{\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}{\left\|\boldsymbol{x}_k - \boldsymbol{v}_j\right\|^2}\right\}^{\frac{1}{m-1}}\right]} \tag{43}$$

In a similar way we have

$$v_i = \frac{\sum_{k=1}^{N} A_i\left(x_k\right)^m x_k}{\sum_{k=1}^{N} A_i\left(x_k\right)^m} \tag{44}$$

Finally

We have that

$$A_i\left(\boldsymbol{x}_k\right)=\frac{1}{\left[\sum_{j=1}^{C}\left\{\frac{\left\|\boldsymbol{x}_k-\boldsymbol{v}_i\right\|^2}{\left\|\boldsymbol{x}_k-\boldsymbol{v}_j\right\|^2}\right\}^{\frac{1}{m-1}}\right]} \tag{43}$$

In a similar way we have

$$v_i=\frac{\sum_{k=1}^{N}A_i\left(\boldsymbol{x}_k\right)^m\boldsymbol{x}_k}{\sum_{k=1}^{N}A_i\left(\boldsymbol{x}_k\right)^m} \tag{44}$$

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(x_k)^m x_k}{\sum_{k=1}^{N} A_i^{(t)}(x_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\| x_k - v_i^{(t)} \right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $$A_i^{(t+1)}(x_k) = \frac{1}{\left[\sum_{j=1}^{C} \left\{ \frac{\left\| x_k - v_i^{(t)} \right\|^2}{\left\| x_k - v_j^{(t)} \right\|^2} \right\}^{\frac{1}{m-1}} \right]}$$
   - Case II: $\left\| x_k - v_i^{(t)} \right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(x_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(x_k) = 1$ and $A_i^{(t+1)}(x_k) = 0$ for $i \notin I$.
4. If $\left| S^{(t+1)} - S^{(t)} \right| = \max_{i,k} \left| A_i^{(t+1)}(x_k) - A_i^{(t)}(x_k) \right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m \boldsymbol{x}_k}{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\| x_k - v_i^{(t)} \right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $A_i^{(t+1)}(x_k) = \frac{1}{\left[\sum_{j=1}^{C} \left\{ \frac{\left\| x_k - v_i^{(t)} \right\|^2}{\left\| x_k - v_j^{(t)} \right\|^2} \right\}^{\frac{1}{m-1}} \right]}$
   - Case II: $\left\| x_k - v_i^{(t)} \right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(x_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(x_k) = 1$ and $A_i^{(t+1)}(x_k) = 0$ for $i \notin I$.
4. If $\left| S^{(t+1)} - S^{(t)} \right| = \max_{i,k} \left| A_i^{(t+1)}(x_k) - A_i^{(t)}(x_k) \right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m \boldsymbol{x}_k}{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\| x_k - v_i^{(t)} \right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $$A_i^{(t+1)}(x_k) = \frac{1}{\left[\sum_{j=1}^{C} \left\{ \frac{\left\| x_k - v_i^{(t)} \right\|^2}{\left\| x_k - v_j^{(t)} \right\|^2} \right\}^{\frac{1}{m-1}}\right]}$$
   - Case II: $\left\| x_k - v_i^{(t)} \right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(x_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(x_k) = 1$ and $A_i^{(t+1)}(x_k) = 0$ for $i \notin I$.
4. If $\left| S^{(t+1)} - S^{(t)} \right| = \max_{i,k} \left| A_i^{(t+1)}(x_k) - A_i^{(t)}(x_k) \right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m \boldsymbol{x}_k}{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\|\boldsymbol{x}_k - \boldsymbol{v}_i^{(t)}\right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $$A_i^{(t+1)}(\boldsymbol{x}_k) = \frac{1}{\left[\sum_{j=1}^{C}\left\{\frac{\left\|\boldsymbol{x}_k - \boldsymbol{v}_i^{(t)}\right\|^2}{\left\|\boldsymbol{x}_k - \boldsymbol{v}_j^{(t)}\right\|^2}\right\}^{\frac{1}{m-1}}\right]}$$
   - Case II: $\left\|\boldsymbol{x}_k - \boldsymbol{v}_i^{(t)}\right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(\boldsymbol{x}_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(\boldsymbol{x}_k) = 1$ and $A_i^{(t+1)}(\boldsymbol{x}_k) = 0$ for $i \notin I$.
4. If $\left|S^{(t+1)} - S^{(t)}\right| = \max_{i,k}\left|A_i^{(t+1)}(\boldsymbol{x}_k) - A_i^{(t)}(\boldsymbol{x}_k)\right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m \boldsymbol{x}_k}{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $$A_i^{(t+1)}(\boldsymbol{x}_k) = \frac{1}{\left[\sum_{j=1}^{C} \left\{ \frac{\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2}{\left\| \boldsymbol{x}_k - \boldsymbol{v}_j^{(t)} \right\|^2} \right\}^{\frac{1}{m-1}} \right]}$$
   - Case II: $\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(\boldsymbol{x}_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(\boldsymbol{x}_k) = 1$ and $A_i^{(t+1)}(\boldsymbol{x}_k) = 0$ for $i \notin I$.
4. If $\left| S^{(t+1)} - S^{(t)} \right| = \max_{i,k} \left| A_i^{(t+1)}(x_k) - A_i^{(t)}(x_k) \right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Algorithm

