DDA3020 Machine Learning: Lecture 14 K-means Clustering

Baoyuan Wu
School of Data Science, CUHK-SZ

April 15/17, 2024

Outline

Definition of K-means Clustering

- K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations/samples into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.
- K-means clustering minimizes within-cluster variances (squared Euclidean distances).

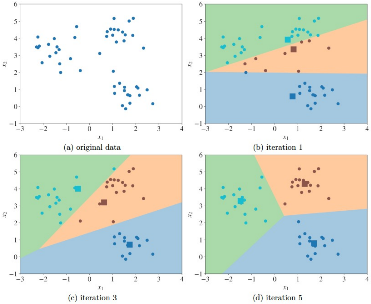
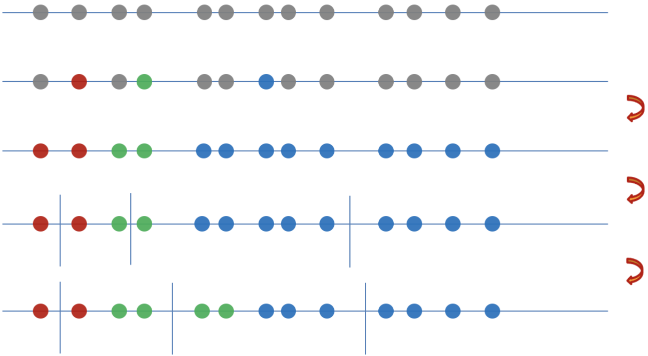


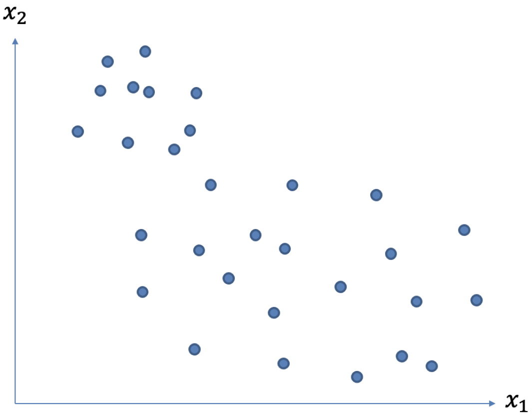
Figure 2: The progress of the k-means algorithm for $k = 3$.

References:
https://en.wikipedia.org/wiki/K-means_clustering
https://en.wikipedia.org/wiki/Vector_quantization

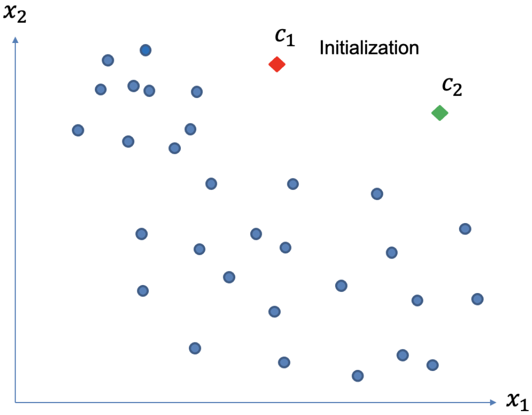
K-means Clustering (1 D)



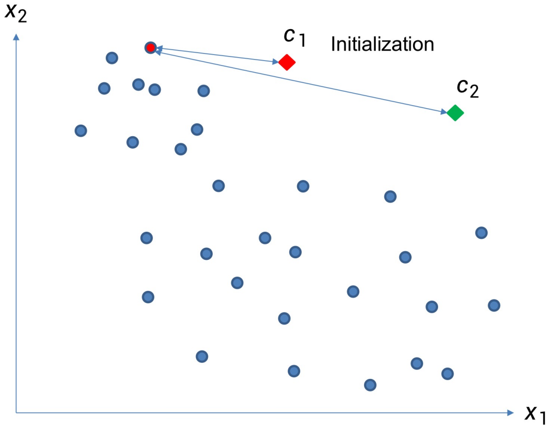
K-means Clustering (2 D)



