Mehrdad Baradaran

99222020

Assignment 3

Unsupervised Learning

K-means – PCA

Prof. Hadi Farahani

Q1 - What is the curse of dimensionality and how does it affect clustering?

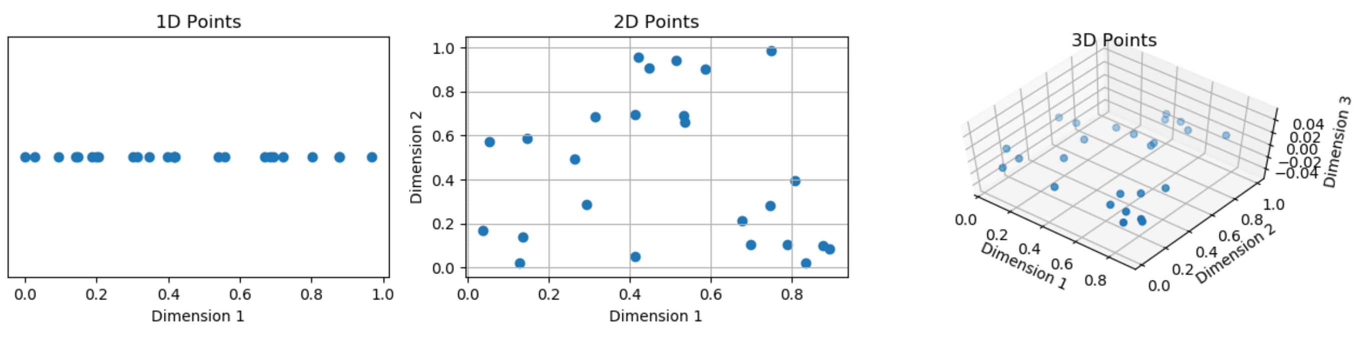
Answer :

The curse of dimensionality refers to the challenges and problems that arise when working with high-dimensional data. It affects clustering in several ways:

1. It increases computational complexity: As the number of dimensions grows, clustering algorithms require significantly more computational resources. The time complexity of many clustering algorithms exponentially depends on the number of dimensions, making them computationally expensive and often impractical for high-dimensional data.

2. It reduces clustering accuracy: With higher dimensions, the notion of proximity between data points becomes less meaningful. In high-dimensional space, most data points are equidistant from each other, making it harder to determine their similarities accurately. Consequently, clustering results may suffer from misassignments, with similar data points being wrongly assigned to different clusters or vice versa.

3. It leads to increased data sparsity: High-dimensional spaces tend to be sparsely populated, meaning that the available data points become increasingly sparse compared to the feature space volume. Sparse data makes it more challenging to identify cohesive clusters, as there may not be enough samples within a specific region to form meaningful clusters.



4. It affects distance metrics: Many clustering algorithms rely on distance metrics to measure similarity between data points. In high-dimensional space, distances between points tend to converge, making it difficult to accurately differentiate between near and far points. Traditional distance metrics, such as Euclidean distance, become less effective, resulting in distorted proximity relationships among data points.

To address the curse of dimensionality in clustering, various techniques and algorithms have been developed. These include dimensionality reduction techniques like Principal Component Analysis (PCA) or t-SNE (t-Distributed Stochastic Neighbor Embedding) to reduce dimensions, feature selection or extraction methods to focus on informative features, or specialized clustering algorithms designed for high-dimensional data, such as subspace clustering or density-based clustering algorithms. These techniques aim to alleviate the challenges posed by the curse of dimensionality and improve clustering performance on high-dimensional datasets.

Q2 - In what cases would you use regular PCA, incremental PCA, randomized PCA, or random projection?

Answer :

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in data analysis and machine learning. It helps transform a high-dimensional dataset into a lower-dimensional space while retaining the most important information or patterns present in the data.

In detail, PCA works as follows:

1. Standardization: So, before getting into the fancy stuff, let's make sure our dataset is playing nice. We start by standardizing the data, which means we make sure all the features have similar scales. We subtract the mean from each feature and divide it by the standard deviation. This ensures that no single feature dominates the analysis just because it has a larger scale.

   For each feature, we subtract the mean (μ) and divide by the standard deviation (σ):

   Standardized value $(x') = (x - \mu) / \sigma$

2. Covariance Matrix: We create a covariance matrix, which tells us about the relationships and variances between different features. Covariance measures how two variables change together. The covariance matrix gives us a glimpse of the correlation structure in the data.

