Section 6: K Means Clustering

Section 6.1: K Means Clustering:Theory

K Means Clustering: What is it?

- K Means Clustering is a centroid based approach
- User specifies the number of clusters K
- User specifies initial guess for the mean (centre) of each cluster
- K Means Clustering employs an iterative approach to determine the mean of each cluster and points associated with each cluster
- K means performs a “hard” clustering - each data point is assigned to exactly one cluster
- See Resources file for links to additional information

K Means Clustering: Basic Step

(A) Start with dataset and current estimate for cluster means

(B) For each point in dataset, determine closest cluster centre and assign point to that cluster

(C) Recompute cluster means based on assignment of points in (A)

- Note change in cluster means from frame B

(D) Go back to (A) and repeat process until means converge

A B C A

K Means Algorithm

- Assume M data points $\{X_i\}$
- Specify tolerance ε and number of clusters K

(1) Randomly choose K data points to be the initial cluster means $\{C_k\}$

(2) While change in cluster means is greater than ε
  - Assign each data point X_i to closest cluster mean C_k
  - Re-compute cluster means $\{C_k\}$ based on latest assignment of points
  - Compute change in cluster means

- Typically, also specify a maximum number of iterations in while loop

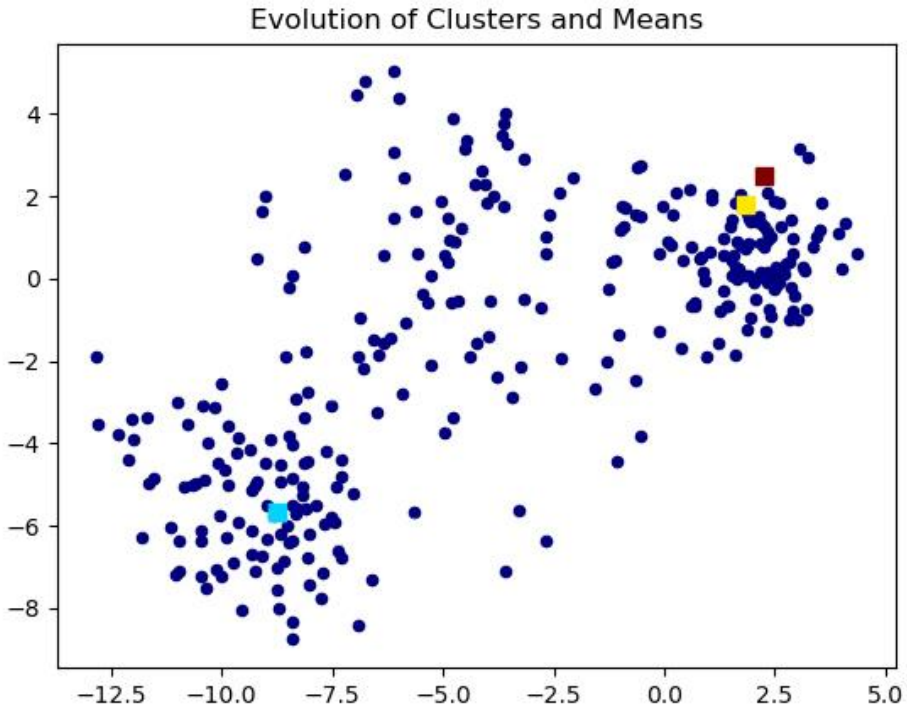
K Means Algorithm: Minimizing Objective Function

