

11강. 차원(Dimension) 축소

- 인공지능에서의 차원의 개념
- 차원의 저주 문제
- PCA(주성분분석)
- LLE / MDS / LDA

■ 비지도학습의 종류

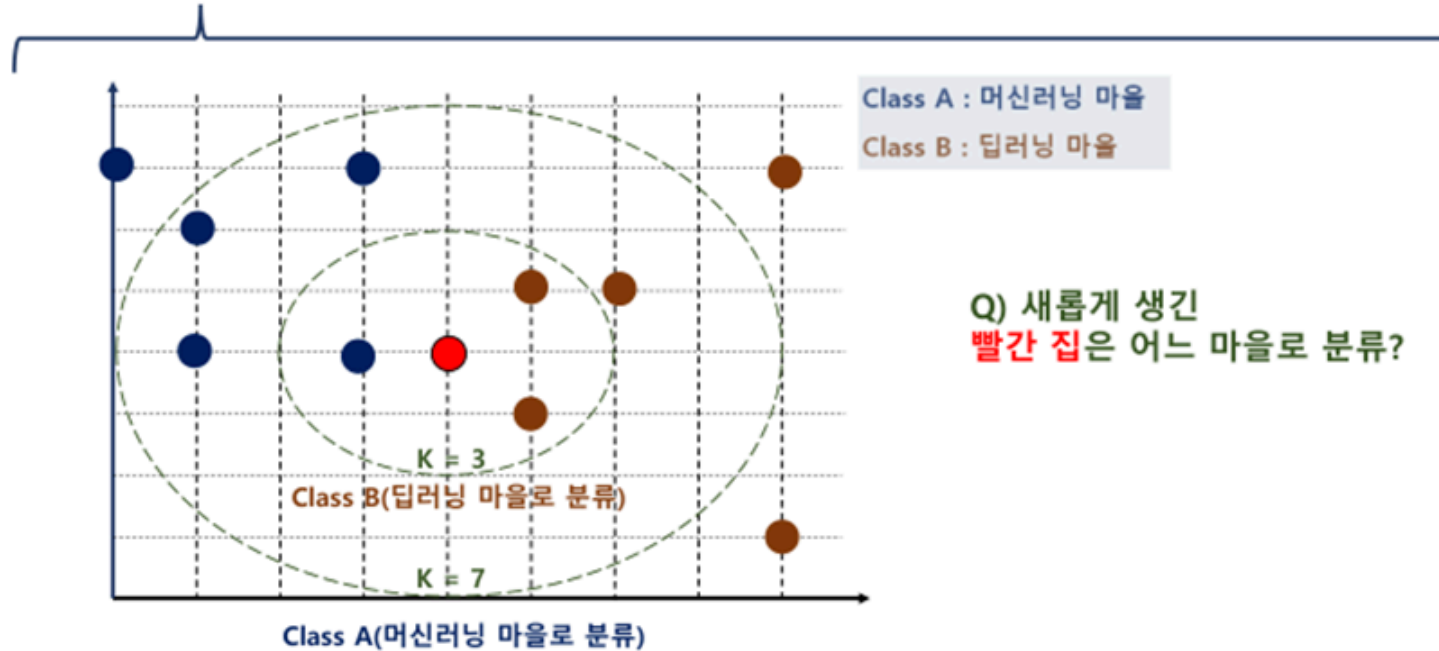


■ 차원(Dimension)의 개념

공간으로 보는 차원

K-Nearest Neighbor

2개의 정보로 대상을 표현 ⇨ 2차원



차원(Dimension)의 개념

벡터로 보는 차원

One-Hot Encoding

3개의 정보로 대상을 표현 \Rightarrow 3차원



■ 차원(Dimension)의 개념

특성으로 보는 차원

회귀 분석

6개의 정보로 대상을 표현 ➡ 6차원

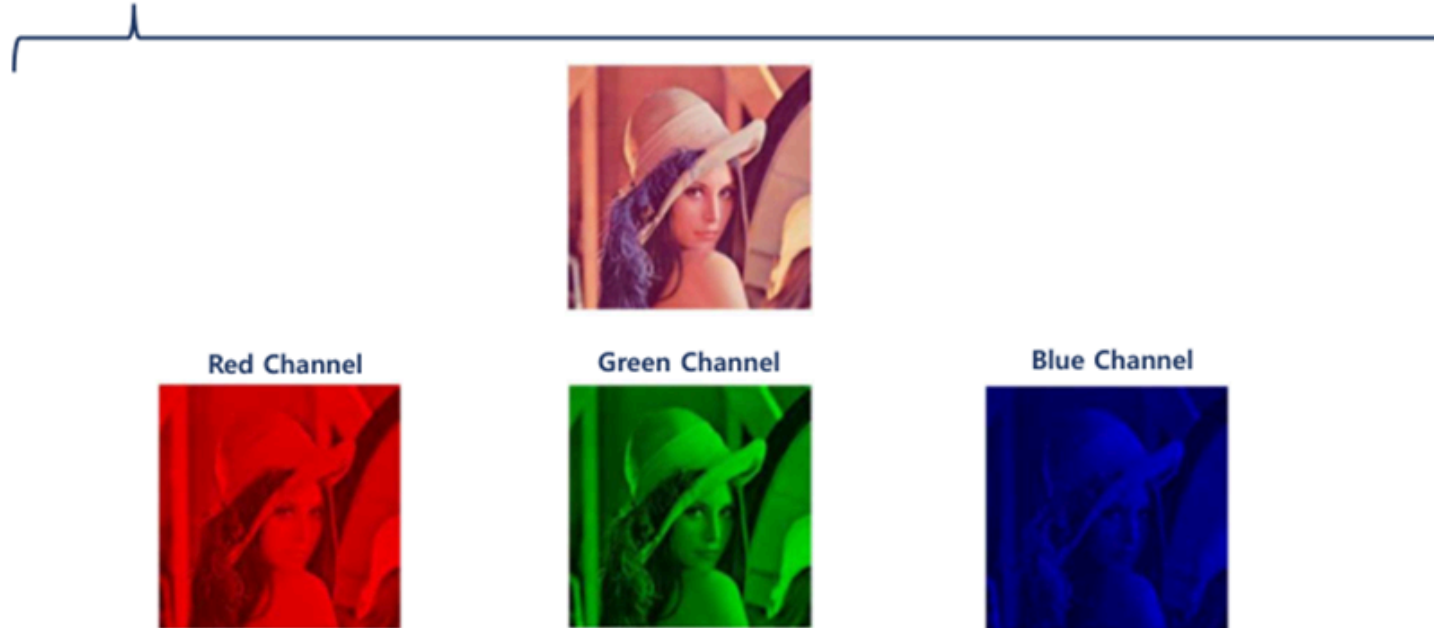


■ 차원(Dimension)의 개념

다채널 이미지에서 보는 차원

Color 이미지

3개의 정보로 대상을 표현 ➡ 3차원



■ 차원(Dimension)의 개념

일반적으로 생각하는 차원

공간의 정보를 결정하기 위한 축의 수