   Given a dataset with n data points and m features, we compute the covariance matrix C as follows:

   $C = (1/n) * \Sigma(x_i - \mu)(x_i - \mu)^T$

where x_i is a column vector representing a data point and μ is the mean vector of the dataset.

3. Eigendecomposition: Now We perform an eigendecomposition on the covariance matrix. This process breaks down the matrix into two important things: eigenvectors and eigenvalues. The eigenvectors represent the directions or principal components that have the most variation in the data. Eigenvalues tell us how much variance is explained by each eigenvector.

   We perform eigendecomposition on the covariance matrix C to obtain eigenvectors and eigenvalues. The eigenvectors (v) and eigenvalues (λ) satisfy the equation:

   $Cv = \lambda v$

4. Selection of Principal Components: We have the eigenvectors and eigenvalues, but we can't keep them all. The principal components that capture most of the variation in the data. So, we sort the eigenvectors based on their eigenvalues and choose the top-k ones. These principal components will be our guide through the dimensionality reduction journey.

5. Projection: We take our original data and project it onto the selected principal components. This projection creates a new, lower-dimensional representation of the data while keeping the essential information intact. We've effectively reduced the complexity without losing the essence.

   To project the original data onto the selected principal components, we multiply the data matrix X (each row representing a data point) by the matrix of eigenvectors V:

   Projected data matrix $Y = X * V$

Regular PCA represents the default way to do Principal Component Analysis (PCA) where you process the entire dataset all at once. It's the go-to approach when you can easily handle the entire dataset without any fuss.

So, when should you opt for regular PCA? Let's explore a few scenarios:

1. Exploratory Data Analysis: You've got a high-dimensional dataset, and you're eager to uncover its secrets. Regular PCA is a powerful tool for exploring and understanding high-dimensional datasets. Regular PCA allows you to uncover hidden patterns, relationships, and structures by condensing the data into a lower-dimensional space. By visualizing the principal components, you can gain insights into the dominant features and their impact on the data.

2. Feature Extraction: PCA can be used to extract a smaller set of meaningful features from a larger feature space. By selecting the top-k principal components, you can capture the most important information while discarding the less significant ones. This feature extraction process reduces noise, removes redundancies, and improves the efficiency of subsequent machine learning algorithms.

3. Data Visualization: Regular PCA is useful when visualizing high-dimensional data in a lower-dimensional space. By projecting the data onto a lower-dimensional subspace defined by the principal components, you can visualize the relationships and clusters in the data. This technique is often used for creating scatter plots, 2D/3D visualizations, and gaining a better understanding of the data.

4. Dimensionality Reduction: Regular PCA is employed to reduce the dimensionality of a dataset while preserving essential information. By

discarding lower-ranked principal components with smaller eigenvalues, you can achieve a lower-dimensional representation of the data. This reduction simplifies subsequent analyses, improves computational efficiency, and mitigates the challenges posed by the curse of dimensionality.

In summary, regular PCA is commonly used for exploratory data analysis, feature extraction, data visualization, and dimensionality reduction tasks. It is suitable when the entire dataset can be loaded into memory and processed efficiently as a whole.

Incremental PCA (Principal Component Analysis) is a variant of the traditional PCA algorithm that enables the analysis of large datasets that may not fit into memory. Unlike standard PCA, which processes the entire dataset at once and can be computationally expensive and memory-intensive for large datasets, incremental PCA breaks down the data into smaller batches or chunks and processes them iteratively.

The core concept behind incremental PCA involves iteratively updating the principal components as new data arrives. It achieves this by continually refining an estimate of the covariance matrix of the data and incorporating new information in small increments. By adopting this approach, incremental PCA mitigates the memory requirements and computational complexity associated with batch PCA.

Here's a high-level overview of how incremental PCA operates:

- Initialization: Initialize the variables and parameters necessary for the analysis.

- Processing data in batches: Divide the dataset into smaller batches or chunks. For each batch, perform the following steps:

  1. Update covariance: Enhance the estimate of the covariance matrix based on the current batch of data.
  2. Compute eigenvectors: Calculate the eigenvectors or principal components of the updated covariance matrix using techniques such as the power iteration method or singular value decomposition (SVD).
  3. Update principal components: Incorporate the new eigenvectors to update the principal components.
- Finalization: Once all the batches have been processed, you can compute the final principal components or continue updating them as new data becomes available.

