
DEFENCE INSTITUTE OF ADVANCED TECHNOLOGY
DEEMED
UNIVERSITY
शास्त्रेण
रक्षाम्
शस्त्रं प्रकरोति

Machine Learning for Cyber Security (CS-602)

L#08

KMeans Clustering

By

Dr Sunita Dhavale

Syllabus

- Data Analytics Foundations: R programming, Python Basics -Expressions and Variables, String Operations, Lists and Tuples, Sets, Dictionaries Conditions and Branching, Loops, Functions, Objects and Classes, Reading/Writing files, Hand ling data with Pandas, Scikit Library, Numpy Library, Matplotlib, scikit programming for data analysis, setting up lab environment, study of standard datasets. Introduction to Machine Learning- Applications of Machine Learning, Supervised, unsupervised classification and regression analysis
- Python libraries suitable for Machine Learning Feature Extraction. Data pre-processing, feature analysis etc., Dimensionality Reduction & Feature Selection Methods, Linear Discriminant Analysis and Principal Component Analysis, tackle data class imbalance problem

Syllabus

- Supervised and regression analysis, Regression, Linear Regression, Non-linear Regression, Model evaluation methods, Classification, K-Nearest Neighbor, Naïve Bayes, Decision Trees, Logistic Regression, Support Vector Machines, Artificial Neural Networks, Model Evaluation. Ensemble Learning, Convolutional Neural Networks, Spectral Embedding, Manifold detection and Anomaly Detection
- Unsupervised classification K-Means Clustering, Hierarchical Clustering, Density-Based Clustering, Recommender Systems-Content-based recommender systems, Collaborative Filtering, machine learning techniques for standard dataset, ML applications, Case studies on Cyber Security problems that can be solved using Machine learning like Malware Analysis, Intrusion Detection, Spam detection, Phishing detection, Financial Fraud detection, Denial of Service Detection.

Text/Reference Books

1. Building Machine Learning Systems with Python – Willi Richert, Luis Pedro Coelho
2. Alessandro Parisi, Hands-On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies Publication date :Aug 2, 2019, Packt, ISBN-13, 9781789804027
3. Machine Learning: An Algorithmic Perspective – Stephen Marsland
4. Sunita Vikrant Dhavale, "Advanced Image-based Spam Detection and Filtering Techniques", IGI Global, 2017
5. Soma Halder , Sinan Ozdemir, Hands-On Machine Learning for Cybersecurity: Safeguard your system by making your machines intelligent using the Python ecosystem, By Publication date : Dec 31, 2018, Packt, ISBN-13 :9781788992282

1. Stuart Russell, Peter Norvig (2009), "Artificial Intelligence – A Modern Approach", Pearson Elaine Rich & Kevin Knight (1999), "Artificial Intelligence", TMH, 2nd Edition
2. NP Padhy (2010), "Artificial Intelligence & Intelligent System", Oxford
3. ZM Zurada (1992), "Introduction to Artificial Neural Systems", West Publishing Company
4. Research paper for study (if any) – White papers on multimedia from IEEE/ACM/Elsevier/Spinger/ Nvidia sources.

Lab assignments

1	**Python Programming part-1**
2	Python Programming part-2
3	Study and Implement Linear Regression Algorithm for any standard dataset like in cyber security domain
4	Study and Implement KMeans Algorithm for any standard dataset in cyber security domain
5	Study and Implement KNN for any standard dataset in cyber security domain
6	Study and Implement ANN for any standard dataset in cyber security domain
7	Study and Implement PCA for any standard dataset in cyber security domain
8	Case Study: Use of ML along with Fuzzy Logic/GA to solve real world Problem in cyber security domain
9	Mini assignment: Apply ML along with PSO/ACO to solve any real world problem in cyber security domain
10	ML Practice Test – 1 Quiz

Defence Institute of Advanced Technology

School of Computer Engineering & Mathematical Sciences

SEMESTER-I TIME TABLE (AUTUMN 2024)$

PROGRAMMES: (I) CS [M.TECH IN CYBER SECURITY] (II) AI [M.TECH CSE (ARTIFICIAL INTELLIGENCE)] BATCH: 2024-2026

Lecture / Day	L1 0900-1000	L2 1000-1100	L3 1100-1200	L4 1200-1300	Lunch Break 1300-1400	L4 1400-1500	L4 1500-1600	L4 1600-1700	L4 1700-1800
Monday	CE-602 (AI) CS-602 (CS)	CE-604 (AI) CS-603 (CS)	CE-601 (AI) CS-604 (CS)	CE-601 (AI) LAB CS-603 (CS)		LAB CE-601 (AI) LAB CS-602 (CS)		AM607	
Tuesday	CE-603 (AI) LAB CS-603 (CS)	CE-602 (AI) CS-602 (CS)	CE-601 (AI) CS-605 (CS)	CE-604 (AI) CS-604 (CS)		PGC 601		AM607	
Wednesday	CS-605 (CS)	CE-603 (AI) CS-602 (CS)	CE-602 (AI) CS-603 (CS)	CE-604 (AI) CS-604 (CS)		CE-605(AI) LAB CS-605 (CS)	LAB CS-605 (CS)	AM607	
Thursday	LAB CE-604 (AI) CS-603 (CS)	LAB CE-604 (AI) CS-605 (CS)	LAB CE-602 (AI) CS-601 (CS)	CE-603 (AI) CS-601 (CS)		PGC 601		AM607	
Friday	LAB CE-603 (AI) LAB CS-601 (CS)		LAB CE-602 (AI) CS-601 (CS)	LAB CS-604 (CS)		CE-605(AI) LAB CS-604 (CS)	CE-605(AI)	LAB CE-605(AI)	

COURSE CODE & COURSE NAME		FACULTY
Programme: CS [M.Tech in Cyber Security] Classroom: Arjun	Programme: AI [M.Tech CSE (Artificial Intelligence)] Classroom: Kaveri	
CS-601 Data Security & Privacy	CE-601 Responsible Artificial Intelligence;	MJN: Dr. Manisha J. Nene
CS-602 ML for Cyber Security	CE-604 Practical Machine Learning;	SVD: Dr. Sunita V. Dhavale
CS-605 Network and Cloud Security	CE-602 Intelligent Algorithms	CRS: Prof. CRS Kumar
CS-604 Advanced System Security	--------	DVV: Dr. Deepti V. Vidyarthi
CS-603 Applied Cryptography	--------	AM: Dr. Arun Mishra
--------	CE-603 Deep Neural Network;	US: Dr. Upasna Singh
--------	CE-605 Mathematics for ML;	Unit-2: Dr Upasna, Unit 4: Dr Sunita, Unit3:MIN, Unit 1: Faculty To be Nominated
AM-607 Mathematics for Engineers	AM-607 Mathematics for Engineers	OO/DS/DP: Dr Odellu O., Dr Dasari S., Dr. Debasis P.
PGC-601 Research Methodology	PGC-601 Research Methodology	Common Subject for All

$ TENTATIVE T.T. SUBJECT TO CHANGE

Program Coordinator,
M.Tech (CS & AI), Batch 2024-26

Director, SoCE&MS

KMeans Clustering

What is clustering?

- Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.
- reveal interesting groups in the data

K means clustering

- K means clustering, assigns data points to one of the K clusters depending on their distance from the center of the clusters.
- It starts by randomly assigning the clusters centroid in the space.
- Then each data point assign to one of the cluster based on its distance from centroid of the cluster.
- After assigning each point to one of the cluster, new cluster centroids are assigned.
- This process runs iteratively until it finds good cluster.
- In the analysis we assume that number of cluster is given in advanced and we have to put points in one of the group.

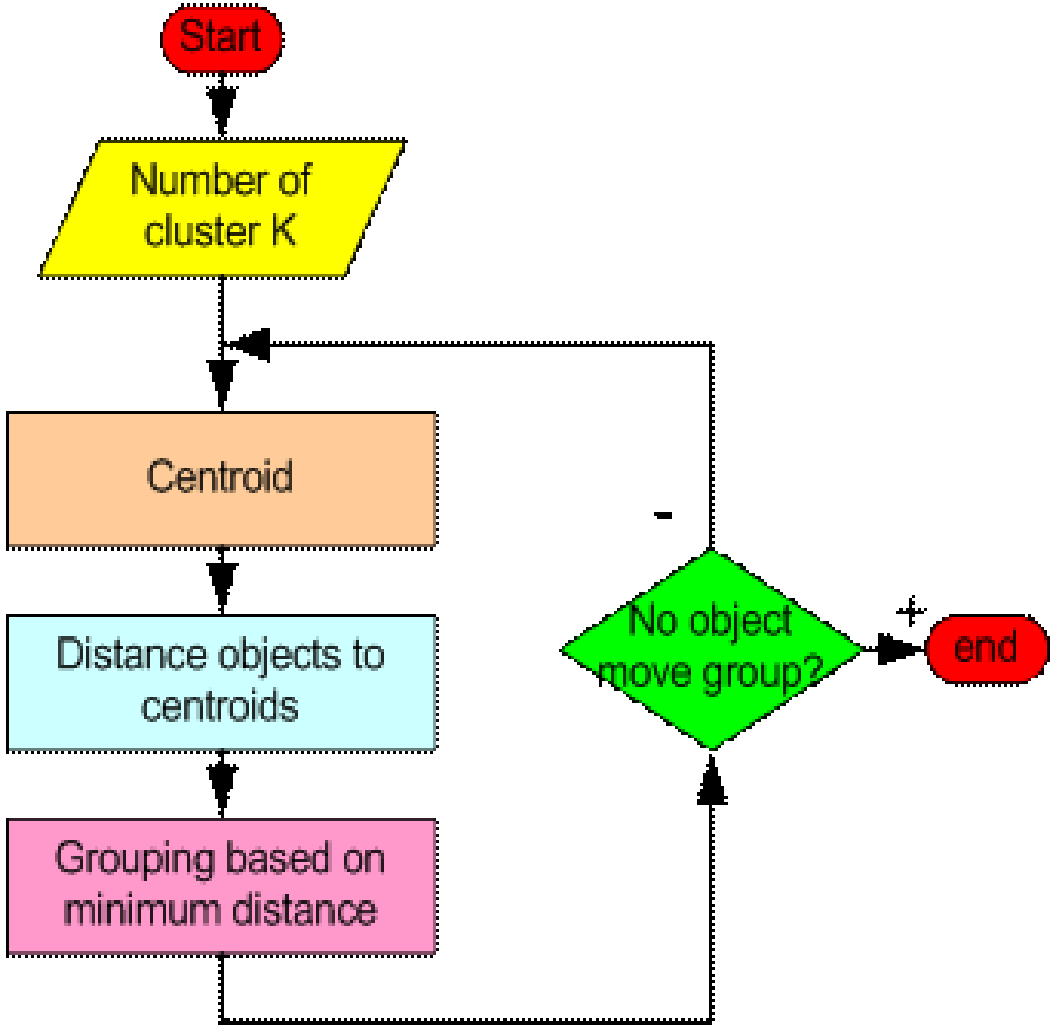
K-MEANS CLUSTERING

- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.
- attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

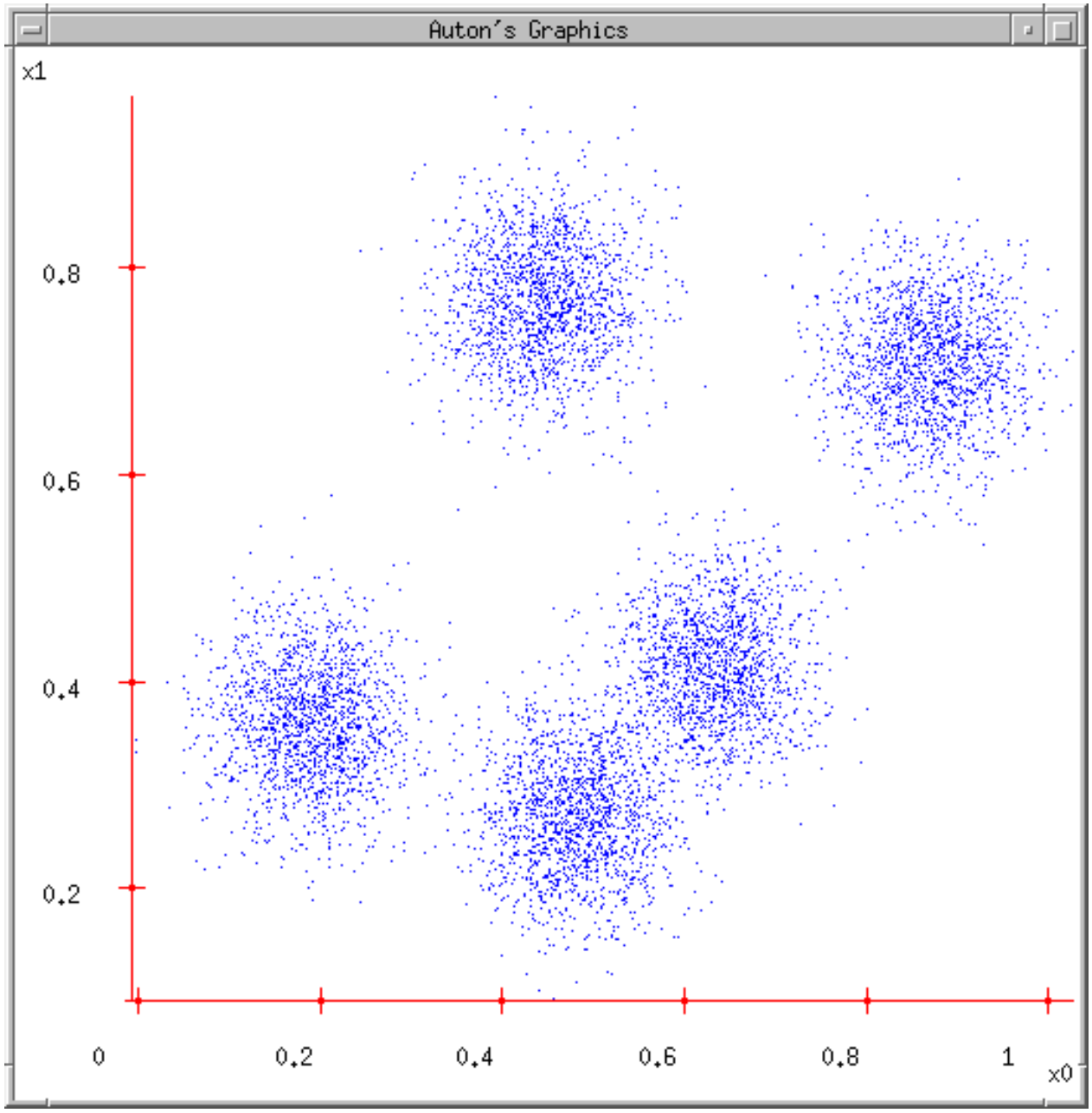
$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - \mu_j \right|^2,$$

