

Day 5: Lab Exercise

June 1, 2018

Prof. Jongwuk Lee

실습 목차

➤ 1. 문서 클러스터링

- ◆ 함께 실습하기 (Dataset: Top-100 IMDB)
 - 텍스트 데이터 읽기, 분석 및 전처리
 - 모델 사용방법: K-means Clustering
 - 시각화: 문서 클러스터링
- ◆ 직접 실습하기

➤ 2. 얼굴 데이터에 차원축소 적용

- ◆ 함께 실습하기 (Dataset: LFW)
 - 이미지 데이터 읽기, 분석 및 전처리
 - 모델 사용방법: PCA, NMF
 - 얼굴 분류 정확도 비교: KNN
 - 시각화 1: 상위 성분
 - 시각화 2: 추출된 특성을 이용한 이미지 재구성

➤ 3. 추천시스템 구현

- ◆ 함께 실습하기
 - 데이터 읽기, 분석 및 전처리
 - 모델 사용방법: SVD
- ◆ 직접 실습하기

문서 클러스터링

(Document Clustering)

문서 클러스터링

➤ 여러 개의 문서들을 비슷한 주제의 그룹으로 어떻게 묶을까?



◆ 실습 주제: 영화의 시놉시스(문서)로 클러스터링을 할 수 있을까?

문서 클러스터링

➤ 여러 개의 문서들을 비슷한 주제의 그룹으로 어떻게 묶을까?



판타지



가족



전쟁



코미디



문서 클러스터링

➤ 1단계: 텍스트 데이터 전처리

- ◆ 토큰화(Tokenization): Text data를 작은 단위로 쪼갬다. ex) words, phrase 단위
- ◆ 어근화(Stemming): walk, walking, walked, walks → walk(어근)
- ◆ 불용어(Stopwords) 제거: a, the, he, she, in 제거

➤ 2단계: TF-IDF 계산

- ◆ 문서의 feature을 뽑아 내기 위해 일반적으로 TF-IDF를 사용하여 topic을 뽑아낸다.

➤ 3단계: 클러스터링

- ◆ TF-IDF를 이용하여 뽑아낸 feature을 이용하여 각 문서들을 군집화한다.

➤ 4단계: 시각화


- ◆ 군집화한 문서들을 2-D로 시각화하여 Topic별로 잘 모여있는지 확인한다.
- ◆ MDS(Multi-Dimensional Scaling): 자료의 Euclidean 거리를 바탕으로 2차원으로 차원축소하여 시각화한다.

필요한 패키지 목록

```
%matplotlib inline
import nltk
import numpy as np
import pandas as pd
import re
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import Kmeans
from sklearn.manifold import MDS
import matplotlib.pyplot as plt
```

데이터 소개

➤ 영화 데이터 크롤링: Top 100 movies from IMDB

- ◆ <http://www.imdb.com/list/ls055592025/>
- ◆  data/
 - titles.txt // 영화 제목
 - synopses.txt // 영화 시놉시스

데이터 소개

➤ 영화 데이터 크롤링: Top 100 movies from IMDB

- ◆ <http://www.imdb.com/list/ls055592025/>


- ◆  data/

- titles.txt
- synopses.txt

```
The Godfather
The Shawshank Redemption
Schindler's List
Raging Bull
Casablanca
One Flew Over the Cuckoo's Nest
Gone with the Wind
Citizen Kane
The Wizard of Oz
Titanic
```

데이터 소개

➤ 영화 데이터 크롤링: Top 100 movies from IMDB

- ◆ <http://www.imdb.com/list/ls055592025/>
- ◆  data/
 - title.txt
 - synopses.txt

In late summer 1945, guests are gathered for the wedding reception of

BREAKS HERE

구분되는 문구

In 1947, Andy Dufresne (Tim Robbins), a banker in Maine, is convicted

BREAKS HERE

구분되는 문구

The relocation of Polish Jews from surrounding areas to Krakow begins

BREAKS HERE

구분되는 문구

데이터 읽기

```
titles = open('data/titles.txt').read().split('\n')
synopses = open('data/synopses.txt').read().split('\n BREAKS HERE')
synopses = synopses[:100]
len(synopses)    # output = 100
```

➤ 매개변수 설명

- ◆ `read().split(A)` : A를 기준으로 하여 데이터를 나누어 불러들인다.

데이터 분석

➤ 영화 데이터 수가 100개가 맞는지 확인한다.

```
print(len(titles))          # output = 100  
print(len(synopses))       # output = 100
```

데이터 전처리

➤ 토큰화(Tokenization) + 어근화(Stemming)

- ◆ Text → Sentence
- ◆ Sentence → Word 로 쪼개기
- ◆ 알파벳이 아닌 문자는 제거하기(a-z, A-Z를 제외한 모든 문자 제거)
- ◆ SnowballStemmer를 사용하여 어근화 하기

```
1 from nltk.stem.snowball import SnowballStemmer
2 stemmer = SnowballStemmer("english")
3 print(stemmer.stem('walked'))
4 print(stemmer.stem('walks'))
5 print(stemmer.stem('walking'))
6 print(stemmer.stem('walk'))
```

```
walk
walk
walk
walk
```

데이터 전처리

➤ 1단계: 토큰화 (Tokenization)

- ◆ 1-1: Text → Sentence 로 쪼개기

```
tokens = [word.lower() for sent in nltk.sent_tokenize(text)  
           for word in nltk.word_tokenize(sent)]
```

데이터 전처리

➤ 1단계: 토큰화 (Tokenization)

- ◆ 1-2: Sentence → Word 로 쪼개기

```
tokens = [word.lower() for sent in nltk.sent_tokenize(text)
           for word in nltk.word_tokenize(sent)]
```

데이터 전처리

➤ 1단계: 토큰화 (Tokenization)

- ◆ 1-3: 알파벳이 아닌 문자는 제거하기(a-z, A-Z를 제외한 모든 문자 제거)

```
def tokenize_stem(text):  
    tokens = [word.lower() for sent in nltk.sent_tokenize(text)  
               for word in nltk.word_tokenize(sent)]  
    final_tokens = []  
    for token in tokens:  
        if re.search('[a-zA-Z]', token):  
            final_tokens.append(token)  
    return stems
```


데이터 전처리

➤ 2단계: 어근화(Stemming)

- ◆ SnowballStemmer를 사용하여 어근화 하기

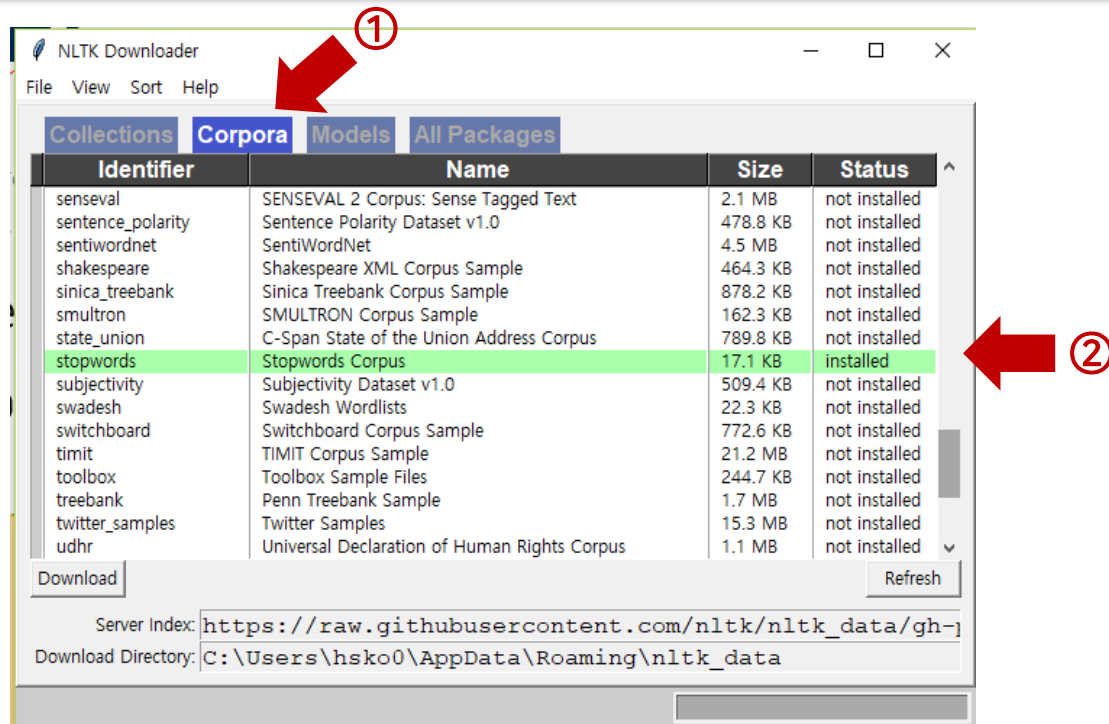
```
import nltk
import re
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english")
def tokenize_stem(text):
    tokens = [word.lower() for sent in nltk.sent_tokenize(text)
               for word in nltk.word_tokenize(sent)]
    final_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            final_tokens.append(token)
    stems = [stemmer.stem(t) for t in final_tokens]
    return stems
```

데이터 전처리

➤ 3단계: 불용어(Stopwords)

- ◆ a, the, in, she, he 등 의미가 없는 단어들의 집합

`nltk.download()` # stopwords 설치: corpora에서 stopwords 클릭



데이터 전처리

➤ 3단계: 불용어(Stopwords)

- ◆ a, the, in, she, he 등 의미가 없는 단어들의 집합

```
import nltk  
stopwords = nltk.corpus.stopwords.words('english')  
print(stopwords)          # output = [i, me, my, myself, ...]  
print(len(stopwords))     # output = 179
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',  
ou'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',  
r', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',  
s', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these
```

데이터 전처리

➤ Recall. 문서클러스터링 단계

➤ 1단계: 텍스트 데이터 전처리

- ◆ 토큰화(Tokenization): Text data를 작은 단위로 쪼갬다. ex) words, phrase 단위
- ◆ 어근화(Stemming): walk, walking, walked, walks → walk(어근)
- ◆ 불용어(Stopwords) 제거: a, the, he, she, in 제거

➤ 2단계: TF-IDF 계산

- ◆ 문서의 feature을 뽑아 내기 위해 일반적으로 TF-IDF를 사용하여 topic을 뽑아낸다.

➤ 3단계: K-means 클러스터링

- ◆ TF-IDF를 이용하여 뽑아낸 feature을 이용하여 각 문서들을 군집화한다.

➤ 4단계: 시각화

- ◆ 군집화한 문서들을 2-D로 시각화하여 Topic별로 잘 모여있는지 확인한다.
- ◆ MDS(Multi-Dimensional Scaling): 자료의 Euclidean 거리를 바탕으로 2차원으로 차원축소하여 시각화한다.

TF-IDF 계산

➤ TF-IDF 계산하여 중요한 feature 추출하기

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=200000, max_df=0.8, min_df=0.2,
                                   stop_words='english', tokenizer=tokenize_stem)
tfidf_matrix = tfidf_vectorizer.fit_transform(synopses)
print(tfidf_matrix.shape)      # output = (100, 548)
terms = tfidf_vectorizer.get_feature_names()
print(terms)
```

➤ 매개변수 설명

- ◆ max_df = 0.8: 80% 이상의 document에 존재하는 term은 중요한 feature가 아니므로 제거한다.
- ◆ min_df = 0.2: 20% 이하의 document에 존재하는 term은 등장인물의 이름 (Michael, Josh etc) 이 등장할 확률이 높는데, 의미는 없으므로 제거한다.

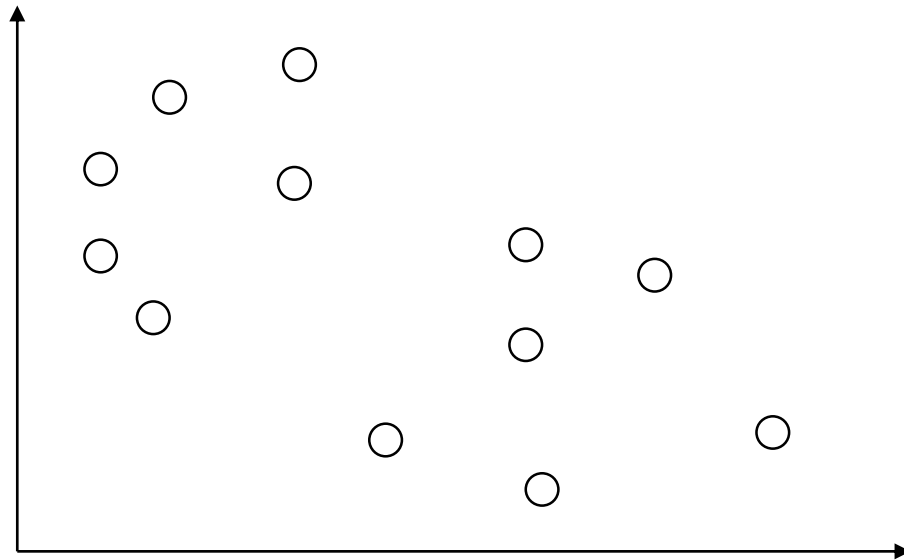
K-means Clustering

➤ 레이블이 없는 데이터를 어떻게 나눌까?



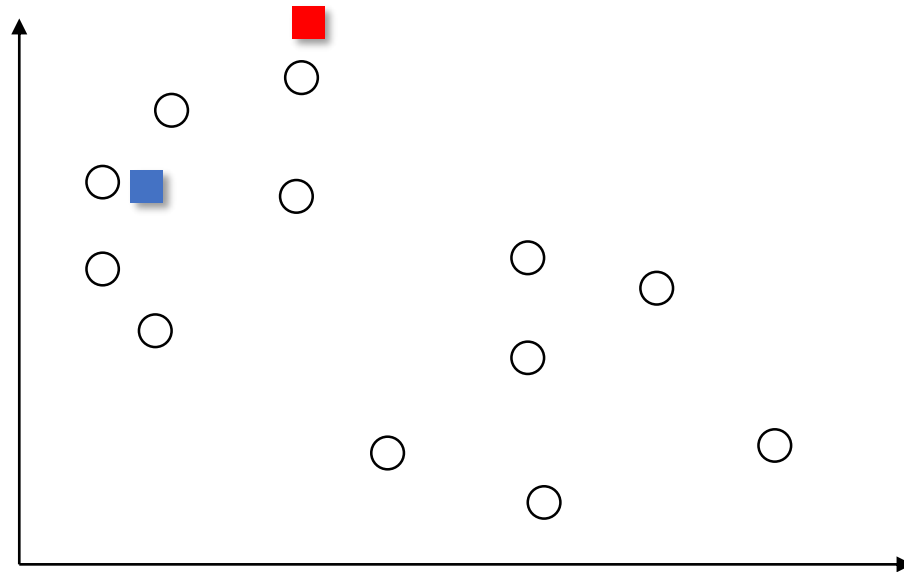
K-means Clustering

➤ 클러스터 개수를 정한다. (예: 2개)



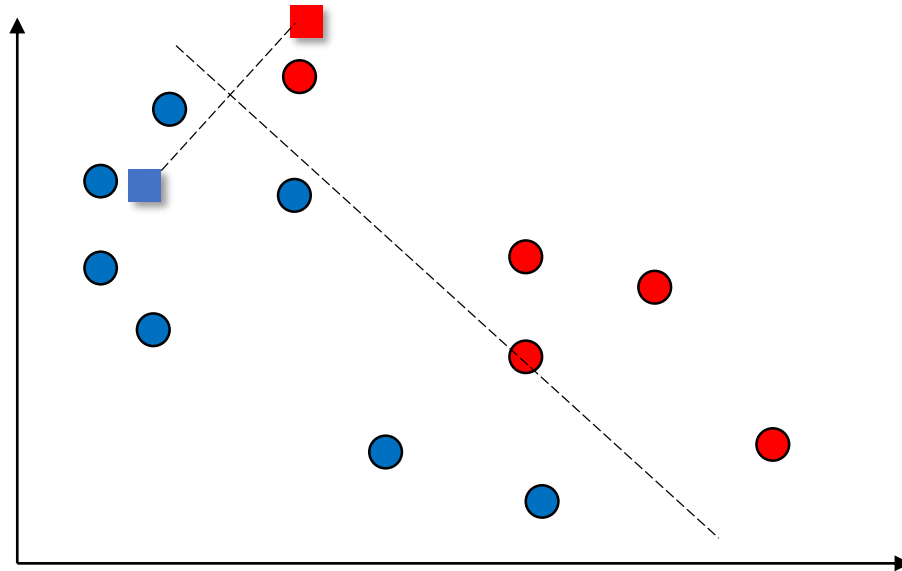
K-means Clustering

➤ 무작위로 2개의 클러스터 중심점을 잡는다.



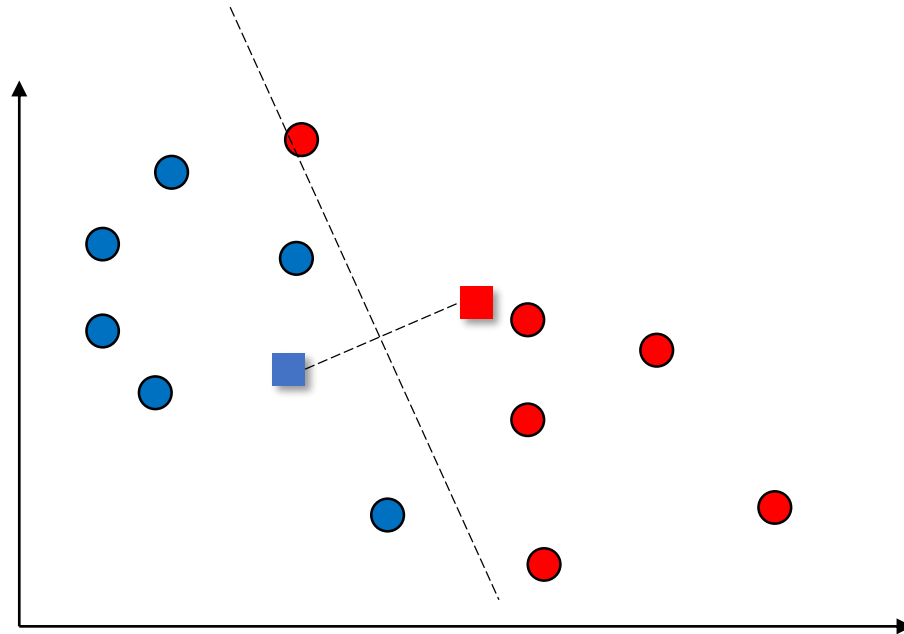
K-means Clustering

- Euclidean distance를 기준으로 가까운 클러스터로 묶는다.



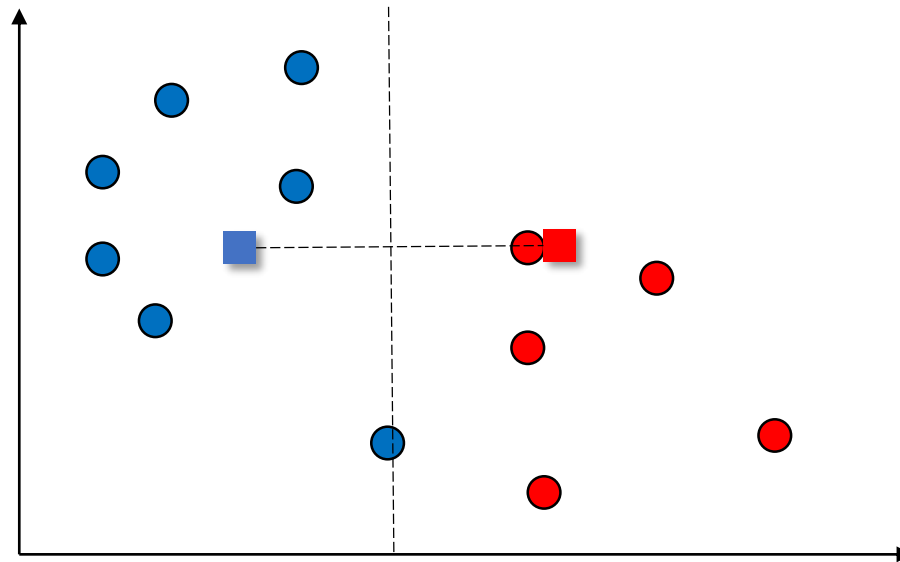
K-means Clustering

➤ 새로운 클러스터의 중심점을 구한다.



K-means Clustering

➤ 수렴할 때까지, 이 과정을 반복한다.



K-means Clustering

- 각 영화를 5개의 그룹으로 클러스터링 한다.