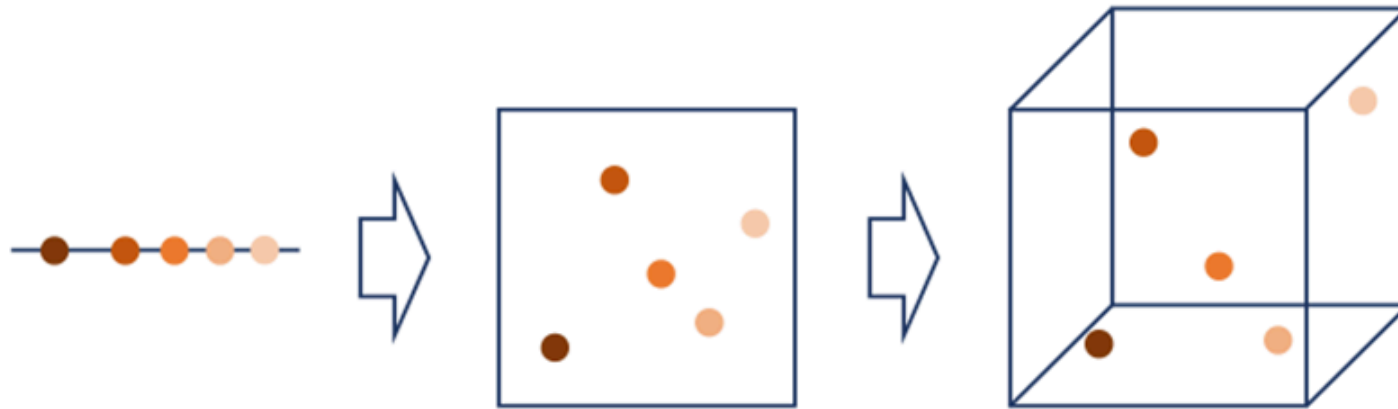
인공지능에서의 차원

대상을 특정하기 위한 정보의 가짓수

■ 차원의 저주 문제

데이터 학습 시 차원이 높아질수록 알고리즘의 성능이 저하되는 현상

수집된 데이터의 수 대비 분석해야 할 특성의 수가 많아질수록 의미 없는 정보가 많아짐



해결방안

밀도가 높아지도록 충분한 데이터를 수집

차원이 높을 수록 필요한 데이터 수가 기하급수적으로 늘어남

차원을 축소하여 특성의 수를 줄임

■ 차원(Dimension) 축소 기법

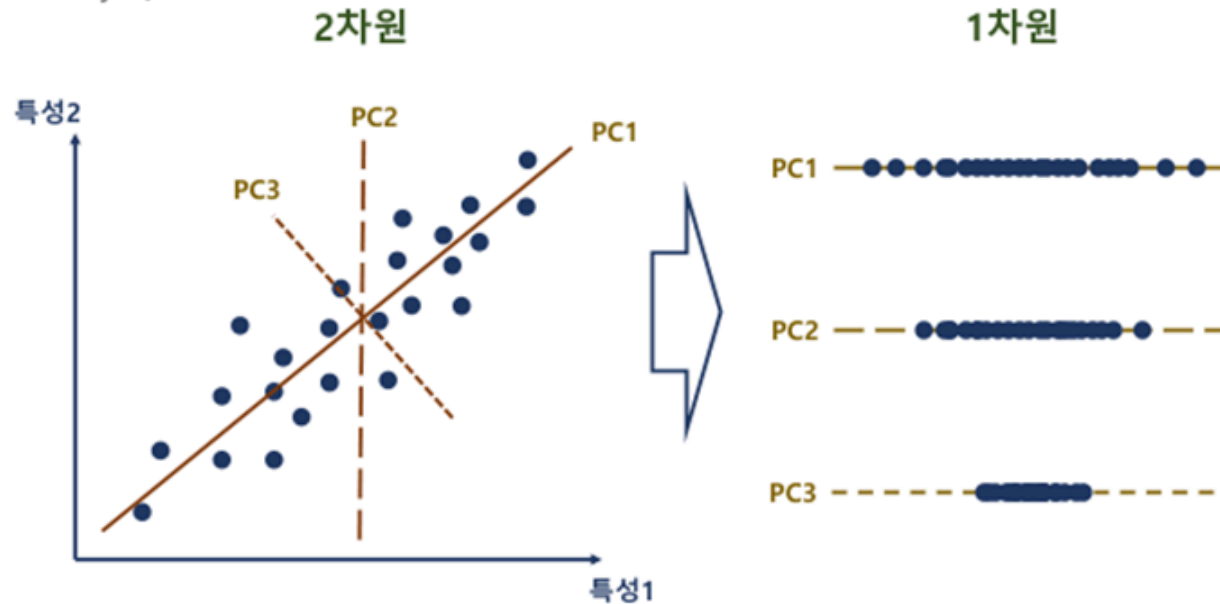
PCA 데이터 분포의 분산이 최대가 되는 정보로 차원을 축소시키는 방법

(Principal Component Analysis)

LLE

MDS

LDA



PC1이 가장 정보가 적게 손실되며 축소되는 주성분

■ 차원(Dimension) 축소 기법

PCA 데이터 분포의 분산이 최대가 되는 정보로 차원을 축소시키는 방법

(Principal Component Analysis)

