QA from GPT

Lecture 13 - Clustering
Quanying Liu
2024.04.18

0. Clustering methods in machine learning

Clustering is a type of unsupervised learning technique used in machine learning to group a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups. It's widely used across various applications, such as market segmentation, anomaly detection, document clustering, and image segmentation. Here are some of the most commonly used clustering methods in machine learning:

1. K-means Clustering
- **Overview**: K-means is one of the simplest and most popular clustering algorithms. It aims to partition a set of points into K clusters in which each point belongs to the cluster with the nearest mean.
- **Procedure**:
  1. Initialize K centroids randomly.
  2. Assign each point to the nearest centroid, forming K clusters.
  3. Recompute the centroid of each cluster.
  4. Repeat steps 2 and 3 until the centroids no longer move significantly.
- **Use Cases**: General-purpose clustering, especially in scenarios where clusters are roughly spherical.

2. Hierarchical Clustering
- **Overview**: This method builds a tree of clusters and does not require a pre-specified number of clusters. Hierarchical clustering can be either agglomerative (bottom-up) or divisive (top-down).
- **Procedure**:
  1. Start with each point as its own cluster.
  2. Merge the closest pair of clusters.
  3. Repeat step 2 until all points are in a single cluster.
  4. The user can cut the tree at different levels to obtain the desired number of clusters.
- **Use Cases**: When the structure of the clusters is important, such as in taxonomy and evolutionary studies.

3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- **Overview**: DBSCAN groups together points that are closely packed together, marking as outliers points that lie alone in low-density regions.
- **Procedure**:
  1. Classify points as core points, border points, or noise based on density.

2. Eliminate noise points.
3. Form a separate cluster for each core point or connected group of core points.
4. Assign each border point to one of the clusters of its associated core points.

- **Use Cases**: Useful in spatial data analysis where noise points can be present, and cluster shape is not spherical.

4. Mean Shift Clustering

- **Overview**: Mean shift clustering aims to discover blobs in a smooth density of samples. It is a centroid-based algorithm, which works by updating candidates for centroids to be the mean of the points within a given region.
- **Procedure**:
1. Begin with an initial estimate for a centroid.
2. Compute the mean of the points within a window centered at that centroid.
3. Translate the window to the mean and repeat until convergence.
- **Use Cases**: Image processing and computer vision for tracking and image segmentation.

5. Spectral Clustering

- **Overview**: Spectral clustering uses the eigenvalues of a similarity matrix of the data to reduce dimensionality before clustering in fewer dimensions.
- **Procedure**:
1. Construct a similarity graph, typically using the Gaussian (radial basis function) kernel.
2. Compute the Laplacian matrix of the graph.
3. Perform dimensionality reduction on the Laplacian via eigen decomposition.
4. Cluster the points in the reduced space using k-means or another method.
- **Use Cases**: Useful in clustering problems where the clusters are connected but not necessarily compact or evenly distributed.

6. Gaussian Mixture Models (GMM)

- **Overview**: GMM is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.
- **Procedure**:
1. Initialize the Gaussian distributions (mean and variance) and mixing coefficients randomly.
2. Use the Expectation-Maximization (EM) algorithm to iteratively improve the parameters.
3. Each point is assigned a probability of belonging to each cluster.
- **Use Cases**: Image segmentation, anomaly detection, and any application requiring soft clustering.

Each of these clustering methods has its strengths and is suitable for specific types of data distributions and specific requirements of the application domain. The choice of a clustering algorithm depends on the size and nature of the data, the desired number of clusters, the expected cluster shape, and the computational efficiency required.

1. Example of LDA in medical application

Clustering has several impactful applications in the medical field, especially in the areas of diagnosis, research, and treatment personalization. One notable example is in the analysis of patient data to identify patterns that signify different subtypes of diseases.