```
from sklearn.cluster import KMeans  
km = KMeans(n_clusters = 5)  
km.fit(tfidf_matrix)      # K-means clustering 적용  
clusters = km.labels_.tolist() # 리스트화  
print(clusters)
```

➤ 매개변수 설명

- ◆ n_clusters: 몇 개의 그룹으로 나눌지 정한다.

K-means Clustering

➤ 데이터 분석: 각 클러스터마다 몇 개의 영화가 있는지 확인

- ◆ `dataframe[column].value_counts()`: 각 그룹의 개수를 구할 때 용이하다.

```
import pandas as pd
films = {'title': titles, 'synopsis': synopses, 'cluster': clusters}
frame = pd.DataFrame(films, index=clusters, columns=['title', 'cluster'])
frame['cluster'].value_counts() # 각 클러스터의 영화 개수 출력
```

```
1 frame['cluster'].value_counts() #number of films per cluster (clusters from 0 to 4)
2      36
1      27
4      21
3       9
0       7
Name: cluster, dtype: int64
```

K-means Clustering

➤ 데이터 분석: 각 클러스터의 중요 topic 및 영화 출력

```
print("Top terms per cluster:")
order_centroids = km.cluster_centers_.argsort()[:, :-1]
for i in range(5):
    print("Cluster %d words:" % i, end='')
    for j in range(5):
        print(terms[order_centroids[i][j]], end=',')
    print('\n')
    print("Cluster %d titles:" % i, end='')
    for title in frame.loc[i]['title'].values.tolist():
        print(' %s,' % title, end='')
    print('\n')
```

K-means Clustering

➤ 데이터 분석: 각 클러스터의 중요 topic 및 영화 출력

Cluster 0 Topic(terms): kill,soldier,captain,armi,command,

Cluster 0 titles: The Shawshank Redemption, Casablanca, Lawrence of Arabia, The Sound of Music, Star Wars, 2001: A Space Odyssey, The Bridge on the River Kwai, Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb, Apocalypse Now, The Lord of the Rings: The Return of the King, Gladiator, From Here to Eternity, Saving Private Ryan, Unforgiven, Raiders of the Lost Ark, Patton, Jaws, The Treasure of the Sierra Madre, Platoon, Dances with Wolves, The Deer Hunter, All Quiet on the Western Front, Shane, The Green Mile, The African Queen, Stagecoach, Mutiny on the Bounty,

Cluster 1 Topic(terms): film,mr.,sing,love,relationship,

Cluster 1 titles: Citizen Kane, Singin' in the Rain, Amadeus, Gandhi, The Best Years of Our Lives, Annie Hall, Out of Africa, Good Will Hunting, Network, Nashville, The Graduate, A Clockwork Orange, Wuthering Heights, Yankee Doodle Dandy,

Cluster 2 Topic(terms): car,polic,kill,drive,train,

Cluster 2 titles: Titanic, Psycho, Sunset Blvd., Vertigo, West Side Story, The Silence of the Lambs, Chinatown, Some Like It Hot, The Good, the Bad and the Ugly, Butch Cassidy and the Sundance Kid, The French Connection, It Happened One Night, Fargo, Close Encounters of the Third Kind, American Graffiti, Pulp Fiction, The Maltese Falcon, Taxi Driver, Double Indemnity, Rebel Without a Cause, Rear Window, The Third Man, North by Northwest,

Cluster 3 Topic(terms): famili,father,home,brother,new,

Cluster 3 titles: The Godfather, Schindler's List, Raging Bull, One Flew Over the Cuckoo's Nest, Gone with the Wind, The Wizard of Oz, The Godfather: Part II, On the Waterfront, Forrest Gump, E.T. the Extra-Terrestrial, 12 Angry Men, Rocky, A Streetcar Named Desire, To Kill a Mockingbird, My Fair Lady, Ben-Hur, Doctor Zhivago, Braveheart, The Apartment, High Noon, The Pianist, Godfrellas, The Exorcist, City Lights, Midnight Cowboy, Mr. Smith Goes to Washington, Rain Man, Terms of Endearment, Giant, The Rapes of Wrath,

Cluster 4 Topic(terms): georg,marri,woman,famili,friend,

Cluster 4 titles: It's a Wonderful Life, The Philadelphia Story, An American in Paris, The King's Speech, A Place in the Sun, Tootsie,

시각화

➤ 클러스터 색깔과 이름을 정한다.

- ◆ 이름은 추출한 Topic으로 정한다.

```
#set up colors per clusters using a dict
```

```
cluster_colors = {0: 'red', 1: 'blue', 2: 'green', 3: 'purple', 4: 'orange'}
```

```
#set up cluster names using a dict
```

```
cluster_names = {0: 'Kill, Solider, Army',  
                 1: 'Sing, Love, Relationship',  
                 2: 'Car, Drive, Train',  
                 3: 'Family, Home, Brother',  
                 4: 'Marry, Woman, Friend'}
```


시각화

➤ Multi-Dimensional Scaling(MDS)

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.manifold import MDS
import matplotlib.pyplot as plt

dist = 1 - cosine_similarity(tfidf_matrix)      # distance equation
mds = MDS(n_components=2, dissimilarity = "precomputed")
Z = mds.fit_transform(dist)                    # 2차원으로 축소
xs, ys = Z[:, 0], Z[:, 1]                     # 시각화를 위해 저장해둔다.
```

➤ 매개변수 설명

- ◆ distance equation: distance는 일반적으로 값이 클수록 거리가 멀기 때문에 cosine similarity에 음수를 곱한 후 1을 더해 최소값이 0이 되도록 distance equation을 만들어준다.

시각화

➤ MDS를 한 클러스터 번호와 영화제목으로 데이터를 생성

- ◆ 데이터를 생성할 때 pandas의 dataframe을 사용하면 편리하다.

```
df = pd.DataFrame(dict(x = xs, y = ys, label=clusters, title=titles))  
groups = df.groupby('label')      # group by cluster
```

시각화

➤ Matplotlib을 사용하여 그린다.

- ◆ dataframe.ix : integer position과 label을 사용하여 데이터에 접근한다.

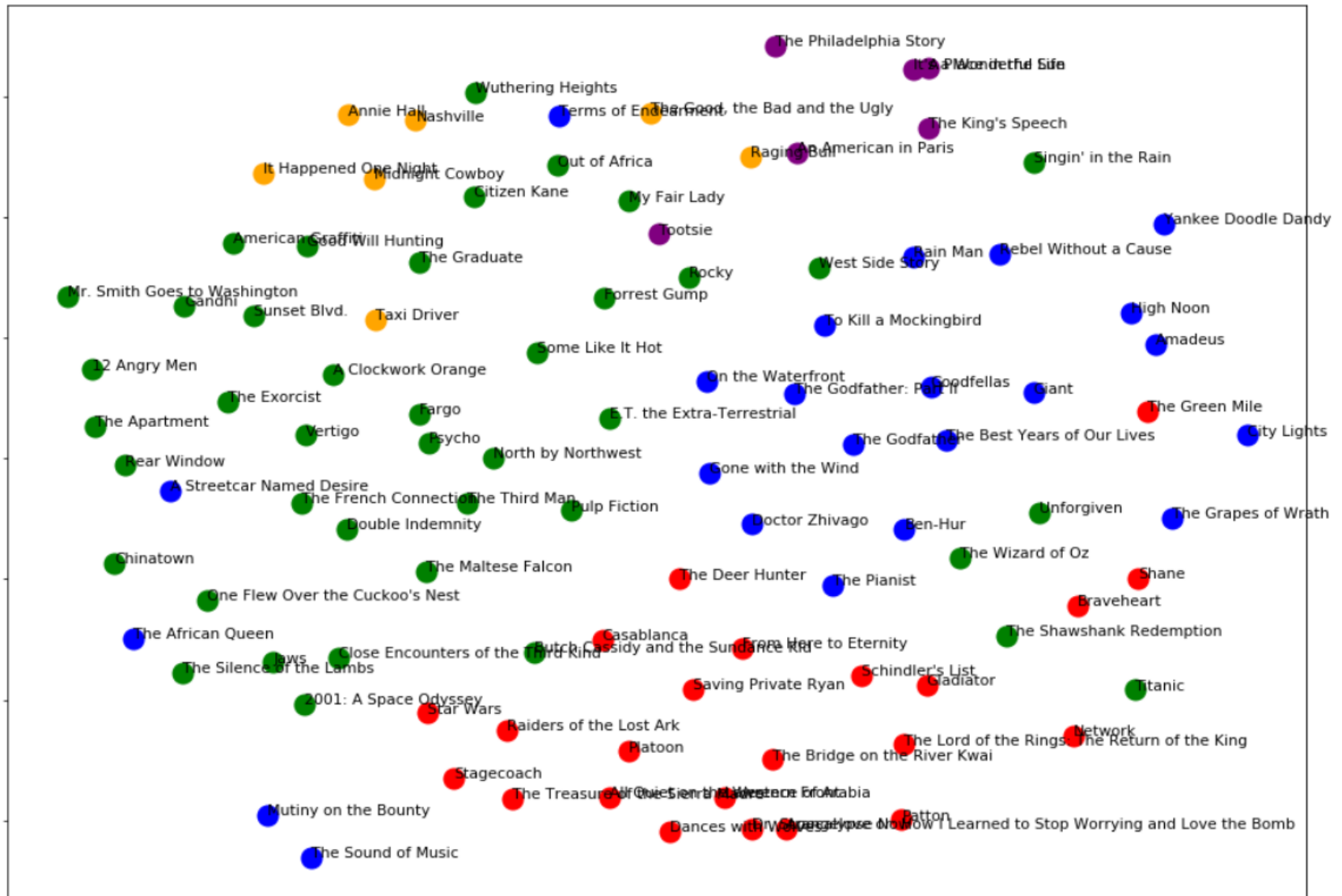
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(17, 12)) # set size

for idx, group in groups:
    ax.plot(group.x, group.y, marker='o', linestyle='', ms=15,
            label=cluster_names[idx], color=cluster_colors[idx])
ax.legend(numpoints=1) #show legend with only 1 point
for i in range(len(df)):
    ax.text(df.ix[i]['x'], df.ix[i]['y'], df.ix[i]['title'], size = 11)
plt.show()
```

시각화



시각화



직접 실습하기

➤ 데이터 소개

- ◆ 뉴스 데이터를 Clustering하여 시각화해보시오.
- ◆ data/
 - news_title.txt // 뉴스 제목
 - news_keyword.txt // 뉴스 키워드

차원 축소 **(Dimensionality Reduction)**

필요한 패키지 목록

```
%matplotlib inline # if jupyter notebook
import mglearn
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_lfw_people # 사용할 얼굴 데이터
from sklearn.neighbors import KNeighborsClassifier # KNN
from sklearn.model_selection import train_test_split
from sklearn.decomposition import NMF
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```


데이터 소개

➤ LFW (Labeled Faces in the Wild)

- ◆ <http://vis-www.cs.umass.edu/lfw/>

Labeled Faces in the Wild



Menu

- LFW Home
 - Mailing
 - Explore
 - Download
 - Train/Test
 - Results
 - Information
 - Errata
 - Reference
 - Resources
 - Contact
 - Support
 - Changes
- Part Labels
- UMass Vision

Labeled Faces in the Wild Home



NEW SURVEY PAPER:

Erik Learned-Miller, Gary B. Huang, Aruni RoyChowdhury, Haoxiang Li, and Gang Hua.

