인공지능

Project1

담당 교수님: 박철수

강의 시간: 금3,4

학 과: 컴퓨터정보공학부

학 번: 2020202055

성 명: 최소윤

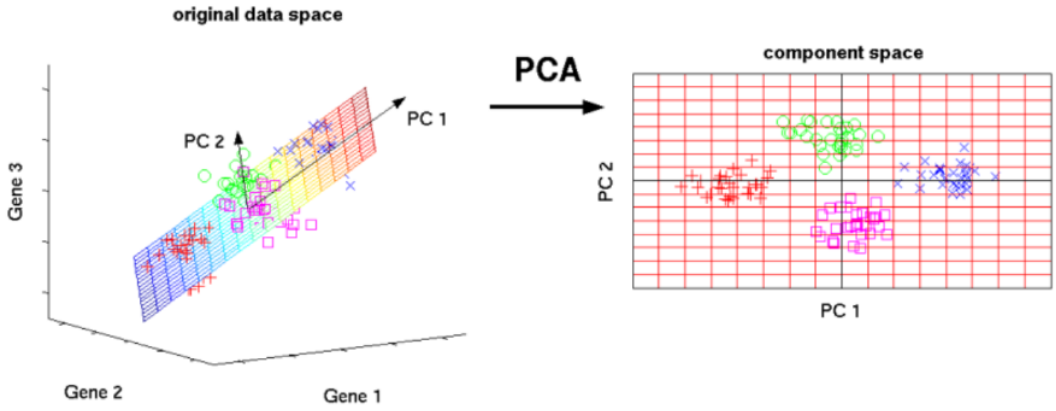
1. Introduction

PCA를 이용하여 얼굴 인식 과정을 시뮬레이션 하는 것이 이번 프로젝트의 목표이다. 학습할 dataset은 LFW 얼굴 dataset을 이용하며 얼굴의 주성분을 추출해보고 PCA whitening옵션 적용시켜 데이터를 조정해보고, KNN알고리즘을 이용하여 정확도를 확인해본다. 또한 CNN 모델을 이용하여 데이터를 학습시키고 이미지를 분류해본다.

2. Algorithm

PCA & PCA whitening

PCA란 Principal Component Analysis로 주성분 분석이라 한다. PCA는 데이터 집합 내에 존재하는 각 데이터의 차이를 가장 잘 나타내 주는 요소, 즉 데이터를 잘 표현할 수 있는 특성을 찾아내는 방법이다.



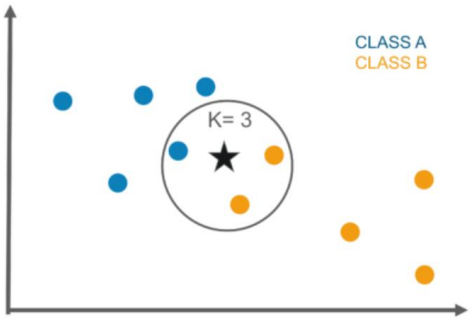
<그림 1. PCA>

PCA를 이용하여 서로 상관성이 높은 여러 변수들의 선형조합으로 만든 새로운 변수들로 요약 및 축약할 수 있다. 데이터의 분산을 최대한 보존하면서 직교하는 새 기저인 축을 찾아 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환시켜준다.

PCA whitening은 이미지나 영상 데이터의 픽셀을 비교할 때 얼굴 위치가 한 픽셀만 오른쪽으로 이동해도 큰 차이를 만들어 완전히 다른 얼굴로 인식하는 문제를 해결하기 위한 방법이다. 주성분으로 변환하여 거리를 계산하여 정확도를 높이는 방법을 사용하여 PCA whitening 옵션을 True로 변경하여 주성분의 스케일이 같아지도록 조정한다. 이를 이용하면 모델 학습 시 정확도를 높일 수 있다.

KNN

KNN은 K-Nearest Neighbors의 약자로 최근접 이웃 알고리즘이다. 새로운 입력으로 들어온 데이터를 특정 값으로 분류하는데 현재 데이터와 가장 가까운 K개의 데이터를 찾아 가장 많은 분류 값으로 현재의 데이터를 분류한다.