where x_n is a vector representing the n^{th} data point and u_j is the geometric centroid of the data points in S_j.

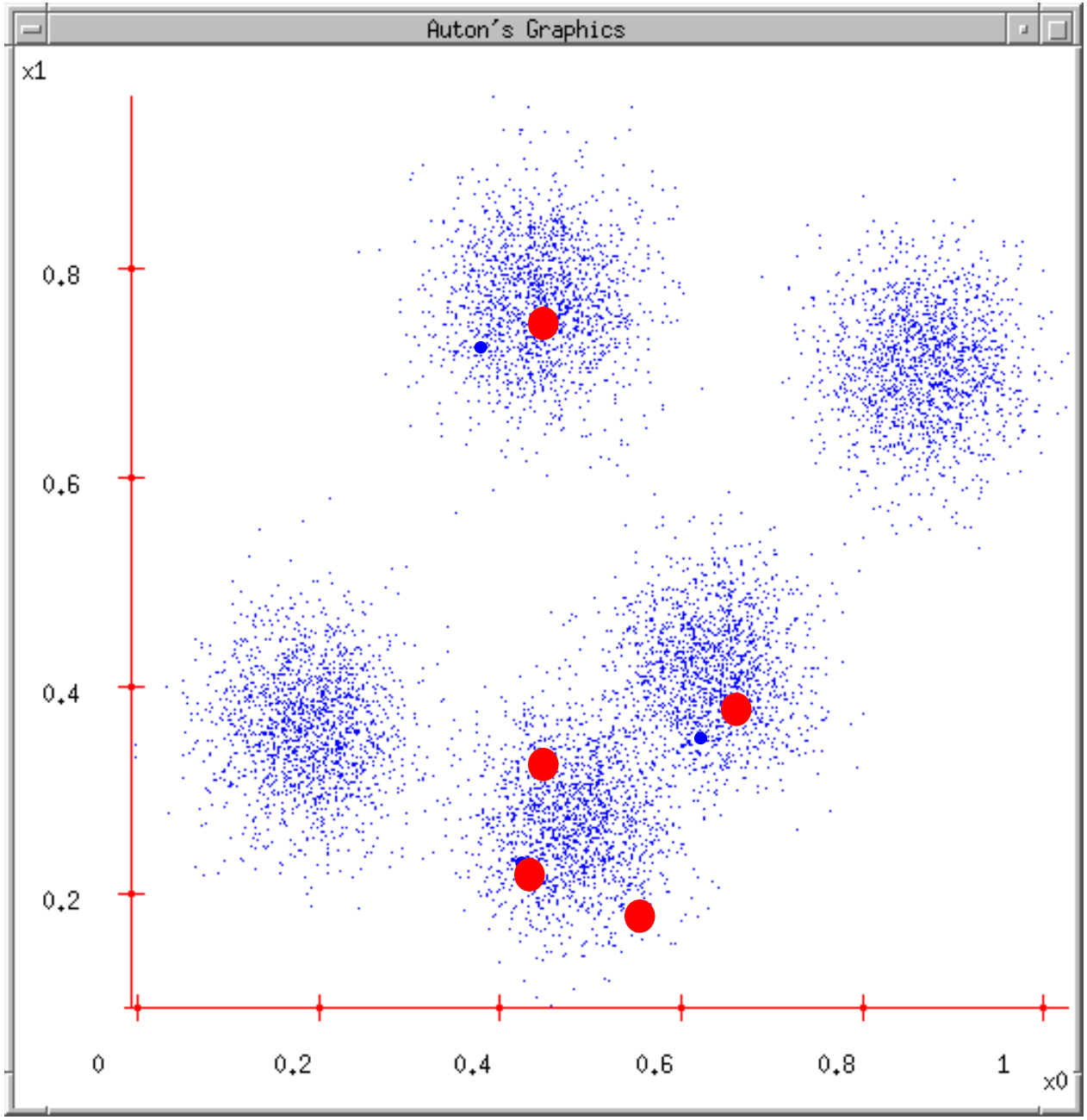
Algorithm


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

