UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

3253 Analytic Techniques and Machine Learning

Module 4: Clustering and Unsupervised Learning

Course Plan

Module Titles
Module 1 – Introduction to Machine Learning
Module 2 – End to End Machine Learning Project
Module 3 – Classification
Current Focus: Module 4 – Clustering and Unsupervised Learning
Module 5 – Training Models and Feature Selection
Module 6 – Support Vector Machines
Module 7 – Decision Trees and Ensemble Learning
Module 8 – Dimensionality Reduction
Module 9 – Introduction to TensorFlow
Module 10 – Introduction to Deep Learning and Deep Neural Networks
Module 11 – Distributing TensorFlow, CNNs and RNNs
Module 12 – Final Assignment and Presentations (no content)

Learning Outcomes for this Module

- Distinguish and describe unsupervised learning
- Identify clustering concepts
- Become familiar with clustering algorithms: k-means, DBSCAN, hierarchical

Topics for this Module

- **4.1** Unsupervised learning
- **4.2** Clustering
- **4.3** K-Means clustering
- **4.4** DBSCAN clustering
- **4.5** Hierarchical clustering
- **4.6** Visual presentation of clusters
- **4.6** Resources and Wrap-up

Module 4 – Section 1

Unsupervised Learning

Supervised vs. Unsupervised Learning

- Algorithms used to build classifiers need supervised data examples
- The input data to the learner consists of examples $(x_1, y_1), \ldots (x_n, y_n)$
- An example (x_i, y_i) shows the correct response y_i to the input x_i
- In unsupervised ML the learner does not have labels, only examples $x_1, \ldots, x_n$

Unsupervised Learning

- A clustering algorithm will still produce an output $C(x) = c$ given an input x
- However, there is no way to know if the output is correct or not
- The learning algorithm does not optimize a cost function based on labels
- But some classification algorithms do optimize a cost function based on the input examples $x_1, \ldots, x_n$

Key utilities of unsupervised learning

- **Dimensionality reduction**: represent each input case using a small number of variables. Example algorithms (will be covered in week 8)
  - principal components analysis (PCA)
  - factor analysis
  - independent components analysis

  Dimensionality reduction can also be used for data compression
- **Grouping similar instances into clusters**
  - Customer segmentation
    - answering call
    - opening email/SMS
  - New article topic detection
  - Customer call reason (apple, broken, expensive, repair, care)

Key utilities of unsupervised learning

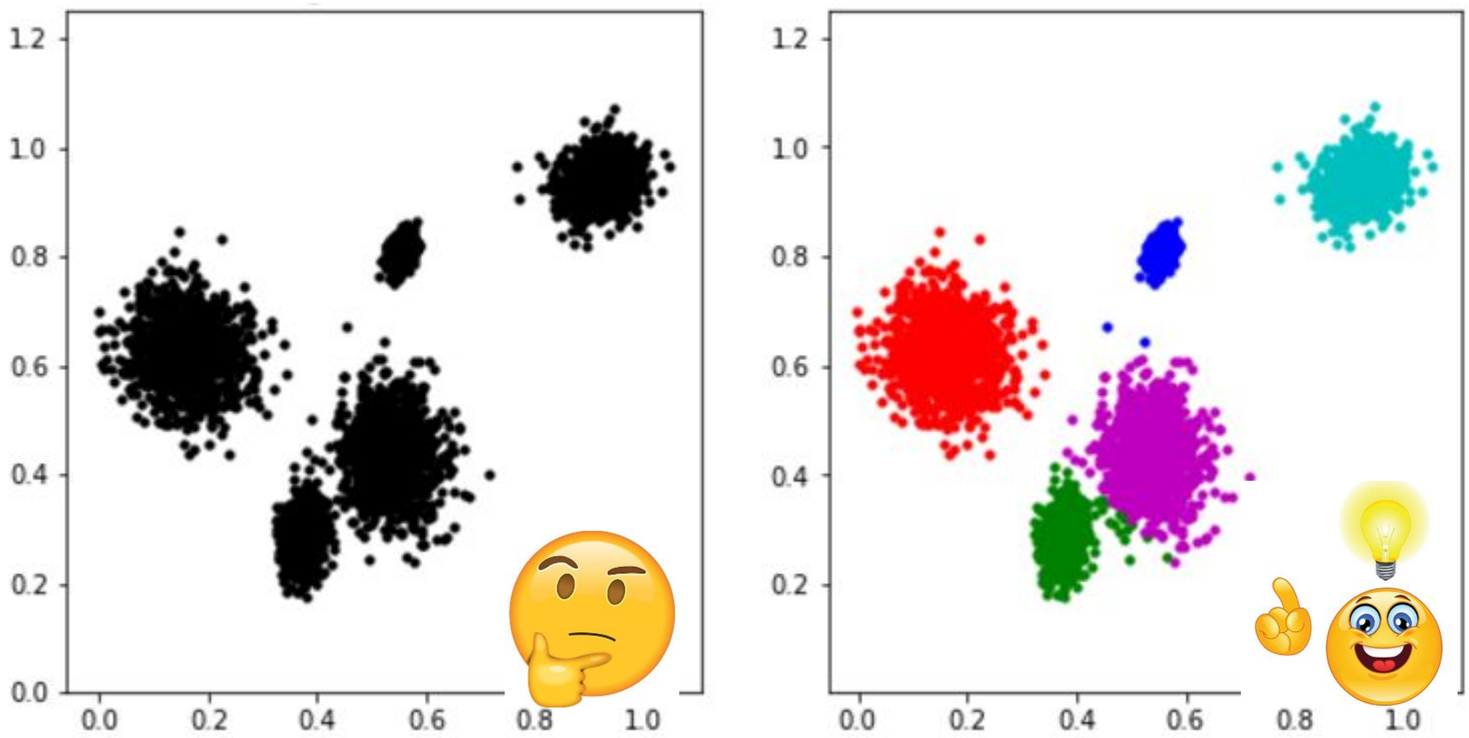
- **Anomaly or outlier detection**
  - Keep normal instances and outliers in separate clusters
  - Fraud detection is an example
  - Manufacturing defects
- **Density estimation**
  - Estimate probability density function (PDF) of random process that generated dataset
  - Instances located in low density can be considered outlier
- **Semi-supervised learning**
  - If you have few labels in your dataset, you can cluster them and then apply clustering on unlabeled instances to generate label so you have bigger labeled dataset
  - Perhaps, labels have marginal errors

Module 4 – Section 2

Clustering

Clustering Goal

- The aim is to group points (examples) into a small number of clusters



Clustering Goal (cont'd)

- Similar instances expected to go to the same cluster
- Dissimilar instances expected to be in different clusters
- The clustering algorithm also learns how to assign a cluster to an unseen instance later

Clustering input

- Input: n vectors, m-dimensional, represent the objects to be clustered

- Can start with object themselves (e.g. documents), but need a vector representation

  Document □ vector of word counts

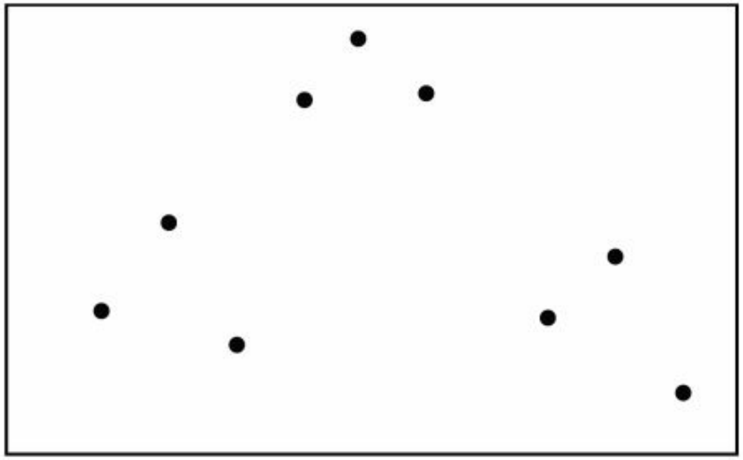
- Vectors have same (fixed length) but clustering can be done over sequences of different length (the matrix of distances is needed)

Clustering assumption

- We assume that the data was generated from a number of different classes.
- The aim is to cluster data from the same class together.
  - How many classes?
  - Why not put each datapoint into a separate class?
  - What is the objective function that is optimized by sensible clustering?

