Silhouette Method — Better than Elbow Method to find Optimal Clusters

Deep dive analysis of Silhouette Method to find optimal clusters in k-Means clustering

Satyam Kumar Oct 18, 2020 · 6 min read ★

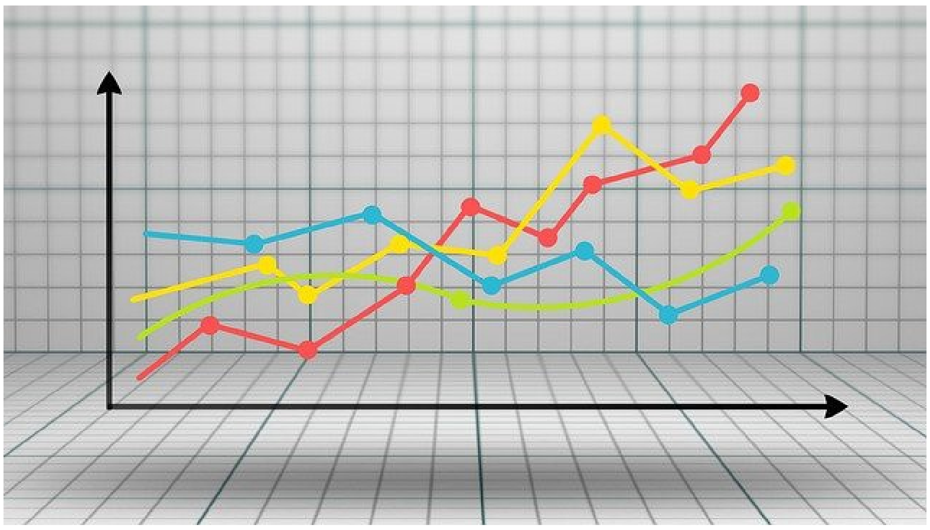


Image by Mediamodifier from Pixabay

Hyperparameters are model configurations properties that define the model and remain constants during the training of the model. The design of the model can be changed by tuning the hyperparameters. For K-Means clustering there are 3 main hyperparameters to set-up to define the best configuration of the model:

- Initial values of clusters
- Distance measures
- Number of clusters

Initial values of clusters greatly impact the clustering model, there are various algorithms to initialize the values. Distance measures are used to find points in clusters to the cluster center, different distance measures yield different clusters.

The number of clusters (**k**) is the most important hyperparameter in K-Means clustering. If we already know beforehand, the number of clusters to group the data into, then there is no use to tune the value of k. For example, k=10 for the MNIST digit classification dataset.

If there is no idea about the optimal value of k, then there are various methods to find the optimal/best value of k. In this article we will cover two such methods:

- **Elbow Method**
- **Silhouette Method**

Elbow Method:

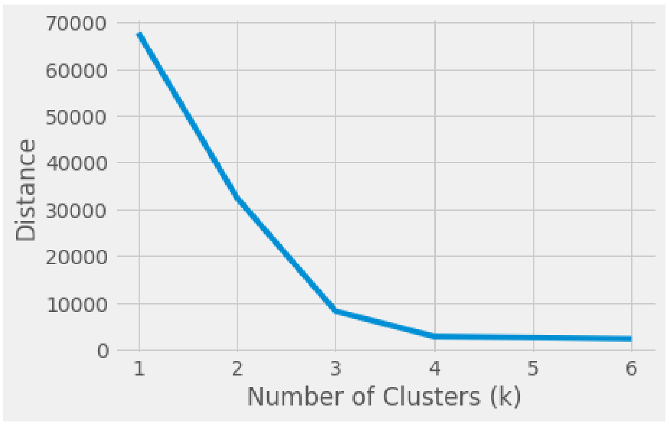
Elbow Method is an empirical method to find the optimal number of clusters for a dataset. In this method, we pick a range of candidate values of k, then apply K-Means clustering using each of the values of k. Find the average distance of each point in a cluster to its centroid, and represent it in a plot. Pick the value of k, where the **average distance falls suddenly**.

