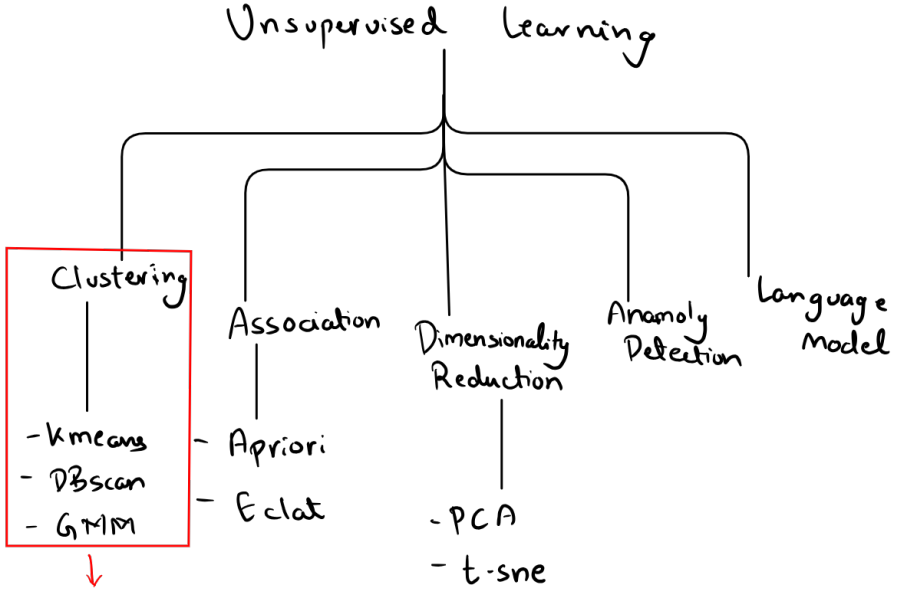
K-Means

Vraj Patel

Types of Machine learning

- Supervised ML
- Unsupervised ML
- Reinforcement Learning



Applications of Clustering

- Customer Segmentation
- Data Analysis
- Semi supervised Learning
- Image Segmentation

K-Means Clustering

- An unsupervised algorithm that groups data into k clusters by assigning each point to the nearest cluster center (centroid) and then updating those centroids to be the mean of their assigned points, repeating until stable.

Geometric Intuition

1. Decide K-clusters – We have to decide this as an input
2. Initialize centroids
3. Start Iterating

   Assign clusters

   Move Centroid

   Check and stop

The nearest centroid to any point is calculated using Euclidean distance and the clusters are re assigned and centroids are moved until the last and current centroid locations are not same

Elbow Method

How do we decide the correct number of clusters (k)?

Inertia / Wcss

Sum of squared distances between each data point and its assigned cluster centroid, summed over all clusters.

Limitations of Elbow Method

Subjectivity in Identifying the Elbow: The biggest challenge with the elbow method is the subjective nature of identifying the "elbow" point. The point where the inertia starts decreasing at a slower rate can be open to interpretation and may not be clear-cut, especially in datasets where the decrease in inertia is gradual.

Not Suitable for All Datasets: The method does not work well if the data is not very clustered or if the clusters have an irregular shape. In such cases, the elbow might not be distinct, leading to ambiguity in choosing the right number of clusters.

Performance with Large Number of Features: The elbow method can become less effective as the number of features in the dataset increases. High-dimensional data can make the identification of a clear elbow more difficult.

Doesn't Consider Cluster Quality: The elbow method focuses solely on the variance within the clusters and does not take into account the quality of the clusters formed. It's possible to choose a k where clusters are not meaningful or well-separated.

Sensitivity to Scaling: Like K-means clustering itself, the results of the elbow method can be sensitive to the scale of the data. Features with larger scales can dominate the result, potentially leading to suboptimal choices of k.

Assumptions of KMeans

Spherical Cluster Shape: K-means assumes that the clusters are spherical and isotropic, meaning they are uniform in all directions. Consequently, the algorithm works best when the actual clusters in the data are circular (in 2D) or spherical (in higher dimensions).

Similar Cluster Size: The algorithm tends to perform better when all clusters are of approximately the same size. If one cluster is much larger than others, K-means might struggle to correctly assign the points to the appropriate cluster.

Equal Variance of Clusters: K-means assumes that all clusters have similar variance. The algorithm uses the Euclidean distance metric, which can bias the clustering towards clusters with lower variance.