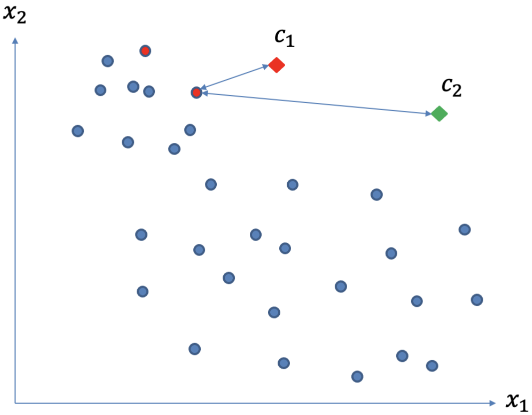
K-means Clustering



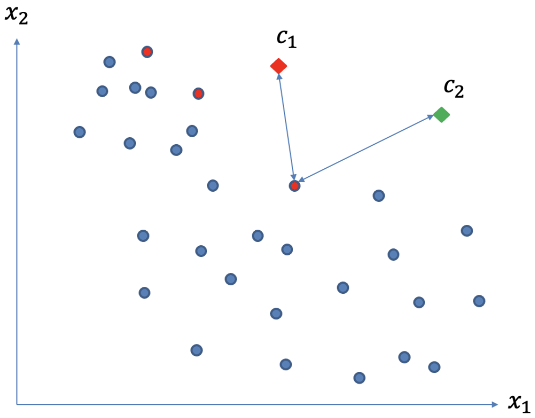
K-means Clustering



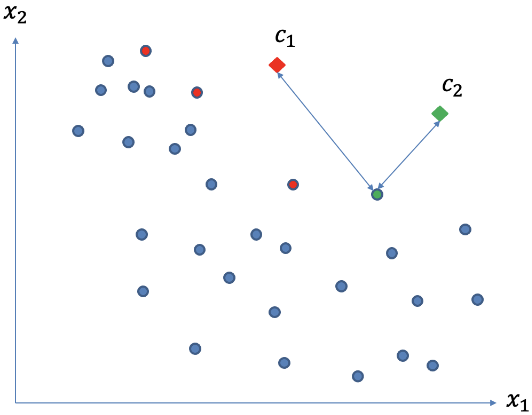
K-means Clustering



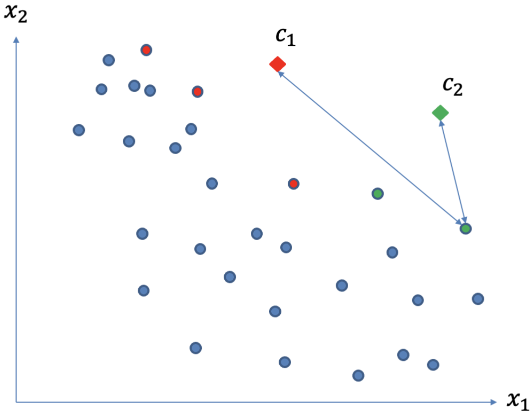
K-means Clustering



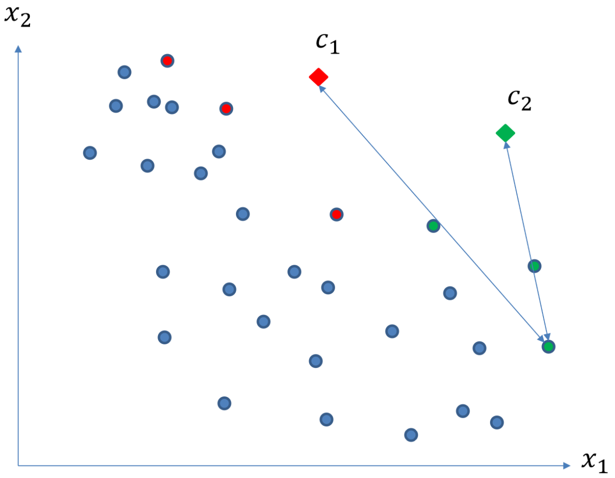
K-means Clustering



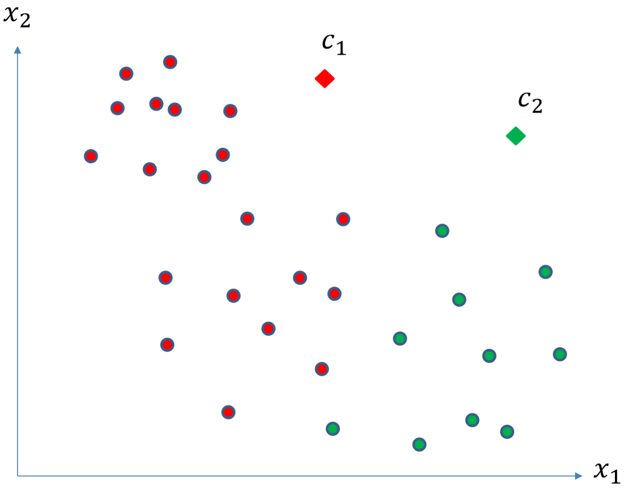
K-means Clustering



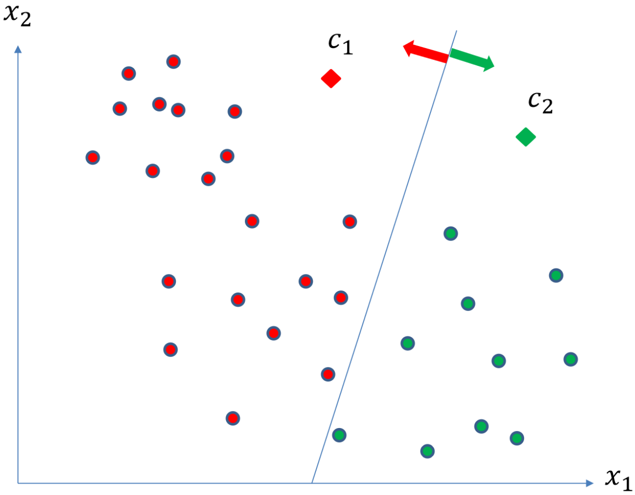
K-means Clustering



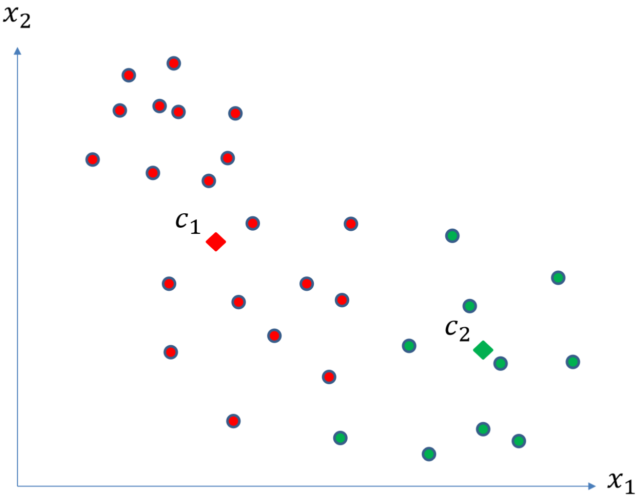
K-means Clustering



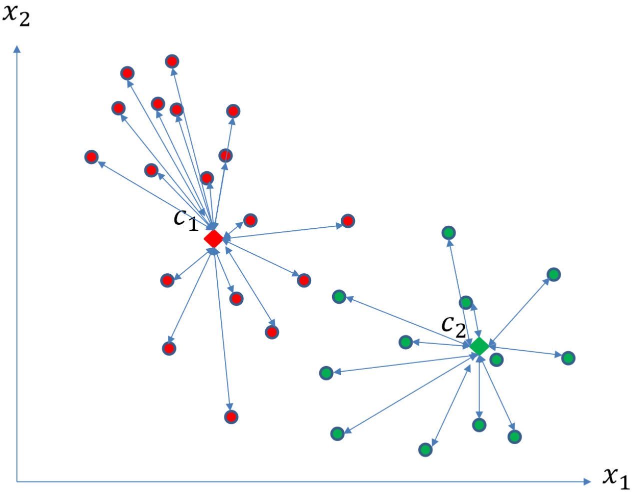
K-means Clustering



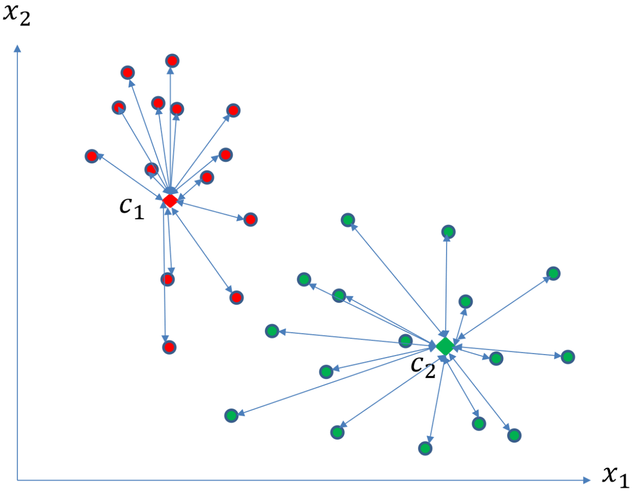
K-means Clustering



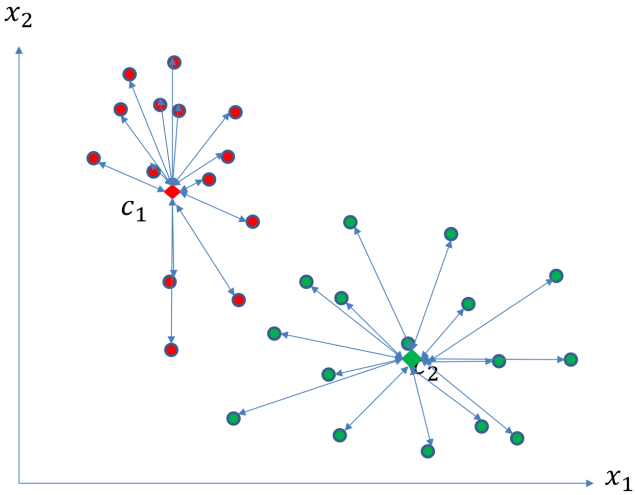
K-means Clustering



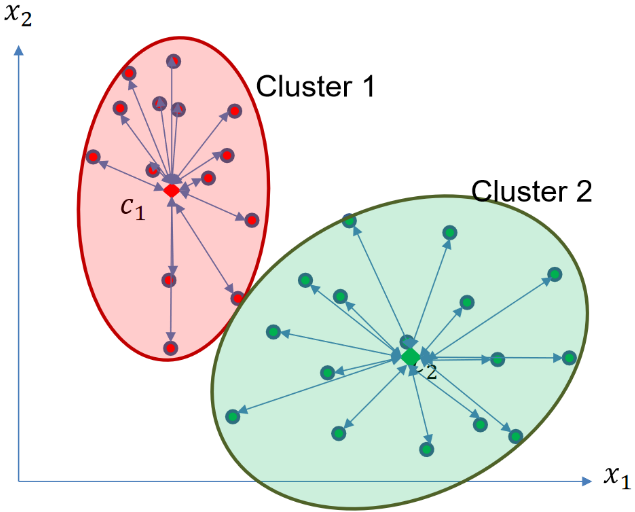
K-means Clustering



K-means Clustering



K-means Clustering



K-means Clustering

Let's see an online demo:
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

Basic K-means Clustering

1. First, you choose K — the number of clusters. Then you randomly put K feature vectors, called **centroids**, to the feature space.
2. Next, compute the distance from each example $\mathbf{x}$ to each centroid $\mathbf{c}$ using some metric, like the Euclidean distance. Then we assign the closest centroid to each example (like if we labeled each example with a centroid id as the label).
