Unsupervised Machine Learning with Python

Section 6.1: K Means Clustering Algorithm

K Means Clustering: What is it?

- K Means Clustering is a centroid based approach
- User specifies the number of clusters K and an initial guess for the mean (centre) of each cluster
- K Means Clustering iteratively calculates improved guesses for the means and points associated with each cluster
- K means performs a “hard” clustering - each data point is assigned to exactly one cluster
- See UnsupervisedML_Resources.pdf for links to additional resources

K Means Clustering: Basic Step

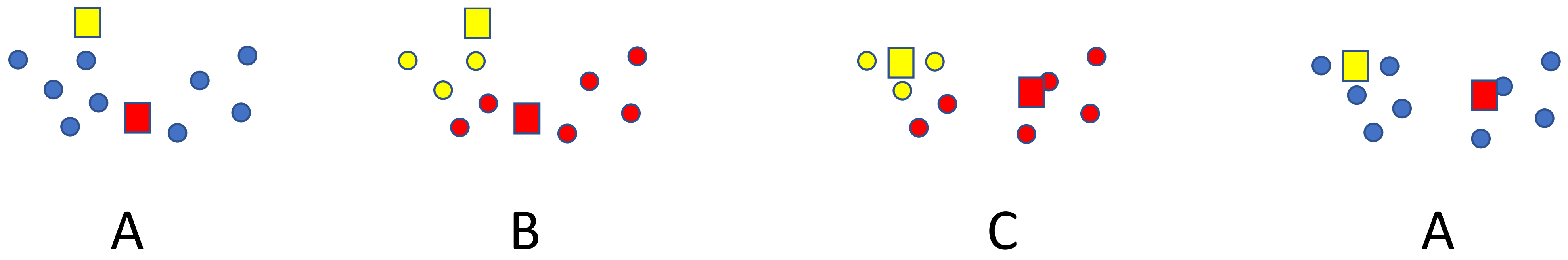
(A) Start with dataset and current estimate for cluster means

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

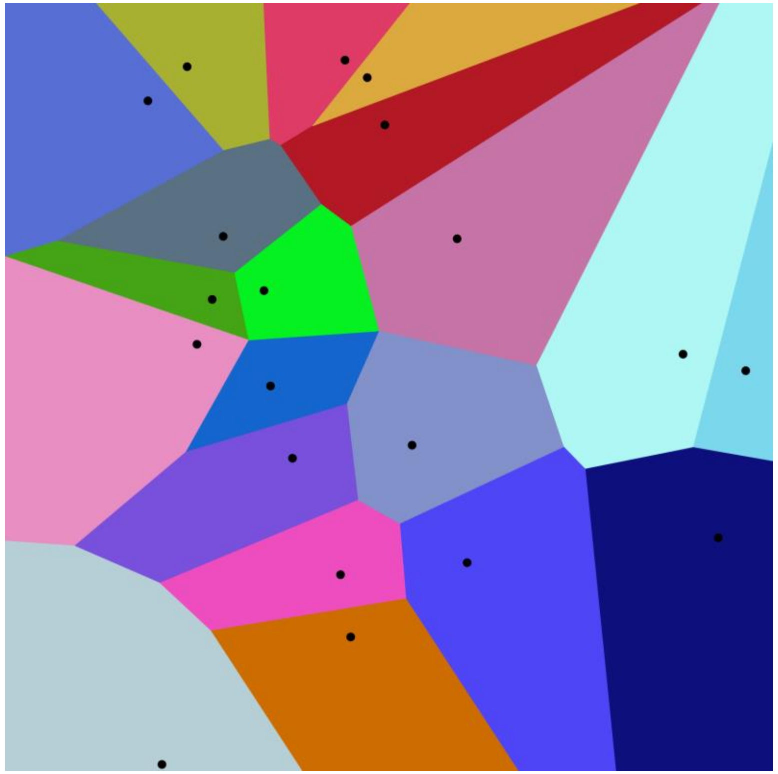
(C) Recompute cluster means based on assignments in (B)

