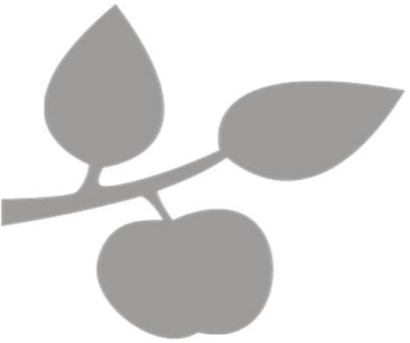


DM583

Data Mining

Clustering Basics and Partitioning Clustering

PART 1

Introduction and Preliminaries on Clustering

Clustering

- Alongside with Regression and Classification, **Clustering** is one of the most widely used and investigated techniques in exploratory data analysis (EDA)
- It is well-established as a subfield of statistics at least since the 1950s
- It is has gained increasing popularity with the rise of *big data*
    - No longer among statisticians only, but also in computer science, and modern data science more broadly

The Clustering Task Itself is Subjective…

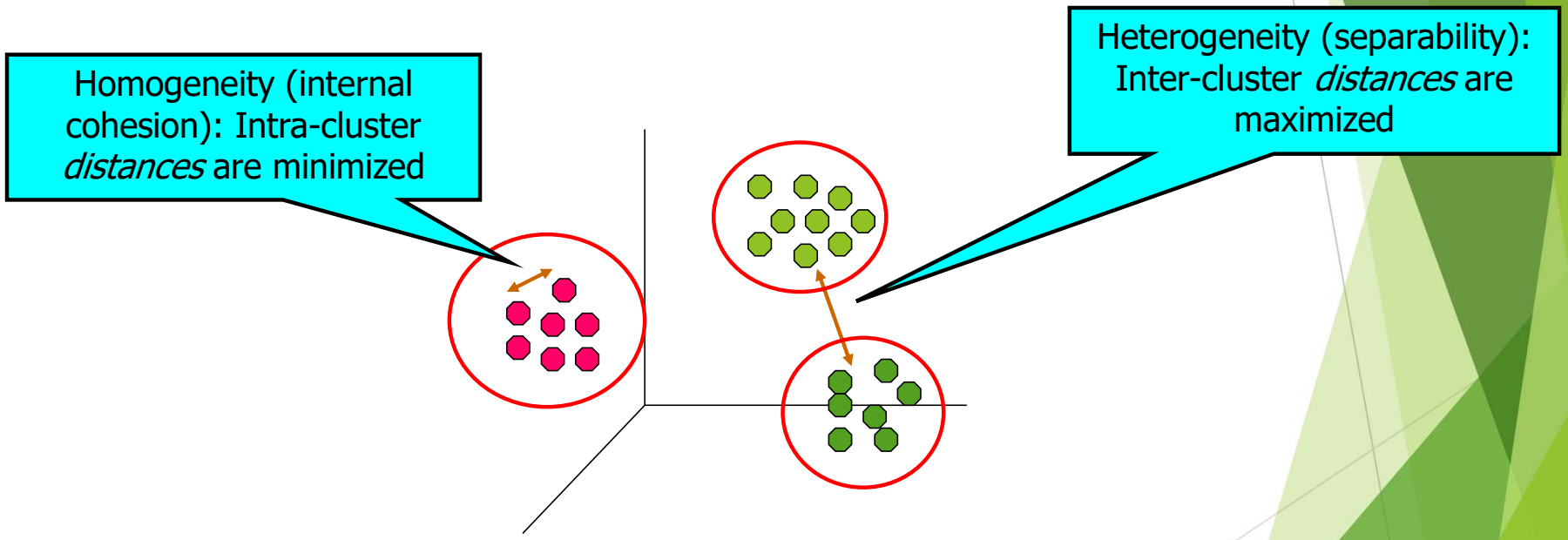
- Clustering is not a well-defined problem, and there are many definitions
- For instance:

"Finding groups of objects such that the objects in a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups" (**Tan et al., 2006**)

"When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other" (**James et al., 2013**)

- The **subjective** nature of "similar, "related", etc. makes the problem far from trivial
  - It gives room to a multitude of different clustering paradigms and algorithms
- To solve a clustering task, one needs to define the problem in an **objective** way

One Possible Mathematical Perspective



Another (related, yet not equivalent) perspective: "A cluster is an aggregation of points in space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it **(Everitt, 1974)**.

Clustering in Practice

- It is a very powerful tool for *data pre-processing* and *exploration* prior to other downstream analyses
  - Initial stages of data analysis, to gain insights about structures in the data, before other techniques are applied
  - It allows for data **organization**, **visualization**, **summarization**, …
- But it can also be the final goal of analysis in many applications
  - From traditional to modern application domains

Clustering in Practice

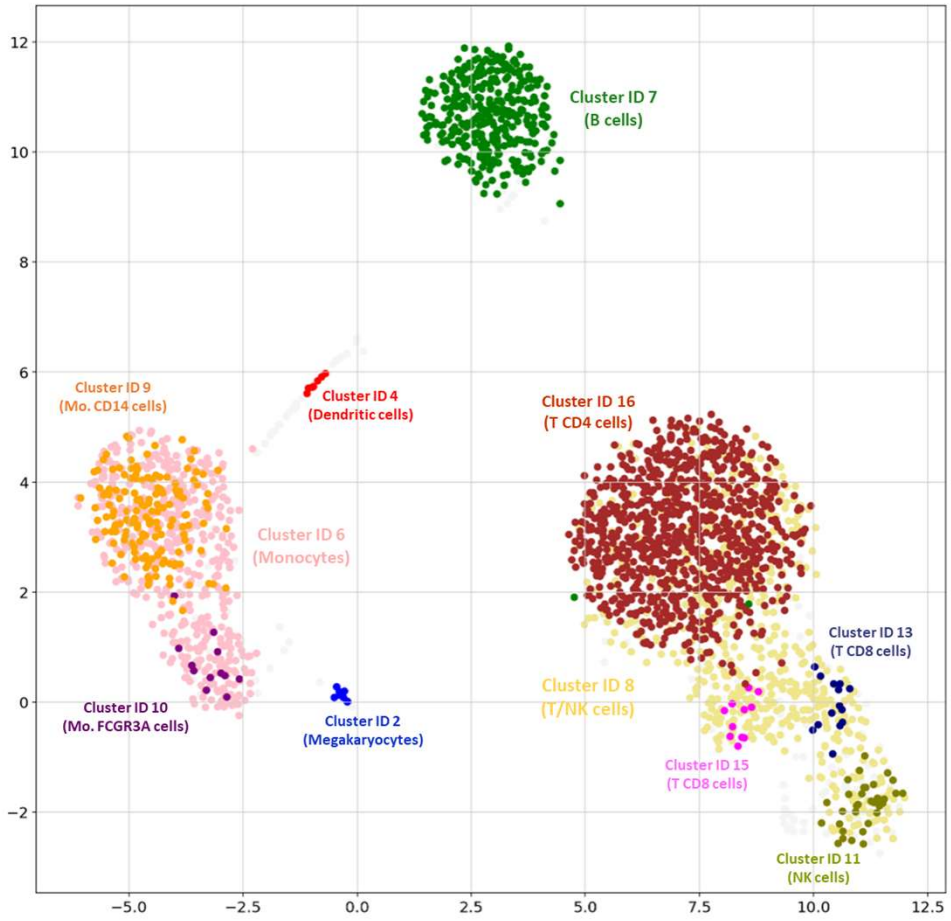
Clustering finds applications in virtually all fields of knowledge:

- Biology, Health, and Medicine
- Psychology
- Business and Finances
- Computing Science
- Engineering
- …

Just a Few Aplication Examples

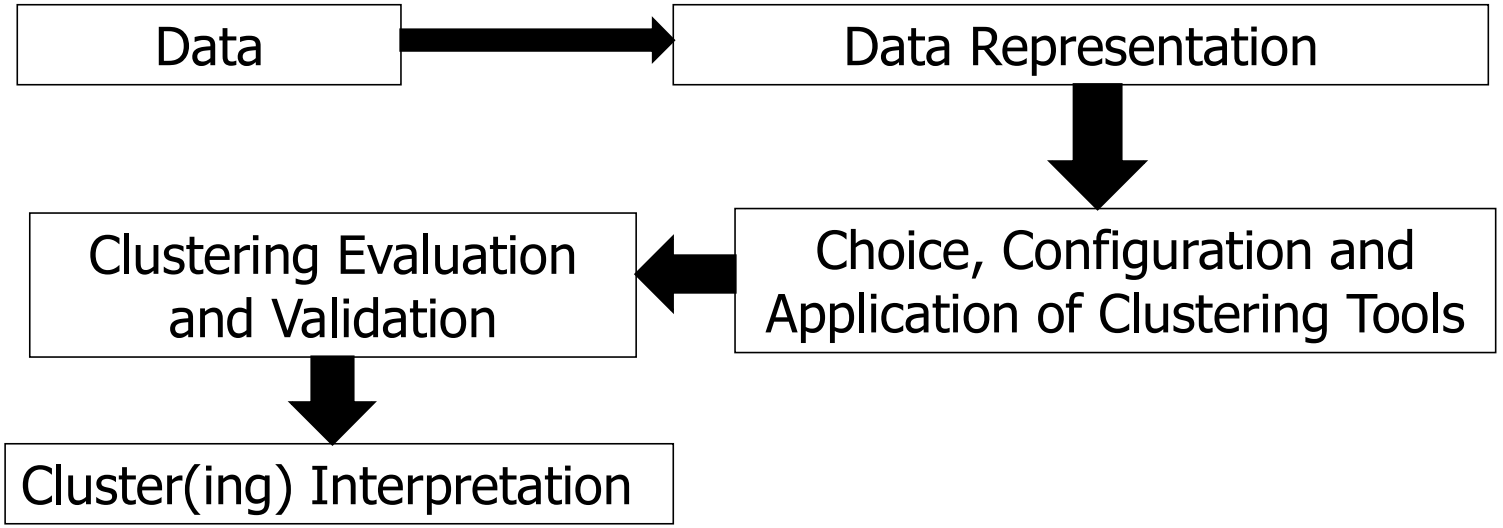
- **Marketing:** groups of customers (market segmentation)
- **Bioinformatics:** groups of genes, tissue types, cell types
- **Astronomy:** groups of stars or galaxies
- **Image Processing:** groups of images or pixels (image segmentation)
- **Text Mining:** automatic document categorization
- …

Clustering-Based Cell Type/Sub-Type Discovery (Bioinformatics):



