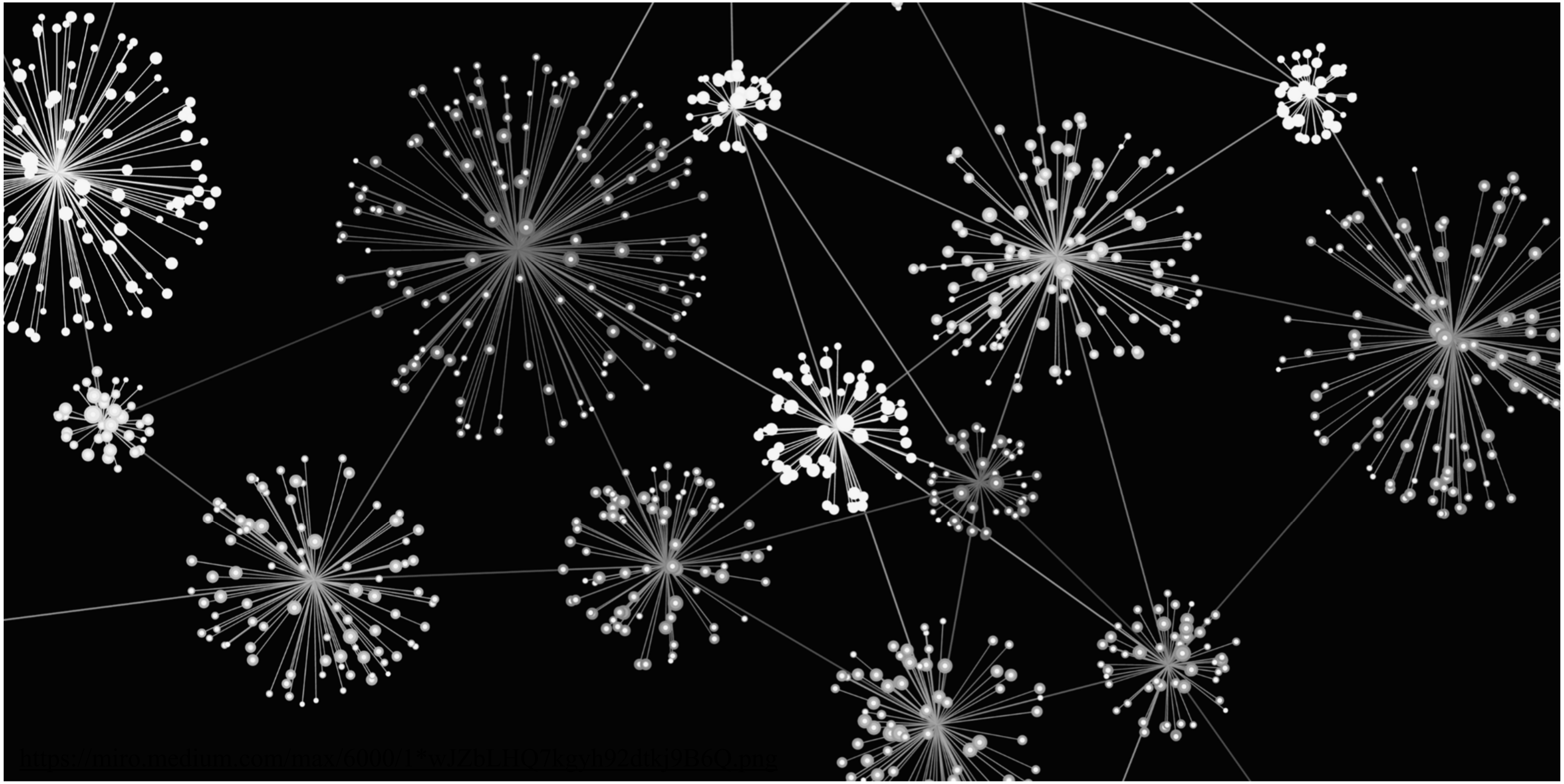

https://miro.medium.com/max/6000/1*wJZbLHQ7kgyh92dtkj9B6Q.png

Lecture 4: Scalable k-means Clustering

COM6012: Scalable ML by Haiping Lu

YouTube Playlist: https://www.youtube.com/c/HaipingLu/

Week 4 Contents / Objectives

- Introduction to Cluster Analysis
- k-means Clustering
- Scalable k-means
- k-means in Spark & Limitations

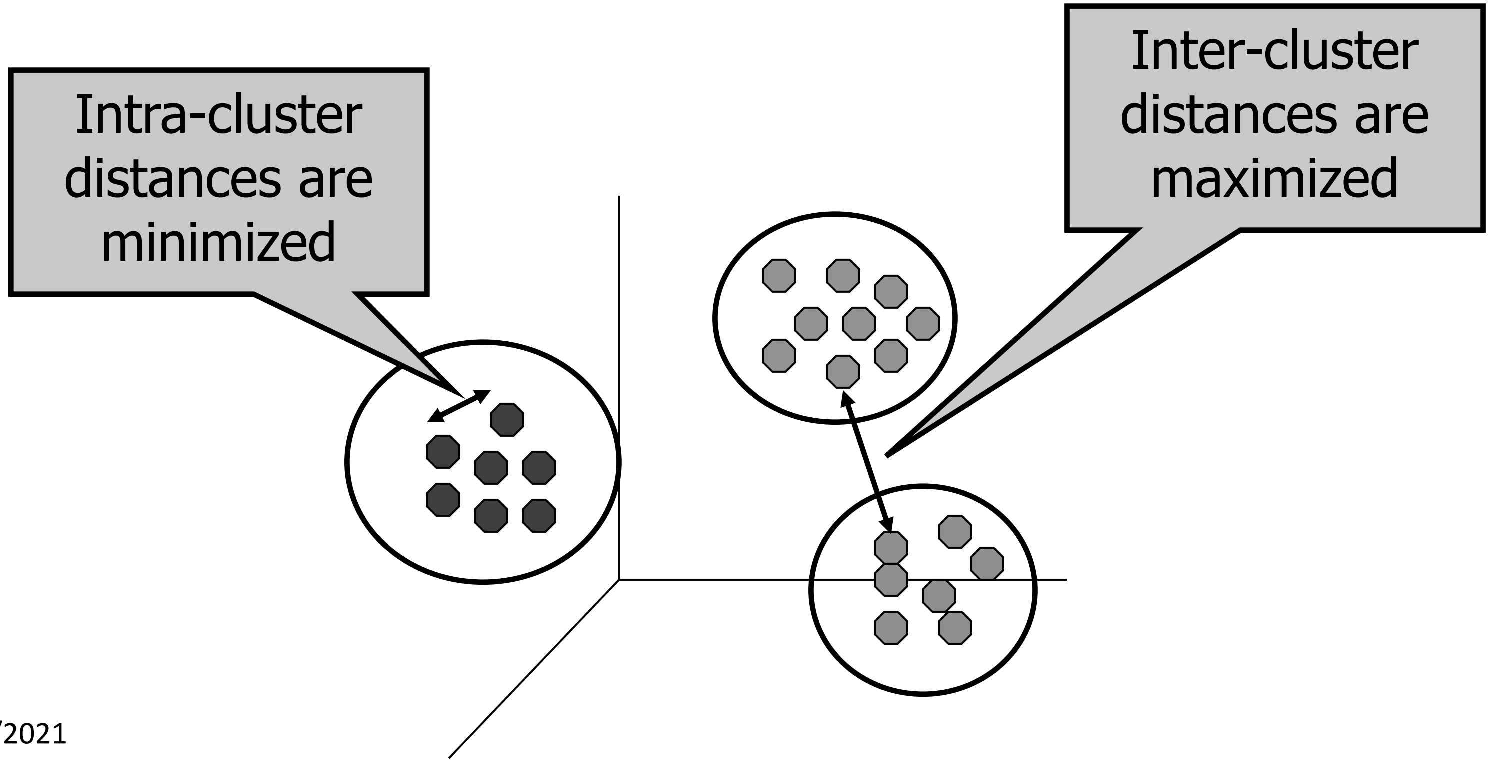
Week 4 Contents / Objectives

- Introduction to Cluster Analysis
- k-means Clustering
- Scalable k-means
- k-means in Spark & Limitations

Cluster Analysis

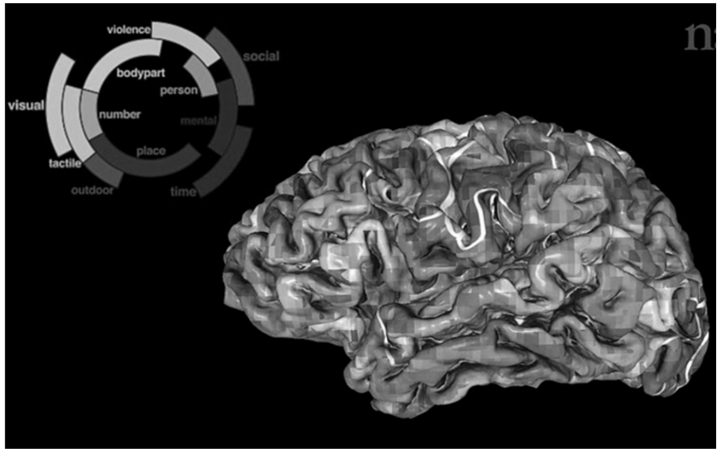
Output	Supervised	Unsupervised
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality Reduction

- "Classification without labelled data"
- "Dimensionality reduction to 1 (cluster index)"



Cluster Analysis

- Divide data into clusters that are meaningful/useful
- Clusters: pseudo-classes
- Key for <u>exploratory data analysis</u> in many areas
  - Brain parcellation, social network analysis, customer segmentation, patient stratification, drug discovery, …



brain-video2.jpg (1014×570) (wp.com)

https://miro.medium.com/max/1145/1*YjvoSe-hGaPpbWi5fDqV2w.png

segmentation3.png (586×236) (visionedgemarketing.com)

Anomaly/Outlier Detection

- Anomalies: data points very different from others
- Applications: credit card fraud, network intrusion, unknow virus, eco disturbance, unusual symptom, …

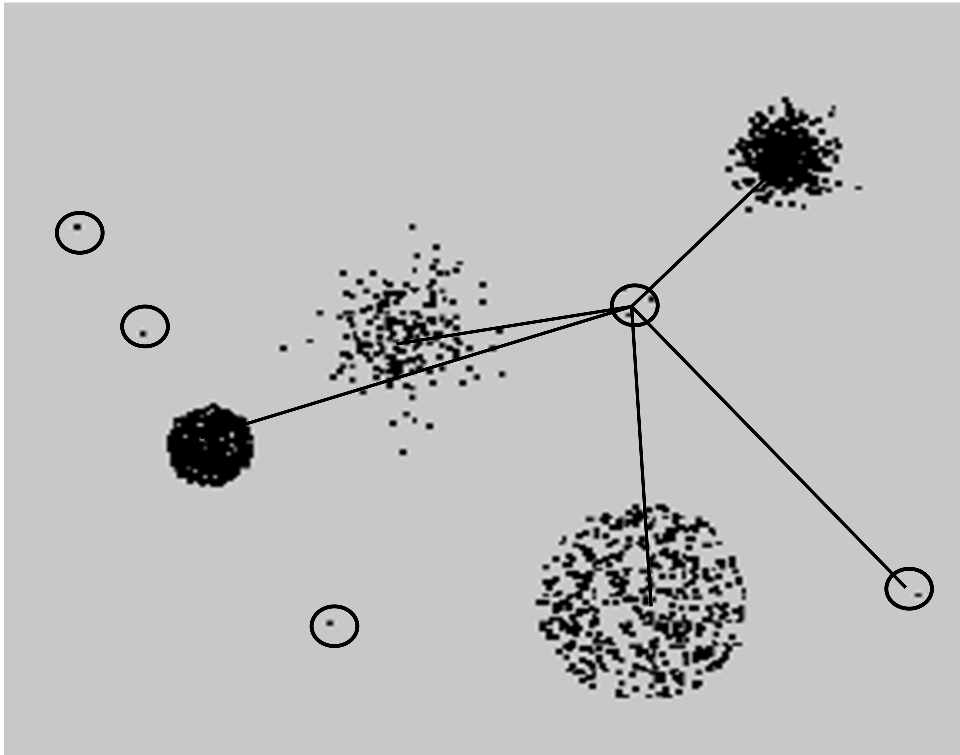
financial_protection_credit_card_fraud_img1.jpg (512×347) (traderdefenseadvisory.com)