1. Ask user how many clusters they'd like. *(e.g. k=5)*

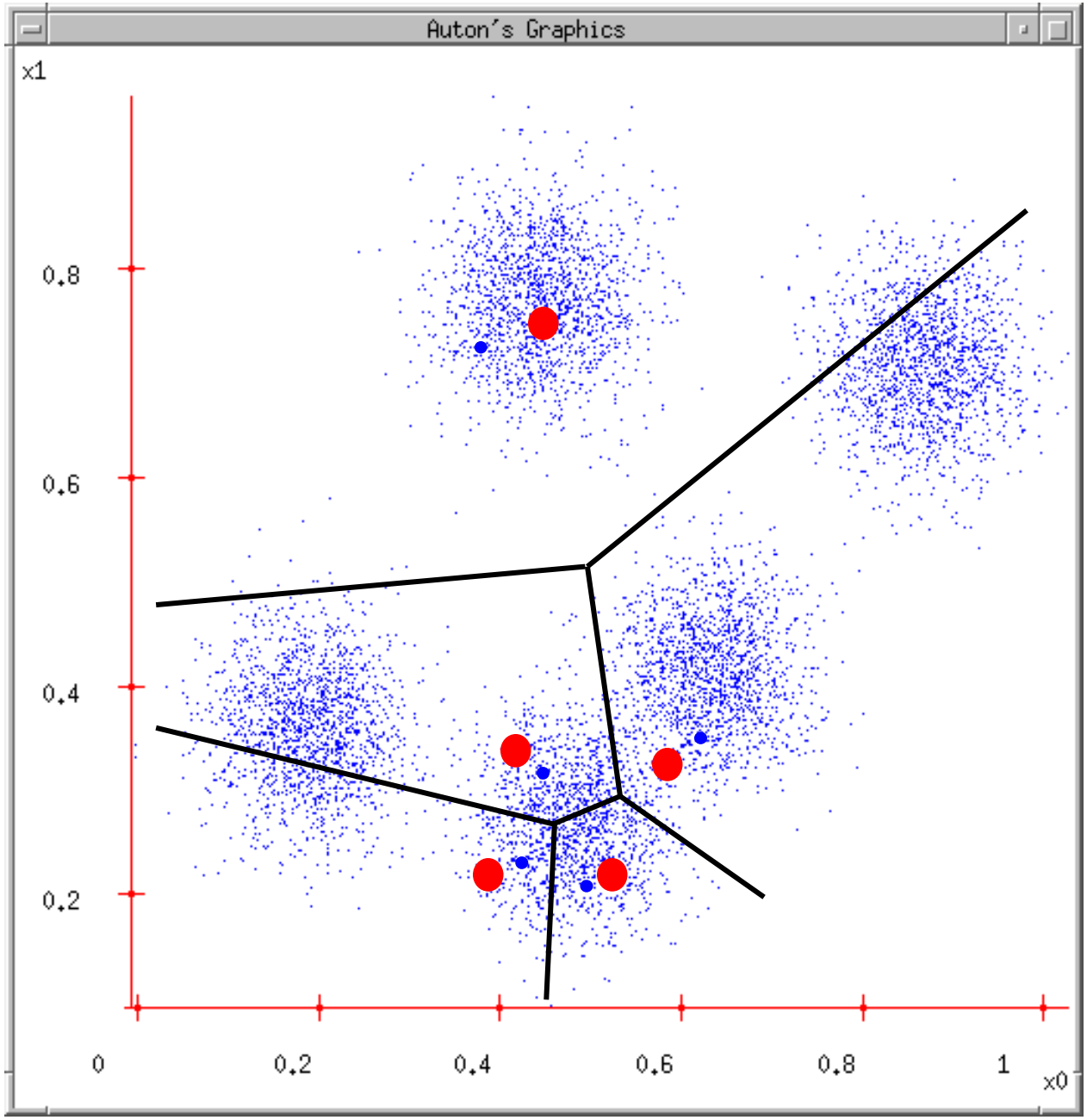


1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



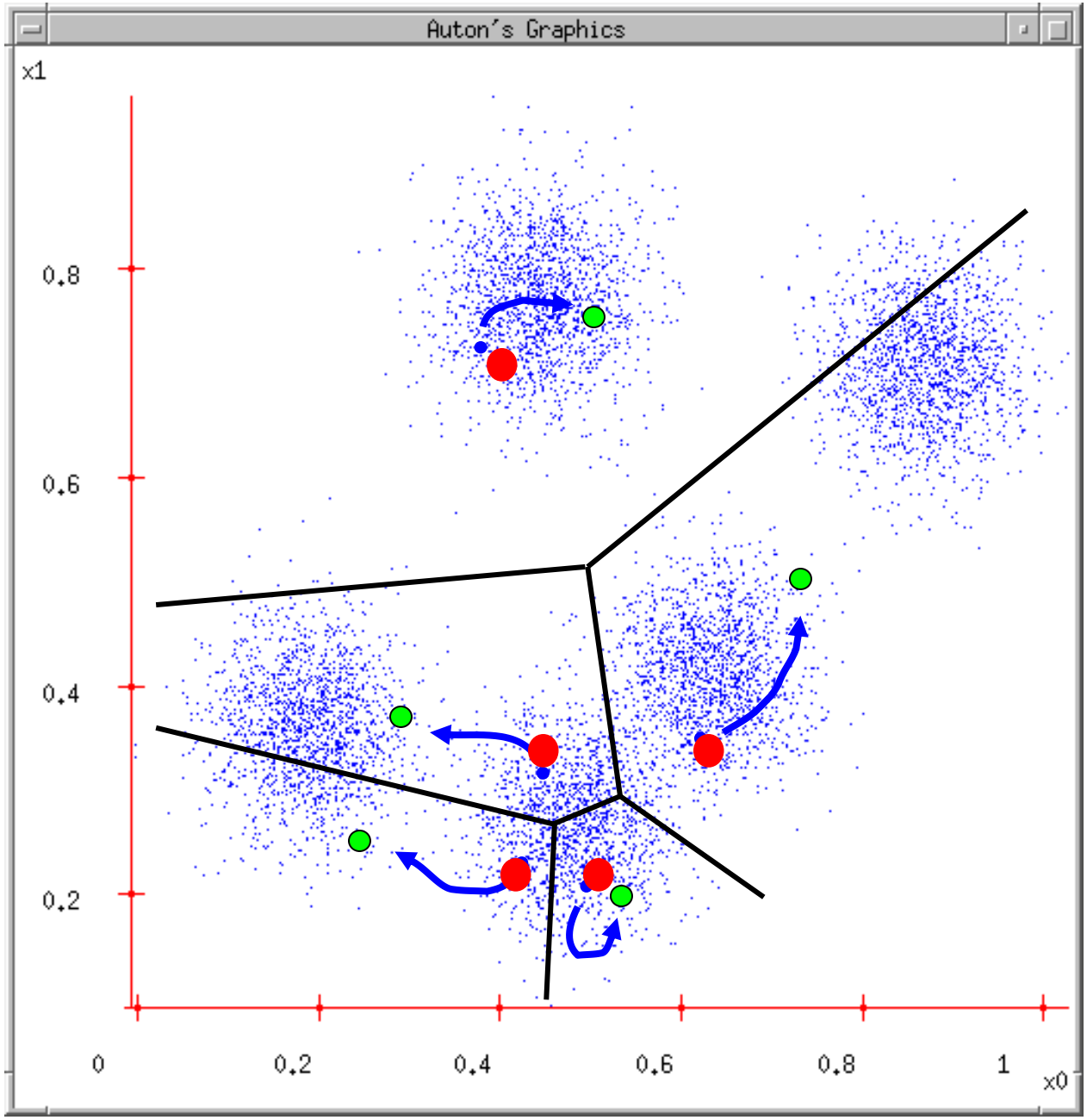
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