Incremental PCA offers several advantages:

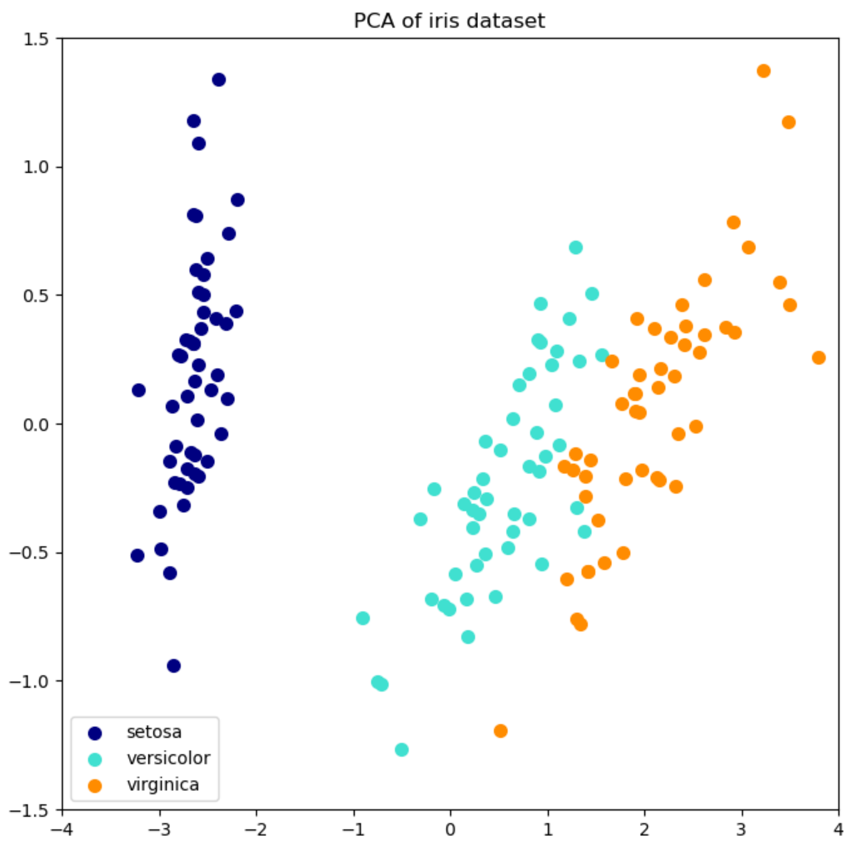
- Memory efficiency: By processing data in smaller chunks, it handles datasets that exceed the memory capacity by avoiding the need to load the entire dataset at once.
- Computational efficiency: The incremental approach minimizes redundant computations on the complete dataset, resulting in faster analysis.
- Online learning: Incremental PCA facilitates continuous learning, allowing easy integration of new data without reprocessing the entire dataset.

However, there are a few considerations to bear in mind:

- Approximation: Incremental PCA provides an approximation of the principal components since it updates the covariance matrix estimate based on partial data.

- Batch size selection: Choosing an appropriate batch size is critical. Small batches may yield noisy estimates, while excessively large batches can undermine the benefits of incremental PCA in terms of memory and computation.
- Data ordering: The sequence in which data arrives can influence the outcomes. Non-random ordering might introduce biases in the incremental PCA process.

In summary, incremental PCA is a valuable technique for conducting PCA on large datasets, particularly when confronted with memory and computational limitations. It enables the efficient computation of principal components by adopting an incremental and memory-friendly approach.