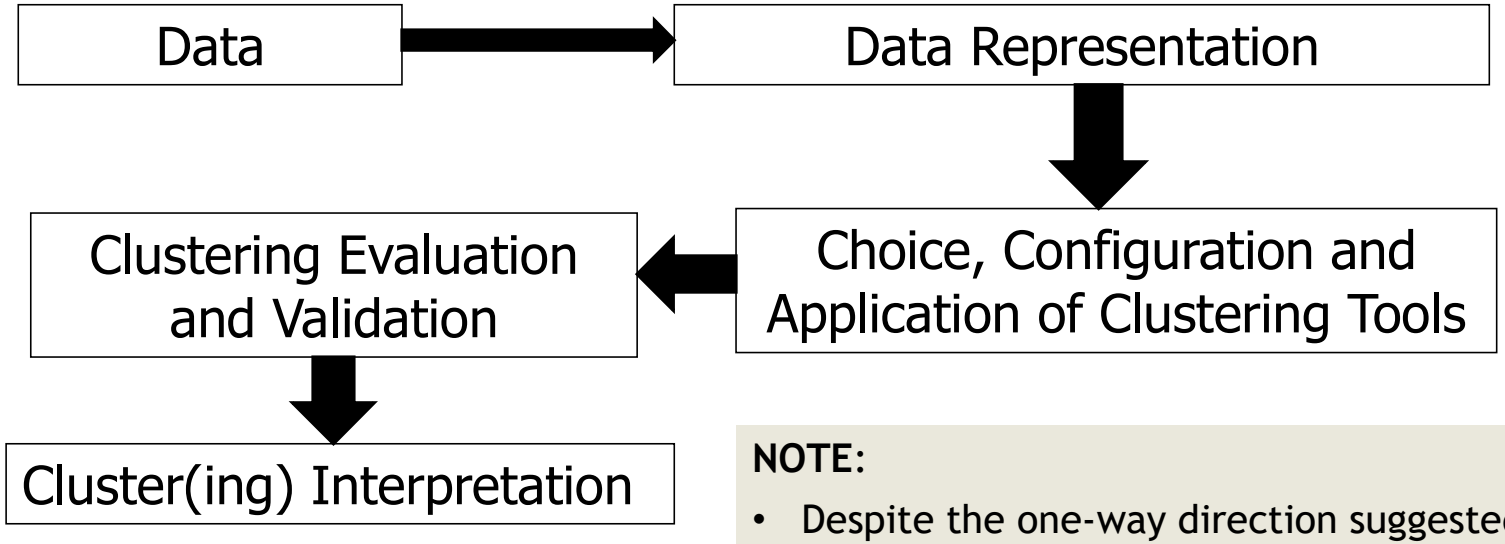
Ricardo Campello

“Cluster Analysis”



Meaningful and Useful Findings?
(Domain Expert)

"Cluster Analysis"



Meaningful and Useful Findings?
(Domain Expert)

NOTE:

- Despite the one-way direction suggested by the arrows, the process usually involves a cycle (EDA!)
- In fact, bear in mind that clustering is *subjective;* any attempt to make it **objective** is explicitly or implicitly based on **assumptions**, which may need to be revisited
- E.g., each clustering algorithm makes the clustering problem objective based on their own assumptions

References (Part 1)

- Jain, A. K. and Dubes, R. C., *Algorithms for Clustering Data*, Prentice Hall, 1988
- Everitt, B. S., *Cluster Analysis*, John Wiley and Sons, 1974
- Everitt, B. S., Landau, S., and Leese, M., *Cluster Analysis*, Arnold, 4th Edition, 2001
- Tan, P.-N., Steinbach, M., and Kumar, V., *Introduction to Data Mining*, Addison-Wesley, 2006
- P.-N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, 2nd Edition, Pearson, 2018.
- G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.
- Wu, X. and Kumar, V. (Editors), *The Top Ten Algorithms in Data Mining*, CRC Press, 2009

PART 2

Partitioning Clustering and k-Means

Data Partition

- Given a data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ consisting of N observations, a (non-overlapping, so-called hard) clustering **partition** of the data is a collection $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k\}$ of k disjoint subsets, called **clusters**, such that:

$$\mathbf{C}_1 \cup \mathbf{C}_2 \cup \ldots \cup \mathbf{C}_k = \mathbf{X}$$

$$\mathbf{C}_i \neq \varnothing \quad \forall\, i$$

$$\mathbf{C}_i \cap \mathbf{C}_j = \varnothing \quad \forall\, i \neq j$$

  - For instance: $\mathbf{P} = \{ (\mathbf{x}_1), (\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6), (\mathbf{x}_2, \mathbf{x}_5) \}$

- One possible way to represent a clustering partition is a **partition matrix**

Partition Matrix

- It is a matrix with k rows (no. of clusters) and N columns (no. of observations) in which entry μ_{ij} stands for the membership degree of the jth observation ($\mathbf{x}_j$) to the ith cluster ($\mathbf{C}_i$)
- In case of hard (non-overlapping) partitioning algorithms, each observation must belong to exactly one cluster, i.e:

$$\mu_{ij} \in \{0,1\}$$
$$\sum_i \mu_{ij} = 1 \quad \forall j$$

observations

$$\mathbf{U}(\mathbf{X}) = \begin{bmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1N} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2N} \\ \vdots & & \ddots & \vdots \\ \mu_{k1} & \mu_{k2} & \cdots & \mu_{kN} \end{bmatrix}$$

clusters

Partition Matrix

- Example:
  - $\mathbf{P} = \{ (\mathbf{x}_1), (\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6), (\mathbf{x}_2, \mathbf{x}_5) \}$

$$\mathbf{U}(\mathbf{X}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Non-Overlapping Partitioning Clustering

- *Hard* or *Non-overlapping Partitioning* clustering refers to methods that (explicitly or implicitly) seek a hard partition of a dataset **X** as collection of objects/observations to be clustered

Finding a Hard Partition Matrix U(X) is equivalent to subdividing the set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ with N objects into a collection $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k\}$ of k disjoint clusters $\mathbf{C}_i$ such that $\mathbf{C}_1 \cup \mathbf{C}_2 \cup ... \cup \mathbf{C}_k = \mathbf{X}$, $\mathbf{C}_i \neq \emptyset$, and $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset \ \ \forall \ \ i \neq j$

K-Means

- K-Means is one of the oldest and most well-known clustering algorithms
- It is one of many algorithms that aim to **minimize the within-cluster distances while maximizing the between-cluster distances**
  - In the case of k-means, the latter goal follows naturally from the former (for fixed k)
- The classic version assumes quantitative observations and measures distance between observations using the **squared Euclidean distance**
- It represents clusters using their multivariate mean, or **centroid** (spatial center)
  - It is one of many algorithms that belong to the **prototype-based clustering paradigm**
- It finds a (non-overlapping) **partition** of the data into disjoint subsets (clusters)

k-Means

- **Reference Mostly Cited as the Original One**:

J. B. MacQueen, *Some methods of classification and analysis of multivariate observations*, In Proceedings 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, USA, 1967, 281–297

- **However:**

"K-means has a rich and diverse history as it was independently discovered in different scientific fields by Steinhaus (1956), Lloyd (proposed in 1957, published in 1982), Ball & Hall (1965) and MacQueen (1967)" [**Jain, *Data Clustering: 50 Years Beyond K-Means*, Patt. Rec. Lett., 2010**]

- **See also:**

Douglas Steinley, *K-Means Clustering: A Half-Century Synthesis*, British Journal of Mathematical and Statistical Psychology, Vol. 59, 2006

K-Means

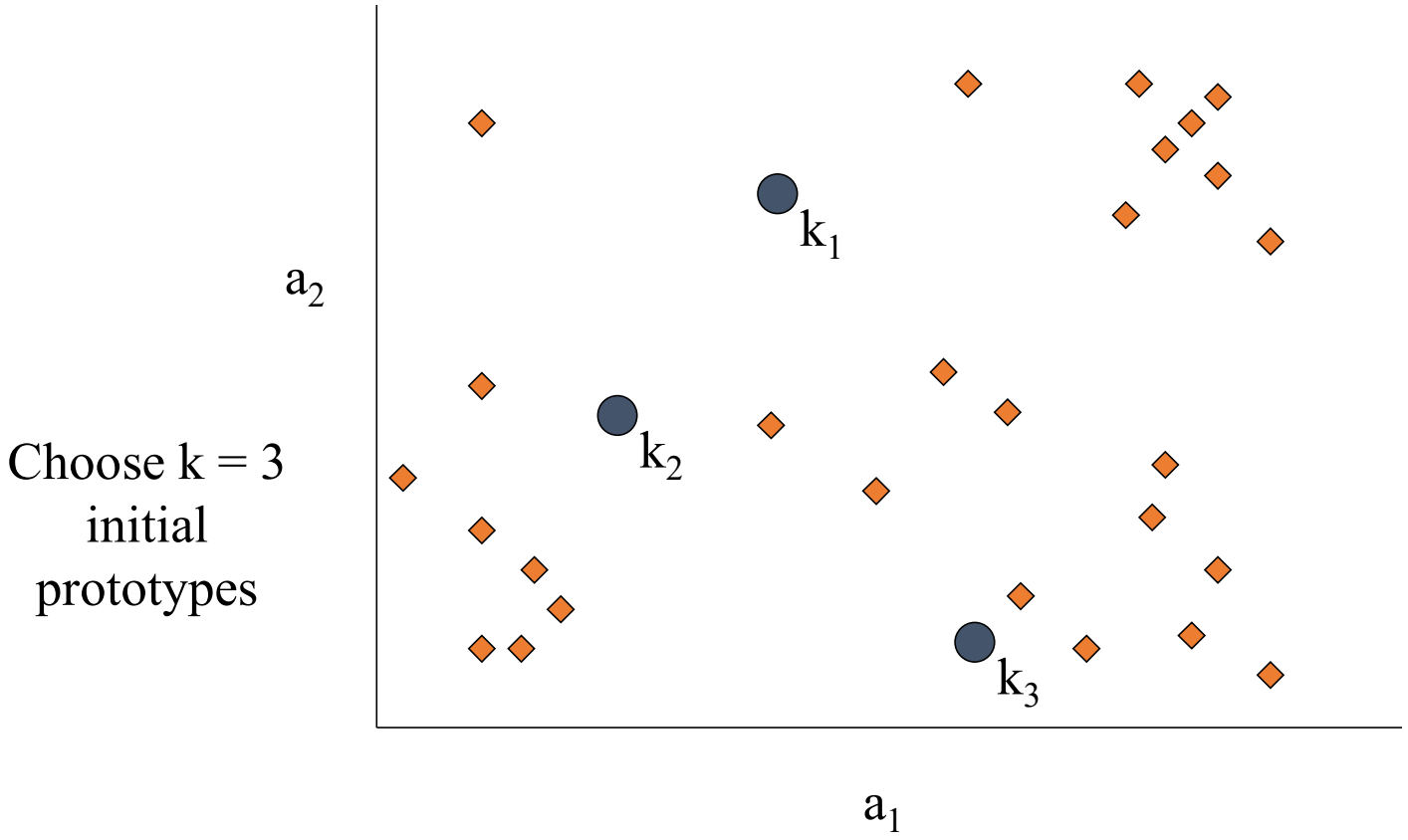
- In spite of having been developed in the 1950*/60s, this algorithm is still widely used in practice, mainly because:
  - It is very simple and intuitive
  - It is computationally fast and can be further optimized in various ways
  - It serves as the basis for other more sophisticated algorithms and analysis tools

Basic K-Means Algorithm

1. Choose k initial cluster prototypes (k observations or random points in the feature space)
2. Assign each observation to the closest prototype (squared Euclidean distance)
3. Update cluster prototypes as the centroids of the corresponding clusters
4. Iterate steps 2 and 3 until no more changes occur