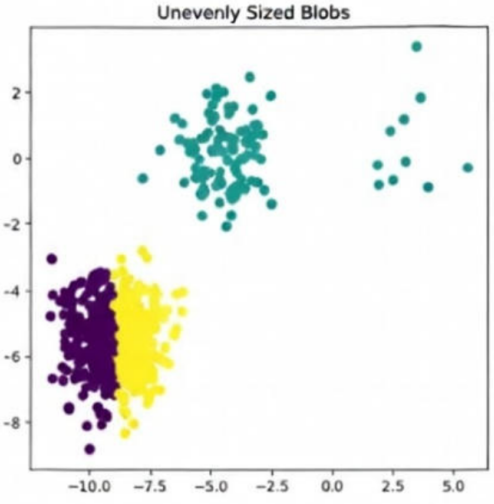
Clusters are Well Separated: The algorithm works best when the clusters are well separated from each other. If clusters are overlapping or intertwined, K-means might not be able to distinguish them effectively.

Number of Clusters (k) is Predefined: K-means requires the number of clusters (k) to be specified in advance. Choosing the right value of k is crucial, but it is not always straightforward and typically requires domain knowledge or additional methods like the Elbow method or Silhouette analysis.

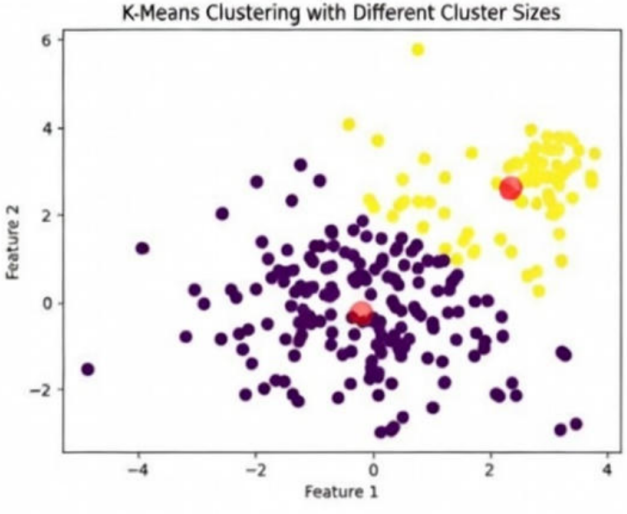
Large n, Small k: K-means is generally more efficient and effective when the dataset is large (large n) and the number of clusters is small (small k).

Limitations of KMeans

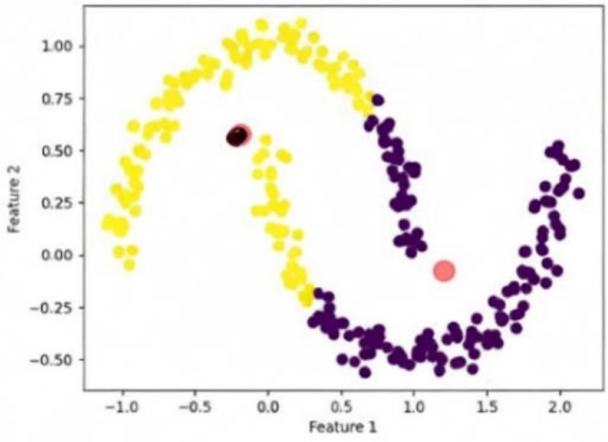
- ✓ **Number of Clusters**: Determining the optimal number of clusters (k) is not straightforward and often requires domain knowledge or methods like the elbow method.
- ✓ **Requires clusters of similar sizes**: Kmeans requires the clusters to be of similar sizes.



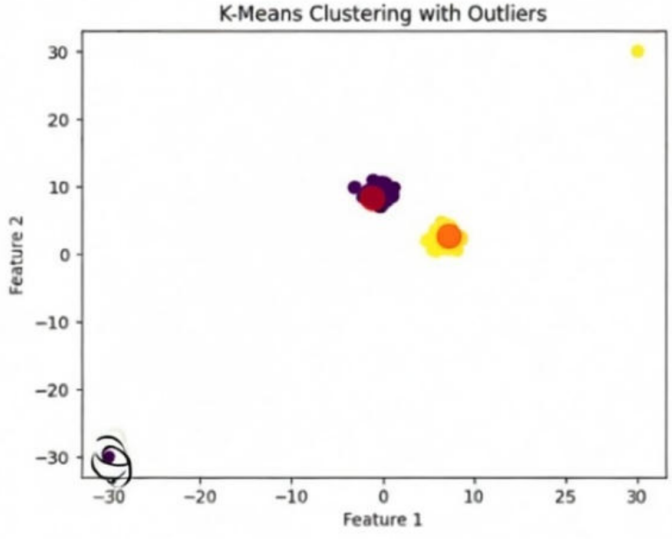
Similar variance between clusters: Kmeans requires the clusters to be of similar variance.