Labeled Faces in the Wild: A Survey.

In *Advances in Face Detection and Facial Image Analysis*, edited by Michal Kawulok, M. Emre Celebi, and Bogdan Smolka, Springer, pages 189-248, 2016.

[[Springer Page](#)] [[Draft pdf](#)]

데이터 읽기

```
from sklearn.datasets import fetch_lfw_people  
lfw_people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
```

➤ 매개변수 설명

- ◆ `min_faces_per_person` : 최소 이만큼의 데이터 개수를 가진 사람의 이미지만 불러들인다.
- ◆ `resize` : 각 사람의 이미지 크기(pixel size)를 재조정하는 비율 (default=0.5)

데이터 분석

➤ 데이터 수 및 기초 통계 정보 확인

```
lfw_people.data.shape          # output = (3023, 5655)
lfw_people.images.shape        # output = (3023, 87, 65)
lfw_people.target.shape        # output = (3023,)
len(lfw_people.target_names)    # output = 62 = # of classes
```

데이터 분석

➤ 데이터셋의 Bias 확인

```
import numpy as np

lfw_counts = np.bincount(lfw_people.target) # 각 클래스별 빈도 계산
for (name, count) in zip(lfw_people.target_names, lfw_counts):
    print("{0:25} {1:4}".format(name, count)) # (이름, 빈도 수) 출력
```

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

Dataset이 고르게 분포되어 있지 않음을 알 수 있다.

데이터 분석

➤ 데이터셋의 Bias 확인

```
import numpy as np

lfw_counts = np.bincount(lfw_people.target) # 각 클래스별 빈도 계산
for (name, count) in zip(lfw_people.target_names, lfw_counts):
    print("{0:25} {1:4}".format(name, count)) # (이름, 빈도 수) 출력
```

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

Dataset이 고르게 분포되어 있지 않음을 알 수 있다.

해결책: 각 사람당 최대 데이터 수를 정해놓자

데이터 전처리

➤ 데이터 마스크 처리

- ◆ 각 클래스별 최대 50개의 데이터만 사용하도록 한다.

```
mask = np.zeros(lfw_people.target.shape, dtype=np.bool)
for target in np.unique(lfw_people.target):
    mask[np.where(lfw_people.target==target)[0][:50]] = True

X_people = lfw_people.data[mask]
y_people = lfw_people.target[mask]
X_people = X_people/255 # 픽셀 크기를 0~1 사이로 조정
print(X_people.shape)   # output = (2063, 5655)
print(y_people.shape)   # output = (2063,)
```

데이터 전처리

➤ 데이터 마스크 처리

```
1 np.unique(lfw_people.target)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61], dtype=int64)
```

```
1 import numpy as np
2 for target in np.unique(lfw_people.target):
3     print(np.where(lfw_people.target==target)[0])
```

```
[ 58  72  90 314 494 496 598 650 960 1078 1212 1243 1265 1356
 1497 1586 1622 1793 1803 1816 1881 1952 1962 2003 2068 2179 2183 2239
 2373 2441 2510 2535 2622 2649 2651 2697 2819 2960 2998]
[  4  17  41  87 142 216 333 413 417 508 857 876 940 944
 949 1165 1307 1505 1529 1539 1729 1807 1837 1933 2211 2243 2254 2269
 2285 2290 2360 2387 2616 2636 2926]
[ 65 188 266 509 912 1055 1102 1353 1379 1468 1868 1924 2073 2115
 2180 2272 2344 2412 2444 2492 2531]
[ 27 237 388 449 461 612 623 676 738 770 1106 1132 1187 1270
 1273 1280 1710 1714 1752 1942 2056 2074 2160 2176 2206 2216 2255 2527
 2601 2608 2675 2738 2742 2801 2876 3010]
[ 43  74 133 221 279 486 1026 1272 1454 1581 1665 1763 1877 1981
 2114 2151 2294 2419 2486 2782]
```

데이터 전처리

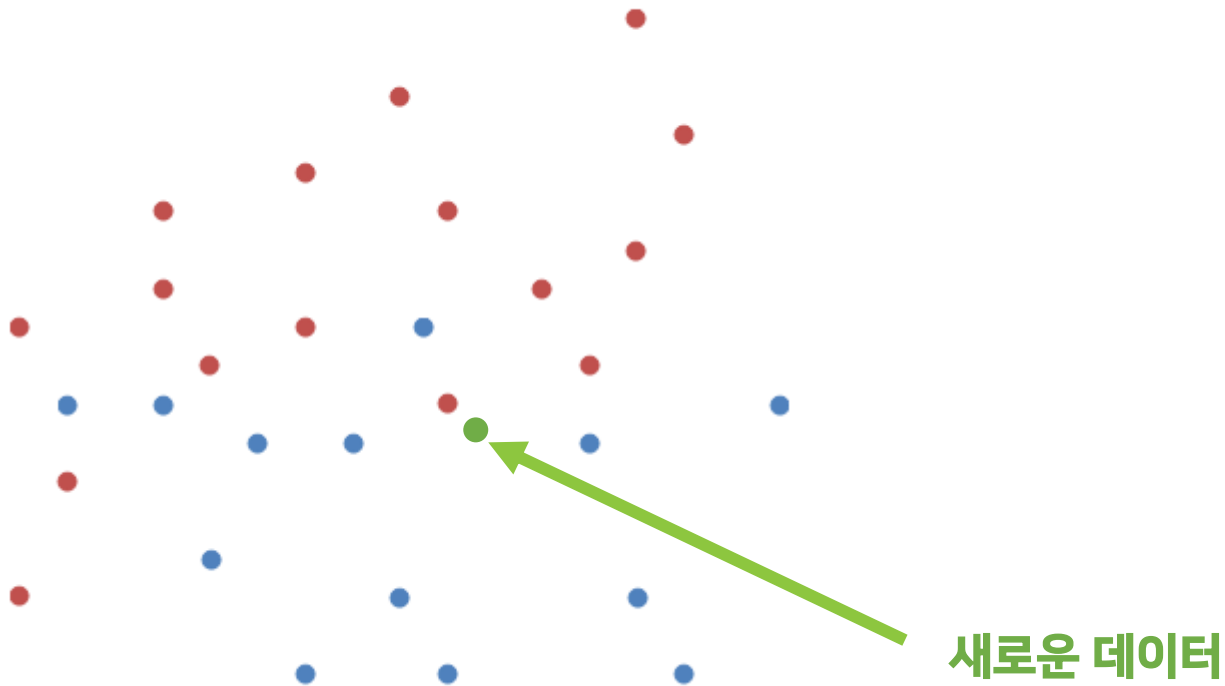
➤ 데이터 분할 하는 법

- ◆ sklearn.model_selection에 train_test_split 라이브러리를 사용하면 간편하다.
- ◆ 기본 분할 비율은 3:1로 되어 있다.
- ◆ stratify: train과 test set의 label이 같은 비율을 갖도록 하는 옵션

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_people,
                                                    y_people, stratify=y_people, random_state=0)
```

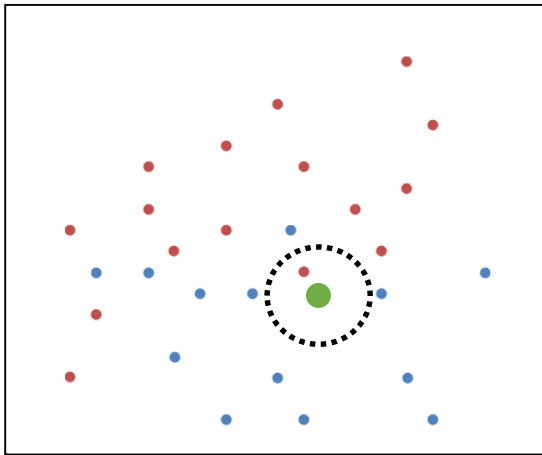

K-Nearest Neighbors

➤ 새로운 데이터의 클래스를 어떻게 예측할까?



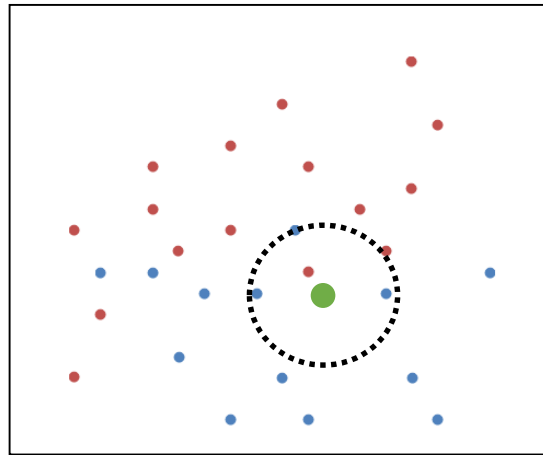
K-Nearest Neighbors

- 1단계: 가장 가까운 k개의 이웃을 선택한다.
- 2단계: 가장 많은 수의 클래스로 새로운 데이터의 클래스를 예측한다.



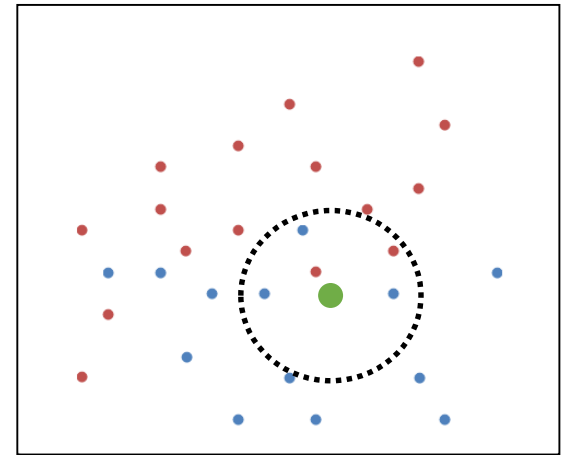
$K = 1$

예측 클래스 = red



$K = 3$

예측 클래스 = blue



$K = 5$

예측 클래스 = blue

차원 축소 전...

➤ 얼굴 분류 정확도: K-Nearest Neighbors

- ◆ 클래스 개수: 62, 무작위 정확도: 0.016

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("Accuracy: {:.2f}".format(knn.score(X_test, y_test)))
# Accuracy = 0.23
```

➤ 매개변수 설명

- ◆ `knn.fit`: train data를 이용하여 K-nearest neighbor 모델을 학습한다.
- ◆ `knn.score`: test data의 평균 accuracy를 반환한다.