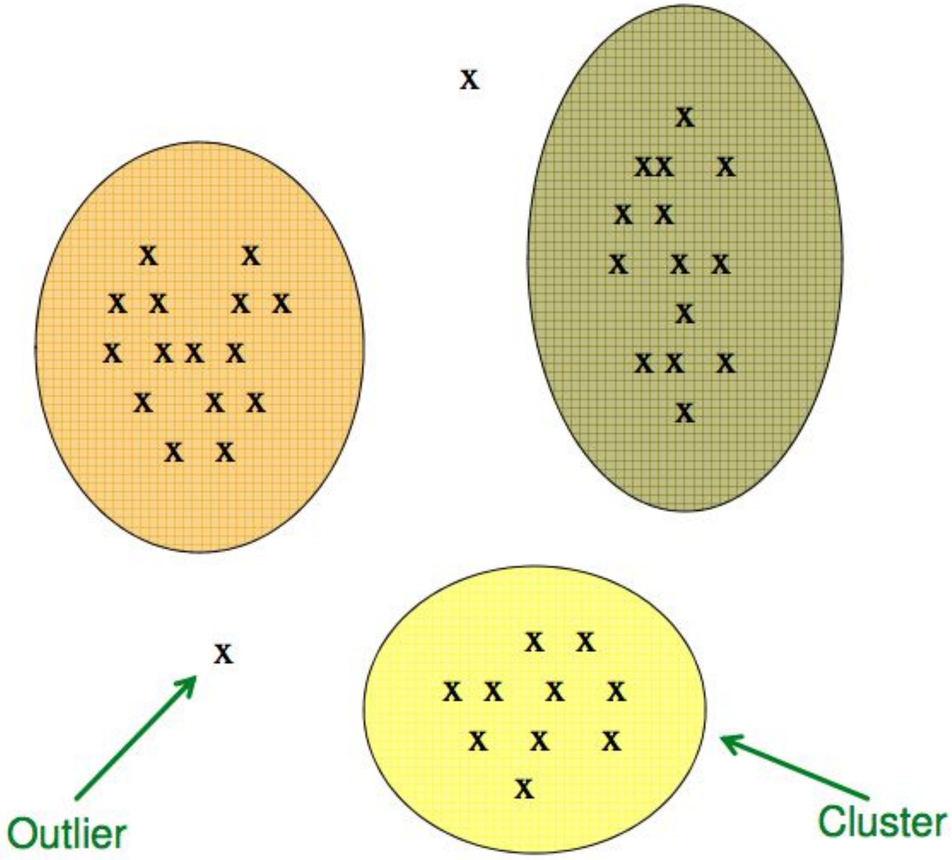
Clustering assumption cont’d

- Assume the data {x(1), . . . , x(N)} lives in a Euclidean space, x(n) ∈ Rd
- Assume the data belongs to K classes (patterns)
- How can we identify those classes (data points that belong to each class)?



UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Clustering and Outliers



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Clustering and Feature reduction

- An important part of building models is feature reduction
- Many variables could be used to predict a target, but some of them could carry little or no information about the target
- Clustering the features (columns, instead of rows) is a way to reduce the dimensionality by picking a representative on each cluster
- Python Scikit-Learn provides this with FeatureAgglomeration

Module 4 – Section 3

K-Means

k-means Algorithm

- K-means or Lloyd algorithm named after Stuart P. Lloyd
- **Input**: vectors $S = \{x^{(1)}, \dots, x^{(n)}\}$
  $k =$ number of desired clusters
- **Output**: a partition of S into k clusters, and the clusters' average (centroid)
- **Goal**: $S_1, \dots, S_k$ should minimize the squared distances between each example x_i and its closest centroid $c(x_i)$

$$\sum_{i=1}^{n} ||x_i - c(x_i)||^2$$

- Lloyd's algorithm finds (a good enough) solution

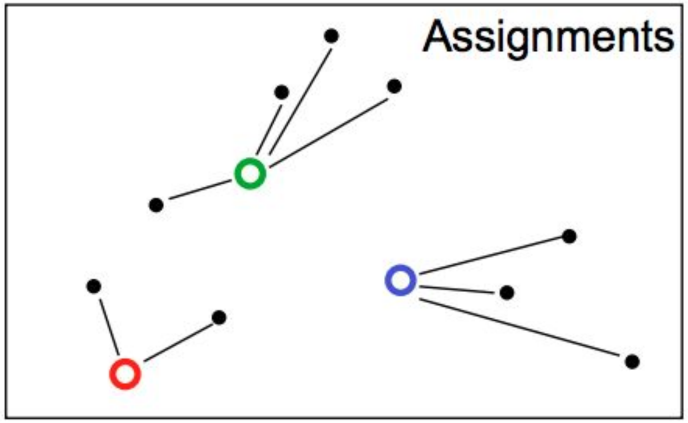
k-means Algorithm

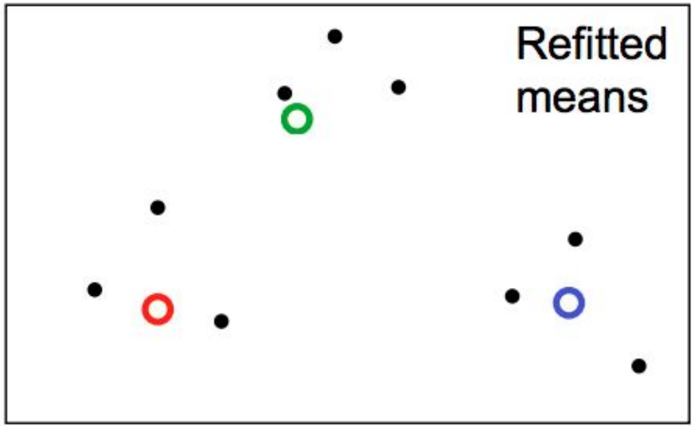
1) Start with a set of k centroids (random points from S)

2) Loop over:

- **a)Assign:** Assign each point to the centroid to which it is closest until all instances are assigned: this defines clusters
- **b) Refit:** Update the centroids as the mean within each cluster

3) Repeat (a) and (b) until the centroids change is very small (threshold)





http://syskall.com/kmeans.js/

http://shabal.in/visuals/kmeans/2.html

k-means (cont'd)

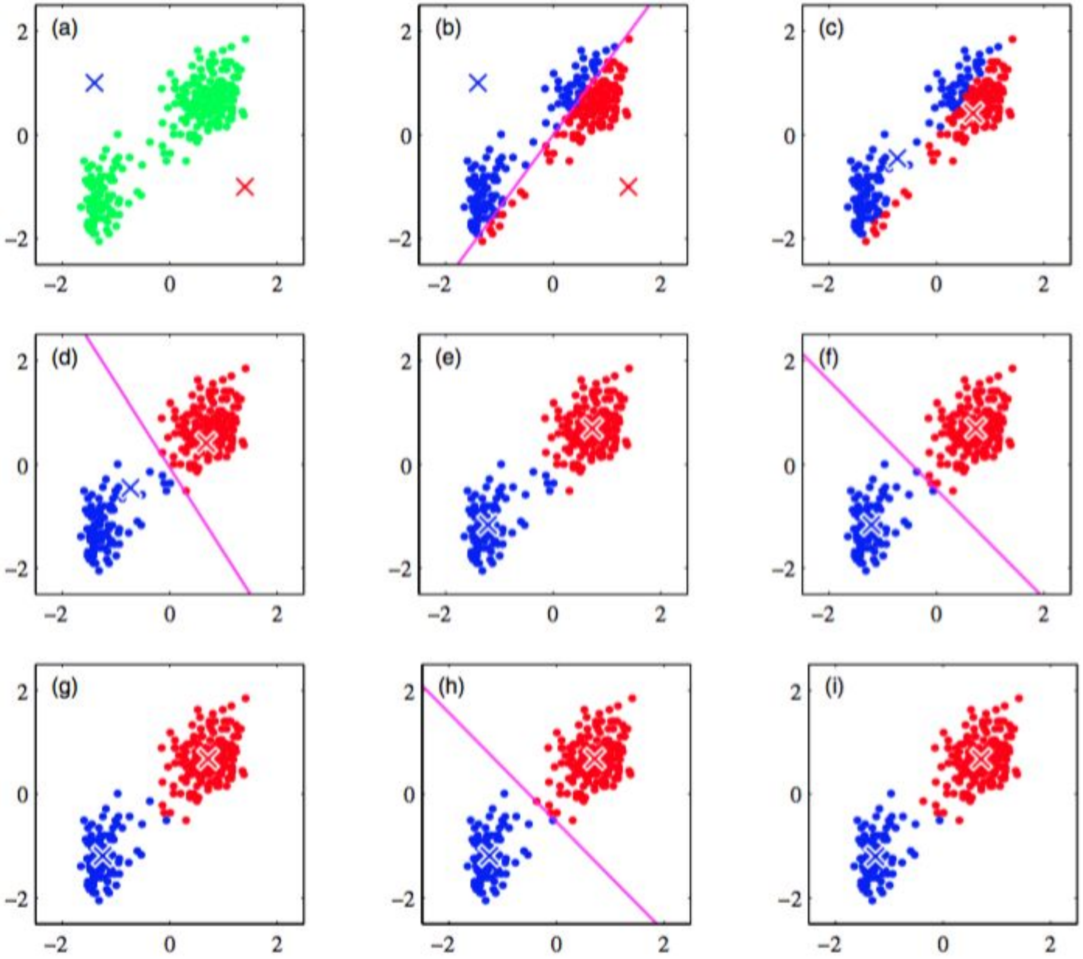


Figure 9.1
Bishop

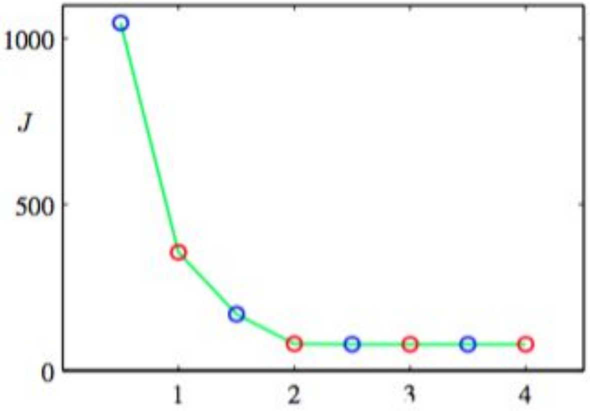


Figure 9.2
Bishop

k-means Optimization

Find cluster centers m and assignments r to ***minimize the sum of squared distances*** of data points $\{x^{(n)}\}$ to their assigned cluster centers

$$\min_{\{\mathbf{m}\},\{\mathbf{r}\}} J(\{\mathbf{m}\},\{\mathbf{r}\}) = \min_{\{\mathbf{m}\},\{\mathbf{r}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

$$\text{s.t.} \sum_k r_k^{(n)} = 1, \forall n, \quad \text{where} \quad r_k^{(n)} \in \{0,1\}, \forall k, n$$

