제2차 빅데이터 분석 교육과정 강의교재

- 가완성본 -

ABRC 인공지능-빅데이터연구센터
AI-BIG DATA RESEARCH CENTER

PCA와 LDA

1. 차원축소

1.1 차원 (Dimensionality)

- 독립 변수의 개수를 의미
- 예
  - 타이타닉호 탑승자들의 생존 여부에 영향을 주는 세 가지 요소 : 좌석, 성별, 나이
  - 타이타닉호 탑승자들의 생존 여부를 결정하는 파라미터 공간은 3차원 (좌석, 성별, 나이)

데이터의 차원이란, 독립변수의 개수를 의미합니다. 쉽게 예를 들어 살펴볼까요? 타이타닉호 탑승자들의 생존 여부에 영향을 주는 요소가 탑승자들의 좌석, 성별, 나이, 이렇게 세 가지 요소라고 가정해 봅시다. 이때 타이타닉호 탑승자들의 생존 여부를 결정하는 파라미터 공간은 바로 독립변수인 좌석, 성별, 나이, 세 가지로 이루어진 3차원 공간입니다. 이렇게 데이터의 독립변수 개수에 따라 데이터의 차원이 결정됩니다. 하지만 이 차원이 클 경우에는 효율적인 작업이 힘들어 집니다. 그러한 상태를 차원의 저주라고 합니다.

1.2 차원의 저주 (Curse of Dimensionality)

- 데이터의 차원이 커질수록 차원의 저주는 커질수록 해당 공간을 설명하기 위한 데이터의 양이 점점 많이 필요하게 됨
- 처음 가지고 있던, 적은 데이터로 공간을 설명해야 하기 때문에 과최적화 문제가 발생
- 그 결과 모델의 성능이 떨어지는 문제가 발생할 수 있음
- 이에 따라 핵심이 되는 파라미터들을 선택해 데이터의 차원을 축소하여 과최적화 되는 것을 방지할 수 있음

데이터의 차원이 커질수록, 즉 변수의 개수가 증가할수록, 모델 구성에 필요한 데이터의 개수가 기하급수적으로 증가하는데, 이 때 해당 특징의 데이터가 제공되지 않는 경우 과최적화 등의 문제가 발생합니다. 이러한 문제점들을 극복하기 위해 데이터에서 핵심이 되는 특징들 또는 이들 특징들로부

터 얻을 수 있는 주성분만 선택해 사용하면 이런 문제를 해결할 수 있습니다. 학습할 대표적인 두 가지 차원축소 방법은 PCA와 LDA입니다.

1.3 차원축소 방법

- Principal Component Analysis (PCA), 주성분 분석
  데이터 분포의 분산이 큰 축을 찾는 방법
- Linear Discriminant Analysis (LDA), 선형 판별 분석
  데이터의 클래스 정보를 유지하면서 분리하는 축을 찾는 방법

PCA는 principal component analysis의 약자로, 주성분 분석이라고 합니다. PCA는 데이터 분포의 분산이 큰 임의의 축을 찾는 것이 첫 단계에서 할 일입니다. 두 번째 대표적인 방법은 LDA, linear discriminant analysis입니다. 선형 판별 분석이라고 하며, 데이터의 클래스 정보를 유지하면서 분리하는 축을 찾는 방법을 말합니다.

그림에서 PCA는 붉은색 데이터와 푸른색 데이터의 분포를 나타내는 선들 중, 데이터를 사상 시켰을 경우 가장 큰 분산을 가진 축을 찾는 작업 을 하고 있음을 확인할 수 있습니다. 그림을 다시 살펴보겠습니다.

데이터를 직선에 사상시킨 결과, 붉은색 클래스와 푸른색 클래스가 적절하게 분리되어 있는 것을 알 수 있습니다. 이와 같이 클래스 정보를 유지하면서 데이터를 분리할 수 있는 축을 찾는 방법을 LDA라고 합니다. 그림 1은 PCA와 LDA에 대한 내용입니다.

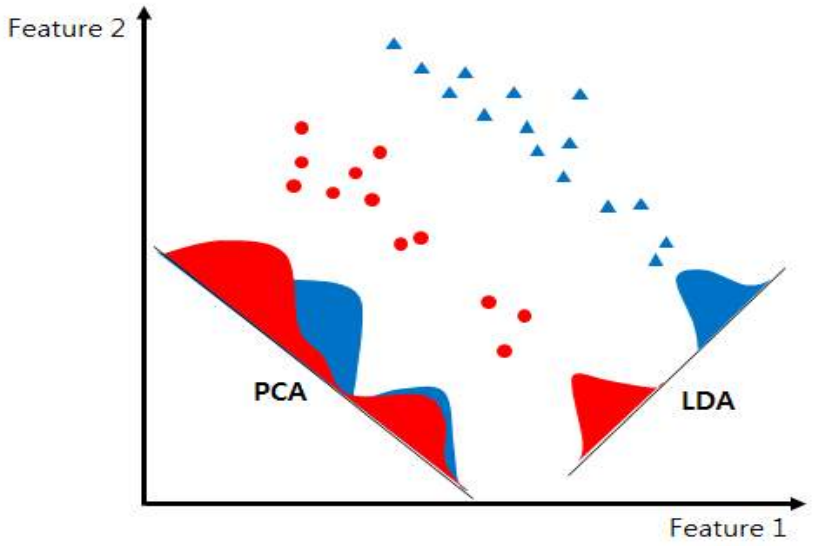

그림 1 PCA와 LDA

지금부터 PCA, LDA, 두 방법을 차례대로 자세히 살펴보도록 하겠습니다. 살펴볼 첫 번째 방법은 PCA 입니다.