Fuzzy c-means

1. Let $t = 0$. Select an initial fuzzy pseudo-partition.
2. Calculate the initial C cluster centers using, $v_i^{(t)} = \frac{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m \boldsymbol{x}_k}{\sum_{k=1}^{N} A_i^{(t)}(\boldsymbol{x}_k)^m}$.
3. Update for each x_k the membership function by
   - Case I: $\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2 > 0$ for all $i \in \{1, 2, ..., C\}$ then
     $$A_i^{(t+1)}(\boldsymbol{x}_k) = \frac{1}{\left[\sum_{j=1}^{C} \left\{ \frac{\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2}{\left\| \boldsymbol{x}_k - \boldsymbol{v}_j^{(t)} \right\|^2} \right\}^{\frac{1}{m-1}} \right]}$$
   - Case II: $\left\| \boldsymbol{x}_k - \boldsymbol{v}_i^{(t)} \right\|^2 = 0$ for some $i \in I \subseteq \{1, 2, ..., C\}$ then define $A_i^{(t+1)}(\boldsymbol{x}_k)$ by any nonnegative number such that $\sum_{i \in I} A_i^{(t+1)}(\boldsymbol{x}_k) = 1$ and $A_i^{(t+1)}(\boldsymbol{x}_k) = 0$ for $i \notin I$.
4. If $\left| \mathcal{S}^{(t+1)} - \mathcal{S}^{(t)} \right| = \max_{i,k} \left| A_i^{(t+1)}(\boldsymbol{x}_k) - A_i^{(t)}(\boldsymbol{x}_k) \right| \leq \epsilon$ stop; otherwise increase t and go to step 2.

Final Output

The Matrix U

The elements of U are $U_{ik} = A_i\left(\boldsymbol{x}_k\right)$.

The centroids

$V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_C\}$

Final Output

The Matrix U

The elements of U are $U_{ik} = A_i\left(\boldsymbol{x}_k\right)$.

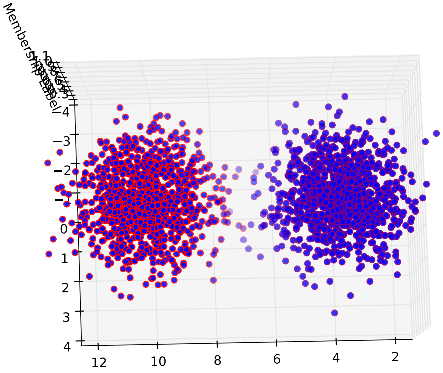
The centroids

$V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_C\}$

Outline

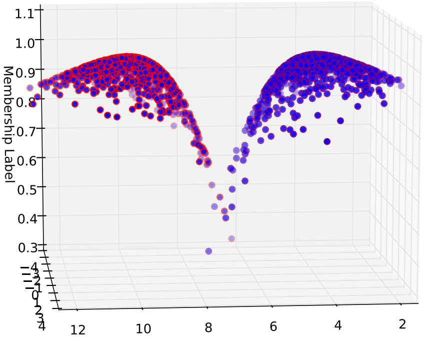
Example

Here the clustering of two Gaussian Clusters with $\mu_1 = (4,0)^T$, $\mu_2 = (10,0)^T$ and variance 1.0



