선형대수학팀

3팀

김다민
이지원
조성우
김수인
방건우

CONTENTS

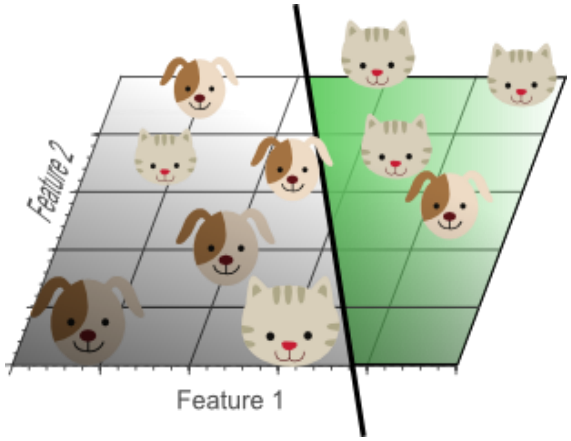
1

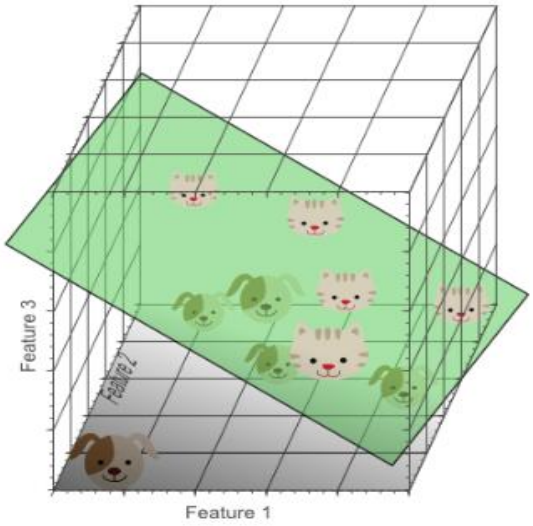
주성분 분석

차원의 저주

차원의 축복(?)

데이터 분석에서 변수의 추가, 즉 **차원**의 증가는 모델의 성능 향상에 도움이 됨





2차원보다 3차원일때의 분류가 쉬움

주성분 분석

차원의 저주

차원의 축복(?)

데이터 분석에서 변수의 추가, 즉 **차원**의 증가는 모델의 성능 향상에 도움이 됨

그렇다면 **사용가능한 모든 변수들**을 통해 모델링하면

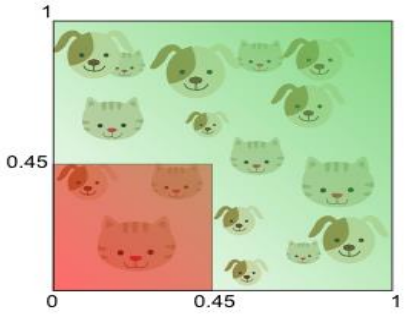
성능이 뛰어날까?

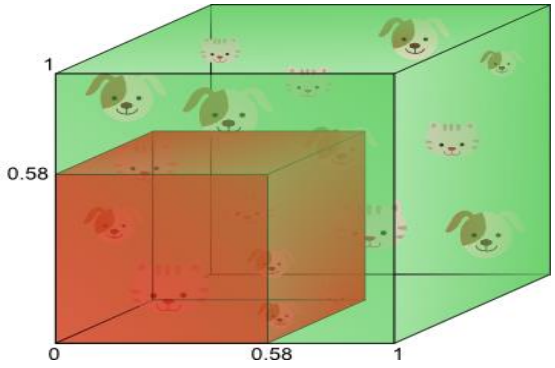
2차원보다 3차원일때의 분류가 쉬움

차원의 저주

차원의 저주 개념

차원이 증가함에 따라 학습 알고리즘이 제대로 작동하지 않는 현상





차원이 증가 ▶ 공간의 크기가 증가 ▶ 데이터간 거리가 멀어짐

적은 데이터로만 이 공간을 표현하는 경우 **모델 성능 저하** & **계산량 증가**

주성분 분석

차원의 저주

차원의 저주를 해결하기 위해서는 차원의 개수를 조절해야 함

⋮

회귀분석의 **변수 선택**과 데이터마이닝의 **변수 필터링** 등

다양한 **차원축소** 기법 존재

본 클린업에서는 선형대수학 기반 다변량 기법인 PCA에 대해 다룸

1 주성분 분석

차원의 저주

차원의 저주를 해결하기 위해서는 차원의 개수를 조절해야 함

⋮

회귀분석의 **변수 선택**과 데이터마이닝의 **변수 필터링** 등

다양한 **차원축소** 기법 존재

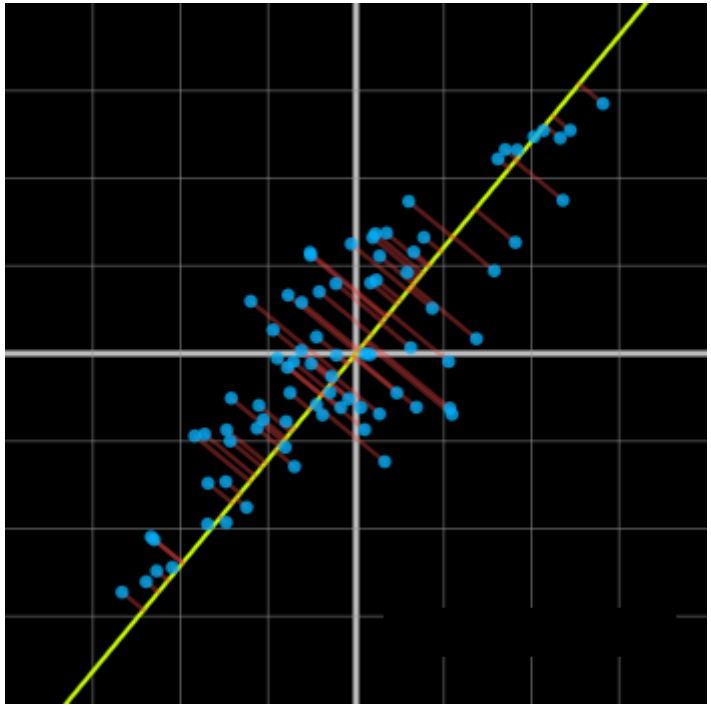
본 클린업에서는 선형대수학 기반 다변량 기법인 PCA에 대해 다룸

주성분 분석 *PCA*

주성분 분석의 개념

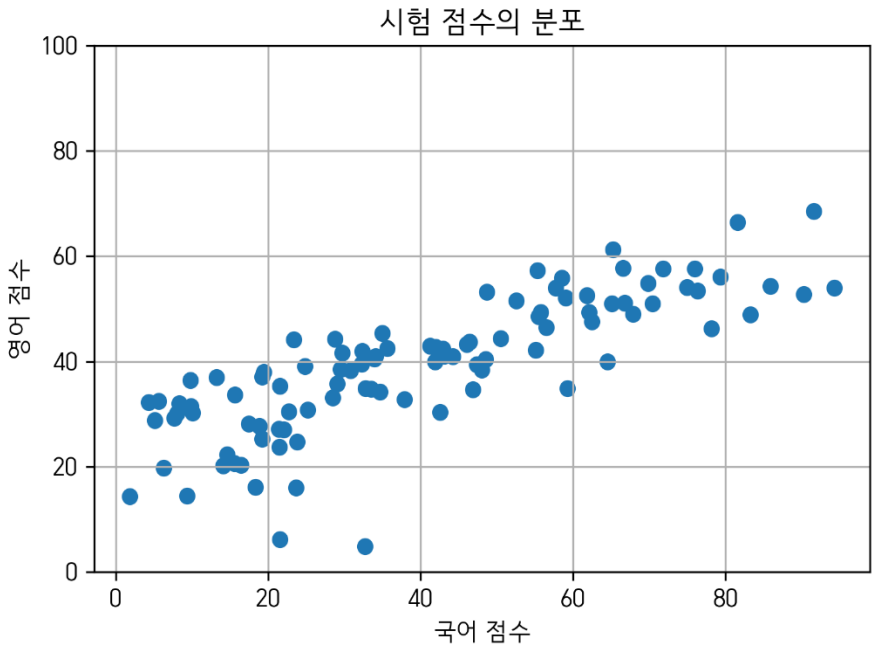
데이터 분석에서 자주 사용되는 **차원 축소** 기법

원 데이터의 **분산**을 최대한 **보존**하는 방향으로 변수 변형



주성분 분석

주성분 분석 *PCA*



X축에는 국어 점수

Y축에는 영어 점수

⋮

국어와 영어의 종합 점수를

만들고 싶다면

어떻게 만들까?

주성분 분석 *PCA*

산술평균 (동등한 가중치)

국어80점, 수학 60점이라면

$$80 \times 0.5 + 60 \times 0.5$$

이를 벡터의 내적으로 표현하면

$$\begin{bmatrix} 80 \\ 60 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

가중평균 (한 과목에 가중치)

국어80점, 수학 60점에
6:4 가중치를 적용하면,

$$80 \times 0.5 + 60 \times 0.5$$

내적으로 표현하면

$$\begin{bmatrix} 80 \\ 60 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

주성분 분석 *PCA*

산술평균 (동등한 가중치)

국어80점, 수학 60점이라면

$80 \times 0.5 + 60 \times 0.5$

이를 벡터의 내적으로 표현하면

$$\begin{bmatrix}80\\60\end{bmatrix} \cdot \begin{bmatrix}0.5\\0.5\end{bmatrix}$$

가중평균 (한 과목에 가중치)

같은 점수에

6:4 가중치를 적용하면,

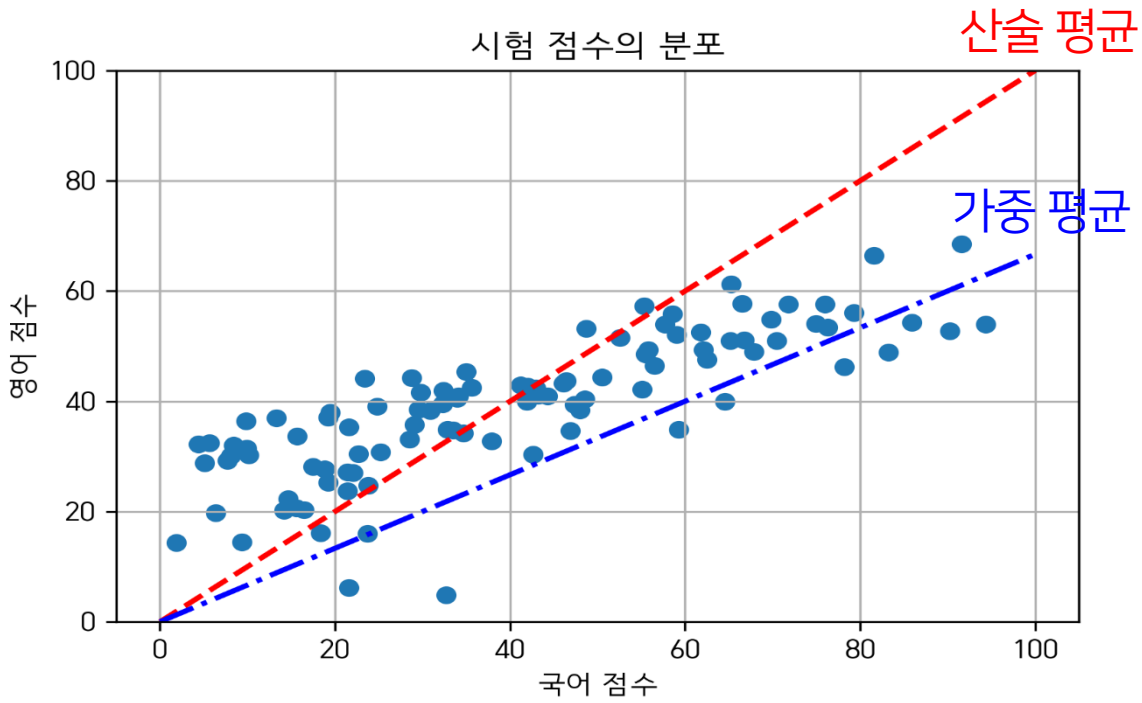
$80 \times 0.6 + 60 \times 0.4$

내적으로 표현하면

$$\begin{bmatrix}80\\60\end{bmatrix} \cdot \begin{bmatrix}0.6\\0.4\end{bmatrix}$$