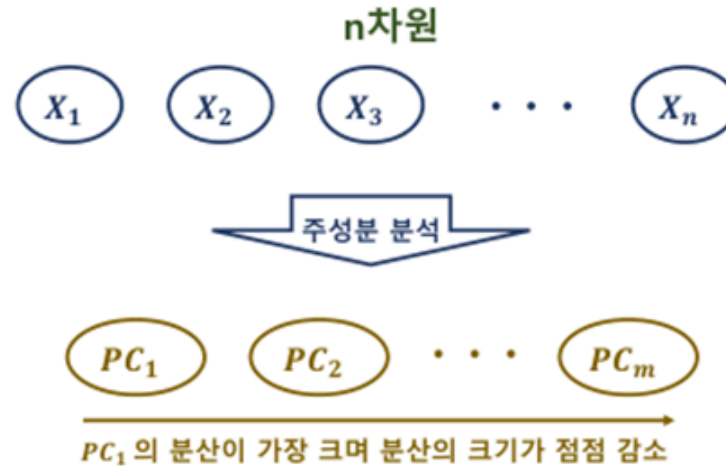
몇 개의 PC를 선택하는 것이 좋은가?

= 몇 차원으로 축소하는 것이 좋은가?

LLE

MDS

LDA



차원(Dimension) 축소 기법

PCA 데이터 분포의 분산이 최대가 되는 정보로 차원을 축소시키는 방법

(Principal Component Analysis)

몇 개의 PC를 선택하는 것이 좋은가?

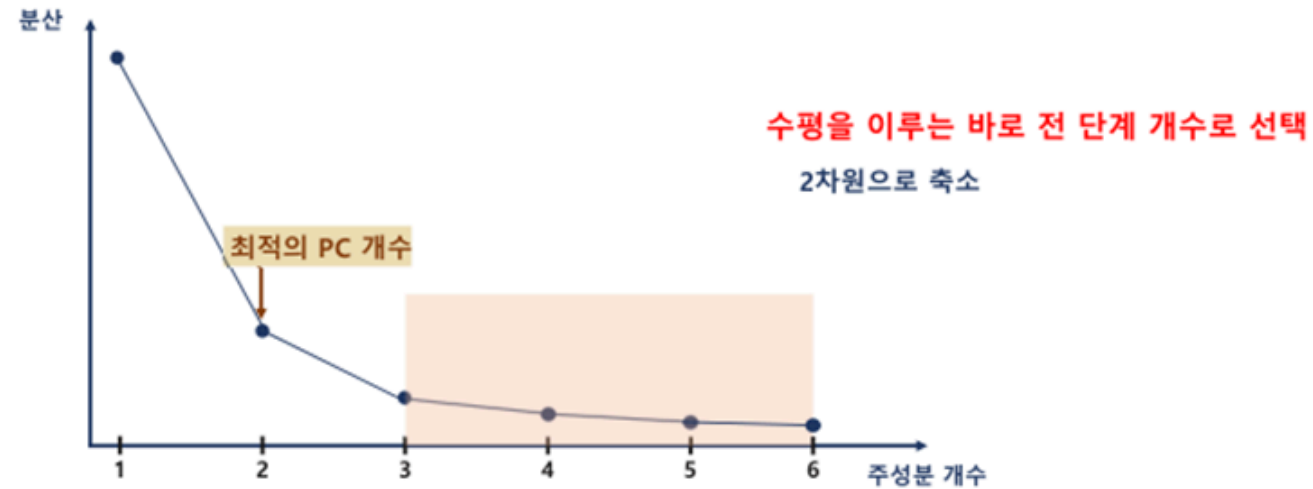
스크리플롯(ScreenPlot)

최적의 PC 개수를 찾기 위한 주성분 개수와 분산을 비교한 그래프

LLE

MDS

LDA



■ 차원(Dimension) 축소 기법

PCA 데이터 분포의 분산이 최대가 되는 정보로 차원을 축소시키는 방법

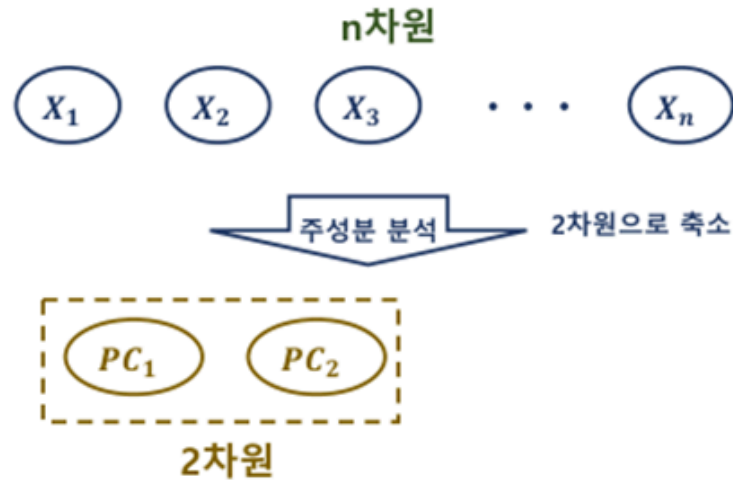
(Principal Component Analysis)

몇 개의 PC를 선택하는 것이 좋은가?

LLE

MDS

LDA



■ 차원(Dimension) 축소 기법

PCA 데이터 분포의 분산이 최대가 되는 정보로 차원을 축소시키는 방법

(Principal Component Analysis)

LLE 데이터들 간의 가까운 이웃에 얼마나 선형적으로 연관되어 있는지 측정하여 저차원의 임베딩 좌표계에 맵핑

(Locality Linear Embedding)

MDS 데이터간의 근접성을 시각화하여 데이터간 거리를 보존하면서 차원을 축소하는 기법

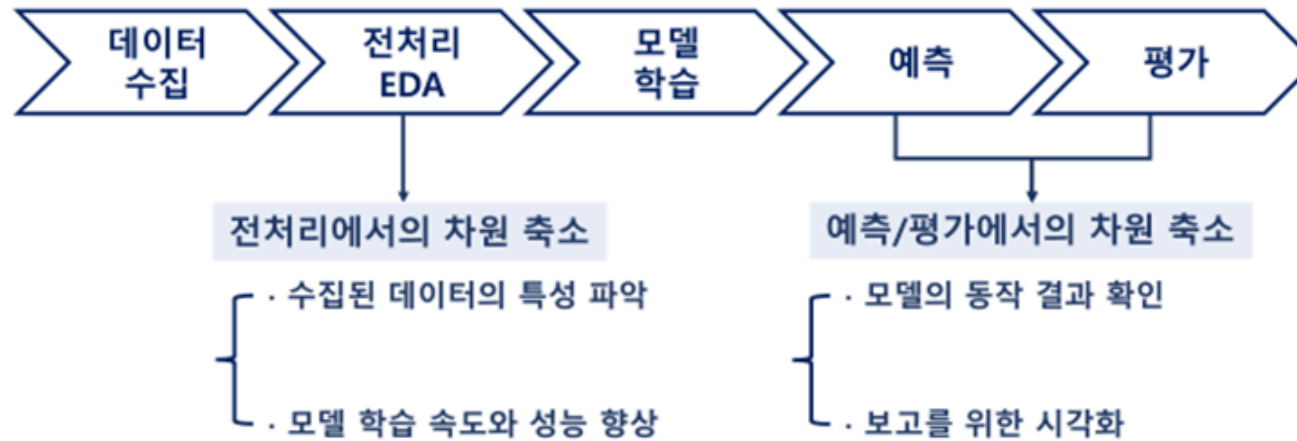
(Multi-Dimensional Scaling)

LDA 개별 클래스를 최대한으로 분리할 수 있는 Hyper Plane을 찾는 방식으로 분류 알고리즘에서도 사용이 가능

(Linear Discriminant Analysis)

■ 차원(Dimension) 축소 기법

머신러닝 파이프라인

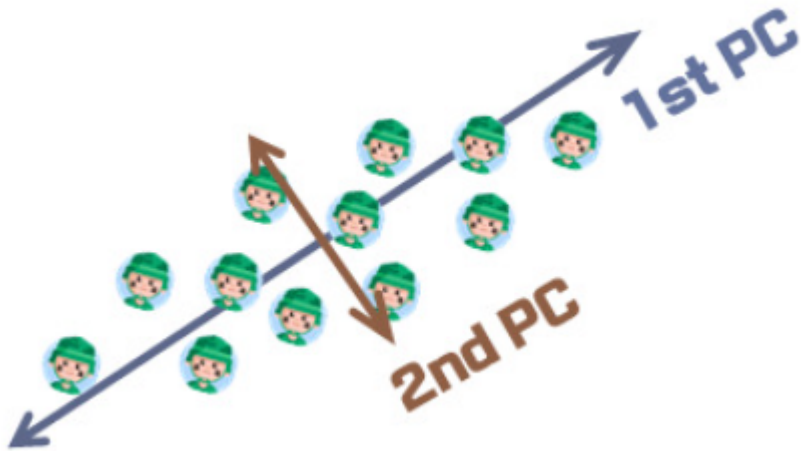


✓ 차원의 개념

✓ 차원 축소(Dimensional Reduction)

✓ 주성분(PC. principal component)

- 1번째 주성분(1st PC. principal component)- 차원 축소에서는 데이터의 분산이 가장 크게 나타나는 방향을 찾는 것
- 2번째 주성분(2nd PC) - 주성분과 수직인 방향 중에서 데이터의 분산을 또 가장 크게 나타내는 방향
 - 이렇게 찾은 주성분 중에서 데이터를 가장 잘 설명하고 있는 성분을 찾으면 차원을 낮출 수 있음
 - B방향에서 분산이 가장 크므로 B방향을 주성분으로 하여 차원을 축소시킬 수 있음
- 차원의 주성분은 데이터를 직접 분석하면서 찾아야 하기 때문에 어떤 차원인지 알려주는 레이블이 별도로 존재하지 않음
 - 직접 데이터를 분석해가며 주성분을 찾아가기 때문에 차원 축소법은 비지도 학습으로 분류됨



✓ 차원의 저주

- 데이터 학습을 위해 차원이 증가하면서 학습데이터 수가 차원의 수보다 적어져 성능이 저하되는 현상

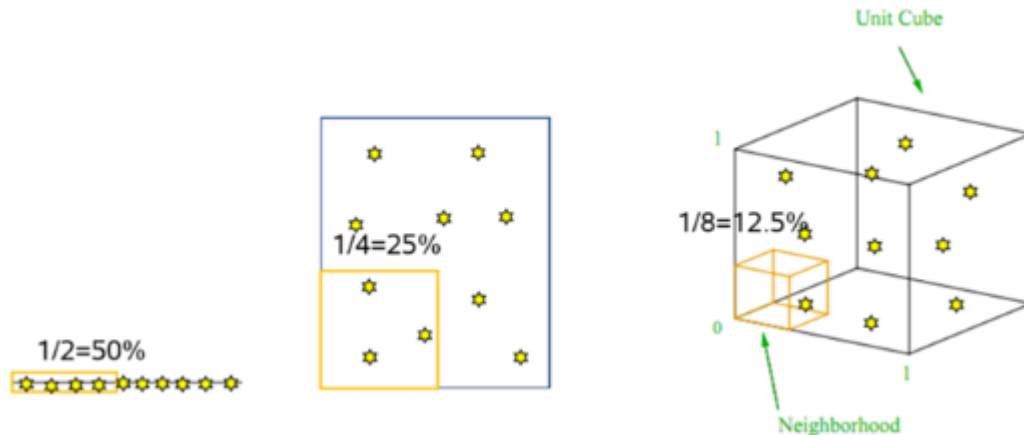
- 차원이 증가할 수록 개별 차원 내 학습할 데이터 수가 적어지는(sparse) 현상 발생
- 해결책 - 차원을 줄이거나(축소시키거나) 데이터를 많이 획득
 - 차원이 증가함에 따라(=변수의 수 증가) 모델의 성능이 안 좋아지는 현상을 의미
 - 무조건 변수의 수가 증가한다고 해서 차원의 저주 문제가 있는 것이 아니라, 관측치 수보다 변수의 수가 많아지면 발생함
 - 예를들어, 관측치 개수는 200개인데, 변수는 7000개인 경우



- ✓ 변수가 1개인, 1차원을 가정할 경우 1차원은 '선' 이기 때문에 선위에 관측치들이 표현될 것 임(선 위에 데이터들이 나란히 있음, 즉, 점들이 빽빽히 들어가 있음)
- ✓ 같은 데이터를, 차원만 2차원('평면')으로 늘릴 경우 점들 사이가 1차원일 때보다, 더 벌어져 있음을 알 수 있음
- ✓ 3차원(축이 3개)으로 차원을 늘릴 경우 점들 사이에 공간이 많이 비었음을 알 수 있음



이처럼, 차원이 증가함에 따라, 빈 공간이 생기는 것을 차원의 저주라고 함



✓ 왜 차원의 저주 현상이 발생할까요?