<그림 2. KNN example>

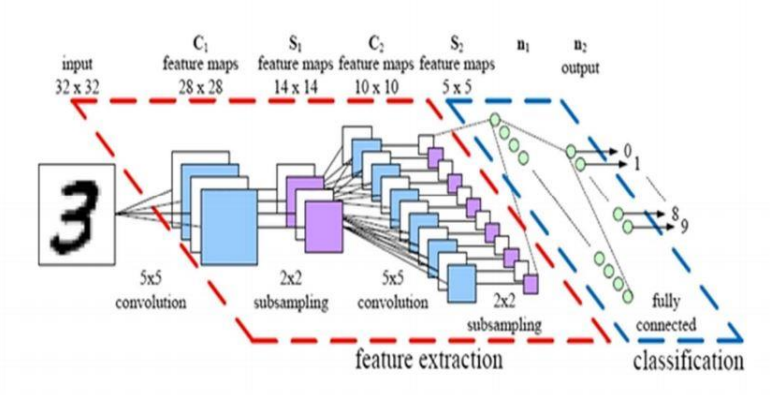
위 그림을 예시로 들면 별 모양의 데이터는 가장 가까운 3개의 기존 데이터의 분류를 살펴보았을 때 class B의 데이터가 2개, class A의 데이터가 1개이므로 별표 데이터는 B 클래스로 분류된다. 수식으로 나타내면 다음과 같으며 (x1,x2,...,xn)의 값을 가지는 데이터와 (y1,y2,...,yn)의 값을 가지는 데이터 사이 거리를 나타낸다.

$$dist = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

<그림 3. KNN distance>

CNN

CNN은 Convolutional Neural Network의 약자로 이미지를 분석하기 위해 패턴을 찾는데 유용한 알고리즘이다. 이미지를 학습하고 패턴을 사용하여 이미지를 분류한다. 2차원의 이미지를 1차원으로 바꿔 신경망을 학습시켰던 이전과는 달리, CNN을 이용하여 이미지의 형태를 보존하도록 행렬 형태의 데이터를 입력 받기 때문에 이미지를 벡터화 하는 과정에서 발생하는 정보 손실을 방지할 수 있다.



<그림 4. CNN>

일반적인 신경망은 fully-connected 연산과 ReLU와 같은 비선형 활성화 함수의 합성으로 정의된 여러 계층을 쌓은 구조이다. CNN은 Feature를 추출하는 Convolution Layer와 추출된 Feature를 Sub-sampling하는 Pooling Layer로 구성되어 있다.

3. Result

LFW Dataset

<그림 5. LFW dataset>

Dataset은 LFW(Labeled Faced in Wild)이며 유명 인사들의 얼굴 이미지로 2000년 초반 이후의 정치인, 가수, 배우, 운동선수들의 얼굴을 포함하고 있다.

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

<그림 6. Dataset 구성>

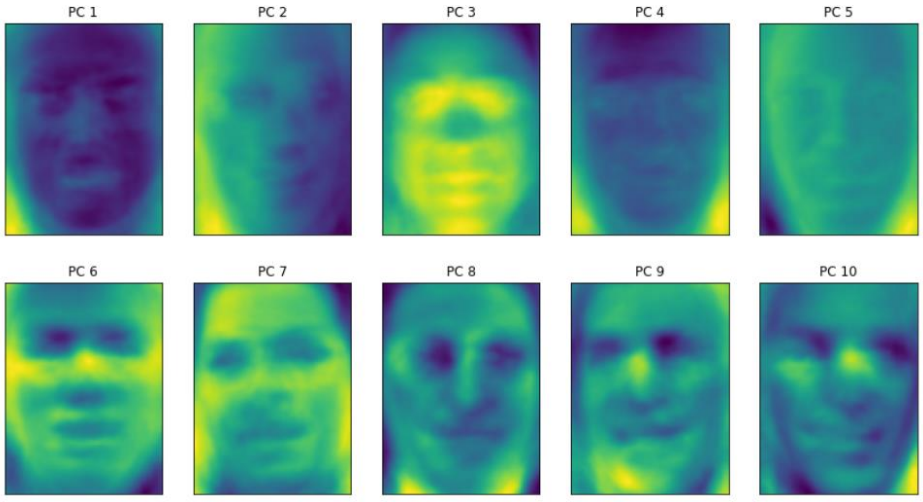
fetch_lfw_people() 명령으로 데이터를 가져올 수 있다. Dataset에는 62명의 얼굴을 찍은 이미지가 총 3023개가 있다. 크기는 87 x 65 픽셀이다.

