spectralclustering

September 6, 2023

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import scipy.cluster.hierarchy as shc
from scipy.linalg import eig
from sklearn import metrics
from sklearn.datasets import make_blobs, make_circles, make_moons,␣
 ↪make_gaussian_quantiles
from sklearn.cluster import SpectralClustering
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
```

0.0.1 common

[ ]:
```
def plt_histogram(x, bins=50):
  plt.hist(x, bins=bins)


def plt_scatter(x, y, labels):
  plt.scatter(x, y, c=labels)


def compare_clusters(data, true_labels=None, cluster_labels=None,␣
 ↪title_true="True clusters", title_cluster="Agglomerative Clustering"):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    ax1.scatter(data[:, 0], data[:, 1], c=true_labels)
    ax1.set_title(title_true)

    if cluster_labels is not None:
        ax2.scatter(data[:, 0], data[:, 1], c=cluster_labels)
        ax2.set_title(title_cluster)

    plt.show()
```

```
def get_scores(true_labels, pred_labels):
  rand_index = metrics.adjusted_rand_score(true_labels, pred_labels)
  mutual_info = metrics.adjusted_mutual_info_score(true_labels, pred_labels,␣
 ↪average_method='arithmetic')
  return rand_index, mutual_info


def get_normalized_laplacian(W):
  D = np.diag(np.sum(W, axis=1))
  I = np.eye(D.shape[0])
  Lraw = I - np.linalg.inv(D) @ W
  return Lraw


def get_k_lower_eigen(L, k=10):
  evalues, evectors = np.linalg.eig(L)
  index = evalues.argsort()
  lower_evalues, lower_evectors = evalues[index[:k]], evectors[:, index[:k]]
  return np.real(lower_evalues), np.real(lower_evectors)


def plot_eigen_values(values):
  x = np.arange(len(values))
  plt.title("eigenvalue")
  plt.plot(x, values, "*")


def plot_eigen_vectors(vectors, num_elems=10, k=5, figsize=(20, 5)):
  fig, ax = plt.subplots(1, k, figsize=figsize)
  for i in range(k):
    ax[i].plot(np.arange(num_elems), vectors[:num_elems, i], "b-")
    ax[i].set_title(f"eigenvector {i}")
```

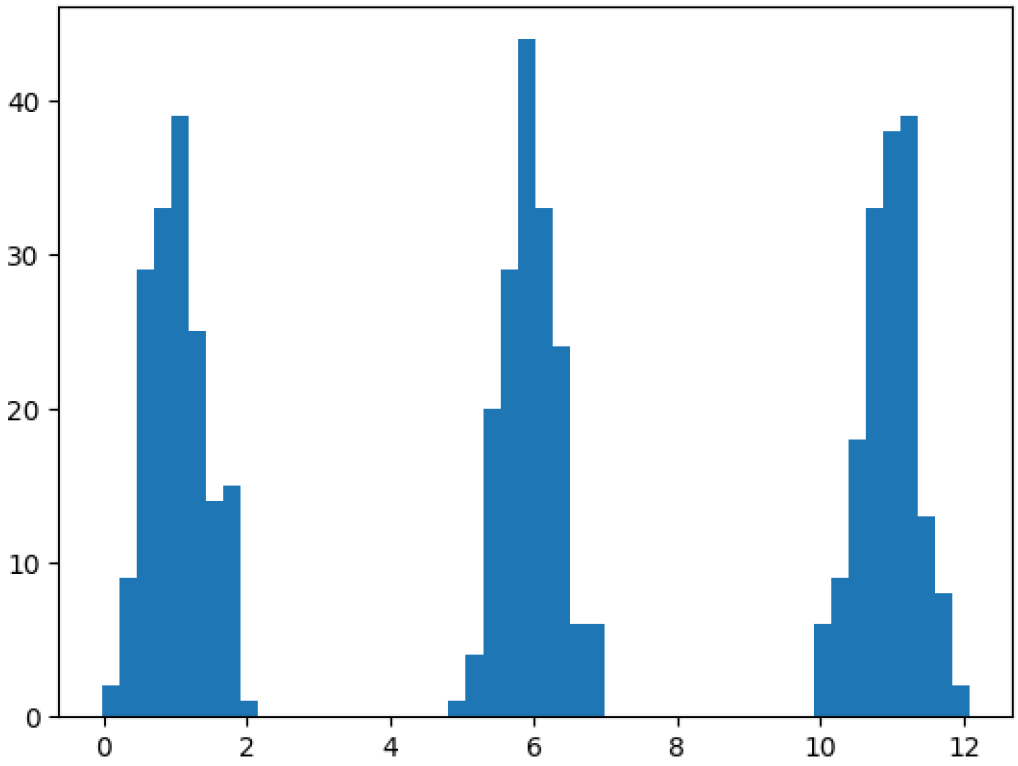
[ ]:

[ ]:
```
seed = 0
```

0.0.2 gaussian data

[ ]:
```
centers = [[1.], [6], [11.]]
gaus_data, gaus_true_labels = make_blobs(n_samples=500, n_features=1,␣
 ↪centers=centers, cluster_std=0.4, random_state=seed)