주성분 분석

주성분 분석 *PCA*

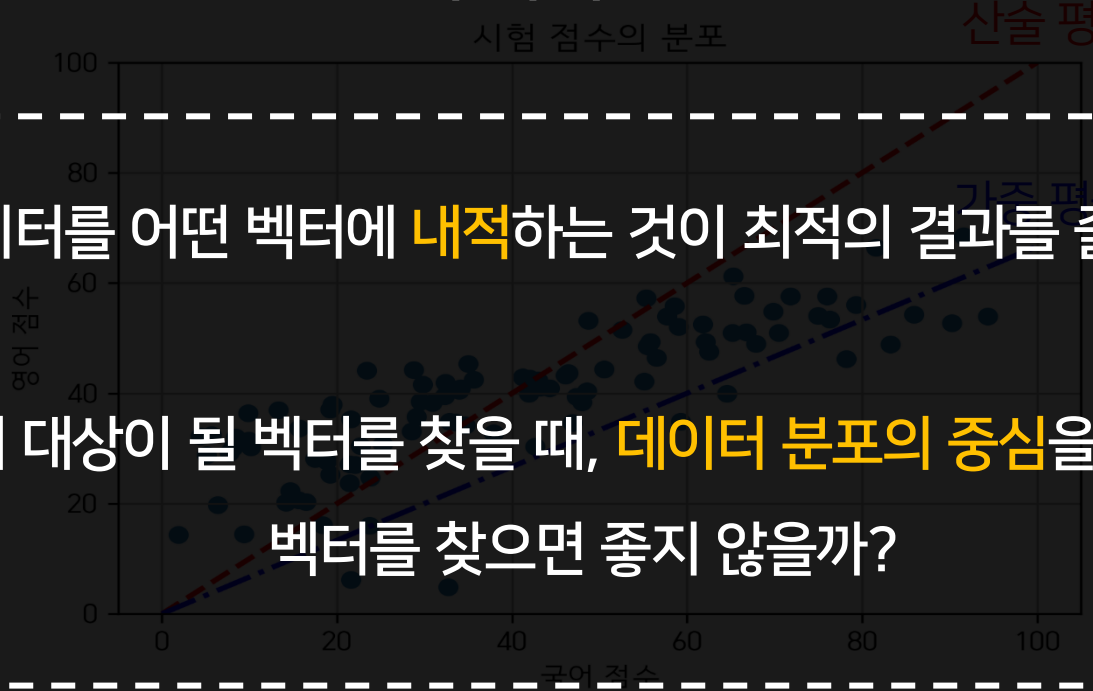


종합 점수를 얻는 방식을 수학적으로는

점수 벡터를 비율 벡터에 **내적(정사영)**하는 문제로 생각

주성분 분석 *PCA*

두가지 고민



① 데이터를 어떤 벡터에 내적하는 것이 최적의 결과를 줄 것인가?

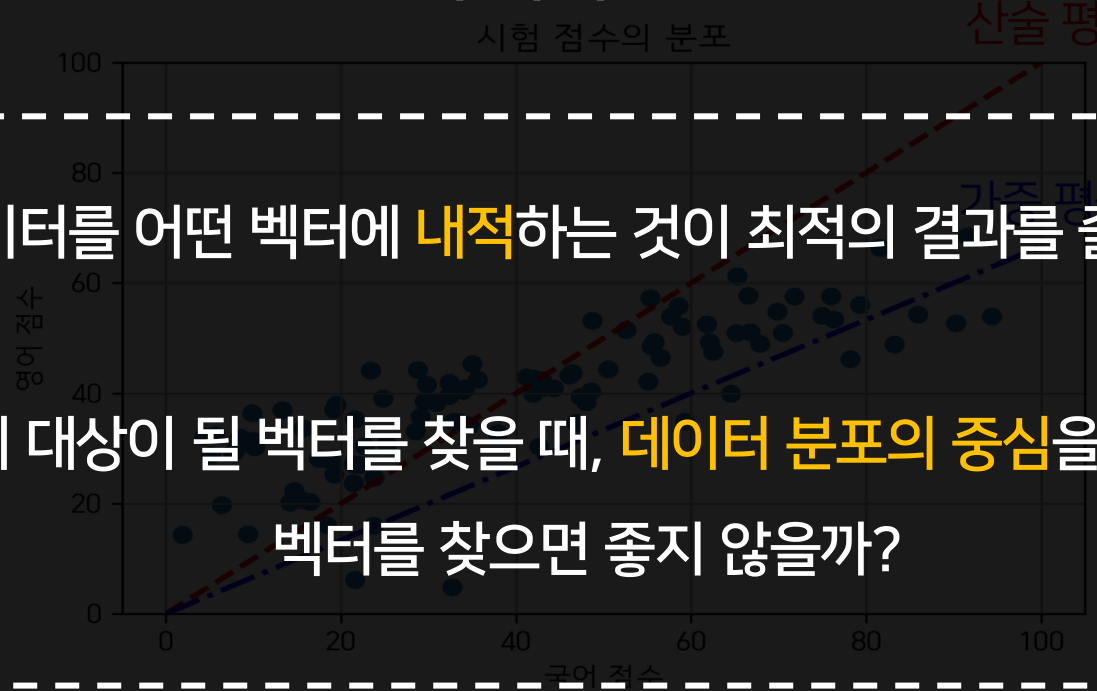
② 내적의 대상이 될 벡터를 찾을 때, 데이터 분포의 중심을 축으로 하는 벡터를 찾으면 좋지 않을까?

종합 점수를 얻는 방식을 수학적으로는

점수 벡터를 비율 벡터에 내적(정사영)하는 문제로 생각

주성분 분석 *PCA*

두가지 고민



① 데이터를 어떤 벡터에 **내적**하는 것이 최적의 결과를 줄 것인가?

② 내적의 대상이 될 벡터를 찾을 때, **데이터 분포의 중심**을 축으로 하는 벡터를 찾으면 좋지 않을까?

➡ **공분산 행렬**로부터 답을 찾을 수 있음

점수 벡터를 비율 벡터에 내적(정사영)하는 문제로 생각

주성분 분석

공분산 행렬

공분산 행렬 *Covariance matrix*

변수들의 **공분산**을 **행렬**로 나타낸 것

$$cov(X,Y) = E[(X-\mu_x)(Y-\mu_y)^T = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{m1} & \cdots & \sigma_{mn} \end{bmatrix}$$

주성분 분석

공분산 행렬

공분산 행렬의 수식적 의미

n개의 관찰값과 p개의 변수가 있는 데이터 행렬 X를 생각

행렬 X의 각 열들은 **정규화**된 상태여야 함

$$X = \begin{bmatrix} | & | & \cdots & | \\ X_1 & X_2 & \cdots & X_p \\ | & | & \cdots & | \end{bmatrix} \in R^{n \times p}$$

왜 정규화를 해야 할까?

① **데이터의 분포**가 원점에서 뻗어나가는 벡터 공간 형태를 만들어 데이터의 **추세** 파악

② 변수 간 조건을 **동등**하게 해놓고 **정보량**을 **비교**하는 것이 목적

공분산 행렬

$$X^T X = \begin{bmatrix} - & X_1 & - \\ - & X_2 & - \\ \cdots & \cdots & \cdots \\ - & X_p & - \end{bmatrix} \begin{bmatrix} | & | & \cdots & | \\ X_1 & X_2 & \cdots & X_p \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} X_1^T X_1 & X_1^T X_2 & \cdots & X_1^T X_p \\ X_2^T X_1 & X_2^T X_2 & \cdots & X_2^T X_p \\ \vdots & \vdots & \ddots & \vdots \\ X_p^T X_1 & X_p^T X_2 & \cdots & X_p^T X_p \end{bmatrix}$$

각 열벡터는 편차벡터이므로 **내적**은 **편차 제곱의 합**

⋮

데이터의 크기(n)가 많을 수록 편차 제곱의 합은 커지므로

위 행렬을 n으로 나누면 **공분산행렬** 완성

$$\boldsymbol{\Sigma} = \frac{1}{n} X^T X$$

주성분 분석

공분산 행렬

공분산 행렬의 기하학적 의미

공분산 행렬은 **데이터의 구조**를 설명하며 **변수 간 관계** 파악에 도움

행렬은 선형변환

공분산 행렬을 통해 데이터를 변환하면 분산과 공분산만큼 공간이 변화

x축 방향으로 퍼진 정도

x, y축 방향으로
함께 퍼진 정도

$$\begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$$

x, y축 방향으로
함께 퍼진 정도

y축 방향으로 퍼진 정도

주성분 분석

공분산 행렬

공분산 행렬의 기하학적 의미

공분산 행렬은 **데이터의 구조**를 설명하며 **변수 간 관계** 파악에 도움

행렬은 **선형변환**

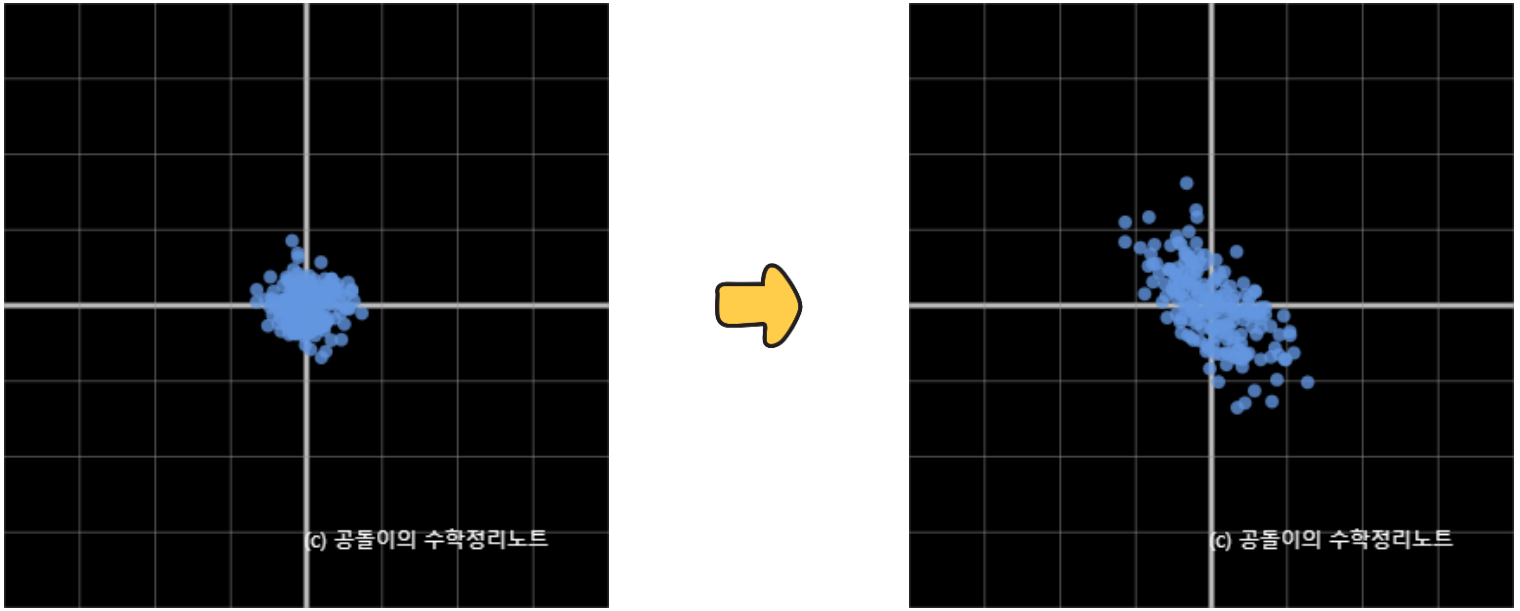
공분산 행렬을 통해 데이터를 변환하면 분산과 공분산만큼 공간이 변화

주성분 분석

공분산 행렬

공분산 행렬 $\begin{bmatrix} 3 & -2 \\ -2 & 4 \end{bmatrix}$ 을 통해 데이터를 **변환**

데이터가 **어느 방향**으로 **어떻게 분포**되어 있는지 알 수 있음



공분산 행렬

데이터의 분산과 정보량의 관계

차원축소는 정보의 손실을 동반하므로
중요한 정보를 보존하면서 차원을 축소해야 함

⋮

분산을 통해 변수의 정보량 파악 가능

항상 정답은 아니지만

설명변수의 분산이 크면 좋은 예측이 가능

(c) 공돌이의 수학정리노트

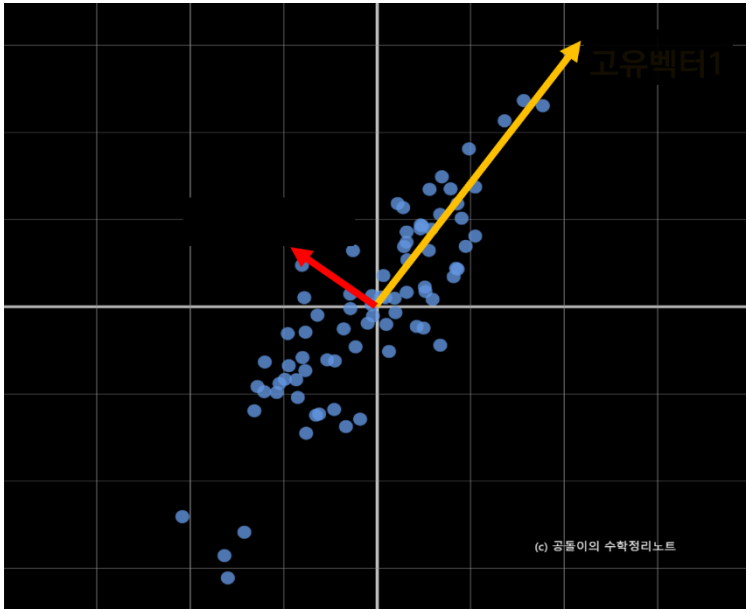
주성분 분석

분산의 보존

주성분 찾기

정보를 최대한 보존하는 선에서 차원을 축소하려면

분산이 큰 방향으로 데이터들을 정사영



그림에서 노란색 벡터가

데이터의 분산을 제일 잘 보존

↓

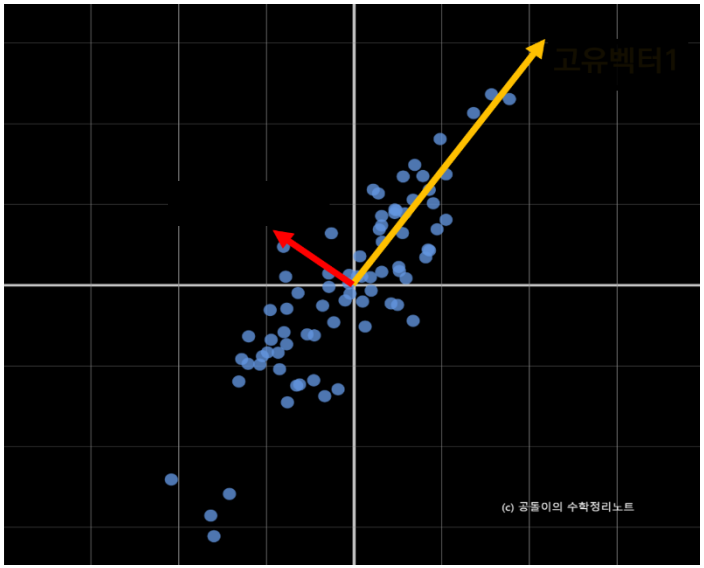
공분산 행렬을 통해 찾을 수 있음

주성분 분석

분산의 보존

공분산 행렬이라는 선형변환을 적용했을 때 노란색 벡터를 찾는 것은

데이터를 **어느 방향**으로 **얼마나** 잡아당겼냐의 문제

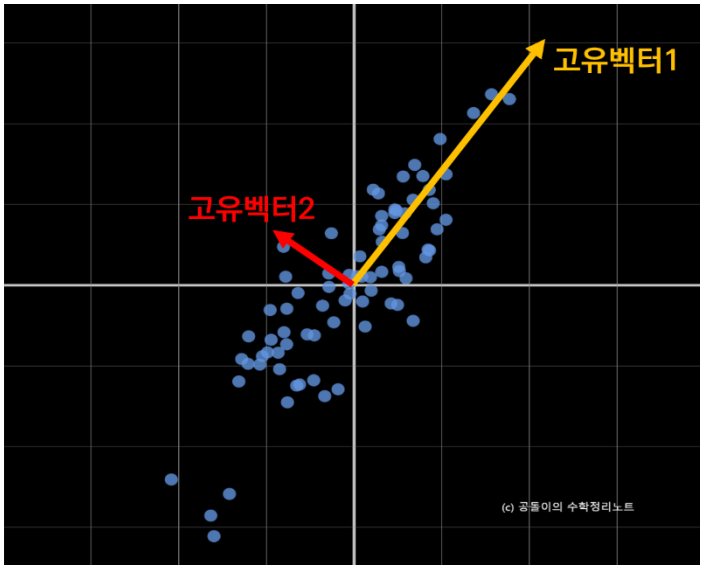


주성분 분석

분산의 보존

공분산 행렬이라는 선형변환을 적용했을 때 노란색 벡터를 찾는 것은

데이터를 **어느 방향**으로 **얼마나** 잡아당겼냐의 문제



해당 벡터는 공분산 행렬의 고유벡터

주성분 분석

분산의 보존

고유벡터는 선형변환 시 방향은 유지한 채로 길이만 변하는 벡터이므로
공분산 행렬의 **고유값**을 통해 잡아당긴 정도인

분산의 크고 작음을 알 수 있음

선대팀 2주차 클린업 참고

아 하!

고유값이 큰 순서대로 고유벡터를 정렬하면
분산이 큰 순서대로 주성분을 구하는 것

1 주성분 분석

분산의 보존

고유벡터는 선형변환 시 방향은 유지한 채로 길이만 변하는 벡터이므로
공분산 행렬의 **고유값**을 통해 잡아당긴 정도인

분산의 크고 작음을 알 수 있음

선대팀 2주차 클린업 참고

고유값이 큰 순서대로 고유벡터를 정렬하면
분산이 큰 순서대로 **주성분**을 구하는 것!

주성분 분석

주성분 선택

p차원 데이터를 m차원까지 선택하는 주성분 분석 진행할 때

고유값은 p개이며 $\lambda_1, \lambda_2, \ldots, \lambda_p$로 표현 가능

공분산 행렬이 full rank임을 가정

전체 데이터의 90% 설명

전체 데이터의 분산 중 90%만큼

설명하는 차원까지 감소

$$\frac{\sum_{j=1}^{m} \lambda_j}{\sum_{i=1}^{p} \lambda_i} \geqq 0.9$$

Elbow Point

Scree plot의 elbow point를 사용

다소 주관적

주성분 분석

주성분 선택

p차원 데이터를 m차원까지 선택하는 주성분 분석 진행할 때

고유값은 p개이며 $\lambda_1, \lambda_2, \ldots, \lambda_p$로 표현 가능

공분산 행렬이 full rank임을 가정

전체 데이터의 90% 설명

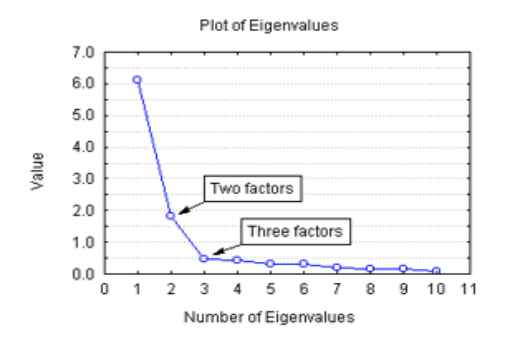
전체 데이터의 분산 중 90%만큼

설명하는 차원까지 감소

$$\frac{\sum_{j=1}^{m} \lambda_j}{\sum_{i=1}^{p} \lambda_i} \geqq 0.9$$

Elbow Point

Scree plot의 elbow point를 사용



다소 **주관적**

주성분 분석 *PCA*

PCA 요약

데이터를 설명하는 **차원 축을 변경**하고,

그 중 **많은 정보를 가진 축**만 남기는 것

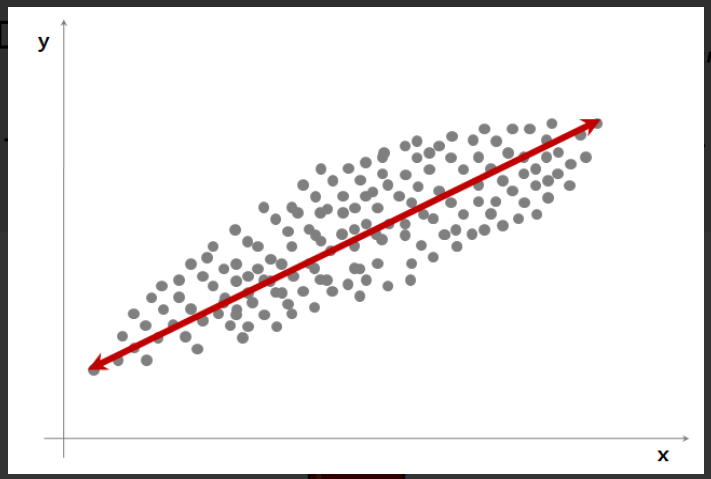
원래 데이터에서 명시되었던

변수 축의 변경으로 인해 **해석력 상실**

주성분 분석 PCA

PCA 요약

그렇다면 언제 PCA를 사용해야 할까?



빨간색 벡터 한 축으로 데이터를 표현해도,

소실되는 정보량이 적어 PCA 사용 적합

축을 변경시켜 원래 데이터에서 명시되었던

⋮

변수들의 축의 변경으로 인한 해석력을 상실

다중공선성 문제가 심각할수록 잘 작동함

2

고차원 시각화

고차원 시각화

고차원 시각화의 필요성

고차원의 데이터를 이용할 때, 클러스터링 결과 등의 시각화는 어려움

해석이 어렵고 그림이 복잡해지기 때문

↓

고차원 시각화 이용!

2 고차원 시각화

고차원 시각화

고차원 시각화의 필요성

고차원의 데이터를 이용할 때, 클러스터링 결과 등의 시각화는 어려움

해석이 어렵고 그림이 복잡해지기 때문

고차원 시각화 주의사항

고차원 시각화는 고차원 데이터를 저차원으로 임베딩

PCA를 제외한 방법들은 저차원의 좌표 정보에만 관심

→ 각 데이터포인트들의 상호 거리만이 중요

고차원 시각화 이용!

앞으로 나올 x는 하나의 데이터포인트를 의미!

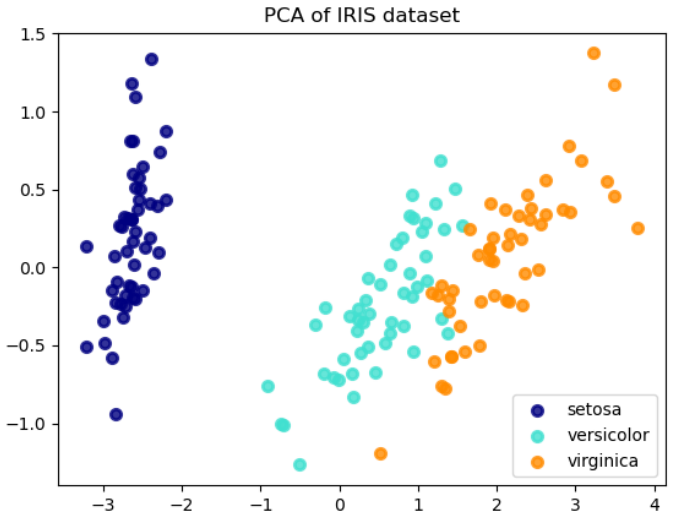
고차원 시각화

PCA *Principal Component Analysis*

PCA

가장 기초적인 고차원 시각화 기법

분산이 큰 상위 2개 축을 선정하여 그들을 축으로 하는 시각화를 진행



IRIS 데이터셋을 PCA로 시각화

2

고차원 시각화

PCA *Principal Component Analysis*

PCA 기반 시각화의 아이디어

PCA

가장 기초적인 고차원 시각화 기법

분산이 큰 상위 2개 축을 선정하여 그들을 축으로 하는 시각화를 진행

시각화는 2차원 혹은 3차원 공간에서 원활히 사용됨

→ 차원축소 알고리즘인 PCA를 통해

3개 이내의 차원으로 축소하여 시각화 가능!

IRIS [illegible] A로 시각화

고차원 시각화

PCA *Principal Component Analysis*

MNIST 데이터 시각화 with PCA

PCA

MNIST 데이터를 잘 분리하지는 못했지만,

시각화를 **행렬분해 기반**으로 가능케 함

이론적으로 기반이 탄탄하며,

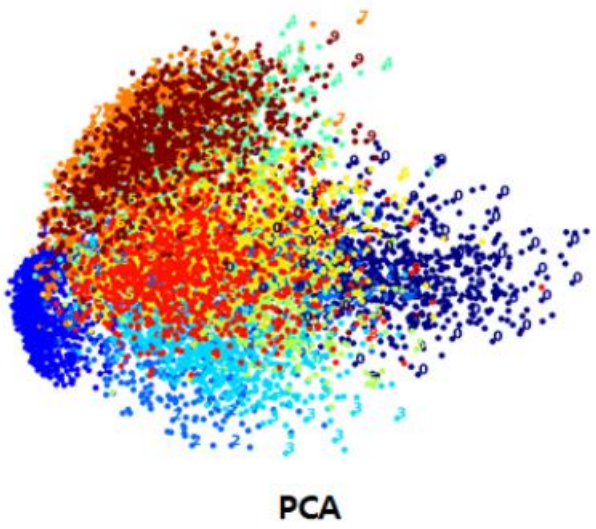
차원 축소 아이디어를 통해

시각화를 해냈다는 사실에 의의가 있음

고차원 시각화

PCA *Principal Component Analysis*

MNIST 데이터 시각화 with PCA



MNIST 데이터를 잘 분리하지는 못했지만,

시각화를 **행렬분해 기반**으로 가능케 함

이론적으로 기반이 탄탄하며,

차원 축소 아이디어를 통해

시각화를 해냈다는 **사실**에 의의가 있음

고차원 시각화

PCA *Principal Component Analysis*

MNIST 데이터 시각화 with PCA

어쨌든 해냈다는 사실이 자랑스러운 PCA

고차원 시각화

PCA *Principal Component Analysis*

PCA 기반 시각화의 특징

장점

행렬분해를 기반으로 함

차원이 크지 않고, 변수의 **분산이 데이터들의 특징을 잘 구별**할 경우

나쁘지 않은 성능 보임

단점

데이터의 유사도가 아닌, 단순히 **분산이 큰 축을 기준**으로 차원 축소

→ 시각화가 잘 되지는 않음

고차원 시각화

PCA *Principal Component Analysis*

PCA 기반 시각화의 특징

장점

행렬분해를 기반으로 함

차원이 크지 않고, 변수의 **분산이 데이터들의 특징을 잘 구별**할 경우

나쁘지 않은 성능 보임

단점

데이터의 유사도가 아닌, 단순히 **분산이 큰 축을 기준**으로 차원 축소

→ 시각화가 잘 되지는 않음

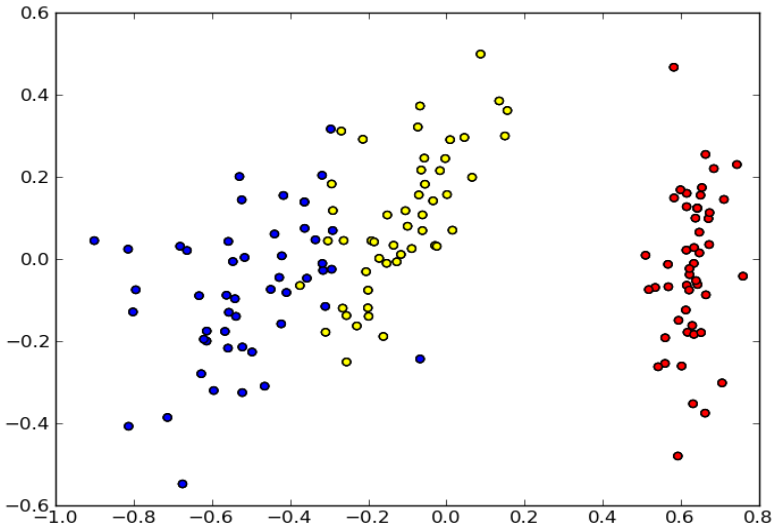
고차원 시각화

MDS *Multidimensional Scaling*

MDS

다차원 척도법으로 불리는 고전적인 시각화 방법

케이스 간의 거리를 바탕으로 이들 간의 관계 구조를 시각적으로 표현



IRIS 데이터셋을 MDS로 시각화

2 고차원 시각화

MDS *Multidimensional Scaling*

MDS

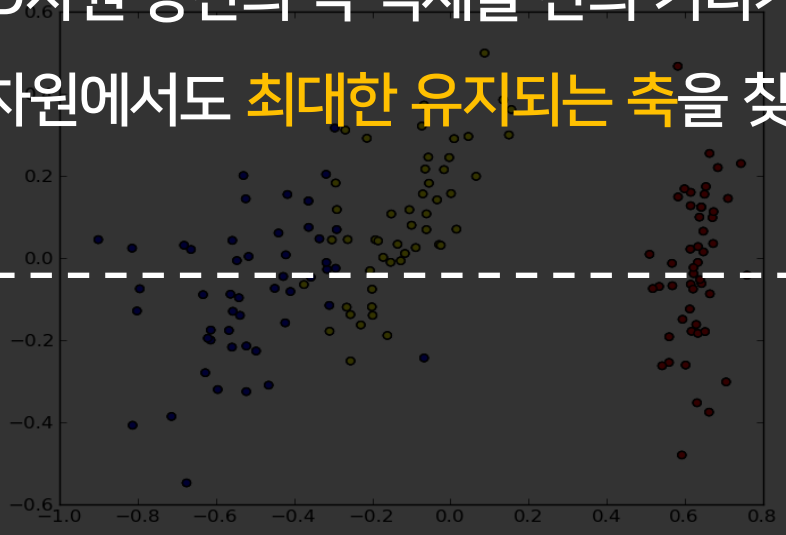
MDS의 목표

다차원 척도법으로 불리는 고전적인 시각화 방법

케이스 간의 거리를 바탕으로 이들 간의 관계 구조를 시각적으로 표현

D차원 공간의 각 객체들 간의 거리가

저차원에서도 최대한 유지되는 축을 찾자!



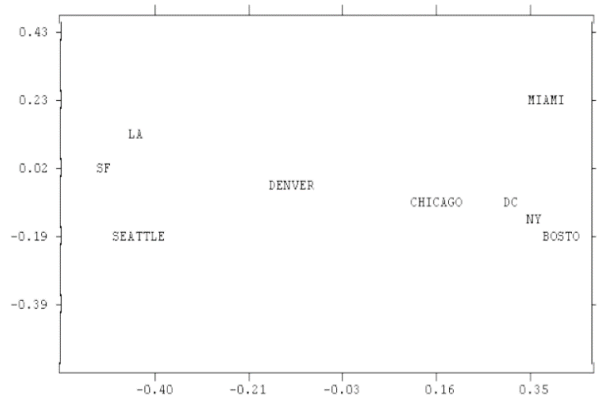
IRIS 데이터셋을 MDS로 시각화

고차원 시각화

MDS *Multidimensional Scaling*

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

미국 도시 간 거리 행렬



MDS를 학습한 결과

거리 행렬이 2차원 좌표로부터 계산된 정보이기 때문에
2차원 지도가 온전히 복원됨

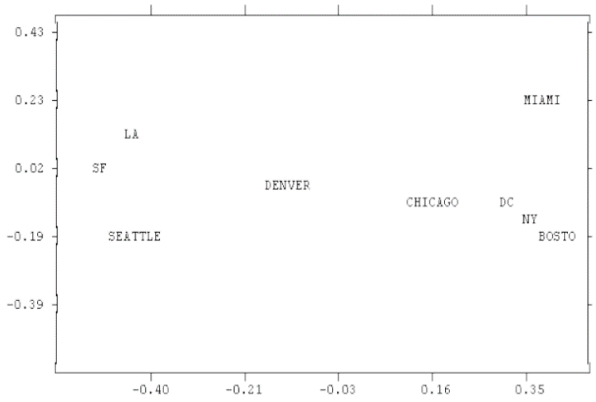
단, 새로 만들어진 공간은 거리 정보만 반영했기 때문에
임의의 방향으로 **회전**될 수 있음

고차원 시각화

MDS *Multidimensional Scaling*

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

미국 도시 간 거리 행렬



MDS를 학습한 결과

거리 행렬을 사용하는 방법론이지만, 실제 거리 뿐만 아니라

변수 간의 차이를 통해 거리를 계산할 수 있기 때문에

활용도가 높은 방법론!

고차원 시각화

MDS *Multidimensional Scaling*

MDS 알고리즘

☝ d차원 데이터 X에서 **유사도** 혹은 **거리 행렬** D를 구하고,

✌ 그 정보를 **내적 행렬** B에 넣은 뒤,

🤟 원하는 p**차원 행렬** $\tilde{X}$를 구현

이때 행렬 D만 필요로 하기 때문에 원데이터 X는 필수가 아님

고차원 시각화

MDS *Multidimensional Scaling*

① 유사도 혹은 거리 행렬 D 구하기

객체 간의 좌표가 존재한다면, 행렬 D의 원소는 다음과 같은 조건을 만족

$$(1) \quad d_{ij} \geq 0$$

$$(2) \quad d_{ii} = 0$$

$$(3) \quad d_{ij} = d_{ji}$$

In addition to $(1), (2), (3)$, it satisfies $d_{ij} \leq d_{ik} + d_{kj}$

이때 다음의 거리 혹은 유사도 행렬을 고려 가능

거리 행렬: Euclidean, Manhattan, etc.

유사도 행렬: Correlation, Jaccard, etc.

자카드 유사도

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

교집합을 합집합으로 나눈 것!