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		

<그림 7. output name and number of times by target>

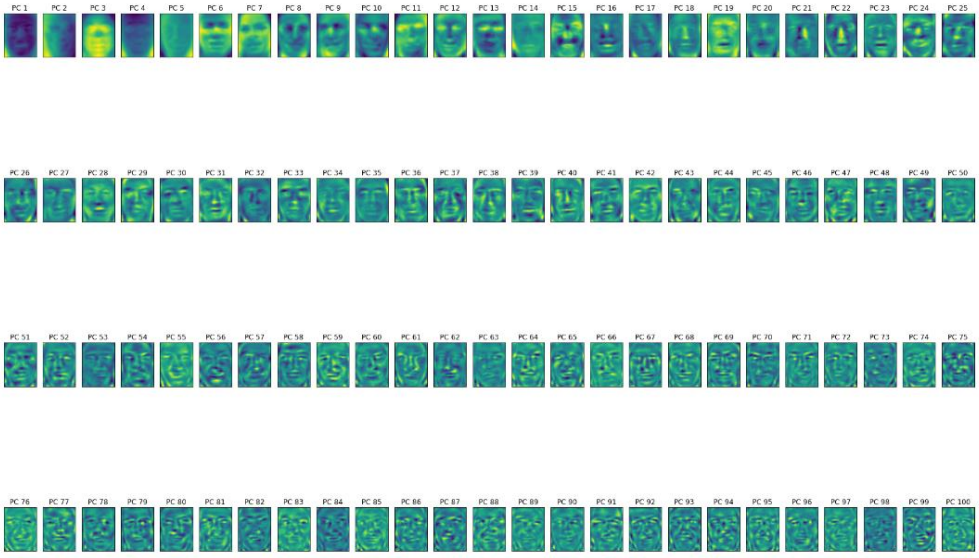
각 target의 이름과 이미지 개수를 출력했을 때 다음과 같이 나왔고 편중된 데이터셋이 있음을 확인하였다. 제일 적은 샘플의 개수는 20개이며, 제일 많은 샘플은 530개다. 학습은 가장 작은 샘플 개수인 20개를 선택하여 모델을 학습시켰다.

PCA whitening & PCA

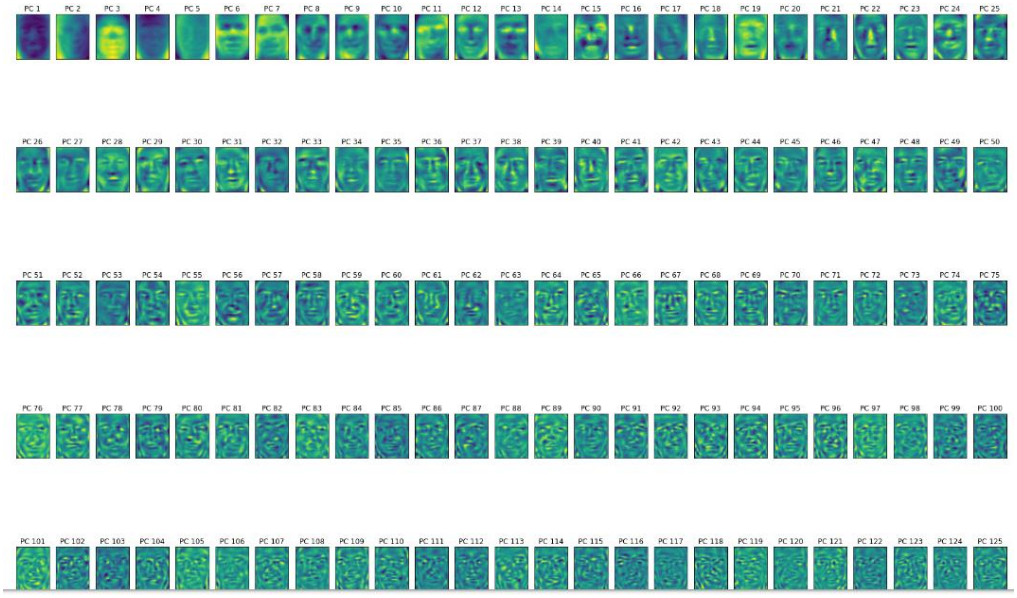


<그림 8. PCA를 이용한 고유 얼굴 성분>

PCA를 이용하여 추출한 고유 얼굴 성분이다. 위와 같은 주성분이 잡아낸 얼굴 이미지에는 명암, 조명, 대략적인 형태 등의 특징이 있다. 얼굴 이미지는 고유 얼굴 이미지에 가중치를 곱하여 더한 값이다.



<그림 9. components가 100일 때의 고유 얼굴 성분>

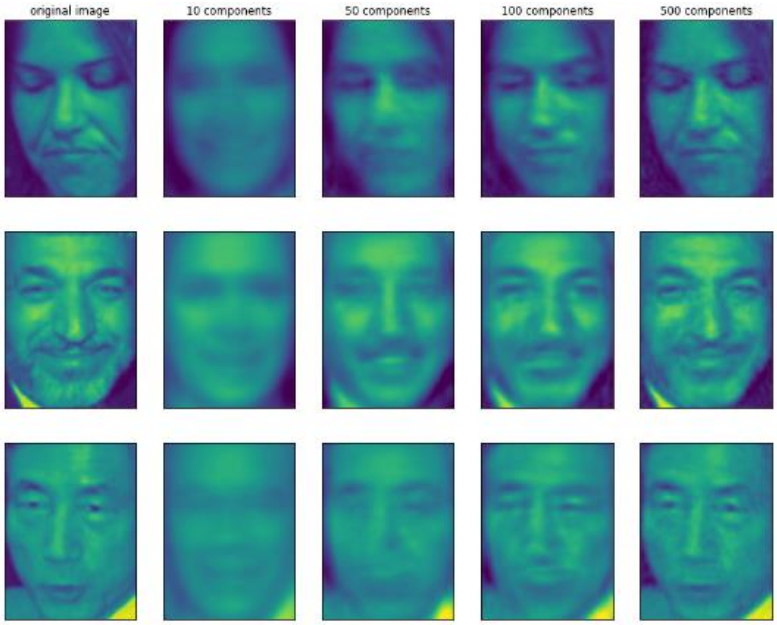


<그림 10. components가 125일 때의 고유 얼굴 성분>

KNN 모델을 학습시킬 때 PCA Whitening option을 False로 했을 때 정확도는 0.20으로 20%의 정확도를 보였고, PCA Whitening option을 True로 하고 주성분 개수를 100으로 설정했을 때는 정확도가 0.24였다. 주성분 개수를 125로 설정하고 PCA Whitening option을 True로 했을 때 정확도는 0.27로 코드를 실행했을 때

제일 높게 나왔다. 주성분 개수를 아주 작게 또는 크게 설정할 경우 정확도가 더 떨어지는 것을 확인하였고 적절한 주성분 개수를 설정했을 때 더 높은 정확도를 낼 수 있다는 것을 알 수 있었다.

KNN 모델로 학습할 시 데이터에 PCA를 적용하여 학습시킨 모델의 정확도가 4% 더 높게 나왔으며 주성분의 개수를 100에서 125로 변경했을 때 정확도가 3%로 더 높게 나온 것을 확인할 수 있었다.



<그림 11. components 개수에 따른 이미지>

mglearn 라이브러리를 이용하여 components 개수에 따른 이미지를 출력할 수 있다. 그림에 따르면 component 개수가 10개였을 때보다 500개였을 때가 더 얼굴이 명확하게 나온다.