2. PCA

2.1 PCA

PCA는 principal component analysis의 약자로 말 그대로 주성분 분석이라고 합니다. 앞에서 살펴보았듯이 고차원의 데이터를 저차원의 데이터로 변환하는 차원축소 방법 중 하나입니다. PCA를 사용해 찾은 주성분은 데이터 분포의 특성을 가장 잘 설명할 수 있는 벡터입니다.

즉, 데이터의 분산이 가장 큰 축이 첫 번째 주성분이 되고, 두 번째로 분산이 큰 축이 두 번째 주성분이 됩니다. PCA를 사용해 찾은 요소들은 데이터의 분포를 가장 잘 나타내는 요소들 이므로, 이를 사용해 데이터를 분석할 수 있습니다.

그림을 통해 쉽게 살펴보겠습니다. 다음의 그림은 2차원 공간에 분포하고 있는 데이터의 산포도입니다. 이 붉은색 점들의 분포를 가장 잘 표현하는 축 두 개가 바로 두 개의 파란색 선입니다.

1^{st} PC(principal component), 즉 첫 번째 주성분 벡터를 볼까요. 이 축에 데

이터를 사상시킬 경우, 데이터의 분산이 가장 크다는 것을 알 수 있습니다. 2^{nd} PC(principal component), 즉 두 번째 주성분 벡터 또한 첫 번째 주성분 벡터 다음으로 데이터의 분산이 큰 축입니다. 그림 2는 데이터 분포와 PCA의 예입니다. 그럼 지금부터 이 주성분 벡터를 구하는 방법을 살펴보겠습니다.

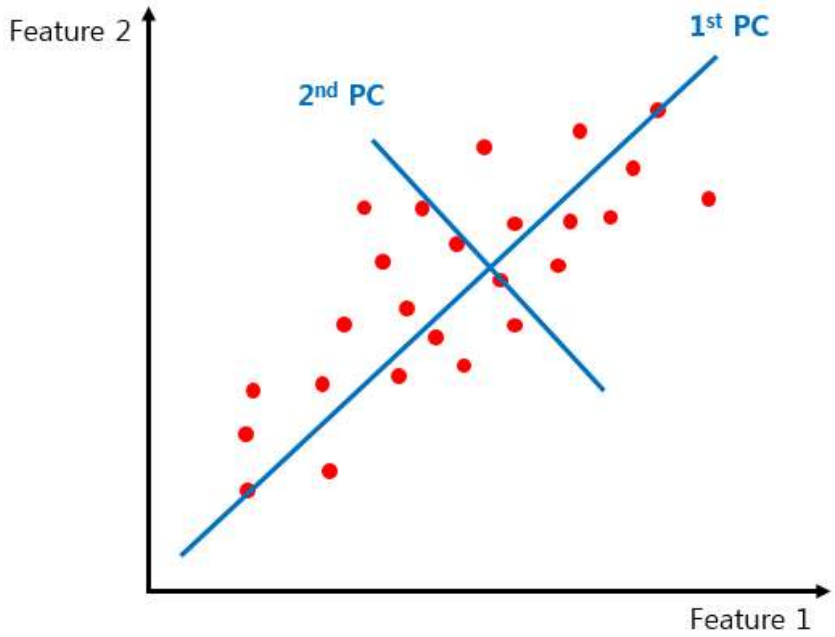

그림 2 데이터 분포와 PCA의 예

2.2 고유벡터(Eigenvectors)와 고유값(Eigenvalues)

주성분은 고유벡터와 고유값을 사용해 구합니다. 고유벡터와 고유값은 항상 쌍으로 존재 합니다.이때 고유벡터는 데이터의 분포를 나타내는 선입니다. 그리고 고유값은 해당 고유벡터에 데이터를 사상시켰을 때 데이터가 분포하는 분산을 의미합니다.

예를 들어 살펴볼까요. 다음의 그림에서 붉은색 점은 데이터를 의미하는데, 이 데이터의 분포를 나타내는 선이 고유벡터입니다. 그리고 이 고유벡터에 데이터를 사상시킨 후 분산을 구하면 이 값이 바로 고유값이 됩니다. 그림 3은 고유벡터와 고유값에 대한 내용입니다.

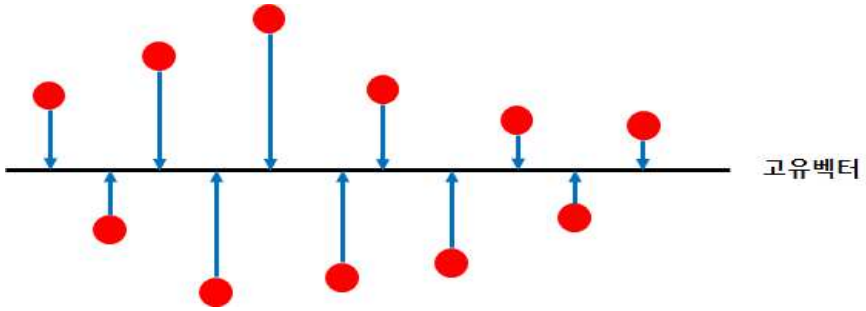

그림 3 고유벡터와 고유값

2.3 PCA를 사용한 데이터 재구성

이렇게 데이터의 주성분을 구해서 데이터에 어떻게 적용해야 할까요? 우선 데이터의 고유벡터와 고유값 쌍의 개수는 데이터가 가지고 있는 속성, 차원의 개수와 같습니다. 예를 들어 데이터가 2개의 속성인 나이, 사용 시간을 가지고 있을 경우, 이 데이터의 고유벡터와 고유값 또한 2쌍입니다.