Go back to (A) and repeat process until means converge (until change in cluster means is less than a tolerance)

Notice change in means between B and C



K Means Algorithm: Voronoi Diagram in 2D



- K Means splits plane into Voronoi cells
- Black dot is “mean”
- Shaded region around black dot is closer to its mean than any other mean
- Results for L2 distance measure (cells different for other distance measures)
- For K Means: Voronoi cell indicates extent of region where data point will assigned to region’s mean

See https://en.wikipedia.org/wiki/Voronoi_diagram

By Balu Ertl - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=38534275

K Means Algorithm

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

(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 cluster with closest mean C_k
   - Re-compute 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

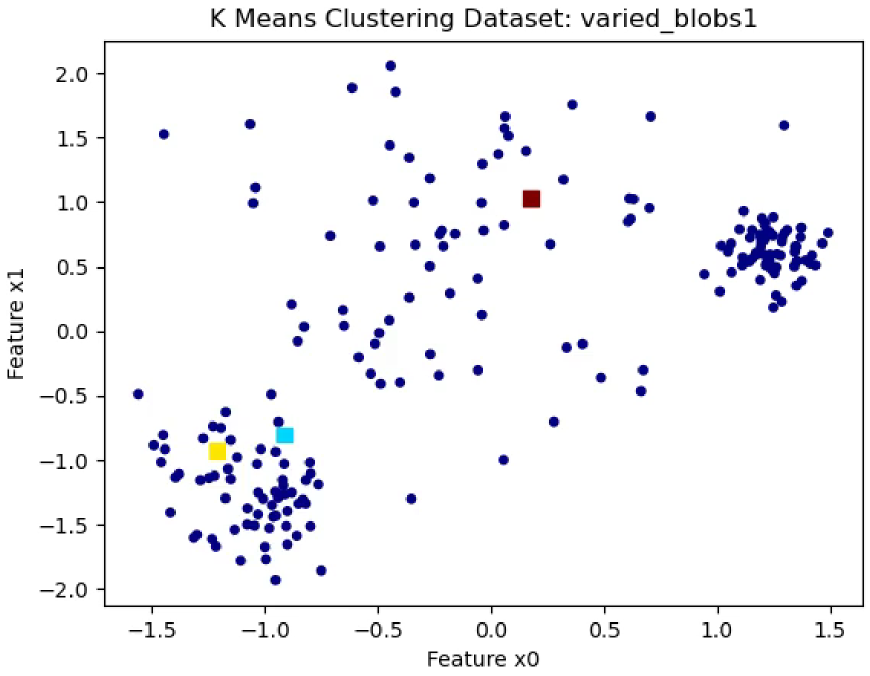
K Means Algorithm: Minimizing Objective Function

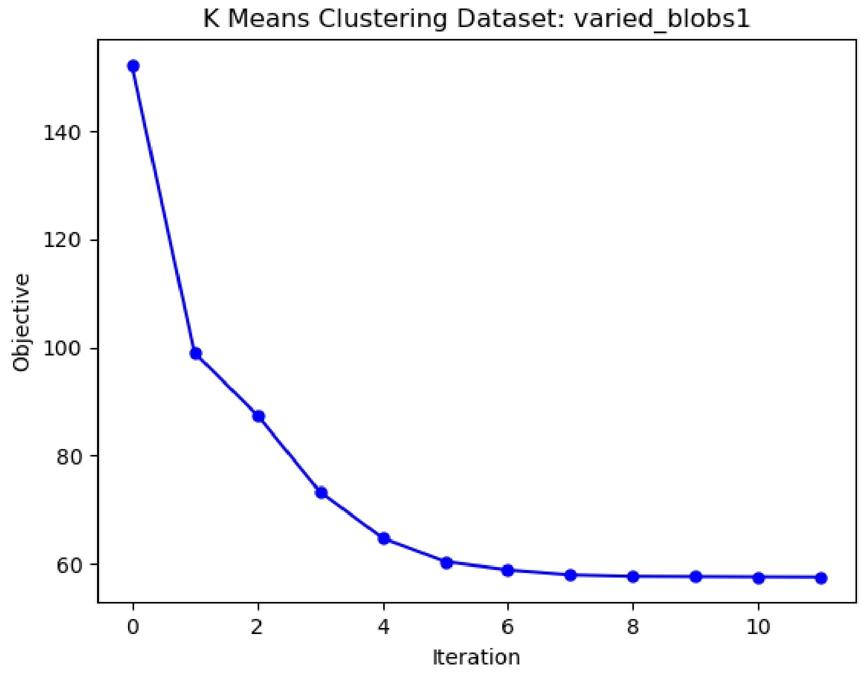
- Let {X} denote the data points (drop subscript for simplicity)
- Let S_k denote the set of points in cluster k=0,…,K-1 and let C_k denote the cluster mean
- K Means tries to partition the points into K cluster so as to minimize the objective function = within cluster sum of squares of distance:

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

K Means Clustering: Example

- Dataset: sklearn varied_blobs1 data set with 200 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 $M \to \infty$
- Amount of memory required is $O(M)$ as $M \to \infty$

K Means ++ for Picking Initial Cluster Means

(1) Pick point from dataset at random as Cluster Mean 1

(2) Loop for Cluster Means = 2, …, K

(a) For each data point X compute distance D(X) to nearest existing cluster mean

(b) Pick point with largest D(X) as next cluster mean

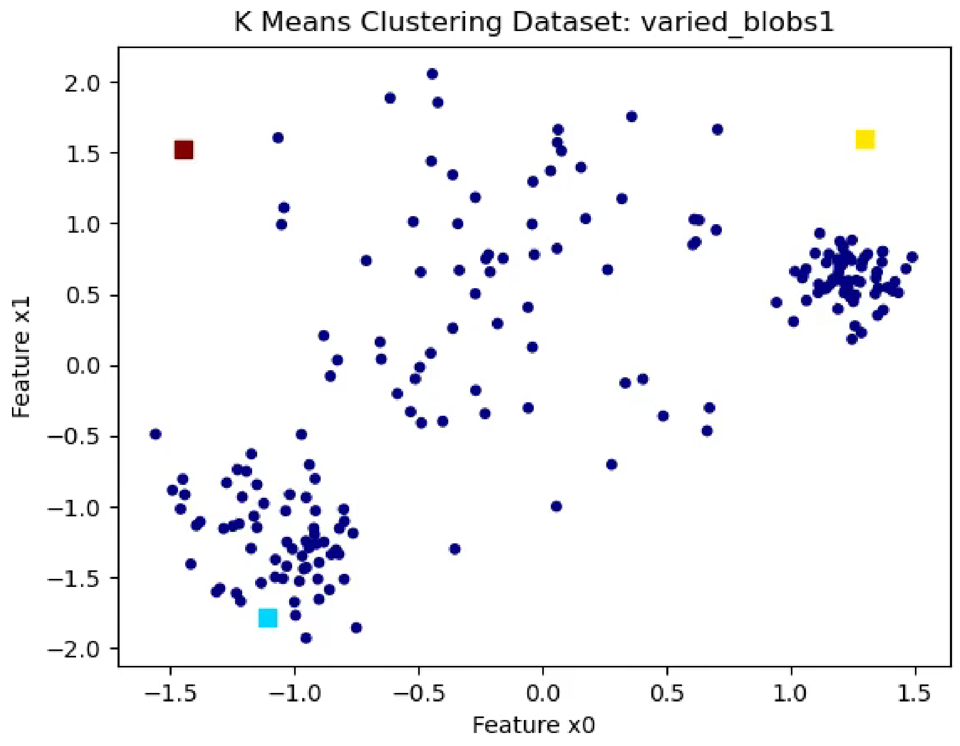
Notes:

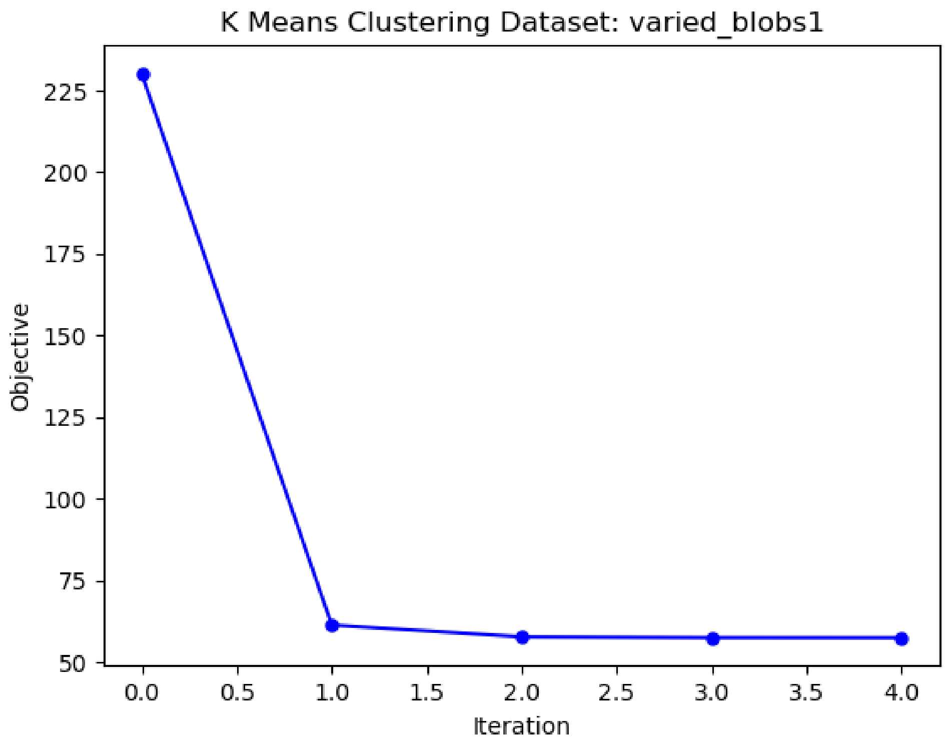
- After 1st mean, approach picks cluster means at the edges of the data set
- Typically, pick point in (b) at random with probability proportional to $[D(x)]^2$

Dataset 1st Mean 2nd Mean 3rd Mean

K Means Clustering: using K Means ++

- Dataset: sklearn varied_blobs1 data set with 200 points
- Specify 3 clusters and pick initial means using kmeans++
- Set stopping tolerance $\varepsilon=10^{-5}$