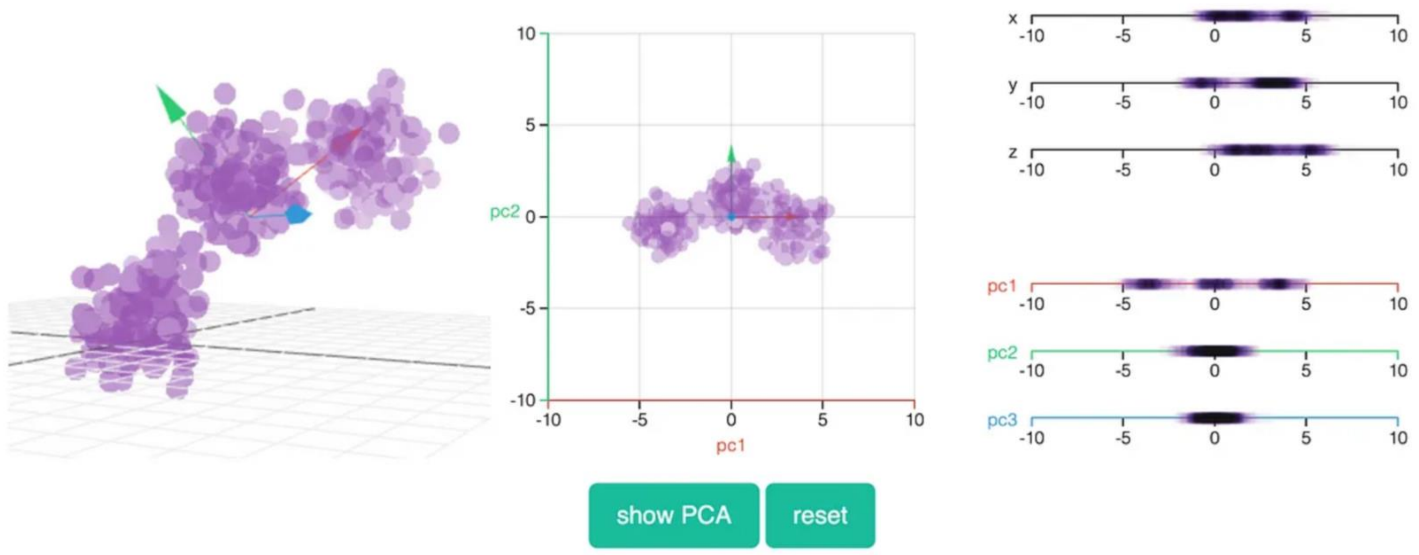


Randomized PCA is a variant of the traditional PCA algorithm that utilizes randomization techniques to approximate the principal components of a dataset. It is employed when dealing with large datasets that possess a high number of dimensions, aiming to reduce computational requirements.

In randomized PCA, the algorithm projects the data into a lower-dimensional subspace using a random matrix. By employing a randomized projection, the algorithm diminishes the problem's dimensionality before calculating the eigenvectors and eigenvalues. To compute the approximate principal components, the algorithm then applies more efficient methods, such as the power iteration method .

Randomized PCA proves valuable in scenarios where computational efficiency is crucial, such as analyzing extensive datasets, performing real-time data analysis, or handling high-dimensional data in machine learning tasks. It provides a satisfactory approximation of the principal components while significantly reducing the computational cost compared to traditional PCA.

However, it is important to note that randomized PCA yields an approximate solution and may not produce the exact principal components as traditional PCA. The degree of approximation depends on the chosen algorithm parameters and the desired accuracy for the specific application. If the accuracy of the principal components is paramount, and computational resources are sufficient, traditional PCA should be considered instead.



Random projection is a technique that actively maps high-dimensional data onto a lower-dimensional space while preserving certain structural properties of the original data. It involves using a random matrix to actively project the data points from a high-dimensional space to a lower-dimensional subspace.

Random projection is applicable in the following cases:

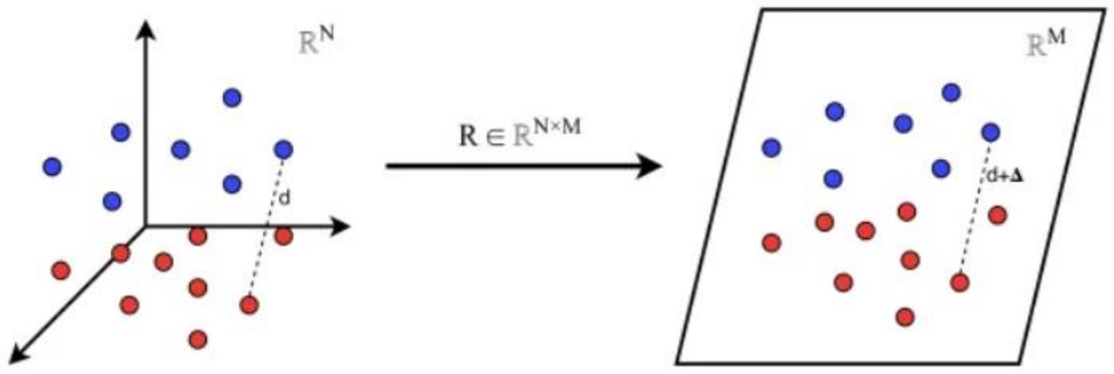
1. Dimensionality Reduction: Random projection actively reduces the number of features or variables when working with high-dimensional data. It combats the curse of dimensionality, where increasing the number of features can degrade the performance of machine learning algorithms. By actively reducing dimensionality, random projection simplifies subsequent analysis and improves computational efficiency.