따라서 2차원 데이터의 경우, 2개의 고유 벡터가 수직일 때, 데이터가 두 고유벡터를 기준으로 큰 분산으로 분포하고 있다는 것을 의미합니다. 그러면 이 두 고유벡터를 기준으로 데이터를 재구성하는 것이 가능하게 됩니다. 그림 4는 고유벡터의 예에 대한 내용입니다.

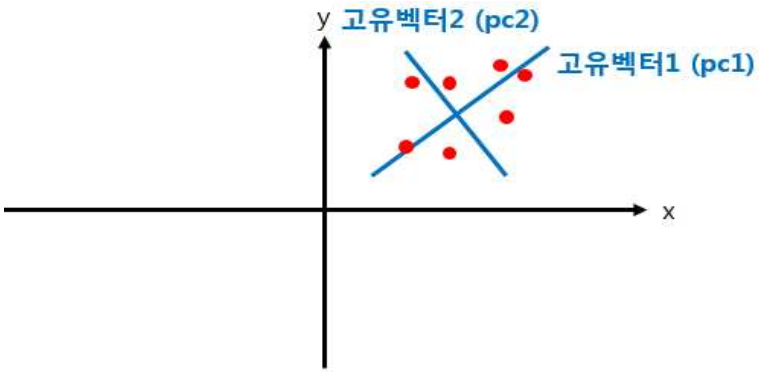

그림 4 고유벡터 예

그림에서 1 사분면에 분포하고 있는 데이터의 주성분 벡터는 파란색 벡터, 고유벡터 1과 고유벡터 2가 됩니다.이 두 개의 고유벡터를 새로운 축 new x, new y로 삼아 새로운 차원을 구성할 수 있습니다.

이때 데이터에는 변화가 생기지 않습니다. 다만 데이터의 분포를 나타내는

축만 변화하게 됩니다. 다음 그림이 바로 앞에서 확인한 주 성분 벡터 두 개를 새로운 축으로 지정한 모습입니다. 두 개의 새로운 축 new x와 new y를 기준으로 데이터의 분포를 표시하고 있습니다. 이와 같이 새로운 축을 기반으로 데이터가 큰 분산을 가지고 분포할 경우, 데이터를 통해 정보를 찾기가 쉬워집니다. 그림 5는 재구성한 고유벡터의 예에 대한 내용입니다.

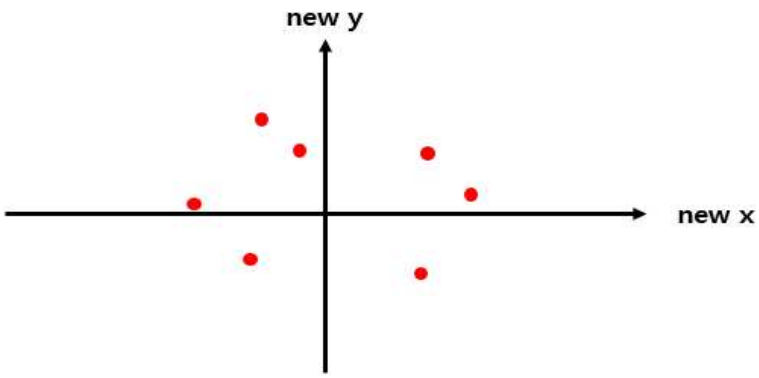

그림 5 재구성한 고유벡터의 예

이번에는 PCA와 함께 대표적으로 사용되는 차원 축소 방법인 LDA에 대해 학습하겠습니다.

3. LDA

3.1 LDA

LDA는 linear discriminant analysis의 약자로, 선형판별분석이라고 합니다. LDA는 데이터를 클래스 별로 잘 분리하는 벡터를 찾는 방법입니다. 그림을 통해 좀 더 쉽게 살펴보겠습니다. 다음 그림은 두 개의 속성 feature 1과 feature 2를 축으로 데이터의 분포를 나타낸 그래프입니다. 여기에서 붉은색 클래스와 푸른색 클래스를 잘 분리할 수 있는 벡터를 찾아내는 방법이 바로 LDA입니다. 그림 6은 LDA의 예입니다.

