Assignment8

July 26, 2024

1 Clustering

1.1 1. DBSCAN

Using DBSCAN iterate (for-loop) through different values of `min_samples` (1 to 10) and `epsilon` (.05 to .5, in steps of .01) to find clusters in the road-data used in the Lesson and calculate the Silohouette Coeff for `min_samples` and `epsilon`. Plot ***one*** line plot with the multiple lines generated from the min_samples and epsilon values. Use a 2D array to store the SilCoeff values, one dimension represents `min_samples`, the other represents epsilon.

Expecting a plot of `epsilon` vs `sil_score`.

[ ]:
```
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
import numpy as np
import matplotlib.pyplot as plt
from joblib import Parallel, delayed

# Load the road-data used in the Lesson
road_data = pd.read_csv('3D_spatial_network.txt')  # Replace 'path_to_road_data.
 ↪csv' with the actual file path

# Create a 2D array to store the Silhouette Coefficient values
silhouette_scores = np.zeros((10, 6))  # 10 min_samples and 6 epsilon values

# Subsample the road-data
subsampled_data = road_data.sample(n=1000, random_state=42)

# Define a function to calculate silhouette score for a given min_samples and␣
 ↪epsilon
def calculate_silhouette(min_samples, epsilon, data):
    dbscan = DBSCAN(eps=epsilon, min_samples=min_samples)
    clusters = dbscan.fit_predict(data)
    if len(set(clusters)) > 1:
        silhouette = silhouette_score(data, clusters)
        return silhouette
    else:
        return np.nan
```

```
# Iterate through different values of min_samples and epsilon
min_samples_range = range(1, 11)
epsilon_values = np.arange(0.1, 0.61, 0.1)

# Parallel computation of silhouette scores
results = Parallel(n_jobs=-1)(delayed(calculate_silhouette)(min_samples,␣
 ↪epsilon, subsampled_data) for min_samples in min_samples_range for epsilon␣
 ↪in epsilon_values)

# Reshape the results into a 2D array
silhouette_scores = np.array(results).reshape((10, 6))

# Plot epsilon vs sil_score
for i in range(10):
    plt.plot(epsilon_values, silhouette_scores[i], label=f"min_samples={i+1}")

plt.xlabel('Epsilon')
plt.ylabel('Silhouette Coefficient')
plt.title('Epsilon vs Silhouette Coefficient for Different min_samples')
plt.legend()
plt.show()
```

1.2 2. Clustering your own data

Using your own data, find relevant clusters/groups within your data (repeat the above). If your data is labeled with a class that you are attempting to predict, be sure to not use it in training and clustering.

You may use the labels to compare with predictions to show how well the clustering performed using one of the clustering metrics (http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation).

If you don't have labels, use the silhouette coefficient to show performance. Find the optimal fit for your data but you don't need to be as exhaustive as above.

Additionally, show the clusters in 2D or 3D plots.

As a bonus, try using PCA first to condense your data from N columns to less than N.

Two items are expected: - Metric Evaluation Plot (like in 1.) - Plots of the clustered data

[ ]:
```
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # for 3D plotting

# Load the road-data used in the Lesson
```

```
road_data = pd.read_csv('3D_spatial_network.txt')  # Replace 'path_to_road_data.
 ↪csv' with the actual file path

# Subsample the road-data
subsampled_data = road_data.sample(n=1000, random_state=42)

# Perform DBSCAN clustering
dbscan = DBSCAN(eps=0.1, min_samples=5)  # Example parameters, you can adjust␣
 ↪these
clusters = dbscan.fit_predict(subsampled_data[['X', 'Y', 'Z']])  # Assuming the␣
 ↪columns are X, Y, Z

# Calculate silhouette score
silhouette = silhouette_score(subsampled_data[['X', 'Y', 'Z']], clusters)
print(f"Silhouette Score: {silhouette}")

# Visualize clusters in 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(subsampled_data['X'], subsampled_data['Y'], subsampled_data['Z'],␣
 ↪c=clusters)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```

[ ]: