zero-base /

Life Changing Education

Chapter 24. HAR using PCA

HAR data

zero-base /

HAR 데이터 읽기

```
import pandas as pd

url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/'+\
                                    'master/dataset/HAR_dataset/features.txt'
feature_name_df = pd.read_csv(url, sep='\s+', header=None,
                              names=['column_index','column_name'])
feature_name = feature_name_df.iloc[:, 1].values.tolist()
```

HAR data


```
X_train_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/'+\
                                'master/dataset/HAR_dataset/train/X_train.txt'
X_test_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/' +\
                                'master/dataset/HAR_dataset/test/X_test.txt'


X_train = pd.read_csv(X_train_url, sep='\s+', header=None)
X_test = pd.read_csv(X_test_url, sep='\s+', header=None)
```

```
X_train.columns = feature_name
X_test.columns = feature_name
```

HAR data


```
y_train_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/' +\
                                'master/dataset/HAR_dataset/train/y_train.txt'
y_test_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/' +\
                                'master/dataset/HAR_dataset/test/y_test.txt'

y_train = pd.read_csv(y_train_url, sep='\s+', header=None, names=['action'])
y_test = pd.read_csv(y_test_url, sep='\s+', header=None, names=['action'])
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((7352, 561), (2947, 561), (7352, 1), (2947, 1))
```

zero-base /

재사용을 위해 함수로 만들고 (이전 것을 복사)

```
from sklearn.decomposition import PCA

def get_pca_data(ss_data, n_components=2):
    pca = PCA(n_components=n_components)
    pca.fit(ss_data)

    return pca.transform(ss_data), pca
```

zero-base /

PCA fit

```
HAR_pca, pca = get_pca_data(X_train, n_components=2)
HAR_pca.shape
```

```
(7352, 2)
```

```
pca.mean_.shape, pca.components_.shape
```

```
((561,), (2, 561))
```

zero-base /

조금 더 재미있는 함수로 만들기 위해

```
cols = ['pca_'+str(n) for n in range(pca.components_.shape[0])]
cols
```

```
['pca_0', 'pca_1']
```

zero-base /

PCA 결과를 저장하는 함수

```
def get_pd_from_pca(pca_data, col_num):
    cols = ['pca_'+str(n) for n in range(col_num)]
    return pd.DataFrame(pca_data, columns=cols)
```

zero-base /

components 2개

```
HAR_pca, pca = get_pca_data(X_train, n_components=2)
HAR_pd_pca = get_pd_from_pca(HAR_pca, pca.components_.shape[0])
HAR_pd_pca['action'] = y_train
HAR_pd_pca.head()
```

	pca_0	pca_1	action
0	-5.520280	-0.290278	5
1	-5.535350	-0.082530	5
2	-5.474988	0.287387	5
3	-5.677232	0.897031	5
4	-5.748749	1.162952	5

zero-base /

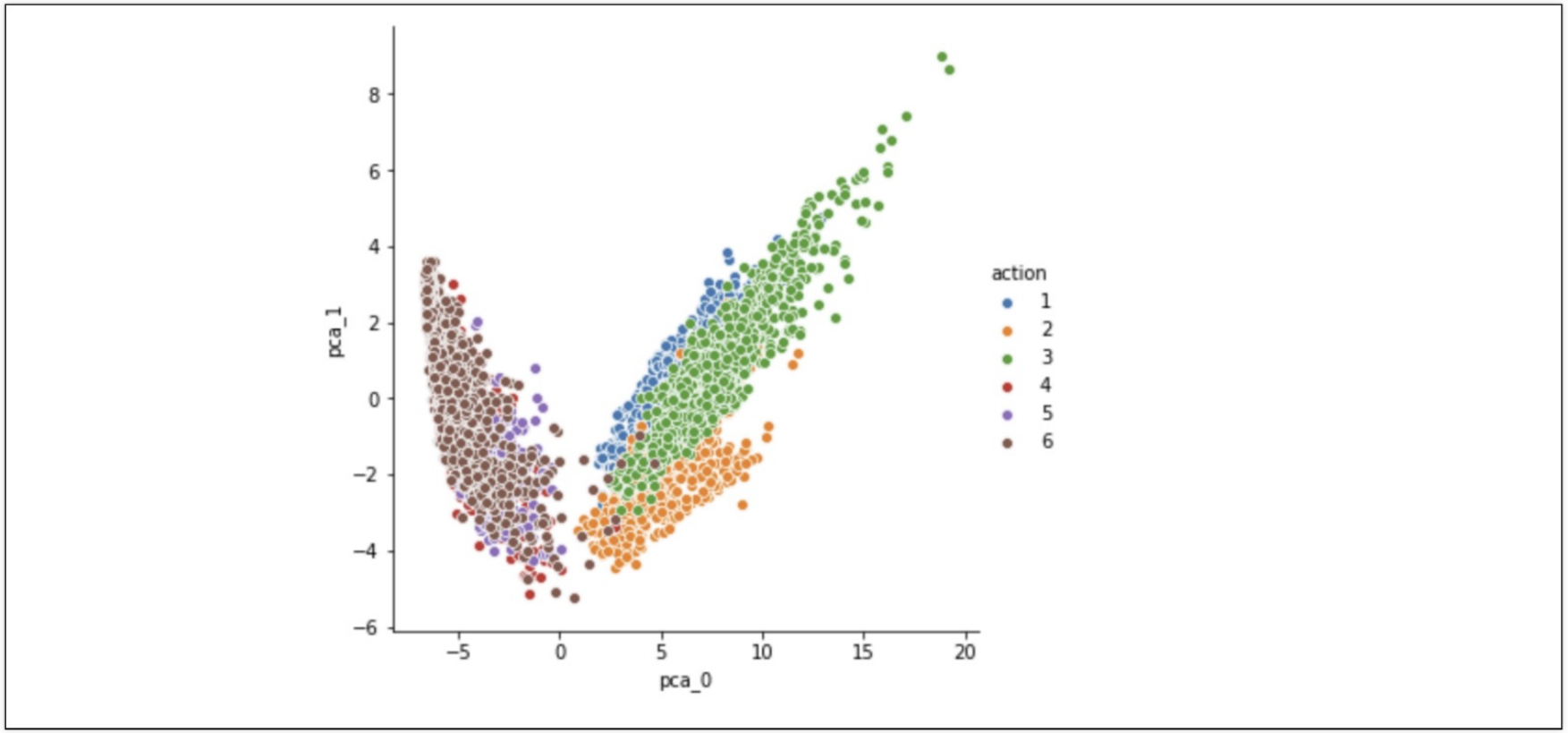
어떤가?

```
import seaborn as sns

sns.pairplot(HAR_pd_pca, hue='action', height=5,
             x_vars=['pca_0'], y_vars=['pca_1']);
```