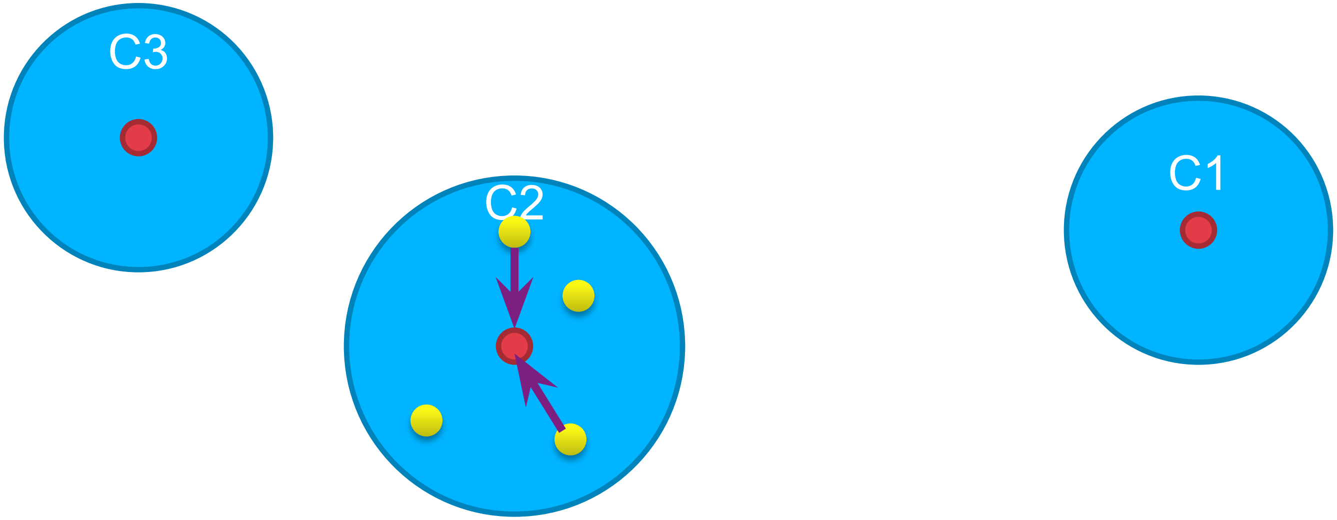
where $r_k^{(n)} = 1$ means that $x^{(n)}$ is assigned to cluster k (with center m_k)

k-means Algorithm

- k is a hyper-parameter: input to the algorithm. User specifies it.
- How to select K?
  - Known (e.g. the goal is to find 2 clusters representing genders)
  - Data-driven:
    - inertia
    - inertia/inertia2
    - silhouette

Inertia

- **Inertia** = Sum of squared distance between each instance x^i and its closest centroid (the center of the cluster to which the point is assigned.)
- It is also called "**within-cluster sum-of-squares**"
- Referred to as "*cost* of a specific clustering"
- K-Means uses inertia to define which model is the best



Only the cluster we assigned the point to is important for Inertia

Inertia Cont’d

- The lower the inertia, the better (be careful as it decreases
- when k increases)
- Thus, K-means minimizes inertia during training

Inertia = $\sum_{i=0}^{n} ||x^i - m^i||^2$, m is cluster centroid

|| x - m || => Euclidean distance between x and m

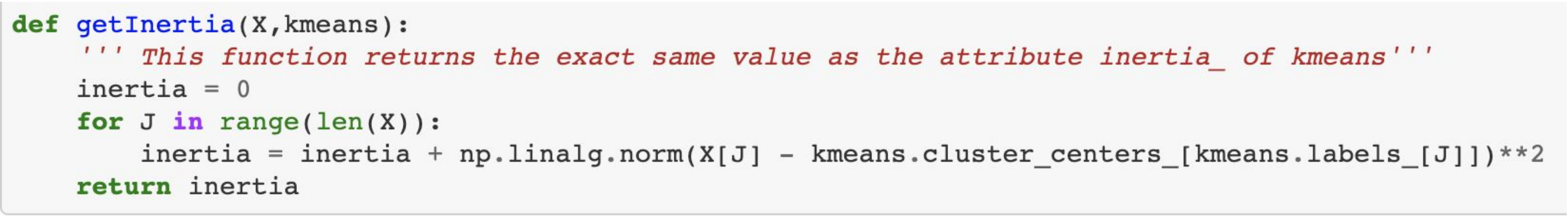
|| x - m || = $\sqrt{\sum_{j=1}^{j}(X_j - m_j)^2}$ $\quad$ j = Number of features for each X

- **Small inertia** means points are *closer* to each other in the cluster itself and **large inertia** means points are *further* from each other in the cluster.
- Inertia can be inflated with large number of features. Feature reduction is useful to alleviate the issue

Inertia Calculation in Python

Inertia = Sum of squared distance between each instance x^i and its closest centroid (the center of the cluster to which the point is assigned.)

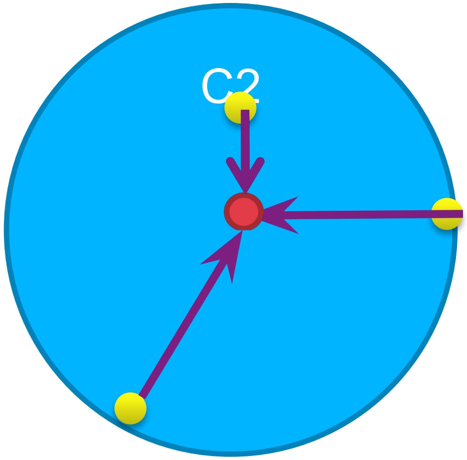
$$\text{Inertia} = \sum_{i=0}^{n} ||x^i - m^i||^2$$

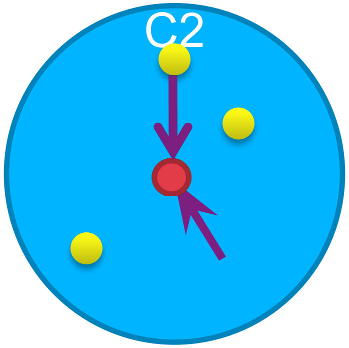


```python
def getInertia(X,kmeans):
    ''' This function returns the exact same value as the attribute inertia_ of kmeans'''
    inertia = 0
    for J in range(len(X)):
        inertia = inertia + np.linalg.norm(X[J] - kmeans.cluster_centers_[kmeans.labels_[J]])**2
    return inertia
```

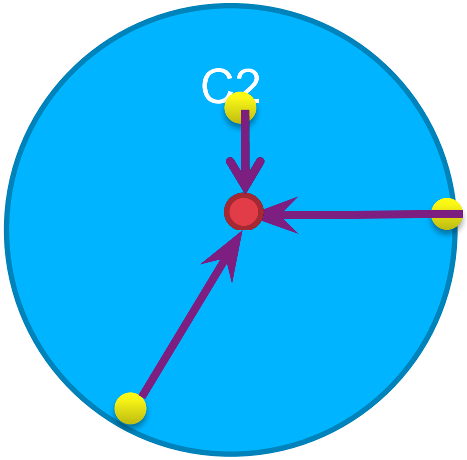
Small vs Large Inertia

Question: Which cluster has bigger inertia?

