# 인공지능 Lab. Project #1

## <PCA를 이용한 얼굴인식>

학부: 컴퓨터정보공학부

학번: 2020202013

이름: 고가연

I. Introduction

=> 해당 프로젝트에서는 PCA를 이용한 얼굴 인식 과정을 시뮬레이션하는 실습을 진행한다. 해당 실습에서 사용할 얼굴 영상 DataBase는 LFW(Labeled Faces in the World)이다. 총 62명의 얼굴을 찍은 이미지를 담고 있고 각 얼굴들의 주성분들을 시각화 해서 확인한 후 PCA를 적용한 훈련 데이터로 학습시켜서 K-Nearest Neighbor(KNN) 알고리즘도 적용하여 정확도를 측정하는 것을 목표로 한다. 또한, CNN(Convolutional neural network) 알고리즘으로 LFW 데이터셋을 학습시켜 결과를 비교해본다.

II. Algorithm

1. KNN (K-Nearest Neighbor)

=> KNN 알고리즘은 판별하고 싶은 데이터와 인접한 K개수의 데이터를 찾아, 해당 데이터의 라벨이 다수인 범주로 데이터를 분류하는 방식이다.

=> 해당 알고리즘을 구현하기 위해 작성한 코드의 flow와 이때 쓰인 method에 대해 설명하겠다. 먼저, LFW 데이터셋을 가져온 다음 데이터들을 class별로 출력해서 어떻게 구성되었는지 확인했다. 그리고나서 데이터편중을 막기 위해서 50개의 이미지만 선택해서 학습시키는 것으로 했다. 그리고 해당 데이터를 전처리하기 위해서 전처리 메소드인 MinMaxScaler를 적용했다. 그리고 전처리한 데이터를 분할하고 그룹화를 진행했다. 머신러신 라이브러리에서 가져온 KNeighborsClassifier()를 통해 이웃의 수는 1로 설정하고 모델을 학습시켰다.

2. PCA (Principal Component Analysis)

=> 주성분 분석이란, 차원 축소 방법으로써 많은 feature(특성)로 구성된 다차원 데이터셋의 차원을 축소하여 불필요한 feature를 제거한다. 그래서 새로운 데이터셋을 생성하는 방법이다.

=> 해당 알고리즘을 구현한 코드의 flow에 대해 설명하겠다. 먼저 matplotlib 라이브러리의 pyplot 메소드와 mglearn 메소드를 사용해 PCA 화이트닝 옵션을 적용하여 scatter plot을 그려 데이터들의 스케일이 같아지도록 조정하였다. 그리고 PCA 모델을 생성한 후 적용하였다. 주성분의 개수를 100으로 하고 화이트닝 옵션을 적용하였다. 그리고 해당 PCA 모델을 데이터에 적용하였고 PCA를 적용한 데이터 형태를 출력하여 확인

하였다. 그리고나서 KNN 머신러닝 모델을 생성 및 학습시킨 후 PCA의 화이트닝 옵션을 적용한 후 정확도의 차이를 비교하였다. 그리고 계산한 주성분을 시각화하여 출력해보는 코드를 작성했다.

=> 그리고나서, 주성분의 개수에 따라서 얼굴 이미지를 재구성한 결과를 출력해보았다. PCA의 주성분의 개수가 많아짐에 따라 이미지의 해상도가 올라가는 것을 확인하였다.

3. CNN (Convolutional neural network)

=> CNN은 합성곱 신경망으로, 합성곱 계층, 폴링 계층, 활성 함수를 N번 반복하다 평탄화, 전결합 계층, 렐루 함수, 전결합 계층, 소프트맥스 함수를 통해 분류 결과를 출력하는 알고리즘이다.

=> 해당 알고리즘을 구현한 코드 flow는 다음과 같다. LFW dataset을 load한 다음 데이터들을 정규화해 주었다. 그리고 앞서 말한 계층들을 사용해 모델을 생성하였다. 모델 요약을 하고 컴파일한 후 훈련하는 코드를 작성하여 생성한 CNN 모델의 정확도를 결과로 출력하였다.

III. Result

1. LWF Dataset load와 데이터셋에 대한 설명

=> 아래의 이미지는 LWF(Labeled Faces in the World) 얼굴 데이터베이스를 사용해서 data의 shape을 출력한 결과화면이다. 총 62개의 class의 데이터 3023로 구성되어 있고 각각의 얼굴 이미지는 87x65로 구성되어 있는 것을 알 수 있다.

```
people.images.shape : (3023, 87, 65)
클래스 개수 : 62
```

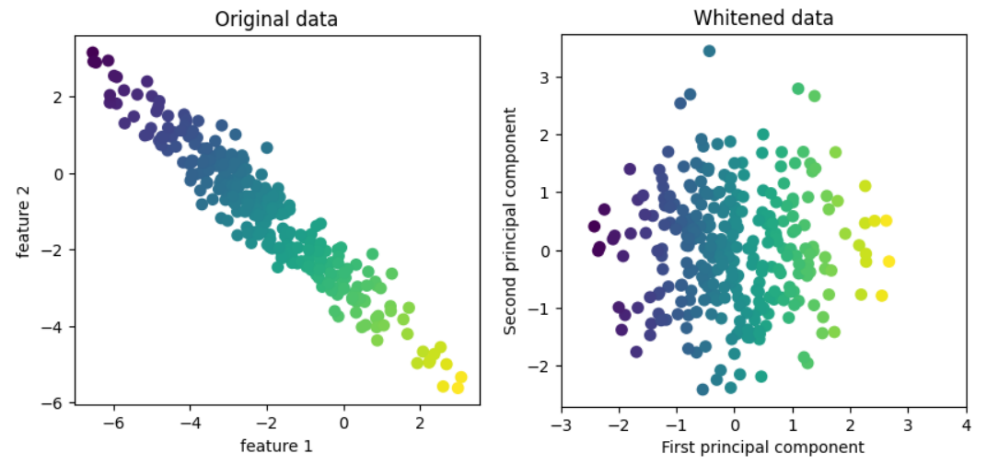
=> 아래의 이미지는 데이터셋에 존재하는 각각의 인물에 대한 이미지의 개수를 출력한 결과 화면이다.

=> 해당 데이터셋에서 편중된 데이터는 George W Bush가 530개인 것으로 볼 수 있다. 그리고 편중 데이터를 없애기 위해서 각 사람마다 50개의 이미지를 선택해서 사용하기로 했다.

| | |
|---|---|
| Alejandro Toledo | 39 |
| Alvaro Uribe | 35 |
| Amelie Mauresmo | 21 |
| Andre Agassi | 36 |
| Angelina Jolie | 20 |
| Ariel Sharon | 77 |
| Arnold Schwarzenegger | 42 |
| Atal Bihari Vajpayee | 24 |
| Bill Clinton | 29 |
| Carlos Menem | 21 |
| Colin Powell | 236 |
| David Beckham | 31 |
| Donald Rumsfeld | 121 |
| George Robertson | 22 |
| George W Bush | 530 |
| Gerhard Schroeder | 109 |
| Gloria Macapagal Arroyo | 44 |
| Gray Davis | 26 |
| Guillermo Coria | 30 |
| Hamid Karzai | 22 |
| Hans Blix | 39 |
| Hugo Chavez | 71 |
| Igor Ivanov | 20 |
| Jack Straw | 28 |
| Jacques Chirac | 52 |
| Jean Chretien | 55 |
| Jennifer Aniston | 21 |
| Jennifer Capriati | 42 |
| Jennifer Lopez | 21 |
| Jeremy Greenstock | 24 |
| Jiang Zemin | 20 |
| John Ashcroft | 53 |
| John Negroponte | 31 |
| Jose Maria Aznar | 23 |
| Juan Carlos Ferrero | 28 |
| Junichiro Koizumi | 60 |
| Kofi Annan | 32 |
| Laura Bush | 41 |
| Lindsay Davenport | 22 |
| Lleyton Hewitt | 41 |
| Luiz Inacio Lula da Silva | 48 |
| Mahmoud Abbas | 29 |
| Megawati Sukarnoputri | 33 |
| Michael Bloomberg | 20 |
| Naomi Watts | 22 |
| Nestor Kirchner | 37 |
| Paul Bremer | 20 |
| Pete Sampras | 22 |
| Recep Tayyip Erdogan | 30 |
| Ricardo Lagos | 27 |
| Roh Moo-hyun | 32 |
| Rudolph Giuliani | 26 |
| Saddam Hussein | 23 |
| Serena Williams | 52 |
| Silvio Berlusconi | 33 |
| Tiger Woods | 23 |
| Tom Daschle | 25 |
| Tom Ridge | 33 |
| Tony Blair | 144 |
| Vicente Fox | 32 |
| Vladimir Putin | 49 |
| Winona Ryder | 24 |

2. PCA Whitening 사용

=> 아래의 이미지는 PCA 화이트닝 옵션을 적용하여 scatter plot을 출력한 결과 화면이다. 이는 화이트닝 옵션 없이 변환한 후에 StandardScaler를 적용하는 것과 같다. 즉, 특성들의 평균을 0, 분산을 1로 스케일링하는 것인데, 특성들을 정규분포로 만드는 것을 말한다. 따라서 PCA 화이트닝 옵션을 사용해서 주성분들의 스케일을 같아지게 만들어 머신러닝이 잘 동작할 수 있도록 하는 것이 목적이다. 아래의 출력 결과를 통해 데이터가 회전하는 것뿐만 아니라 스케일도 조정되어 그래프가 원모양으로 바뀐 것을 볼 수 있다.