plt_histogram(gaus_data)
```



[ ]:

```
gaus_spec_cluster = SpectralClustering(n_clusters=4,␣
  ↪affinity="nearest_neighbors", n_neighbors=10, assign_labels="kmeans")
gaus_spec_cluster.fit(gaus_data)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/manifold/_spectral_embedding.py:274: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

[ ]:
```
SpectralClustering(affinity='nearest_neighbors', n_clusters=4)
```

[ ]:
```
rand_index, mutual_info = get_scores(gaus_true_labels, gaus_spec_cluster.
  ↪labels_)
print(f"rand_index: {rand_index:.3f}  mutual_info: {mutual_info:.3f}")
```

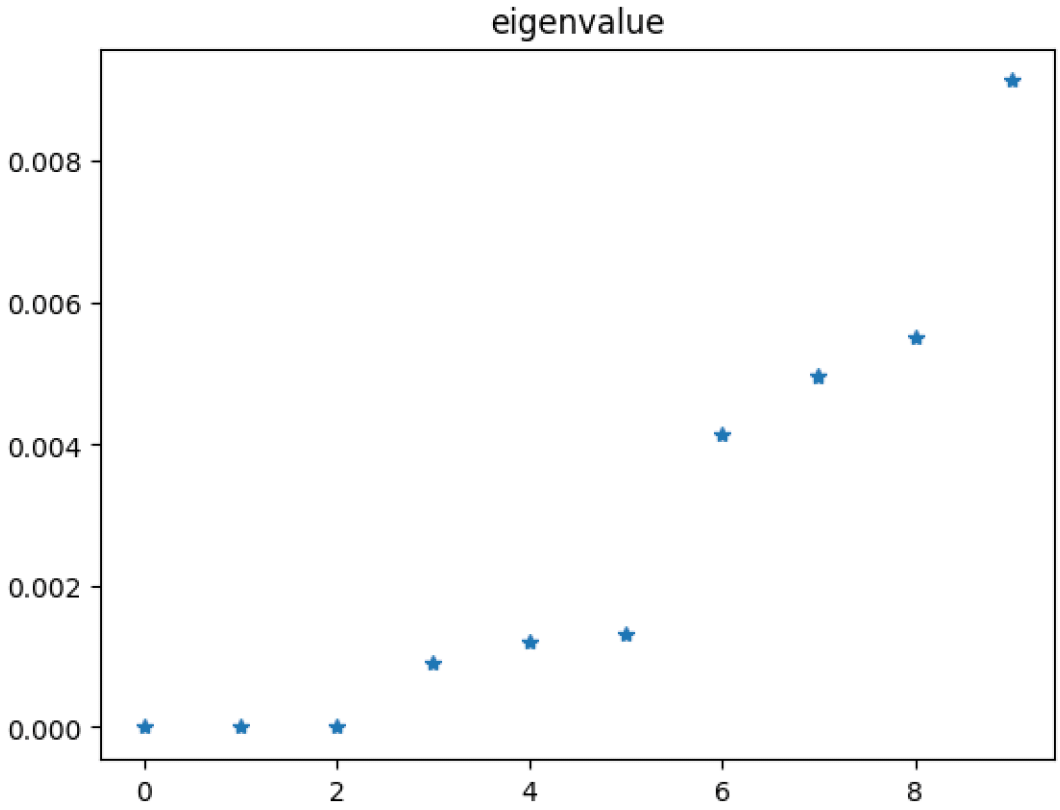
```
rand_index: 0.482  mutual_info: 0.657
```

[ ]:

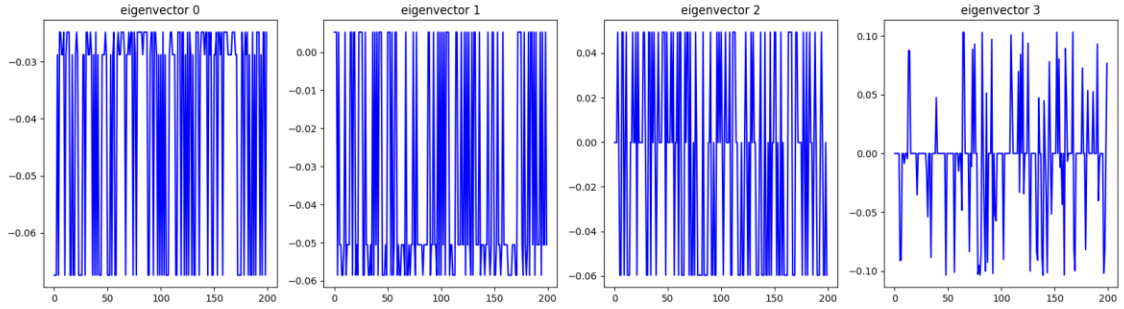
[ ]:
```
W = gaus_spec_cluster.affinity_matrix_.A
```

[ ]:
```
Lraw = get_normalized_laplacian(W)
evalues, evectors = get_k_lower_eigen(Lraw, k=10)
```

[ ]:
```
plot_eigen_values(evalues)
```



[ ]:
```
plot_eigen_vectors(evectors, num_elems=200, k=4)
```

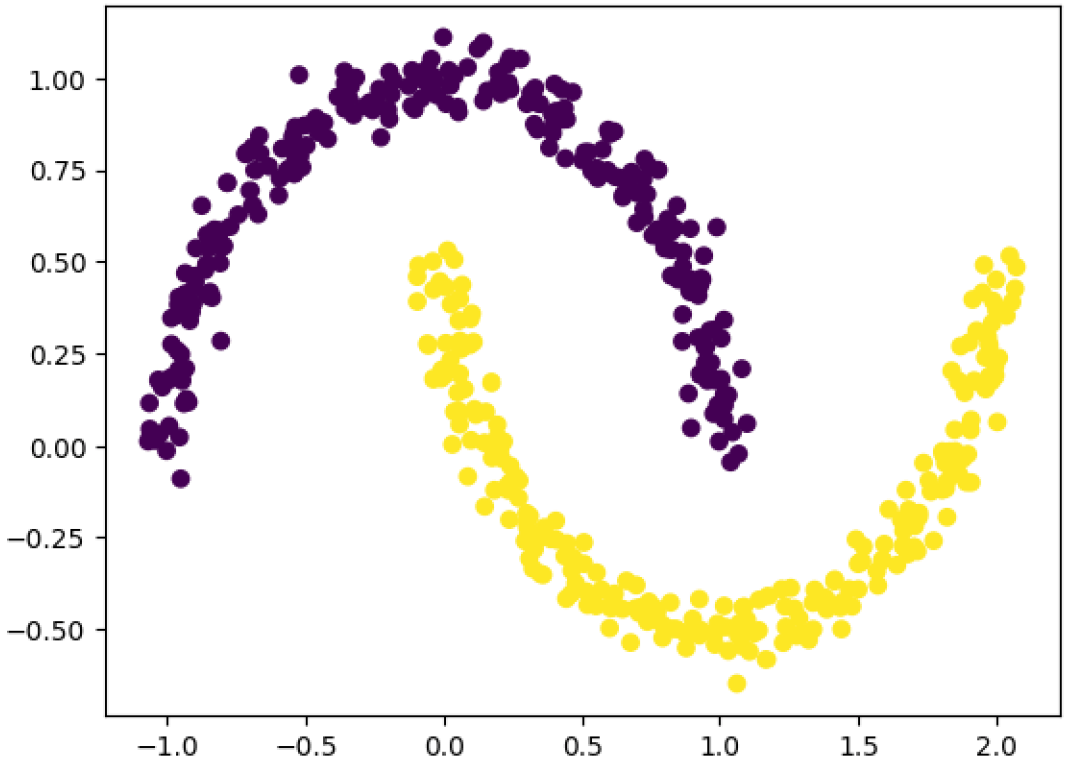


[ ]:

0.0.3 moon data

[ ]:
```
moons_data, moons_true_labels = make_moons(n_samples=500, noise=0.05,␣
↪random_state=seed)

plt_scatter(moons_data[:, 0], moons_data[:, 1], moons_true_labels)
```



[ ]:
```
moons_spec_cluster = SpectralClustering(n_clusters=2,␣
↪affinity="nearest_neighbors", n_neighbors=10, assign_labels="kmeans")
moons_spec_cluster.fit(moons_data)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/manifold/_spectral_embedding.py:274: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

[ ]:
```
SpectralClustering(affinity='nearest_neighbors', n_clusters=2)
```

[ ]:
```
rand_index, mutual_info = get_scores(moons_true_labels, moons_spec_cluster.
↪labels_)
print(f"rand_index: {rand_index:.3f}  mutual_info: {mutual_info:.3f}")
```

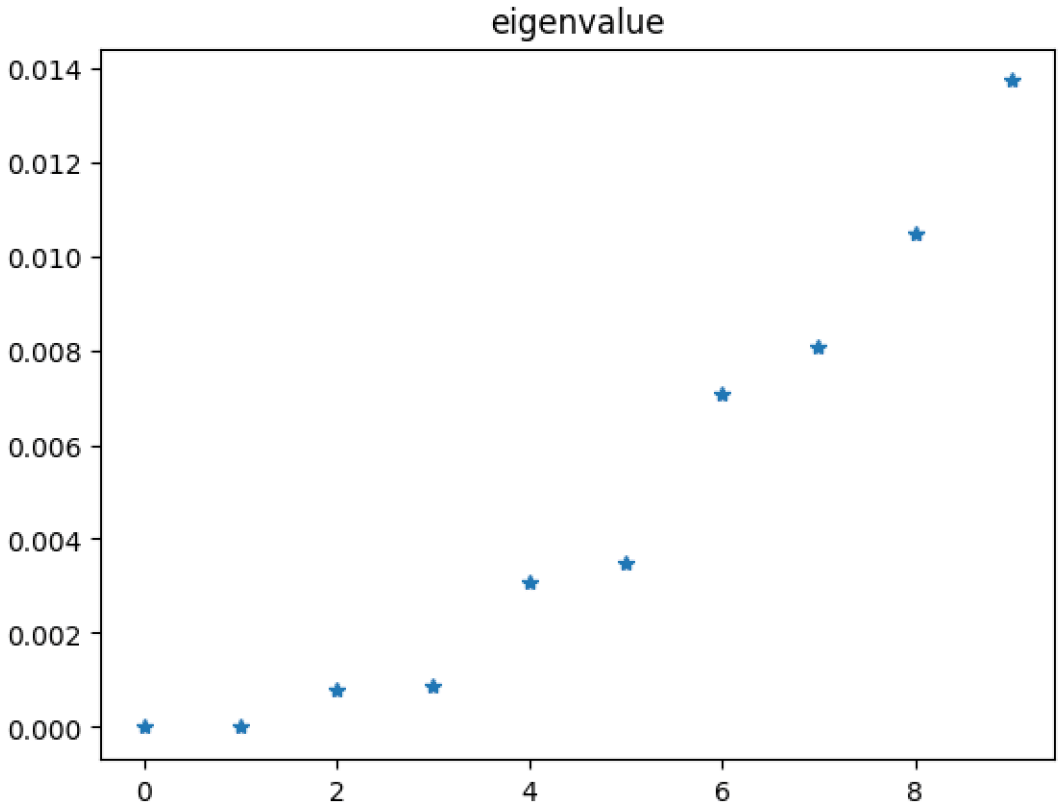
```
rand_index: 1.000  mutual_info: 1.000
```

[ ]:

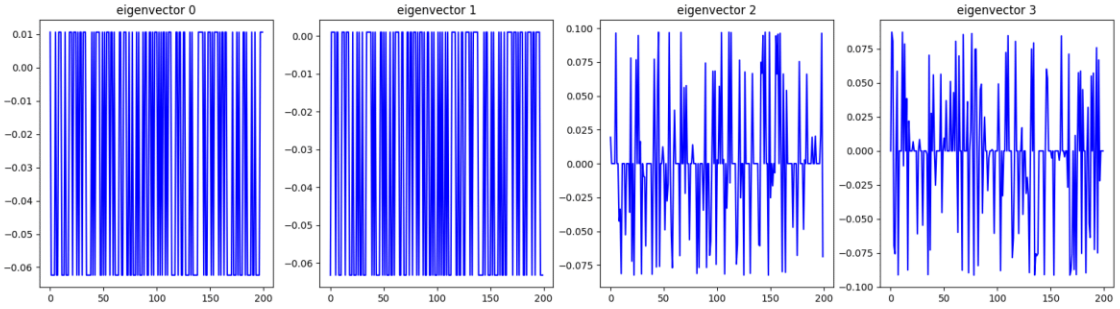
[ ]:
```
moons_W = moons_spec_cluster.affinity_matrix_.A
```

[ ]:
```
Lraw = get_normalized_laplacian(moons_W)
evalues, evectors = get_k_lower_eigen(Lraw, k=10)
```

[ ]:
```
plot_eigen_values(evalues)
```



[ ]:
```
plot_eigen_vectors(evectors, num_elems=200, k=4)
```

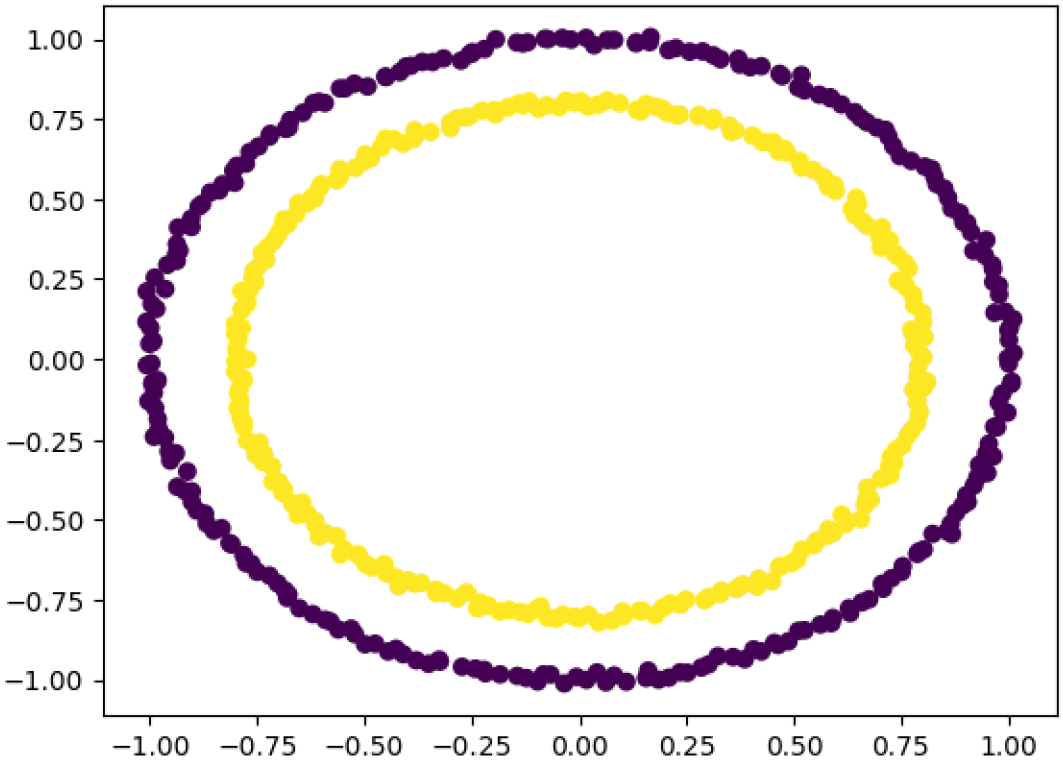


[ ]:

0.0.4 Circle data

[ ]:
```
circle_data, circle_true_labels = make_circles(n_samples=500, noise=.01,␣
↪random_state=seed)