- ✓ **Assumption of Spherical Clusters**: KMeans assumes that clusters are spherical and of similar size, which might not be the case in real-world data.



- ✓ **Vulnerability to Outliers**: Outliers can significantly distort the mean value of a cluster, leading to misleading results.



- ✓ **Hard Clustering**: Each data point is forced into exactly one cluster, which may not be suitable for all applications, especially where data can belong to multiple clusters.
- ✓ **High-Dimensional Challenges**: In very high-dimensional spaces, the distance between data points can become less meaningful, affecting the performance of KMeans.
- ✓ **Sensitive to Scale**: The measure is sensitive to the scale of the features. Hence, feature scaling (like standardization) is often recommended before applying K-means.

Silhouette Score

Cohesion

Definition: Cohesion refers to the degree to which elements within the same cluster are close to each other. It measures how tightly grouped the data points in a cluster are.

Ideal Scenario: High cohesion means that data points in a cluster are similar or near to each other, indicating a good clustering where each cluster is distinct and meaningful.

Measurement: Cohesion can be quantified using metrics such as the sum of squared distances of data points from their respective cluster centroids. In K-Means, this is often referred to as inertia or within-cluster sum of squares.

Separation

Definition: Separation, on the other hand, refers to how distinct or well-parted different clusters are from each other. It measures the extent to which clusters are different or distant from each other.

Ideal Scenario: High separation means that clusters are well-differentiated and far apart, indicating that the algorithm has done a good job in distinguishing between different groups in the data.

Measurement: Separation can be quantified by metrics such as the distance between cluster centroids, or more complex measures like the silhouette score, which considers both cohesion and separation.

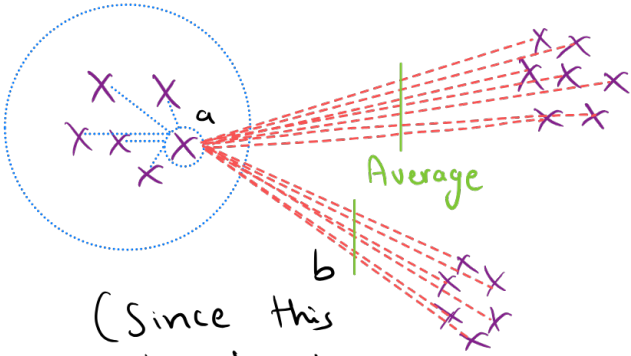
Silhouette Score

The silhouette score is a measure used to assess the quality of clusters created by a clustering algorithm. It provides a succinct graphical representation of how well each data point lies within its cluster, which is a combination of both cohesion and separation. The value of the silhouette score ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

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

Here a is the average distance of one point to all the other points within the same cluster

b is the average distance of all points in the nearest cluster



(Since this is closest cluster)

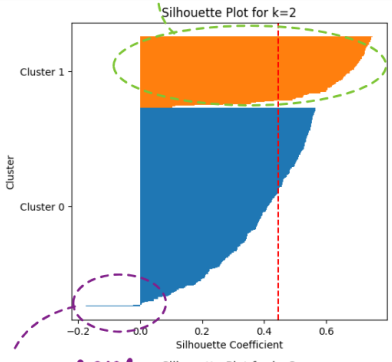
Interpretation:

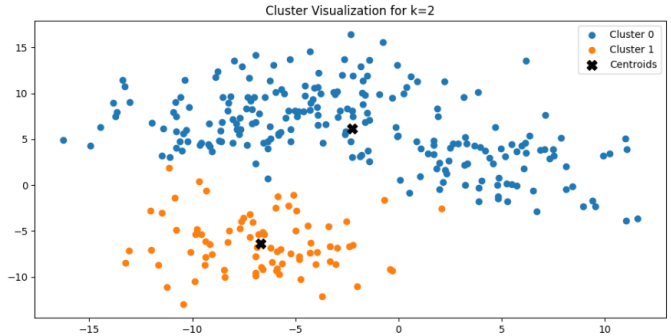
Close to +1 : Indicates that the data point is far away from the neighbouring clusters

Close to 0: Indicates that the data point is on or very close to the decision boundary btwn 2 neighbouring clusters.