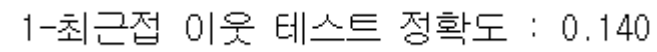


3. KNN 모델을 생성, 학습시켜 결과 분석 (data preprocessing, whitening 차이)

① PCA 화이트닝 옵션 없는 경우 (MinMaxScaler 사용)

=> 1-최근접 이웃 분류기 (KNeighborsClassifier)를 사용해서 KNN 모델을 생성하는데 PCA 화이트닝 옵션을 사용해서 데이터 스케일링 작업을 한 것이 아니라 전처리 메소드인 MinMaxScaler를 적용한 경우이다.

=> 이때 KNN의 정확도는 아래의 이미지와 같이 0.140, 즉 14%인 것으로 확인했다.

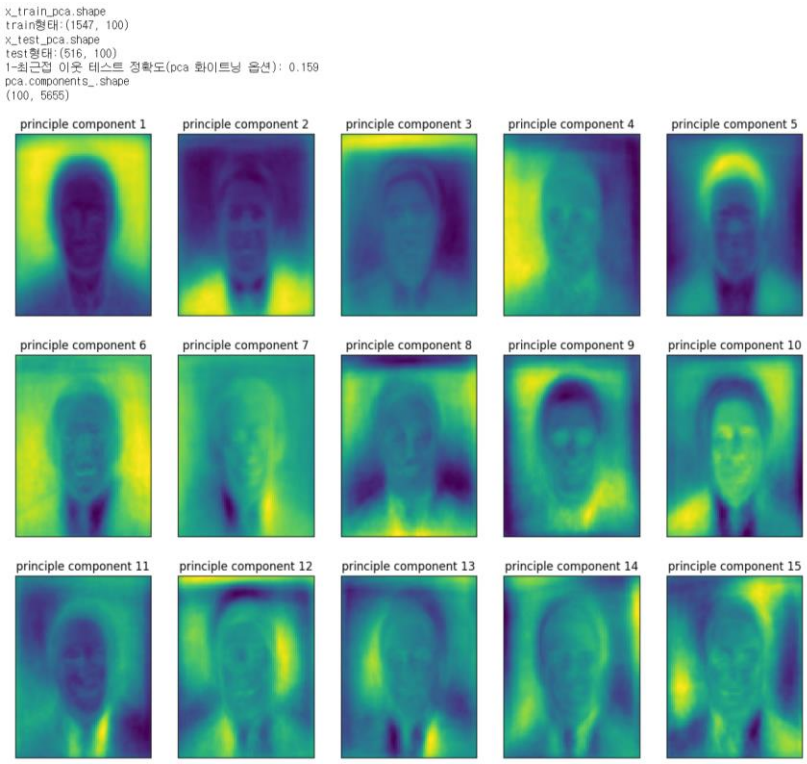



② PCA 화이트닝 옵션 있는 경우 (PCA를 통해서 고유 얼굴 성분 출력)

=> KNN 모델을 생성하는데 PCA 화이트닝 옵션을 사용해서 데이터 스케일을 조정하였다. 그 결과 아래의 이미지에서 보듯이 0.159, 약 16%인 것을 알 수 있다.

=> 이를 통해 KNN 모델을 학습시키는 과정에서 PCA Whitening 여부를 통한 정확도 차이를 비교한 결과, PCA 화이트닝 옵션을 적용했을 때가 정확도가 더 높아지는 것을 알 수 있었다. (14% -> 16%)

=> 그리고 PCA를 하여 고유 얼굴 성분을 아래의 이미지처럼 출력한 것을 볼 수 있다.




4. 주성분의 개수의 차이를 두어 얼굴 이미지 재구성

   => 아래의 이미지는 주성분의 개수의 차이를 두어서 얼굴 이미지를 재구성한 결과 화면이다. 맨 왼쪽의 열은 원본 이미지이고 두 번째 열이 10개의 주성분만 사용했을 때이다. 10개의 주성분만 사용한 경우는 얼굴의 각도나, 조명과 같은 기본적인 이미지의 요소만 나타나는 것을 볼 수 있다. 그리고 50, 100, 500개의 주성분을 사용한 경우를 각각 비교해보았을 때 주성분의 개수가 많아질수록 이미지가 점점 상세해지는 것을 알 수 있었다. 따라서, 주성분의 개수를 픽셀 수만큼 사용한다면 원본처럼 완벽하게 이미지를 재구성할 수 있을 것이다.