3. For each centroid, we calculate the average feature vector of the examples labeled with it. These average feature vectors become the new locations of the centroids.
4. We recompute the distance from each example to each centroid, modify the assignment and repeat the procedure until the assignments don't change after the centroid locations are recomputed.
5. Finally we conclude the clustering with a list of assignments of centroids IDs to the examples.

Optimization perspective of K-means Clustering

- What is actually being optimized by the basic K-means clustering algorithm?
- Given the data set $\{\mathbf{x}_i\}_{i=1}^n$, K-means aims to find cluster centers $\mathbf{c} = \{\mathbf{c}_j\}_{j=1}^K$ and assignments $\mathbf{r}$, by minimizing the sum of squared distances of data points to their assigned cluster centers. In short, K-means will minimize the within-cluster variance, as follows:

$$\min_{\mathbf{c},\mathbf{r}} J(\mathbf{c},\mathbf{r}) = \min_{\mathbf{c},\mathbf{r}} \sum_i^n \sum_k^K r_{ik}(\mathbf{x}_i - \mathbf{c}_k)^2,$$

$$\text{Subject to } \ \mathbf{r} \in \{0,1\}^{n \times K}, \ \sum_k^K r_{ik} = 1,$$

where $r_{ik} = 1$ denotes $\mathbf{x}_i$ is assigned to cluster k.