주성분분석(PCA)

➤ PCA 사용법

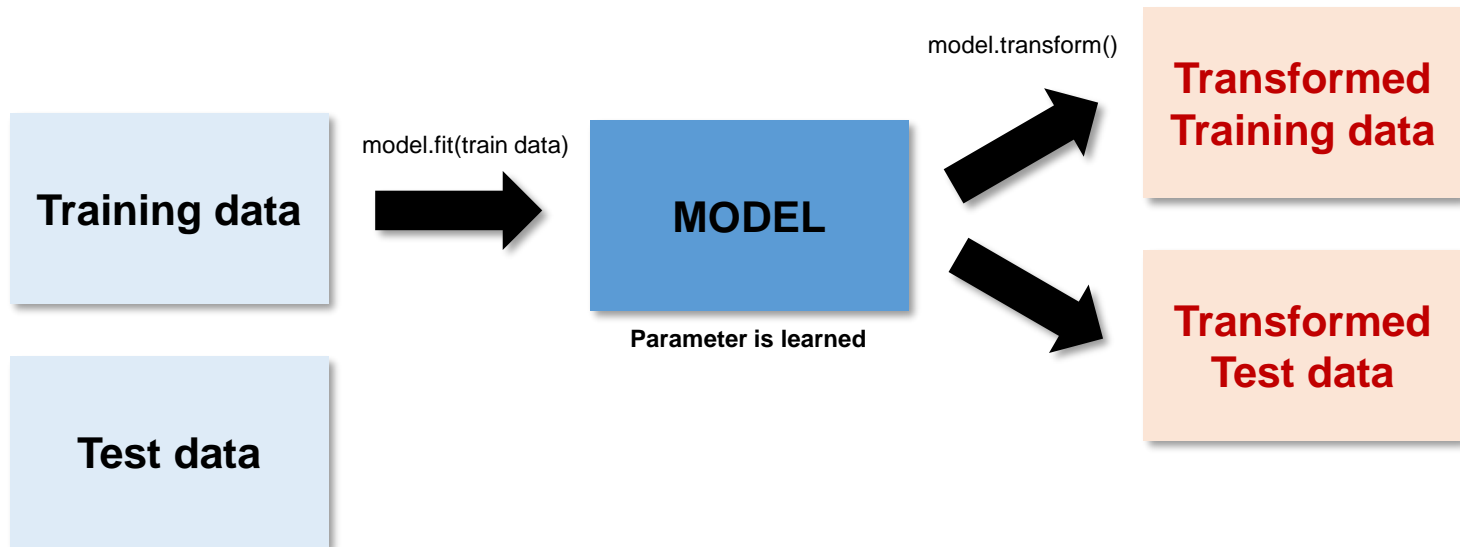
- ◆ sklearn.decomposition에 있는 PCA 라이브러리를 사용하면 간편하다.

```
from sklearn.decomposition import PCA
pca=PCA(n_components=100, whiten=True, random_state=0).fit(X_train)
X_train_pca = pca.transform(X_train) # X_train에 차원축소 방법인 PCA 적용
X_test_pca = pca.transform(X_test) # X_test에 차원축소 방법인 PCA 적용
print(X_train_pca.shape)           # output = (1547, 100)
print(X_test_pca.shape)            # output = (516, 100)
```

주성분분석(PCA)

➤ 함수 추가설명: fit(), transform()

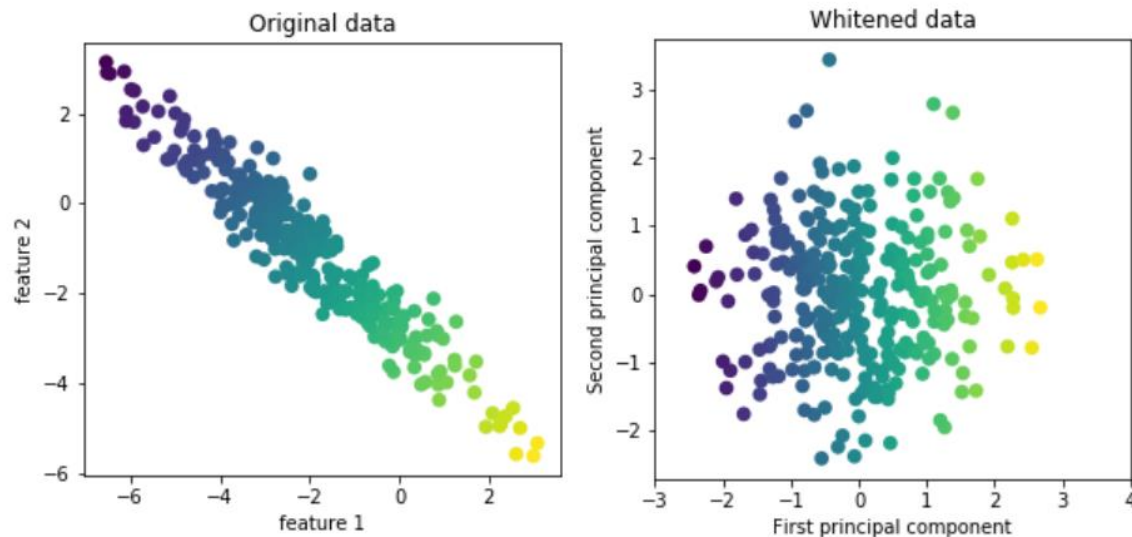
- ◆ fit(): train data로 학습된 모델의 변수를 저장한다.
- ◆ transform(): 학습된 모델을 사용하여 data에 모델(차원축소)을 적용한다.



주성분분석(PCA)

➤ whitening 효과

```
%matplotlib inline  
import mglearn  
mglearn.plots.plot_pca_whitening()
```



주성분분석(PCA)

➤ 얼굴 분류 정확도: 성분 TOP 100개 사용

- ◆ 무작위 정확도 = 0.016, 차원축소 사용하기 전의 정확도 = 0.23

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_pca, y_train)
print("Accuracy: {:.2f}".format(knn.score(X_test_pca, y_test)))
# Accuracy = 0.31
```

주성분분석(PCA)

➤ 시각화하기: 성분 TOP 10개

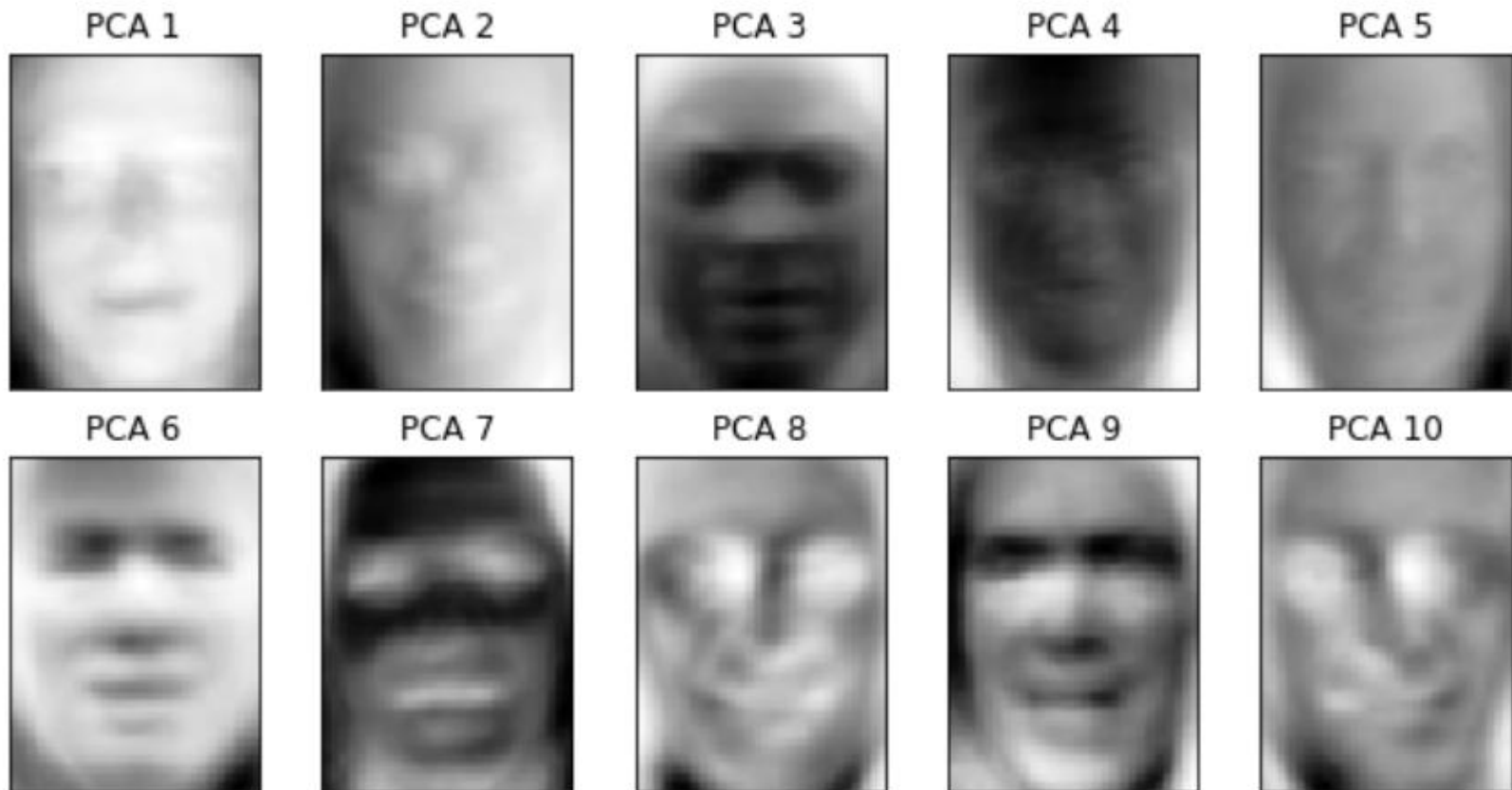
```
%matplotlib inline
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 5, figsize=(10, 5),
                          subplot_kw={'xticks':(), 'yticks':()})