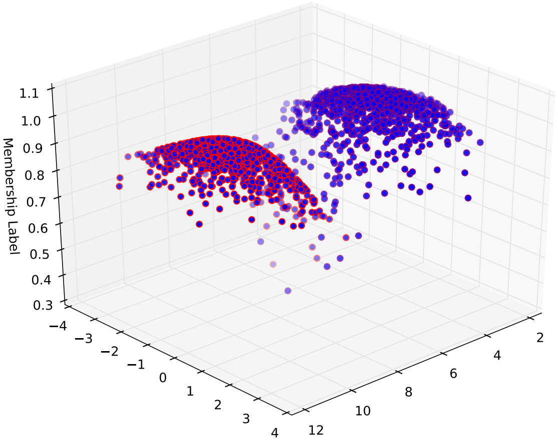
Example

Here the clustering of two Gaussian Clusters



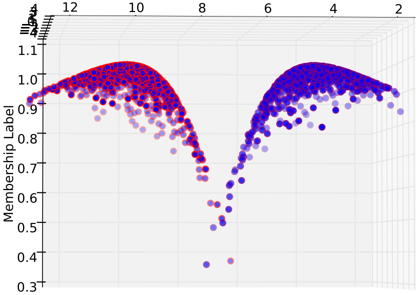
Example

Here the clustering of two Gaussian Clusters



Example

Here the clustering of two Gaussian Clusters



Outline

Pros and Cons of Fuzzy C-Means

Advantages

- Unsupervised
- Always converges

Pros and Cons of Fuzzy C-Means

Advantages

- Unsupervised
- Always converges

Disadvantages

- Long computational time
- Sensitivity to the initial guess (speed, local minima)
- Sensitivity to noise
  - One expects low (or even no) membership degree for outliers (noisy points)

Pros and Cons of Fuzzy C-Means

Advantages

- Unsupervised
- Always converges

Disadvantages

- Long computational time
- Sensitivity to the initial guess (speed, local minima)
- Sensitivity to noise
  - One expects low (or even no) membership degree for outliers (noisy points)

Pros and Cons of Fuzzy C-Means

Advantages

- Unsupervised
- Always converges

Disadvantages

- Long computational time
- Sensitivity to the initial guess (speed, local minima)
- Sensitivity to noise
  - One expects low (or even no) membership degree for outliers (noisy points)

Pros and Cons of Fuzzy C-Means

Advantages

- Unsupervised
- Always converges

Disadvantages

- Long computational time
- Sensitivity to the initial guess (speed, local minima)
- Sensitivity to noise
    - One expects low (or even no) membership degree for outliers (noisy points)

Pros and Cons of Fuzzy C-Means

Advantages

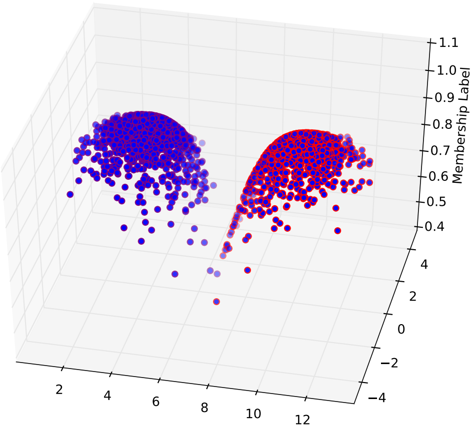
- Unsupervised
- Always converges

Disadvantages

- Long computational time
- Sensitivity to the initial guess (speed, local minima)
- Sensitivity to noise
  - One expects low (or even no) membership degree for outliers (noisy points)