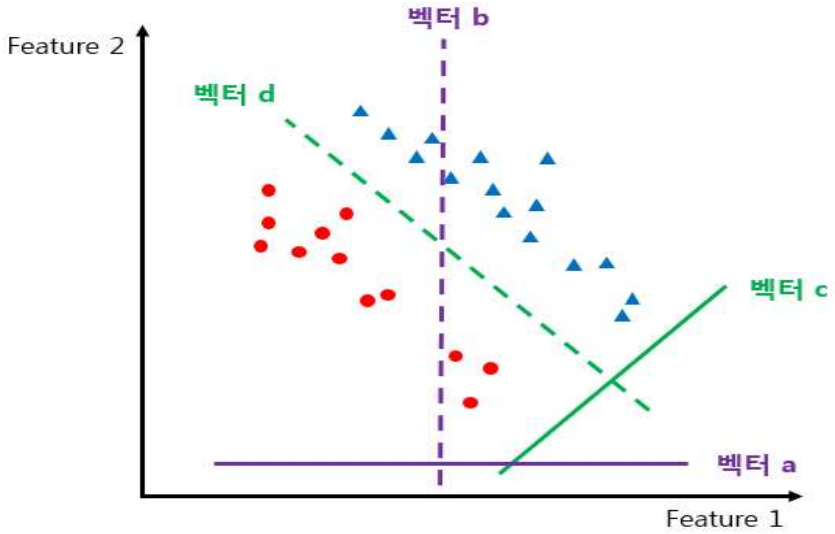

그림 6 LDA의 예

두 클래스를 잘 분리할 벡터를 찾기 위해 벡터 b와 벡터 d를 기준으로 데이터를 분리합니다. 그리고 각 벡터에 수직인 벡터 a와 벡터 c에 데이터를 사상시킵니다. 사상 시킨 결과는 다음 그림과 같습니다. 우선 벡터 b를 기준으로 데이터를 분리하고, 이에 수직인 벡터 a에 데이터를 사상시킨 첫 번째 그림을 살펴보겠습니다. 그림에서 확인할 수 있듯이 두 개의 클래스가 제대로 분리되어 있지 않음을 확인할 수 있습니다. 그림 7은 LDA의 예에서 벡터 a와 b에 대한 내용입니다.

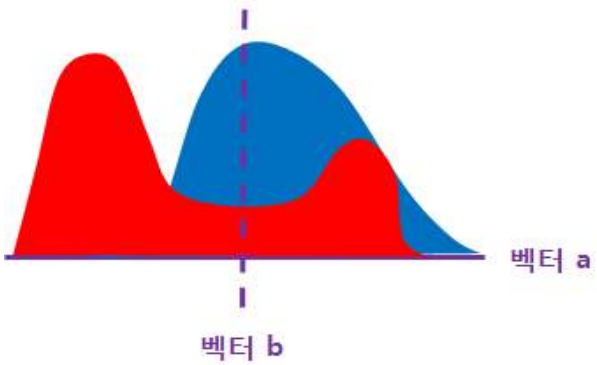

그림 7 LDA의 예 - 벡터 a와 벡터 b

그럼 다음 그림을 살펴보겠습니다. 벡터 d를 기준으로 데이터를 분리하고, 이에 수직인 벡터 c에 데이터를 사상시킨 결과를 볼 수 있는데요. 여기에서 우리는 벡터 c에 데이터가 클래스 별로 적절하게 분리되어 있음을 확인할 수 있습니다. 그림 8은 LDA의 예에서 벡터 c와 d에 대한 내용입니다.

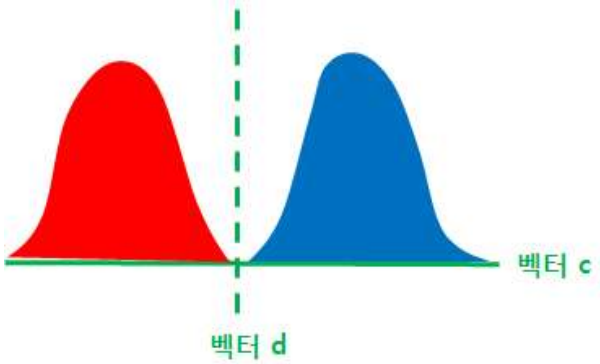

그림 8 LDA의 예 - 벡터 c와 벡터 d

이와 같이 LDA는 데이터를 사상시켰을 때 다음의 조건들을 만족하는 벡터를 찾는 작업입니다. 첫 번째 조건은 각 클래스의 중심간 거리가 최대인 벡터 즉, 클래스간 분리를 잘 이루어 내야 한다는 것입니다. 두 번째 조건은 각 클래스 내 데이터의 분산이 최소인 벡터 즉, 같은 클래스 내의 데이터들이 밀집되게 분리해야 한다는 것입니다. 따라서 벡터 a를 사용하면 두 클래스를 잘 분리시키면서 차원을 축소할 수 있습니다.

지금까지 우리는 PCA와 LDA가 각각 어떠한 방법으로 데이터의 차원을 축소하는지 살펴보았습니다. 그럼 지금부터 데이터를 사용해 PCA와 LDA 실습을 해보도록 하겠습니다.

4. 데이터를 사용한 실습

4.1 필요한 패키지 import

분석에 필요한 패키지들을 import 해야 합니다. 차원 축소 결과를 시각화해 살펴보기 위해 첫 번째 줄에서 환경을 설정합니다. 그리고 pyplot 패키지를 import 합니다. scikit-learn에서 기본 제공되는 데이터 셋을 사용하기 위해 datasets 모듈을 import 합니다. 마지막으로 PCA와 LDA를 진행하기 위해 각 모듈을 import 합니다. 그림 9는 필요한 패키지 import하는 내용입니다.