2. Approximate Nearest Neighbor Search: Random projection actively accelerates nearest neighbor search algorithms. By actively projecting high-dimensional data to a lower-dimensional space, the search complexity is reduced, making it more efficient to find approximate nearest neighbors.

3. Data Compression: Random projection actively compresses data by representing high-dimensional data using a smaller number of dimensions. This is valuable when storage or memory constraints are a concern.

4. Large-scale Data Analysis: Random projection actively reduces the computational complexity of subsequent analyses when dealing with large datasets. By actively projecting the data to a lower-dimensional space, tasks such as clustering, classification, or visualization become more manageable.

It is important to note that random projection introduces some approximation in the data representation, and there may be a loss of information during the projection process. The extent of this loss depends on the specific random projection algorithm and the chosen dimensions for projection. Therefore, evaluating the trade-off between dimensionality reduction and the preservation of important data characteristics based on the specific application requirements is crucial.



Q3 - Does it make sense to chain two different dimensionality reduction algorithms?

Answer :

Yes, there are instances where it makes sense to combine two different dimensionality reduction strategies. In fact, employing one strategy, then another, step-by-step, can be beneficial.

- The reason for this is that each technique has its advantages and disadvantages. Together, we can make use of their various advantages and get a superior end result. Combining several techniques can help us better comprehend the structure of the data because each one captures a different component of the data.

- Imagine it as building blocks. By significantly lowering the dimensionality of the data, the first strategy establishes the framework. The second method then improves the reducing process by building on the first. It's like giving the data another layer of comprehension.

- This mix of methods can also strengthen the reducing process. The other strategy can make up for any shortcomings or shortcomings of the first one. In order to ensure that we extract the most crucial information from the data, it's like having a backup plan.

However, there are a few factors to keep in mind.

- First off, combining approaches can make it harder to evaluate the data. We might need to put in extra work to comprehend and justify the final reduced data.
- Second, it might lengthen the reduction process's duration and computing complexity. As a result, we must weigh the advantages against the increased computational expense.

under conclusion, integrating several dimensionality reduction strategies can be advantageous under some circumstances. It enables us to benefit from their advantages, enhance the end product, and strengthen the reduction process. However, we must be aware of the difficulties with interpretability and probable rise in computational complexity.

Q4 - What are the main assumptions and limitations of PCA?

Answer :

When using the technique, it's vital to take into account the assumptions and limits of principal component analysis (PCA). The basic presumptions and restrictions of PCA are as follows:

Assumptions:

1. Linearity: PCA counts on the linearity of the relationships between variables. It looks for linear combinations of variables that can account for the greatest amount of data variance. A nonlinear underlying relationship may prevent PCA from fully capturing the complexity of the data.

2. Independence: PCA posits that the variables are uncorrelated or have a low degree of interdependence. Strong correlations between variables can cause a few principal components to dominate, which may skew how the data is represented.

3. Equal Variances: Variables with equal variances are assumed by PCA. Variables with higher variances may predominate in the analysis and possibly overshadow those with lower variances if the variances differ significantly.

Limitations:

1. Sensitivity to Outliers: Because PCA seeks to account for as much variance in the data as possible, it is sensitive to outliers. The principle components may be disproportionately affected by outliers, which could skew the study.

2. Not resilient to data scaling: The scale of the variables affects PCA. Even though they have a minor role in describing the underlying structure, bigger scale variables can dominate the study. To get relevant results, the data must be scaled correctly.

3. Lack of Interpretability: Although PCA is a useful method for reducing dimensionality, the resulting principle components could not be easily understood in terms of the original variables. They are linear combinations of the variables, and their meaning might not be immediately obvious.

4. Assumes Linear Relationships: The PCA method relies on the linearity of the underlying relationships between the variables. Strong nonlinear correlations in the data may prevent PCA from capturing the intricate patterns and giving the data an ideal representation.

5. Information Loss: By projecting the data onto a lower-dimensional space, PCA reduces the dimensionality of the data. This projection inevitably results in some information loss. There is a trade-off between dimensionality reduction and maintaining the variability of the original data, and the quantity of information retained relies on the number of primary components that are kept.