- 빈 공간이 생겼다는 것은, 컴퓨터 상으로 0으로 채워졌다는 뜻임(이는 정보가 없는 것임)
 - 정보가 적으니, 당연히 모델을 돌릴때, 성능이 저하될 수 밖에 없음!!!
 - 차원의 저주 문제에 치명적인 알고리즘이 KNN임
 - KNN(최근접이웃) 알고리즘은 자신과 가장 가까운 이웃 K개를 보고 라벨(=결과값)을 정하게 되는데, 차원이 커질 수록 내 주변의 이웃이 점점 더 멀어져가게 됨
 - KNN 알고리즘을 쓸 때, 너무 큰 차원일 경우 되도록이면 다른 알고리즘을 쓰거나 차원을 줄이는 방법으로 데이터를 한번 정제해야 함

✓ 데이터 차원 축소 기술(Dimensionality Reduction Techniques)

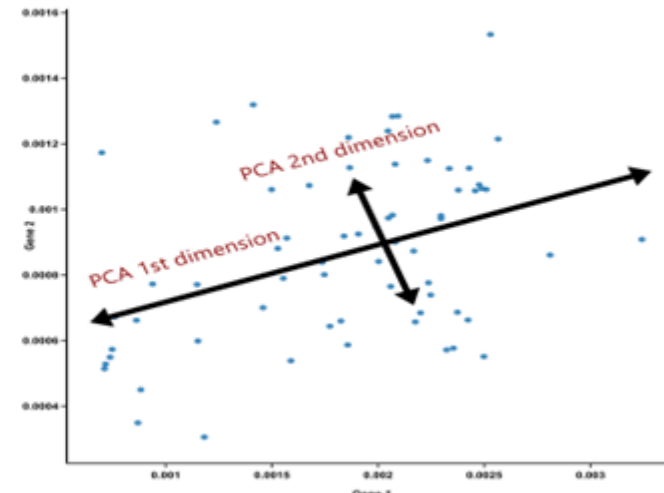
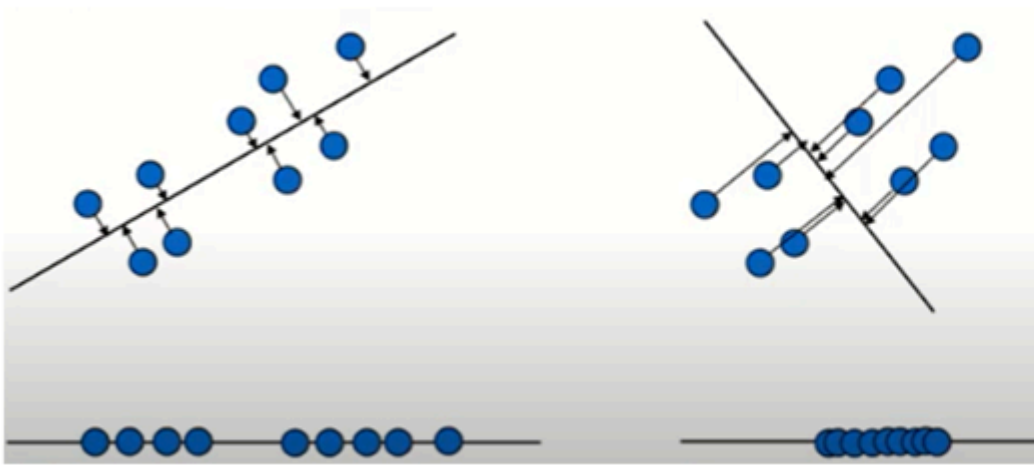
- 고차원의 데이터를 저차원의 데이터로 변환하는 방법으로, 데이터 분석에서 중요한 기술 중 하나
- 축소 기술은 다음과 같은 특징을 가짐
 - 소스 데이터의 의미 있는 속성을 그대로 유지하면서 더 적은 수의 특징을 사용하여 데이터를 표현함
 - 무관하거나 중복된 특징, 노이즈가 있는 데이터를 제거하여 더 적은 수의 변수를 갖는 모델을 생성함
 - 데이터의 차원을 축소하고 중요한 특성을 선택함으로써 모델의 복잡성을 줄이고 계산 효율성을 향상시킴

기술	설명	필요성	예시
PCA (주성분 분석)	고차원 데이터의 주요 특성을 추출하여 데이터를 낮은 차원으로 투영하는 기술	데이터의 차원을 줄여 계산 비용을 절감하고 데이터 시각화와 분석을 용이하게 함	이미지 압축, 얼굴 인식, 유전자 발현 데이터 분석
SVD (특이값 분해)	특이값 분해를 통해 행렬을 세 개의 행렬로 분해하는 기술	대규모 데이터의 차원을 줄이고 행렬의 특성을 추출함	자연어 처리, 이미지 압축, 추천 시스템
t-SNE (t-분포 확률적 임베딩)	고차원 데이터를 저차원 공간에 매핑하여 데이터 간의 유사성을 보존하는 기술	데이터의 시각화와 군집화에 활용되며, 비선형 관계를 파악하는 데 유용함	고차원 데이터의 시각화, 클러스터링, 특징 추출
LDA (잠재 디리클레 할당)	문서 집합 내의 숨겨진 토픽을 추론하고 각 문서의 토픽 분포를 파악하는 확률적 생성 모델	문서 간의 유사성을 파악하고, 토픽을 이해하여 문서의 구조를 분석함	텍스트 마이닝, 문서 분류, 토픽 모델링
NMF (비음수 행렬 분해)	행렬을 두 개 이상의 비음수 행렬로 분해하여 데이터의 특성을 추출하는 기술	데이터의 차원을 줄이고, 중요한 특성을 추출하여 분석함	이미지 처리, 음성 처리, 텍스트 마이닝