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

그림 9 필요한 패키지 import

4.2 원본 데이터 확인

4.2.1 데이터 살펴보기

분석에 사용할 데이터를 살펴보겠습니다. scikit-learn에서 제공하는 toy data 중 유방암 데이터를 사용합니다. 이 데이터를 사용하기 위해 load breast cancer 함수를 통해 데이터를 로드하고, 변수 data에 저장합니다. 그리고 데이터의 속성들을 확인하기 위해 feature names 함수를 사용해 출력해 봅니다. 그 결과 다음과 같은 속성들의 이름을 확인할 수 있고, 이 유방암 데이터가 30개의 속성, 즉 30차원의 데이터 셋이라는 것을 알 수 있습니다. 그림 10은 로드한 breast cancer 데이터의 속성 확인하는 내용입니다.

```
data = datasets.load_breast_cancer()
data.feature_names
```

```
array(['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'],
      dtype='<U23')
```

그림 10 데이터의 속성 확인

4.2.2 데이터 셋 준비

이 원본 데이터의 분포를 시각화 해 살펴보기 위해 첫 번째와 두 번째 속성, 두 가지만 선택해 사용할 예정입니다. 이를 위해 data의 data 즉, 속성 부분에서 앞 두 개의 속성을 선택해 변수 x에 저장합니다. 그러면 x에는 mean radius와 mean texture가 저장됩니다.

다음으로 데이터의 클래스 정보를 target 함수를 사용해 가져와 변수 y에 저장합니다. 더불어 클래스 이름을 target names 라는 함수를 사용해 가져와 변수 target names에 저장합니다. 그러면 target names에는 악성, malignant와 양성, benign 정보가 저장됩니다. 그림 11은 사용할 데이터 셋 준비에 대한 내용입니다.

```
x = data.data[:, :2]
y = data.target
target_names = data.target_names
```

그림 11 사용할 데이터 셋 준비

4.2.3 산포도 표현

이제 준비한 원형 데이터를 산포도로 그려 살펴보겠습니다. 우선 산포도를 그리기 위한 figure의 크기를 fig size를 사용해 설정합니다. 그리고 클래스마다 다른 색상으로 표시하기 위해 color 변수에 red, blue 값을 저장해 둡니다.

이제 for 문을 사용해 데이터를 하나씩 산포도에 그리는 작업을 진행합니다. 우선 zip 함수를 사용해 색상, 클래스, 클래스 이름을 묶습니다. zip 함수를 사용하면 동일한 개수로 이루어진 자료형을 묶을 수 있는데, 여기에서는 길이 2인 리스트 세 개를 한 묶음으로 묶습니다.

그리고 색상, 클래스 정보, 클래스 이름을 for 문 내부에 적용합니다. scatter 함수를 사용해 데이터 한 건을 하나의 점으로 표현하는데, 속성 x의 클래스 y가 i, 즉, 클래스가 0인지 1인지 확인한 후, x의 0번 속성을 x축 값, 1번 속성을 y축 값으로 지정합니다.

그리고 해당 클래스 인덱스에 해당하는 색상으로 데이터를 표시하고, 해당 라벨을 클래스 이름으로 설정합니다. legend 함수를 사용해 도표 설명 즉, 클래스 색상과 클래스 이름 정보를 표시하고, xlabel과 ylabel 함수를 사용해 각 축의 이름 mean radius와 mean texture를 표시합니다. 마지막으로 이 산포도를 보기 위해 show 함수를 사용합니다. 이렇게 구성해 그린 결과를 확인하면 다음 그림과 같습니다. 그림을 살펴보면 두 속성에 따라 그린 산포도 상에서 두 클래스 악성 malignant와 양성 benign 데이터가 분리하기 힘들게 섞여 있음을 알 수 있습니다. 그림 12는 준비한 원형 데이터를 산포도로 시각화하는 내용입니다. 그림 13은 원형 데이터의 시각화한 결과입니다.

```
plt.figure(figsize=(10, 10))
colors = ['red', 'blue']

for color, i, target_name in zip(colors, [0, 1], target_names):
    plt.scatter(x[y == i, 0], x[y == i, 1], color=color, label=target_name)

plt.legend()
plt.xlabel('Mean Radius')
plt.ylabel('Mean Texture')
plt.show()
```

그림 12 원형 데이터의 산포도 시각화

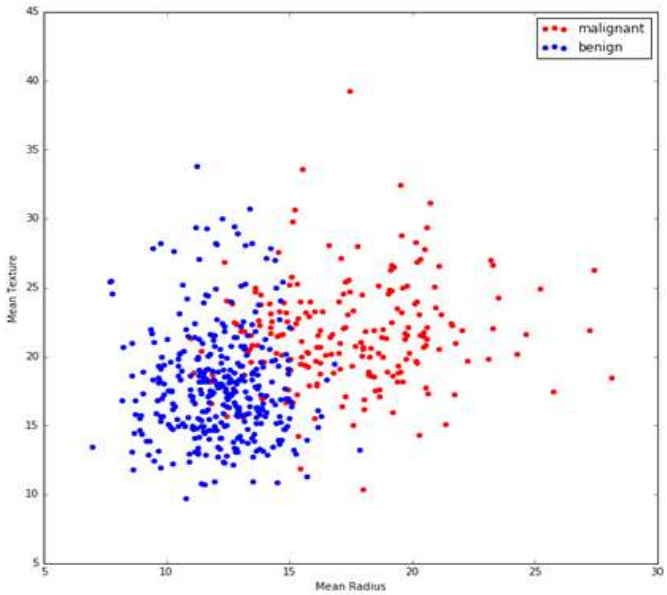

그림 13 원형 데이터의 시각화 결과

이렇게 기본적으로 주어진 속성에 따라 분리하기 어려운 데이터 셋에 차원 축소 방법을 적용해 그 차이를 확인해 보도록 하겠습니다. 그러면 먼저 PCA를 적용해 보도록 하겠습니다.