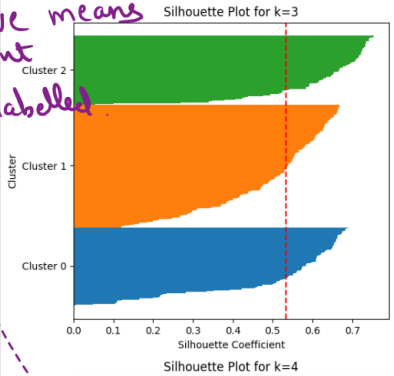
Close to -1 : Indicates that the data point may have been assigned to the wrong cluster

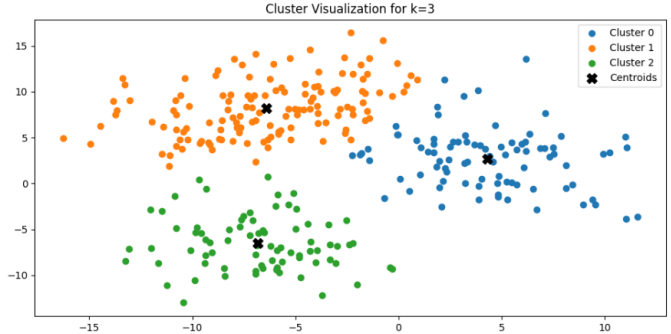
This is silhouette score for each data point

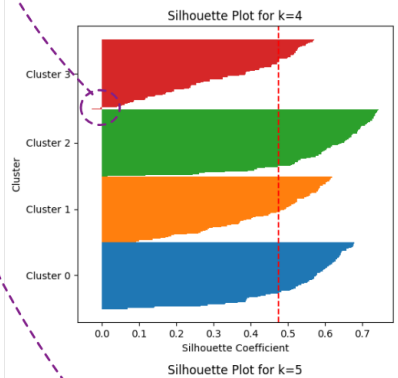


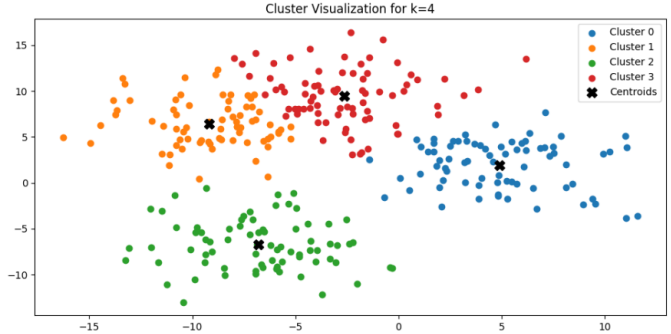


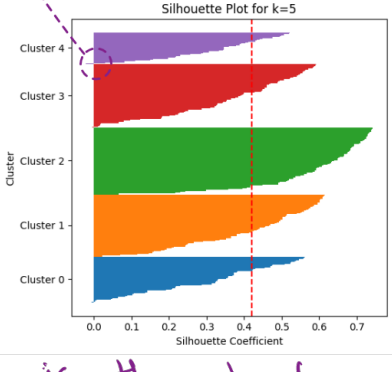
This -ve means that point is mislabelled

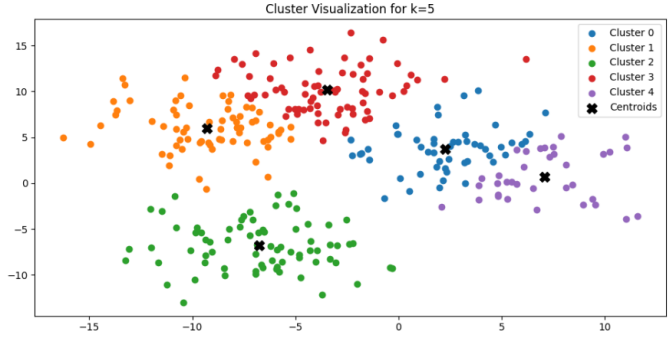












K=3 is the best value since there's no mislabelling

Hyper parameters

> *class* **sklearn.cluster.KMeans**(*n_clusters=8, *, init='k-means++', n_init='warn', max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd'*)

Number of Clusters (k):

Description: This is the number of clusters you want the algorithm to form, as well as the number of centroids to generate.
Importance: Choosing the right number of clusters is crucial as it significantly influences the clustering results. Too many clusters can overfit the data, while too few can miss important patterns.

Initialization Method:

Description: This parameter specifies the method for initializing the centroids. Common methods include 'random' (randomly choosing k data points as initial centroids) and 'k-means++' (a smarter way of initializing centroids to improve convergence).
Importance: Good initialization can lead to faster convergence and better clustering. K-means++ is generally preferred over random initialization.

Number of Initialization Runs (n_init):

Description: This is the number of times the KMeans algorithm will be run with different centroid initializations. The final results will be the best output of n_init consecutive runs in terms of inertia.
Importance: Multiple initializations can prevent the algorithm from falling into sub-optimal solutions, but increase computational cost.

Maximum Number of Iterations (max_iter):

Description: The maximum number of iterations the algorithm will run for each initialization.
Importance: A higher number of iterations allows more time for convergence but increases computational time. Usually, KMeans converges well before reaching the maximum number of iterations.

Tolerance (tol):

Description: This is the tolerance to declare convergence. If the centroids do not move significantly (as defined by this parameter) in consecutive iterations, the algorithm stops.
Importance: A smaller tolerance can lead to a more precise solution, but might increase computation time. A larger tolerance might speed up the algorithm but can lead to less precise clustering.

Algorithm:

Description: The computational algorithm to use. Options typically include 'auto', 'full', or 'Elkan'. 'Full' corresponds to the classic EM-style algorithm, while 'elkan' is an optimized variant that is generally more efficient.

KMeans++ is an algorithm for choosing the initial values (or "seeds") for the KMeans clustering algorithm. The standard KMeans algorithm is sensitive to the initial starting points (centroids), and KMeans++ provides a way to overcome this problem by specifying a procedure to initialize the centroids before proceeding with the standard KMeans iterative algorithm.

Here's a simplified overview of how KMeans++ works:

Initial Centroid Selection: The first centroid is chosen uniformly at random from the data points that are being clustered.

Distance Calculation: Calculate the distance of each data point from the nearest, previously chosen centroid.

Probabilistic Selection of Next Centroids: Choose the next centroid from the data points with a probability proportional to the square of the distance from the point to its nearest centroid. This step biases the algorithm to select data points that are far from the existing centroids.

Repeat Until K Centroids: Repeat steps 2 and 3 until k centroids have been chosen.

Proceed with Standard KMeans: Once the initial centroids are chosen, proceed with the standard KMeans clustering algorithm.

This method tends to spread out the initial centroids, which can lead to better clustering results compared to selecting the initial centroids randomly, as the standard KMeans algorithm does. By doing so, KMeans++ can often lead to faster convergence and better clustering.

The reason this algorithm (kmeans++) is used is because, alot of times, the initialization of initial centroid location matters alot, which might result in different clusters.
Therefore, this one makes sure that the new centroids selected are far from the current centroids.

Time and Space Complexity

Time Complexity : $O(n\,k\,d\,i)$

- n : no. of rows
- k : no. of clusters
- d : no. of dimensions
- i : no. of iterations

Space Complexity : $O(nd + kd)$

- In the RAM, we only need to store :
  1. The entire data.
  2. Centroids

Mini - Batch K-Means

Mini batch K-means is a variant of K-means that updates centroids using small random subsets (mini-batches) of the data each iteration, making clustering much faster and more memory-efficient on large datasets at the cost of a slight loss in cluster quality (accuracy).

Intuition:

- Standard K-means recomputes centroids using all data every iteration, which is expensive for web scale or streaming data.
- Mini batch K-means instead samples a small batch of points, assigns only those to clusters, and updates the corresponding centroids with a stochastic / gradient-like step, so each iteration is cheap and can be done with limited RAM.

ABSTRACT

We present two modifications to the popular k-means clustering algorithm to address the extreme requirements for latency, scalability, and sparsity encountered in user-facing web applications. First, we propose the use of mini-batch optimization for k-means clustering. This reduces computation cost by orders of magnitude compared to the classic batch algorithm while yielding significantly better solutions than online stochastic gradient descent. Second, we achieve sparsity with projected gradient descent, and give a fast ϵ-accurate projection onto the $L1$-ball. Source code is freely available: `http://code.google.com/p/sofia-ml`

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*

General Terms

Algorithms, Performance, Experimentation

Keywords

unsupervised clustering, scalability, sparse solutions

1. CLUSTERING AND THE WEB

Unsupervised clustering is an important task in a range of web-based applications, including grouping search results, near-duplicate detection, and news aggregation to name but a few. Lloyd's classic k-means algorithm remains a popular choice for real-world clustering tasks [6]. However, the standard batch algorithm is slow for large data sets. Even optimized batch k-means variants exploiting triangle inequality [3] cannot cheaply meet the latency needs of user-facing applications when clustering results on large data sets are required in a fraction of a second.