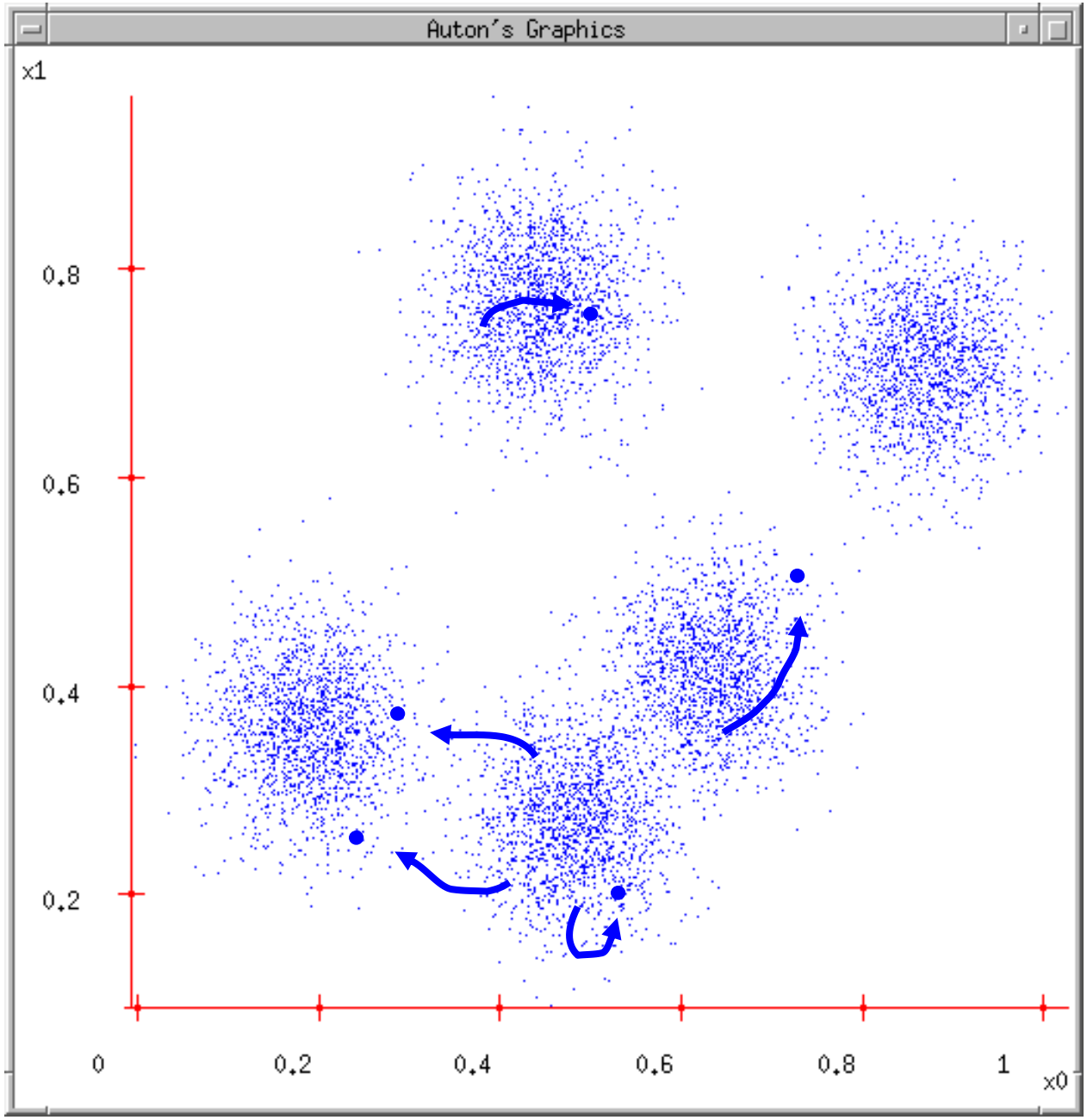
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Example (using K=2)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

- Thus, we obtain two clusters containing:

  {1,2,3} and {4,5,6,7}.

- Their new centroids are:

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

$$d(m_1, 2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

- Therefore, the new clusters are:

  {1,2} and {**3**,4,5,6,7}

- Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- Step 4 :