image_shape =lfw_people.images[0].shape          # output = (87, 65)

for i, (component, ax) in enumerate(zip(pca.components_, axes.ravel())):
    ax.imshow(component.reshape(image_shape), cmap='Greys')
    ax.set_title("PCA {}".format((i+1)))
```

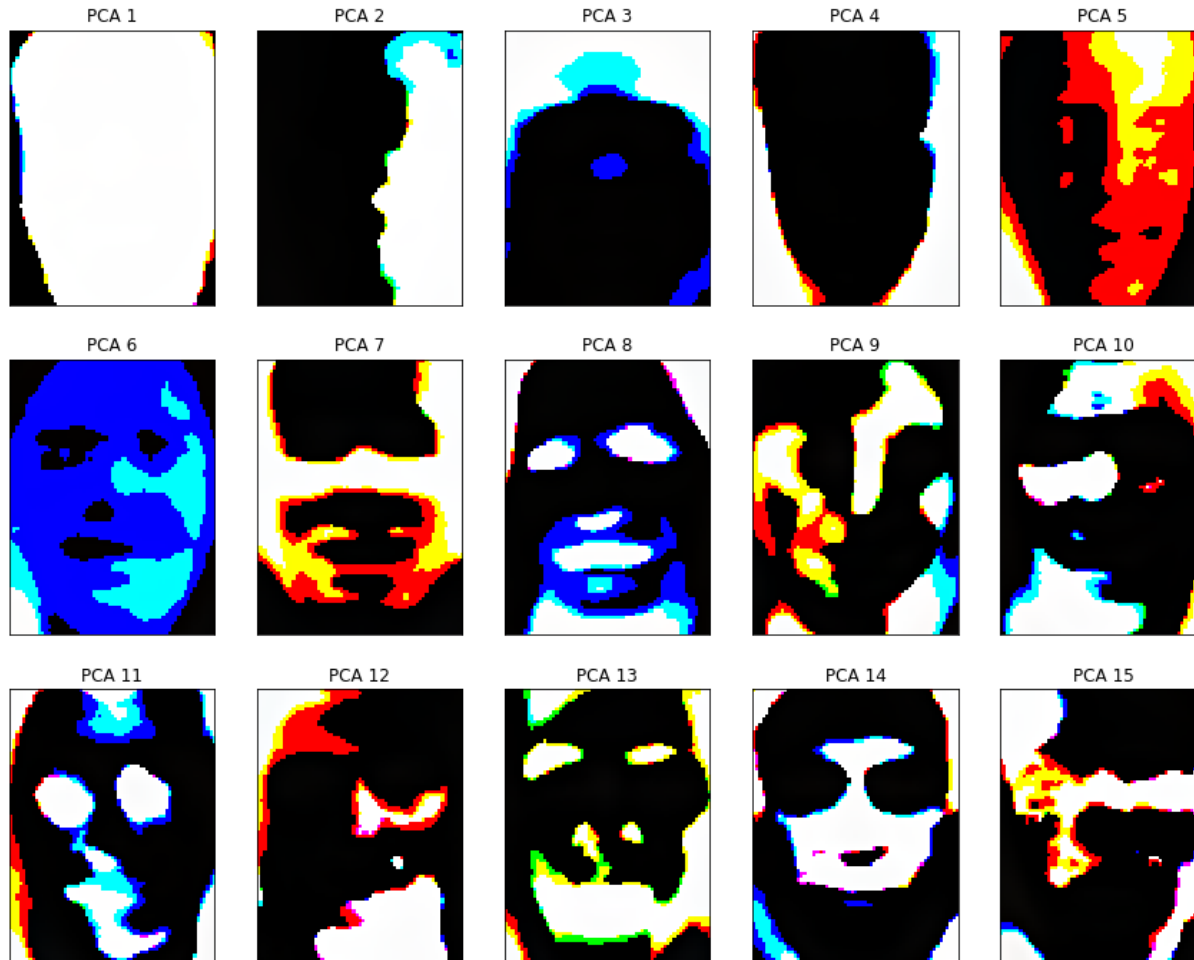

주성분분석(PCA)

➤ 시각화하기: 성분 TOP 10개



주성분분석(PCA)

➤ 시각화하기: 성분 TOP 15개 (데이터셋 = color)



주성분분석(PCA)

➤ 성분 갯수별 얼굴분류 정확도 비교

```
for i in range(50, 510, 50):  
    pca=PCA(n_components=i, whiten=True, random_state=0).fit(X_train)  
    X_train_pca = pca.transform(X_train)  
    X_test_pca = pca.transform(X_test)  
    knn = KNeighborsClassifier(n_neighbors=1)  
    knn.fit(X_train_pca, y_train)  
    print("Component: {}".format(i), "Accuracy:  
        {:.2f}".format(knn.score(X_test_pca, y_test)))
```

주성분분석(PCA)

➤ 성분 갯수별 얼굴분류 정확도 비교

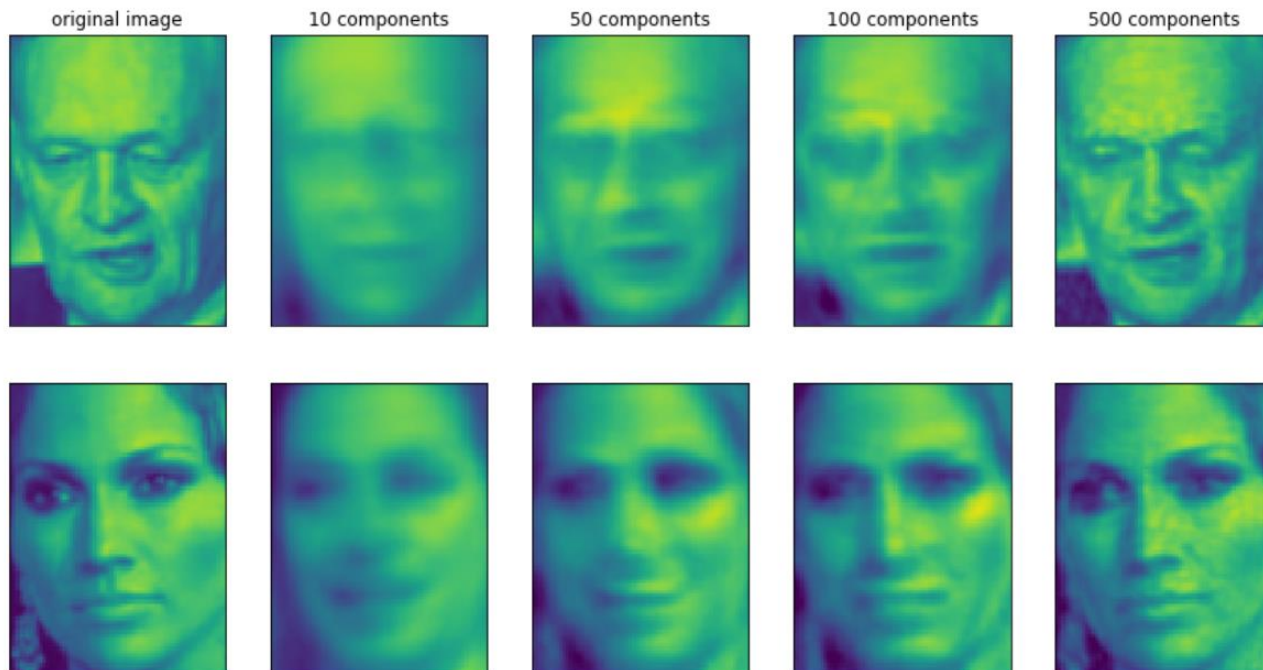
Component:	50	Accuracy:	0.31
Component:	100	Accuracy:	0.31
Component:	150	Accuracy:	0.30
Component:	200	Accuracy:	0.26
Component:	250	Accuracy:	0.23
Component:	300	Accuracy:	0.21
Component:	350	Accuracy:	0.17
Component:	400	Accuracy:	0.16
Component:	450	Accuracy:	0.16
Component:	500	Accuracy:	0.14

Component 개수가 50~100일 때 Accuracy가 높다.

주성분분석(PCA)

➤ 성분 갯수별 얼굴 재구성

```
import mglearn
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)
```



비음수행렬분해(NMF)

➤ NMF 란?

	pixel 1	pixel 2	...	pixel 10,000
Obama	0.6	0		0.9
Bush	0.2	0.2		0.4
Gandhi	0.5	0.4		0.8

≈

	feature 1	feature 2
Obama	0.8	0.3
Bush	0.2	0.7
Gandhi	0.7	0.3

X

	pixel 1	pixel 2	...	pixel 10,000
feature 1	0.8	0		0.9
feature 2	0	0.2		0.4

가중치 행렬

특성 행렬

비음수행렬분해(NMF)

➤ NMF 사용법

- ◆ sklearn.decomposition에 있는 NMF 라이브러리를 사용하면 간편하다.