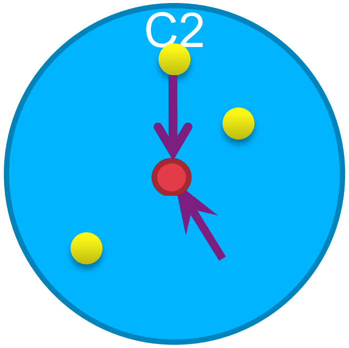




Small vs Large Inertia

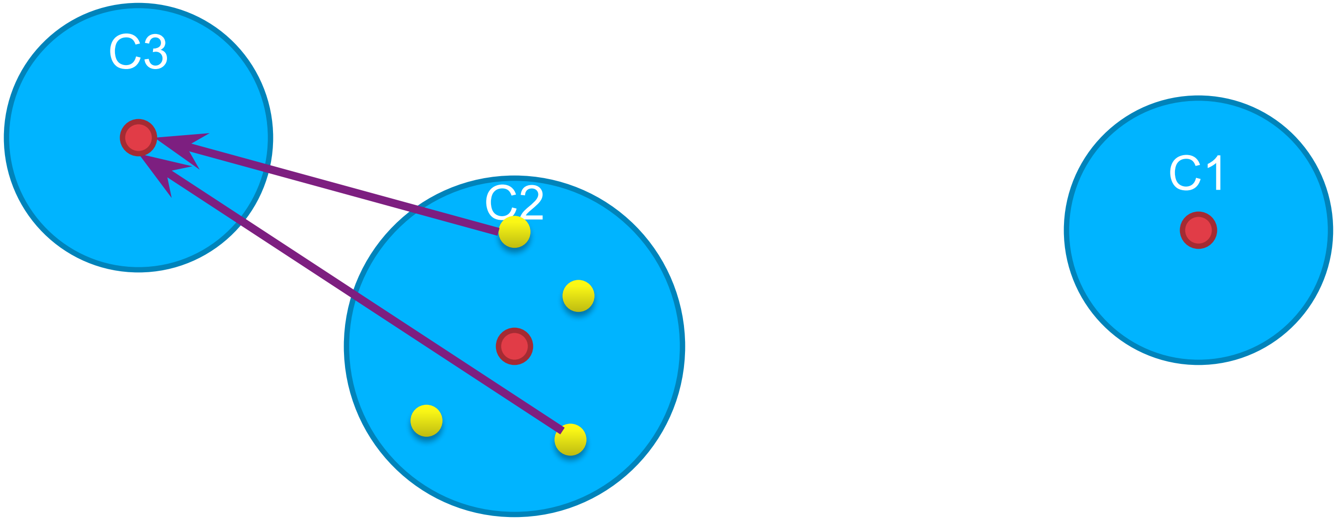


Larger Inertia



Inertia 2

- Sum of the squared distances between each point and the 2nd closest cluster
- **Between clusters sum of square**
- A nice clustering solution should have small inertia, and large inertia2:
  - points are close to the center of their cluster
  - points are far from the center of the other cluster



Silhouette

- Ratio between *inertia* and *inertia2*.

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

a: mean distance to the other instances in the same cluster

b: mean distance to the other instance in the nearest-cluster

- Between [-1, +1]
  - **Silhouette close to +1 means**: instance is in good cluster and far from other clusters (b >> a)
  - **Silhouette equals 0 means:** All instances are in one cluster
  - **Silhouette close to 0 means:** it is close to a cluster boundary
  - **Silhouette close to -1 means:** instance may be in the wrong cluster

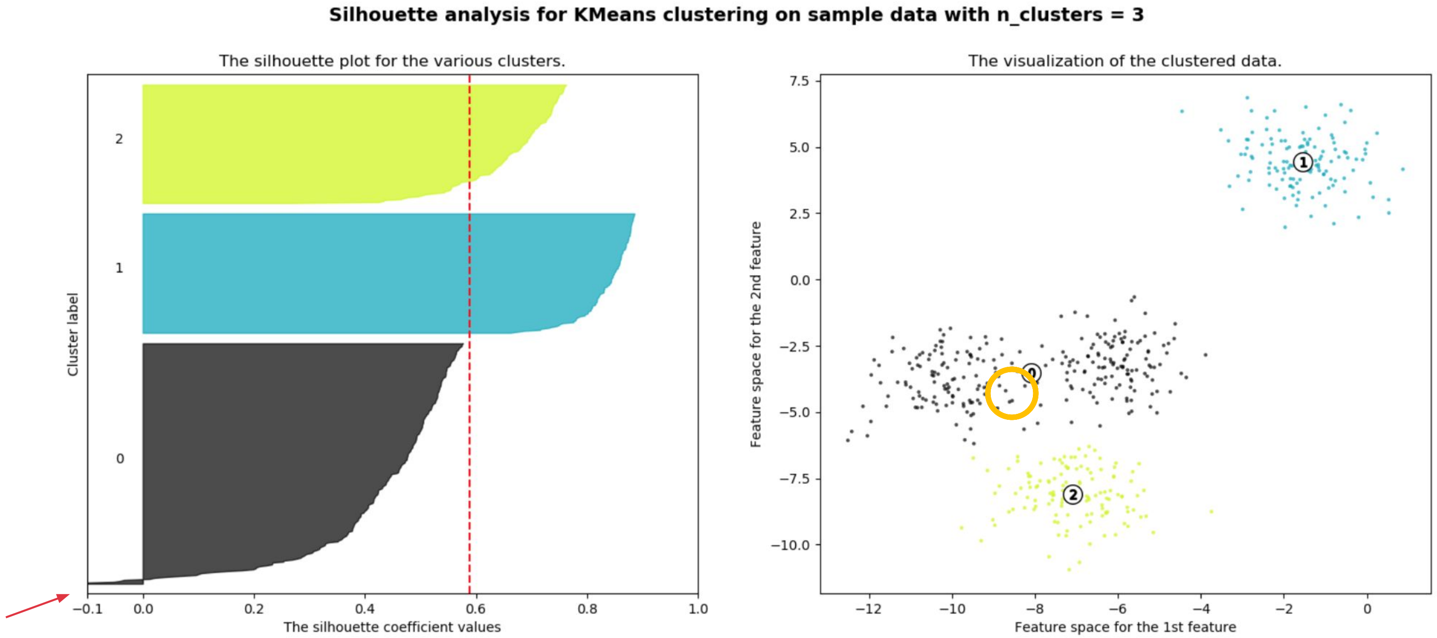
From sklearn.metrics import silhouette_score
Silhouette(X, kmeans.labels_)

Clustering Evaluation

-Silhouette score is calculated for every instance in the dataset and reported as average

Silhouette near +1 indicate the sample is far away from the neighboring clusters 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster

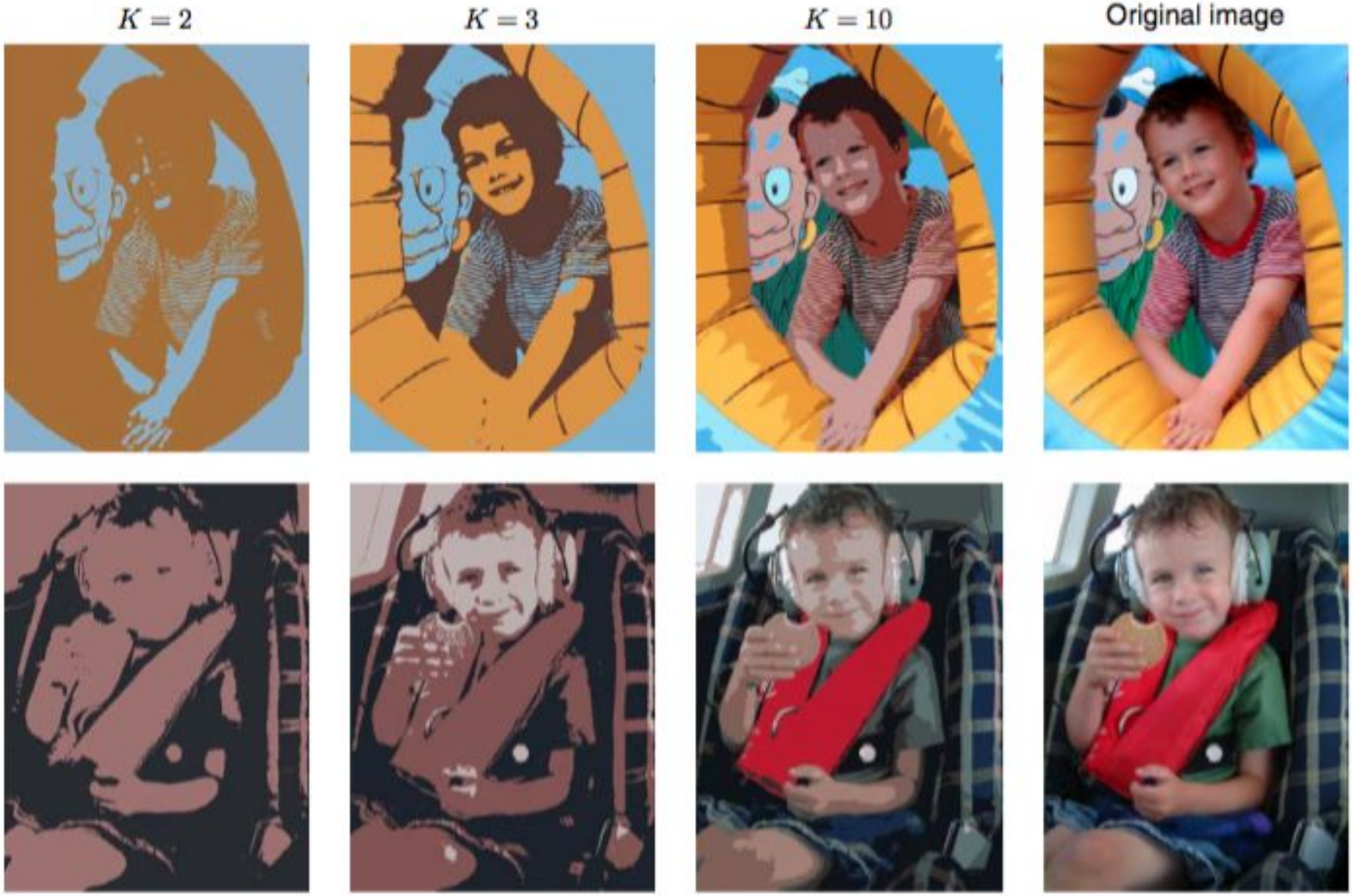
Red line at Silhouette = 0.6 is where the value is at peak for all clusters.



