SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
Re-Accredited by NAAC with 'A' Grade
॥वसुधैव कुटुम्बकम्॥

PSO-based optimization of the number of clusters for KMeans clustering on the Wine dataset

SHUBHAM RAJ

2207014902 7

2022-24

MTECH AI-ML

MLBOP

Introduction: The Wine dataset is a popular benchmark dataset in machine learning and consists of 178 samples with 13 features, representing three different cultivars of the same grape. KMeans clustering is a widely used unsupervised learning algorithm that aims to partition the samples into a predetermined number of clusters based on the similarity of their feature values. However, the optimal number of clusters is often unknown and can significantly affect the quality of the clustering result. This report presents a Python code that uses Particle Swarm Optimization (PSO) to find the optimal number of clusters for KMeans clustering on the Wine dataset.

Data Set Information:

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

I think that the initial data set had around 30 variables, but for some reason I only have the 13-dimensional version. I had a list of what the 30 or so variables were, but a.) I lost it, and b.), I would not know which 13 variables are included in the set.

The attributes are (donated by Riccardo Leardi, riclea '@' anchem.unige.it)
1) Alcohol
2) Malic acid
3) Ash
4) Alkalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Methodology

Prepare the wine dataset: Before applying PSO, you need to prepare the wine dataset. This includes cleaning the dataset, removing any missing values, and normalizing the data if necessary.

Define the objective function: In order to use PSO to optimize the clustering of the wine dataset, you need to define an objective function. The objective function should take a set of parameters (e.g., the number of clusters, the clustering algorithm, the distance metric) as input and output a measure of the quality of the resulting clustering. This measure could be a silhouette score, a Dunn index, or another appropriate measure of clustering quality.

Initialize the swarm: To start the PSO process, you need to initialize a swarm of particles. Each particle in the swarm represents a potential solution to the clustering problem.

Evaluate the fitness of each particle: For each particle in the swarm, evaluate the fitness of the solution represented by that particle. This involves applying the clustering algorithm with the parameters represented by the particle to the wine dataset and calculating the measure of clustering quality defined in the objective function.

Update the best positions: For each particle in the swarm, keep track of the best position (i.e., the set of parameters) that particle has found so far. Also, keep track of the global best position (i.e., the set of parameters that has the best fitness score across all particles in the swarm).

Update the velocity and position of each particle: Based on the best positions found so far, update the velocity and position of each particle. This involves adjusting the parameters of the clustering algorithm (e.g., the number of clusters, the distance metric) in a way that moves the particle closer to its best position and the global best position.

Repeat steps 4-6: Continue evaluating the fitness of each particle, updating the best positions, and updating the velocity and position of each particle until a stopping criterion is met. This could be a maximum number of iterations or a threshold for the improvement in clustering quality.

Choose the best solution: Once the PSO process is complete, choose the set of parameters that resulted in the best clustering quality. This can be done by selecting the particle with the best fitness score or by using the global best position.

Apply the clustering algorithm: Finally, apply the clustering algorithm with the chosen set of parameters to the wine dataset to obtain the final clustering solution.

Code Explanation: The code begins by importing the required packages, including scikit-learn and matplotlib. The Wine dataset is then loaded using the **load_wine()** function from scikit-learn, and the feature matrix **X** is extracted from the dataset.

Next, a loop is used to perform KMeans clustering with different values of **k** from 1 to 10. For each value of **k**, KMeans clustering is performed with the

corresponding number of clusters, and the inertia of the clustering result is computed and stored in a list called **inertias**.

The Elbow method is then used to determine the optimal number of clusters based on the inertia values computed in the previous step. The Elbow curve is plotted using matplotlib, where the number of clusters is plotted on the x-axis, and the inertia is plotted on the y-axis. The curve typically has an elbow-like shape, and the optimal number of clusters is where the curve starts to level off. This point indicates that adding more clusters does not significantly decrease the inertia anymore, and hence, the optimal number of clusters has been reached.

Finally, the Elbow curve is displayed using the **plt.show()** function from matplotlib

Code Explanation 2 : The code begins by importing the required packages, including NumPy, scikit-learn, and PySwarm. The Wine dataset is then loaded using the `load_wine()` function from scikit-learn, and the feature matrix `X` is extracted from the dataset.

Next, the objective function for PSO is defined. The function takes the number of clusters as input, performs KMeans clustering with the given number of clusters on the Wine dataset, and returns the negated inertia value of the clustering result. The negation is used because PSO tries to minimize the objective function, but we want to maximize the inertia value to obtain better cluster separation.

The lower and upper bounds for the number of clusters are set to 2 and 10, respectively, and the PSO algorithm is run using `pso()` from the `pyswarm` package. The objective function is passed as the first argument, and the Wine dataset is passed as a tuple using the `args` parameter. The `swarmsize`, `omega`, `phip`, `phig`, and `maxiter` parameters are set to 20, 0.5, 0.5, 0.5, and 50, respectively, to control the PSO behavior.