✓ PCA(Principal Component Analysis)

- 가장 대표적인 차원 축소 기법
 - 여러 변수 간에 존재하는 상관관계를 이용해 이를 대표하는 주성분을 추출해 차원을 축소하는 기법
 - PCA 기법의 핵심은 데이터를 축에 사영했을 때 가장 높은 분산을 가지는 데이터의 축을 찾아 그 축으로 차원을 축소하는 것인데, 이 축을 주성분이라함
 - 높은 분산을 가지는 축을 찾는 이유는 정보의 손실을 최소화하기 위함

- 사영했을 때 분산이 크다는 것은 원래 데이터의 분포를 잘 설명할 수 있다는 것을 뜻하고 정보의 손실을 최소화할 수 있다는 것을 뜻함



PCA

PCA-1

✓ PCA 예 - 원래 데이터 세트와 축소된 데이터 세트가 어떻게 달라지는지 확인

```

1 #사이킷런에 내장되어 있는 iris 데이터 세트는 4개의 속성 (sepal length, sepal width, petal length, petal width)로 되어 있는데,
2 #이를 2개의 속성으로 차원 축소하여 원래 데이터 세트와 축소된 데이터 세트가 어떻게 달라지는지 확인
3 from sklearn.datasets import load_iris
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7
8 iris=load_iris()
9 columns=['sepal_length','sepal_width','petal_length','petal_width']
10 df_iris=pd.DataFrame(iris.data,columns=columns)
  
```

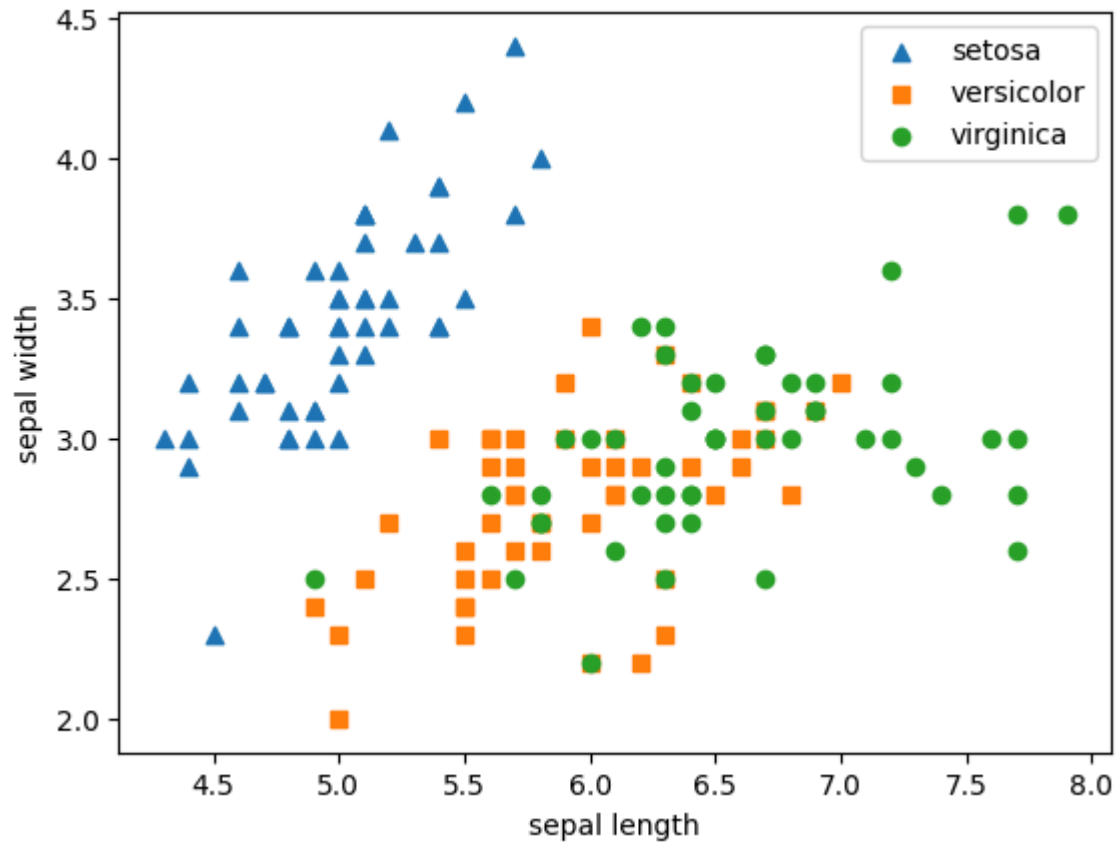
```
11 df_iris['target']=iris.target
12 df_iris.head()
```



	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

다음 단계: [df_iris 변수로 코드 생성](#) [추천 차트 보기](#) [New interactive sheet](#)

```
1 #PCA를 적용하기 전에 개별 속성을 함께 스케일링해야 함
2 #PCA는 여러 속성의 값을 연산해야 하기 때문에 속성의 스케일에 영향을 받음
3 #setosa는 세모, versicolor는 네모, virginica는 동그라미
4 markers=['^','s','o']
5
6 for i, marker in enumerate(markers):
7     x_axis_data=df_iris[df_iris['target']==i]['sepal_length']
8     y_axis_data=df_iris[df_iris['target']==i]['sepal_width']
9     plt.scatter(x_axis_data,y_axis_data,marker=marker,label=iris.target_names[i])
10
11 plt.legend()
12 plt.xlabel('sepal length')
13 plt.ylabel('sepal width')
14 plt.show()
15
16 #원래 데이터는 versicolor와 virginica가 분류가 어려운 것을 확인할 수 있음
```