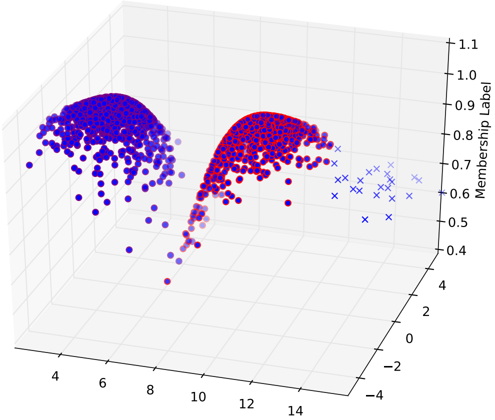
Outliers, Disadvantage of FCM

After running without outliers



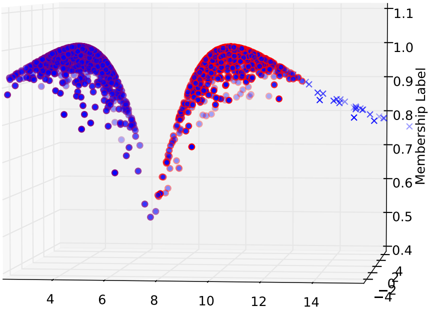
Outliers, Disadvantage of FCM

Now add outliers (Shown in blue $x's$) and their high memberships



Outliers, Disadvantage of FCM

Now add outliers (Shown in blue $x's$) and their high memberships



Outline

Krinshapuram and Keller

Following Zadeh

They took in consideration that each class prototype as defining an elastic constraint.

What?

Giving the $t_i(x_k)$ as degree of compatibility of sample x_k with cluster C_i.

We do the following

If we consider the C_i as fuzzy sets over the set of samples $X = \{x_1, x_2, ..., x_N\}$

Krinshapuram and Keller

Following Zadeh

They took in consideration that each class prototype as defining an elastic constraint.

What?

Giving the $t_i\left(\boldsymbol{x}_k\right)$ as degree of compatibility of sample $\boldsymbol{x}_k$ with cluster C_i.

We do the following

If we consider the C_i as fuzzy sets over the set of samples $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Krinshapuram and Keller

Following Zadeh

They took in consideration that each class prototype as defining an elastic constraint.

What?

Giving the $t_i(\boldsymbol{x}_k)$ as degree of compatibility of sample $\boldsymbol{x}_k$ with cluster C_i.

We do the following

If we consider the C_i as fuzzy sets over the set of samples $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Here is the Catch!!!

We should not use the old membership

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \tag{45}$$

Because

This is quite probabilistic... which is not what we want!!!

Thus

We only ask for membership, now using the possibilistic notation of $t_i\left(\boldsymbol{x}_k\right)$ (This is known as **typicality** value), to be in the interval $[0, 1]$.

Here is the Catch!!!

We should not use the old membership

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \tag{45}$$

Because

This is quite probabilistic... which is not what we want!!!

Thus

We only ask for membership, now using the possibilistic notation of $t_i\left(x_k\right)$ (This is known as **typicality** value), to be in the interval $[0, 1]$.

Here is the Catch!!!

We should not use the old membership

$$\sum_{i=1}^{C} A_i\left(\boldsymbol{x}_k\right) = 1 \tag{45}$$

Because

This is quite probabilistic... which is not what we want!!!

Thus

We only ask for membership, now using the possibilistic notation of $t_i\left(\boldsymbol{x}_k\right)$ (This is known as **typicality** value), to be in the interval $[0, 1]$.

New Constraints

First

$$t_i\left(\boldsymbol{x}_k\right) \in [0,1] \;\; \forall i,k \tag{46}$$

Second

$$0 < \sum_{k=1}^{N} t_i\left(\boldsymbol{x}_k\right) < N \;\; \forall i \tag{47}$$

Third

$$\max_i t_i\left(\boldsymbol{x}_k\right) > 0 \;\; \forall k \tag{48}$$

New Constraints

First

$$t_i\left(\boldsymbol{x}_k\right) \in [0,1] \;\; \forall i, k \tag{46}$$

Second

$$0 < \sum_{k=1}^{N} t_i\left(\boldsymbol{x}_k\right) < N \;\forall i \tag{47}$$

Third

$$\max_i t_i\left(\boldsymbol{x}_k\right) > 0 \;\forall k \tag{48}$$

New Constraints

First

$$t_i\left(\boldsymbol{x}_k\right) \in [0,1] \;\; \forall i, k \tag{46}$$

Second

$$0 < \sum_{k=1}^{N} t_i\left(\boldsymbol{x}_k\right) < N \;\forall i \tag{47}$$

Third

$$\max_{i} t_i\left(\boldsymbol{x}_k\right) > 0 \;\forall k \tag{48}$$

Outline

We have the following cost function

Cost Function

$$\sum_{k=1}^{N} \sum_{i=1}^{C} \left[t_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{49}$$

Problem

Unconstrained optimization of first term will lead to the trivial solution $t_i\left(\boldsymbol{x}_k\right) = 0$ for all i, k.