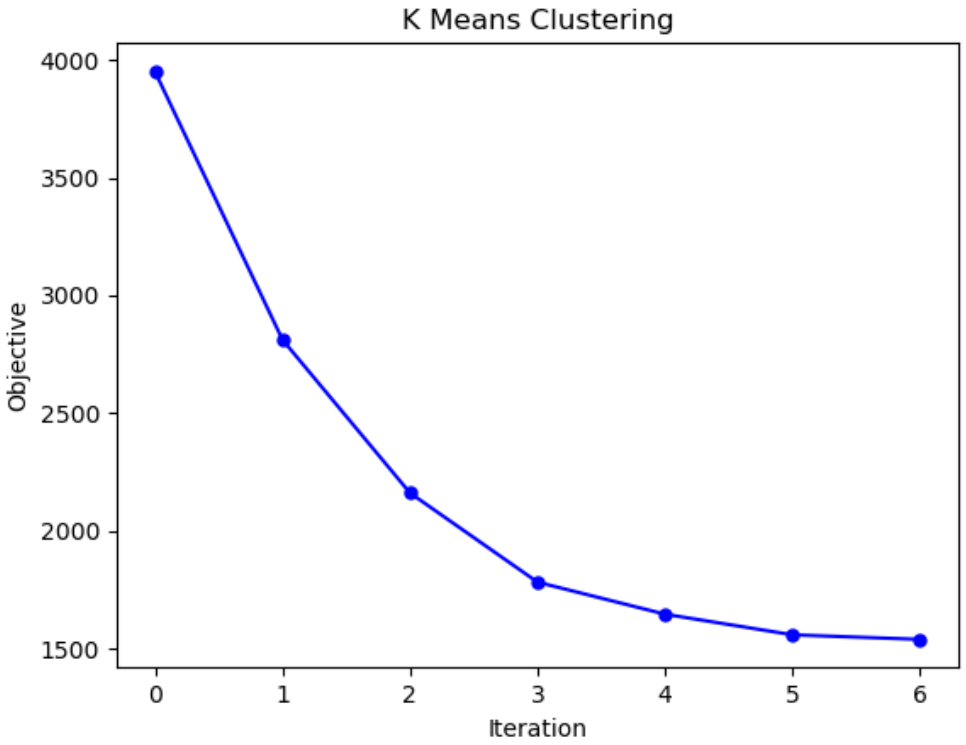
- Let {X} denote the data points (drop subscript for simplicity)
- Let S_k denote the set of points in cluster k=0,…,K-1 and let C_k denote the cluster mean
- K Means tries to partition the points into K cluster so as to minimize the within cluster sum of squares of distance:

$$Objective = \sum_{k=0}^{K-1} \sum_{X \in S_k} dist(X, C_k)^2$$

K Means Clustering: Example

- Dataset: sklearn varied_blobs1 data set with 300 points
- Specify 3 clusters and pick initial means at random from data points
- Set stopping tolerance $\varepsilon = 10^{-5}$



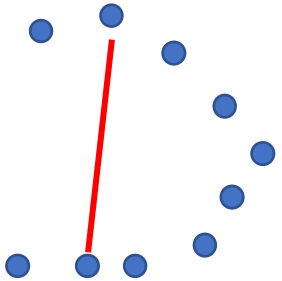


K Means Algorithm: Complexity

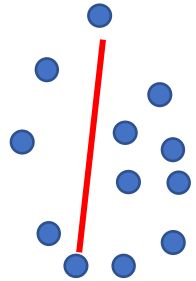
- Assume: M data points in d dimensions, K clusters, I iterations
- At each iteration, need to compute distance between each data point and each cluster centre
- If we assume fixed number of iterations, then algorithm requires $O(M)$ operations as $M \to \infty$
- Number of operations is proportional to number of dimensions d, number clusters K, and number of iterations I
- Amount of memory required is $O(M)$ as $M \to \infty$

K Means Clustering: Notes

- User must specify number of clusters
- User specifies distance measure
- No guarantee that the objective function is minimized – typically will find local minimum
- Final cluster means and assignments depend on initial guesses
- K Means does not do well for non-convex clusters



Non-convex: line between points “not within cluster”



Convex: line between points “within cluster”

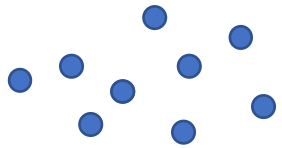
K Means ++ for Picking Initial Cluster Means

(1) Pick point from dataset at random as first cluster mean

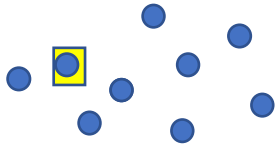
(2) Loop for Cluster Means = 1, …, K-1

- For each point x not yet chosen, compute distance to nearest existing cluster mean
- Among all distances, pick point at random using probability distribution proportional to $[D(x)]^2$

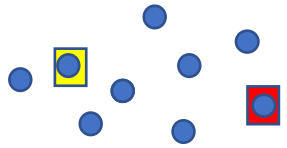
After initial cluster mean, this approach tends to pick cluster means at the edges of the data set – in code pick furthest point from other means