Network-Intrusion-Detection-and-Prevention-.jpg (1600×962) (wp.com)

newseventsimage_1613554577167_mainnews2012_x1.jpg (700×484) (imperial.ac.uk)

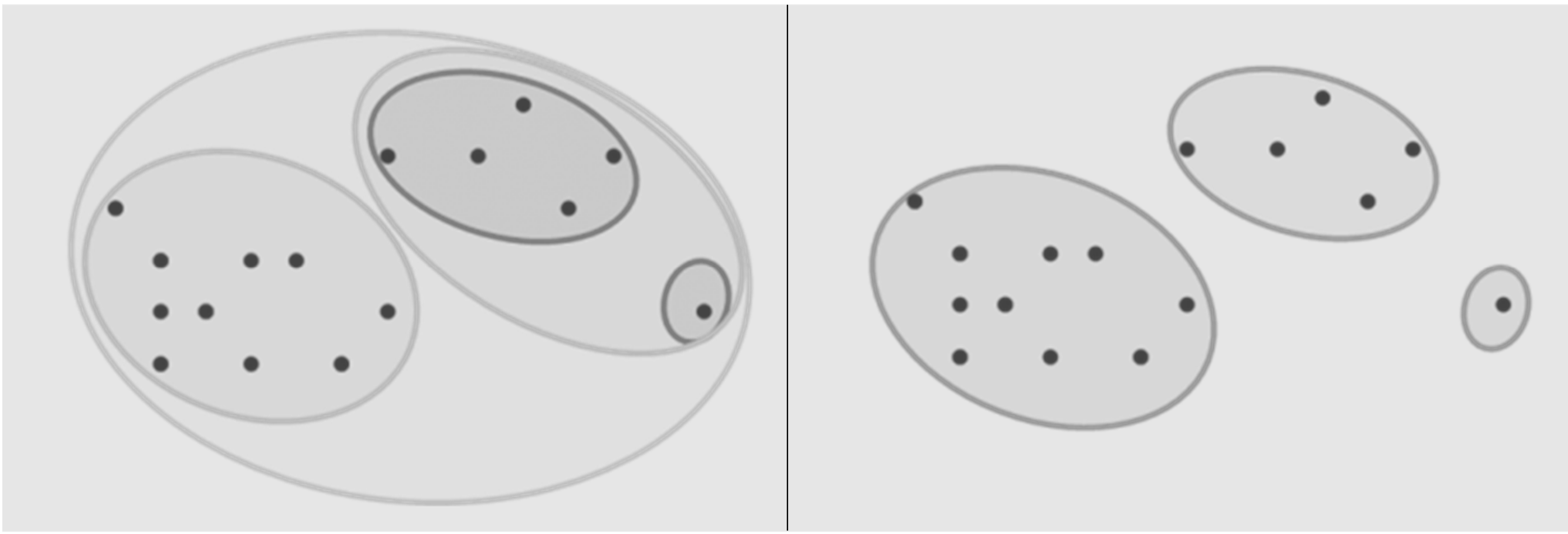
Clustering-based Anomaly Detection

- Cluster the data points
- Those in small clusters → candidate outliers
- Compute the distance between candidate points and non-candidate clusters
- Candidate points far from all other non-candidate points → outliers

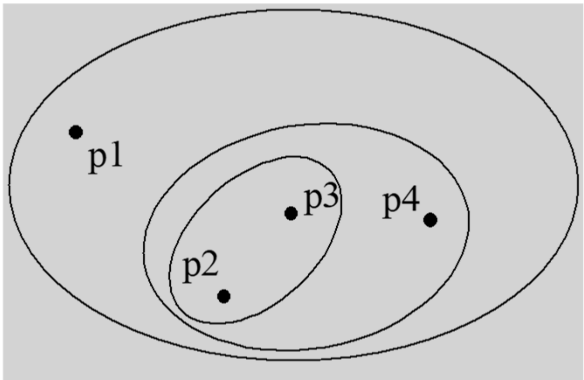


Hierarchical vs Partitional Clustering

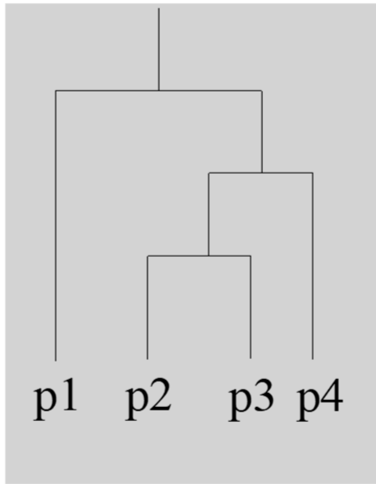
- Hierarchical: nested clusters as a hierarchical tree
  - Each node (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters)
  - The root of the tree → the cluster containing all data points
- Partitional: non-overlapping clusters
  - Each data point is in exactly one cluster