Example: Clustering in Cancer Genomics

Application: Identifying Cancer Subtypes
Cancer is not a single disease but a collection of related diseases, which can vary significantly in their genetic makeup, prognosis, and response to treatment. Clustering algorithms can analyze genetic data from tumor samples to identify groups (or subtypes) of cancers that share similar genetic profiles.

Process:
1. **Data Collection**: Gather genetic expression data from a large number of cancer patients.
2. **Feature Selection**: Identify genes that show significant variation across different samples, which are likely to be relevant for cancer differentiation.
3. **Clustering Implementation**: Apply clustering algorithms (such as k-means, hierarchical clustering, or more sophisticated methods like spectral clustering) to group the cancer samples based on their genetic expression profiles.
4. **Analysis and Validation**: Analyze the resulting clusters to identify which genes are most characteristic of each cluster. Validate whether these clusters correspond to differences in clinical outcomes, such as survival rates, response to treatment, or recurrence of cancer.

Benefits:
- **Personalized Medicine**: Understanding cancer subtypes can help in tailoring treatment strategies to individual patients, potentially increasing the effectiveness of the treatments while minimizing side effects.
- **New Insights**: Clustering can lead to the discovery of previously unrecognized cancer subtypes, which can guide further research and clinical trials.
- **Improved Diagnostic Accuracy**: Clustering genetic data can improve the accuracy of cancer diagnoses by clearly distinguishing between types of cancer that may look similar under a microscope but are genetically distinct.

Real-World Implementation

A real-world implementation of this is seen in breast cancer research, where clustering methods have been used to identify distinct subtypes such as luminal A, luminal B, HER2-enriched, and basal-like, each of which has different treatment protocols and prognoses. This stratification is crucial for the so-called "precision medicine" approach, which aims to customize healthcare, with medical decisions and treatments tailored to individual patients.

This example of clustering in cancer genomics demonstrates how machine learning techniques can be leveraged to enhance our understanding and treatment of complex diseases, ultimately leading to better patient outcomes.

2. Similarity measurements used in machine learning?

In machine learning, similarity measures are used to quantify how similar two data items are. These measurements are crucial in various applications such as clustering, classification, recommendation systems, and anomaly detection. Here's a rundown of some commonly used similarity measures:

1. **Euclidean Distance**
One of the simplest and most widely used methods, Euclidean distance measures the straight-line distance between two points in Euclidean space. It's particularly common in clustering algorithms like k-means and in systems where geometric relationships in high-dimensional spaces are studied.

2. **Cosine Similarity**
Cosine similarity measures the cosine of the angle between two vectors. This metric is particularly useful in text analysis where the magnitude of the vector might not be as important as the angle between vectors. It is widely used in text mining, information retrieval, and recommendation systems to assess how similar two documents or items are based on their vector representations.

3. **Manhattan Distance**
Also known as the taxicab or city block distance, this measure sums the absolute differences of their Cartesian coordinates. It is a useful alternative to Euclidean distance when the effect of outliers needs to be diminished.

4. **Jaccard Index**
The Jaccard index, or Jaccard similarity coefficient, is used to compare the similarity and diversity of sample sets. It measures similarity between finite sample sets as the size of the intersection divided by the size of the union of the sample sets. This is particularly useful in clustering and is widely applied in ecological and environmental data analysis.

5. **Pearson Correlation Coefficient**
This measures the linear correlation between two variables. It gives a value between +1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. It's commonly used in statistics and data mining for feature selection and to analyze relationships between data sets.

6. **Hamming Distance**
This is used to measure the distance between two binary vectors (commonly strings of the

same length). It's often used in information theory and coding, especially for error detection and correction.

7. **Mahalanobis Distance**
This distance measure is scale-invariant, taking into account the correlations of the data set and is not biased by scale. It differs from Euclidean distance in that it requires knowledge of the data distribution (covariance). It's used in anomaly detection where detecting the outliers with respect to the underlying distribution of the data is crucial.

8. **Levenshtein Distance (Edit Distance)**
This metric measures how many single-character edits (insertions, deletions, or substitutions) are required to change one word into another. It's particularly useful in natural language processing and speech recognition to assess similarity between words.

These similarity measures are foundational in designing and training machine learning models, allowing algorithms to make decisions about data classification, clustering, and retrieval based on how closely data points resemble one another. Each metric has its own strengths and is chosen based on the specific requirements and characteristics of the data and the problem domain.

3. K-means clustering

K-means clustering is one of the simplest and most commonly used clustering algorithms in machine learning. It aims to partition a set of points into *K* clusters in which each point belongs to the cluster with the nearest mean. This results in a partitioning of the data space into Voronoi cells.

How K-means Works
1. **Initialization**: Start by selecting *K* initial centroids randomly from the data points. These centroids are the initial guesses for the centers of the clusters.
2. **Assignment**: Assign each data point to the closest centroid. The "closeness" is typically determined by the Euclidean distance between the point and the centroid.
3. **Update**: Recompute the centroids of the clusters by taking the mean of all points assigned to each cluster.
4. **Iteration**: Repeat the assignment and update steps until the centroids no longer move significantly or the assignments no longer change. This indicates convergence.

Key Features of K-means
- **Efficiency**: K-means is generally faster and more scalable compared to other clustering methods, especially for large datasets.
- **Applicability**: It works well when the clusters are spherical and relatively well separated.
- **Limitations**:
  - The need to specify *K* in advance.

- Sensitivity to the initial choice of centroids.
- Poor performance with non-convex clusters or clusters of varying sizes and densities.

Choosing the Number of Clusters

Determining the optimal number of clusters *K* is a significant challenge in K-means. Methods such as *the Elbow method*, the *Silhouette method*, or *cross-validation techniques* are often used. The Elbow method, for example, involves plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use.

Variants and Extensions

- **K-means++**: Improves the initialization phase by spreading out the initial centroids before proceeding with the standard K-means algorithm.
- **Mini-Batch K-means**: Uses small, random batches of data points to update the centroids, rather than the entire dataset. This speeds up the computation.
- **Spherical K-means**: Adapts K-means for use with non-Euclidean similarity measures, such as cosine similarity, which is particularly useful in text clustering.

Applications

K-means clustering is used across various domains:

- **Market segmentation**: Identifying distinct customer groups based on purchasing patterns.
- **Document clustering**: Grouping text documents into similar topics for organization or summary.
- **Image segmentation**: Partitioning an image into segments that contain pixels with similar attributes.
- **Anomaly detection**: Clustering can help detect unusual patterns that deviate significantly from expected behavior.

K-means clustering is a powerful tool in both exploratory data analysis and as a preprocessing step for other algorithms. It provides an intuitive and straightforward way to quickly understand the underlying structure of the data and group similar data points together. However, its effectiveness can depend heavily on the shape of the dataset and the value of \(K \) chosen.

4. python code to perform K-means clustering for Clustering in Cancer Genomics

To perform K-means clustering on cancer genomics data using Python, you can use libraries such as `pandas` for data manipulation and `scikit-learn` for applying the K-means algorithm. Here's a simple example using synthetic data to demonstrate how you could set up your code. This will assume you have gene expression data where each row represents a sample (e.g., a tumor sample), and each column represents expression levels of different genes.

Step-by-Step Python Code:

1. **Import Necessary Libraries:**

First, you'll need to import the libraries necessary for handling the data and performing clustering.

```python
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

2. **Load and Prepare the Data:**

Load your data into a pandas DataFrame. Here, I'll create a synthetic dataset for demonstration.

```python
# Synthetic example data: 100 samples and 5 genes
data = {
    'Gene1': np.random.rand(100),
    'Gene2': np.random.rand(100),
    'Gene3': np.random.rand(100),
    'Gene4': np.random.rand(100),
    'Gene5': np.random.rand(100)
}
df = pd.DataFrame(data)
```

3. **Perform K-means Clustering:**

Apply the K-means algorithm. You need to specify the number of clusters `n_clusters` which should be determined based on domain knowledge, exploratory analysis, or techniques like the Elbow method.