plt_scatter(circle_data[:, 0], circle_data[:, 1], circle_true_labels)
```



[ ]:
```
circle_spec_cluster = SpectralClustering(n_clusters=2,␣
↪affinity="nearest_neighbors", n_neighbors=10, assign_labels="kmeans")
circle_spec_cluster.fit(circle_data)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/manifold/_spectral_embedding.py:274: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

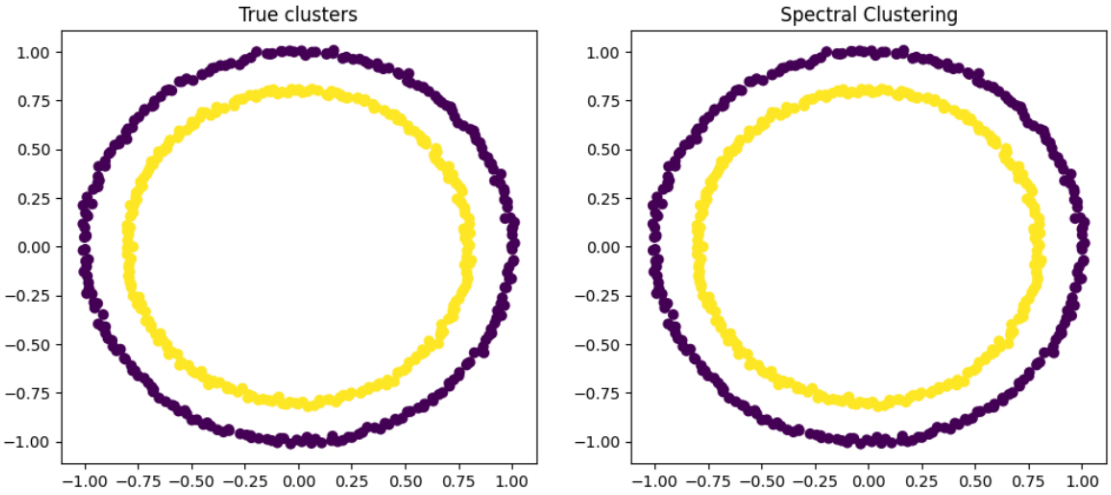
[ ]:
```
SpectralClustering(affinity='nearest_neighbors', n_clusters=2)
```

[ ]:
```
rand_index, mutual_info = get_scores(circle_true_labels, circle_spec_cluster.
↪labels_)
print(f"rand_index: {rand_index:.3f}  mutual_info: {mutual_info:.3f}")
```

```
rand_index: 1.000  mutual_info: 1.000
```

[ ]:

[ ]:
```
compare_clusters(circle_data,
                 true_labels=circle_true_labels,
                 cluster_labels=circle_spec_cluster.labels_,
                 title_true="True clusters",
                 title_cluster="Spectral Clustering")
```