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

range_n_clusters = [1, 2, 3, 4, 5, 6]
avg_distance=[]
for n_clusters in range_n_clusters:
  clusterer = KMeans(n_clusters=n_clusters, random_state=42).fit(X)
  avg_distance.append(clusterer.inertia_)

style.use("fivethirtyeight")
plt.plot(range_n_clusters, avg_distance)
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Distance")
plt.show()
```

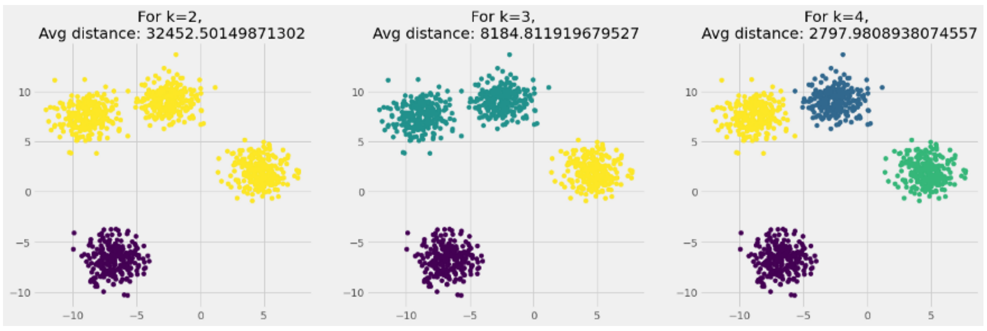
elbow_method.py hosted with ❤ by **GitHub** **view raw**



(Image by Author), Elbow Method to find optimal k

With an increase in the number of clusters (k), the average distance decreases. To find the optimal number of clusters (k), observe the plot and find the value of k for which there is a sharp and steep fall of the distance. This is will be an optimal point of k where an elbow occurs.

In the above plot, there is a sharp fall of average distance at **k=2, 3, and 4**. Here comes a confusion to pick the best value of k. In the below plot observe the clusters formed for **k=2, 3, and 4** with their average distance.



(Image by Author), Scatter plot of clusters formed at k=2, 3, and 4

This data is 2-D, so it's easy to visualize and pick the best value of k, which is k=4. For higher-dimensional data, we can employ the **Silhouette Method** to find the best k, which is a **better alternative to Elbow Method**.

. . .

Silhouette Method:

The **silhouette Method** is also a method to find the optimal number of clusters and interpretation and validation of consistency within clusters of data. The silhouette method computes silhouette coefficients of each point that measure how much a point is similar to its own cluster compared to other clusters. by providing a **succinct graphical representation** of how well each object has been classified.