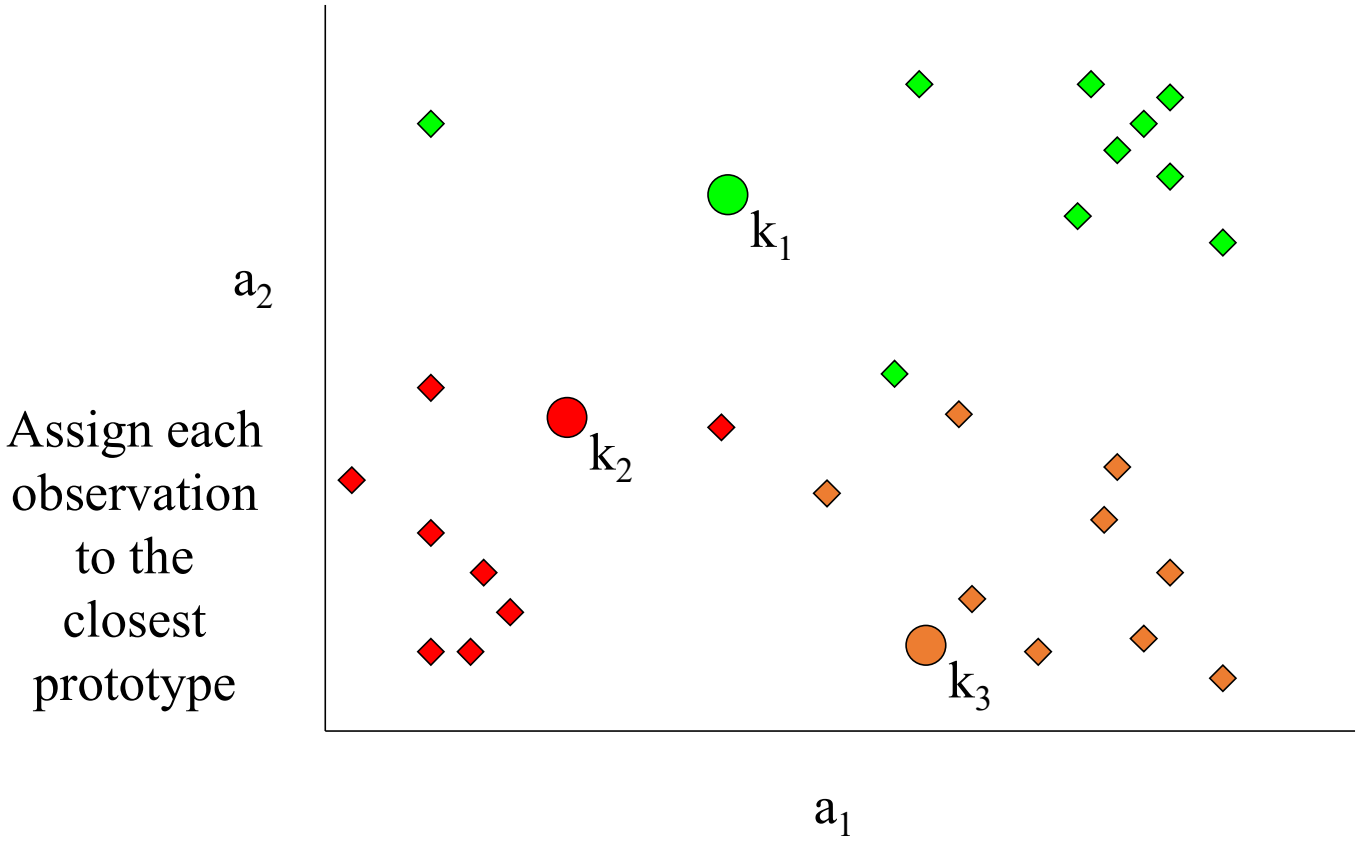
   or until an early stopping criterion is met (e.g. maximum number of iterations)

k-Means: Step 1



Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

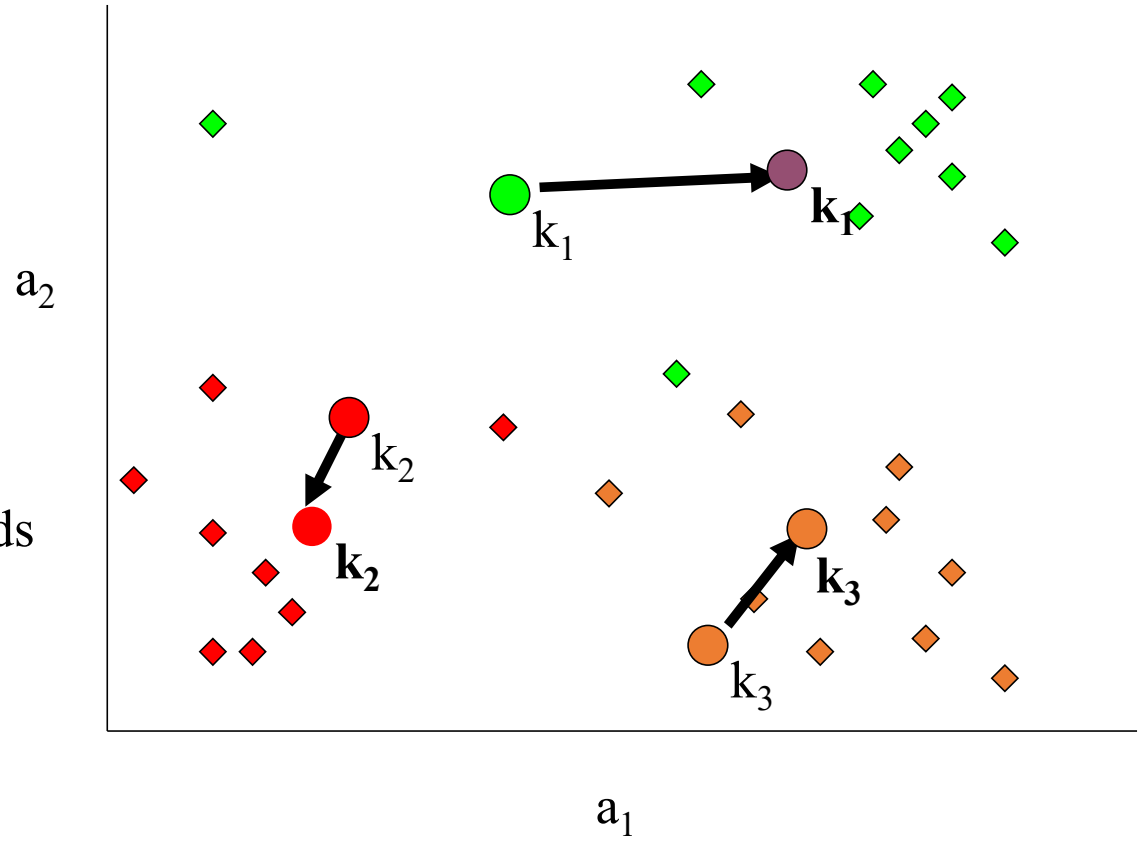
k-Means: Step 2



Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

k-Means: Step 3

Update prototypes as cluster centroids

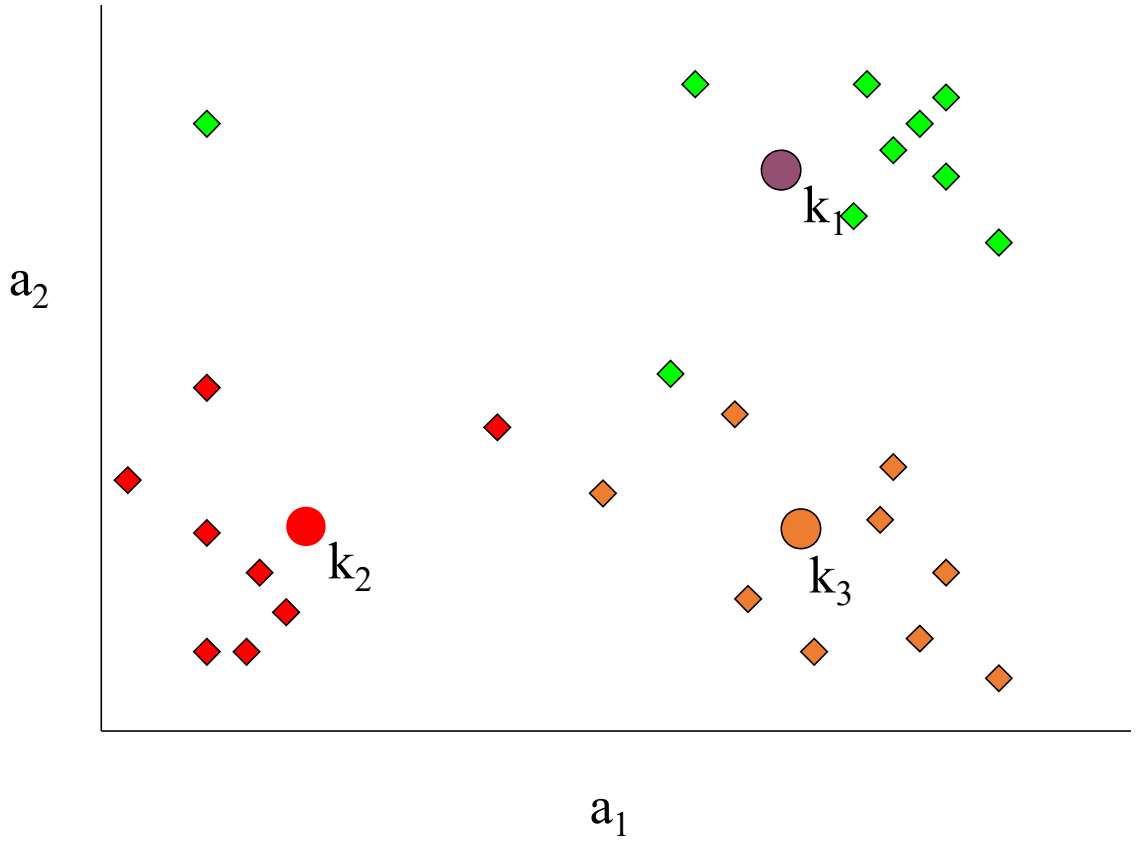


Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

k-Means: Step 2...

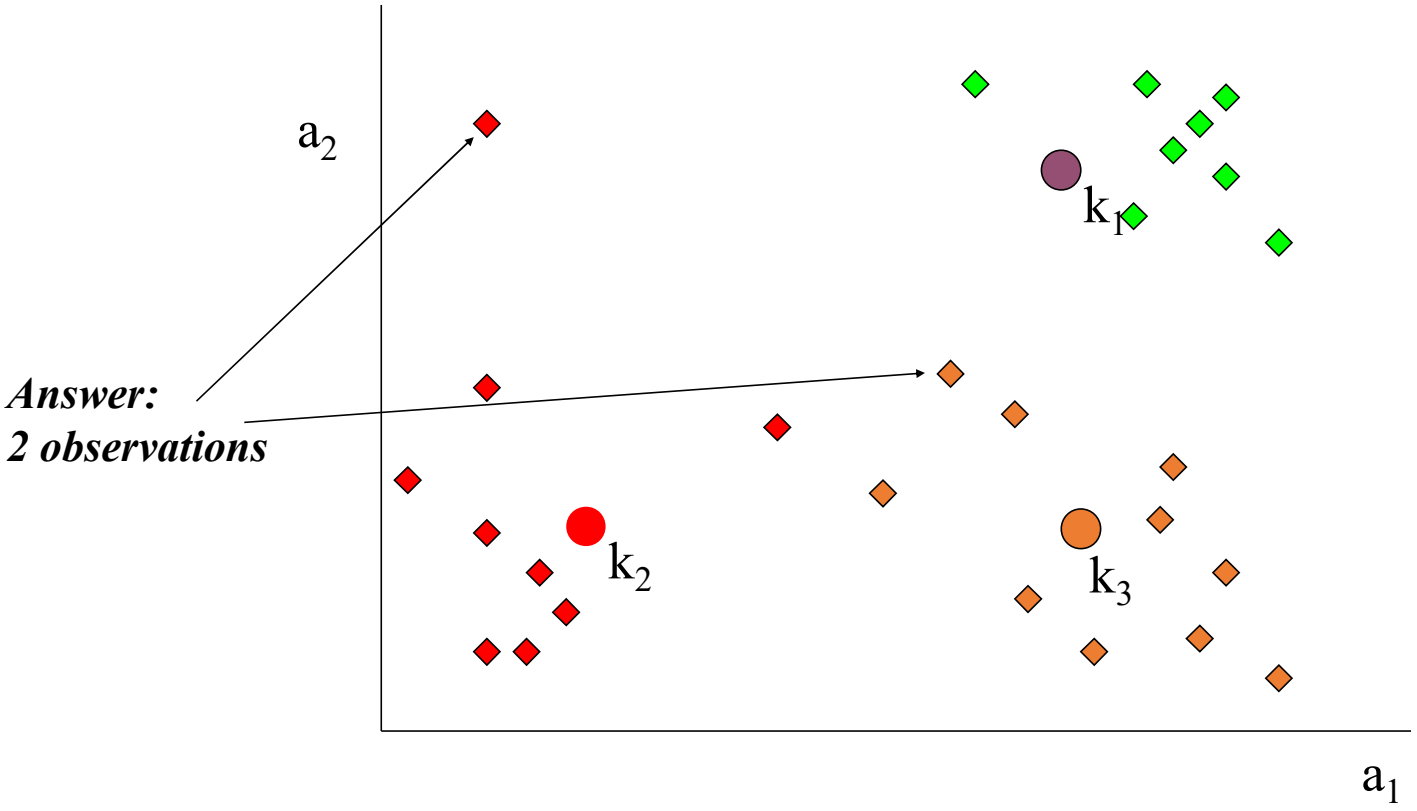
Assign each observation to the closest prototype...

which observations will change clusters...?