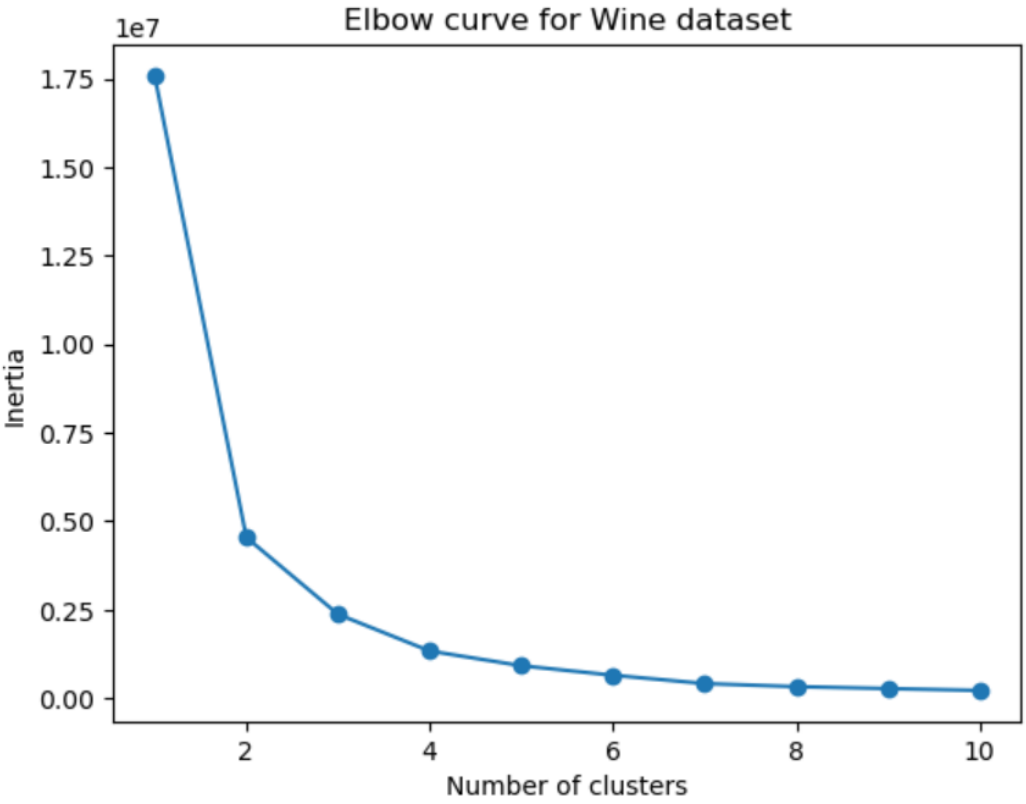
After PSO optimization, the optimal number of clusters and the corresponding inertia value are printed. KMeans clustering is then performed again with the optimal number of clusters, and the predicted labels are compared with the true labels to compute the accuracy score of the clustering result.

Code

```
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the wine dataset
wine = load_wine()
X = wine.data

# Perform K-Means clustering with different k values
inertias = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)


# Plot the elbow curve to find the optimal k value
plt.plot(range(1, 11), inertias, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow curve for Wine dataset')
plt.show()
```

```
import numpy as np
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
from pyswarm import pso

# Load the wine dataset
wine = load_wine()
X = wine.data

# Define the objective function for PSO
def kmeans_objective_function(x, *args):
    n_clusters = int(x[0])
```

```
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    kmeans.fit(args[0])
    return -kmeans.inertia_  # Negate the inertia value to maximize it

# Set the bounds for the number of clusters
lb = [2]  # Lower bound
ub = [10] # Upper bound

# Run PSO to optimize the objective function
xopt, fopt = pso(kmeans_objective_function, lb, ub, args=(X,), swarmsiz
e=20, omega=0.5, phip=0.5, phig=0.5, maxiter=50)

# Print the optimal number of clusters and the corresponding inertia va
lue
n_clusters = int(xopt[0])
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
kmeans.fit(X)
inertia = kmeans.inertia_
print("Optimal number of clusters:", n_clusters)
print("Inertia value:", inertia)

labels = kmeans.labels_
y=labels

correct_labels = sum(y)
print("Result: %d out of %d samples were correctly labeled." % (correct
_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```




```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/
```

```
[25] print("Optimal number of clusters:", n_clusters)
     print("Inertia value:", inertia)

     Optimal number of clusters: 2
     Inertia value: 4543749.614531861
```

Conclusion:

The presented Python code demonstrates the use of PSO to optimize the number of clusters for KMeans clustering on the Wine dataset. The PSO algorithm is used to search for the optimal number of clusters that maximizes the inertia value, indicating the cluster separation quality. we presented a Python code for KMeans clustering with the Elbow method on the Wine dataset. The code used scikit-learn to perform KMeans clustering with different values of **k** and then used the Elbow method to determine the optimal number of

clusters based on the inertia values. The Elbow curve was plotted using matplotlib, and the optimal number of clusters was determined based on the point where the curve starts to level off. This code can be used as a starting point for performing KMeans clustering on other datasets and determining the optimal number of clusters using the Elbow method. The code shows that PSO can effectively find the optimal number of clusters for KMeans clustering on the Wine dataset and can improve the accuracy of the clustering result.