This paper proposes a mini-batch k-means variant that yields excellent clustering results with low computation cost on large data sets. We also give methods for learning sparse cluster centers that reduce storage and network cost.

2. MINI-BATCH K-MEANS

The k-means optimization problem is to find the set C of cluster centers $\mathbf{c} \in \mathbb{R}^m$, with $|C| = k$, to minimize over a set X of examples $\mathbf{x} \in \mathbb{R}^m$ the following objective function:

$$\min \sum_{\mathbf{x} \in X} ||f(C, \mathbf{x}) - \mathbf{x}||^2$$

Here, $f(C, \mathbf{x})$ returns the nearest cluster center $\mathbf{c} \in C$ to $\mathbf{x}$ using Euclidean distance. It is well known that although this problem is NP-hard in general, gradient descent methods converge to a local optimum when seeded with an initial set of k examples drawn uniformly at random from X [1].

The classic batch k-means algorithm is expensive for large data sets, requiring $O(kns)$ computation time where n is the number of examples and s is the maximum number of non-zero elements in any example vector. Bottou and Bengio proposed an online, stochastic gradient descent (SGD) variant that computed a gradient descent step on one example at a time [1]. While SGD converges quickly on large data sets, it finds lower quality solutions than the batch algorithm due to stochastic noise [1].

Algorithm 1 Mini-batch k-Means.

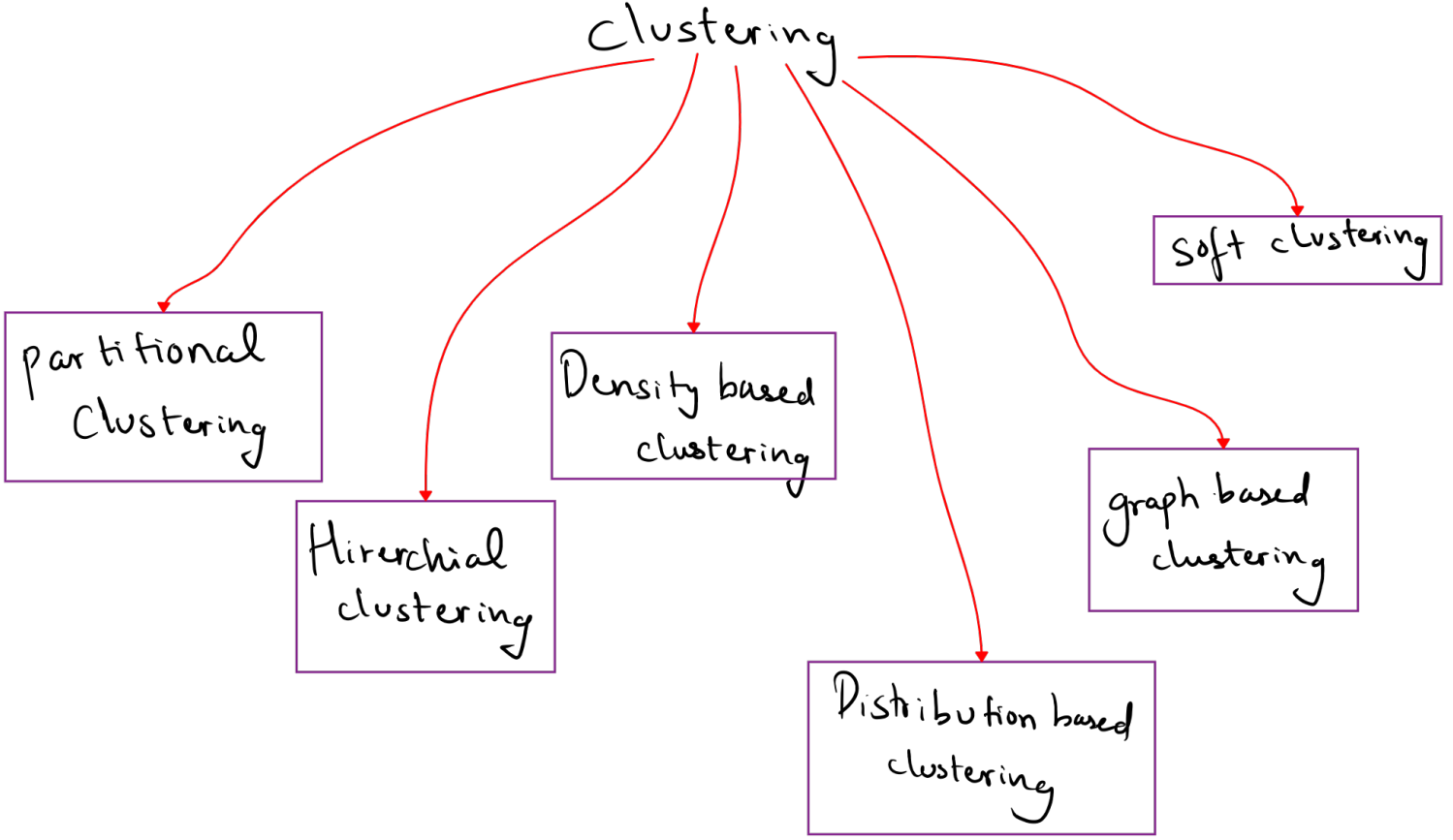
```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
    M ← b examples picked randomly from X
    for x ∈ M do
        d[x] ← f(C, x)    // Cache the center nearest to x
    end for
    for x ∈ M do
        c ← d[x]          // Get cached center for this x
        v[c] ← v[c] + 1   // Update per-center counts
        η ← 1/v[c]        // Get per-center learning rate
        c ← (1 − η)c + ηx // Take gradient step
    end for
end for
```