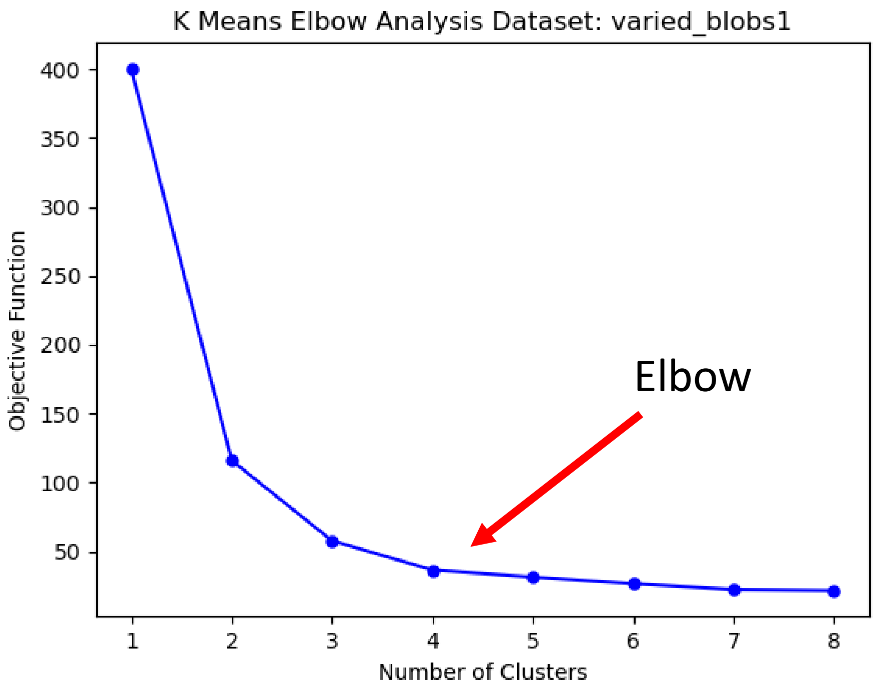


K Means++ Initialization: Notes

- Significantly fewer iterations required for convergence using kmeans++ versus random initialization
- Additional work to pick initial guesses using kmeans++ is compensated by reduction in work in main K Means algorithm

Determining Number of Clusters: Elbow Method

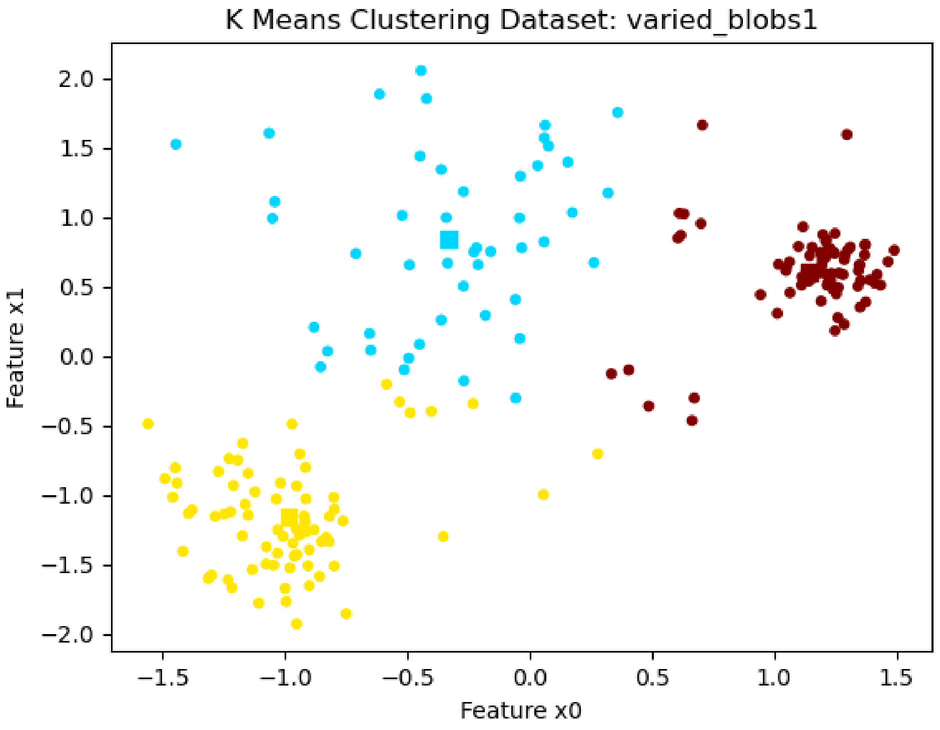
- K Means Elbow is heuristic approach for determining number of clusters
- Perform K Means for range of number of clusters and plot final objective function value
- Identify "Elbow" in plot to determine number of clusters