```python
# Number of clusters
k = 3

# Creating the kmeans object with k clusters
kmeans = KMeans(n_clusters=k)

# Fitting the kmeans object to the data
kmeans.fit(df)

# Getting the cluster labels
clusters = kmeans.labels_
```

```

**4. **Analyze the Cluster Output**:**

Add the cluster labels back to your original DataFrame to see which sample belongs to which cluster.

```python
df['Cluster'] = clusters
```

5. **Visualization (Optional):**

Visualize the results using a plot, which can be helpful to see the distribution of samples in each cluster. For a high-dimensional dataset like genomics, consider using PCA (Principal Component Analysis) or t-SNE for dimensionality reduction before plotting.

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df.drop('Cluster', axis=1))
principalDf = pd.DataFrame(data=principalComponents, columns=['PC1', 'PC2'])
principalDf['Cluster'] = clusters

# Plotting
fig, ax = plt.subplots()
scatter = ax.scatter(principalDf['PC1'], principalDf['PC2'], c=principalDf['Cluster'], cmap='viridis')
legend = ax.legend(*scatter.legend_elements(), title="Clusters")
ax.add_artist(legend)
plt.show()
```

This example provides a basic framework. When working with real genomic data, consider the scale of the data, potential noise, and biological relevance of the genes involved. It's crucial to preprocess the data (e.g., normalization) appropriately before clustering to ensure meaningful results.

5. Rewrite the code to use the Elbow method to choose the number of clusters

Here's an updated version of the Python code that uses the Elbow method to determine the optimal number of clusters for K-means clustering, particularly useful for applications such as clustering in cancer genomics:
```

Step-by-Step Python Code with Elbow Method:

1. **Import Necessary Libraries:**

Begin by importing the libraries necessary for handling the data, performing clustering, and plotting.

```python
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

2. **Generate Synthetic Data:**

For demonstration, we'll use synthetic data representing gene expression levels.

```python
# Generate synthetic data: 100 samples and 5 genes
np.random.seed(42)  # for reproducibility
data = {
    'Gene1': np.random.rand(100),
    'Gene2': np.random.rand(100),
    'Gene3': np.random.rand(100),
    'Gene4': np.random.rand(100),
    'Gene5': np.random.rand(100)
}
df = pd.DataFrame(data)
```

3. **Elbow Method to Determine Optimal Number of Clusters:**

Calculate the *Within-Cluster Sum of Squares (WCSS)* for a range of cluster numbers to find the optimal `k`.

```python
wcss = []
for i in range(1, 11):  # test 1 to 10 clusters
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)

# Plotting the results onto a line graph to observe the 'Elbow'
plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), wcss)
```

```
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

4. **Run K-means Clustering with Optimal Cluster Number:**

Choose the number of clusters where the WCSS begins to decrease more slowly (elbow point).

```python
# Let's say the elbow is observed at k = 3 from the plot
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(df)
clusters = kmeans.labels_
df['Cluster'] = clusters
```

5. **Dimensionality Reduction for Visualization:**

Use PCA to reduce the dimensions to two principal components for visualization.

```python
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df.drop('Cluster', axis=1))
principalDf = pd.DataFrame(data=principalComponents, columns=['PC1', 'PC2'])
principalDf['Cluster'] = clusters

# Plotting the clusters
fig, ax = plt.subplots()
scatter = ax.scatter(principalDf['PC1'], principalDf['PC2'], c=principalDf['Cluster'], cmap='viridis')
legend = ax.legend(*scatter.legend_elements(), title="Clusters")
ax.add_artist(legend)
plt.show()
```

This code provides a comprehensive method for choosing the optimal number of clusters using the Elbow method, fitting the K-means model, and visualizing the results, which can be especially valuable in fields like genomics for grouping similar expression profiles.