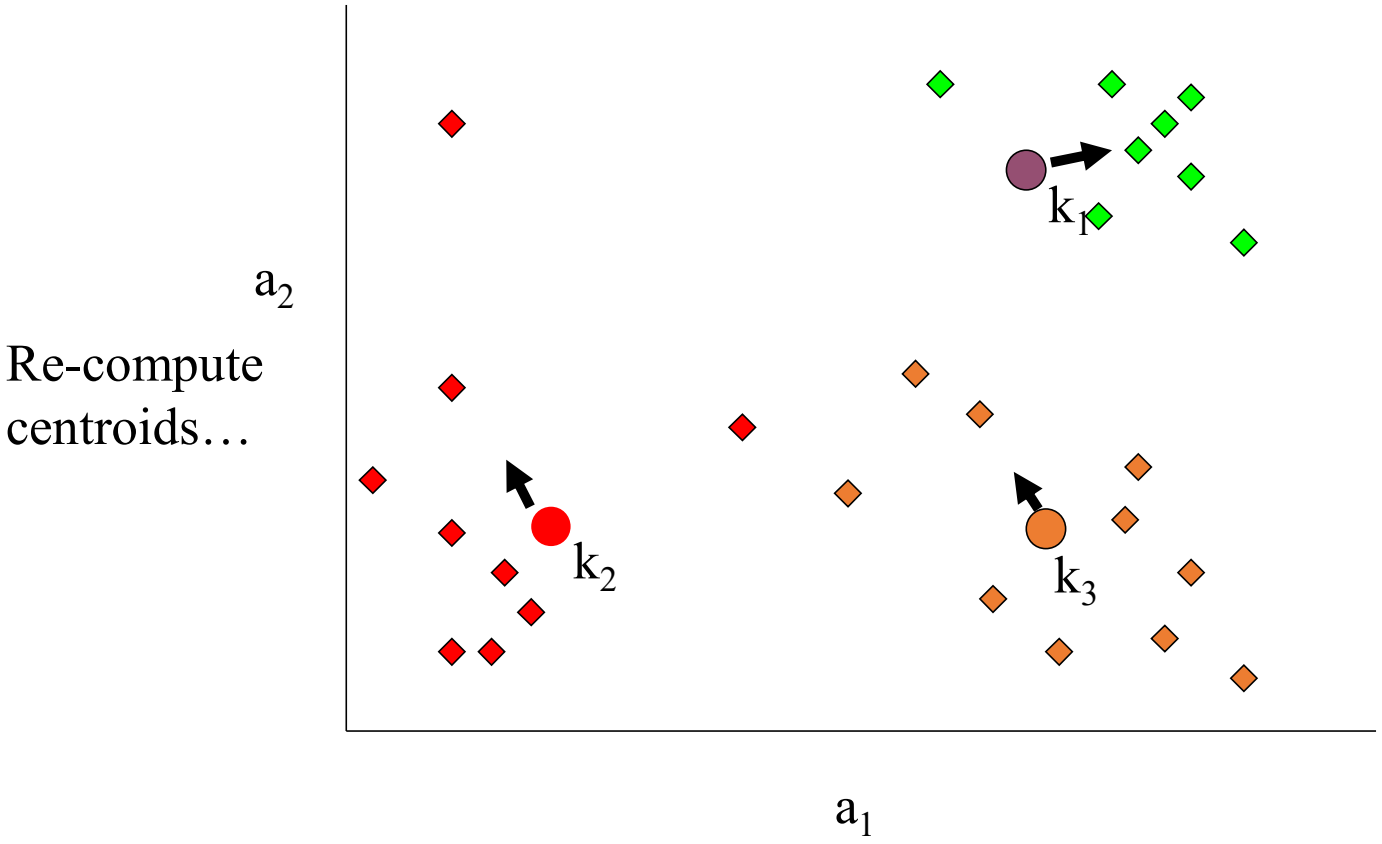
Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

k-Means: Step 2



Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

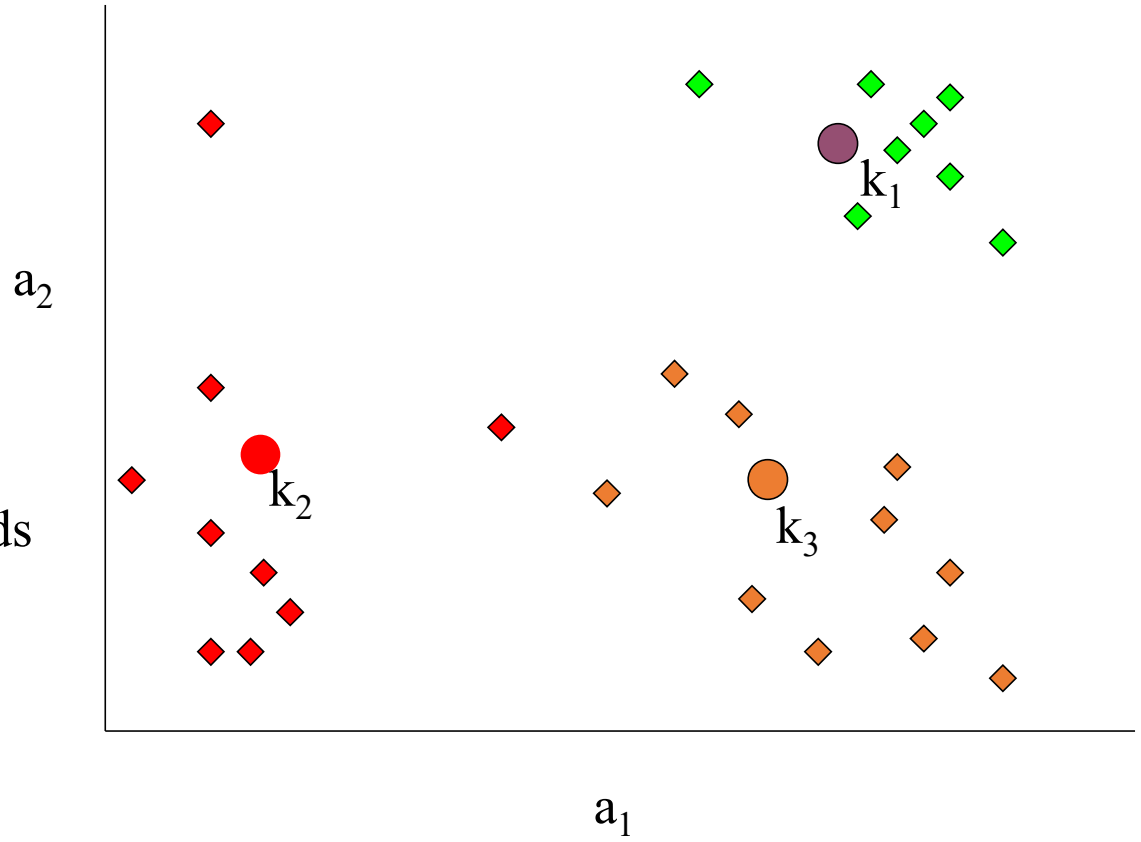
k-Means: Step 3...



Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

k-Means: Step 3

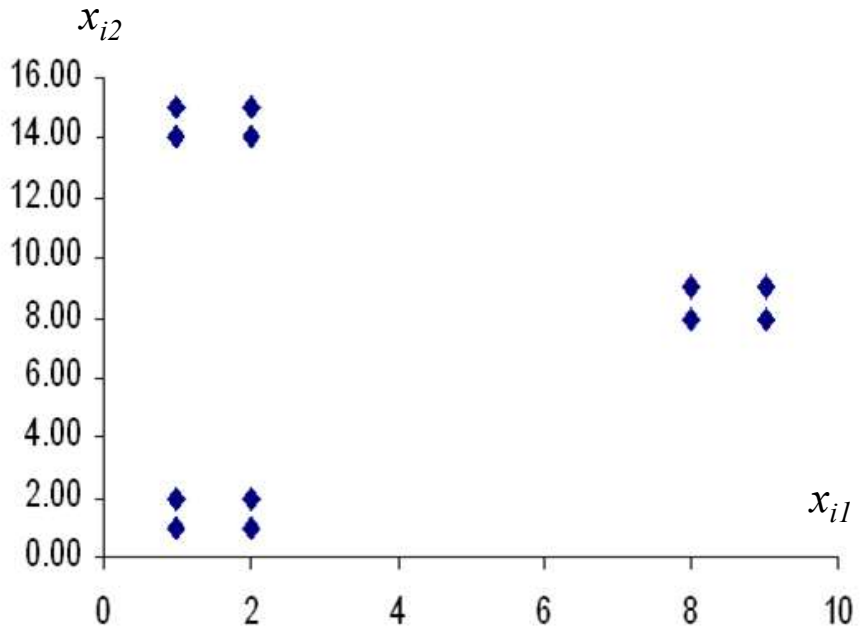
Update prototypes as cluster centroids



Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

Exercise

Observation $\mathbf{x}_i$	x_{i1}	x_{i2}
1	1	2
2	2	1
3	1	1
4	2	2
5	8	9
6	9	8
7	9	9
8	8	8
9	1	15
10	2	15
11	1	14
12	2	14



- Perform k-means in the above dataset starting from prototypes [6 6], [4 6] and [5 10]

K-Means as an Optimization Problem

- Objective function to be optimized (minimized):
  - **SSE** = *Sum of Squared Errors* (within-cluster variances)

$$J = \sum_{c=1}^{k} \sum_{\mathbf{x}_j \in \mathbf{C}_c} d\left(\mathbf{x}_j, \bar{\mathbf{x}}_c\right)^2 \propto \sum_{c=1}^{k} \frac{1}{|\mathbf{C}_c|} \sum_{\mathbf{x}_j, \mathbf{x}_i \in \mathbf{C}_c} d\left(\mathbf{x}_j, \mathbf{x}_i\right)^2$$

where d = Euclidean distance and $\bar{\mathbf{x}}_c$ is the prototype of the cth group, $\boldsymbol{C}_c$ (with cardinality $|\boldsymbol{C}_c|$)