```
from sklearn.decomposition import NMF
nmf=NMF(n_components=100, random_state=0).fit(X_train)
X_train_nmf = nmf.transform(X_train)      # apply NMF to X_train
X_test_nmf = nmf.transform(X_test)       # apply NMF to X_test
print(X_train_nmf.shape)                  # output = (1547, 100)
print(X_test_nmf.shape)                   # output = (516, 100)
```

비음수행렬분해(NMF)

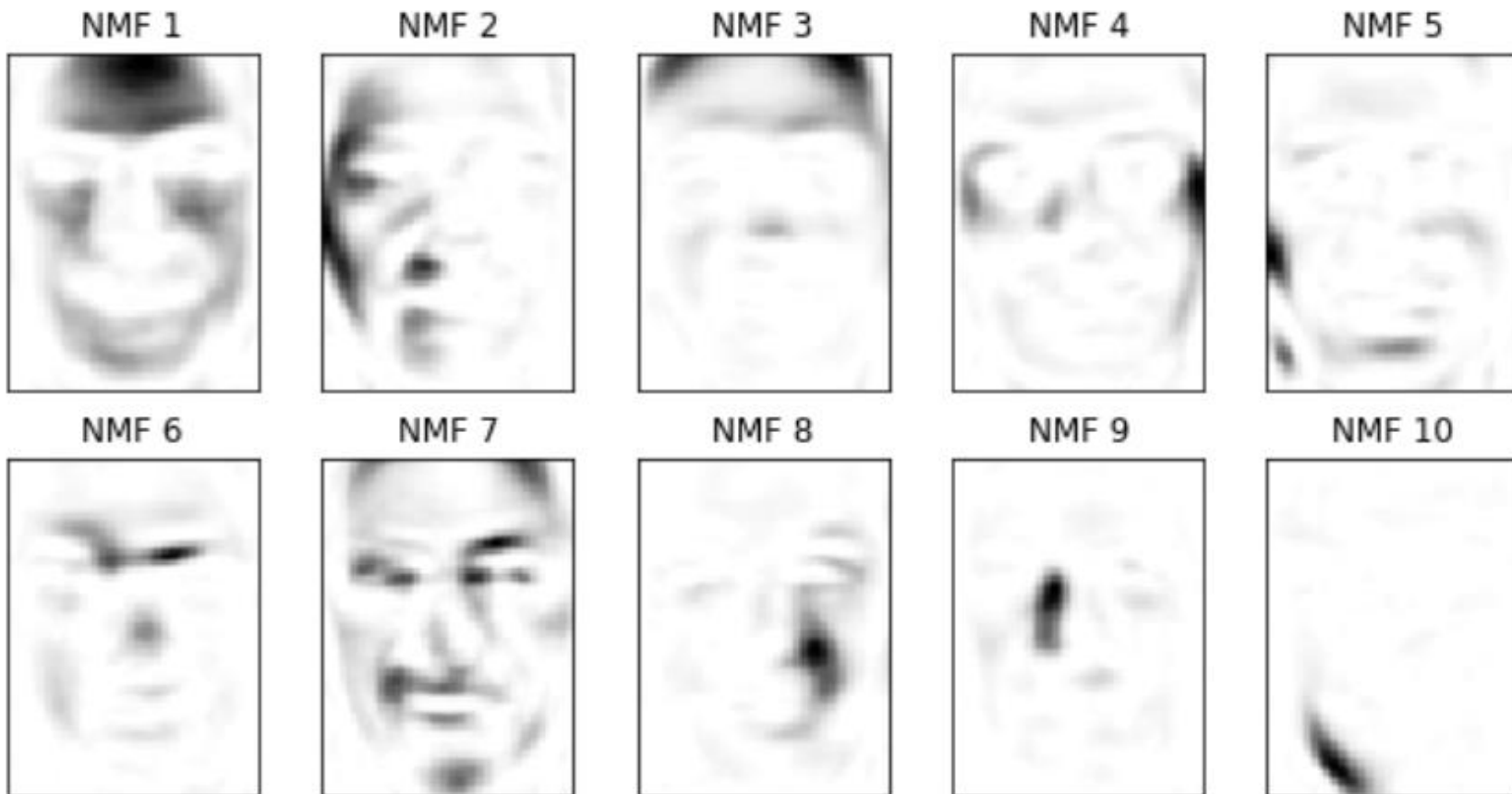
➤ 시각화하기: 성분 TOP 10개

```
fig, axes = plt.subplots(2, 5, figsize=(10, 5),
                          subplot_kw={'xticks':(), 'yticks':()})

for i, (component, ax) in enumerate(zip(nmf.components_, axes.ravel())):
    ax.imshow(component.reshape(image_shape), cmap='Greys')
    ax.set_title("NMF {}".format((i+1)))
```


비음수행렬분해(NMF)

➤ 시각화하기: 성분 TOP 10개



비음수행렬분해(NMF)

➤ 얼굴 분류 정확도: 성분 TOP 100개 사용

- ◆ 무작위 정확도 = 0.016, PCA 정확도 = 0.31

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_nmf, y_train)
print("Accuracy: {:.2f}".format(knn.score(X_test_nmf, y_test)))
# Accuracy = 0.23
```

비음수행렬분해(NMF)

➤ 성분 갯수별 얼굴분류 정확도 비교

- ◆ warning: 시간 매우 오래 걸림

```
for i in range(50, 510, 50):  
    nmf=NMF(n_components=i, random_state=0).fit(X_train)  
    X_train_nmf = nmf.transform(X_train)  
    X_test_nmf = nmf.transform(X_test)  
    knn = KNeighborsClassifier(n_neighbors=1)  
    knn.fit(X_train_nmf, y_train)  
    print("Component: {}".format(i), "Accuracy:  
          {:.2f}".format(knn.score(X_test_nmf, y_test)))
```

비음수행렬분해(NMF)

➤ 성분 갯수별 얼굴분류 정확도 비교

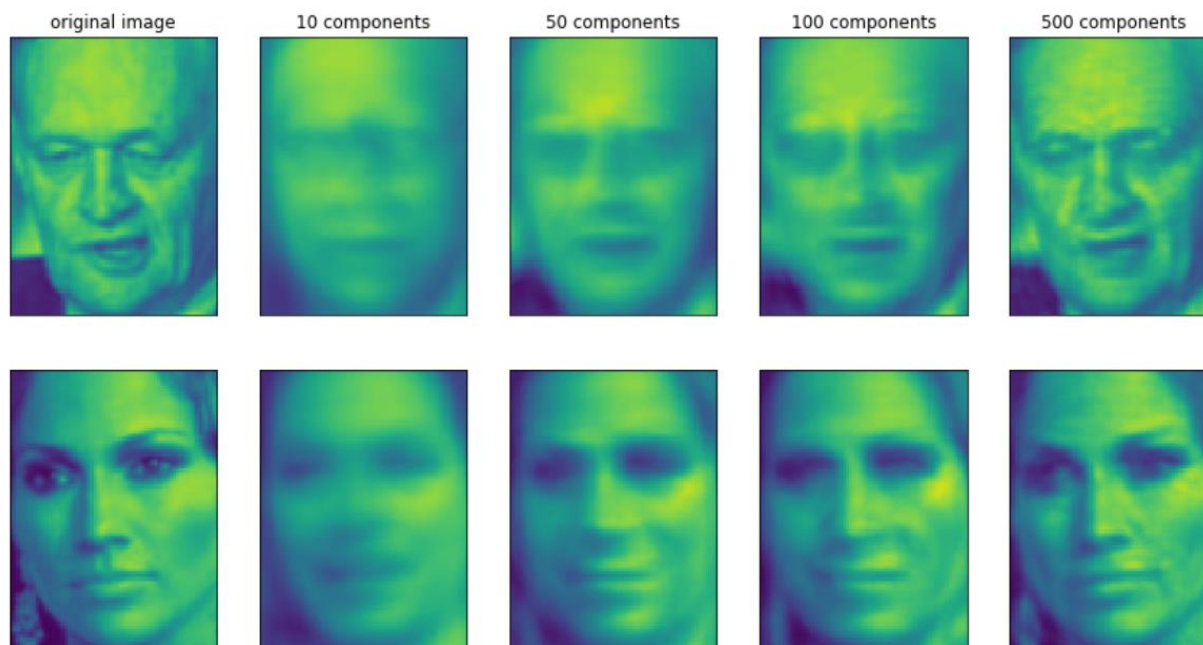
Component: 50	Accuracy: 0.24
Component: 100	Accuracy: 0.23
Component: 150	Accuracy: 0.26
Component: 200	Accuracy: 0.22
Component: 250	Accuracy: 0.21
Component: 300	Accuracy: 0.22
Component: 350	Accuracy: 0.22
Component: 400	Accuracy: 0.22
Component: 450	Accuracy: 0.23
Component: 500	Accuracy: 0.22

Component 개수 별 큰 차이가 없다.

비음수행렬분해(NMF)

➤ 성분 갯수별 얼굴 재구성

```
import mglearn
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape) # time: 8.8min
```



요약

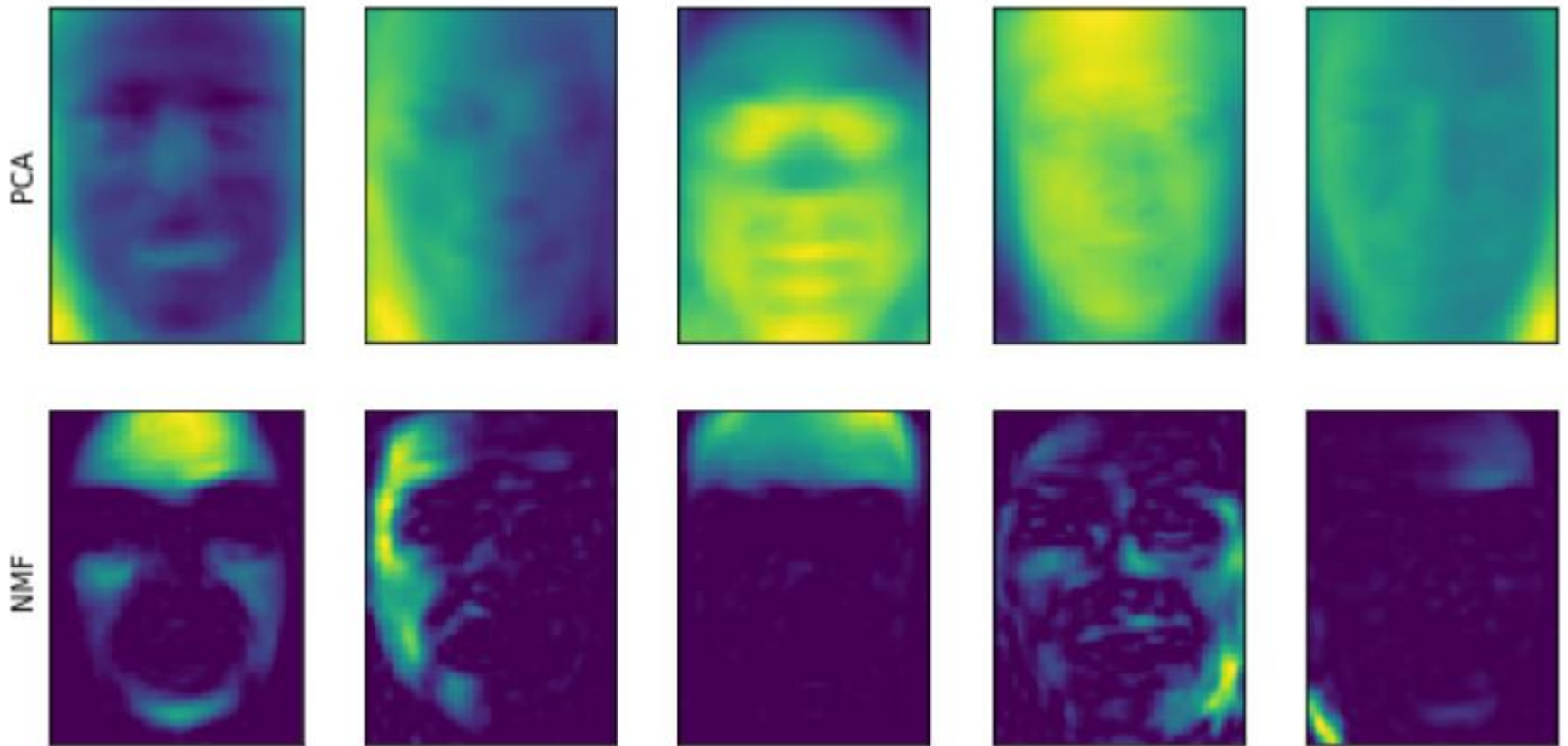
➤ 특성 추출 (Feature Extraction)

```
fig, axes = plt.subplots(2, 5, figsize=(10, 5),
                          subplot_kw={'xticks': (), 'yticks': ()})
for ax, fe_pca, fe_nmf in zip(axes.T, pca.components_,
                              nmf.components_):
    ax[0].imshow(fe_pca.reshape(image_shape))
    ax[1].imshow(fe_nmf.reshape(image_shape))

axes[0,0].set_ylabel("PCA")
axes[1,0].set_ylabel("NMF")
```