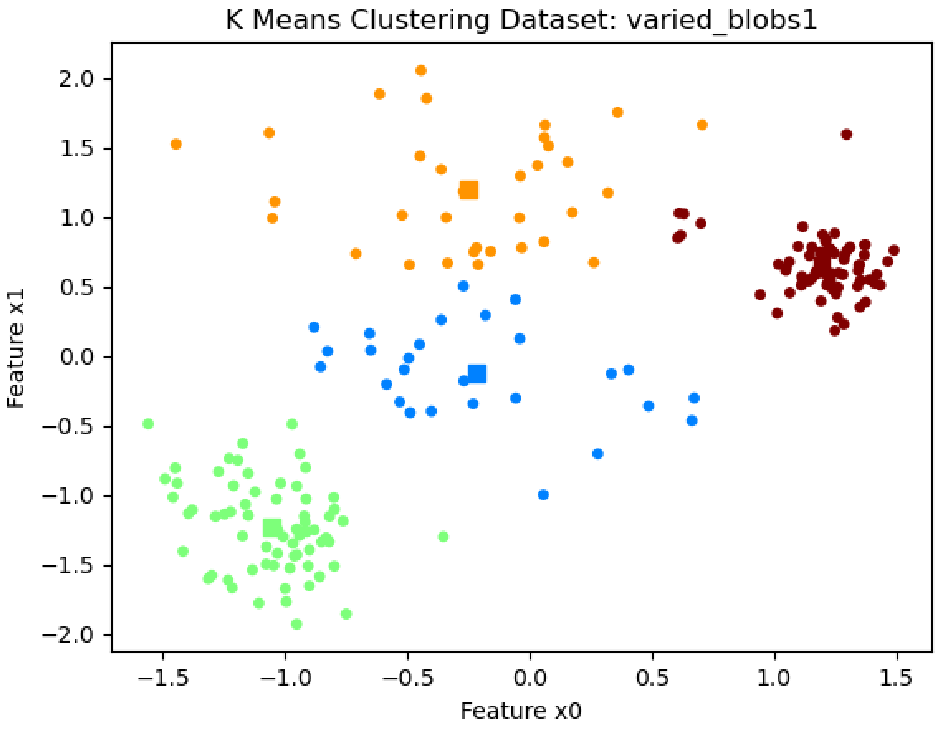


- Choose number of clusters K so that if number is greater than K then, objective function decreases gradually
- See Resources file for link for additional information

K Means Clustering: 3 versus 4 Clusters

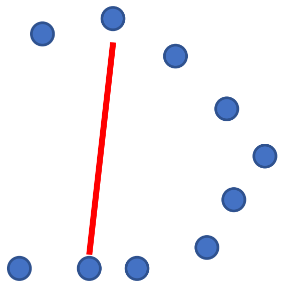
- Dataset: sklearn varied_blobs1 data set with 200 points
- Pick initial means at random
- Set stopping tolerance $\varepsilon=10^{-5}$



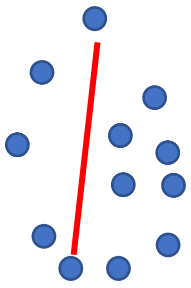


K Means Clustering: Notes

- User must specify number of clusters (can use elbow approach)
- User specifies distance measure (L2 is used in this course)
- 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 Clustering: Notes

K Means is an excellent starting point for clustering:

- Easy to implement and fast - $O(M)$ operations as $M \to \infty$
- Can use K Means ++ to choose initial means
- Can use Elbow Method to determine number of clusters K
- Can handle large numbers of dimensions

Unsupervised Machine Learning with Python

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 6.1 and makes use of numpy functionality
- Stop video here, if you would like to do code design yourself

K Means Code Design

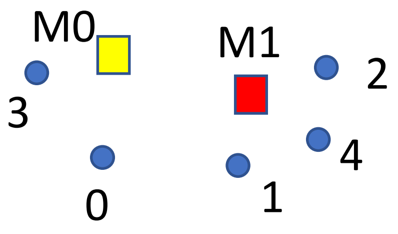
(1) Derive kmeans class from clustering_base class

(2) Key to implementation is Distance Squared Matrix

(3) Create K Means versions of plot_cluster and plot_cluster_animation to be able to plot evolution of cluster means