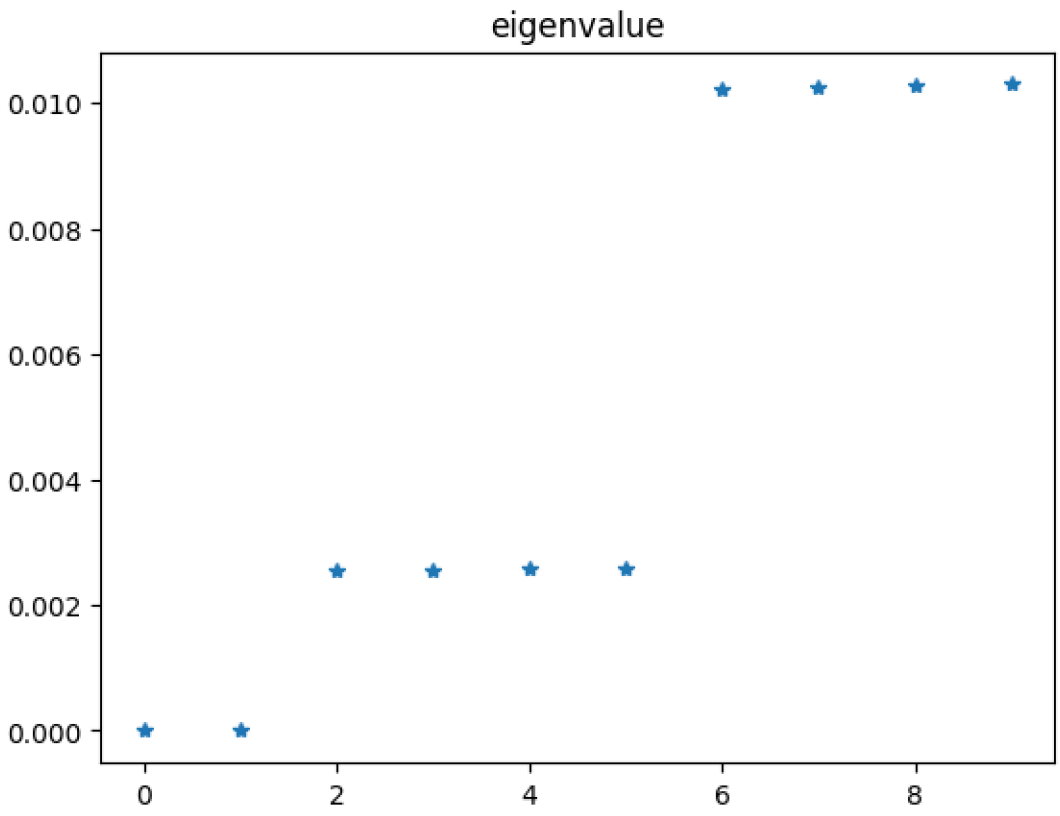


[ ]:

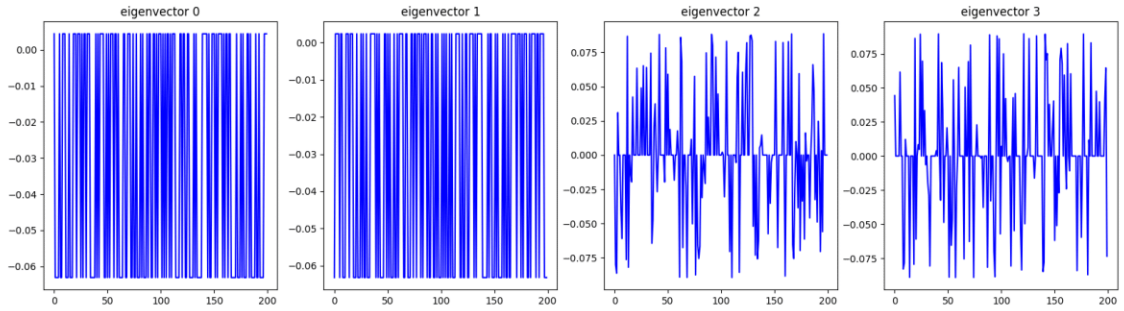
[ ]:
```
circle_W = circle_spec_cluster.affinity_matrix_.A
```

[ ]:
```
Lraw = get_normalized_laplacian(circle_W)
evalues, evectors = get_k_lower_eigen(Lraw, k=10)
```

[ ]:
```
plot_eigen_values(evalues)
```



[ ]:
```
plot_eigen_vectors(evectors, num_elems=200, k=4)
```



[ ]:
```

```