Dataset



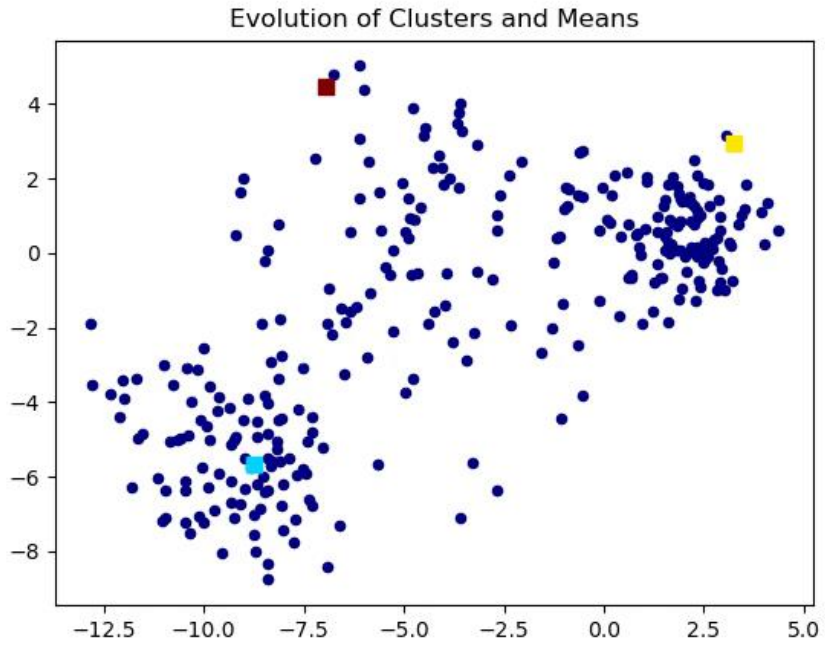
1st Mean

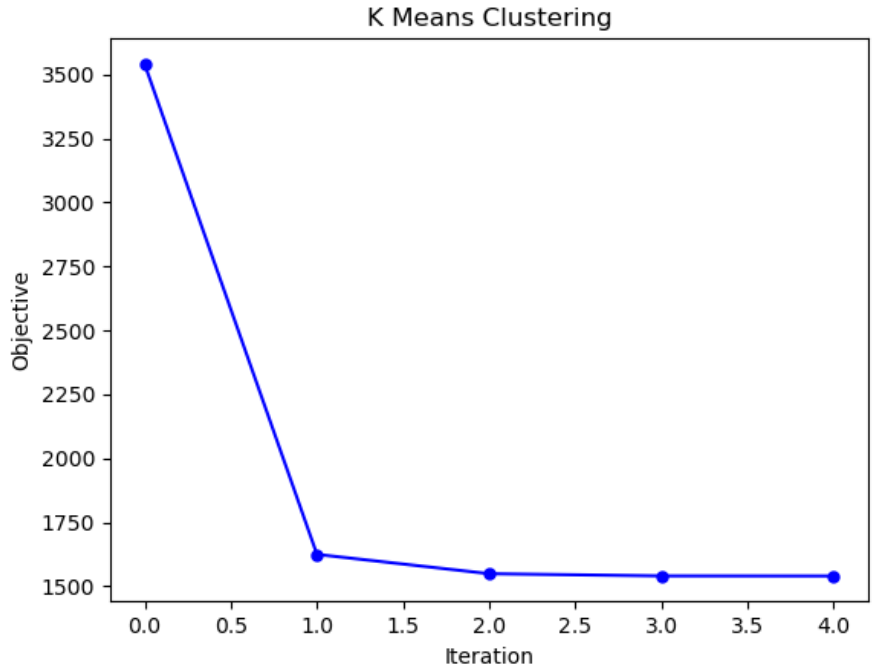


2nd Mean

K Means Clustering: using K Means ++

- Dataset: sklearn varied_blobs1 data set with 300 points
- Specify 3 clusters and pick initial means using kmeans++
- Set stopping tolerance $\varepsilon=10^{-5}$