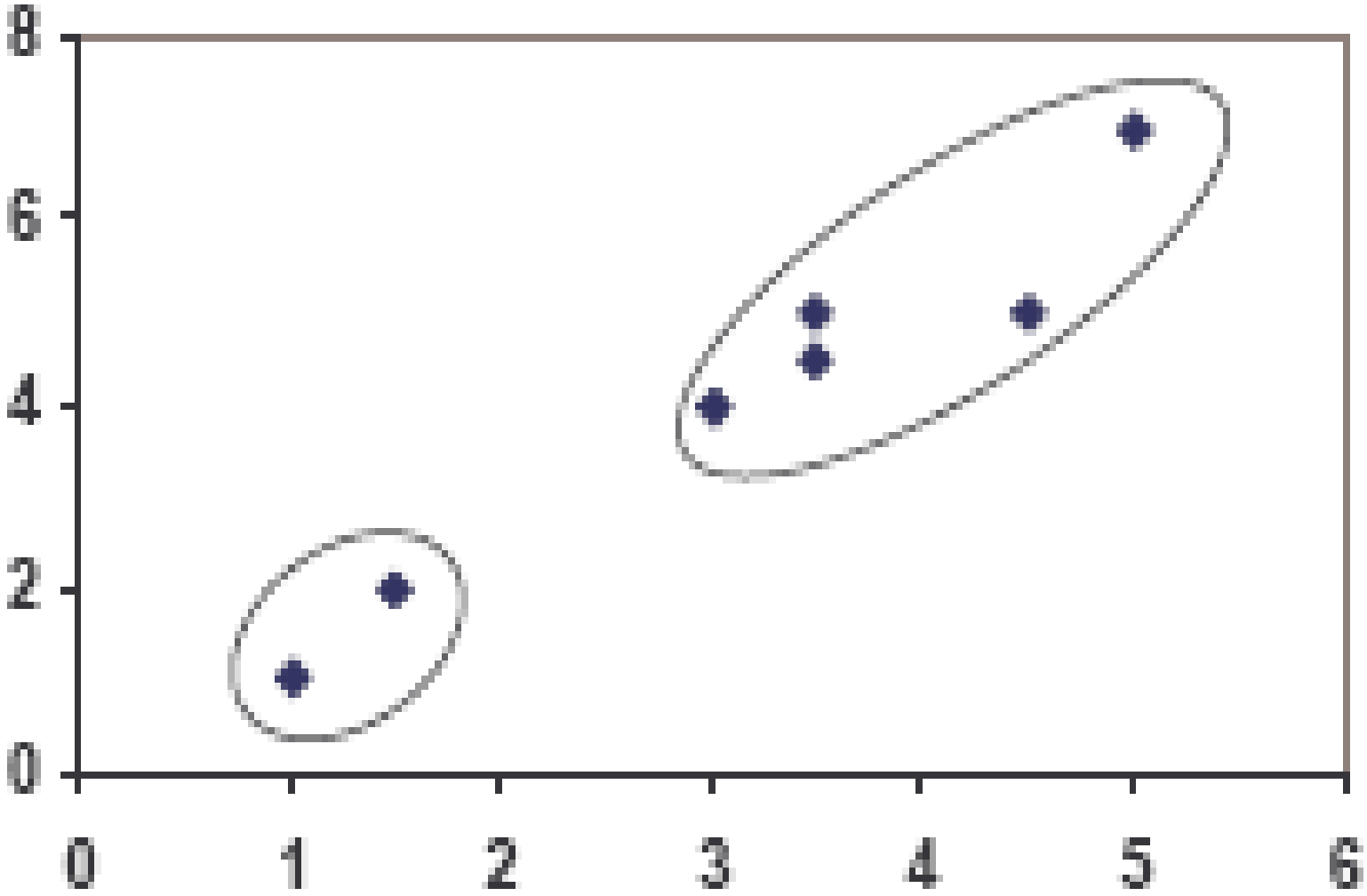
  The clusters obtained are:

  {1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT

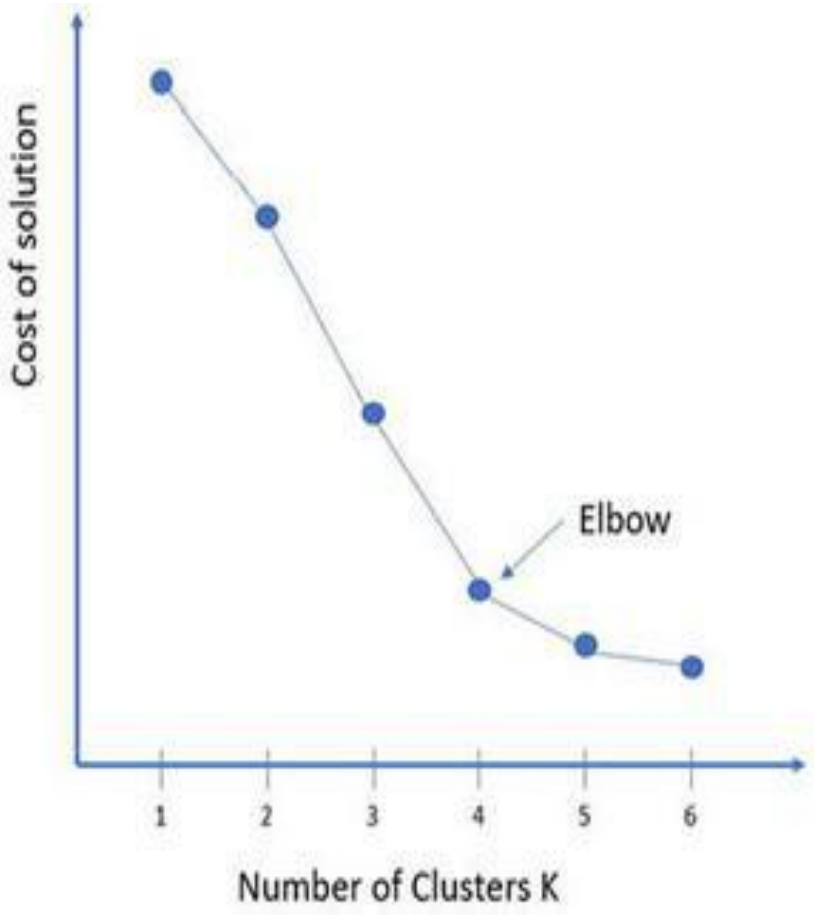


Pros/cons

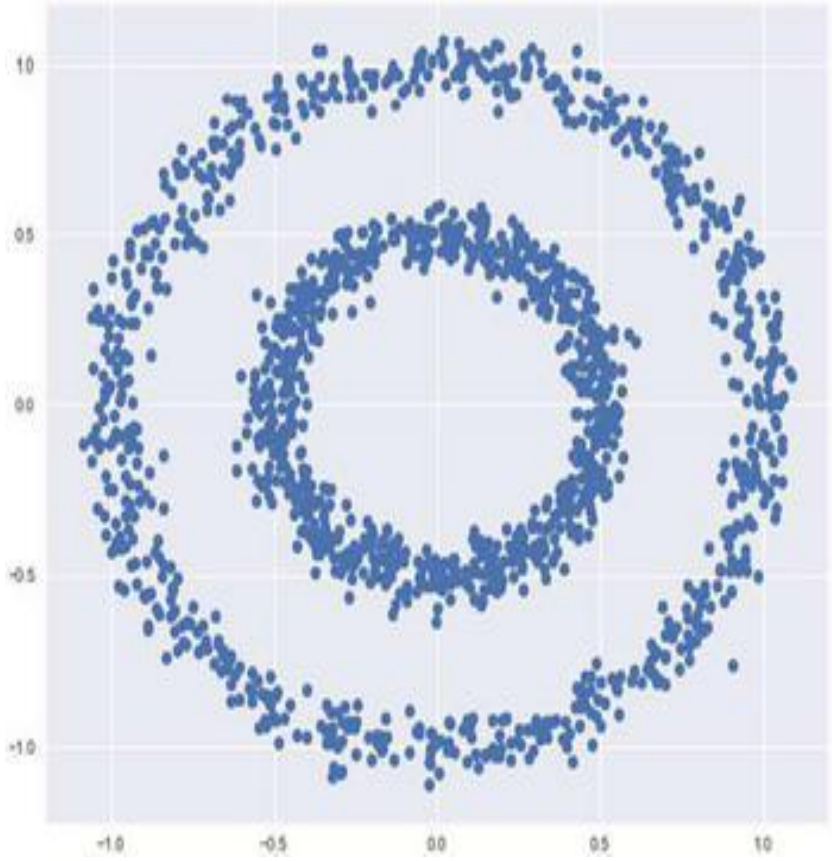
1. It is relatively *efficient and fast.* It computes result at **O(tkn),** where n is number of objects or points, k is number of clusters and t is number of iterations.
2. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
3. The number of cluster, K, must be determined before hand. Hence, it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the *local optimum*.
5. performs best if data is well separated. When data points overlapped this clustering is not suitable.
6. K Means cluster do not provide clear information regarding the quality of clusters.
7. K Means algorithm is sensitive to noise. It may have stuck in local minima.
8. https://www.geeksforgeeks.org/difference-between-k-means-and-hierarchical-clustering/