Optimization perspective of K-means Clustering

$$\min_{\mathbf{c},\mathbf{r}} J(\mathbf{c},\mathbf{r}) = \min_{\mathbf{c},\mathbf{r}} \sum_{i}^{n} \sum_{k}^{K} r_{ik}(\mathbf{x}_i - \mathbf{c}_k)^2,$$

$$\text{Subject to } \ \mathbf{r} \in \{0,1\}^{n \times K}, \ \sum_{k}^{K} r_{ik} = 1,$$

The above problem can be solved by coordinate descent algorithm, *i.e.*, update $\mathbf{c}$ and $\mathbf{r}$ alternatively:

- Given the cluster centers $\mathbf{c}$, update the assignments $\mathbf{r}$
- Given the assignments $\mathbf{r}$, update the cluster centers $\mathbf{c}$

Optimization perspective of K-means Clustering

Optimization of K-means clustering:

- Initialization: set K cluster centers $\mathbf{c}$ to random values
- Repeat until convergence (the assignments don't change):
  - Assignment: Given the cluster centers $\mathbf{c}$, update the assignments $\mathbf{r}$ by solving the following sub-problem

$$\min_{\mathbf{r}} \sum_{i}^{n} \sum_{k}^{K} r_{ik} (\mathbf{x}_i - \mathbf{c}_k)^2, \quad \text{subject to} \quad \mathbf{r} \in \{0,1\}^{n \times K}, \ \sum_{k}^{K} r_{ik} = 1.$$

    Note that the assignment for each data $\mathbf{x}_i$ can be solved independently. It is easy to know that assign $\mathbf{x}_i$ to the closest cluster is the optimal solution.
  - Refitting: Given the assignments $\mathbf{r}$, update the cluster centers $\mathbf{c}$:

$$\min_{\mathbf{c}} \sum_{i}^{n} \sum_{k}^{K} r_{ik} (\mathbf{x}_i - \mathbf{c}_k)^2.$$

    Note that $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$ can be optimized independently. By setting the derivative *w.r.t.* $\mathbf{c}_k$ as 0, it is easy to obtain the optimal solution:

$$\mathbf{c}_k = \frac{\sum_i^n r_{ik} \mathbf{x}_i}{\sum_i^n r_{ik}}.$$

Optimization perspective of K-means Clustering

Assignment:

- Given the cluster centers $\mathbf{c}$, update the assignments $\mathbf{r}$ by solving the following sub-problem

$$\min_{\mathbf{r}} \sum_{i}^{n} \sum_{k}^{K} r_{ik} (\mathbf{x}_i - \mathbf{c}_k)^2, \text{ subject to } \mathbf{r} \in \{0,1\}^{n \times K}, \sum_{k}^{K} r_{ik} = 1.$$

- Note that the assignment for each data $\mathbf{x}_i$ can be solved independently, *i.e.*,

$$\min_{\mathbf{r}_i} \sum_{k}^{K} r_{ik} (\mathbf{x}_i - \mathbf{c}_k)^2, \text{ subject to } \mathbf{r}_i \in \{0,1\}^{1 \times K}, \sum_{k}^{K} r_{ik} = 1.$$

- It is easy to obtain the solution as follows

$$k^* = \arg\min\{(\mathbf{x}_i - \mathbf{c}_k)^2\}_{k=1}^{K}, \text{ and } r_{ik^*} = 1.$$

- Thus, we assign $\mathbf{x}_i$ to the closest cluster, exactly same with the assignment step in basic K-means algorithm.

Optimization perspective of K-means Clustering

Refitting:

- Given the assignments $\mathbf{r}$, update the cluster centers $\mathbf{c}$:

$$\min_{\mathbf{c}} \sum_{i}^{n} \sum_{k}^{K} r_{ik}(\mathbf{x}_i - \mathbf{c}_k)^2.$$

- Note that $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$ can be optimized independently, as follows

$$\min_{\mathbf{c}_k} \sum_{i}^{n} r_{ik}(\mathbf{x}_i - \mathbf{c}_k)^2.$$

- By setting the derivative *w.r.t.* $\mathbf{c}_k$ as 0, *i.e.*,

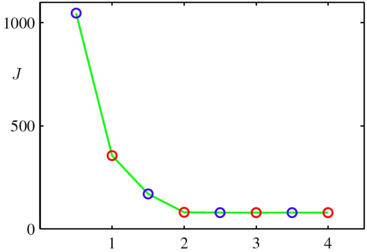
$$\sum_{i}^{n} 2r_{ik}(\mathbf{x}_i - \mathbf{c}_k) = 0 \;\Rightarrow\; \mathbf{c}_k = \frac{\sum_i^n r_{ik}\mathbf{x}_i}{\sum_i^n r_{ik}},$$

  where $\sum_i^n r_{ik}$ denotes the number of samples assigned to the kth cluster, and $\sum_i^n r_{ik}\mathbf{x}_i$ is the summation of all samples of the kth cluster.
- Thus, $\mathbf{c}_k$ is the center of the kth cluster, which is exactly same with the step of calculating the cluster center in basic K-means clustering.

Optimization perspective of K-means Clustering

Why K-means converges?

- Convergence guarantee:
  - Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, J is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).
- Example: As shown below, the objective function of K-means is reduced after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.

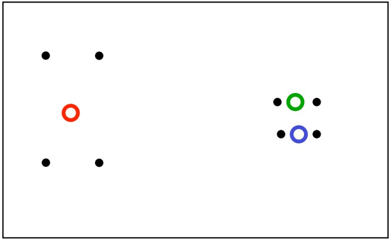


Optimization perspective of K-means Clustering

Local minimum of K-means

- Since the objective function J is non-convex, the coordinate descent on J is not guaranteed to converge to the global minimum
- There is nothing to prevent k-means getting stuck at local minimum, and sometimes it may stuck at poor local minimum (shown below)
- What we could do is running K-means with multiple random initializations, and picking the one with the lowest objective value as the final clustering result

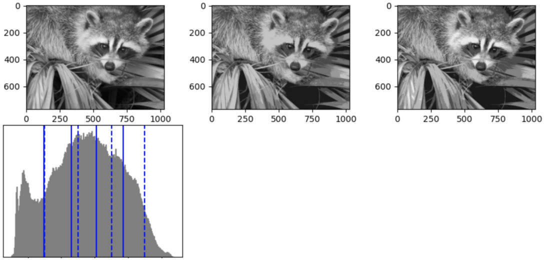
A bad local optimum



Example of K-means Clustering

Example: K-means for vector quantization

- Vector quantization is is a classical quantization technique from signal processing
- It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means
- As shown below, vector quantization is used for compressing image