5. CNN 모델 학습, 결과 비교

=> 마지막 실습 단계로 CNN 모델을 학습하고 PCA 모델과 결과를 비교하였다. PCA의 정확도는 약 16%가 나왔는데 아래의 CNN의 결과화면을 보면 약 20%가 나오는 것을 볼 수 있다. 이를 통해서 CNN이 PCA보다 딥러닝 기술의 효율이 더 좋은 것을 알 수 있었다. 또한, CNN은 이미지 데이터가 가지고 있는 특징, 즉 local feature들을 잘 잡아내서 성능이 더 좋다.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d_6 (MaxPooling  (None, 63, 63, 32)        0
2D)
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling  (None, 30, 30, 64)        0
2D)
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_8 (MaxPooling  (None, 14, 14, 64)        0
2D)
_________________________________________________________________
flatten_2 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              12846080
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 5125
=================================================================
Total params: 12,907,525
Trainable params: 12,907,525
Non-trainable params: 0
```

```
Epoch 1/10
40/40 [==============================] - 10s 246ms/step - loss: 17.8320 - accuracy: 0.1925 - val_loss: 1.6095 - val_accuracy: 0.2000
Epoch 2/10
40/40 [==============================] - 12s 309ms/step - loss: 1.6102 - accuracy: 0.1900 - val_loss: 1.6095 - val_accuracy: 0.2000
Epoch 3/10
40/40 [==============================] - 11s 276ms/step - loss: 1.6101 - accuracy: 0.1800 - val_loss: 1.6095 - val_accuracy: 0.2000
Epoch 4/10
40/40 [==============================] - 10s 252ms/step - loss: 1.6098 - accuracy: 0.1975 - val_loss: 1.6094 - val_accuracy: 0.2000
Epoch 5/10
40/40 [==============================] - 10s 257ms/step - loss: 1.6097 - accuracy: 0.2000 - val_loss: 1.6095 - val_accuracy: 0.2000
Epoch 6/10
40/40 [==============================] - 10s 257ms/step - loss: 1.6098 - accuracy: 0.1750 - val_loss: 1.6094 - val_accuracy: 0.2000
Epoch 7/10
40/40 [==============================] - 11s 274ms/step - loss: 1.6099 - accuracy: 0.2000 - val_loss: 1.6095 - val_accuracy: 0.2000
Epoch 8/10
40/40 [==============================] - 11s 273ms/step - loss: 1.6097 - accuracy: 0.1625 - val_loss: 1.6094 - val_accuracy: 0.2000
Epoch 9/10
40/40 [==============================] - 11s 269ms/step - loss: 1.6097 - accuracy: 0.1950 - val_loss: 1.6094 - val_accuracy: 0.2000
Epoch 10/10
40/40 [==============================] - 11s 269ms/step - loss: 1.6098 - accuracy: 0.1700 - val_loss: 1.6094 - val_accuracy: 0.2000
Test loss: 1.609439730644226
Test accuracy: 0.20000000298023224
```

## IV. Consideration(고찰작성)

=> 해당 과제를 진행하면서 다양한 머신러닝 기법을 직접 실습해볼 수 있었다. KNN 알고리즘과 PCA, CNN을 사용해보고 해당 결과들의 차이점을 비교하는 과정을 통해 각각의 기법들의 특징들을 더 깊이 이해할 수 있었던 것 같다. 또한, 머신러닝 기법을 직접 사용해보는 것은 처음인데 모델에 적용하기 전의 전처리 과정도 이해할 수 있었다. 데이터의 스케일을 맞추는 정규화하는 과정도 거쳐야 하고 또한 데이터의 편중을 막기 위해 각 class에서 이미지의 개수를 선택해서 뽑아내는 과정도 있어야 한다는 것을 깨달았다.

=> 아쉬운 점은 정확도가 높지 않아서 출력 결과가 제대로 나올까 걱정했지만 그래도 PCA의 주성분들의 개수를 다르게 해서 이미지를 재구성한 결과를 확인했을 때 정상적으로 출력돼서 다행이었다. 다음에 머신러닝 알고리즘을 구현할 때는 정확도를 어떻게 더 높일 수 있는지 더 분석해 봐야겠다.