kmeans_l1

September 20, 2020

1 KMeans with norm L1

This demonstrates how results change when using norm L1 for a k-means algorithm.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

[2]:
```
%matplotlib inline
```

[3]:
```
import matplotlib.pyplot as plt
```

1.1 Simple datasets

[4]:
```
import numpy
import numpy.random as rnd
N = 1000
X = numpy.zeros((N * 2, 2), dtype=numpy.float64)
X[:N] = rnd.rand(N, 2)
X[N:] = rnd.rand(N, 2)
#X[N:, 0] += 0.75
X[N:, 1] += 1
X[:N//10, 0] -= 2
X.shape
```

[4]:
```
(2000, 2)
```

[5]:
```
fig, ax = plt.subplots(1, 1)
ax.plot(X[:, 0], X[:, 1], '.')
ax.set_title("Two squares");
```

1.2 Classic KMeans

It uses euclidean distance.

[6]:
```
from sklearn.cluster import KMeans
km = KMeans(2)
km.fit(X)
```

[6]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[7]:
```
km.cluster_centers_
```

[7]:
```
array([[0.25701017, 0.48916124],
       [0.51962715, 1.46900491]])
```

[8]:
```
def plot_clusters(km_, X, ax):
    lab = km_.predict(X)
    for i in range(km_.cluster_centers_.shape[0]):
        sub = X[lab == i]
        ax.plot(sub[:, 0], sub[:, 1], '.', label='c=%d' % i)
    C = km_.cluster_centers_
    ax.plot(C[:, 0], C[:, 1], 'o', ms=15, label="centers")
    ax.legend()

fig, ax = plt.subplots(1, 1)
plot_clusters(km, X, ax)
ax.set_title("L2 KMeans");
```

1.3 KMeans with L1 norm

[9]:
```
from mlinsights.mlmodel import KMeansL1L2
kml1 = KMeansL1L2(2, norm='L1')
kml1.fit(X)
```

[9]:
```
KMeansL1L2(algorithm='full', copy_x=True, init='k-means++', max_iter=300,
           n_clusters=2, n_init=10, n_jobs=None, norm='l1',
           precompute_distances='auto', random_state=None, tol=0.0001,
           verbose=0)
```

[10]:
```
kml1.cluster_centers_
```

[10]:
```
array([[0.41526759, 0.47949327],
       [0.51837524, 1.47608362]])
```

[11]:
```
fig, ax = plt.subplots(1, 1)
plot_clusters(kml1, X, ax)
ax.set_title("L1 KMeans");
```

1.4 When clusters are completely different

[12]:
```
N = 1000
X = numpy.zeros((N * 2, 2), dtype=numpy.float64)
X[:N] = rnd.rand(N, 2)
X[N:] = rnd.rand(N, 2)
#X[N:, 0] += 0.75
X[N:, 1] += 1
X[:N//10, 0] -= 4
X.shape
```

[12]:
```
(2000, 2)
```

[13]:
```
km = KMeans(2)
km.fit(X)
```

[13]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[14]:
```
kml1 = KMeansL1L2(2, norm='L1')
kml1.fit(X)
```

[14]:
```
KMeansL1L2(algorithm='full', copy_x=True, init='k-means++', max_iter=300,
           n_clusters=2, n_init=10, n_jobs=None, norm='l1',
           precompute_distances='auto', random_state=None, tol=0.0001,
           verbose=0)
```

[15]:
```
fig, ax = plt.subplots(1, 2, figsize=(10, 4))
plot_clusters(km, X, ax[0])
plot_clusters(kml1, X, ax[1])
ax[0].set_title("L2 KMeans")
ax[1].set_title("L1 KMeans");
```

[16]:
```

```