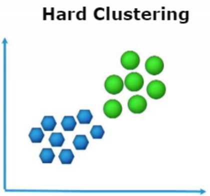


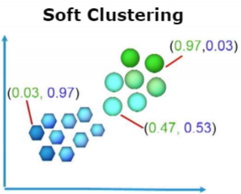
Demo with code:
https://scikit-learn.org/stable/auto_examples/cluster/plot_face_comp
html#sphx-glr-auto-examples-cluster-plot-face-compress-py

Soft Clustering

Hard vs Soft Clustering

- **Hard clustering**:
  - Each data point can belong to only one cluster
  - For example, an apple can be red OR green (hard clustering)
- **Soft clustering** (also known as **Fuzzy clustering**):
  - Each data point can belong to more than one cluster
  - For example, an apple can be red AND green (fuzzy clustering). Here, the apple can be red to a certain degree as well as green to a certain degree. Instead of the apple belonging to green [green $= 1$] and not red [red $= 0$], the apple can belong to green [green $= 0.5$] and red [red $= 0.5$].





Fuzzy C-means Clustering

- Soft K-means clustering is also called as fuzzy c-means clustering. Its objective function is formulated as follows:

$$\min_{\mathbf{c},\mathbf{r}} J(\mathbf{c}, \mathbf{r}) = \min_{\mathbf{c},\mathbf{r}} \sum_{i}^{n} \sum_{k}^{K} (r_{ik})^m (\mathbf{x}_i - \mathbf{c}_k)^2,$$

$$\text{Subject to } \ \mathbf{r} \in [0,1]^{n \times K}, \ \sum_{k}^{K} r_{ik} = 1,$$

  where r_{ik} is the degree to which a sample $\mathbf{x}_i$ belongs to a cluster $\mathbf{c}_k$.
- The hyper-parameter $m > 1$ is called *fuzzifier*, and it defines the level of cluster fuzziness. Note that, a value of m close to 1 gives a cluster solution which becomes increasingly similar to the solution of hard clustering such as k-means; whereas a value of m close to infinite leads to complete fuzziness (explain later).
- The above problem can also be solved by coordinate descent algorithm:
  - Given $\mathbf{r}$, update $\mathbf{c}$;
  - Given $\mathbf{c}$, update $\mathbf{r}$.

Sub-problem of $\mathbf{r}$

- Given $\mathbf{c}$, we update $\mathbf{r}$ by solving the following sub-problem:

$$\min_{\mathbf{r}} J(\mathbf{r}) = \min_{\mathbf{c},\mathbf{r}} \sum_{i}^{n} \sum_{k}^{K} (r_{ik})^m (\mathbf{x}_i - \mathbf{c}_k)^2,$$

$$\text{Subject to } \ \mathbf{r} \in [0,1]^{n \times K}, \ \sum_{k}^{K} r_{ik} = 1.$$

- Note that above constraints are equivalent to $\mathbf{r} \geq \mathbf{0}, \ \sum_k^K r_{ik} = 1$.
- The optimal solution of the above problem can be obtained according to the KKT conditions. Firstly, we write the Lagrangian function, as follows:

$$\mathcal{L}(\mathbf{r}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = J(\mathbf{r}) + \sum_{i}^{n} \alpha_i \big(1 - \sum_{k}^{K} r_{ik}\big) + \sum_{i}^{n} \sum_{k}^{K} \beta_{ik}(-r_{ik}).$$

Fuzzy C-means Clustering

Sub-problem of $\mathbf{r}$

- Lagrangian function

$$\mathcal{L}(\mathbf{r}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = J(\mathbf{r}) + \sum_{i}^{n} \alpha_i \big(1 - \sum_{k}^{K} r_{ik}\big) + \sum_{i}^{n} \sum_{k}^{K} \beta_{ik}(-r_{ik}).$$

- The KKT conditions are:

$$\text{Stationary:} \quad \frac{\partial \mathcal{L}}{\partial r_{ik}} = 0 \;\Rightarrow\; r_{ik} = \big(\frac{\alpha_i + \beta_{ik}}{m}\big)^{\frac{1}{m-1}} \cdot \big(\frac{1}{d_{ik}^2}\big)^{\frac{1}{m-1}} \tag{1}$$

$$\text{Primal feasibility:} \quad \sum_{k}^{K} r_{ik} = 1,\; r_{ik} \geq 0 \tag{2}$$

$$\text{Dual feasibility:} \quad \beta_{ik} \geq 0, \forall i, k \tag{3}$$

$$\text{Complementary:} \quad \beta_{ik} \cdot r_{ik} = 0, \forall i, k, \tag{4}$$

where $d_{ik}^2 = (\mathbf{x}_i - \mathbf{c}_k)^2$.

Fuzzy C-means Clustering

Sub-problem of $\mathbf{r}$

- From (1), we know that if $\alpha_i + \beta_{ik} \neq 0$, then $r_{ik} > 0$. Thus, according to (4), we have $\beta_{ik} = 0$, then

$$r_{ik} = \Big(\frac{\alpha_i}{m}\Big)^{\frac{1}{m-1}} \cdot \Big(\frac{1}{d_{ik}^2}\Big)^{\frac{1}{m-1}}. \tag{5}$$

- What will happen if $\alpha_i + \beta_{ik} = 0, \forall i, k$? If that case, $r_{ik} = 0, \forall k$, which violates the primal feasibility constraint $\sum_k^K r_{ik} = 1$. Thus, this case will not happen.
- Replace (5) into (2), we have

$$\Big(\frac{\alpha_i}{m}\Big)^{\frac{1}{m-1}} = \frac{1}{\sum_k^K \big(\frac{1}{d_{ik}^2}\big)^{\frac{1}{m-1}}}. \tag{6}$$