> *Compute **silhouette coefficients** for each of point, and average it out for all the samples to get the **silhouette score**.*

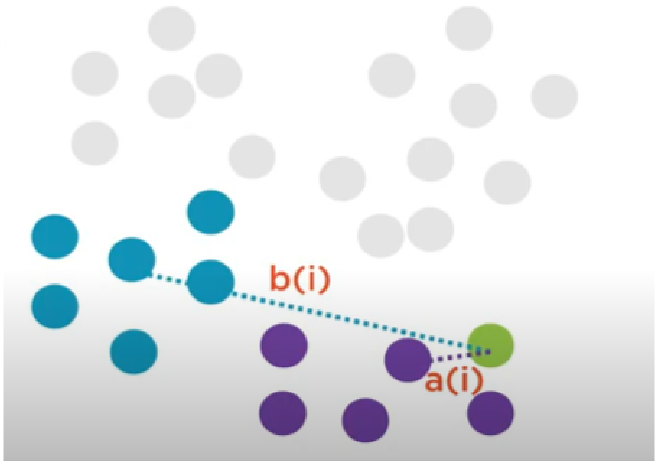
The silhouette value is a measure of how similar an object is to its own cluster (**cohesion**) compared to other clusters (**separation**). The value of the silhouette ranges between [1, -1], where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

Computing Silhoutte Coefficient:

Steps to find the silhouette coefficient of an i'th point:

1. Compute a(i): The average distance of that point with all other points in the same clusters.
2. Compute b(i): The average distance of that point with all the points in the closest cluster to its cluster.
3. Compute s(i) — silhouette coefficient or i'th point using below mentioned formula.

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



(Image by Author), Diagramatic representation of a(i) and b(i) from the above-mentioned formula to compute silhouette coefficient — s(i)

After computing the silhouette coefficient for each point, average it out to get the silhouette score.

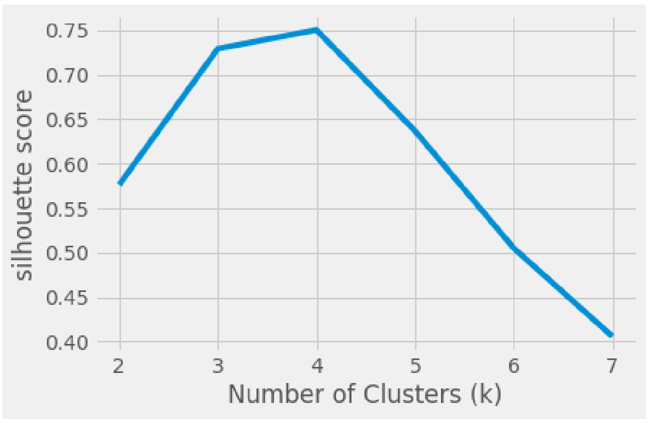
Silhouette Analysis:

Silhouette is a measure of how a clustering algorithm has performed. After computing the silhouette coefficient of each point in the dataset, plot it to get a visual representation of how well the dataset is clustered into k clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like the number of clusters visually. This measure has a range of [-1, 1].

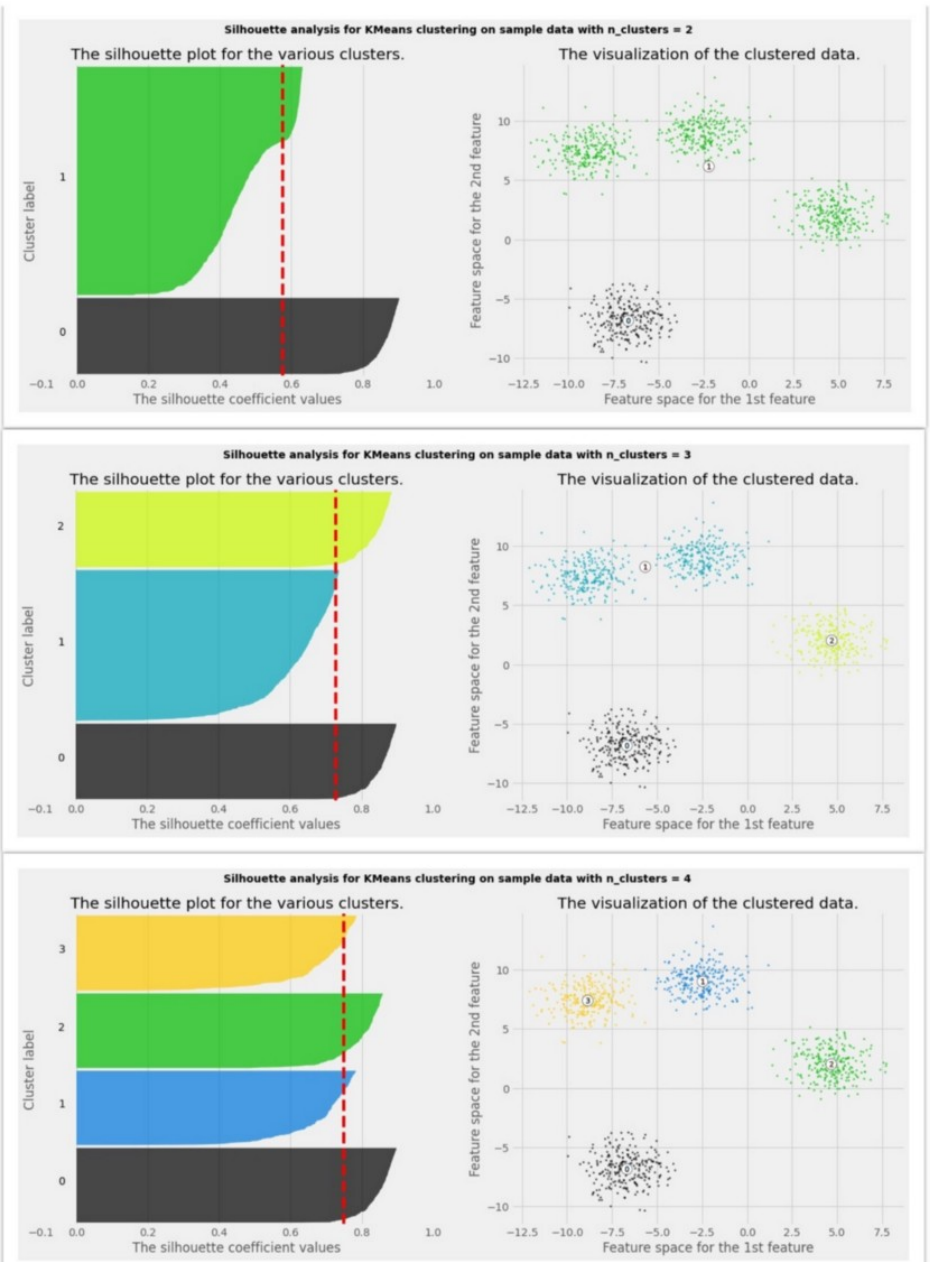
```
Important Points:
The Silhouette coefficient of +1 indicates that the sample is far
away from the neighboring clusters.