4.3 PCA

4.3.1 데이터 셋 준비

사용할 데이터 셋을 준비하겠습니다. 데이터의 속성들을 변수 x에 저장하는데, 앞에서 한 방식과는 다르게 30개의 속성 모두를 변수 x에 저장합니다. 우리는 PCA를 사용해서 30차원의 데이터를 축소하고 사용할 계획입니다.

다음으로 유방암 악성과 양성 클래스 정보를 target 함수를 사용해 변수 y에 저장하고, 클래스 이름을 변수 target names에 저장합니다. 이제 데이터 셋

에 PCA를 적용해 보도록 하겠습니다. 그림 14는 PCA에 사용할 데이터를 준비하는 내용입니다.

```
x = data.data
y = data.target
target_names = data.target_names
```

그림 14 PCA에 적용할 데이터 셋 준비

우선 PCA 라이브러리를 사용해 주성분 분석을 하기 위한 모듈 PCA를 생성합니다. 이때 주성분을 두 개 추출하기 위해 n_component를 2로 설정합니다. 이때 주성분을 두 개를 추출하는 이유는 2차원 공간에 산포도를 그려보고 비교하기 위해서입니다. 생성된 모듈을 fit 함수를 사용해 속성 데이터 x에 대해 훈련시킵니다.

4.3.2 데이터 차원 축소

이 훈련된 모델에 transform 함수를 사용해 데이터 x에 대한 차원 축소를 진행합니다. 그리고 그 결과를 변수 x_p에 저장합니다. 그리고 우리는 훈련된 PCA 모델의 두 주성분에 대한 분산을 확인할 수 있습니다. explained variance ratio 함수를 사용하면 되는데요. 그 결과 첫 번째 주 성분의 분산은 0.98, 두 번째 주 성분의 분산은 0.01임을 알 수 있습니다. 그림 15는 PCA 생성 및 훈련한 모델에 차원축소를 진행하고 주성분 분산을 출력하는 내용입니다.

```
pca = PCA(n_components=2)
x_p = pca.fit(x).transform(x)
print('가장 큰 주성분 두 개에 대한 분산: %s' % str(pca.explained_variance_ratio_))
```

```
가장 큰 주성분 두 개에 대한 분산: [ 0.98204467  0.01617649]
```

그림 15 PCA 모델에 차원축소 진행한 주성분 분산 출력

4.3.3 산포도 표현

이렇게 PCA 분석을 마쳤는데요, 그 결과를 시각화해 차이를 확인해 보아야겠죠. 방법은 앞 원본 데이터에 적용한 방법과 같습니다. 다만 산포도에 표시할 데이터 셋이 x가 아닌 PCA 결과로 추출한 주성분 두 개인 x_p라는 것이 차이점입니다. 그리고 주 성분 두 개에 따른 데이터의 산포도 이므로 그래프 정보도 수정을 합니다. x축 이름은 principal component 1로, y축 이름은 principal component 2로 표시합니다. 그 결과를 시각화 하면 다음 그림과

같습니다.

30차원의 데이터를 2차원으로 축소해 산포도를 그려본 결과, 두 개의 클래스가 원본과는 다르게 분리되어 있음을 확인할 수 있습니다. 양성 클래스 데이터의 경우 한곳에 뭉쳐 분포하고 있고, 악성 클래스 데이터의 경우 상반되게 옆에 넓게 분포하고 있습니다.

비록 일부 겹쳐져 있긴 하지만, 원본에 비해 두 클래스의 분포가 잘 분리되어 있음을 알 수 있습니다. 그림 16은 PCA에서 추출된 주성분을 산포도로 시각화하는 내용입니다. 그림 17은 주성분을 산포도로 시각화하는 내용입니다.

```
plt.figure(figsize=(10, 10))
colors = ['red', 'blue']

for color, i, target_name in zip(colors, [0, 1], target_names):
    plt.scatter(x_p[y == i, 0], x_p[y == i, 1], color=color, label=target_name)

plt.legend()
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.show()
```

그림 16 PCA에서 추출된 주성분의 산포도 시각화

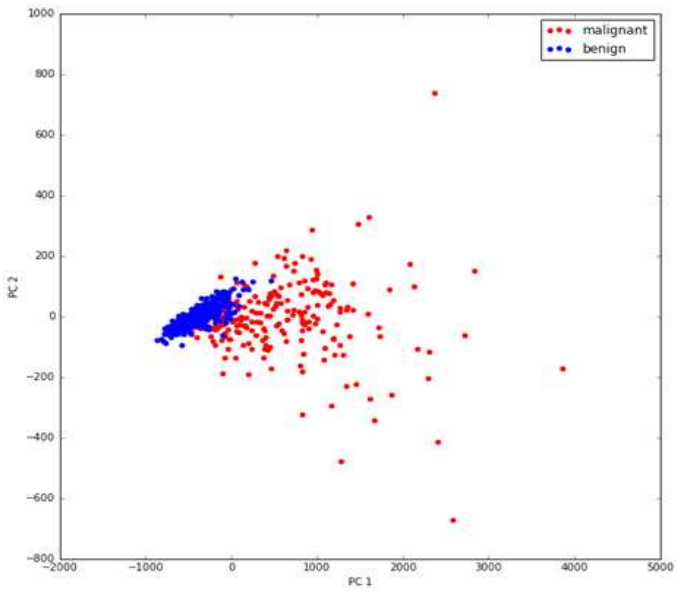

그림 17 추출된 주성분의 시각화 결과

그러면 이제 두 번째 차원 축소 방법인 LDA를 실습해 보겠습니다. 이번에도 마찬가지로 사용할 데이터 셋을 준비해야겠죠.