고차원 시각화

MDS *Multidimensional Scaling*

① 유사도 혹은 거리 행렬 D 구하기

객체 간의 좌표가 존재한다면, 행렬 D의 원소는 다음과 같은 조건을 만족

$$(1) \quad d_{ij} \geq 0$$

$$(2) \quad d_{ii} = 0$$

$$(3) \quad d_{ij} = d_{ji}$$

In addition to $(1), (2), (3)$, it satisfies $d_{ij} \leq d_{ik} + d_{kj}$

이때 다음의 거리 혹은 유사도 행렬을 고려 가능

거리 행렬: Euclidean, Manhattan, etc.

유사도 행렬: Correlation, Jaccard, etc.

자카드 유사도

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

교집합을 합집합으로 나눈 것!

고차원 시각화

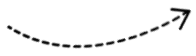
MDS *Multidimensional Scaling*

① 유사도 혹은 거리 행렬 D 구하기

데이터 X로부터 행렬 D를 구하는 것을 그림으로 표현하면 다음과 같음

X (d by n)

	x_1	x_2	x_3	x_4	…	x_n
v_1						
v_2						
v_3						
…						
v_d						



D (n by n)

	x_1	x_2	x_3	x_4	…	x_n
x_1						
x_2						
x_3						
x_4						
…						
x_n						

고차원 시각화

MDS *Multidimensional Scaling*

② 내적 행렬 B 만들기

행렬 D의 각 원소 d_{rs}들은 다음과 같이 p차원 벡터의 내적으로 표현 가능

거리 정보가 d에 들어있다고 생각하고, 최후의 $\boldsymbol{x} \in \mathbb{R}^{p\times 1}$들을 찾는 게 목적!

$$d_{rs}^2 = (\boldsymbol{x}_r - \boldsymbol{x}_s)^T(\boldsymbol{x}_r - \boldsymbol{x}_s)$$

⋮

내적 행렬 B는 거리행렬 D로부터 다음과 같이 얻음

$$[B]_{rs} = b_{rs} = \boldsymbol{x}_r^T\boldsymbol{x}_s$$

단, 위의 $\boldsymbol{x}$는 원데이터의 열벡터가 아니라, 구현할 $d \times n$행렬 $\tilde{X}$의 열벡터

MDS *Multidimensional Scaling*

② 내적 행렬 B 만들기

행렬 D의 각 원소 d_{rs}들은 다음과 같이 p차원 벡터의 내적으로 표현 가능

거리 정보가 d에 들어있다고 생각하고, 최후의 $\boldsymbol{x} \in \mathbb{R}^{p\times 1}$들을 찾는 게 목적!

$$d_{rs}^2 = (\boldsymbol{x}_r - \boldsymbol{x}_s)^T(\boldsymbol{x}_r - \boldsymbol{x}_s)$$

⋮

내적 행렬 B는 거리행렬 D로부터 다음과 같이 얻음

$$[B]_{rs} = b_{rs} = \boldsymbol{x}_r^T\boldsymbol{x}_s$$

단, 위의 $\boldsymbol{x}$는 원데이터의 열벡터가 아니라, 구현할 $d \times n$행렬 $\tilde{X}$의 열벡터

고차원 시각화

MDS *Multidimensional Scaling*

② 주어진 게 d_{rs} 밖에 없는데, B의 원소들을 어떻게 구할까?

행렬 D의 각 원소 d_{rs}들은 다음과 같이 p차원 벡터의 내적으로 표현 가능

거리 정보가 d에 들어있다고 생각하고, 최후의 x 를 찾는 게 목적!

몇 가지 가정과 복잡한 연산이 필요하며

정보가 부족하기에 각 변수별 평균이 0이라는 가정 필요

$$\sum_{r=1}^{n} x_{ri} = 0, (i = 1, 2, \cdots, p)$$

내적 행렬 B는 거리행렬 D로부터 다음과 같이 얻음

$$[B]_{rs} \equiv b_{rs} \equiv \boldsymbol{x}_r^T \boldsymbol{x}_s$$

단, 위의 x는 원데이터의 열벡터가 아니라, 구현할 $d \times n$행렬 $\tilde{X}$의 열벡터

MDS *Multidimensional Scaling*

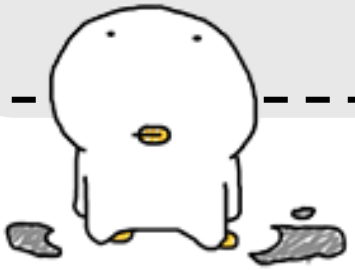
② 내적 행렬 B 만들기

일련의 연산 과정을 거치면

⋮

$$b_{rs} = -\frac{1}{2}\left(d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2 \right)$$

인 $n \times n$ 행렬 B의 원소를 구할 수 있음



수식이 궁금하다면 팀장에게 **개인적**으로 문의

MDS *Multidimensional Scaling*

③ 거리 정보를 보존하는 좌표 추출

구해진 행렬 B를 바탕으로 $p \times n$ 행렬인 $\tilde{X}$를 구해보자!

$$B = \tilde{X}\tilde{X}^T,\ rank(B) = rank\left(\tilde{X}\tilde{X}^T\right) = \text{rank}\left(\tilde{X}\right) = p$$

이때 B는 대칭행렬이고, positive semi-definite이며 rank가 p임

↓

→ p개의 음이 아닌 고윳값, $n-p$개의 0인 고윳값을 구할 수 있음 (by EVD)

$$B = V\Lambda V^T, \Lambda = diag(\lambda_1, \cdots, \lambda_p), V = \left[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_p\right]$$

MDS *Multidimensional Scaling*

③ 거리 정보를 보존하는 좌표 추출

구해진 행렬 B를 바탕으로 $p \times n$ 행렬인 $\tilde{X}$를 구해보자!

$$B = \tilde{X}\tilde{X}^T,\ rank(B) = rank\left(\tilde{X}\tilde{X}^T\right) = \text{rank}\left(\tilde{X}\right) = p$$

이때 B는 대칭행렬이고, positive semi-definite이며 rank가 p임

→ p개의 음이 아닌 고윳값, $n-p$개의 0인 고윳값을 구할 수 있음 (by EVD)

$$B = V\Lambda V^T, \Lambda = diag(\lambda_1, \cdots, \lambda_p), V = \left[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_p\right]$$

MDS *Multidimensional Scaling*

③ 거리 정보를 보존하는 좌표 추출

구해진 행렬 B를 바탕으로 $p \times n$ 행렬인 $\tilde{X}$를 구해보자!

0인 고윳값이 $n-p$개 있으므로, B는 다음과 같이 표현 가능

$$B_1 = V_1 \Lambda_1 V_1^T, \Lambda_1 = diag(\lambda_1, \cdots, \lambda_p), V_1 = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_p]$$

따라서 최종적으로 구한 p차원 좌표 행렬 $\tilde{X}$는 다음과 같음

$$\tilde{X} = V_1 \Lambda_1^{\frac{1}{2}}$$

MDS *Multidimensional Scaling*

③ 거리 정보를 보존하는 좌표 추출

구해진 행렬 B를 바탕으로 $p \times n$ 행렬인 $\tilde{X}$를 구해보자!

0인 고윳값이 $n-p$개 있으므로, B는 다음과 같이 표현 가능

$$B_1 = V_1 \Lambda_1 V_1^T, \Lambda_1 = diag(\lambda_1, \cdots, \lambda_p), V_1 = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_p]$$

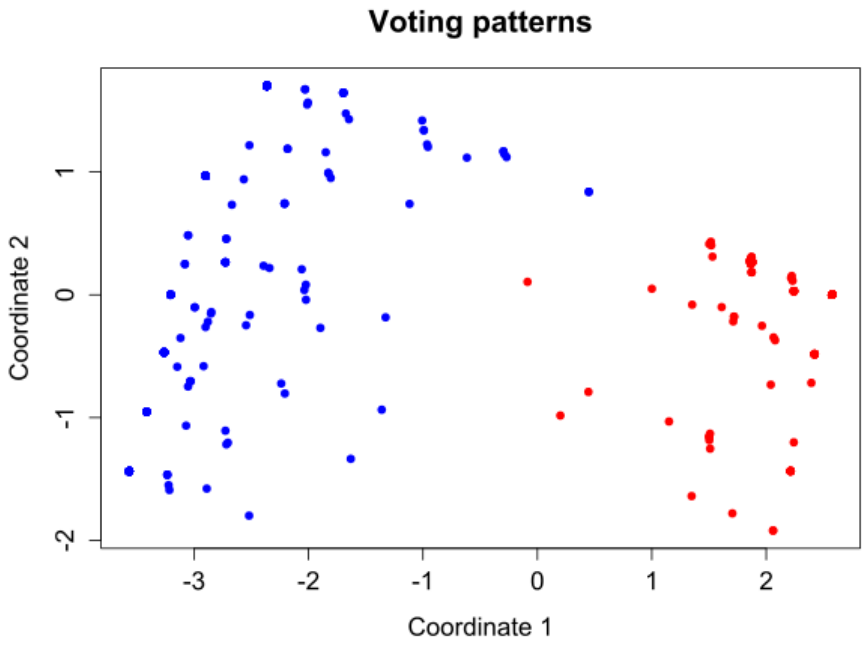
따라서 최종적으로 구한 p차원 좌표 행렬 $\tilde{X}$는 다음과 같음

$$\tilde{X} = V_1 \Lambda_1^{\frac{1}{2}}$$

고차원 시각화

MDS *Multidimensional Scaling*

③ 거리 정보를 보존하는 좌표 추출



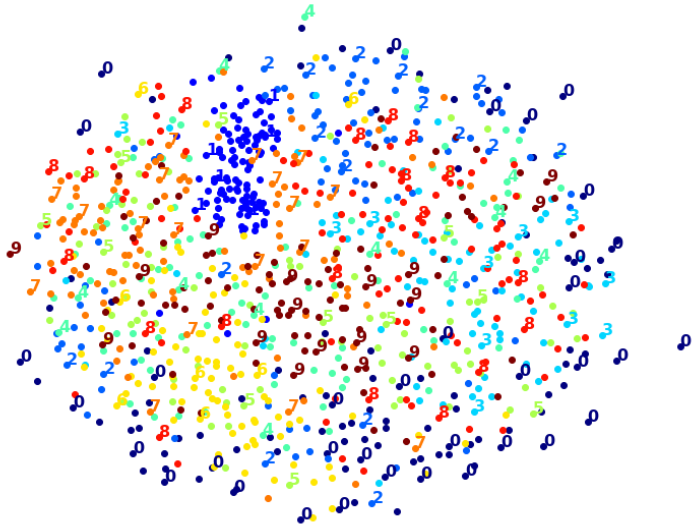
미국 하원 투표에 적용된
고전적인 MDS 예시

빨간색 점은 공화당,
파란색 점은 민주당 의원

고차원 시각화

MDS *Multidimensional Scaling*

MNIST 데이터 시각화 with MDS



차이가 잘 보이지 않음

MDS는 **차원의 증가에 매우 취약**

MDS는 모든 점들 간의 **거리 정보 중요도가 같음**

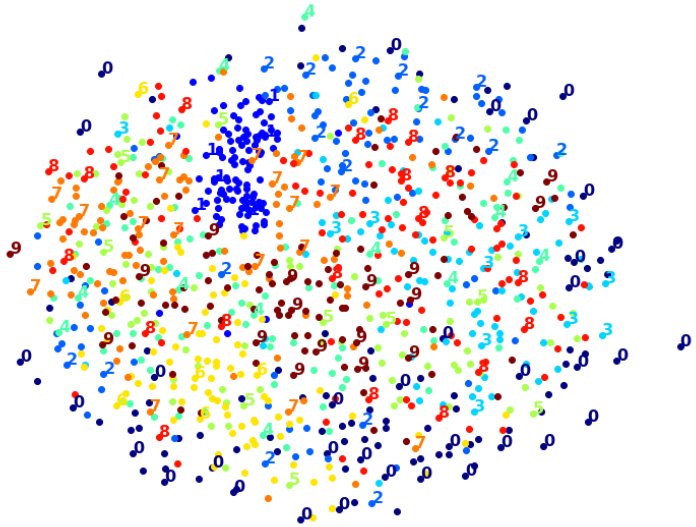
→ 가까운 점들 간의 거리 정보보다 멀리 떨어진 점들 간의 정보 영향력이 큼