K-Means as an Optimization Problem

- The SSE can be rewritten using a partition matrix as:

$$J = \sum_{j=1}^{N}\sum_{c=1}^{k} \mu_{cj} \left\| \mathbf{x}_j - \bar{\mathbf{x}}_c \right\|^2 \; ; \; \sum_{c=1}^{k} \mu_{cj} = 1 \;\; \forall j \;\; ; \;\; \mu_{cj} \in \{0,1\}$$

- We want to minimise J with respect to $\{\bar{\mathbf{x}}_c\}$ and $\{\mu_{cj}\}$
- We can do this iteratively (alternating 2 steps, **E** and **M**):
  a) Fix $\{\bar{\mathbf{x}}_c\}$ and minimise J with respect to $\{\mu_{cj}\}$ (**E**)
  b) Fix $\{\mu_{cj}\}$ and minimise J w.r.t. $\{\bar{\mathbf{x}}_c\}$ (**M**)

K-Means as an Optimization Problem

$$J = \sum_{j=1}^{N} \sum_{c=1}^{k} \mu_{cj} \left\| \mathbf{x}_j - \bar{\mathbf{x}}_c \right\|^2 \; ; \; \sum_{c=1}^{k} \mu_{cj} = 1 \;\; \forall j \; ; \; \mu_{cj} \in \{0,1\}$$

a) Fix $\{\bar{\mathbf{x}}_c\}$ and minimise J w.r.t. $\{\mu_{cj}\}$ (**Step E**)

- Summation components in j are independent…
- As such, we can minimise them independently (i.e. separately for each observation)
- $\mu_{cj}=1$ for c that provides the smallest squared error $\|\mathbf{x}_j - \bar{\mathbf{x}}_c\|^2$, $\mu_{cj}=0$ otherwise

*** Assign $\mu_{cj}=1$ for the cluster c closest to jth observation**

K-Means as an Optimization Problem

$$J = \sum_{j=1}^{N}\sum_{c=1}^{k} \mu_{cj} \left\| \mathbf{x}_j - \bar{\mathbf{x}}_c \right\|^2 \; ; \; \sum_{c=1}^{k} \mu_{cj} = 1 \;\; \forall j \;\; ; \;\; \mu_{cj} \in \{0,1\}$$

b) Minimise J w.r.t. $\{\bar{\mathbf{x}}_c\}$, fixing $\{\mu_{cj}\}$ **(Step M)**

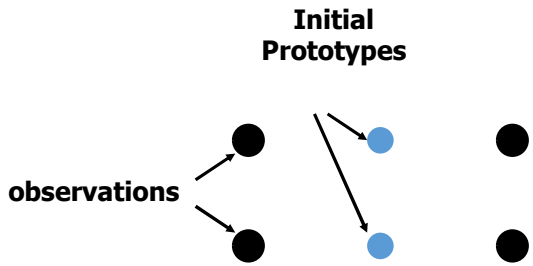
- Make the gradient of J w.r.t. each $\bar{\mathbf{x}}_c$ equal to zero and solve:

$$\nabla_{\bar{\mathbf{x}}_c} J = \sum_{j=1}^{N} \mu_{cj} \nabla_{\bar{\mathbf{x}}_c} \left[\left(\mathbf{x}_j - \bar{\mathbf{x}}_c \right)^T \left(\mathbf{x}_j - \bar{\mathbf{x}}_c \right) \right] = 2 \sum_{j=1}^{N} \mu_{cj} \left(\bar{\mathbf{x}}_c - \mathbf{x}_j \right) = \mathbf{0} \;\; \rightarrow \;\; \bar{\mathbf{x}}_c = \frac{\sum_{j=1}^{N} \mu_{cj} \mathbf{x}_j}{\sum_{j=1}^{N} \mu_{cj}}$$

$$\bar{\mathbf{x}}_c = \frac{1}{\left| \mathbf{C}_c \right|} \sum_{\mathbf{x}_j \in \mathbf{C}_c} \mathbf{x}_j$$

K-Means as an Optimization Problem

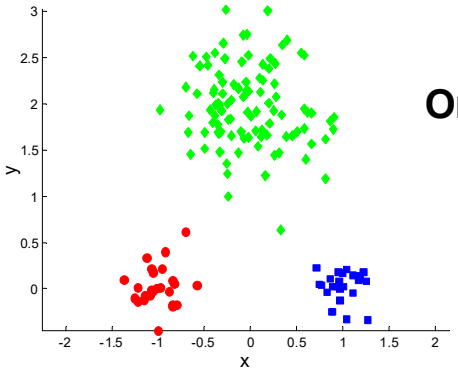
- K-Means is guaranteed to produce local (rather than global) optimal solutions only
  - Proof by counter-example:



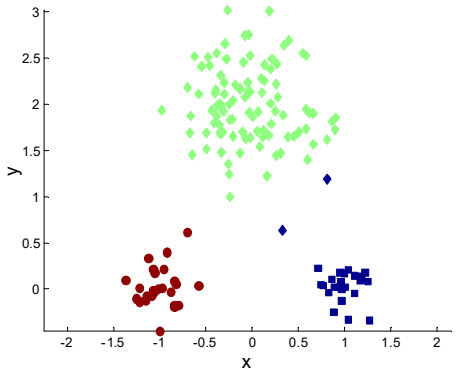
- Final solution depends on the initial prototypes