4.4 LDA

4.4.1 데이터 셋 준비

PCA의 데이터 셋과 마찬가지로 30개 속성 모두를 사용하며 변수 x에 저장하고, 클래스 정보는 변수 y에, 클래스 이름은 변수 target names에 저장합니다. 그림 18은 LDA에 사용할 데이터를 준비하는 내용입니다.

```
x = data.data
y = data.target
target_names = data.target_names
```

그림 18 LDA에 적용할 데이터 셋 준비

이제 데이터 셋에 LDA를 적용해 보겠습니다. 우선 LDA를 사용하기 위해 라이브러리 linear discriminant analysis를 사용해 선형 판별 분석을 위한 모듈 lda를 생성합니다. 이때 고유값을 사용해 클래스를 구분하는 벡터를 구하기 위해 solve를 eigen으로 설정합니다. 그리고 앞과 마찬가지로 2 차원으로 축소하기 위해 n_components를 2로 설정합니다.

4.4.2 데이터 차원 축소

이렇게 생성한 모듈 lda를 fit 함수를 사용해 속성 데이터 x와 클래스 데이터 y에 대한 학습을 진행합니다. PCA와는 다르게 LDA는 클래스 정보를 가지고 진행하기 때문에 fit 함수에 클래스 데이터 y를 함께 적용합니다. 훈련된 모델 lda를 사용하기 위해 transform 함수를 사용하는데, x, 속성 데이터에 대해 차원 축소를 진행합니다. 이 차원축소 결과, 즉 도출된 주성분 2개를 변수 x_l에 저장합니다. 그림 19는 LDA 생성 및 학습된 모델에 차원축소를 진행하고 도출된 축소결과를 저장하는 내용입니다.

```
lda = LinearDiscriminantAnalysis(solver='eigen', n_components=2)
x_l = lda.fit(x, y).transform(x)
```

그림 19 LDA 모델에 차원축소 진행한 주성분 저장

4.4.3 산포도 표현

이제 LDA 적용 결과를 시각화 해보도록 하겠습니다. 여기도 앞과 같은 방법으로 산포도를 그립니다. 다만 차이점은 산포도에 표시할 데이터가 LDA 분석 결과인 x_l 이라는 것입니다. 그리고 각 축의 이름도 LD1, LD2로 바꿔 표시합니다. 그 결과 그려진 산포도를 살펴보면 다음 그림과 같습니다. 그림 20은 LDA에서 추출된 주성분을 산포도로 시각화하는 내용입니다. 그림 21은 주성분을 산포도로 시각화하는 내용입니다.

```
plt.figure(figsize=(10, 10))
colors = ['red', 'blue']

for color, i, target_name in zip(colors, [0, 1], target_names):
    plt.scatter(x_l[y == i, 0], x_l[y == i, 1], color=color, label=target_name)

plt.legend()
plt.xlabel('LD 1')
plt.ylabel('LD 2')
plt.show()
```

그림 20 PCA에서 추출된 주성분의 산포도 시각화

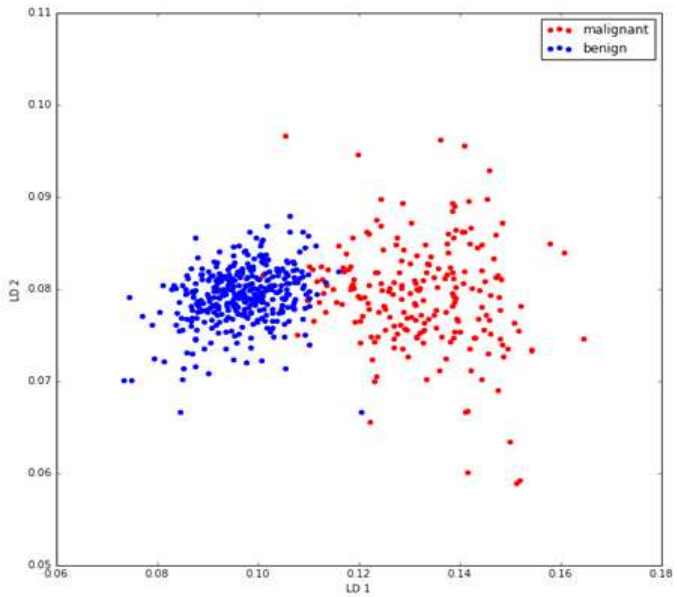

그림 21 추출된 주성분의 시각화 결과

LD1축에 데이터를 사상시킨다고 가정해 볼까요. 그러면 두 클래스가 각각 적절하게 분리됨을 알 수 있습니다.

4.5 원본, PCA, LDA 시각화 결과 비교

이렇게 원본 데이터와 두 개의 차원축소 방법, PCA, LDA를 실습해 보고, 산포도로 비교를 해 보았습니다. 다음 세 개의 그림을 보면 그 차이를 더욱

확실히 느낄 수 있습니다. 세 그림은 차례대로 원본, PCA, LDA 결과입니다. 그림 22는 원본 데이터의 산포도, 그림 23은 PCA 차원축소의 산포도, 그림 24는 LDA 차원축소의 산포도입니다.