Negative value indicates wrong cluster

Example of cluster analysis here

k-means for Image Segmentation



$K = 2$ $K = 3$ $K = 10$ Original image

k-means Challenges

- High-dimensional spaces look different:
  - Almost all pairs of points are at about the same distance
- There is nothing to prevent k-means getting stuck at local minima.

K-means++

- Proposed in 2006 paper*
- Main idea is to spend sometimes to optimize initial value of centroids
- **Optimal centroids should be distant from one another**
- Sklearn by default uses K-means++

KMeans(*n_clusters=8*, *, *init='k-means++'*, *n_init=10*, *max_iter=300*, *tol=0.0001*, *precompute_distances='deprecated'*, *verbose=0*, *random_state=None*, *copy_x=True*, *n_jobs='deprecated'*, *algorithm='auto'*)

init*{‘k-means++’, ‘random’}, default=’k-means++’*

* Arthur, D.; Vassilvitskii, S. (2007). "*k*-means++: the advantages of careful seeding" (PDF). *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.}

Module 4 – Section 5

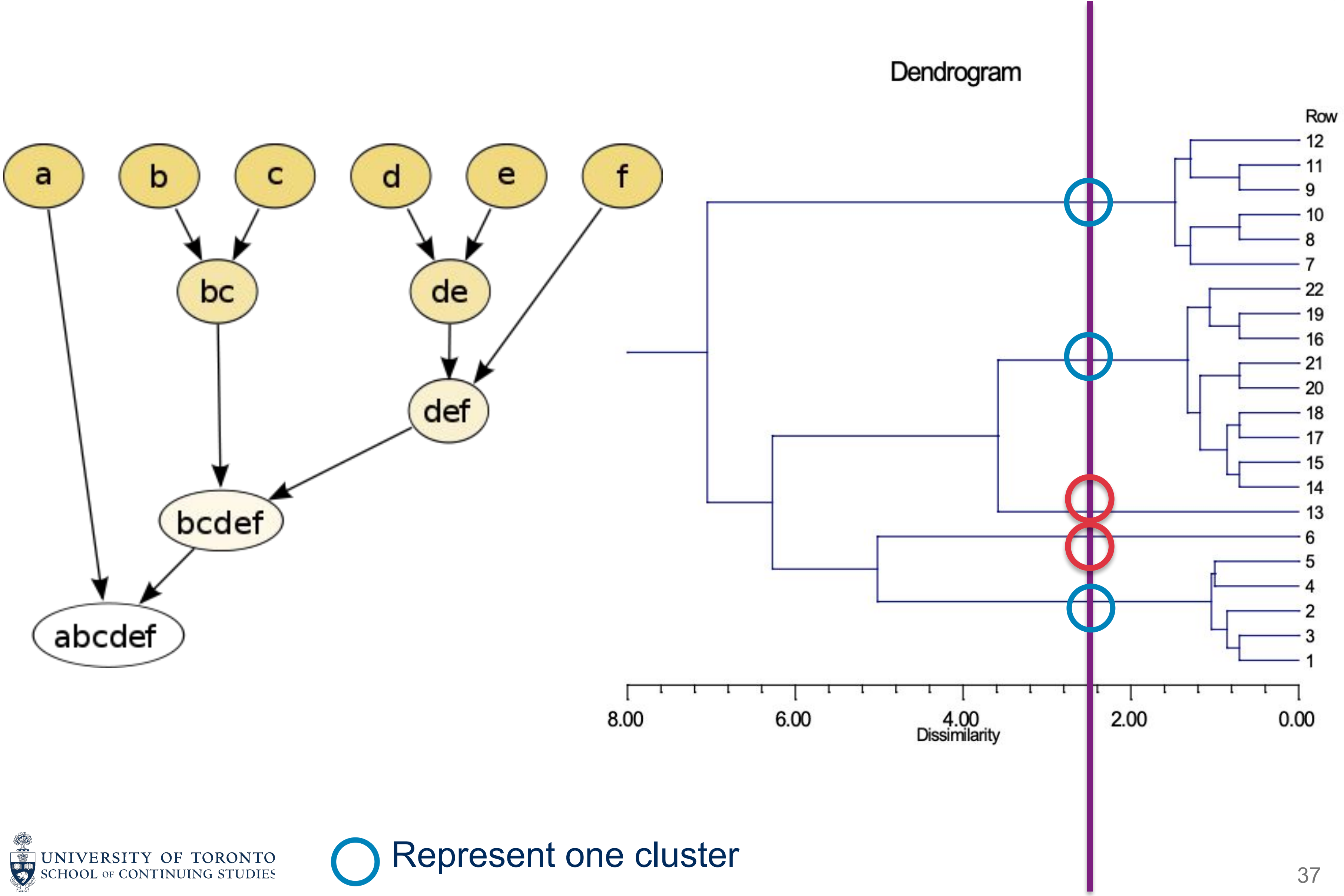
Hierarchical Clustering

Hierarchical Clustering

- Agglomerative clustering is a bottom-up hierarchical clustering technique
- It starts with as many clusters as points, and merges them iteratively
- **Dendrograms** cannot tell you how many clusters you should have
- Steps:
  0) Make each data point a distinct cluster
  1) Find the two closest clusters and merge them
  2) Repeat (1) until all points belong to one single cluster

Hierarchical Clustering

Using the purple line, you can specify your clusters and acceptable dissimilarity



○ Represent one cluster

Hierarchical Clustering (cont'd)

- Key operation: Repeatedly combine two nearest clusters
- How to represent a cluster of many points?
  - Key problem: As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
  - Euclidean case: each cluster has a centroid = average of its (data) points
- How to determine "nearness" of clusters?
  - Measure cluster distances by distances of centroids

Hierarchical Clustering Merge Method

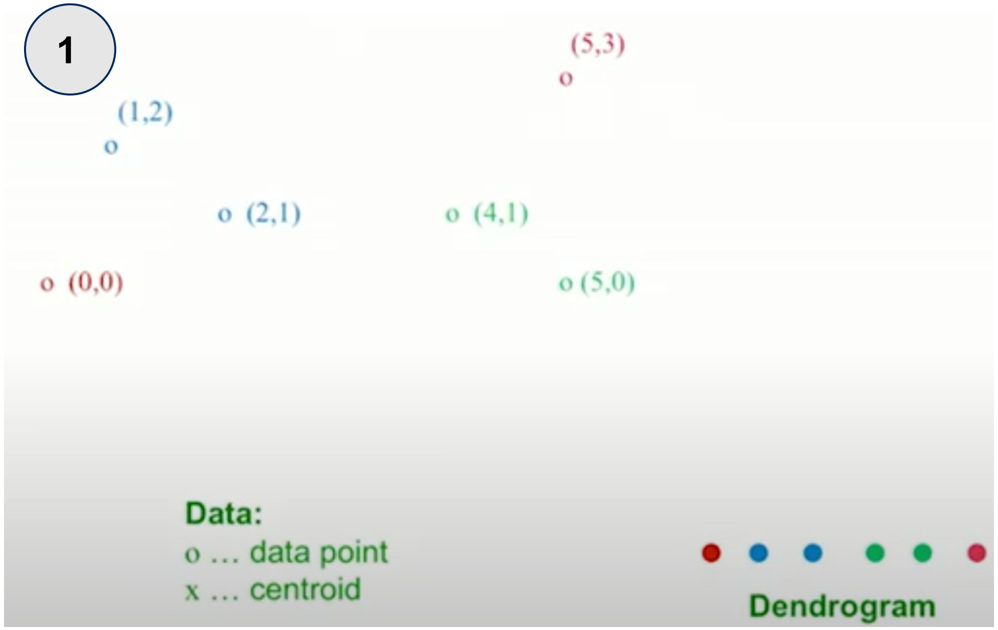
- There are different ways to determine the 2 clusters to be merged in each step:
  - **Ward's method**: minimize the total **within-cluster** variance
  - **Average linkage**: minimize the distance between each pair of observations in each cluster.
  - **Complete linkage**: minimize maximum distance between a pair of points, one in each cluster

- **Average-linkage** and **complete-linkage** are the two most popular distance metrics in hierarchical clustering.