We propose the use of mini-batch optimization for k-means clustering, given in Algorithm 1. The motivation behind this method is that mini-batches tend to have lower stochastic noise than individual examples in SGD (allowing convergence to better solutions) but do not suffer increased computational cost when data sets grow large with redundant examples. We use per-center learning rates for fast convergence, in the manner of [1]; convergence properties follow closely from this prior result [1].

Experiments. We tested the mini-batch k-means against both Lloyd's batch k-means [6] and the SGD variant of [1]. We used the standard `RCV1` collection of documents [4] for

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

Mini batch k-means is simply used in scenarios where speed is top priority but it comes with trade in with a bit less accuracy.


Clustering
partitional Clustering
Hirerchial clustering
Density based clustering
Distribution based clustering
graph based clustering
soft clustering

Partitional Clustering

1. **Basic Concept**: Partitioning clustering algorithms divide a dataset into a set of non-overlapping subgroups or clusters, where each data point belongs to exactly one cluster.
2. **Examples**: The most famous partitioning clustering algorithm is k-means. It assigns data points to clusters in such a way that each point belongs to the cluster with the closest mean, which serves as a prototype of the cluster.
3. **Defining Number of Clusters**: A key requirement is pre-specifying the number of clusters (k). The selection of k significantly affects the outcome and quality of the clustering.
4. **Iterative Process**: These algorithms typically use an iterative refinement technique. For instance, in k-means, the process involves repeatedly assigning points to the nearest cluster centroid and then recalculating the centroids.
5. **Objective Function Optimization**: They aim to optimize an objective function, such as minimizing the total within-cluster variance or the sum of squared distances between data points and their respective cluster centroids.
6. **Suitability for Certain Data Shapes**: Partitioning methods are most effective when clusters are spherical or globular in shape. They assume homogeneity in cluster shapes and sizes.
7. **Sensitivity to Initial Conditions**: These algorithms can be sensitive to the initial starting conditions (like initial cluster centroids in k-means). Different initializations can lead to different clustering results.
8. **Handling of Outliers**: Partitioning algorithms can be influenced by outliers, as these can significantly skew the mean or centroid of a cluster.
9. **Scalability and Efficiency**: They are generally more scalable and efficient for larger datasets compared to hierarchical clustering, making them suitable for many practical applications.
10. **Use Cases and Limitations**: While widely used in various fields like market research, pattern recognition, and image processing, these algorithms have limitations in handling non-spherical clusters, varying cluster sizes, and noisy datasets. Advanced versions and variations of partitioning algorithms have been developed to address some of these