Distance Squared Matrix

- Consider example with 5 data points and 2 clusters



- dist2[i,j] = distance squared between mean i and point j
- Example:

$$dist2 = \begin{bmatrix} 1.1 & 5.4 & 4.3 & 0.5 & 1.4 \\ 2.1 & 0.9 & 0.7 & 3.5 & 4.4 \end{bmatrix}$$

Distance squared between data points and M0

Distance squared between data points and M1

Computing Cluster Assignment

Start with dist2 matrix:

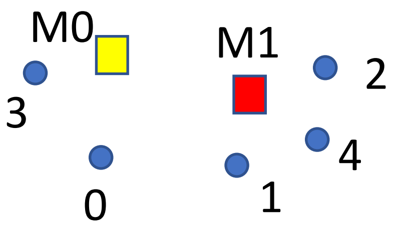
$$dist2 = \begin{bmatrix} \boxed{1.1} & 5.4 & 4.3 & \boxed{0.5} & 4.4 \\ 2.1 & \boxed{0.9} & \boxed{0.7} & 3.5 & \boxed{1.4} \end{bmatrix}$$

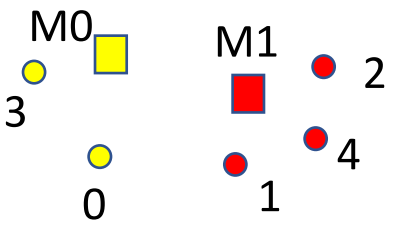
Distance squared between data points and M0

Distance squared between data points and M1

Cluster assignment

- Point is assigned to cluster with closest mean
  - For each data point determine row index of dist2 with smallest entry (numpy argmin)
- Cluster Assignment = $\begin{bmatrix} \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} \end{bmatrix}$





Objective Function

Start with dist2 matrix:

$$dist2 = \begin{bmatrix} \boxed{1.1} & 5.4 & 4.3 & \boxed{0.5} & 4.4 \\ 2.1 & \boxed{0.9} & \boxed{0.7} & 3.5 & \boxed{1.4} \end{bmatrix}$$

Distance squared between data points and M0

Distance squared between data points and M1

Objective Function

- Let S_k denote cluster k=0,…,K-1 and let C_k denote mean

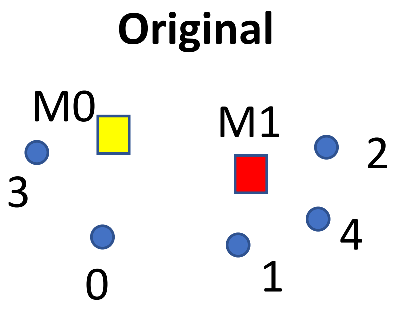
$$Objective = \sum_{k=0}^{K-1} \sum_{X \in S_k} dist(X, C_k)^2$$

- Equivalent to sum of distance squared to closest mean
- Distance squared to closest mean: $\begin{bmatrix} \boxed{1.1} & \boxed{0.9} & \boxed{0.7} & \boxed{0.5} & \boxed{1.4} \end{bmatrix}$
  - Use numpy min function
- Sum is 4.6

Updating Cluster Means

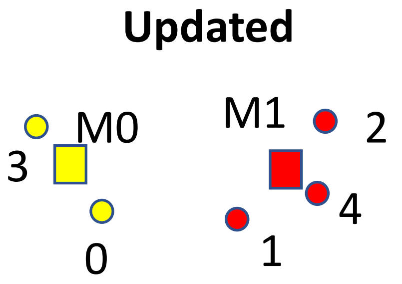
- Data

$$X = \begin{bmatrix} 1 & 2 & -1 & -2 & 3 \\ 3 & 4 & -4 & -5 & 6 \\ 5 & 6 & -7 & -8 & 9 \end{bmatrix}$$