Based on the original from Gregory Piatetsky-Shapiro at http://www.kdnuggets.com

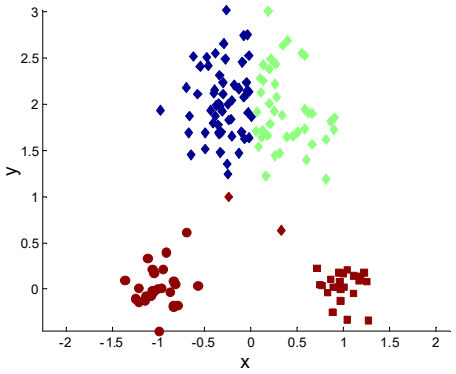
Two different K-means Clusterings



Original Points

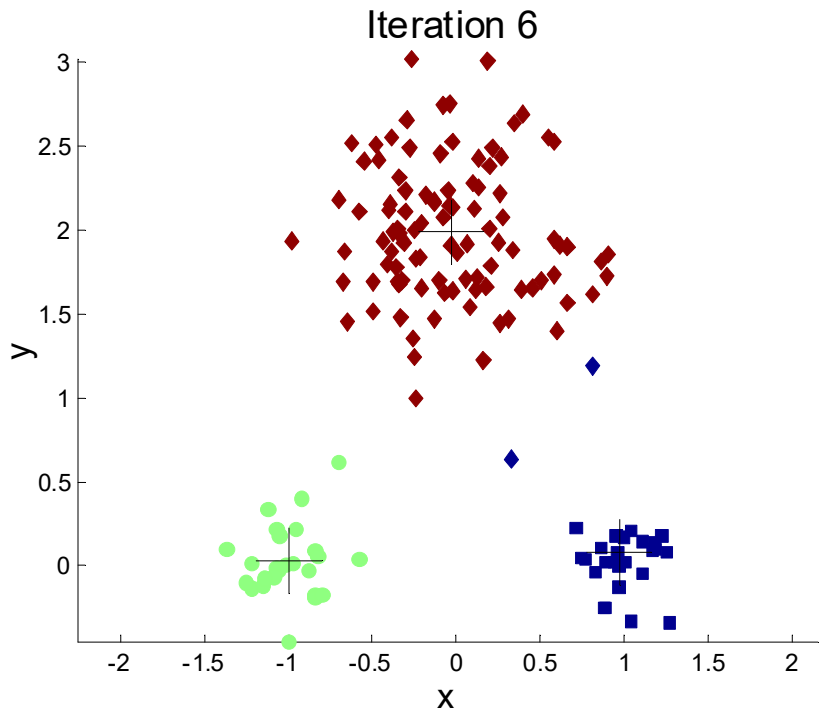


Optimal Clustering

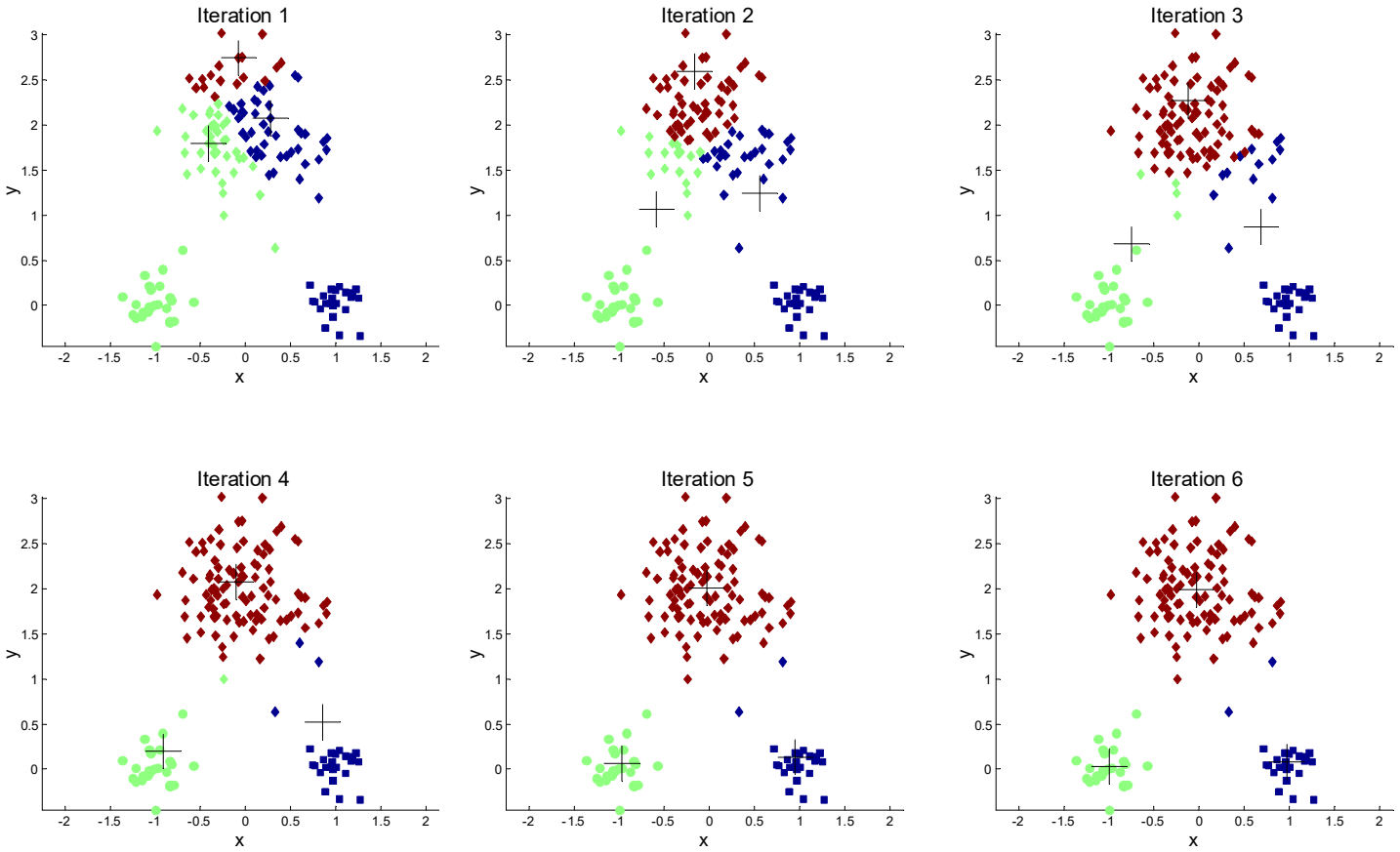


Sub-optimal Clustering

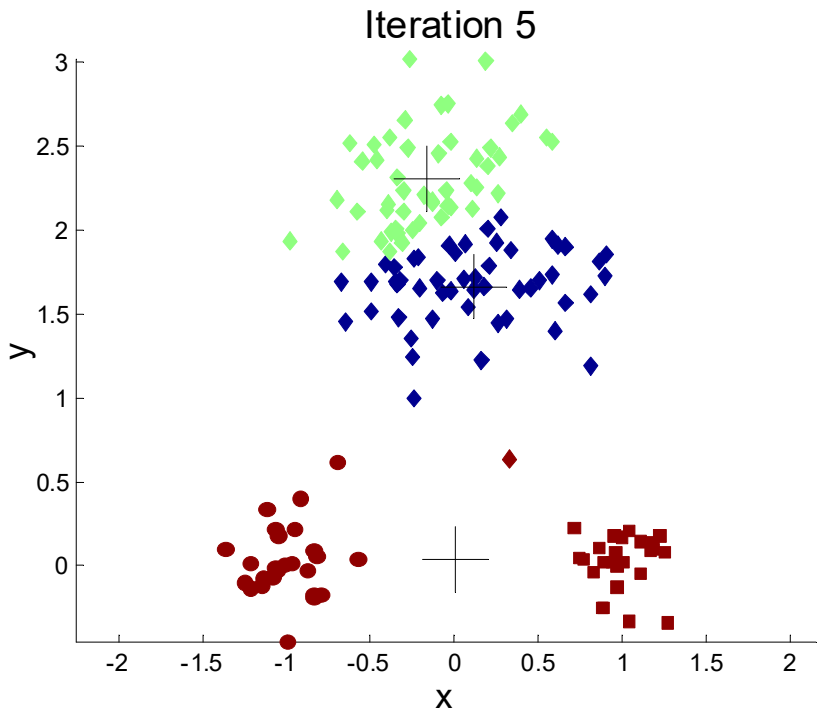
Importance of Choosing Initial Centroids



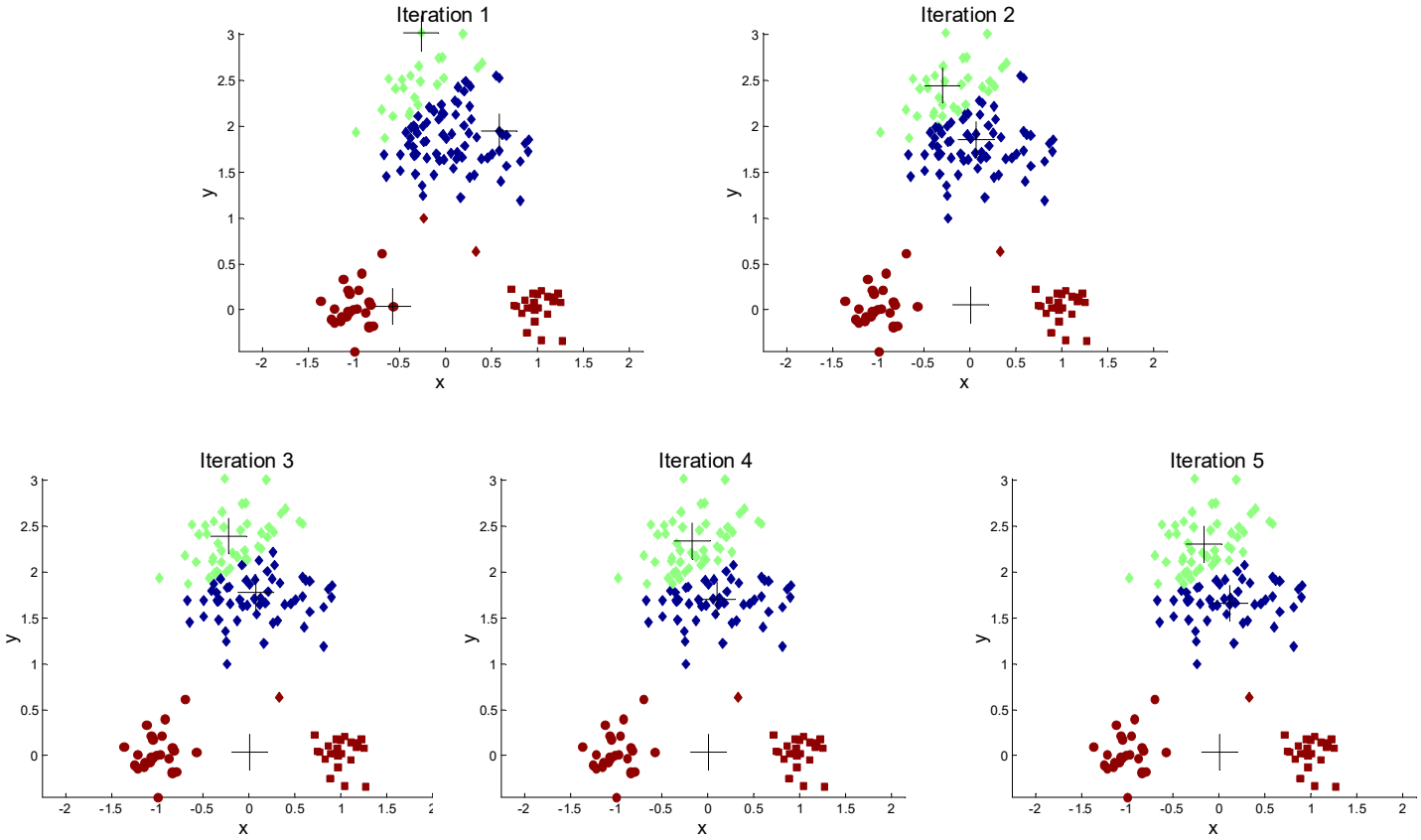
Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids ...



Importance of Choosing Initial Centroids ...



Initialization Strategies

- Multiple Random Initializations of Cluster Prototypes/Centres:
  - Best k-Means result according to a given criterion (e.g. *SSE* for fixed *k*) is selected
  - It works well in many problems
  - In certain (more complex) problems and datasets, however, it may require a large (unknown in advance) number of runs to be effective
    - Especially for large *k*
- Improved Versions of Random Initializations:
  - E.g. **Kmeans++**: "the 1st center is chosen uniformly at random … after which each subsequent center is chosen from the remaining data with probability proportional to its squared distance from the point's closest existing center" (KMeans++ Wikipedia Entry)

Efficient Implementations for Big Data

- Computational performance can be significantly improved...
  - **Specialized Data Structures**, e.g.
    - **kd-trees**
  - **Smart Algorithms**, e.g.
    - **Use of the Triangle Inequality**
    - **Recursive Update of Centroids**
      - New centroids depend only on their previous values as well as on the objects that changed clusters, in addition to the size (cardinality) of each cluster
        - They don't need to be recomputed over and over again from scratch at each iteration!
      - **Exercise:** starting from the equation for centroid computation, rewrite the equation so it can be used to recursively update the centroids
    - **Parallelization** (next)

Parellel / Distributed K-Means

- Data are distributed across multiple *data sites* or *processors*
- Possibly in architectures that are NOT memory-shared
- **Algorithm**:
  - Same initial prototypes are distributed to every data site
  - Each site runs (in parallel) one iteration of k-means on its share of data
  - Local prototypes and local cluster sizes are communicated to a central node
  - Global prototypes are computed (exactly) and retransmitted to all data sites
    - How are they computed? Show it as an **exercise**!
  - This process is repeated for multiple k-Means iterations

Exercise

Object $\mathbf{x}_i$	x_{i1}	x_{i2}	Processor/Site
1	1	2	A
2	2	1	B
3	1	1	A
4	2	2	B
5	8	9	A
6	9	8	B
7	9	9	B
8	8	8	A
9	1	15	B
10	2	15	A
11	1	14	B
12	2	14	A

Perform parallel k-Means in this dataset, with k=3 and initial prototypes [6 6], [4 6] and [5 10]

k-Means Summary

Pros

- Simple and Intuitive
- **Linear** computational complexity: O(N·n·k)
- Effective in many application scenarios, producing results that are simpler to interpret
- Its very well-known and widely used in practice

Cons

- k = ?
- Sensitive to prototype initialization (i.e. local, suboptimal solutions)
- Volumetric / globular clusters only
- Real-valued variables only
- Mean-based: sensitive to *outliers*

Hyper-Parameter Selection in k-Means

- **Multiple Runs within a Range of *k*:**
  - Technically, it is a procedure rather than a variant
  - It runs k-means repeatedly for different values of $k \in [k_{min}, k_{max}]$ and random initial prototypes
  - The best result (partition) according to *some goodness of fit criterion* is taken
  - **Pros:** It can estimate *k* while being less sensitive to local minima
  - **Cons:** Increased computational cost

Question…

- Can't the algorithm's objective function itself (SSE) be used as goodness of fit criterion to choose the best k-means solution among a collection of candidates?

Question...

- Can't the algorithm's objective function itself (SSE) be used as goodness of fit criterion to choose the best k-means solution among a collection of candidates?
  - Yes, IF they all have the same number of clusters (i.e., *k* is fixed)
  - But what if the number of clusters is unknown, and our collection of candidates has varied *k* instead ?

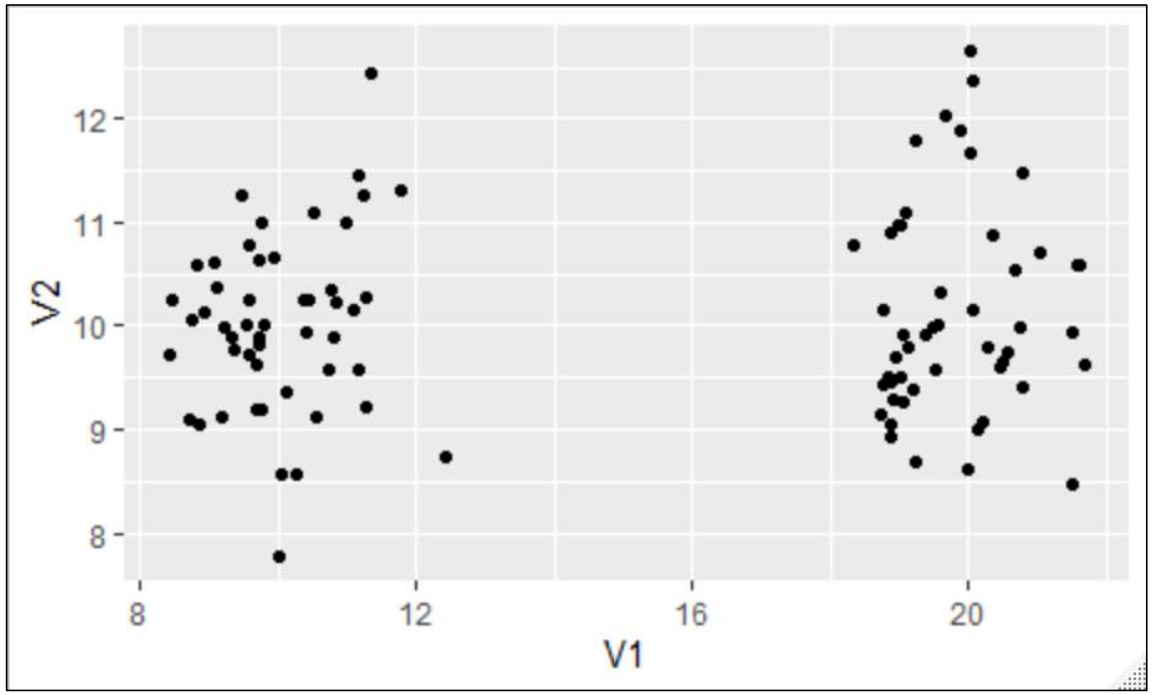
How to Determine k ?