- Replace it back into (5), we obtain

$$r_{ik} = \frac{1}{\sum_j^K \big(\frac{1}{d_{ij}^2}\big)^{\frac{1}{m-1}}} \cdot \Big(\frac{1}{d_{ik}^2}\Big)^{\frac{1}{m-1}} = \frac{1}{\sum_j^K \big(\frac{d_{ik}^2}{d_{ij}^2}\big)^{\frac{1}{m-1}}}. \tag{7}$$

- What is the effect of $m > 1$ for r_{ik}? If $m \to 1$, then $\mathbf{r}_i$ is close to one-hot vector, *i.e.*, hard assignment; if $m \to \infty$, then $\mathbf{r}_i$ is close to uniform vector,

Fuzzy C-means Clustering

Sub-problem for $\mathbf{c}$

- Given $\mathbf{r}$, the centroid $\mathbf{c}$ is updated by optimizing the following sub-problem:

$$\min_{\mathbf{c}} J(\mathbf{c}) = \min_{\mathbf{c}} \sum_{i}^{n} \sum_{k}^{K} (r_{ik})^m (\mathbf{x}_i - \mathbf{c}_k)^2.$$

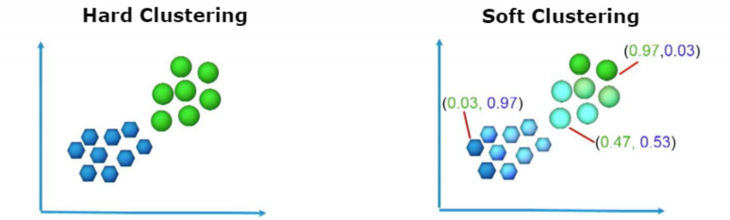
- By setting the derivative to 0, we obtain

$$\frac{\partial J(\mathbf{c})}{\partial \mathbf{c}_k} = 0 \ \Rightarrow \ \mathbf{c}_k = \frac{\sum_i^n [(r_{ik})^m \mathbf{x}_i]}{\sum_i^n (r_{ik})^m}.$$

Fuzzy C-means Clustering

Basic K-means vs. Fuzzy C-means:

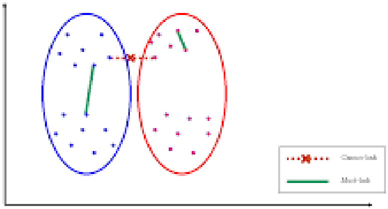
- Basic K-means: hard assignment, *i.e.*, $r_{ik} \in \{0, 1\}$
- Fuzzy C-means: soft assignment, $r_{ik} \in [0, 1]$



Further readings for fuzzy c-means clustering:
Convex optimization: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
Demo with code: https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_cmeans.html

Constrained K-means Clustering

- In practice, we may know some additional evidences or preferences about some data points we want to do clustering, such as:
  - Must-link constraints: two points must be partitioned to the same cluster (as shown in green line)
  - Cannot-link constraints: two points cannot be partitioned to the same cluster (as shown in red line)
- K-means with must-link/cannot-link constraints is called constrained K-means or semi-supervised K-means.
- How to modify the basic K-means clustering algorithm to satisfy such constraints?



Constrained K-means Clustering

- The only change is adding a violate-constraints check to the assignment stage, to ensure all constraints are satisfied.
- For each point, it is firstly assigned to the closest cluster, and check all constraints it involves: if all constraints are satisfied, then this assignment is accepted; if any constraint is violated, then assign it to the next closest cluster and repeat the violate-constraints check; if no legal cluster can be found, then the whole clustering fails.

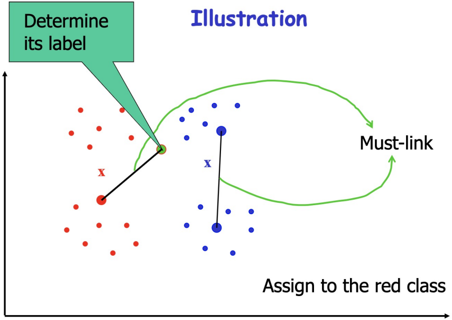
COP-KMEANS(data set D, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. Let $C_1 \dots C_k$ be the initial cluster centers.
2. For each point d_i in D, assign it to the closest cluster C_j **such that** VIOLATE-CONSTRAINTS**(**d_i**,** C_j**,** $Con_=$**,** $Con_{\neq}$**) is false. If no such cluster exists, fail (return {})**.
3. For each cluster C_i, update its center by averaging all of the points d_j that have been assigned to it.
4. Iterate between (2) and (3) until convergence.
5. Return $\{C_1 \dots C_k\}$.

VIOLATE-CONSTRAINTS(data point d, cluster C, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

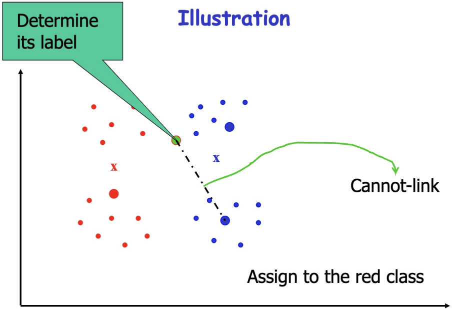
1. For each $(d, d_=) \in Con_=$: If $d_= \notin C$, return true.
2. For each $(d, d_{\neq}) \in Con_{\neq}$: If $d_{\neq} \in C$, return true.
3. Otherwise, return false.