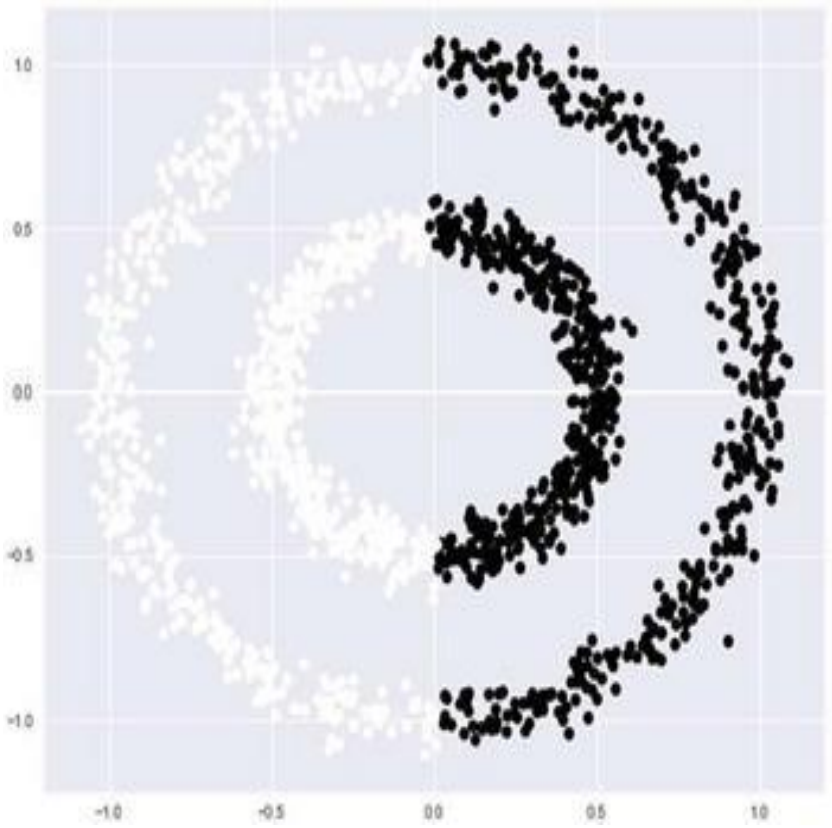
Finding Value of K

- Selecting the right value of K is very important.
- we use a technique called Elbow method to determine the optimal value of K.
- We use different value of K starting from 1 and keep increasing it.
- We plot a graph on value of K and cost of assignment.
- After some value of K, cost is not decreased significantly.
- plot shows an elbow.
- That value where elbow appears is taken as optimal value of K.



limitation of KMeans





RCode

- x=c(2,1,3,2,4)
- y=c(0,3,5,2,6)
- x=c(1, 1.5,3.0, 5.0, 3.5, 4.5, 3.5)
- y=c(1,2,4, 7, 5, 5, 4.5)
- z=data.frame(x,y)
- ic=kmeans(z,2,nstart=2)
- Ic
- ic$cluster
- ic$centers
- plot(z, col=ic$cluster)
- library(ggplot2)
- ggplot(z, aes(x,y, color = as.factor(ic$cluster))) + geom_point()

RCode

- **require**("datasets")
- data("iris") *# load Iris Dataset*
- str(iris) *#view structure of dataset*
- summary(iris) *#view statistical summary of dataset*
- head(iris) *#view top rows of dataset*
- irisdata<- iris[,c(1,2,3,4)]
- irisclass<- iris[,"Species"]
- head(irisdata)
- head(irisclass)
- result<- kmeans(irisdata,3) *#apply k-means algorithm with no. of centroids(k)=3*
- result$size *# gives no. of records in each cluster*
- result$cluster *#gives cluster vector showing the cluster where each record falls*

RCode

- par(mfrow=c(2,2), mar=c(5,4,2,2))
- plot(irisdata[c(1,2)], col=result$cluster)
- *# Plot to see how Sepal.Length and Sepal.Width data points have been distributed in clusters*
- plot(irisdata[c(1,2)], col=irisclass)*# Plot to see how Sepal.Length and Sepal.Width data points have been distributed originally as per "class" attribute in dataset*
- plot(irisdata [c(3,4)], col=result$cluster)*# Plot to see how Petal.Length and Petal.Width data points have been distributed in clusters*
- plot(irisdata[c(3,4)], col=irisclass)

Accuracy

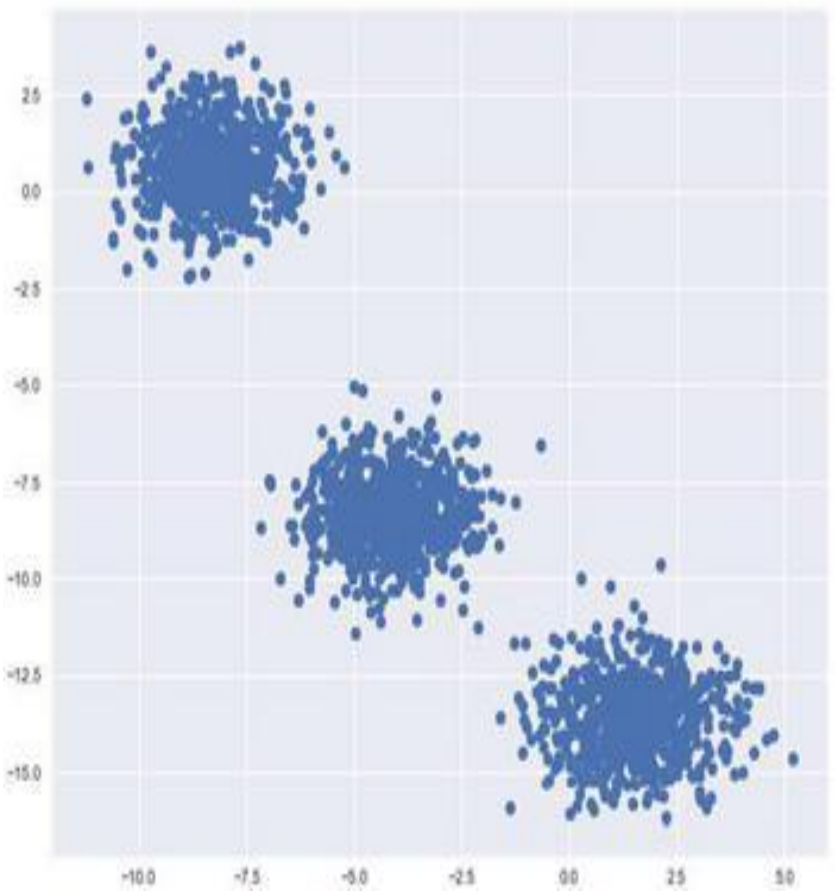
- table(result$cluster,irisclass)
- ## iris.class
- ## srno. setosa versicolor virginica
- ## 1 0 47 14
- ## 2 0 3 36
- ## 3 50 0 0
- Total number of correctly classified instances are: 36 + 47 + 50= 133
- Total number of incorrectly classified instances are: 3 + 14= 17
- Accuracy = 133/(133+17) = 0.88 i.e. our model has achieved 88% accuracy!