Thus, we can introduce the following constraint

$$t_i\left(\boldsymbol{x}_k\right) \to 1 \tag{50}$$

- Roughly it means to make the typicality values as large as possible.

We have the following cost function

Cost Function

$$\sum_{k=1}^{N}\sum_{i=1}^{C}\left[t_i\left(\boldsymbol{x}_k\right)\right]^m\left\Vert\boldsymbol{x}_k-\boldsymbol{v}_i\right\Vert^2 \tag{49}$$

Problem

Unconstrained optimization of first term will lead to the trivial solution $t_i\left(\boldsymbol{x}_k\right)=0$ for all i,k.

Thus, we can introduce the following constraint

$$t_i\left(\boldsymbol{x}_k\right)\rightarrow 1 \tag{50}$$

- Roughly it means to make the typicality values as large as possible.

We have the following cost function

Cost Function

$$\sum_{k=1}^{N}\sum_{i=1}^{C}\left[t_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k-\boldsymbol{v}_i\right\|^2 \tag{49}$$

Problem

Unconstrained optimization of first term will lead to the trivial solution $t_i\left(\boldsymbol{x}_k\right)=0$ for all i,k.

Thus, we can introduce the following constraint

$$t_i\left(\boldsymbol{x}_k\right)\to 1 \tag{50}$$

- Roughly it means to make the typicality values as large as possible.

We can try to control this tendency

By putting all them together in

$$\sum_{k=1}^{N} \left(1 - t_i\left(\boldsymbol{x}_k\right)\right)^m \tag{51}$$

With m to control the tendency of $t_i\left(\boldsymbol{x}_k\right) \rightarrow 1$

We can also run this tendency over all the cluster using a suitable $w_i > 0$ per cluster

$$\sum_{i=1}^{C} w_i \sum_{k=1}^{N} \left(1 - t_i\left(\boldsymbol{x}_k\right)\right)^m \tag{52}$$

We can try to control this tendency

By putting all them together in

$$\sum_{k=1}^{N} \left(1 - t_i\left(\boldsymbol{x}_k\right)\right)^m \qquad (51)$$

With m to control the tendency of $t_i\left(\boldsymbol{x}_k\right) \rightarrow 1$

We can also run this tendency over all the cluster using a suitable $w_i > 0$ per cluster

$$\sum_{i=1}^{C} w_i \sum_{k=1}^{N} \left(1 - t_i\left(\boldsymbol{x}_k\right)\right)^m \qquad (52)$$

Possibilistic C-Mean Clustering (PCM)

The final Cost Function

$$J_m(\mathcal{S}) = \sum_{k=1}^{N} \sum_{i=1}^{C} [t_i(\boldsymbol{x}_k)]^m \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 + \sum_{i=1}^{C} w_i \sum_{k=1}^{N} (1 - t_i(\boldsymbol{x}_k))^m \quad (53)$$

Where

- $t_i(\boldsymbol{x}_k)$ are **typicality** values.
- w_i are cluster weights

Possibilistic C-Mean Clustering (PCM)

The final Cost Function

$$J_m(\mathcal{S}) = \sum_{k=1}^{N}\sum_{i=1}^{C}\left[t_i(\boldsymbol{x}_k)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 + \sum_{i=1}^{C} w_i \sum_{k=1}^{N}\left(1 - t_i(\boldsymbol{x}_k)\right)^m \quad (53)$$

Where

- $t_i(\boldsymbol{x}_k)$ are **typicality** values.
- w_i are cluster weights

Explanation

First Term

$$\sum_{k=1}^{N} \sum_{i=1}^{C} \left[t_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{54}$$

It demands that the distance from feature vector to prototypes be as small as possible!!!