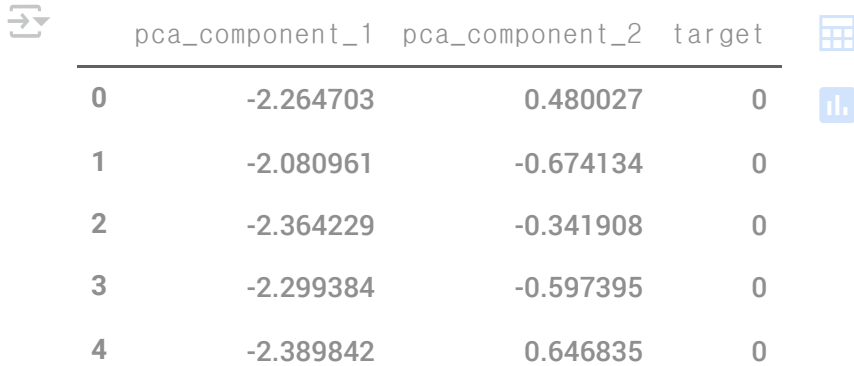


```
1 #PCA를 통해 속성을 2개로 축소해보고 어떻게 변화되는지 확인
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.decomposition import PCA
4
5 scaler=StandardScaler()
6 iris_scaled=scaler.fit_transform(df_iris.iloc[:, :-1])
7
8 #2차원으로 차원 축소
9 pca=PCA(n_components=2)
10 pca.fit(iris_scaled)
11 iris_pca=pca.transform(iris_scaled)
12
```

```

13 pca_columns=['pca_component_1','pca_component_2']
14 df_iris_pca=pd.DataFrame(iris_pca,columns=pca_columns)
15 df_iris_pca['target']=iris.target
16 df_iris_pca.head()

```



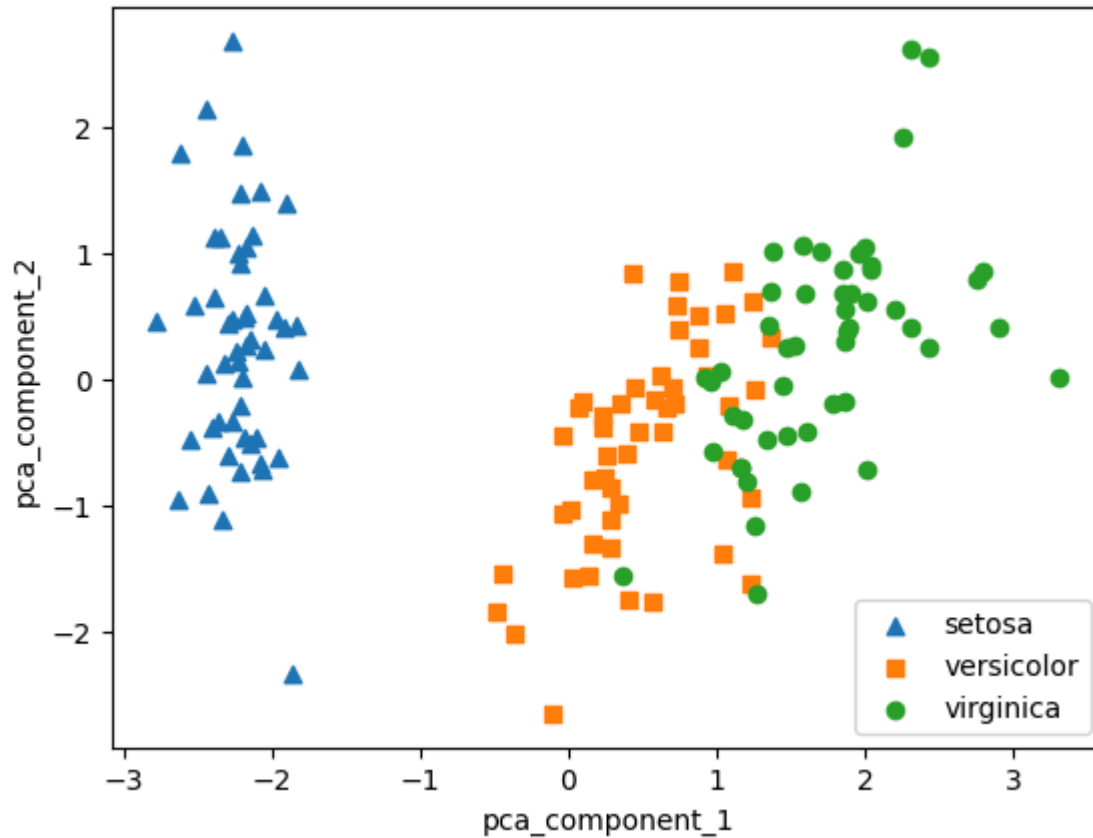
	pca_component_1	pca_component_2	target
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0

다음 단계: [df_iris_pca 변수로 코드 생성](#) [추천 차트 보기](#) [New interactive sheet](#)

```

1 # setosa는 세모, versicolor는 네모, virginica는 동그라미
2 markers=['^','s','o']
3
4 for i, marker in enumerate(markers):
5     x_axis_data=df_iris_pca[df_iris_pca['target']==i]['pca_component_1']
6     y_axis_data=df_iris_pca[df_iris_pca['target']==i]['pca_component_2']
7     plt.scatter(x_axis_data,y_axis_data,marker=marker,label=iris.target_names[i])
8
9 plt.legend()
10 plt.xlabel('pca_component_1')
11 plt.ylabel('pca_component_2')
12 plt.show()
13
14 #versicolor와 virginica가 완벽히 분류되지는 않았지만 원래 데이터에 비해 훨씬 분류가 잘 되어 있는 것을 확인할 수 있음
15 #그래프를 보면 데이터가 pca_component_1 축을 기반으로 서로 겹치는 부분이 별로 존재하지 않게 잘 분류되어 있는 것을 확인할 수 있음
16 #이는 첫 번째 주성분인 pca_component_1이 원본 데이터의 변동성을 잘 반영했기 때문임

```



```

1 #PCA 주성분 별로 원본 데이터의 주성분을 얼마나 반영하는지 알아보자!!!
2 pca.explained_variance_ratio_
3
4 #첫 번째 주성분이 약 73%, 두 번째 주성분이 약 23%로 두 주성분이 원본 데이터의 96%를
5 #설명할 수 있는 것을 확인할 수 있음