Hierarchical Clustering

1. **Nature of Clustering**: Hierarchical clustering builds a hierarchy of clusters either by successively merging smaller clusters into larger ones (agglomerative approach) or by successively splitting larger clusters into smaller ones (divisive approach).
2. **No Need to Specify Number of Clusters**: Unlike partitioning algorithms like k-means, hierarchical clustering does not require pre-specifying the number of clusters. The number of clusters can be determined by analysing the dendrogram.
3. **Dendrogram Visualization**: It provides a tree-like diagram called a dendrogram, which is a visual representation of the clustering process showing the order of cluster combination and the distance at which clusters are merged.
4. **Distance Metrics and Linkage Criteria**: Hierarchical clustering uses various distance metrics (like Euclidean or Manhattan distance) and linkage criteria (like single linkage, complete linkage, average linkage, and Ward's method) to decide which clusters to merge or split.
5. **Flexibility in Identifying Cluster Shapes**: Hierarchical clustering can identify clusters with various shapes and sizes, unlike partitioning methods that generally assume spherical clusters.
6. **Computational Complexity**: It is generally more computationally intensive than partitioning methods, especially for large datasets, due to the need to compute and store distances between all pairs of points.
7. **Sensitivity to Noise and Outliers**: The method can be sensitive to noise and outliers, as these can influence the formation of clusters and the structure of the dendrogram.
8. **Applications**: It is widely used in fields like biology (for gene and protein sequencing), social science, and linguistics, and is particularly useful for exploratory data analysis where understanding the hierarchical relationship between objects is important.

Density Based Clustering

05 December 2023 10:58

- ✓ **Principle**: Density-based clustering groups data points based on the density of data points in a region. It defines clusters as areas of high density separated by areas of low density. The algorithm identifies clusters as regions where data points are densely packed together, with areas of low density or noise between them.
- ✓ **Examples**: DBSCAN is one of the most popular density-based clustering algorithms. It is known for its efficiency and ability to find clusters of arbitrary shapes.
- ✓ **No Need to Specify Number of Clusters**: Unlike partitioning methods, density-based clustering doesn't require pre-specifying the number of clusters.
- ✓ **Handling Noise and Outliers**: It is robust to outliers and noise, as these are typically not part of the dense regions that form clusters.
- ✓ **Ability to Find Arbitrary Shapes**: Density-based clustering can discover clusters of arbitrary shapes, unlike methods like k-means which are biased towards spherical clusters.
- ✓ **Parameter Sensitivity**: The performance of these algorithms is sensitive to the input parameters, like the radius of neighborhood (e) and the minimum number of points required to form a dense region (MinPts in DBSCAN).
- ✓ **Scalability Issues**: Some density-based algorithms may struggle with very large datasets due to computational and memory constraints.
- ✓ **Applications**: Widely used in fields such as anomaly detection, geospatial data analysis (like identifying geographic regions of interest), and image processing, especially where the shape of the clusters is not known in advance or the data contains noise.

Distribution/Model based Clustering

1. **Statistical Distribution Models**: The central concept of distribution-based clustering is that data points in a cluster follow a certain statistical distribution, most commonly Gaussian or normal distributions.
2. **Parameter Estimation**: These algorithms focus on estimating the parameters (like mean, variance) of the assumed distributions for each cluster. The fit of these parameters to the actual data determines the quality of the clustering.
3. **Expectation-Maximization (EM) Algorithm**: A key algorithm used in distribution-based clustering is EM, which alternates between assigning data points to the most likely distribution (expectation step) and updating the distribution parameters to maximize data fit (maximization step).
4. **Handling of Complex Cluster Shapes**: Unlike methods such as K-Means, distribution-based clustering can identify clusters of various shapes and sizes, making it more flexible in handling real-world data complexities.
5. **Computational Intensity**: The process of estimating distribution parameters and assigning data points can be computationally demanding, especially for large datasets and when the number of features (dimensions) is high.
6. **Handling of Outliers**: These methods can be more robust to outliers, as outliers are less likely to significantly affect the parameters of the overall distribution.
7. **Scalability Issues**: While effective for small to medium-sized datasets, scalability to very large datasets can be challenging due to the computational complexity of the algorithms and the need for more sophisticated optimization techniques.
8. **Soft Clustering**: In soft clustering, each data point is associated with a probability distribution across different clusters, indicating the degree of belonging to each cluster. This is in contrast to hard clustering, where each data point is assigned to exactly one cluster.