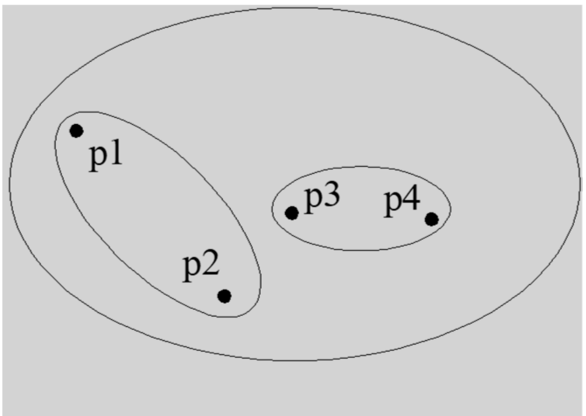
Hierarchical Clustering



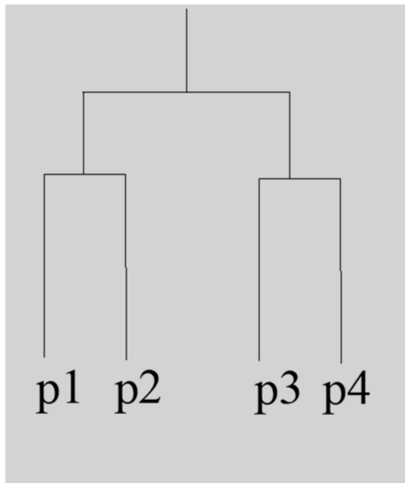
Traditional Hierarchical Clustering



Traditional Dendrogram



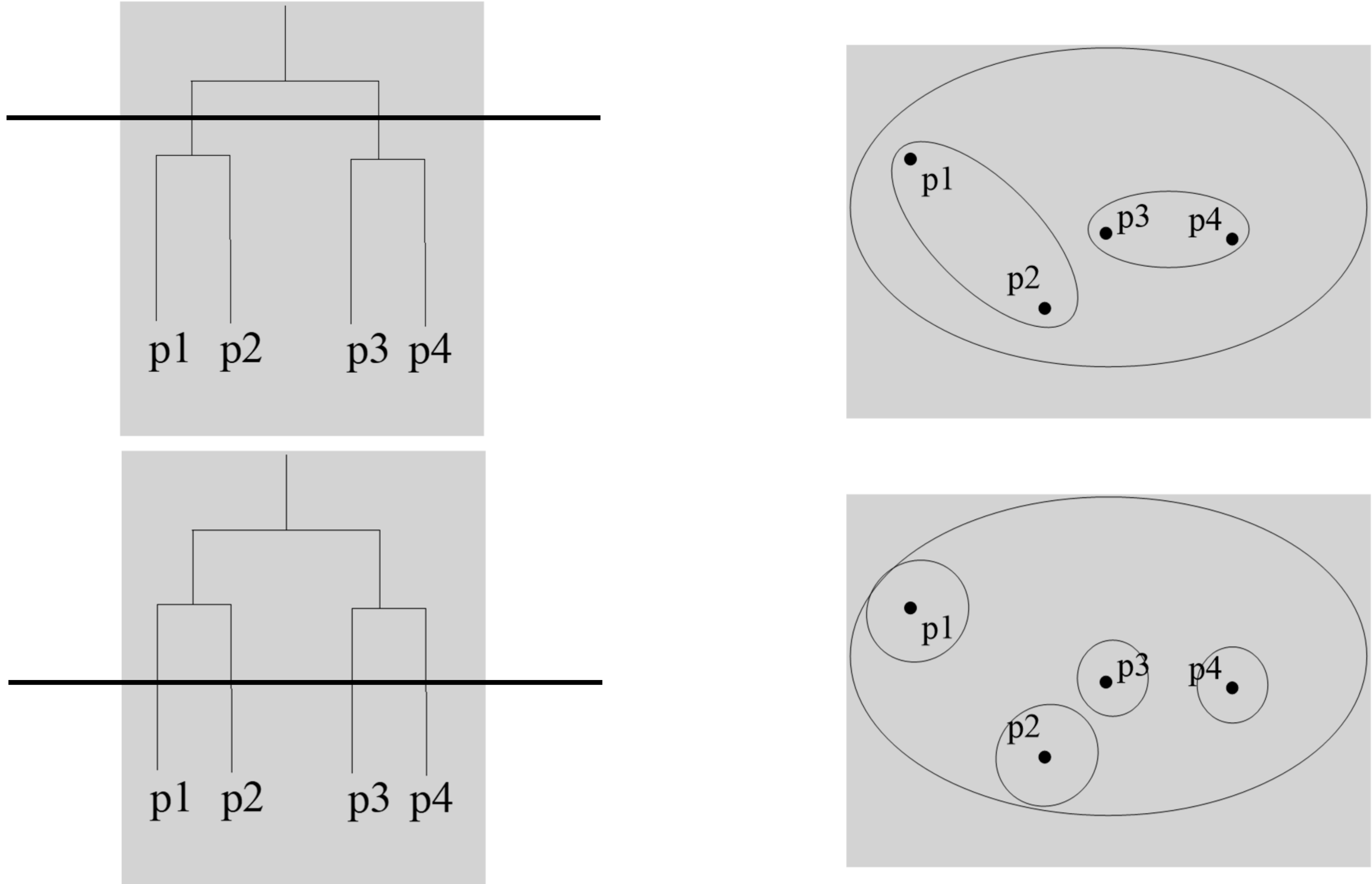
Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Hierarchical → Partitional

- Hierarchical = a sequence of partitional clustering cutting the hierarchical tree at a particular level



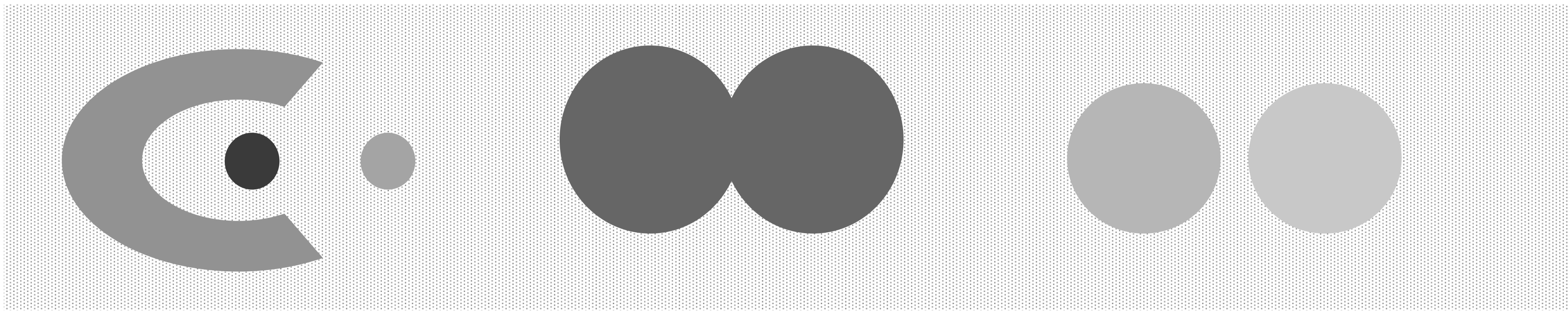
Centre/Prototype-based Clusters

- Data points in a cluster are closer (more similar) to the centre of that cluster than to the centre of any others
- Centre
  - Centroid: the average of all the points in a cluster
  - Medoid: the most representative point of a cluster (e.g., categorical)