무의미한 정보에 집중하는 경향이 있음

고차원 시각화

MDS *Multidimensional Scaling*

MNIST 데이터 시각화 with MDS



차이가 잘 보이지 않음

MDS는 **차원의 증가에 매우 취약**

MDS는 모든 점들 간의 **거리 정보 중요도가 같음**

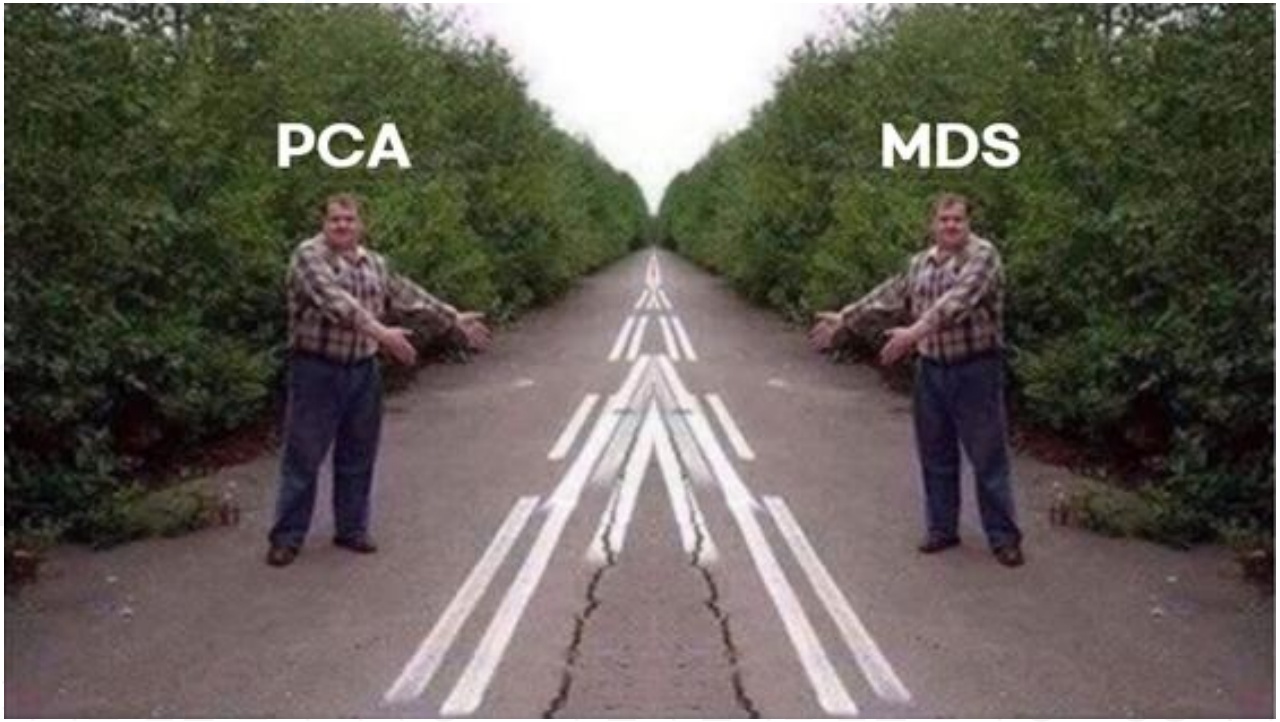
→ 가까운 점들 간의 거리 정보보다 멀리 떨어진 점들 간의 정보 영향력이 큼

무의미한 정보에 집중하는 경향이 있음

고차원 시각화

MDS *Multidimensional Scaling*

MNIST 데이터 시각화 with MDS



어쨌든 해냈다는 사실이 자랑스러운 PCA와 MDS

고차원 시각화

MDS *Multidimensional Scaling*

MDS 기반 시각화의 특징

장점

고차원에서의 **객체 간 거리**를 저차원에서 **보존**하는 것에 집중

비선형적인 데이터에 대해서도 잘 작동하며 **다양한 거리 척도**를 사용 가능

단점

고차원에서는 **무의미한 정보에 집중**해 시각화가 잘 이루어지지 않음

→ 보다 저차원에서 직관적인 시각화 가능

고차원 시각화

MDS *Multidimensional Scaling*

MDS 기반 시각화의 특징

장점

고차원에서의 **객체 간 거리**를 저차원에서 **보존**하는 것에 집중

비선형적인 데이터에 대해서도 잘 작동하며 **다양한 거리 척도**를 사용 가능

단점

고차원에서는 **무의미한 정보에 집중**해 시각화가 잘 이루어지지 않음

→ **보다 저차원에서 직관적인 시각화 가능**

고차원 시각화

ISOMAP

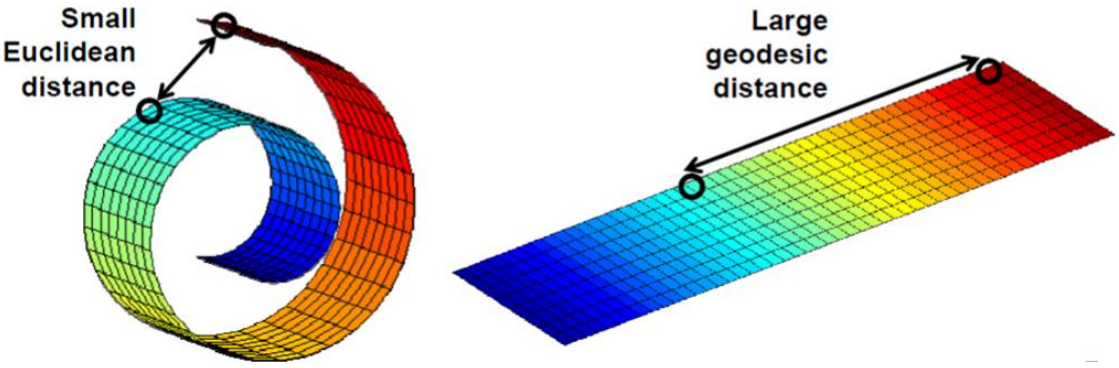
ISOMAP *Isometric Mapping*

MDS와 PCA를 결합한 기법

모든 점 사이의 **측지 거리를 유지**하는 임베딩을 추구

측지 거리란?

두 측점 사이의 **매니폴드를 따라** 이루어진 거리



측점 사이의 유클리드 거리는 가깝지만 **측지 거리는 멂**

ISOMAP

ISOMAP *Isometric Mapping*

ISOMAP의 목표

MDS와 PCA를 결합한 기법

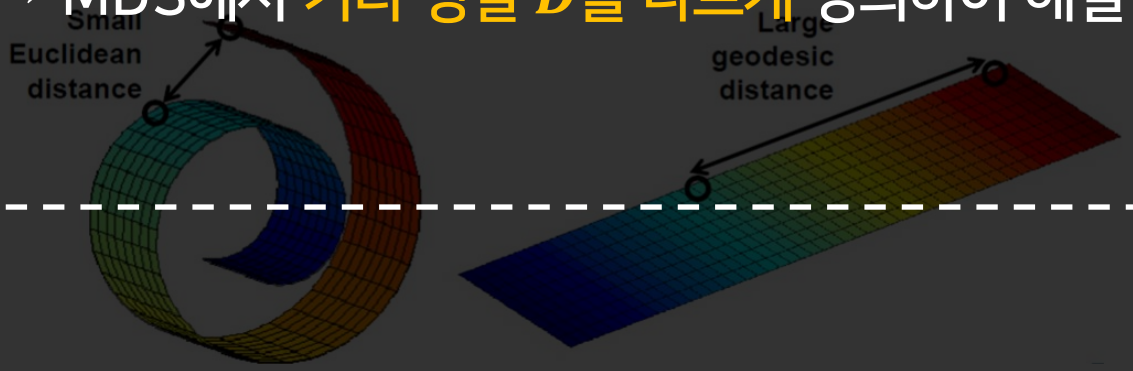
모든 점 사이의 **측지 거리를 유지**하는 임베딩을 추구

측지 거리란?

내재된 매니폴드 구조를 **평면으로 풀어**내어 거리를 구하자!

두 측점 사이의 **매니폴드를 따라** 이루어진 거리

→ MDS에서 **거리 행렬 D를 다르게** 정의하여 해결



측점 사이의 유클리드 거리는 가깝지만 **측지 거리는 멈**

고차원 시각화

ISOMAP 프로세스

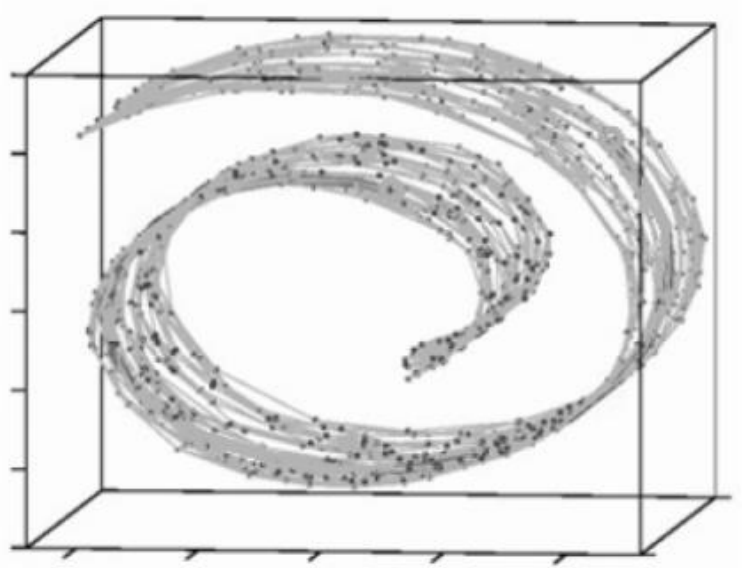
① 인접 그래프 구축

ε-ISOMAP

두 점 사이의 거리가
ε**보다 작은 점**을 이웃으로 선택

k-ISOMAP

k-nearest neighbor에
해당하는 점을 이웃으로 선택



ISOMAP 프로세스

② 최단 경로 구하기

두 점들 간의 **최단 경로를 탐색**해 거리 행렬 D를 만듦

최단 경로 탐색을 위해 다익스트라 or 플로이드 이용

다익스트라 *Dijkstra*

한 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(ElogV)$

플로이드 *Floyd*

모든 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(n^3)$

V: 노드, E: 간선

고차원 시각화

ISOMAP 프로세스

② 최단 경로 구하기

두 점들 간의 **최단 경로를 탐색**해 거리 행렬 D를 만듦

최단 경로 탐색을 위해 다익스트라 or 플로이드 이용

다익스트라 *Dijkstra*

한 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(ElogV)$

플로이드 *Floyd*

모든 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(n^3)$

V: 노드, E: 간선

ISOMAP 프로세스

② 최단 경로 구하기

두 점들 간의 **최단 경로를 탐색**해 거리 행렬 D를 만듦

최단 경로 탐색을 위해 다익스트라 or 플로이드 이용

다익스트라 *Dijkstra*

한 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(ElogV)$

V: 노드, E: 간선

플로이드 *Floyd*

모든 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(n^3)$

ISOMAP 프로세스

② 최단 경로 구하기

두 점들 간의 **최단 경로를 탐색**해 거리 행렬 D를 만듦

최단 경로 탐색을 위해 다익스트라 or 플로이드 이용

다익스트라 *Dijkstra*

한 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(ElogV)$

플로이드 *Floyd*

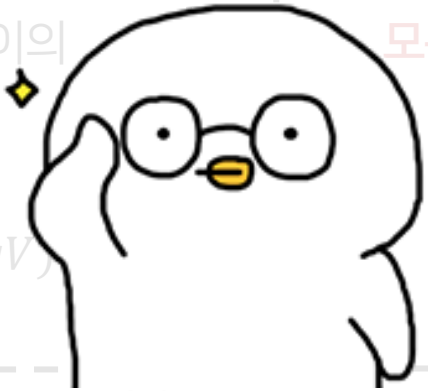
모든 점에서 모든 점 사이의

최단거리 탐색

시간복잡도: $O(n^3)$

V: 노드, E: 간선

→ 자세한 내용은 알고리즘 or 그래프 이론에서!