The Silhouette coefficient of 0 indicates that the sample is on or
very close to the decision boundary between two neighboring clusters.
Silhouette coefficient <0 indicates that those samples might have
been assigned to the wrong cluster or are outliers.
```

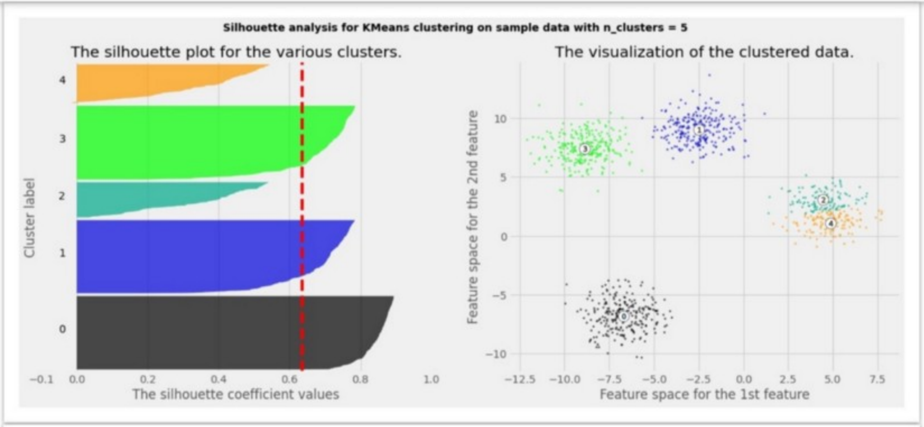
Find the optimal value of 'k' using Silhoutte Analysis:

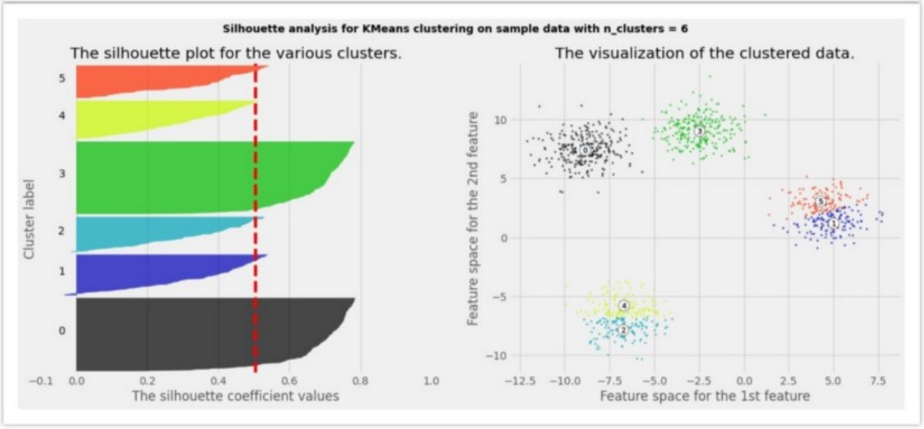
Similar to the previous Elbow method, we pick a range of candidate values of k (number of clusters), then train K-Means clustering for each of the values of k. For each k-Means clustering model represent the silhouette coefficients in a plot and observe the fluctuations and outliers of each cluster.



(Image by Author), Silhoutte score vs the number of clusters







(Image by Author), Silhoutte Analysis and scatter plot for each cluster in KMeans clustering on entire data with n_cluster=[2,3,4,5,6]

Observations from above Silhouette Plots:

- The silhouette plot shows that the n_cluster value of **3** is a bad pick, as all the points in the cluster with cluster_label=0 are below-average silhouette scores.
- The silhouette plot shows that the n_cluster value of **5** is a bad pick, as all the points in the cluster with cluster_label=2 and 4 are below-average silhouette scores.
- The silhouette plot shows that the n_cluster value of **6** is a bad pick, as all the points in the cluster with cluster_label=1,2,4 and 5 are below-average silhouette scores, and also due to the presence of outliers.
- Silhouette analysis is more ambivalent in deciding between 2 and 4.
- The thickness of the silhouette plot for the cluster with cluster_label=1 when n_clusters=2, is bigger in size owing to the grouping of the 3 sub-clusters into one big cluster.