4 centre-based clusters

Density-based Clusters

- Cluster: a dense region of points separated by low-density regions from other regions of high density
- Used when the clusters are irregular/intertwined, and when noise and outliers are present



6 density-based clusters

Week 4 Contents / Objectives

- Introduction to Cluster Analysis
- k-means Clustering
- Scalable k-means
- k-means in Spark & Limitations

k-means Clustering

- A centre-based, partitional clustering approach
- Input: a set of n data points $X=\{x_1, x_2, \ldots, x_n\}$ and the number of clusters k
- For a set $C=\{c_1, c_2, \ldots, c_k\}$ of cluster centres, define the Sum of Squared Error (SSE) as:

$$SSE_X(C) = \sum_{x \in X} d(x, C)^2$$

$d(x,C)$: distance from x to the closest centre in C

- Goal: find C centres minimising $SSE_X(C)$

Lloyd Algorithm for k-means

- Start with k centres $\{c_1, c_2, \ldots, c_k\}$ chosen uniformly at random from data points
- Assign clusters and compute centroids till convergence
  - Alternating optimisation again, similar to ALS
- Limitations
  - Many iterations to converge
  - Sensitive to initialisation
  - Random initialisation can get two centres in the same cluster→ stuck in a local optimum (example on next slide)

Initialisation → Stuck

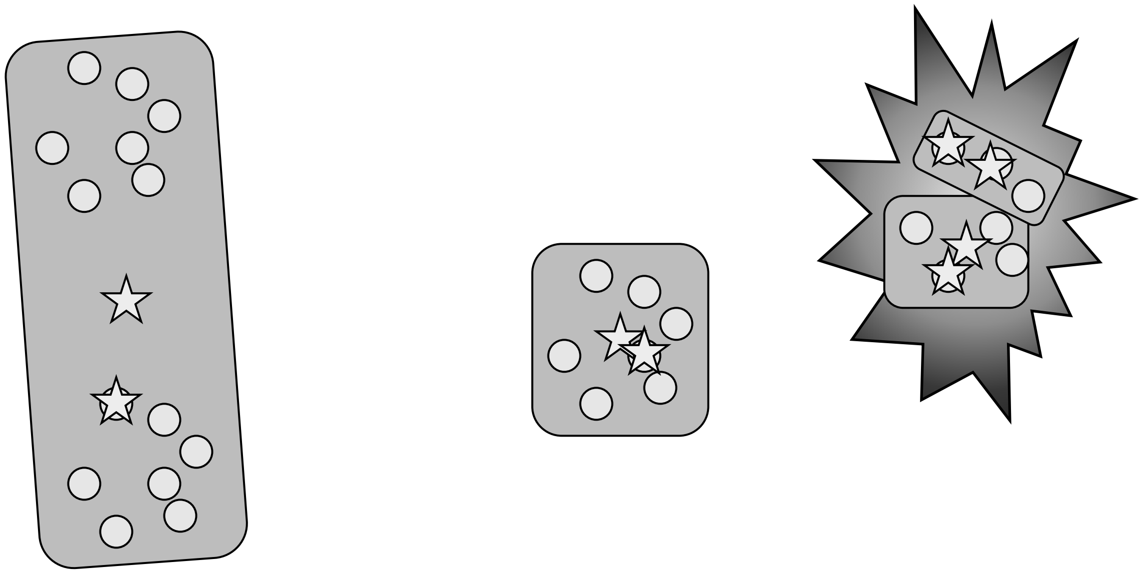


Figure credited to David Arthur

Week 4 Contents / Objectives

- Introduction to Cluster Analysis
- k-means Clustering
- Scalable k-means
- k-means in Spark & Limitations

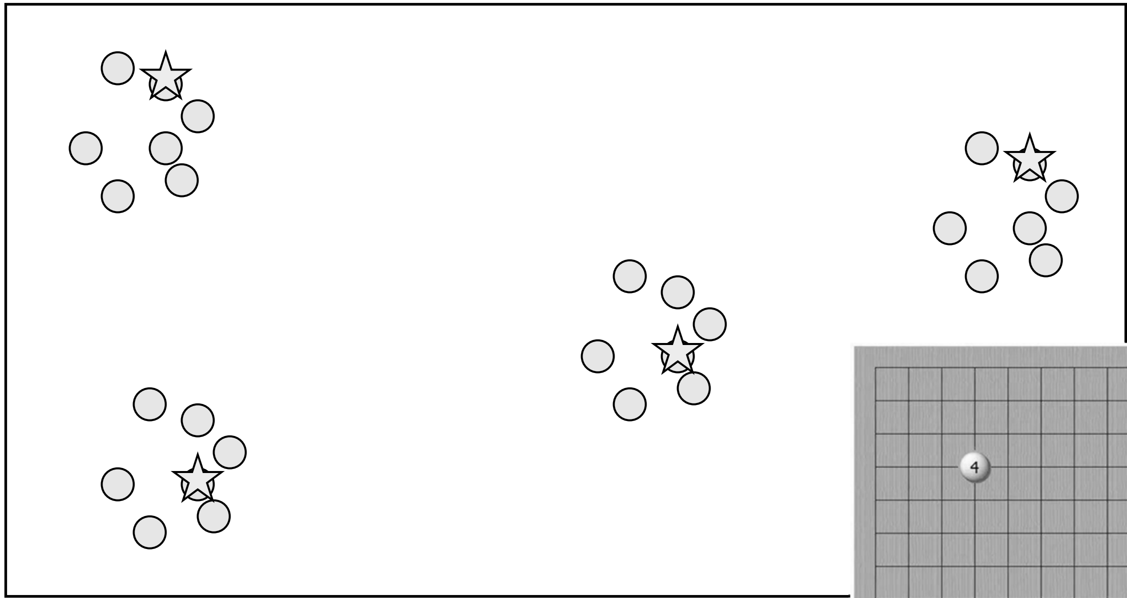
k-means++ [Arthur et al. ’07]

