アルゴリズム

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from sklearn.metrics import silhouette_samples
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
import pandas as pd
import seaborn as sns
```

In [2]:
```
data_wine = load_wine()
X = data_wine.data
y = data_wine.target
```

In [16]:
```
wine_df=pd.DataFrame(X,columns = data_wine.feature_names )
wine_df['target']=y
print(wine_df)
sns.pairplot(wine_df,hue = 'target')
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  ¥
0      14.23        1.71  2.43               15.6      127.0           2.80
1      13.20        1.78  2.14               11.2      100.0           2.65
2      13.16        2.36  2.67               18.6      101.0           2.80
3      14.37        1.95  2.50               16.8      113.0           3.85
4      13.24        2.59  2.87               21.0      118.0           2.80
..       ...         ...   ...                ...        ...            ...
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  ¥
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          ...                   ...              ...              ...   ...
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
176        0.68                  0.53             1.46             9.30  0.60
177        0.76                  0.56             1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline  target
0                            3.92   1065.0       0
1                            3.40   1050.0       0
2                            3.17   1185.0       0
3                            3.45   1480.0       0
4                            2.93    735.0       0
..                            ...      ...     ...
173                          1.74    740.0       2
174                          1.56    750.0       2
175                          1.56    835.0       2
176                          1.62    840.0       2
177                          1.60    560.0       2

[178 rows x 14 columns]
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x132c02eb348>
```

In [3]:

```
#kmeans
km = KMeans(n_clusters=3)
y_km = km.fit_predict(X)
df = pd.DataFrame({'y_km':y_km})
df['y_wine'] = y

print( pd.crosstab(df['y_km'],df['y_wine']) )
```

```
y_wine   0   1   2
y_km
0        0  50  19
1       46   1   0
2       13  20  29
```

In [17]:

```
#kmeans シルエット図
cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
```

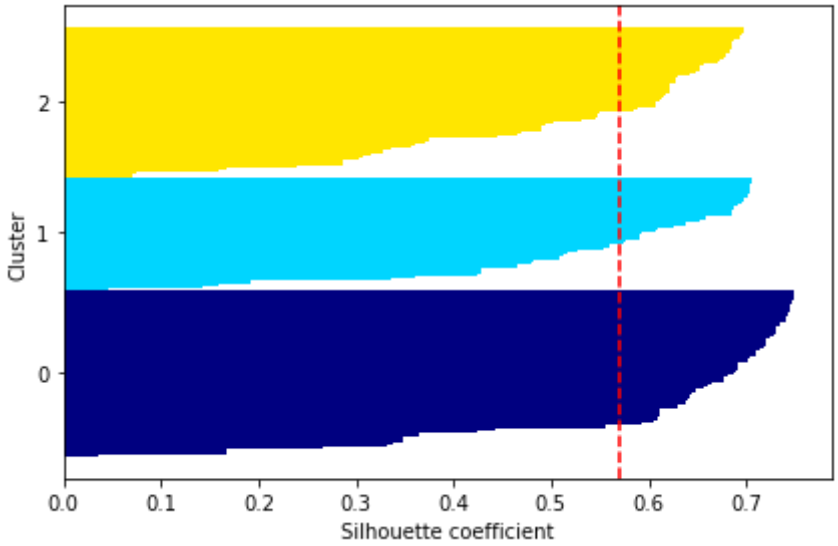
```
                edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels )
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
plt.show()
```



In [5]:

```
#K近傍法
from sklearn.neighbors import KNeighborsClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
knc = KNeighborsClassifier(n_neighbors=10).fit(X_train, y_train)
y_test_knc = knc.predict(X_test)
df2 = pd.DataFrame({'y_knc':y_test_knc})
df2['y_wine'] = y_test
print( pd.crosstab(df2['y_knc'],df2['y_wine']) )
```

```
y_wine   0   1  2
y_knc
0       14   1  0
1        0  12  6
2        4   4  4
```

Kmeansを用いたデータのクラスタリングと評価を行った。
データの特徴マップをみると、閾値で切るタイプの方がきれいに分類できそうな雰囲気もあるが、実際にアルゴリズムにかけるとそれなりに分類できたが、元のデータのクラス２（分類後のクラス２）はあまり良好ではない印象がある。
そういう意味で言うと、教師がある場合は素直に使った方がいいといえるかもしれない。
シルエット図を見ると、0は比較的シルエット係数が平均値を超えている。しかし、1,2は半数程度シルエット係数が低く、モデルとしては良好とはいえないかもしれない。
K近傍法での分類も行った。利用する近傍データの数によって分類のされ方が変わることも確認できた。
こちらの結果からもわかるように、元のデータのクラス２（分類後のクラス２）は他のクラスと空間的に近いことがわかる。

実際に教師データがないとしたら、既知の知見から特徴量を絞るといった工夫が必要かもしれない。