6. Handling Difficulties category Variables: PCA struggles with category or ordinal variables and is primarily intended for numerical variables. Categorical variables must be preprocessed further, such as using one-hot encoding or other suitable encoding methods, before being included in PCA.

Understanding these assumptions and limitations is essential when applying PCA. It helps in appropriately interpreting the results, evaluating the suitability of PCA for the data at hand, and considering alternative techniques when the assumptions of PCA are violated or the limitations are problematic for the analysis.

Q5 - How can clustering be used to improve the accuracy of the linear regression model?

Answer :

Cluster-aided regression is a method that uses clustering to increase the precision of a linear regression model. An overview of how clustering can be used to improve a linear regression model's accuracy is provided below:

- The initial step is to find unique clusters within the dataset. K-means, hierarchical clustering, or DBSCAN are a few examples of clustering algorithms that can be used to combine related data points depending on their characteristics.

- Feature Engineering: After the clusters have been located, additional features can be created to record each data point's membership in each cluster. This can be achieved by making binary variables (sometimes known as "dummy variables") to represent each cluster and giving each data point a value of 1 if it belongs to that cluster and a value of 0 otherwise. These indications of cluster membership act as extra features for the regression model.

- Models That Are Specific to a Cluster: For each cluster, distinct linear regression models can be created using the original characteristics and the markers of cluster membership as input variables. This enables the models to accurately represent the distinctive traits and patterns found within each cluster.

- Predictions and Aggregation: New data points can be predicted for by placing them in the proper clusters according to their feature values after training the cluster-specific models. There will be a predicted value for each data point from the relevant cluster-specific model. The outcome can be determined by combining all cluster-specific model predictions, usually by taking the average.

Benefits of Cluster-Aided Regression:

- Greater Accuracy: The linear regression approach can more accurately capture the subtleties and trends within each cluster by utilizing cluster-specific models. This can produce forecasts that are more accurate, particularly when the data shows heterogeneity or non-linear correlations.

- Better Representation: The cluster membership indicators provide the regression model more details about the data points, enabling it to take distinct subgroups or clusters into account. As a result, the model is better able to capture and describe the underlying structure.

- Improved Interpretability: When compared to a single global regression model, cluster-aided regression can produce findings that are easier to understand. Insights into how various factors affect the outcome variable across various clusters can be gained thanks to the diverse interpretations within each subgroup made possible by the individual models for each cluster.

It's important to note that the effectiveness of cluster-aided regression depends on the quality of clustering, appropriate feature engineering, and the suitability of linear regression for the specific problem. Additionally, the choice of clustering algorithm, the determination of the optimal number of clusters, and the evaluation of the overall model performance are critical aspects to consider when implementing this approach.

Q6 - How is entropy used as a clustering validation measure?

Answer :

Entropy serves as a clustering validation measure by assessing the quality and homogeneity of the clusters generated by clustering algorithms. It actively evaluates how well the clusters separate the data points and measures the purity or mixture of the clusters in terms of their assigned class or label distribution. Here's how entropy is actively used as a clustering validation measure:

1. Assigning Labels or Classes: Clustering algorithms actively require known class labels to be assigned to the data points, especially in supervised or semi-supervised scenarios. These labels actively indicate the true groupings or classes to which the data points belong.

2. Cluster Assignments: After applying a clustering algorithm, each data point is actively assigned to a particular cluster. The cluster assignments actively determine which cluster each data point belongs to.

3. Entropy Calculation: Entropy actively quantifies the degree of impurity or mixed class assignments within each cluster. It actively measures the uncertainty or randomness by considering the class distribution of the data points within a cluster.

4. Cluster-wise Entropy Aggregation: The entropies of individual clusters are actively aggregated to obtain an overall measure of clustering quality. This aggregation can be accomplished by actively calculating the mean, weighted mean, or weighted sum of the individual cluster entropies.

5. Interpreting Entropy Scores: Lower entropy values actively indicate more homogeneous clusters with similar class distributions, signifying that the clustering algorithm has effectively separated the data points based on their classes. Higher entropy values actively imply more mixed or impure clusters, suggesting that the clustering algorithm may not have accurately captured the underlying structure.

6. Comparing Clustering Results: Entropy actively facilitates the comparison of different clustering algorithms or parameter settings. Lower entropy values actively indicate better clustering results, while higher entropy values actively suggest poorer performance.