- Cluster assignment: $Assignment = \boxed{[0}\ \boxed{1}\ \boxed{1}\ \boxed{0}\ \boxed{1]}$
- Update cluster means using assigmments
  - use numpy mean in column direction

$$M0 = mean(X[col = \boxed{0,3]}]) = mean\begin{bmatrix} 1 & -2 \\ 3 & -5 \\ 5 & -8 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ -1.5 \end{bmatrix}$$



$$M1 = mean(X[col = \boxed{1,2,4]}]) = mean\begin{bmatrix} 2 & -1 & 3 \\ 4 & -4 & 6 \\ 6 & -7 & 9 \end{bmatrix} = \begin{bmatrix} 1.33 \\ 2 \\ 2.66 \end{bmatrix}$$

kmeans class: Principal Variables

Variable	Type	Description
self.time_fit	float	Time for clustering
self.objectivesave	list	Value of the objective function for each iteration Example with 3 iterations: [525, 425, 310]
self.X	2d numpy array	Dataset Number of rows = number of dimensions for data Number of cols = number of data points Example: 2 dimensions and 5 data points: $\begin{bmatrix} 1 & 1.1 & 0.8 & 0.6 & 0.6 \\ 0.9 & 1.0 & 0.7 & 0.5 & 0.5 \end{bmatrix}$
self.clustersave	list of 1d numpy arrays	self.clustersave[i][j] is cluster assignment for iteration i, data point j Example: for 3 iterations: $[[-1 \quad -1 \quad -1 \quad -1 \quad -1], [0 \quad 1 \quad 1 \quad 0 \quad 1], [0 \quad 0 \quad 1 \quad 0 \quad 1]]$
self.meansave	list of list of means	Cluster means: meansave[i][j] is the mean for iteration i and cluster j Example with 3 iterations and 2 means $[[\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}],[\begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \end{bmatrix}],[\begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \end{bmatrix}]]$
dist2	2d numpy array	dist2[i,j] is distance squared between mean i and data point j Example with 2 means and 5 data points: $\begin{bmatrix} 1.1 & 5.4 & 4.3 & 0.5 & 1.4 \\ 2.1 & 0.9 & 0.7 & 3.5 & 4.4 \end{bmatrix}$

kmeans class – Key Methods

Method	Input	Description
__init__	ncluster (integer) initialization (string)	Constructor for the class – input the number of clusters and initialization method (“random” or “kmeans++”)
initialize_algorithm		Initialize self.clustersave, self.objectivesave, and self.meansave Return: nothing
fit	X (2d numpy array) max_iter (integer) tolerance(float) verbose (boolean)	Performs K means algorithm until distance between current and previous means is less than tolerance for maximum of max_iter iterations. Return: nothing
compute_distance2	list_mean (list numpy column vectors)	Compute distance squared between each data point and each mean in list_mean Return: dist2 (2d numpy array) See: UnsupervisedML/Examples/Section03/Distance.ipynb
update_cluster_assignment	dist2 (2d numpy array)	Compute the cluster assignments based on info in dist2 Return: nothing (update self.clustersave)

kmeans class – Key Methods

Method	Input	Description
update_objective	dist2 (2d numpy array)	Compute latest value of objective function Return: nothing (append to self.objectivsave)
update_mean		Compute means based on current cluster assignments Return: nothing (append to self.meansave)
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 specified level (iteration) Return: nothing See UnsupervisedML/Exercises/Section02/Exercise_2.4.2.ipynb
plot_cluster_animation	level (integer) interval (float) 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 See UnsupervisedML/Exercises/Section02/Exercise_2.4.2.ipynb

Unsupervised Machine Learning with Python

Section 6.3: K Means Clustering Code Walkthrough

K Means Clustering: Code Walkthrough

Code located at:

- UnsupervisedML/Code/Programs

Files to Review	Description
clustering_base.py	Base class for clustering
kmeans.py	Class for K Means clustering
driver_kmeans.py	Driver for K Means clustering

Course Resources at:

- https://github.com/satishchandrareddy/UnsupervisedML/

- Stop video if you would like to implement code yourself first