Explanation

First Term

$$\sum_{k=1}^{N} \sum_{i=1}^{C} \left[t_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2 \tag{54}$$

It demands that the distance from feature vector to prototypes be as small as possible!!!

Second Term

$$\sum_{i=1}^{c} w_i \sum_{k=1}^{n} \left(1 - t_i\left(x_k\right)\right)^m \tag{55}$$

It forces the typicality values $t_i\left(x_k\right)$ to be as large as possible.

Explanation

First Term

$$\sum_{k=1}^{N}\sum_{i=1}^{C}\left[t_i\left(\boldsymbol{x}_k\right)\right]^m\left\|\boldsymbol{x}_k-\boldsymbol{v}_i\right\|^2 \tag{54}$$

It demands that the distance from feature vector to prototypes be as small as possible!!!

Second Term

$$\sum_{i=1}^{c} w_i \sum_{k=1}^{n}\left(1-t_i\left(x_k\right)\right)^m \tag{55}$$

It forces the typicality values $t_i\left(x_k\right)$ to be as large as possible.

Explanation

First Term

$$\sum_{k=1}^{N}\sum_{i=1}^{C}\left[t_i\left(\boldsymbol{x}_k\right)\right]^m\left\|\boldsymbol{x}_k-\boldsymbol{v}_i\right\|^2 \tag{54}$$

It demands that the distance from feature vector to prototypes be as small as possible!!!

Second Term

$$\sum_{i=1}^{c} w_i \sum_{k=1}^{n}\left(1-t_i\left(x_k\right)\right)^m \tag{55}$$

It forces the typicality values $t_i\left(x_k\right)$ to be as large as possible.

Final Updating Equations

Typicality Values

$$t_i\left(\boldsymbol{x}_k\right) = \frac{1}{1 + \left(\frac{\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}{w_i}\right)^{\frac{1}{m-1}}},\ \forall i, k \tag{56}$$

Cluster Centers

$$v_i = \frac{\sum_{k=1}^{N} t_i\left(\boldsymbol{x}_k\right)^m \boldsymbol{x}_k}{\sum_{k=1}^{n} t_i\left(\boldsymbol{x}_k\right)^m} \tag{57}$$

Final Updating Equations

Typicality Values

$$t_i\left(\boldsymbol{x}_k\right) = \frac{1}{1 + \left(\frac{\left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}{w_i}\right)^{\frac{1}{m-1}}}, \ \forall i, k \quad (56)$$

Cluster Centers

$$v_i = \frac{\sum_{k=1}^{N} t_i\left(\boldsymbol{x}_k\right)^m x_k}{\sum_{k=1}^{n} t_i\left(\boldsymbol{x}_k\right)^m} \quad (57)$$

Final Updating Equations

Weights

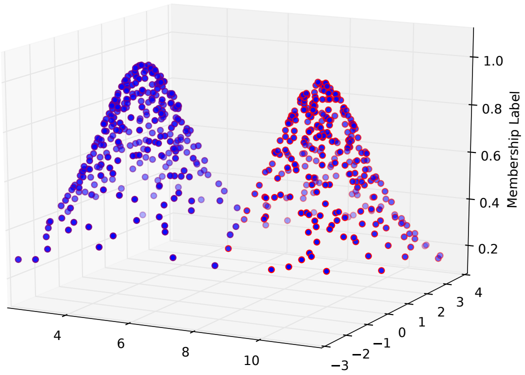
$$w_i = M \frac{\sum_{k=1}^{N} \left[t_i\left(\boldsymbol{x}_k\right)\right]^m \left\|\boldsymbol{x}_k - \boldsymbol{v}_i\right\|^2}{\sum_{k=1}^{n} \left[t_i\left(\boldsymbol{x}_k\right)\right]^m}, \tag{58}$$

with $M > 0$.