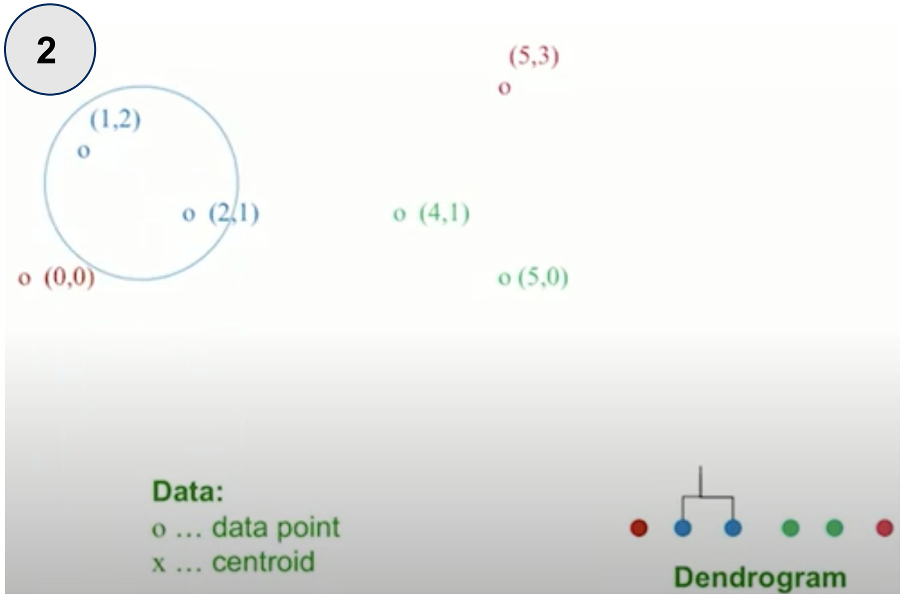
- The user decides the number of clusters to use
- Refer to this Colab Notebook

Hierarchical Clustering Example

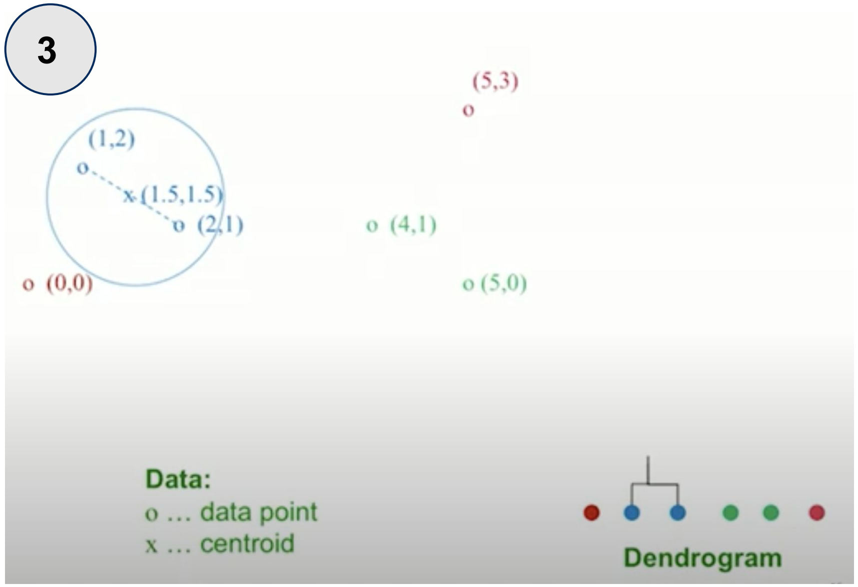
1



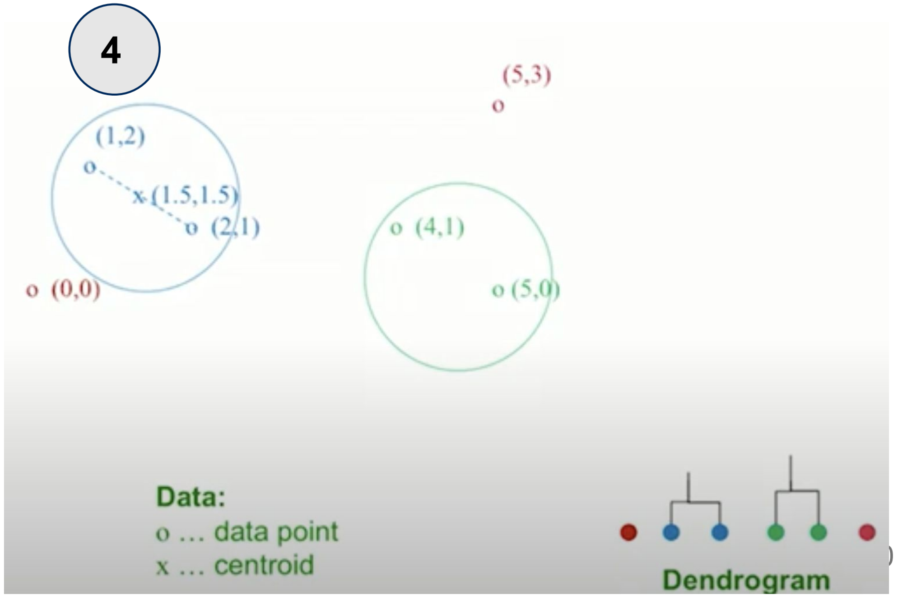
2



3

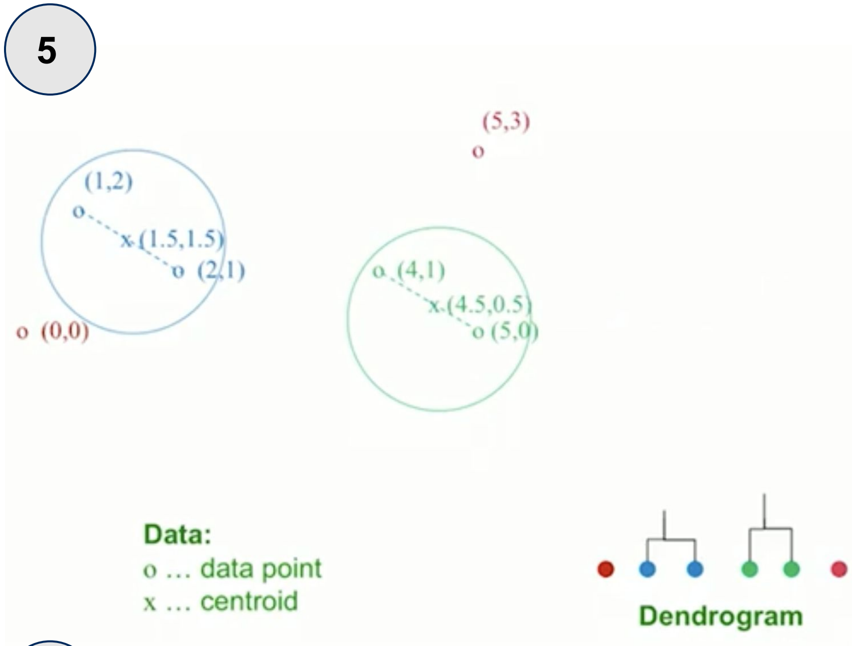


4



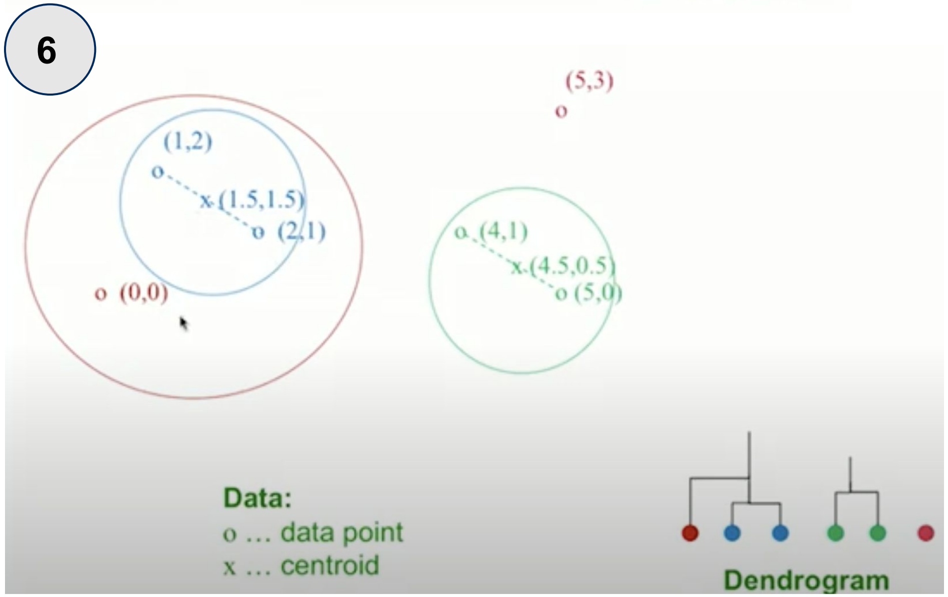
Hierarchical Clustering Example

5



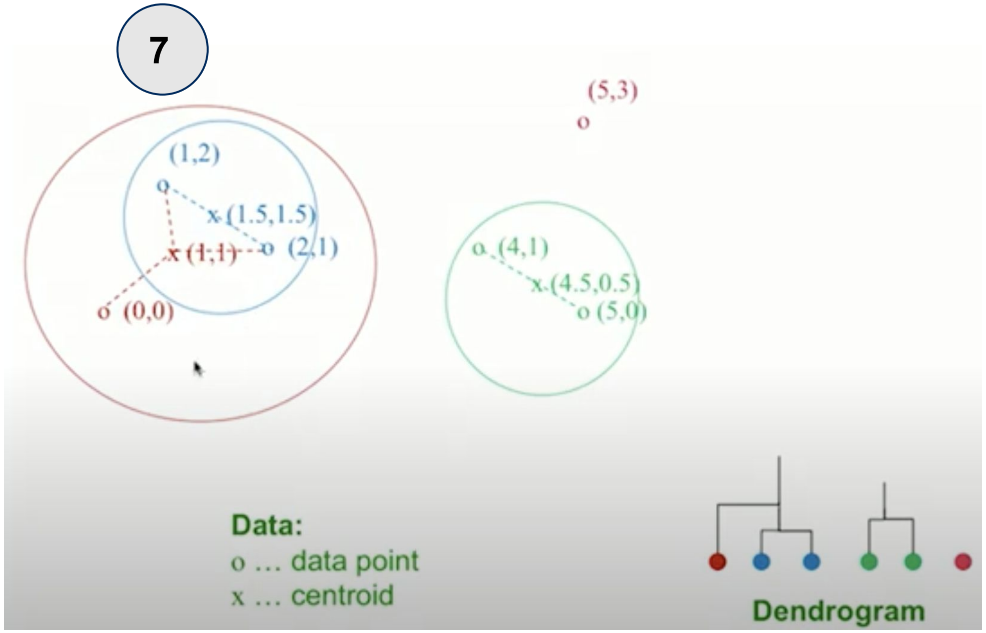
Data:
o … data point
x … centroid

6



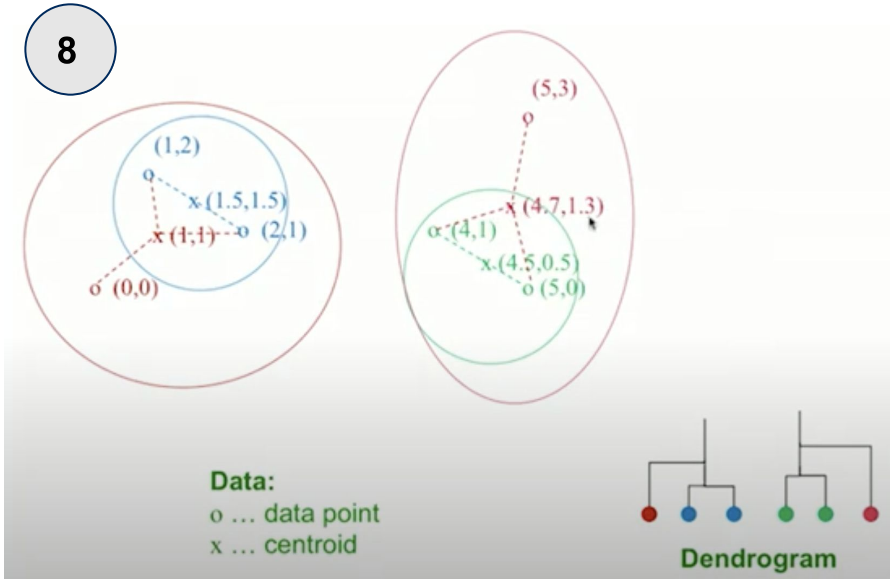
Data:
o … data point
x … centroid

7



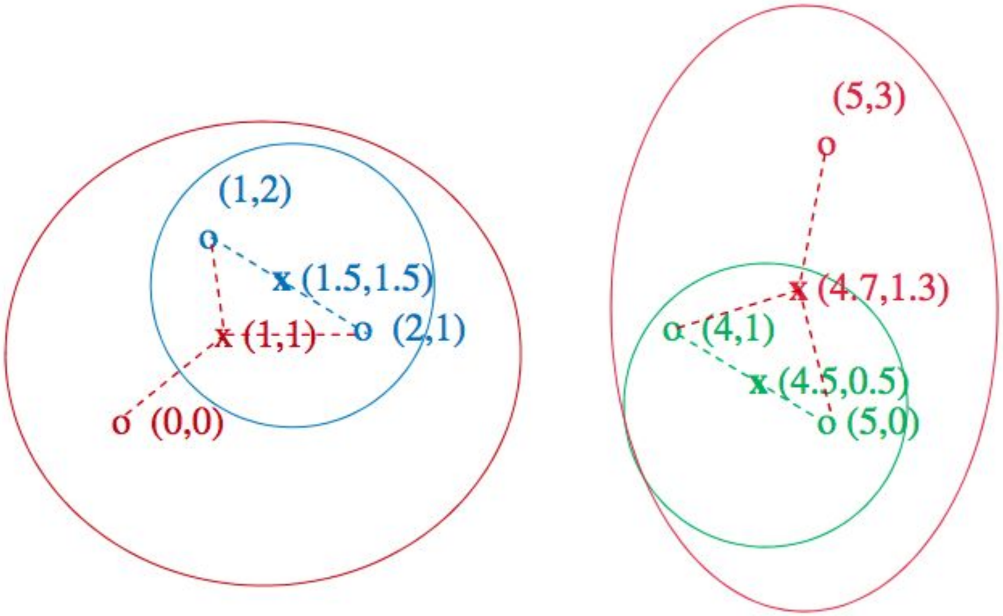
Data:
o … data point
x … centroid

8

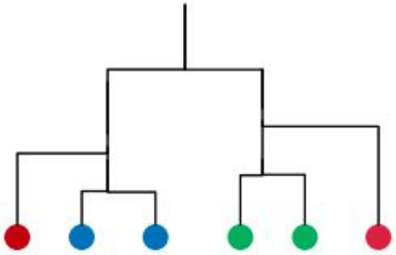


Data:
o … data point
x … centroid

Hierarchical Clustering Example



Data:
o … data point
x … centroid



Dendrogram

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Trivia Questions

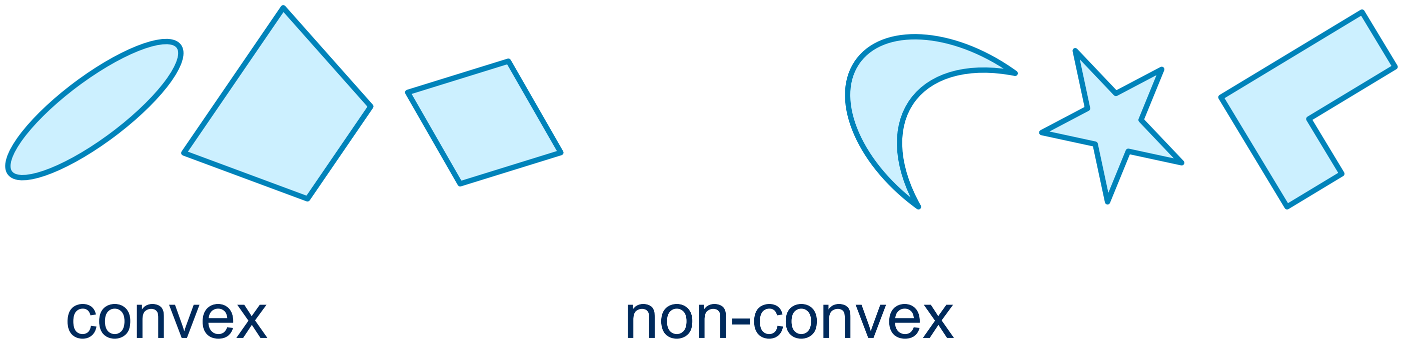
https://pollev.com/saeidabolfazli600

Module 4 – Section 4

DBSCAN Clustering

DBSCAN Clustering

- Density-based spatial clustering of applications with noise
- k-means clusters tend to be delimited by convex regions