KNN & CNN

LFW Dataset을 train_test_split 함수를 사용하여 data를 train과 test data로 나누었다. 기본적으로 train dataset를 75:25 비율로 나누며 비율을 조정하고 싶다면 train_size, test_size 매개변수를 설정하면 된다. KNN 모델로 학습 시 n_neighbors 매개변수가 있는데 이를 작게 설정하면 model의 결정 경계가 복잡해지면서 overfitting이 생길 수 있고, 반대로 크게 설정하면 model의 결정 경계가 단순해져 underfitting이 생길 수 있다.

PCA whitening option을 사용하지 않고 KNN으로 학습 시 정확도는 0.20으로 20%

의 정확도를 보였다.

```
(930, 87, 65, 1)
(930, 1)
(310, 87, 65, 1)
(310, 1)
```

<그림 12. trainset과 testset의 shape>

CNN으로 학습 시 train set과 test set의 shape를 맞춰야 학습이 제대로 되기 때문에 reshape 함수를 이용하여 x_train의 형태를 (930, 87, 65, 1)로 설정하였고 y_train의 형태를 (930, 1), x_test 형태를 (310, 87, 65, 1), y_test 형태를 (310, 1)로 바꿔주었다.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 87, 65, 32)        320

max_pooling2d_2 (MaxPooling  (None, 43, 32, 32)        0
2D)

conv2d_3 (Conv2D)            (None, 43, 32, 64)        8256

max_pooling2d_3 (MaxPooling  (None, 21, 16, 64)        0
2D)

dropout_2 (Dropout)          (None, 21, 16, 64)        0

flatten_1 (Flatten)          (None, 21504)             0

dense_2 (Dense)              (None, 32)                688160

dropout_3 (Dropout)          (None, 32)                0

dense_3 (Dense)              (None, 62)                2046

=================================================================
Total params: 698,782
Trainable params: 698,782
Non-trainable params: 0
_________________________________________________________________
```

<그림 13. CNN 모델>

CNN 모델을 model.add() 함수를 통해 Con2D와 Maxpooling2D 등으로 층을 쌓았고 중간에 활성화 함수는 sigmoid 대신 relu 함수를 사용하였다. 그리고 마지막에는 softmax함수를 사용하여 model을 구성하였다.

```
8/8 [==============================] - 0s 22ms/step - loss: 2.9791 - accuracy: 0.1602
Epoch 995/1000
8/8 [==============================] - 0s 22ms/step - loss: 3.0191 - accuracy: 0.1559
Epoch 996/1000
8/8 [==============================] - 0s 24ms/step - loss: 3.1029 - accuracy: 0.1333
Epoch 997/1000
8/8 [==============================] - 0s 24ms/step - loss: 3.0536 - accuracy: 0.1409
Epoch 998/1000
8/8 [==============================] - 0s 23ms/step - loss: 3.0627 - accuracy: 0.1355
Epoch 999/1000
8/8 [==============================] - 0s 23ms/step - loss: 3.0720 - accuracy: 0.1484
Epoch 1000/1000
8/8 [==============================] - 0s 23ms/step - loss: 2.9770 - accuracy: 0.1516
Test loss:  3.8918204307556152
Test accuracy:  0.2548387050628662
```

<그림 14. CNN 학습>

model.fit 함수를 사용하여 학습을 시키고 test set으로 결과를 확인했을 때 정확도는 약 25%를 보였다. 앞서 PCA를 적용한 KNN 모델 학습에서는 약 27%가 나온 것과 비교하면 정확도가 CNN이 조금 더 떨어진 것을 확인할 수 있었다.

4. Consideration

CNN모델 학습 시 과제 제안서에 나와 있는 그대로 코드를 돌렸을 때 에러가 발생하여 모델을 학습시키지 못했던 어려움이 있었다. 이는 input data의 shape가 모델과 달라 생긴 에러였고 reshape 함수를 이용하여 이를 해결할 수 있었다. CNN 모델 학습했을 때 정확도가 1%대로 나와 이를 해결하기 위해 모델의 활성화 함수를 sigmoid 대신 relu 함수를 사용하였고, 학습 epoch를 100에서 1000으로 변경하여 학습량을 늘렸다. 수정한 모델을 학습시켰고 정확도는 25%로, KNN과 비슷한 정확도를 보인 것을 확인할 수 있었다.