- For a **fixed *k***, we can choose the best solution, among those obtained by running the algorithm several times (from random prototypes), as the one that results in the **minimum value of *J*** (min. SSE)

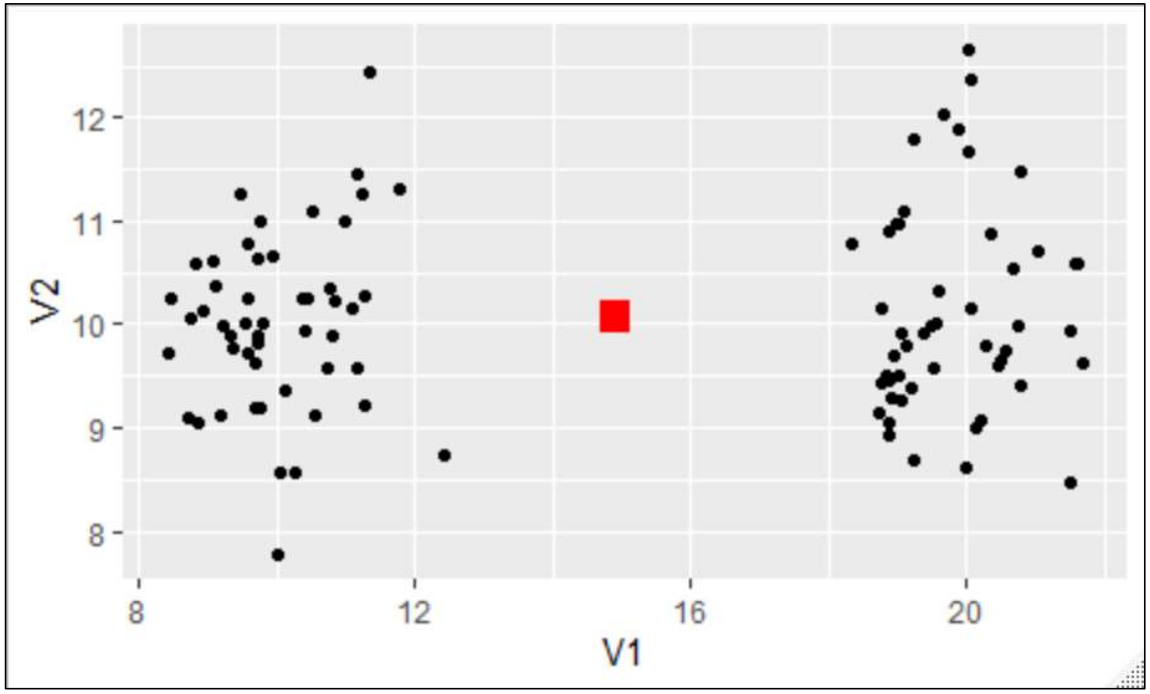
- However, for **variable *k***, the SSE tends to decrease monotonically as k increases…

$$\text{SSE:}\quad J = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in \mathbf{C}_i} d\left(\mathbf{x}_j, \overline{\mathbf{x}}_i\right)^2$$

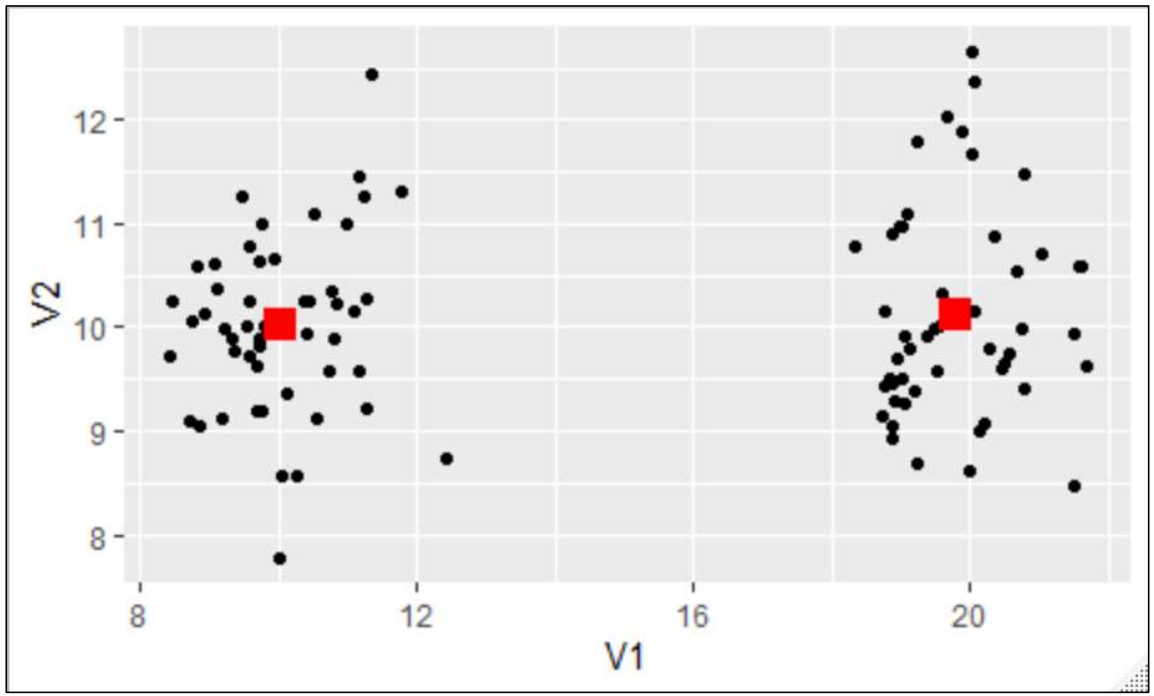
How to Determine k ?



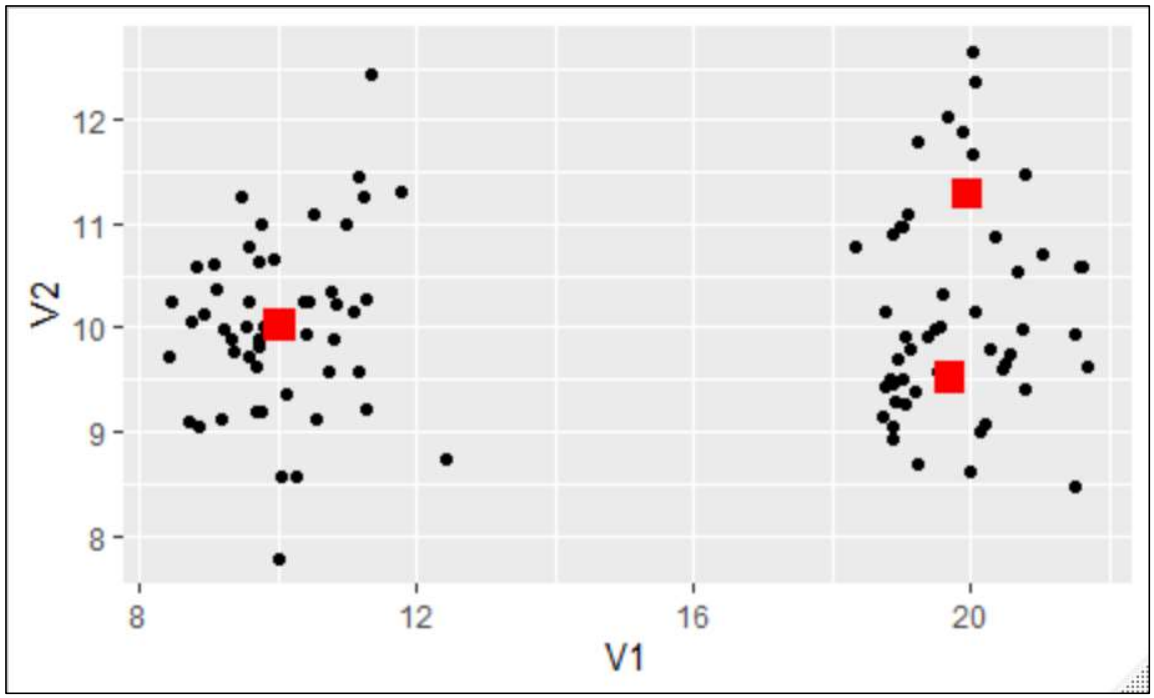
$k = 1 \Rightarrow J = 2550.9$



$k = 2 \Rightarrow J = 166.7$

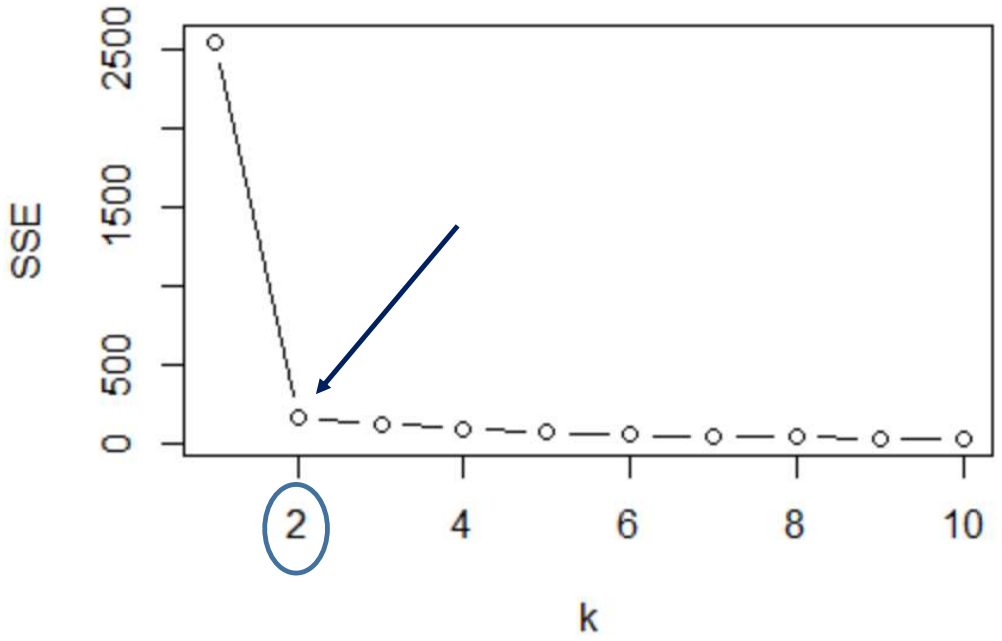


$k = 3 \Rightarrow J = 130.7$



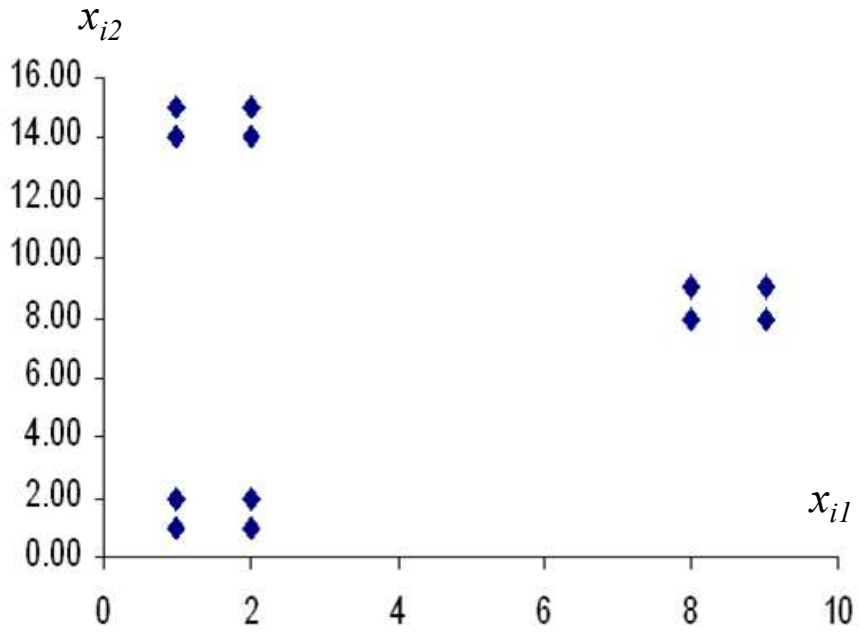
How to Determine k ?

