## Optimizing the number of centroids

- Centroids are difficult to interpret, and it can also be very difficult to determine whether we have the correct number of centroids.
- It's important to understand whether your data is unlabeled or not as this will directly influence the evaluation measures we can use.
- Evaluating the model performance for unsupervised techniques is a challenge.
- Consequently, sklearn has several methods to evaluate clustering when a ground truth is known, and very few for when it isn't.

### Getting ready

We'll start with a single cluster model and evaluate its similarity. This is more for the purpose of mechanics as measuring the similarity of one cluster count is clearly not useful in finding the ground truth number of clusters.

### Implementation

To get started we'll create several blobs that can be used to simulate clusters of data:

```
>>> from sklearn.datasets import make_blobs
>>> import numpy as np
>>> blobs, classes = make_blobs(500, centers=3)
>>>

>>> from sklearn.cluster import KMeans
>>> kmean = KMeans(n_clusters=3)
```

```
>>> kmean.fit(blobs)

KMeans(copy_x=True, init='k-means++', max_iter=300,
  n_clusters=3,
n_init=10, n_jobs=1, precompute_distances=True,
random_state=None, tol=0.0001, verbose=0)
```

**Silhouette Distance**

First, we'll look at silhouette distance.

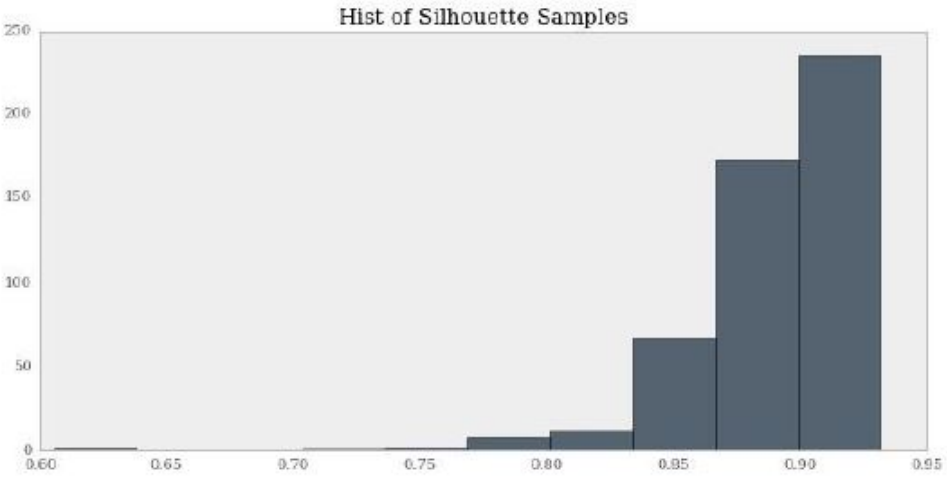
- Silhouette distance is the ratio of the difference between in-cluster dissimilarity, the closest out-of-cluster dissimilarity, and the maximum of these two values.
- It can be thought of as a measure of how separate the clusters are.

Let's look at the distribution of distances from the points to the cluster centers; it's useful to understand silhouette distances:

```
>>> from sklearn import metrics
>>> silho_samples = metrics.silhouette_samples(blobs,
kmean.labels_)
>>>
>>> np.column_stack((classes[:5], silho_samples[:5]))
array([[ 1., 0.87617292],
       [ 1., 0.89082363],
       [ 1., 0.88544994],
```

```
       [ 1., 0.91478369],
       [ 1., 0.91308287]])
>>> f, ax = plt.subplots(figsize=(10, 5))
>>> ax.set_title("Hist of Silhouette Samples")
>>> ax.hist(silho_samples)
```

The following is the output:




Notice that generally the higher the number of coefficients are closer to 1 (which is good) the better the score.

**How it works**

The average of the silhouette coefficients is often used to describe the entire model's fit:

```
>>> silho_samples.mean()
0.57130462953339578
```

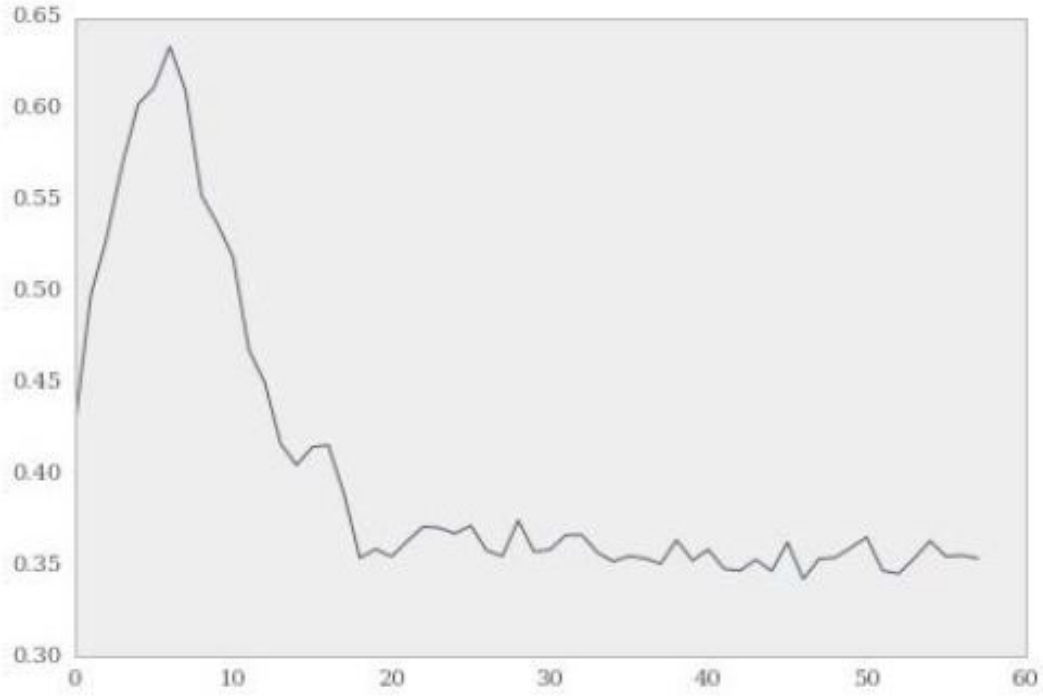
It's very common; in fact, the metrics module exposes a function to arrive at the value we just got:

```
>>> metrics.silhouette_score(blobs, kmean.labels_)
0.57130462953339578
```

Now, let's fit the models of several cluster counts and see what the average silhouette score looks like:

```
# first new ground truth
>>> blobs, classes = make_blobs(500, centers=10)
>>> sillhouette_avgs = []
# this could take a while
>>> for k in range(2, 60):
   kmean = KMeans(n_clusters=k).fit(blobs)
   sillhouette_avgs.append(metrics.silhouette_score(blobs,
   kmean.labels_))
>>> f, ax = plt.subplots(figsize=(7, 5))
>>> ax.plot(sillhouette_avgs)
```

The following is the output:




- This plot shows that the silhouette averages as the number of centroids increase.
- We can see that the optimum number, according to the data generating process, is 3, but here it looks like it's around 6 or 7.
- This is the reality of clustering; quite often, we won't get the correct number of clusters, we can only really hope to estimate the number of clusters to some approximation.