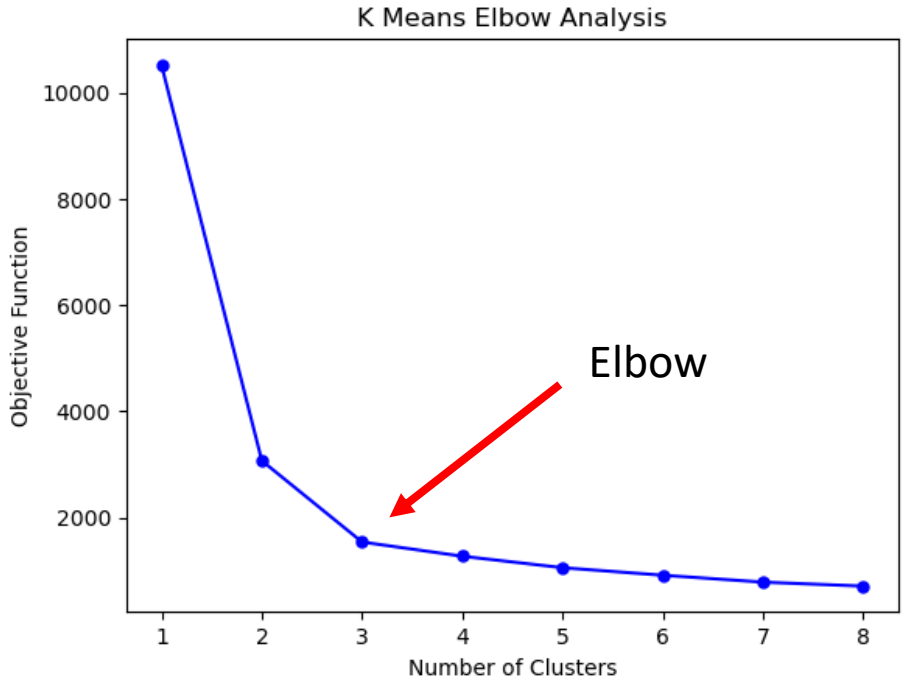


K Means++ Initialization: Notes

- Additional work to pick initial guesses is compensated by reduction in work in main K Means algorithm (fewer iterations required for convergence)

Determining Number of Clusters: Elbow Method

- K Means Elbow is heuristic approach for determining number of clusters
- Perform K Means for range of number of clusters and plot final objective function value
- Identify “Elbow” in plot to determine number of clusters



- Choose number of clusters K so that if number is greater than K then, objective function does not decrease significantly

Section 6.2: K Means: Code Design

K Means Code Design

- This section contains information about design of the K Means code
- Design is based on algorithm described in Section 7.1
- Stop video here, if you would like to do code design yourself

K Means Code Design: To Do

Component	Description
class kmeans	class kmeans
driver_kmeans	driver for kmeans algorithm
driver_kmeans_elbow	driver for performing kmeans elbow calculation to determine optimal number of clusters

kmeans class: Principal Variables

Variable	Type	Description
self.meansave	list of list of means	Contains cluster means for each iteration Example with 2 iterations and 3 means $[[\begin{bmatrix}1\\2\end{bmatrix},\begin{bmatrix}2\\1\end{bmatrix},\begin{bmatrix}3\\2\end{bmatrix}],[\begin{bmatrix}3\\1\end{bmatrix},\begin{bmatrix}2\\2\end{bmatrix},\begin{bmatrix}3\\3\end{bmatrix}]]$ meansave[i][j] is the mean for iteration i and cluster j
self.objectivesave	list of floats	Contains the value of K means objective function after each iteration: Example: case of 4 iterations [500, 400, 325, 200]
dist	2d numpy array	dist[i,j] is distance squared between data point j and mean i Example with 4 data points and 2 means $\begin{bmatrix}4 & 7.1 & 3.7 & 1.3\\2.2 & 3.1 & 2.3 & 1.7\end{bmatrix}$
self.ncluster	integer	Number of clusters

kmeans class – Key Methods

Method	Input	Description
__init__	ncluster (integer)	Constructor for the class – input the number of clusters
initialize_algorithm	initialization (string)	Initialize self.clustersave, self.objectivesave, and self.meansave using “random” or “kmeans++” initialization Return: nothing
fit	X (2d numpy array) max_iter (integer) tolerance(float)	Performs K means algorithm until distance between current and previous means is less than tolerance. Maximum of max_iter iterations. Return: self.objectivesave
compute_distance	list_mean (list numpy column vectors)	Compute distance squared between each data point and each mean in list_mean Return: dist (2d array)
update_cluster_ assignment		Compute the cluster assignments based on distance information in dist Return: nothing (update self.clustersave)

kmeans class – Key Methods

Method	Input	Description
update_objective		Append current estimate for objective function to self.objectivesave Return: nothing
update_mean		Update cluster means using latest cluster assignments Return nothing
compute_diff_mean		Determine maximum distance between current and previous estimate for means Return: maximum difference in means
plot_cluster	level (integer) title,xlabel,ylabel (strings)	Plot the clusters and the means for the specified level (iteration) Return: nothing
plot_cluster_animation	level (integer) title,xlabel,ylabel(strings)	Create animation showing data points and evolution of cluster assignments and means for iterations up to specified level (iteration) Return: nothing

Section 6.3: Hierarchical Clustering: Code Walkthrough

Hierarchical Clustering: Code Walkthrough

- Code is located in:

  Folder: UnsupervisedML/Code/Clustering

  Files: kmeans.py, driver_kmeans.py, driver_kmeans_elbow.py
- Stop video here, if you would like to do coding yourself before seeing my implementation