6. How to evaluate the clustering performance?

Evaluating the performance of clustering algorithms is crucial, particularly because clustering is often used to understand the underlying patterns in data without prior knowledge of the outcomes. Several metrics can be used to assess the quality of clustering, depending on whether true cluster labels are known.

When True Labels Are Known

If you have ground truth labels for your data, you can use external metrics that compare the clustering results to these labels:

1. **Adjusted Rand Index (ARI)**:
   - ARI measures the similarity between two assignments, ignoring permutations and with a chance normalization. It ranges from -1 to 1, where 1 indicates perfect agreement between the clusters and the true labels.

2. **Normalized Mutual Information (NMI)**:
   - NMI is a normalization of the Mutual Information (MI) score that measures the agreement of the true labels and the predicted labels, corrected for chance. It ranges from 0 (no mutual information) to 1 (perfect correlation).

3. **Homogeneity, Completeness, and V-Measure**:
   - **Homogeneity**: Each cluster contains only members of a single class.
   - **Completeness**: All members of a given class are assigned to the same cluster.
   - **V-Measure**: The harmonic mean between homogeneity and completeness.

When True Labels Are Not Known

In many practical applications, true labels aren't available. In these cases, internal metrics that measure the quality of the clustering structure itself are used:

1. **Silhouette Score**:
   - Measures how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1 to +1, where a high value indicates that objects are well matched to their own cluster and poorly matched to neighboring clusters.

2. **Davies-Bouldin Index**:
   - The ratio of within-cluster distances to between-cluster distances. A lower Davies-Bouldin Index indicates a better partitioning.

3. **Calinski-Harabasz Index**:
   - Also known as the Variance Ratio Criterion, this index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters. Higher values generally indicate better clustering.

Practical Considerations

- **Choosing Metrics**: The choice of metric should align with the specific characteristics of the data and the purpose of the clustering. For example, if compact and well-separated clusters are desired, Silhouette Score might be particularly informative.

- **Visualization**: Visualizing the data using techniques like PCA (Principal Component Analysis) or t-SNE can also provide insights into the effectiveness of clustering by showing the spatial distribution of clusters.

- **Stability**: Evaluating the stability of clustering results by repeating the clustering with different subsets of data or initial conditions can help validate the robustness of the conclusions.

Overall, the evaluation of clustering is context-dependent and should be guided by the specific objectives of the analysis and the nature of the data.

7. Can you recommend some recent papers about clustering methods published on top ML conferences?

If you're interested in recent advancements and discussions around clustering methods, particularly those presented at top machine learning conferences or in reputable journals, here are a few noteworthy papers you might find intriguing:

1. **"A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions"** - This paper provides an extensive overview of deep clustering methods, discussing various designs and their applications, datasets, and evaluations. It thoroughly explains the interaction between representation learning and clustering in the context of deep learning frameworks.

2. **"Deep Clustering: A Comprehensive Survey"** - This survey focuses on the integration of deep learning with clustering, specifically detailing methods based on autoencoders, graph neural networks, variational autoencoders, and generative adversarial networks. It highlights the importance of feature extraction in unsupervised learning and reviews various clustering-oriented losses.

3. **"A Survey of Deep Graph Clustering: Taxonomy, Challenge, Application, and Open Resource"** - This paper delves into graph clustering using deep learning, exploring adversarial deep graph clustering methods, unified frameworks for data fusion, and the application of contrastive learning in graph clustering.

4. **"Smoothed Multi-View Subspace Clustering"** - It explores efficient clustering in multi-view scenarios, discussing algorithmic details for achieving robust multi-view subspace clustering and providing comparative analysis with other state-of-the-art

methods.

These papers showcase the diverse approaches to clustering in modern machine learning, encompassing deep learning techniques and their applications across different data types and structures. If you need a deeper dive into any specific methodology or application, these surveys and studies can provide a solid foundation and comprehensive insights.