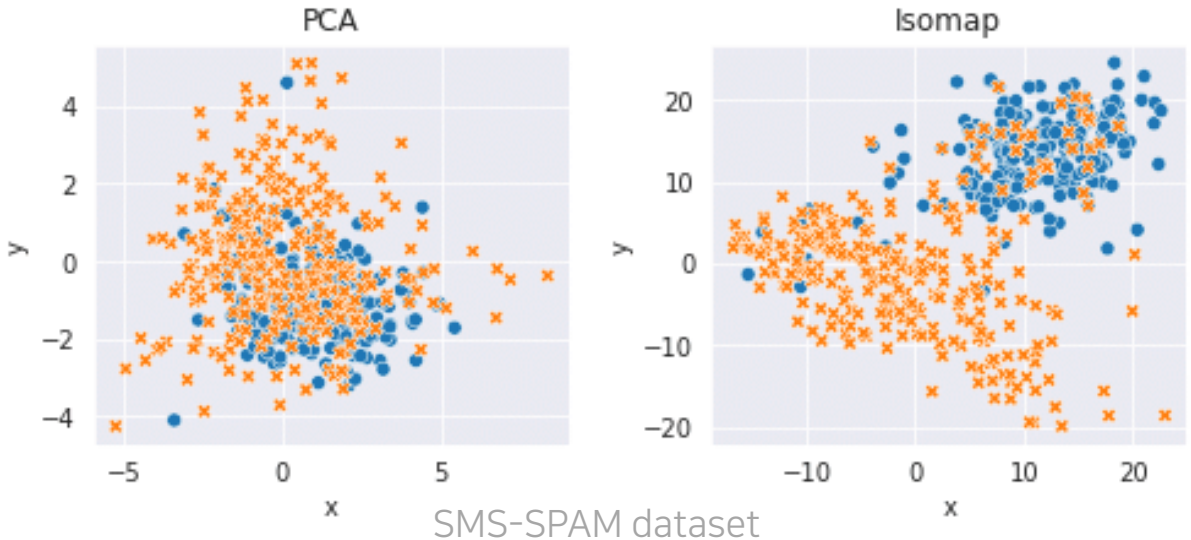


고차원 시각화

ISOMAP 프로세스

③ MDS 알고리즘 수행하기

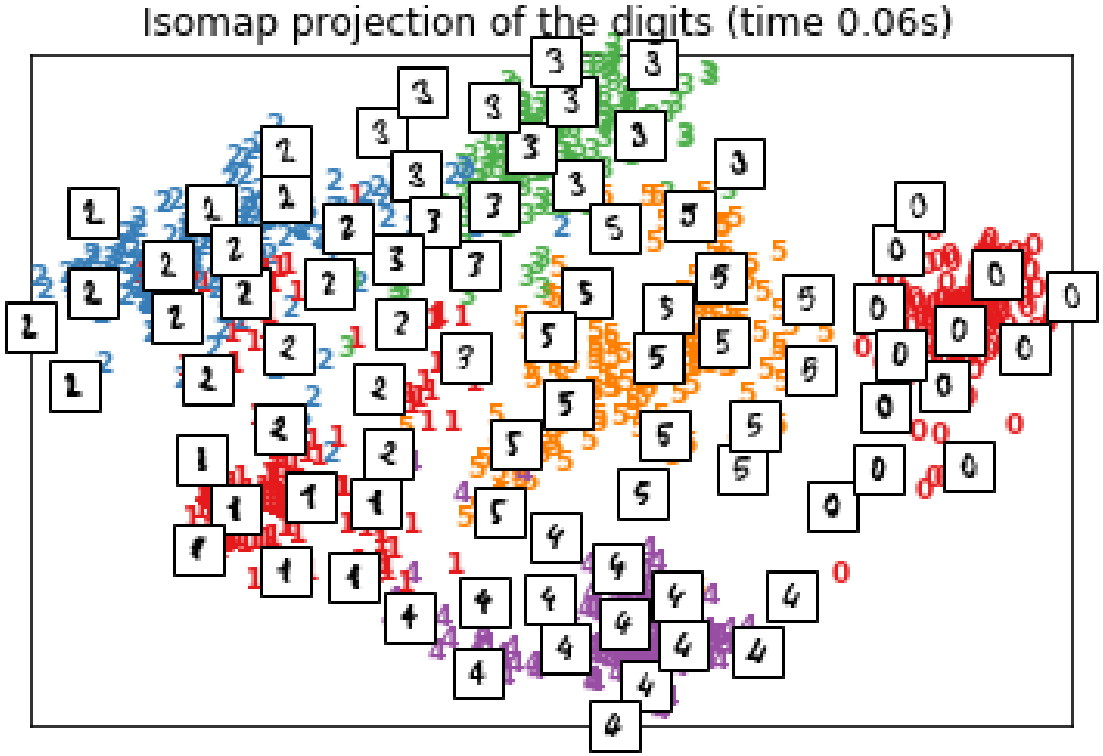
재정의한 거리 행렬 D를 이용해 똑같이 MDS를 수행



→ PCA에 비해 스팸을 매우 잘 분류

고차원 시각화

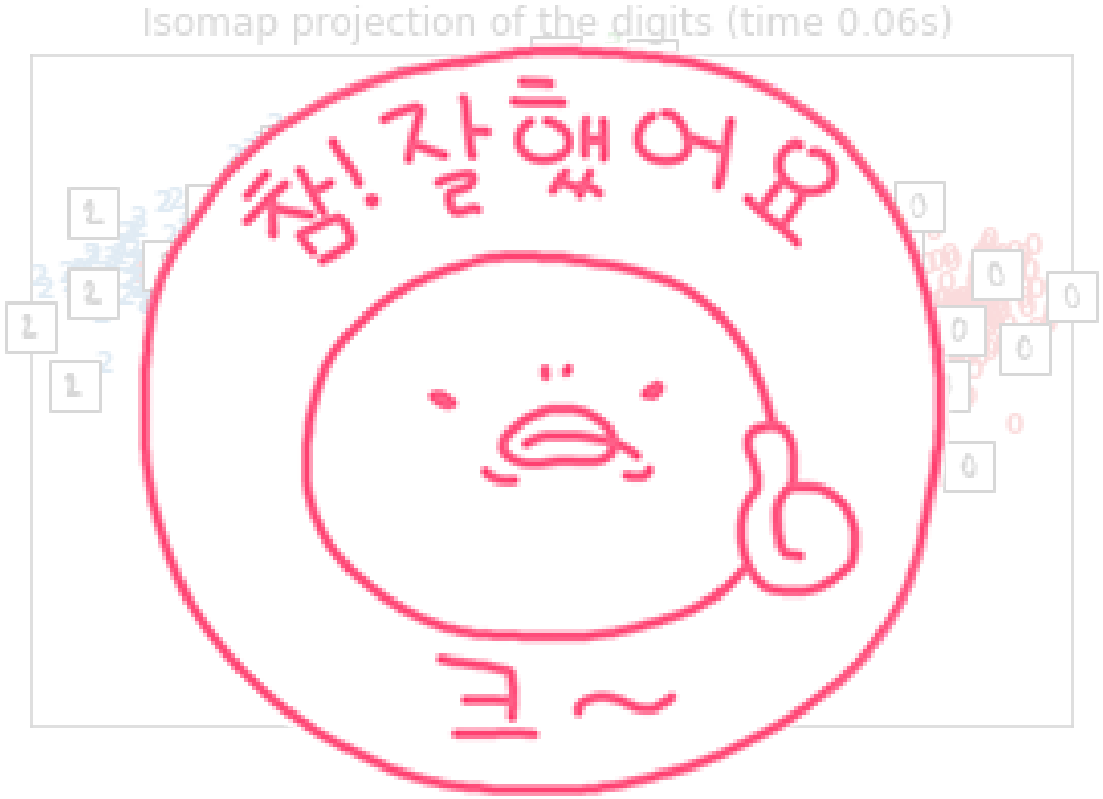
MNIST 데이터 시각화 with ISOMAP



→ 7개의 숫자만 사용하긴 했지만 이전의 방법보다 훨씬 좋은 성능을 보임

고차원 시각화

MNIST 데이터 시각화 with ISOMAP



→ 7개의 숫자만 사용하긴 했지만 이전의 방법보다 훨씬 좋은 성능을 보임

고차원 시각화

ISOMAP의 특징

장점

비선형 차원 축소기법으로, 내재된 **매니폴드**를 잘 반영함

비선형의 경우에 좋은 성능을 보임

단점

매니폴드가 없다면, 성능이 좋지 않음

계산 비용이 매우 큼 (다익스트라, 플로이드)

→ 데이터의 형태를 잘 파악하고 사용해야 함

ISOMAP의 특징

장점

비선형 차원 축소기법으로, 내재된 **매니폴드**를 잘 반영함

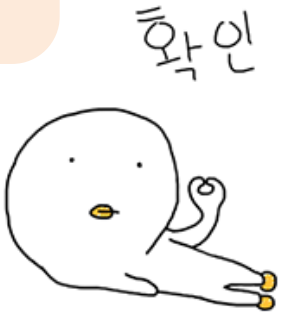
비선형의 경우에 좋은 성능을 보임

단점

매니폴드가 없다면, 성능이 좋지 않음

계산 비용이 매우 큼 (다익스트라, 플로이드)

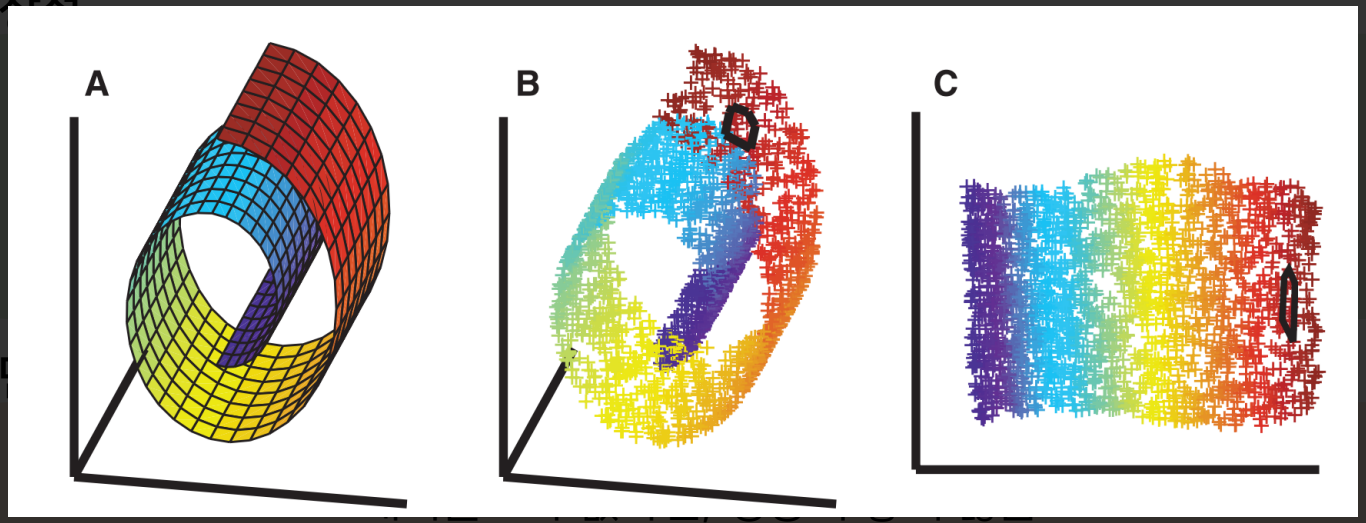
→ **데이터의 형태를 잘 파악하고 사용해야 함**



고차원 시각화

ISOMAP의 특징

얼마나 매니폴드를 잘 반영하나요?