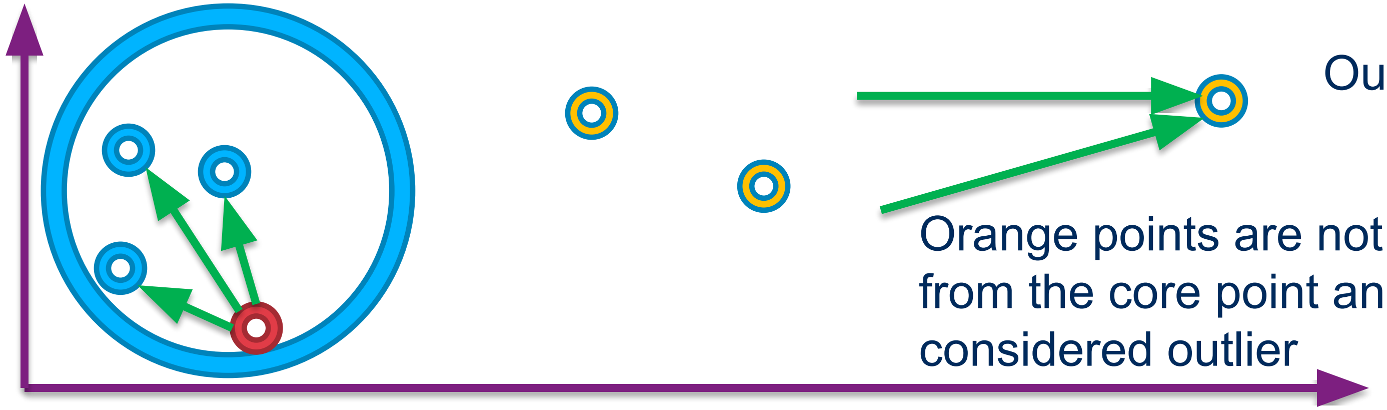
- Both k-means and hierarchical clusters assign a cluster to every point
  - outliers are forced to belong to a cluster
- Number of clusters is **NOT** decided by the user

DBSCAN Clustering (cont'd)

- DBSCAN is an algorithm that allows:
  - clusters with non-convex shapes
  - **outlier** detection: If x belongs to C1, x should be close to lots of other x in C1, unless x is outlier
- Other algorithms allow non-convex shaped clusters:
  - agglomerative with ward linkage
  - spectral clustering
- Parameters:
  - *min_samples* (non-negative integer)
  - *epsilon* (positive number)

How DBSCAN works?

- Uses the parameters provided by users to identify **core points**
- Core points are member of clusters
- At any point in time during training, it randomly picks a point and identify if it is core point or not
- If it is, it adds the core point to the cluster, otherwise ignores
- A **core point** is a point that has at least *min_samples* points within *epsilon* distance. Min_sample below is 4



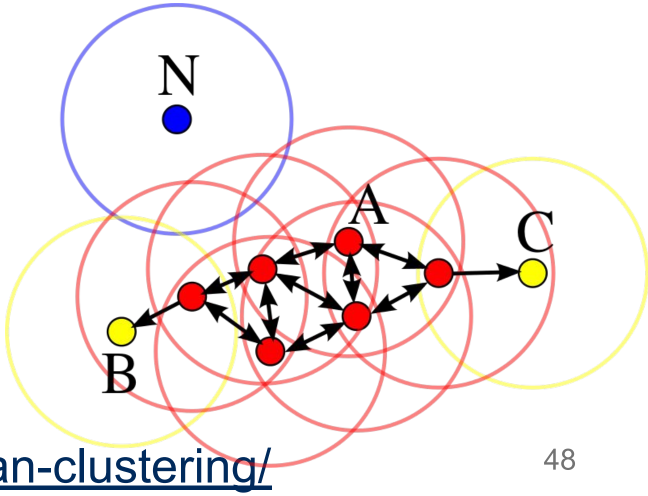
Outlier

Orange points are not reachable from the core point and hence are considered outlier

How to detect outliers?

- Core points are determined first
- Core points belonging to a cluster are computed iteratively:
  - take a core point
  - find all core points within *epsilon* distance
  - repeat until no more core points exist within *epsilon*
  - continue creating other clusters until no core points exists
- Non-core points:
  - Add to each cluster non-core points within *epsilon* distance from a core point
- Points that do not belong to any cluster are outliers

In this example, min-Pts = 4. Red Points are core, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself).
Because they are all reachable from one another, they form a single cluster. Points B and C are not core but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable



Demo: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Module 4 – Section 6

Visual presentation of Clusters

How would business see it?

- As clustering is unsupervised learning, there is no lable and hence there is no way we can name any cluster, except cluster 1, cluster 2, or cluster 3
- Often business needs to understand clusters and how they are different from each other so that they can take relevant action(s)
- For instance:

  cluster 1= “Tech savvy teens”

  cluster 2= “Yonge wealthy & educated families”

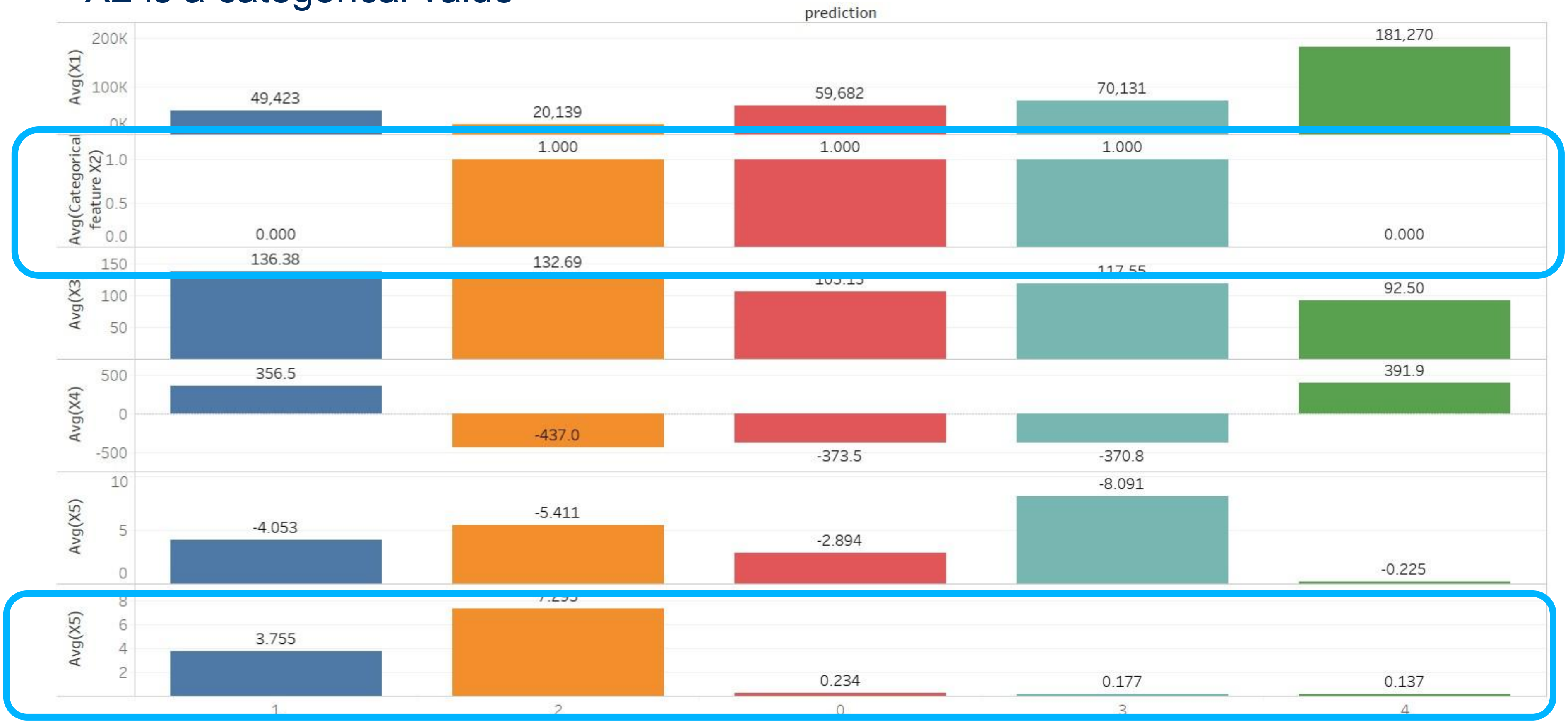
  cluster 3= “Senior citizen without kids”
- In marketing, presenting certain product/service to the customers in each of above cluster is differentiating factor

How to simplify communication to business?

- Labeling clusters is often a joint activity with **business prime.**
- Data scientist walks business through the results and they collectively label the clusters based on the expected action as well as clustering features
- Plotting the clusters against different attributes/features is a useful tool to share the results with business in an intuitive manner and drive action

Example with categorical features

- Clusters represent bars and rows are features used for clustering
- Each label is the average value of the feature for respective cluster
- Some features are good to separate clusters well, like X5 or X2 and some clusters not
- X2 is a categorical value



Module 4 – Section 7

Resources and Wrap-up

Resources

- Clustering: http://scikit-learn.org/stable/modules/clustering.html
- Data Science from Scratch, Joel Grus
- An Introduction to Statistical Learning, James, G.; Witten, D.; Hastie, T.; Tibshirani, R

Next Class

- Training Models and Features Selection
- Reading Hands-on ML (Chapter 4)

Follow us on social

Join the conversation with us online:

- facebook.com/uoftscs
- @uoftscs
- linkedin.com/company/university-of-toronto-school-of-continuing-studies
- @uoftscs

Any questions?

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Thank You

Thank you for choosing the University of Toronto School of Continuing Studies