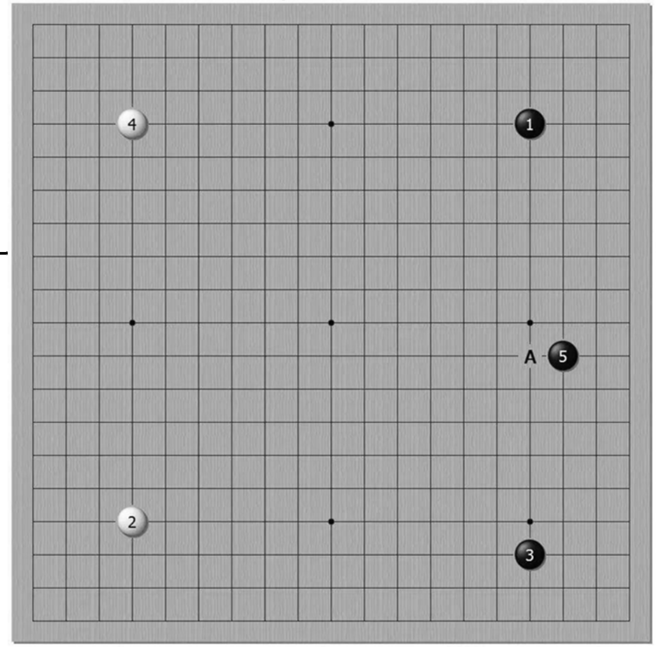
- Key idea: spread out the centres
- Choose the first centre c_1 uniformly at random
- Repeat for $2 \leq i \leq k$:
  - Choose c_i to be equal to a data point x_0 sampled from the distribution:

$$\frac{d(x_0, C)^2}{SSE_X(C)} \propto d(x_0, C)^2$$

  - Reminder: $d(x,C) =$ distance from x to the closest centre in C
- Theorem: $O(\log k)$-approximation to the optimum

k-means++ Initialisation





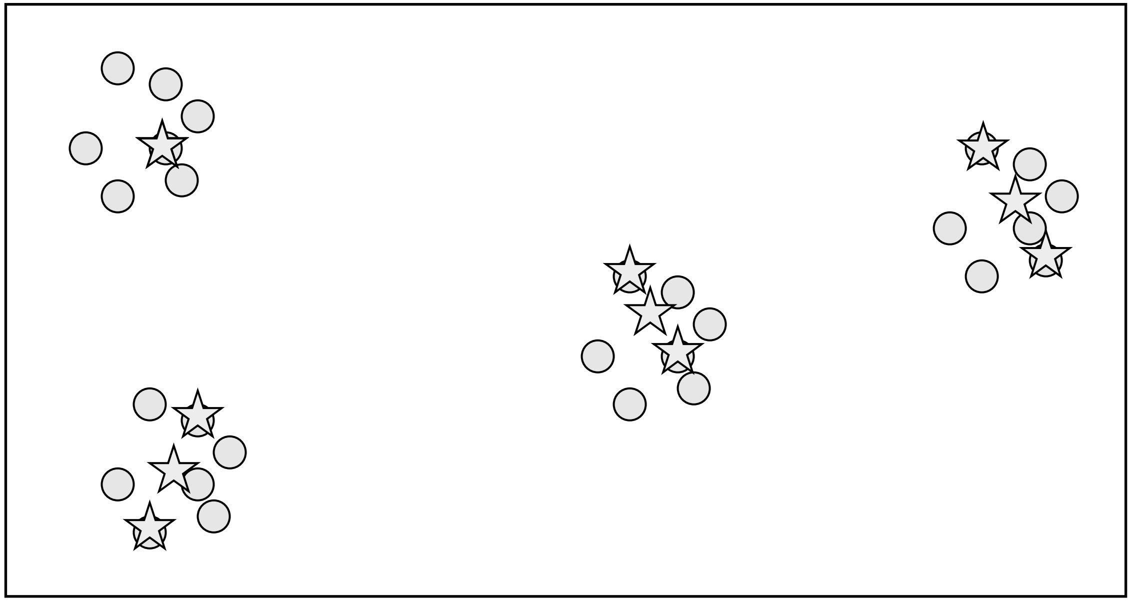
k-means++ → k-means||

- k-means++ limitations
  - Needs k passes over the data for initilisation
  - In big data applications, k is typically large (e.g., 1000) → not scalable!
- A promising solution
  - k -means++ samples just one point per iteration
  - What if we oversample by sampling each point independently with a larger probability?
  - Equivalent to updating the distribution less frequently
  - Coarser sampling → k-means|| [Bahmani et al. ’12]

k-means|| Initialisation

k=4, oversampling factor L=3

Cluster the intermediate centres



k-means|| [Bahmani et al. '12]

- Choose the oversampling factor L>1
- Initialise C to an arbitrary set of points
- For R iterations do:
  - Sample each point x in X independently with probability $p_x = Ld(x,C)^2/SSE_X(C)$.
  - Add all the sampled points to C
- Cluster the intermediate centres in C using k-means++
- Benefits over k-means++
  - Less susceptible to noisy outliers
  - More reduction in the number of Lloyd iterations

k-means||: Relationships

- An interpolation between Lloyd and k-means++

$R=k$: k-means++ (L=1) → strong guarantee

Number of iterations (R)

Small R: k-means|| → guaranteed approximation to the optimal

R=0: Lloyd → no guarantees

Week 4 Contents / Objectives

- Introduction to Cluster Analysis
- k-means Clustering
- Scalable k-means
- k-means in Spark & Limitations

k-means in Spark

- Implemented in the RDD API Mllib - `Kmeans`
- Scala `ml` code: [org/apache/spark/ml/clustering/KMeans.scala]()

```scala
import org.apache.spark.ml.util._
import org.apache.spark.ml.util.Instrumentation.instrumented
import org.apache.spark.mllib.clustering.{DistanceMeasure, KMeans => MLlibKMeans, KMeansModel => MLlibKMeansModel}
import org.apache.spark.mllib.linalg.{Vector => OldVector, Vectors => OldVectors}
import org.apache.spark.mllib.linalg.VectorImplicits._
```