zero-base /

썩 성능이...



zero-base /

전체 500개가 넘는 특성을 단 두개로 줄이면 이정도~

```
import numpy as np

def print_variance_ratio(pca):
    print('variance_ratio: ', pca. explained_variance_ratio_)
    print('sum of variance_ratio: ', np.sum(pca.explained_variance_ratio_))

print_variance_ratio(pca)
```

```
variance_ratio:  [0.6255444  0.04913023]
sum of variance_ratio:  0.6746746270487932
```

zero-base /

컴포넌트 3개는?

```
HAR_pca, pca = get_pca_data(X_train, n_components=3)
HAR_pd_pca = get_pd_from_pca(HAR_pca, pca.components_.shape[0])
HAR_pd_pca['action'] = y_train

print_variance_ratio(pca)
```

```
variance_ratio:  [0.6255444  0.04913023 0.04121467]
sum of variance_ratio:  0.7158893015785993
```

zero-base /

컴포넌트 10개 정도?

```
HAR_pca, pca = get_pca_data(X_train, n_components=10)
HAR_pd_pca = get_pd_from_pca(HAR_pca, pca.components_.shape[0])
HAR_pd_pca['action'] = y_train

print_variance_ratio(pca)
```

```
variance_ratio:  [0.6255444  0.04913023 0.04121467 0.01874956 0.0169486  0.0127
2069
 0.01176685 0.01068973 0.00969378 0.00858017]
sum of variance_ratio:  0.8050386858978642
```

zero-base /

혹시 이 결과가 시간이 길게 나왔다면 지금은?

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

params = {
    'max_depth' : [ 6, 8 ,10],
    'n_estimators' : [50, 100, 200],
    'min_samples_leaf' : [8, 12],
    'min_samples_split' : [8, 12]
}

rf_clf = RandomForestClassifier(random_state=13, n_jobs=-1)
grid_cv = GridSearchCV(rf_clf, param_grid=params, cv=2, n_jobs=-1)
grid_cv.fit(HAR_pca, y_train.values.reshape(-1,))
```

zero-base /

성능은 조금 나쁘다

```
cv_results_df = pd.DataFrame(grid_cv.cv_results_)
target_col = ['rank_test_score', 'mean_test_score', 'param_n_estimators',
              'param_max_depth']
cv_results_df[target_col].sort_values('rank_test_score').head()
```

	rank_test_score	mean_test_score	param_n_estimators	param_max_depth
35	1	0.838547	200	10
32	1	0.838547	200	10
14	3	0.837595	200	8
17	3	0.837595	200	8
26	5	0.837323	200	10

zero-base /

best 파라미터

```
grid_cv.best_params_
```

```
{'max_depth': 10,
 'min_samples_leaf': 12,
 'min_samples_split': 8,
 'n_estimators': 200}
```

```
grid_cv.best_score_
```

```
0.8385473340587595
```

zero-base /

테스트 데이터에 적용해보자

```
from sklearn.metrics import accuracy_score

rf_clf_best = grid_cv.best_estimator_
rf_clf_best.fit(HAR_pca, y_train.values.reshape(-1,))

pred1 = rf_clf_best.predict(pca.transform(X_test))

accuracy_score(y_test , pred1)
```

```
0.8551068883610451
```

zero-base /

시간이 많이 걸렸던 xgboost는?

```
import time
from xgboost import XGBClassifier

evals = [(pca.transform(X_test), y_test)]

start_time = time.time()
xgb = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgb.fit(HAR_pca, y_train.values.reshape(-1,),
        early_stopping_rounds=10, eval_set=evals)
print('Fit time : ', time.time() - start_time)
```

zero-base /

시간과 성능의 trade-off가 필요하다면

```
[149]   validation_0-merror:0.13539
[150]   validation_0-merror:0.13607
[151]   validation_0-merror:0.13607
[152]   validation_0-merror:0.13607
[153]   validation_0-merror:0.13539
[154]   validation_0-merror:0.13641
[155]   validation_0-merror:0.13641
Stopping. Best iteration:
[145]   validation_0-merror:0.13539

Fit time :  4.580002069473267
```

zero-base /

accuracy

```
accuracy_score(y_test, xgb.predict(pca.transform(X_test)))
```

```
0.8646080760095012
```