Possibilistic can deal with outliers

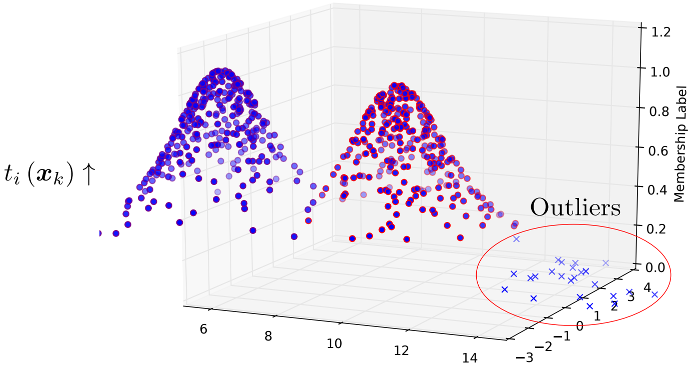
Two Gaussian Clusters with $\mu_1 = (4,0)^T$, $\mu_2 = (10,0)^T$ and $\sigma^2 = 1.0$

$t_i(\boldsymbol{x}_k) \uparrow$



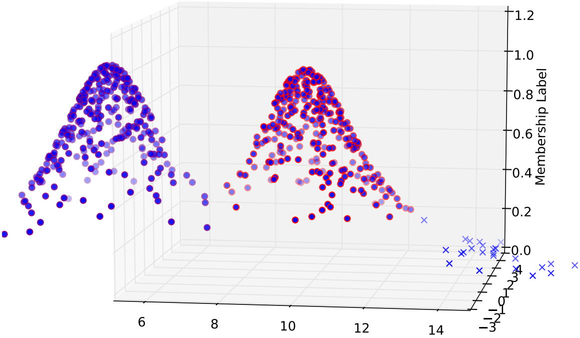
Possibilistic can deal with outliers

Now add outliers



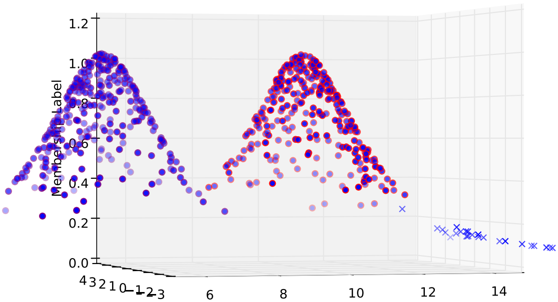
Possibilistic can deal with outliers

Another Angle



Possibilistic can deal with outliers

Another Angle



Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Disadvantages

- Very sensitive to good initialization.
- For example, I needed to run fuzzy C-means to obtain good initial typicalities

- Because the columns and rows of the typicality matrix are independent of each other.
- This could be advantageous (start with a large value of C and get less distinct clusters)

Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Disadvantages

- Very sensitive to good initialization.
- For example, I needed to run fuzzy C-means to obtain good initial typicalities

In Between!!!

Coincident clusters may result.

- Because the columns and rows of the typicality matrix are independent of each other.
- This could be advantageous (start with a large value of C and get less distinct clusters)

Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Disadvantages

- Very sensitive to good initialization.
- For example, I needed to run fuzzy C-means to obtain good initial typicalities

In Between!!!

Coincident clusters may result.

- Because the columns and rows of the typicality matrix are independent of each other.
- This could be advantageous (start with a large value of C and get less distinct clusters)

Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Disadvantages

- Very sensitive to good initialization.
- For example, I needed to run fuzzy C-means to obtain good initial typicalities

In Between!!!

Coincident clusters may result.

- Because the columns and rows of the typicality matrix are independent of each other.
- This could be advantageous (start with a large value of C and get less distinct clusters)

Pros and Cons of Fuzzy C-Means

Advantages

Clustering noisy data samples.

Disadvantages

- Very sensitive to good initialization.
- For example, I needed to run fuzzy C-means to obtain good initial typicalities

In Between!!!

Coincident clusters may result.

- Because the columns and rows of the typicality matrix are independent of each other.
- This could be advantageous (start with a large value of C and get less distinct clusters)

Nevertheless

There are more advanced clustering methods based on the possibilistic and fuzzy idea

Pal, N.R.; Pal, K.; Keller, J.M.; Bezdek, J.C., "A Possibilistic Fuzzy c-Means Clustering Algorithm," Fuzzy Systems, IEEE Transactions on , vol.13, no.4, pp.517,530, Aug. 2005.