- For n_clusters=4, all the plots are more or less of similar thickness and hence are of similar sizes, as can be considered as **best 'k'.**

. . .

Implementation:

```python
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
import matplotlib.style as style

range_n_clusters = [2, 3, 4, 5, 6]
silhouette_avg_n_clusters = []

for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(X)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    silhouette_avg_n_clusters.append(silhouette_avg)
    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')

    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()


style.use("fivethirtyeight")
plt.plot(range_n_clusters, silhouette_avg_n_clusters)
plt.xlabel("Number of Clusters (k)")
plt.ylabel("silhouette score")
plt.show()
```

Code Source, Edited by Author

. . .

Conclusion:

Elbow and Silhouette methods are used to find the optimal number of clusters. Ambiguity arises for the elbow method to pick the value of k. Silhouette analysis can be used to study the separation distance between the resulting clusters and can be considered a better method compared to the Elbow method.

Silhouette analysis also has added **advantage** to find the **outliers if present in a cluster**.

References:

[1] Wikipedia: Silhouette (clustering), (14 Sep 2020)

[2] Scikit Learn documentation: Selecting the number of clusters with silhouette analysis on KMeans clustering

. . .

Loved the article? Become a Medium member to continue learning without limits. I'll receive a small portion of your membership fee if you use the following link, with no extra cost to you.

Thank You for Reading

Artificial Intelligence · Machine Learning · Data Science · Education · Clustering