- Scala `mllib`: [org/apache/spark/mllib/clustering/KMeans.scala]()

```scala
/**
 * K-means clustering with a k-means++ like initialization mode
 * (the k-means|| algorithm by Bahmani et al).
 *
 * This is an iterative algorithm that will make multiple passes over the data, so any RDDs given
 * to it should be cached by the user.
 */
@Since("0.8.0")
class KMeans private (
    private var k: Int,
```

k-means API

- k: the number of desired clusters
- maxIter: the maximum number of iterations
- initMode: random or via k-means|| (default) initialization
- initSteps: the number of steps in the k-means|| algorithm (default=2, oversampling factor $L = 2$ x k)
- tol: the distance threshold for checking convergence
- seed: the random seed
- distanceMeasure: Euclidean (default) or cosine dist measure
- weightCol: optional weighting of data points

```
    // On each step, sample 2 * k points on average with probability proportional
    // to their squared distance from the centers. Note that only distances between points
    // and new centers are computed in each iteration.
    var step = 0
    val bcNewCentersList = ArrayBuffer[Broadcast[_]]()
    while (step < initializationSteps) {
```

Remember to Cache

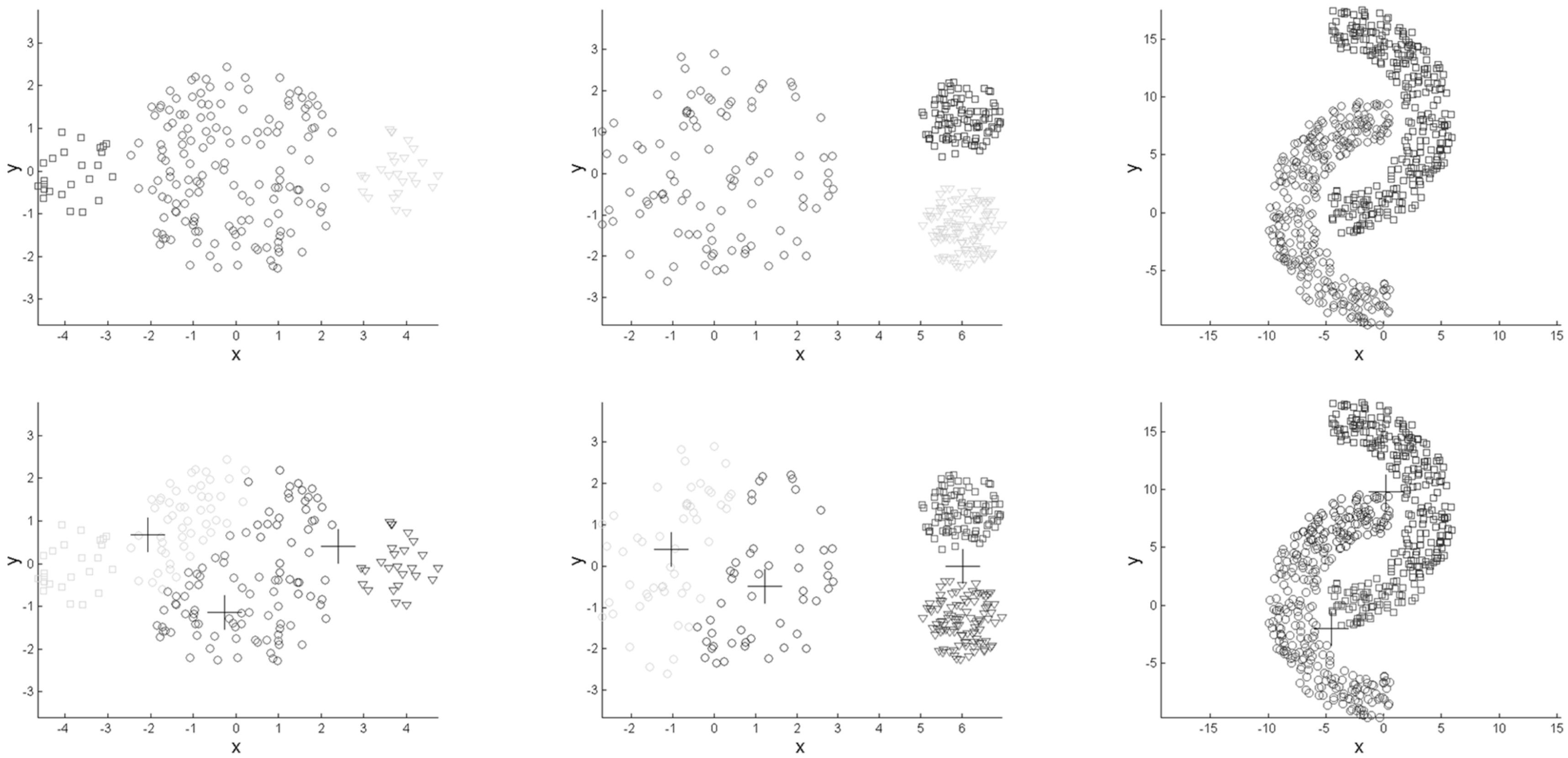
- Data need to be cached for high performance
- See from the source code

```
/**
 * K-means clustering with a k-means++ like initialization mode
 * (the k-means|| algorithm by Bahmani et al).
 *
 * This is an iterative algorithm that will make multiple passes over the data, so any RDDs given
 * to it should be cached by the user.
 */
@Since("0.8.0")
class KMeans private (
```

```
   * Train a K-means model on the given set of points; `data` should be cached for high
   * performance, because this is an iterative algorithm.
   */
  @Since("0.8.0")
  def run(data: RDD[Vector]): KMeansModel = {
    val instances = data.map(point => (point, 1.0))
    val handlePersistence = data.getStorageLevel == StorageLevel.NONE
```

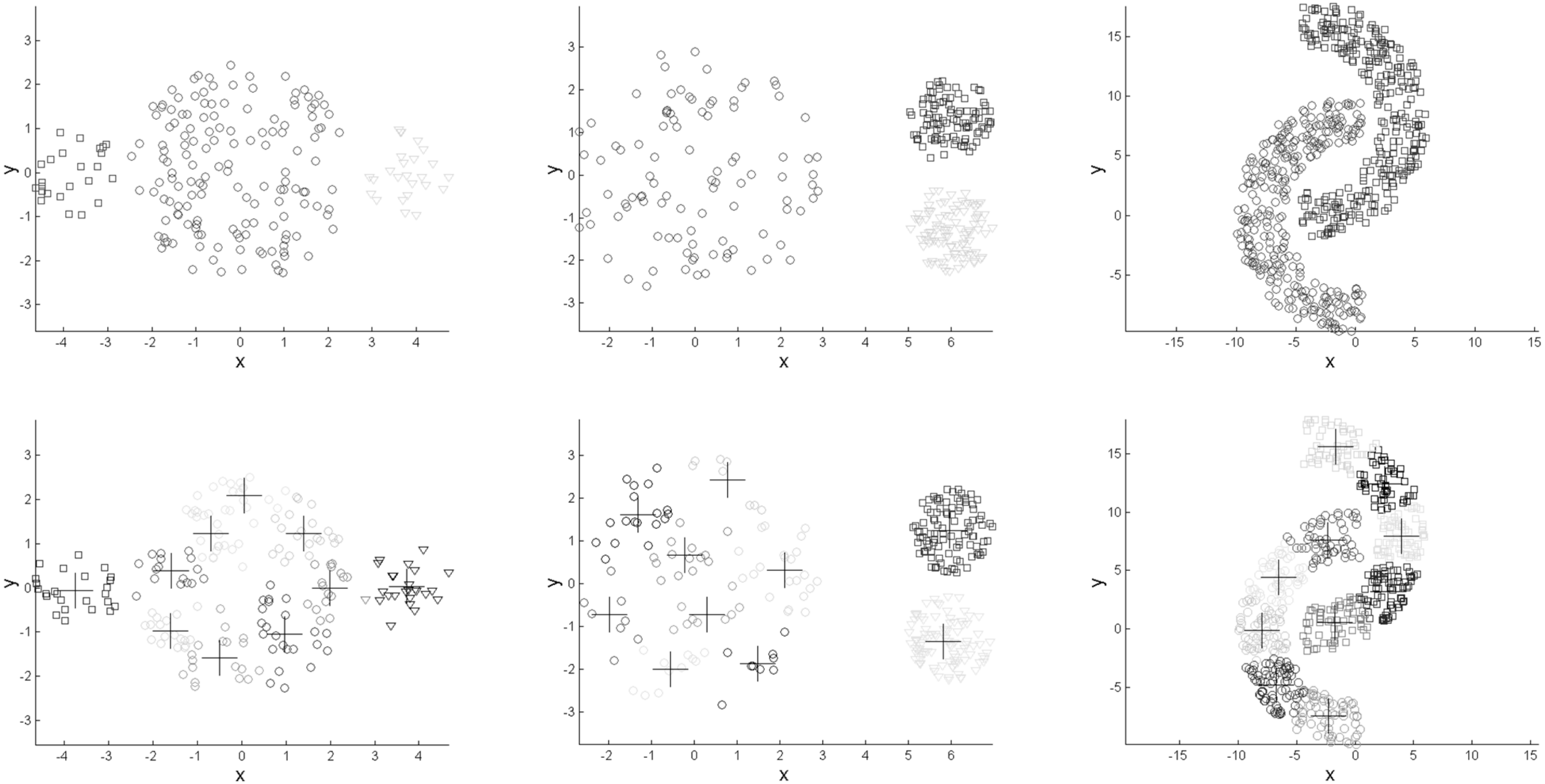
Limitations of k-means

- Problems when clusters are of differing sizes, densities, or non-globular shapes



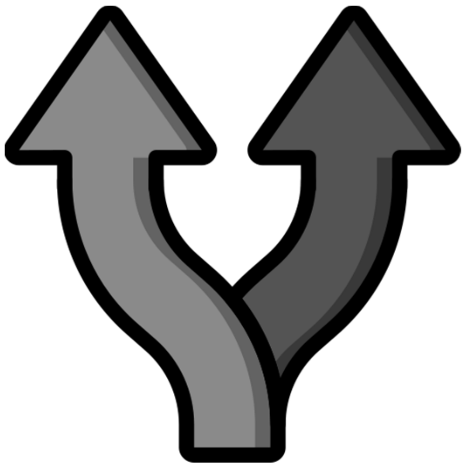
Overcoming k-means Limitations

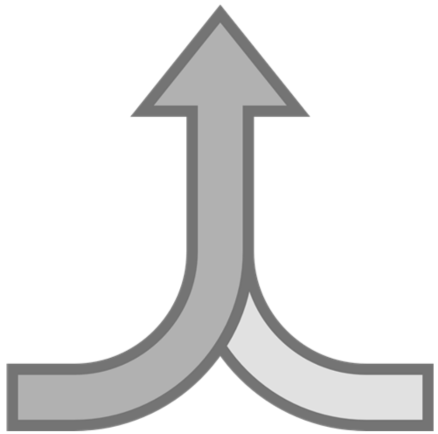
- Use many clusters to find parts of clusters →put together



Pre-processing & Post-processing

- Pre-processing
  - Normalise the data (distance measure)
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split loose clusters (with relatively high SSE)
  - Merge clusters that are close (with relatively low SSE)





2164571.png (512×512) (flaticon.com)

351-3511018_merge-png-split-merge-icon-clipart.png (1521×1521) (pinclipart.com)

Acknowledgement & References

- Acknowledgement
  - Some slides are adapted from 1) the k-means|| slides by Bahman Bahmani, Stanford University, 2012, and 2) Tan, Steinbach, Kumar's slides for the book Introduction to Data Mining
- References
  - Chapter on clustering from the book above (88 pages)
  - k-means overview
  - k-means ++ paper
  - k-means || paper and video