Constrained K-means Clustering



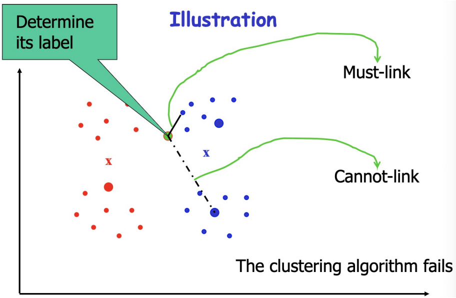
- While all other points have been assigned, we want to determine the cluster of the green point. It involves a must-link.
- Firstly, we compute its distances to two cluster centers, *i.e.*, $\mathbf{x}$ and $\mathbf{x}$.
- It is shown that it is closer to $\mathbf{x}$ (*i.e.*, the blue cluster). Thus, we assign it to the blue cluster firstly.
- Then, we check the constraint, and find the must-link constraint is violated. Thus, we assign it to the 2nd closest cluster (*i.e.*, the red cluster), and find that the constraint is satisfied. Thus, this assignment is accepted.

Constrained K-means Clustering



- While all other points have been assigned, we want to determine the cluster of the green point. It involves a cannot-link.
- Firstly, we compute its distances to two cluster centers, *i.e.*, $\mathbf{x}$ and $\mathbf{x}$.
- It is shown that it is more close to $\mathbf{x}$ (*i.e.*, the blue cluster). Thus, we assign it to the blue cluster firstly.
- Then, we check the constraint, and find the cannot-link constraint is violated. Thus, we assign it to the 2nd closest cluster (*i.e.*, the red cluster), and find that the constraint is satisfied. Thus, this assignment is accepted.

Constrained K-means Clustering



- While all other points have been assigned, we want to determine the cluster of the green point. It involves a must-link and a cannot-link.
- Firstly, we compute its distances to two cluster centers, *i.e.*, **x** and **x**.
- It is shown that it is more close to **x** (*i.e.*, the blue cluster). Thus, we assign it to the blue cluster firstly.
- Then, we check the constraint, and find the cannot-link constraint is violated. Thus, we assign it to the 2nd closest cluster (*i.e.*, the red cluster), but find that the must-link constraint is violated. Thus, you cannot find a legal assignment to the green point. The clustering fails.

Constrained K-means Clustering

- Constrained K-means Clustering with Background Knowledge (ICML 2001): `https://web.cse.msu.edu/~cse802/notes/ConstrainedKmeans.pdf`
- Code: `https://github.com/Behrouz-Babaki/COP-Kmeans`

Cost of Basic K-means Clustering

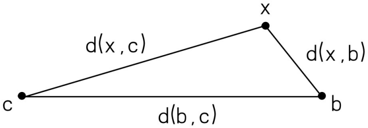
Computational cost of basic K-means algorithm:

- For the assignment stage, you have to compute the distance between every pair of data and cluster center, *i.e.*, $(\mathbf{x}_i, \mathbf{c}_k)$. Thus, the cost is $O(n \times K \times d)$, with n being the number of points, K being the number of clusters, and d being the feature dimension
- For the center calculation stage, you have to calculate the center of every cluster, then the cost is $O(\sum_{i=1}^{K} n_i \times d) = O(n \times d)$
- The total cost is $O(T \times (n \times (K+1) \times d))$, with T being the number of iterations.
- For large scale and high dimensional data, the cost is high.
- Is it possible to accelerate the algorithm, while the performance is not affected?

Accelerated K-means Clustering

Triangle inequality theorem:
Let x be a data point, and b and c be two centers. $d(x, b)$ denotes the distance between x and b. We have

- $d(x, b) + d(x, c) > d(b, c)$
- $d(x, b) + d(b, c) > d(x, c)$
- $d(b, c) + d(x, c) > d(x, b)$



Further, we can derive the following lemmas:

- Lemma 1: if $d(b, c) \geq 2d(x, b)$, then $d(x, c) > d(x, b)$
- Lemma 2: $d(x, c) > \max\{0, d(x, b) - d(b, c)\}$

Accelerated K-means Clustering

Lemma 1: if $d(b,c) \geq 2d(x,b)$, then $d(x,c) > d(x,b)$

Usage of Lemma 1 in K-means:

- Let x be any data point, and c be the center to which x is currently assigned, and let c' be any other center
- If $d(c,c') \geq 2d(x,c)$, then $d(x,c') \geq d(x,c)$.
- In this case, it is unnecessary to compute $d(x,c')$, leading to the cost reduction

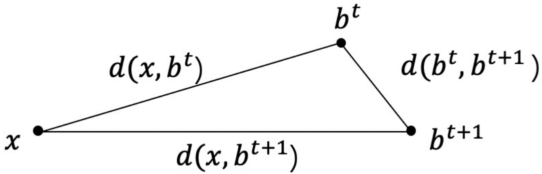
Accelerated K-means Clustering

Lemma 2: $d(x,c) > \max\{0, d(x,b) - d(b,c)\}$

Usage of Lemma 2 in K-means:

- Let x be any data point, b^{t+1} be any center at the $t+1$ iteration, and b^t be the previous version of the same center. For example, suppose the centers are numbered 1 through k, and b^{t+1} is the center number j, then b^t is the center number j in the previous iteration.
- Lower bound: Suppose that in the previous iteration t, we knew a lower bound $l(x,b^t)$ such that $d(x,b^t) \geq l(x,b^t)$, then we can update a new lower bound $l(x,b^{t+1})$ for the current iteration $t+1$

$$d(x,b^{t+1}) \geq \max\{0, d(x,b^t) - d(b^{t+1},b^t)\} \geq \max\{0, l(x,b^t) - d(b^{t+1},b^t)\} = l(x,b^{t+1})$$

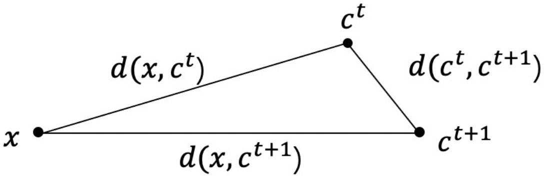


Accelerated K-means Clustering

Lemma 2: $d(x, c) > \max\{0, d(x, b) - d(b, c)\}$

Usage of Lemma 2 in K-means:

- Upper bound: Suppose $u(x) \geq d(x, c^t)$ is an upper bound on the distance between x and its currently assigned centroid c^t (the centroid at iteration t).
  - And, suppose $l(x, (c')^t) \leq d(x, (c')^t)$ is a lower bound on the distance between x and some other center $(c')^t$.
  - If $u(x) \leq l(x, (c')^t)$, then $d(x, c^t) \leq u(x) \leq l(x, (c')^t) \leq d(x, (c')^t)$.
  - Thus, it is necessary to calculate neither $d(x, c^t)$ nor $d(x, (c')^t)$, leading to cost reduction. Note that it will never be necessary to calculate $d(x, (c')^t)$ in the current iteration, but it may be necessary to calculate $d(x, c^t)$, as $u(x) \leq l(x, (c'')^t)$ may not true for some other center c''.
- Update the upper bound $u(x)$: $u(x) = u(x) + d(c^t, c^{t+1}) > d(x, c^{t+1})$



The complete algorithm of accelerated K-means clustering algorithm:

Putting the observations above together, the accelerated k-means algorithm is as follows.

First, pick initial centers. Set the lower bound $l(x, c) = 0$ for each point x and center c. Assign each x to its closest initial center $c(x) = \text{argmin}_c\, d(x, c)$, using Lemma 1 to avoid redundant distance calculations. Each time $d(x, c)$ is computed, set $l(x, c) = d(x, c)$. Assign upper bounds $u(x) = \min_c d(x, c)$.

Next, repeat until convergence:

1. For all centers c and c', compute $d(c, c')$. For all centers c, compute $s(c) = \frac{1}{2} \min_{c' \neq c} d(c, c')$.
2. Identify all points x such that $u(x) \leq s(c(x))$.
3. For all remaining points x and centers c such that

   (i) $c \neq c(x)$ and

   (ii) $u(x) > l(x, c)$ and

   (iii) $u(x) > \frac{1}{2} d(c(x), c)$:

3a. If $r(x)$ then compute $d(x, c(x))$ and assign $r(x) =$ false. Otherwise, $d(x, c(x)) = u(x)$.

3b. If $d(x, c(x)) > l(x, c)$ or $d(x, c(x)) > \frac{1}{2} d(c(x), c)$ then
   Compute $d(x, c)$
   If $d(x, c) < d(x, c(x))$ then assign $c(x) = c$.

4. For each center c, let $m(c)$ be the mean of the points assigned to c.
5. For each point x and center c, assign
$$l(x, c) = \max\{l(x, c) - d(c, m(c)), 0\}.$$
6. For each point x, assign
$$u(x) = u(x) + d(m(c(x)), c(x))$$
$$r(x) = \text{true}.$$
7. Replace each center c by $m(c)$.

Accelerated K-means Clustering

Further readings:

- Using the Triangle Inequality to Accelerate K-means (ICML 2003):
  https://www.aaai.org/Papers/ICML/2003/ICML03-022.pdf
- Code:
  https://github.com/siddheshk/Faster-Kmeans

Performance evaluation of clustering

There are two types of evaluation metrics for clustering:

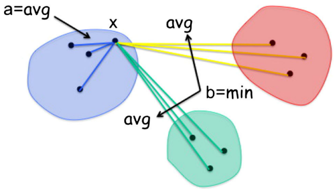
- Internal evaluation metrics: Silhouette coefficient
- External evaluation metrics: These metrics require the knowledge of the ground truth classes while almost never available in practice or requires manual assignment by human annotators (as in the supervised learning setting).

Silhouette coefficient

- Given a clustering, we define
  - a: The mean distance between a point and all other points in the **same** cluster.
  - b: The smallest mean distance of a point to all points in any **other** cluster.
- Silhouette coefficient s for a single sample is formulated as:

$$s = \frac{b-a}{\max(a,b)} \;\Rightarrow\; s = \begin{cases} 1-\frac{a}{b} & \text{if } a<b \\ 0 & \text{if } a=b \\ \frac{b}{a}-1 & \text{if } a>b \end{cases}$$

- It is easy to know that $s \in (-1,1)$, and larger s value indicates better clustering performance.
- Silhouette coefficient s for a set of samples is defined as the mean of the Silhouette Coefficient for each sample.



Rand index

- Given a set of n samples $S = \{o_1, o_2, \ldots, o_n\}$, there are two clusterings/partitions of S to compare, including:
  - $X = \{X_1, X_2, \ldots, X_r\}$ with r clusters
  - $Y = \{Y_1, Y_2, \ldots, Y_s\}$ with s clusters
- We can calculate the following values:
  - a: The number of pairs of elements in S that are in the **same** subset in X and in the **same** subset in Y
  - b: The number of pairs of elements in S that are in the **different** subset in X and in the **different** subset in Y
  - c: The number of pairs of elements in S that are in the **same** subset in X and in the **different** subset in Y
  - d: The number of pairs of elements in S that are in the **different** subset in X and in the **same** subset in Y
- The rand index (RI) can be computed as follows:

$$\text{RI} = \frac{a+b}{a+b+c+d} = \frac{a+b}{\frac{n(n-1)}{2}}$$

Note that $\text{RI} \in [0, 1]$, and higher score corresponds higher similarity.

Performance evaluation of Clustering

More evaluation metrics for clustering, as well as the demos with code, can be found in the following links:

- Wiki: https://en.wikipedia.org/wiki/Cluster_analysis#Internal_evaluation
- Demo with code: https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation

Other clusterings

Clustering is always an active area in machine learning. Despite of the introduced K-means algorithm, there are lots of other clustering algorithms, such as

- Hierarchical clustering
- Graph based clustering
- Density based clustering
- Probabilistic clustering

Further reading "Survey of Clustering Algorithms":

- https://axon.cs.byu.edu/Dan/678/papers/Cluster/Xu.pdf