We can plot the best value of *J* (SSE) for each *k* (from multiple runs with random prototypes), varying *k*, and seeking a prominent "elbow“:



Exercise

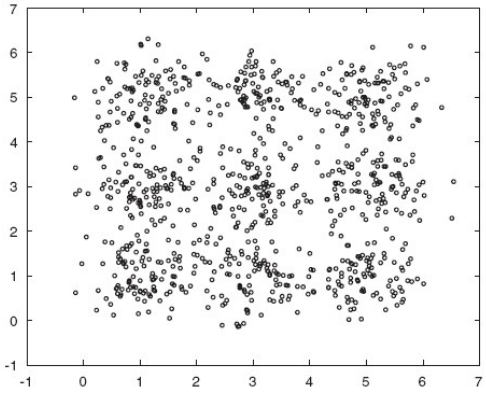
Obs. $\mathbf{x}_i$	x_{i1}	x_{i2}
1	1	2
2	2	1
3	1	1
4	2	2
5	8	9
6	9	8
7	9	9
8	8	8
9	1	15
10	2	15
11	1	14
12	2	14

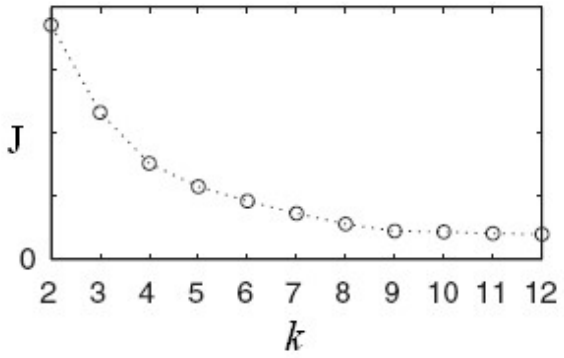


- Run k-means varying k from k=2 up to k=5 in the above dataset, and then plot J as a function of k

How to Determine k ?

- The previous procedure works when clusters are well separated from each other. However, in practice, clustering is oftentimes not so trivial





- There is a broad subarea of clustering, called clustering validation, that is related to quantitative evaluation, model selection, and statistical validation of clustering results
  - In the following we illustrate one basic procedure commonly used in practice

Silhouette Width Criterion

SWC = Average of the Silhouette of Individual Observations: $SWC = \frac{1}{N}\sum_{i=1}^{N} s(i)$

Silhouette of the ith observation: $s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$ (s(i) := 0 for singletons)

$a(i)$: Average distance from the ith observation to the other observations in its cluster

$b(i)$: Average distance from the ith observation to the observations in the nearest neighbouring cluster

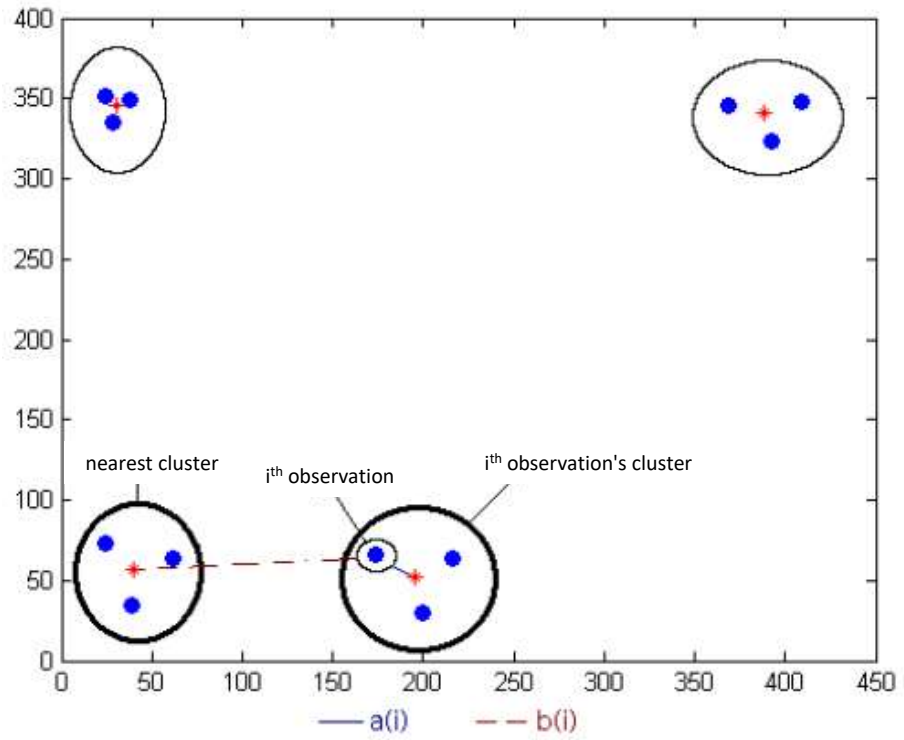
Simplified Version (Computationally More Efficient): $a(i)$ is the distance from the ith observation to its cluster centroid, whereas $b(i)$ is the distance from the ith observation to the centroid of the nearest cluster (other than the one the ith observation belongs to)

SWC Range: SWC $\in$ [–1 ,+1]

Simplified Silhouette Criterion (Illustration)

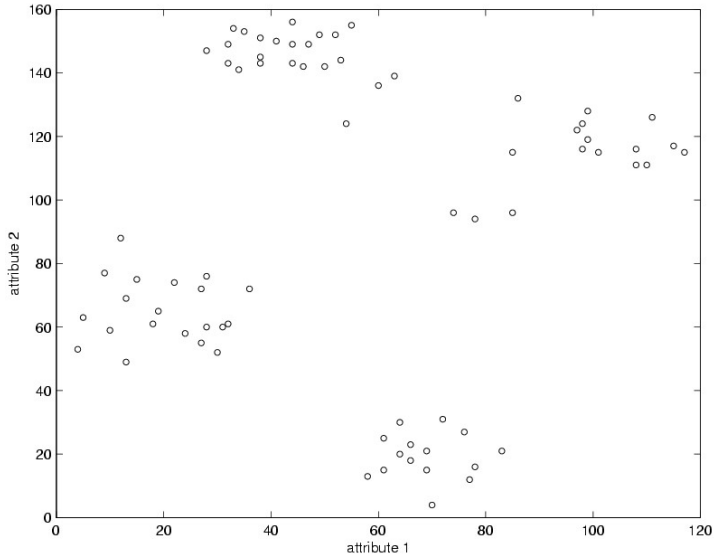
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

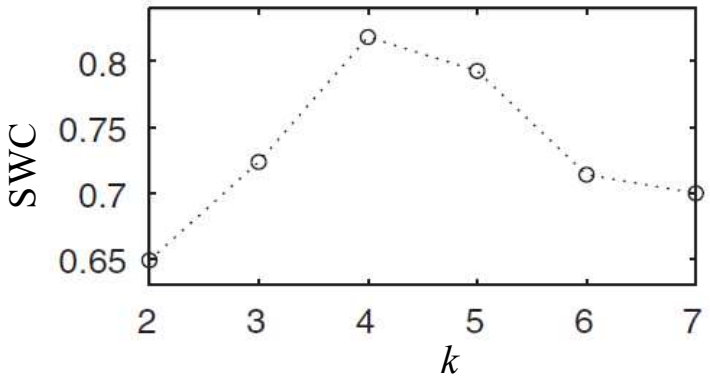
$$SWC = \frac{1}{N}\sum_{i=1}^{N} s(i)$$



Silhouette Criterion (Model Selection Example)

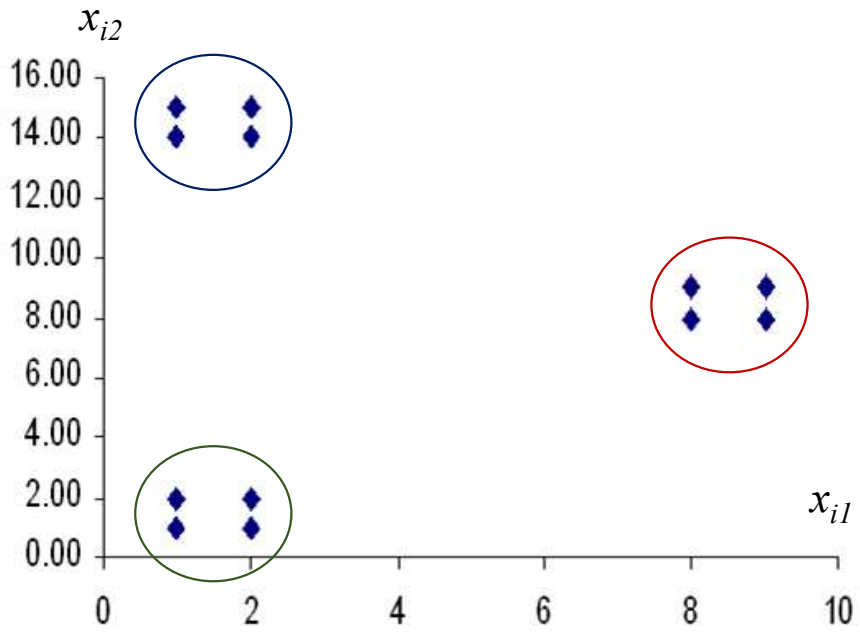
- Determining k for k-means using the Silhouette Criterion





Exercise

Obs. $\mathbf{x}_i$	x_{i1}	x_{i2}
1	1	2
2	2	1
3	1	1
4	2	2
5	8	9
6	9	8
7	9	9
8	8	8
9	1	15
10	2	15
11	1	14
12	2	14



- Compute the simplified version of the SWC for the natural partitioning of the dataset above, highlighted in circles. Repeat for a different partition, say merging two clusters or splitting a cluster into two, and compare the results

Exercise

Discuss the computational complexity (runtime) of both the original as well as the simplified silhouette in terms of the size N (number of data observations, i.e., rows) of an (N by n) dataset, assuming that a clustering partition is provided as input in the form of cluster labels for each observation, in addition to the dataset itself (which is required to compute cluster centroids in the simplified silhouette). Since the simplified silhouette (unlike the original silhouette) cannot be computed solely based on pre-computed pairwise distances between observations, for a fair comparison, assume that distances are not given as input and need to be computed from the inputed data and clusters as part of the silhouette computations. For the sake of simplicity, you can assume that the number of clusters k and the number of variables n (dataset columns) are both small constants as compared to N (n, k << N). You can also assume squared Euclidean distance.

References (Part 2)

- D. Steinley, K-Means Clustering: A Half-Century Synthesis, British J. of Mathematical and Stat. Psychology, V. 59, 2006.
- A. K. Jain, Data Clustering: 50 Years Beyond K-Means, Pattern Recognition Letters, 2010.
- P.-N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, 2nd Edition, Pearson, 2018.
- G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2013.
- G. L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill, 1968.