Assignment 4

2015313254 노인호

November 17th 2019

1 Progress

1.1 Problem

붓꽃(Iris)의 데이터를 이용해서 PCA(principal component analysis)와, T-SNE(t-distributed Stochastic Neighbor Embedding)을 수행하고 시각화를 하는 것이 문제이다. PCA와 T-SNE 모두 많은 차원에서 의미있는 정보만을 선택하여 차원축소를 진행하고 축소된 결과의 시각화를 하는데 이용된다. 첫번째로 PCA를 수행하고 두번째로 T-SNE를 수행한다.

1.2 Data Load

scikit-learn에 있는 iris데이터를 'load_iris()'를 이용하여 로드한다.

1.3 Data preprocessing

(1) iris data를 학습시키기 위한 입력변수와 종속변수는 다음과 같게 된다.

- input variables(x) : sepal length, sepal width, petal length, petal width
- output variables(y) : species (setosa, versicolor, virginica)

(2) 그래프를 그려서 시각화하기 위해 데이터 프레임 형태로 변환시켜준다. 컬럼에 4가지 feature가 들어가고 마지막 5번째 컬럼에는 target이 들어간다. iris data는 target값이 0,1,2 로 나타내어지는데 (각각 setosa, versicolor, virginica) 이것을 map을 이용해서 원래 품종이름으로 바꿔서 최종 데이터프레임을 구성한다.

(3) PCA와 TSNE를 이용해서 model 을 fit하는 과정에서 y데이터는 필요가 없다. 데이터프레임에서 x에 해당하는 데이터를 X로 지정해준다.

(4) data scaling은 데이터를 정규화 시켜주는 StandardScaler함수를 사용하였다. 평균이 0이 되는 스케일링을 사용하는 것이 중요하다.

1.4 PCA and hyperparameter

Scikit-learn의 PCA함수를 사용하였다. sklearn.decomposition에 있다. fit과 transform과정 후 나온 데이터를 데이터프레임화 시켜준다. 가장 큰 eigenvalues값 즉 가장 큰 분산값을 가지는 eigenvector를 찾고 그것을 principal component1, 그 다음 eigenvector를 principal component2로 정하게 된다.또한 principal component 2는 1과 수직이 되는 eigenvector이다.

- 시각화를 위해 n_component=2로 설정했다.
- 그 이외의 hyperparameter는 기본값을 사용하였다.

1.5 T-SNE and hyperparameter

Scikit-learn의 TSNE 함수를 사용하였다. sklearn.manifold에 있다. fit과 transform과정을 따로 거칠수 없고(t-sne는 새로운 데이터에 대해 transform 과정을 지원해주지 않는다.) fit_transform 통해 나온 데이터를 데이터프레임화 시켜준다. 또한 PCA는 저차원의 공간에 사영하는 개념을 사용하지만 TSNE는 저 차원의 manifold라는 개념을 사용한다.

- 시각화를 위해 n_component=2로 설정했다.(기본값)
- 그 이외의 hyperparameter는 기본값을 사용하였다. (기본값 설정도 잘 작동하는 편이다.)

2 Conclusion

plt.text를 이용해서 데이터를 텍스트 형식으로 시각화하였다. PCA의 경우 pc1 과 pc2가 각각 전체 분산의 72.9%, 22.9%를 설명하여 4개를 2개로 축소시키더라도 전체 분산의 약 96%를 설명함을 알 수 있었다. 그래프 시각화 결과는 gist 코드를 통해 볼 수 있도록 하였다.

3 Python code in jupyter notebook

웹사이트 https://nbviewer.jupyter.org/ 에서 다음의 gist 주소를 입력하면

- https://gist.github.com/nosy0411/b94ad67e444e4790f8701668a3bff330

assignment4.ipynb 파일의 코드를 볼 수 있다.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris_dataset = load_iris()

print("Target names:", iris_dataset['target_names'])
print("Feature names:\n", iris_dataset['feature_names'])
print("Type of data:", type(iris_dataset['data']))
print("Shape of data:", iris_dataset['data'].shape)
print("==========================")
print("Type of target:", type(iris_dataset['target']))
print("Shape of target:", iris_dataset['target'].shape)

import pandas as pd
import numpy as np
feature_columns = ['sepal length', 'sepal width', 'petal length', 'petal width']
df=pd.DataFrame(data=np.c_[iris_dataset['data'],iris_dataset['target']], columns=iris_dataset['feature_
df['target']=df['target'].map({0:'setosa',1:"versicolor",2:"virginica"})
X=df.iloc[:,:-1].values
y=df.iloc[:,[-1]].values
print(type(X), type(y))

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X_scaled=scaler.transform(X)

from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(X_scaled)
X_pca=pca.transform(X_scaled)

temp_df = pd.DataFrame(data = X_pca, columns = ['principal component 1', 'principal component 2'])

pca_df = pd.concat([temp_df, df[['target']]], axis = 1)
print(pca_df.head(10))
print(pca.explained_variance_ratio_)

import matplotlib.pyplot as plt
# targets = ['setosa', 'versicolor', 'virginica']
# colors = ['g', 'b', 'r']
# plt.xlabel('Principal Component 1')
# plt.ylabel('Principal Component 2')
# plt.title('PCA of 2 component about iris dataset')
# for target, color in zip(targets,colors):
#     index = data_df['target'] == target
#     plt.scatter(data_df.loc[index, 'principal component 1'], data_df.loc[index, 'principal componen
#                 , color = color, s = 2000, marker=r"".format(target))

targets = ['setosa', 'versicolor', 'virginica']
colors = ['#476A2A', '#A83683', '#BD3430']
plt.figure(figsize=(10,10))
plt.xlim(pca_df['principal component 1'].min(),pca_df['principal component 1'].max())
plt.ylim(pca_df['principal component 2'].min(),pca_df['principal component 2'].max())
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of 2 component about iris dataset')

for target, color in zip(targets,colors):
  index = pca_df['target'] == target
  for i in range(len(pca_df.loc[index, 'principal component 1'].values)):
    plt.text(pca_df.loc[index, 'principal component 1'].values[i],
             pca_df.loc[index, 'principal component 2'].values[i],
             s=target,color = color, fontdict={'weight':'bold','size':9})

plt.show

from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=0, learning_rate=200)
X_tsne = tsne.fit_transform(X)

temp_df = pd.DataFrame(data = X_tsne, columns = ['t-SNE feature 1', 't-SNE feature 2'])
tsne_df = pd.concat([temp_df, df[['target']]], axis = 1)
tsne_df.head(10)
```

```
import matplotlib.pyplot as plt

targets = ['setosa', 'versicolor', 'virginica']
colors = ['#476A2A', '#A83683', '#BD3430']
plt.figure(figsize=(10,10))
plt.xlim(tsne_df['t-SNE feature 1'].min(),tsne_df['t-SNE feature 1'].max())
plt.ylim(tsne_df['t-SNE feature 2'].min(),tsne_df['t-SNE feature 2'].max())
plt.xlabel('t-SNE feature 1')
plt.ylabel('t-SNE feature 2')
plt.title('t-SNE about iris dataset')

for target, color in zip(targets,colors):
  index = tsne_df['target'] == target
  for i in range(len(tsne_df.loc[index, 't-SNE feature 1'].values)):
    plt.text(tsne_df.loc[index, 't-SNE feature 1'].values[i],
    tsne_df.loc[index, 't-SNE feature 2'].values[i],
    s=target,color = color, fontdict={'weight':'bold','size':9})
plt.show
```