ggplot2

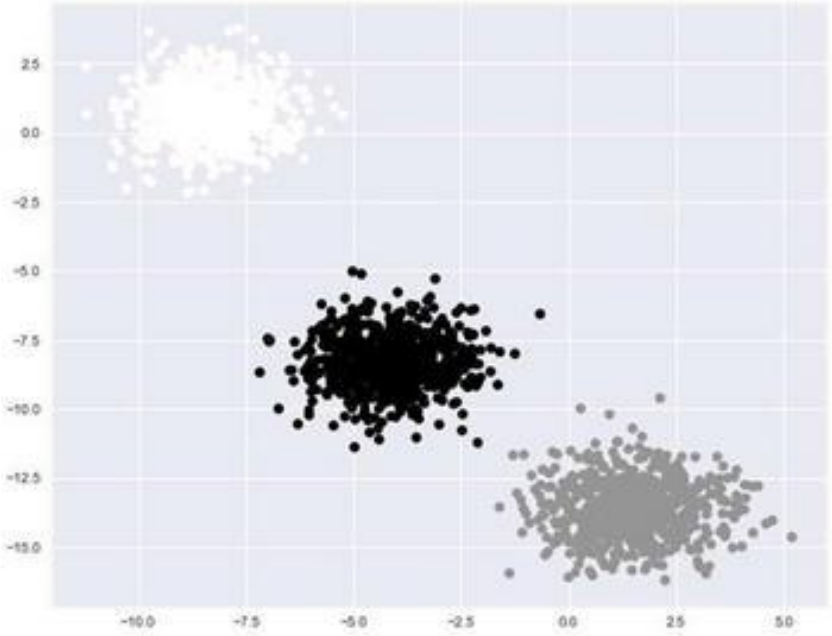
- library(ggplot2)
- ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
- set.seed(20)
- irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
- irisCluster
- table(irisCluster$cluster, iris$Species)
- irisCluster$cluster <- as.factor(irisCluster$cluster)
- ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()

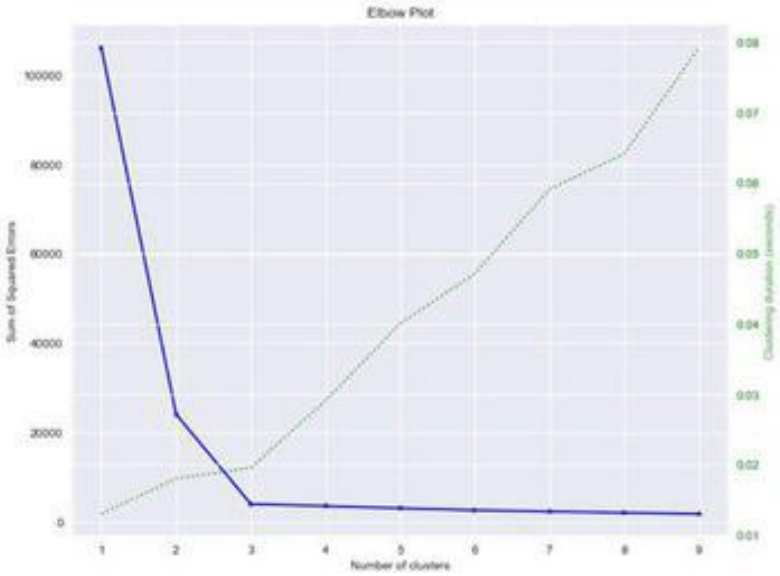
Python

- import pandas
- import seaborn
- import numpy
- import scikitplot
- from sklearn.cluster import KMeans,AgglomerativeClustering, SpectralClustering
- import matplotlib.pyplot as plt
- %matplotlib inline
- from pylab import rcParams
- rcParams['figure.figsize'] = 10, 8
- lustering_data_1 = pandas.read_csv('./data/
- clustering_data_1.csv')
- Check the shape of the data
- clustering_data_1.shape
- Data contains 2000 points with X and Y feature. We can plot this on 2D plane.
- plt.scatter(clustering_data_1['X'],clustering_data_1['Y'])
- From data we can see presence of tree clusters
- #First create a KMeans clustering object and define number
- of cluster = 2
- kmeans = KMeans(n_clusters=2)



- Fit KMeans clustering algorithm
- kmeans_3 = KMeans(n_clusters=3).fit(clustering_data_1)
- #Get cluster assignment
- cluster_assignment = kmeans_3.predict(clustering_data_1)
- #Plot cluster
- plt.scatter(clustering_data_1['X'],clustering_data_1['Y'], c=cluster_assignment)
- kmeans_elbow_check = KMeans()
- scikitplot.cluster.plot_elbow_curve(kmeans_elbow_check,
- X=clustering_data_1, cluster_ranges=range(1, 10))





Thank you

- **Q and As**

References

- https://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf
- Christopher Bishop: ”Pattern Recognition and Machine Learning” , 2006
- Kevin Murphy: ”Machine Learning: a Probabilistic Perspective”
- David Mackay: ”Information Theory, Inference, and Learning Algorithms”
- Ethem Alpaydin: ”Introduction to Machine Learning” , 2nd edition, 2010.
- R. Duda, P. Hart & D. Stork, ***Pattern Classification*** (2nd ed.), Wiley T. Mitchell, ***Machine Learning,*** McGraw-Hill