```



```
array([0.72962445, 0.22850762])
```

```

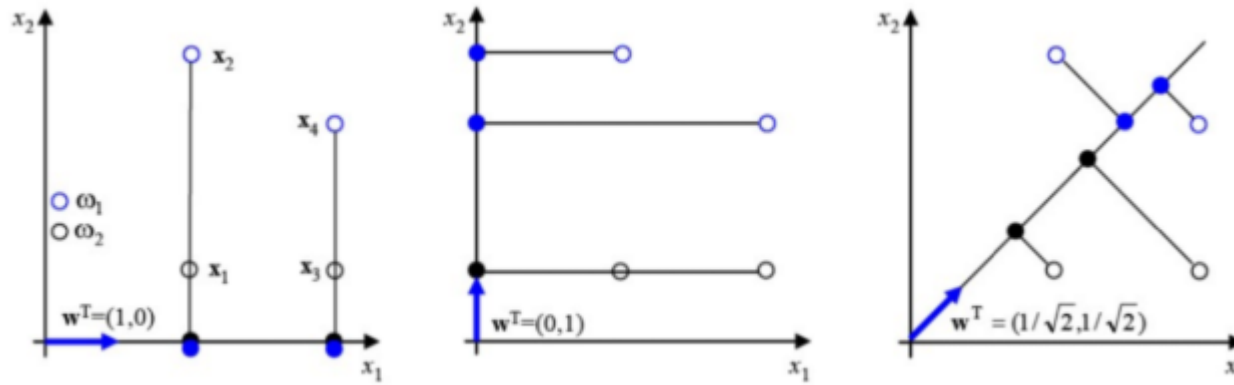
1 #분류 예측 정확도는 PCA 변환 차원 개수에 따라 떨어질 수밖에 없음
2 #모델은 RandomForestClassifier를 이용하고 교차 검증 세트로 정확도를 비교해보자!!!
3
4 from sklearn.ensemble import RandomForestClassifier

```

```
5 from sklearn.model_selection import cross_val_score
6 import numpy as np
7
8 #원본 데이터
9 rf_clf=RandomForestClassifier(random_state=156)
10 scores=cross_val_score(rf_clf,iris.data,iris.target,scoring='accuracy',cv=3)
11 print('원본 데이터 교차 검증 평균 정확도:{0:.4f}'.format(np.mean(scores)))
12
13 #PCA변환 후 데이터
14 pca_X=df_iris_pca[['pca_component_1','pca_component_2']]
15 scores_pca=cross_val_score(rf_clf,pca_X,iris.target,scoring='accuracy',cv=3)
16 print('PCA 변환 후 교차 검증 평균 정확도:{0:.4f}'.format(np.mean(scores_pca)))
17
18 #PCA를 통해 2차원으로 축소한 결과 예측 성능이 8% 하락
19 #하지만 속성 개수가 50% 감소한 것을 고려하면 PCA 변환 후에도 원본 데이터의 특성을
20 #상당 부분 유지하고 있음을 알 수 있음
21
22
23 #하지만, PCA의 주요 한계점으로 최대의 분산의 각 축이 반드시 클래스 간의 구별을
24 #잘하는 좋은 피처를 뽑아준다는 보장이 없다는 점이 있음
```

⇒ 원본 데이터 교차 검증 평균 정확도:0.9600
PCA 변환 후 교차 검증 평균 정확도:0.8800

- 다음 그림을 보면 가장 데이터가 안 겹치게 사영되는 것은 3번째 축이지만 사실 사영했을 때 분류가 제일 잘 되는 것은 2번째 축임
- 따라서 차원을 축소하는 데 있어 클래스 간의 차별성을 최대화할 수 있는 방향으로 수행하는 것이 LDA임



PCA-2

✓ PCA, 주성분의 개수는 어떤 기준으로 설정할까?


- PCA를 통해 종속변수 예측력에 영향을 준다는 내용보다는 독립변수들에 PCA 수행을 해주었을 때 몇 개의 주성분으로 축소되어야 적절한지 살펴보자!!!
- 주성분 개수를 설정(결정)하는 기준에는 크게 3가지가 있음
 - 1) 고윳값(설명 가능한 분산)
 - 2) 누적 기여율
 - 3) Scree plot



R로 구현한 PCA


```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 # 데이터 읽기
6 heptathlon = pd.read_csv("/content/drive/MyDrive/머신러닝&딥러닝/머신러닝-11-차원축소-차원의개념 차원저주 PCA/heptathlon.csv")
7 # heptathlon = pd.read_csv("/Users/DataAnalytics/MultivariateAnalysis/mva/heptathlon.csv")
8 heptathlon.head(3)

```




	rownames	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score	
0	Joyner-Kersey (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51	7291	
1	John (GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12	6897	
2	Behmer (GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20	6858	

다음 단계: [heptathlon 변수로 코드 생성](#) [추천 차트 보기](#) [New interactive sheet](#)

```

1 # 변수이름 확인하기
2 heptathlon.columns

```



```

Index(['rownames', 'hurdles', 'highjump', 'shot', 'run200m', 'longjump',
      'javelin', 'run800m', 'score'],
      dtype='object')

```

```

1 # 기술통계량 구하기 - 소수점 이하 2자리 반올림 표시
2 round(heptathlon.describe(), 2)

```



	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score	
count	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	
mean	13.84	1.78	13.12	24.65	6.15	41.48	136.05	6090.60	
std	0.74	0.08	1.49	0.97	0.47	3.55	8.29	568.47	
min	12.69	1.50	10.00	22.56	4.88	35.68	124.20	4566.00	
25%	13.47	1.77	12.32	23.92	6.05	39.06	132.24	5746.00	

1 # 변환: 변수최댓값 - 변수값

2 heptathlon.hurdles = np.max(heptathlon.hurdles) - heptathlon.hurdles

3 heptathlon.run200m = np.max(heptathlon.run200m) - heptathlon.run200m

4 heptathlon.run800m = np.max(heptathlon.run800m) - heptathlon.run800m

5 heptathlon.head()



	rownames	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score	
0	Joyner-Kersey (USA)	3.73	1.86	15.80	4.05	7.27	45.66	34.92	7291	