

Ex 1: Breast cancer

Sử dụng tập dữ liệu ung thư, một vấn đề phân loại nhiều lớp rất nổi tiếng. Số liệu này được tính toán từ một hình ảnh số hóa của FNA về ung thư vú. Chúng mô tả các đặc điểm của nhân tế bào có trong hình ảnh.

Dữ liệu này có hai loại ung thư: ác tính (có hại) và lành tính (không có hại). Ta có thể xây dựng một mô hình để phân loại loại ung thư.

Cho dữ liệu breast_cancer nằm trong sklearn.datasets

Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán PCA và SVM để thực hiện việc dự đoán có bị ung thư hay không dựa trên thông tin được cung cấp

1. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
2. Áp dụng thuật toán PCA & SVM
3. Tìm kết quả
4. Kiểm tra độ chính xác
5. Đo thời gian thực hiện thuật toán, nhận xét thời gian và độ chính xác so với việc chỉ sử dụng SVN

```
In [1]: # https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\_breast\_cancer.html
```

```
In [2]: import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

```
In [3]: import datetime
x1 = datetime.datetime.now()
print(x1)
```

2020-10-14 15:48:47.704755

```
In [4]: cancer = datasets.load_breast_cancer()
```

```
In [5]: # print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer ('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

Features: ['mean radius' 'mean texture' 'mean perimeter' 'mean area' 'mean smoothness' 'mean compactness' 'mean concavity' 'mean concave points' 'mean symmetry' 'mean fractal dimension' 'radius error' 'texture error' 'perimeter error' 'area error' 'smoothness error' 'compactness error' 'concavity error' 'concave points error' 'symmetry error' 'fractal dimension error' 'worst radius' 'worst texture' 'worst perimeter' 'worst area' 'worst smoothness' 'worst compactness' 'worst concavity' 'worst concave points' 'worst symmetry' 'worst fractal dimension']

Labels: ['malignant' 'benign']

```
In [6]: cancer.data.shape
```

Out[6]: (569, 30)

```
In [7]: # print the cancer data features (top 3 records)
print(cancer.data[0:3])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
 7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
 5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
 2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
 1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
 6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
 2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
 3.613e-01 8.758e-02]]
```

```
In [8]: # Class: có giá trị là 0 và 1
X = cancer.data
```



```

In [9]: X[:, :3]

Out[9]: array([[ 17.99,  10.38, 122.8 ],
               [ 20.57,  17.77, 132.9 ],
               [ 19.69,  21.25, 130.   ],
               ...,
               [ 16.6  ,  28.08, 108.3 ],
               [ 20.6  ,  29.33, 140.1 ],
               [  7.76,  24.54,  47.92]])

In [10]: # print the cancer labels (0:malignant, 1:benign)
y = cancer.target
y[:5]

Out[10]: array([0, 0, 0, 0, 0])

In [11]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=109) # 70% training and 30% test

In [12]: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

# Apply transform to both the training set and the test set.
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

In [13]: from sklearn.decomposition import PCA

In [14]: # Make an instance of the Model
pca = PCA(.95)

In [15]: pca.fit(X_train)

Out[15]: PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=None,
            svd_solver='auto', tol=0.0, whiten=False)

In [16]: pca.n_components_

Out[16]: 10

In [17]: # Apply the mapping (transform) to both the training set and the test set.
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)

In [18]: clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)

Out[18]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
            kernel='linear', max_iter=-1, probability=False, random_state=None,
            shrinking=True, tol=0.001, verbose=False)

In [19]: y_pred = clf.predict(X_test)

In [20]: y_pred

Out[20]: array([1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
                1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
                1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
                1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,
                0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
                1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1])

In [21]: from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test, y_pred)*100, "%")

Accuracy is  98.24561403508771 %

In [22]: from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

[[ 60  3]
 [ 0 108]]

```

	precision	recall	f1-score	support
0	1.00	0.95	0.98	63
1	0.97	1.00	0.99	108
accuracy			0.98	171
macro avg	0.99	0.98	0.98	171
weighted avg	0.98	0.98	0.98	171


```
In [23]: from sklearn import metrics
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:", metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:", metrics.recall_score(y_test, y_pred))
```

Precision: 0.972972972972973
Recall: 1.0

```
In [24]: x2 = datetime.datetime.now()
print(x2)
```

2020-10-14 15:48:47.868597

```
In [25]: d = x2 - x1
```

```
In [26]: print(d)
```

0:00:00.163842

```
In [27]: plt.figure(figsize=(8,6))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
```

Out[27]: Text(0, 0.5, 'Cumulative explained variance')