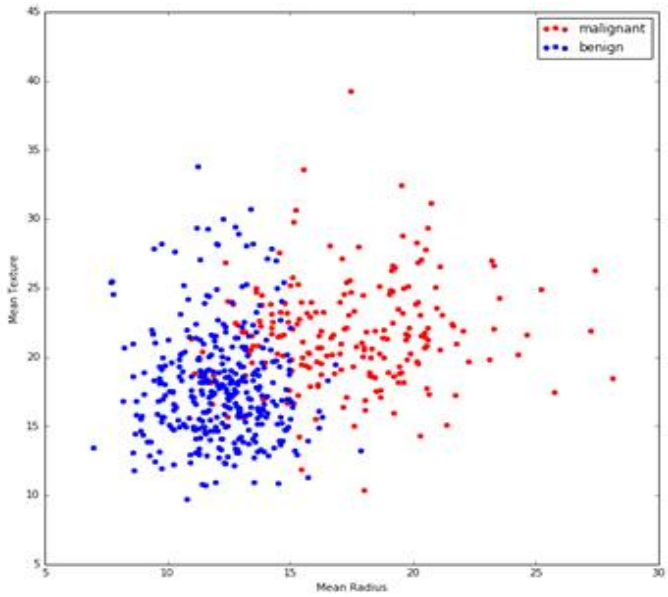

그림 22 원본 데이터의 산포도 시각화 결과

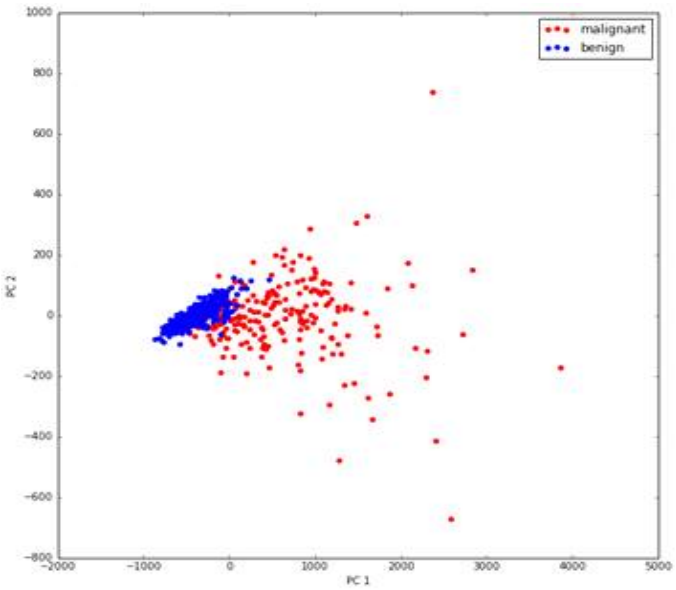

그림 23 PCA 차원축소의 산포도 시각화 결과

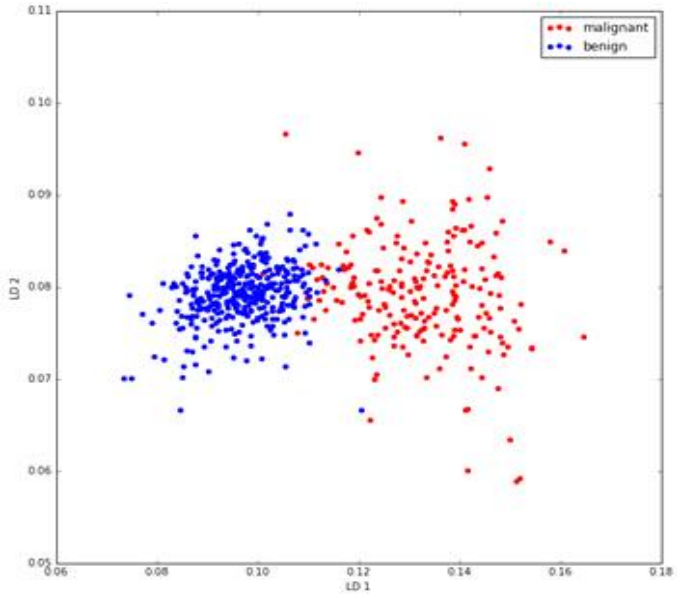

그림 24 LDA 차원축소의 산포도 시각화 결과

30개의 속성 즉, 30차원의 데이터에 PCA, LDA 를 적용해 2차원으로 축소한 결과입니다. 원본 데이터는 두 클래스의 데이터가 분리하기 힘들게 서로 섞여서 분포하고 있는데, 반면에 차원 축소 결과인 두 그림은 차이가 보입니다. PCA, LDA 결과 모두 두 클래스가 눈으로 확인 할 수 있을 만큼 적절하게 분리되어 있습니다.

이와 같이 데이터가 가지고 있는 기본 속성으로 구분하기 어려운 높은 차원의 데이터 셋에서 주성분을 추출하면 데이터를 표현하는데 훨씬 효과적이라는 것을 확인할 수 있었습니다. 이와 같이 데이터 셋의 차원이 클 경우 발생하는 분석의 어려움을 차원 축소를 통해 극복할 수 있습니다.