7. Combined Validation Measures: Entropy is often actively used in conjunction with other clustering validation measures, such as the silhouette coefficient, Dunn index, or Rand index, to actively gain a more comprehensive understanding of the clustering quality.

It's important to note that the availability of true class labels and the assumption of class-based clustering actively influence the use of entropy-based validation measures. In cases where class labels are unavailable or clustering is performed in an unsupervised manner, other measures, such as the silhouette coefficient or cohesion-separation measures, may be more suitable.

Q7 - What is label propagation? Why would you implement it, and how? (Extra Point)

Answer :

Label propagation is a semi-supervised learning algorithm that aims to propagate or extend class labels from labeled data points to unlabeled data points in a dataset. It leverages the assumption that neighboring data points in feature space are likely to share similar labels. The algorithm assigns labels to unlabeled data points based on the labels of their neighboring data points.

Label propagation is implemented to make use of the available labeled data to infer labels for the unlabeled data points. It is particularly useful in scenarios where obtaining labeled data is expensive, time-consuming, or limited, while unlabeled data is abundant. By utilizing the information from the labeled data and propagating labels to the unlabeled data, label propagation can effectively expand the labeled dataset and improve classification or prediction accuracy.

Label propagation would be implemented in situations where there is a scarcity of labeled data but an abundance of unlabeled data. Here are a few reasons why you would implement label propagation:

- Limited Labeled Data: Obtaining labeled data can be expensive, time-consuming, or impractical in many real-world scenarios. Label propagation allows you to make the most of the available labeled data by leveraging the relationships between labeled and unlabeled data points.

- Semi-Supervised Learning: Label propagation is a popular technique in semi-supervised learning, where you have a small set of labeled data points and a large set of unlabeled data points. By propagating labels from the labeled data to the unlabeled data, label propagation expands the labeled dataset, which can improve the accuracy of subsequent classification or prediction tasks.

- Neighborhood Information: Label propagation takes advantage of the assumption that neighboring data points in the feature space are likely to share similar labels. It exploits the inherent structure and patterns within the data to propagate labels. This can be beneficial in scenarios where the local structure of the data plays a significant role in determining the class memberships.

- Addressing Class Imbalance: Label propagation can help mitigate the issue of class imbalance in the labeled data. By propagating labels to the unlabeled data, it can increase the representation of minority classes, leading to a more balanced dataset and potentially improving the performance of classification models.

- Incremental Learning: Label propagation supports incremental learning, allowing new unlabeled data points to be incorporated and labeled based on the existing labeled data. This enables the model to adapt and improve over time as new data becomes available.

- Unsupervised Learning Exploration: Label propagation can be used as a tool to explore and understand the underlying structure of the data. By propagating labels, it provides insights into the potential clusters or groups present in the unlabeled data.

Here's an overview of how label propagation can be implemented:

1. Data Representation: Represent the dataset as a graph, where each data point is a node, and the edges represent the relationships or similarities between the

data points. Commonly used graphs include k-nearest neighbor graphs or similarity graphs.

2. Label Initialization: Start with a subset of data points that have known labels. Assign these labeled data points with their respective class labels.

3. Label Propagation: Propagate the labels from the labeled data points to the unlabeled data points iteratively. The labels are propagated based on the similarity or proximity between the data points in the graph. The neighboring data points influence each other's labels, and the process is repeated until convergence.

4. Label Updating: Update the labels of the unlabeled data points based on the propagated labels from the neighboring data points. The update can be done by various methods, such as weighted averaging or considering a voting scheme based on the labels of neighboring data points.

5. Convergence: Repeat the label propagation and updating process until the labels stabilize or until a predefined stopping criterion is met. The algorithm typically converges when the labels of the data points no longer change significantly between iterations.

6. Label Inference: Once the algorithm converges, the propagated labels of the unlabeled data points can be considered as predicted labels. These inferred labels can be used for further analysis, such as classification, clustering, or other downstream tasks.

Label propagation requires careful parameter tuning, such as the number of neighbors to consider, the weighting scheme, and the stopping criterion.

Additionally, it is crucial to consider the quality and representativeness of the labeled data and the graph construction method to ensure reliable and accurate label propagation.

Implementing label propagation involves constructing the graph, initializing the labels, propagating and updating the labels iteratively, and finally inferring the labels for the unlabeled data points. Several libraries and frameworks, such as scikit-learn in Python, provide implementations of label propagation algorithms that can be readily utilized in data analysis tasks.