요약

➤ 특성 추출 (Feature Extraction)



요약

➤ 이미지 재구성(Image Reconstruction)

```
X_reconstructed_pca=pca.inverse_transform(pca.transform(X_test))
X_reconstructed_nmf=np.dot(nmf.transform(X_test), nmf.components_)
fig, axes=plt.subplots(3, 5, figsize=(10, 8), subplot_kw={'xticks': (),
    'yticks':()})
for ax, ori, ir_pca, ir_nmf in zip(axes.T, X_test, X_reconstructed_pca,
    X_reconstructed_nmf):
    ax[0].imshow(ori.reshape(image_shape))
    ax[1].imshow(ir_pca.reshape(image_shape))
    ax[2].imshow(ir_nmf.reshape(image_shape))

axes[0,0].set_ylabel("Original")
axes[1,0].set_ylabel("PCA")
axes[2,0].set_ylabel("NMF")
```


요약

➤ 이미지 재구성(Image Reconstruction)



추천 시스템 **(Recommender Systems)**

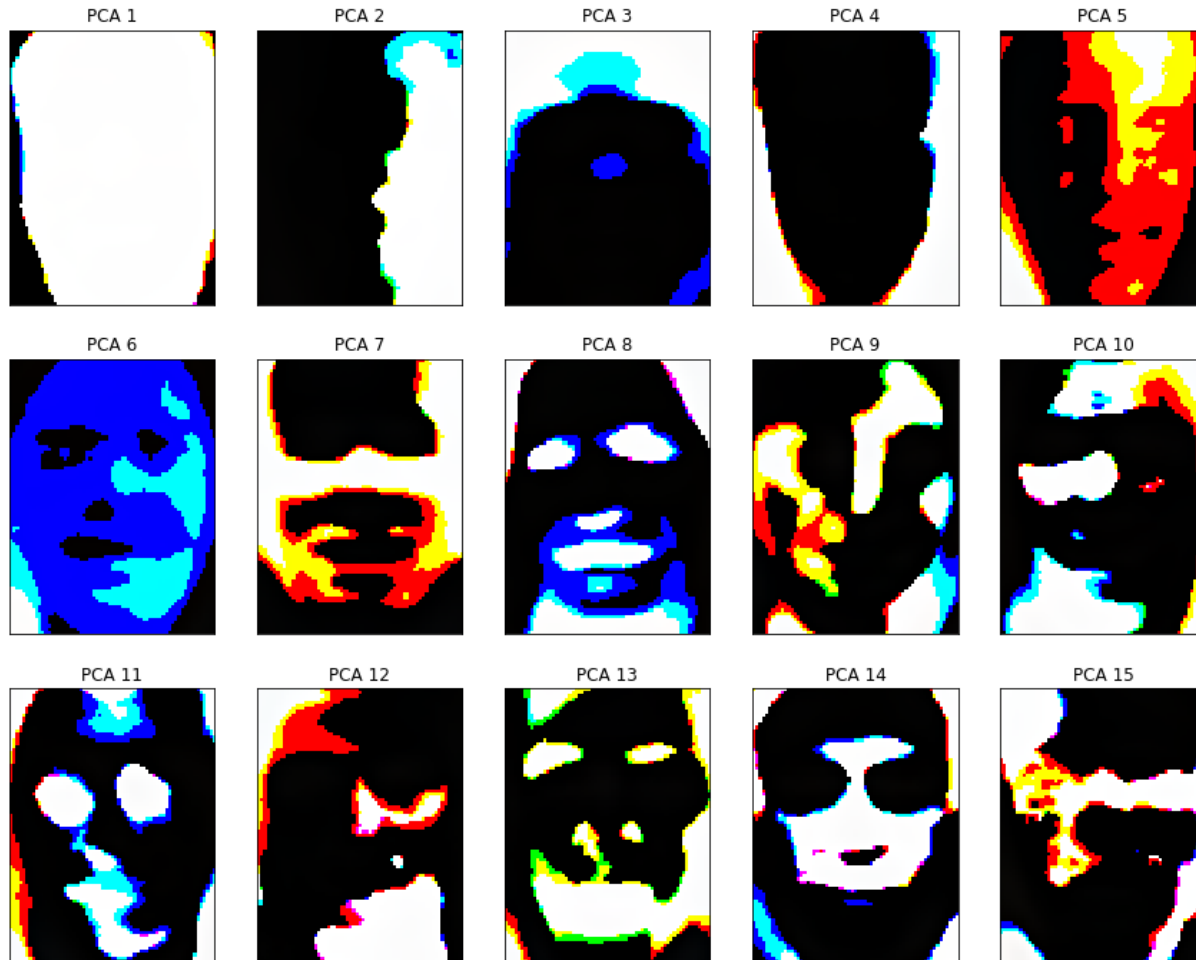
it's
Q & A
TIME!



Appendix.

주성분분석(PCA)

➤ Visualize top-15 components (set color=True)



주성분분석(PCA)

PCA 1



difference between **face** and **background**

PCA 2

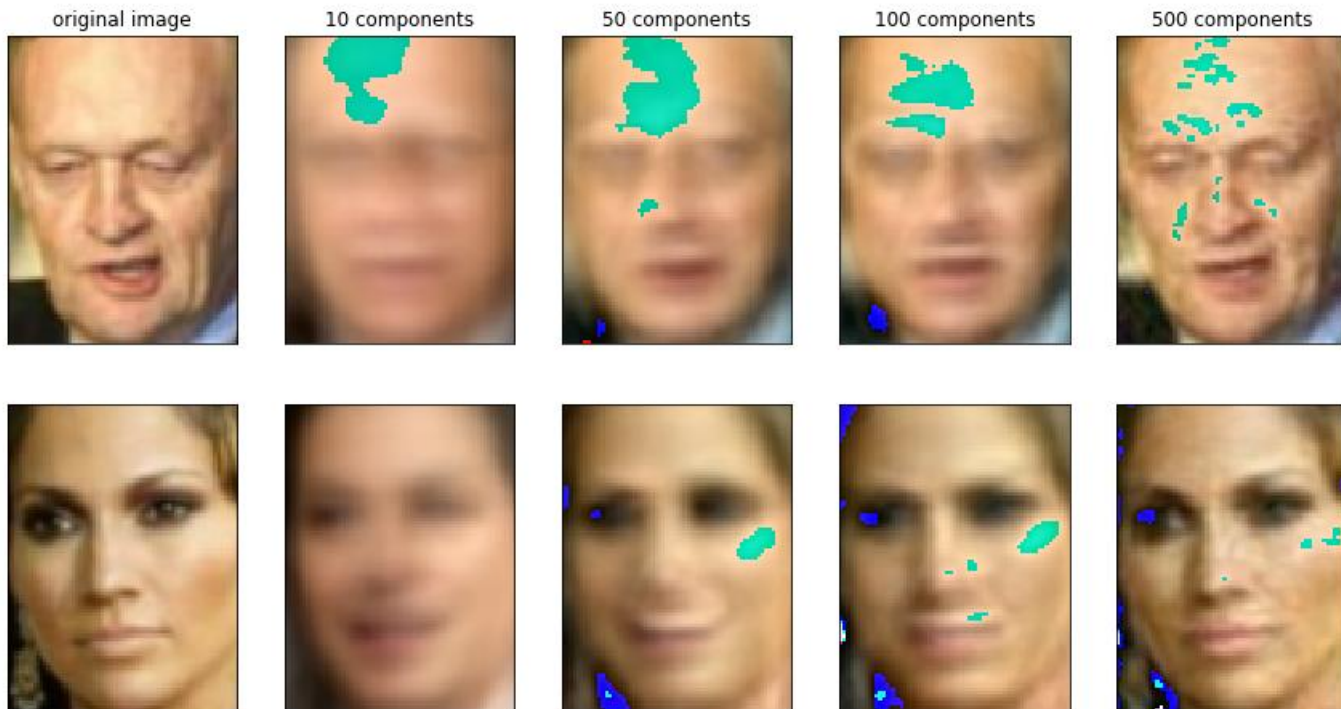


difference between **left** and **right**

주성분분석(PCA)

➤ 성분 갯수별 얼굴 재구성 (데이터셋이 color일 경우)

```
import mglearn
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)
```



K-means

➤ How to apply K-means

```
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=100, random_state=0).fit(X_train)
X_train_kmeans = kmeans.transform(X_train)
X_test_kmeans = kmeans.transform(X_test)
print(X_train_kmeans.shape)           # output = (1547, 100)
print(X_test_kmeans.shape)            # output = (516, 100)
```


K-means

➤ Face Classification

- ◆ random accuracy = $0.016 = 1/62$
- ◆ PCA accuracy = 0.31
- ◆ NMF accuracy = 0.23

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_kmeans, y_train)
print("Accuracy: {:.2f}".format(knn.score(X_test_kmeans, y_test)))
# Accuracy = 0.10
```