계산 비용이 매우 큼 (다익스트라, 플로이드)

매니폴드를 **완벽하게** 펴낸 모습을 볼 수 있음

→ 데이터의 형태를 잘 파악하고 사용해야 함

고차원 시각화

LLE

LLE *Locally Linear Embedding*

고차원 공간에서 인접한 데이터들 간의 **선형적 구조를 보존**하며 임베딩

그럼 LLE는 선형 모델인가요?

⋮

LLE는 **좁은 범위**(Locally)에서 **선형 모델**(Linear)을 연결하며
매니폴드를 표현하는 알고리즘

→ 좁은 범위에서 선형적으로 표현했으므로 넓게 보면 **비선형 모델**!

2 고차원 시각화

LLE

LLE *Locally Linear Embedding*

LLE의 목표

고차원 공간에서 인접한 데이터들 간의 **선형적 구조를 보존**하며 임베딩

고차원에서 이웃관계인 점들

⌄

그럼 LLE는 선형 모델인가요?

저차원에서도 이웃관계여야 한다!

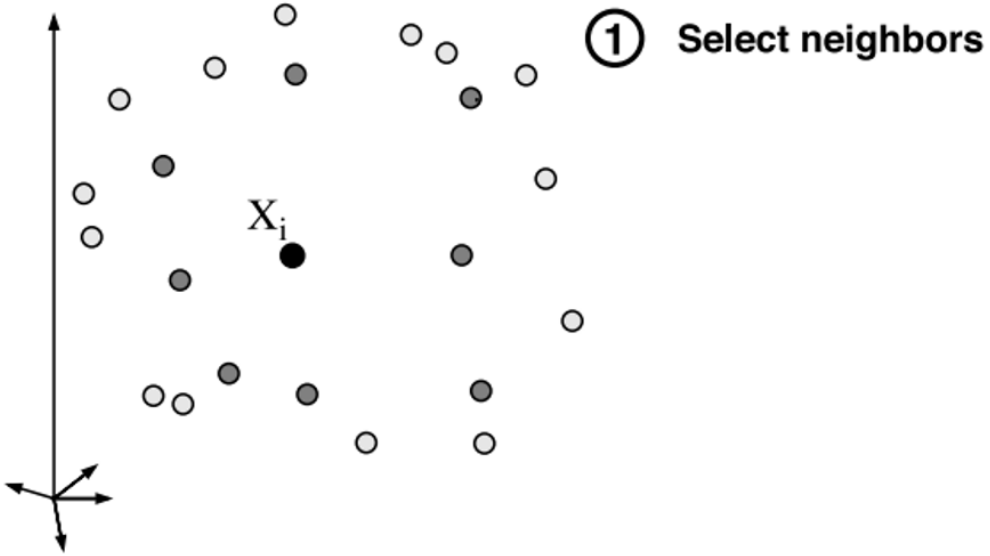
LLE는 **좁은 범위**(Locally)에서 **선형 모델**(Linear)을 연결하며

매니폴드를 표현하는 알고리즘

→ 좁은 범위에서 선형적으로 표현했으므로 넓게 보면 비선형이 가능

LLE 프로세스

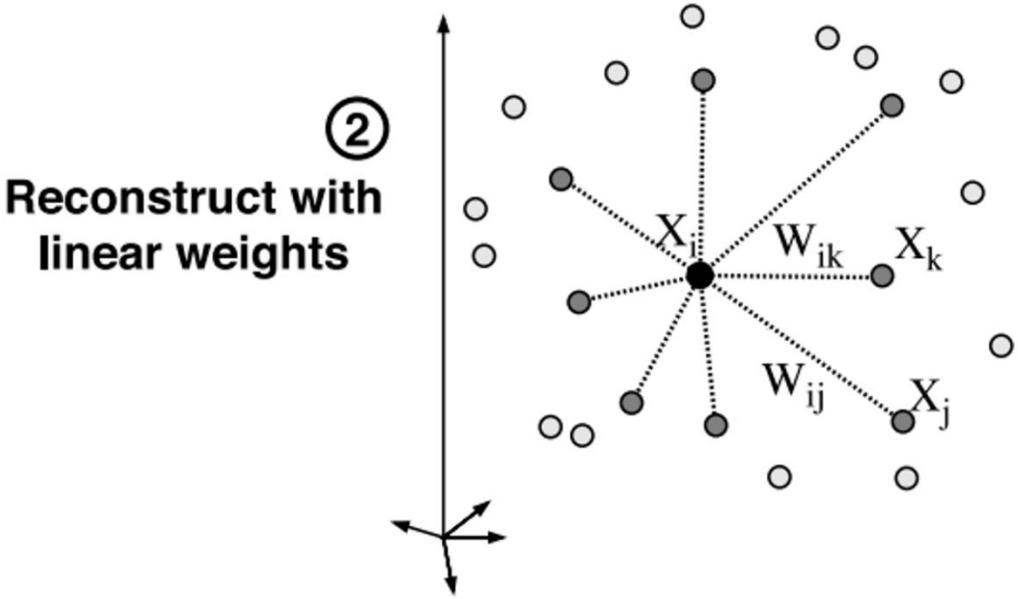
① 가장 가까운 이웃 탐색



각 데이터포인트에서 ***k*개의 이웃**을 선정

LLE 프로세스

② 가중치 매트릭스 구성 (by local linearity)



k개의 이웃의 선형결합으로 X_i를 최대한 구현하는 W_{ij}를 학습

LLE 프로세스

② 가중치 매트릭스 구성 (by local linearity)

k개의 이웃의 선형결합으로 X_i를 최대한 구현하는 W_{ij}를 학습

$$E(W) = \sum_i \left| \boldsymbol{x}_i - \sum_j W_{ij} \boldsymbol{x}_j \right|^2$$

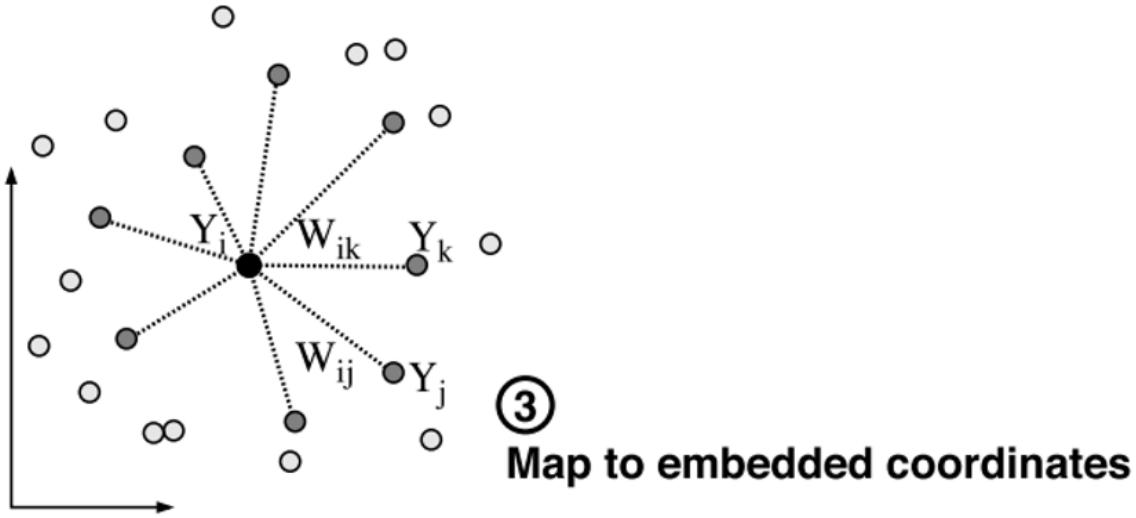
$$st.\ W_{ij} = 0\ does\ not\ belong\ to\ the\ neighbor\ of\ \boldsymbol{x}_i$$

$$\forall_i, \sum_i W_{ij} = 1$$

손실 함수로서 이웃 점들에게 둘러 쌓이면 0, 아니면 0보다 커져 **총합을 줄이는 방향으로 최적화**

LLE 프로세스

③ 부분 고윳값 분해



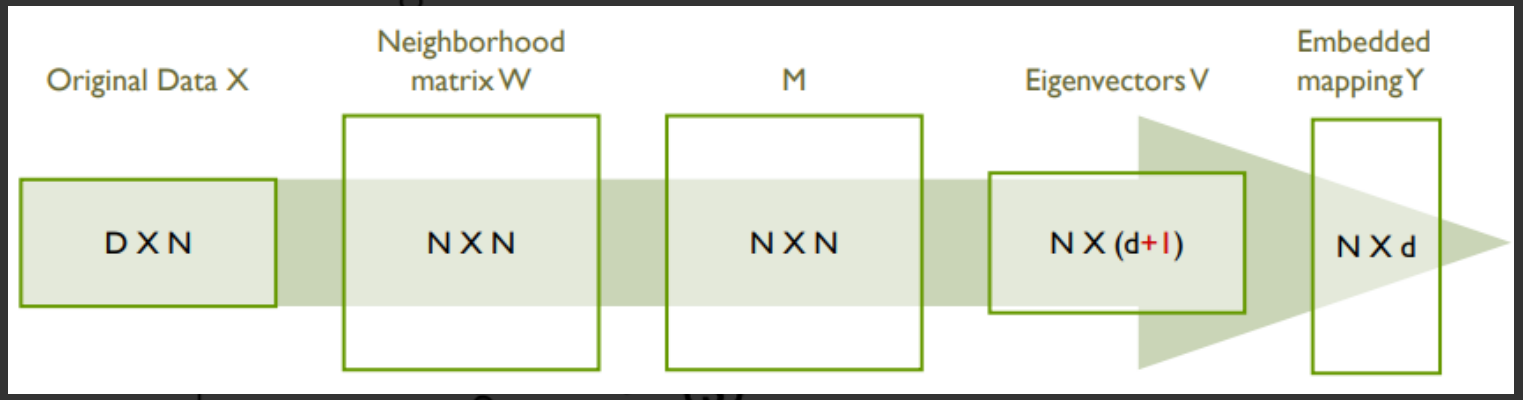
가중치를 보장하며 **차원을 축소**하여 최적의 임베딩을 찾음

$$\phi(W) = \sum_i \left| \boldsymbol{y}_i - \sum_j W_{ij} \boldsymbol{y}_j \right|^2$$

고차원 시각화

LLE 프로세스 어떻게 SVD로 임베딩을 찾나요?

③ 부분 고윳값 분해



Map to embedded coordinates

Sparse Matrix $M = (I - W)^T(I - W)$를 구한 후

SVD를 통해 하위 n개 고유벡터 추출

가중치를 보정하여 차원을 축소하여 최적의 임베딩을 찾음

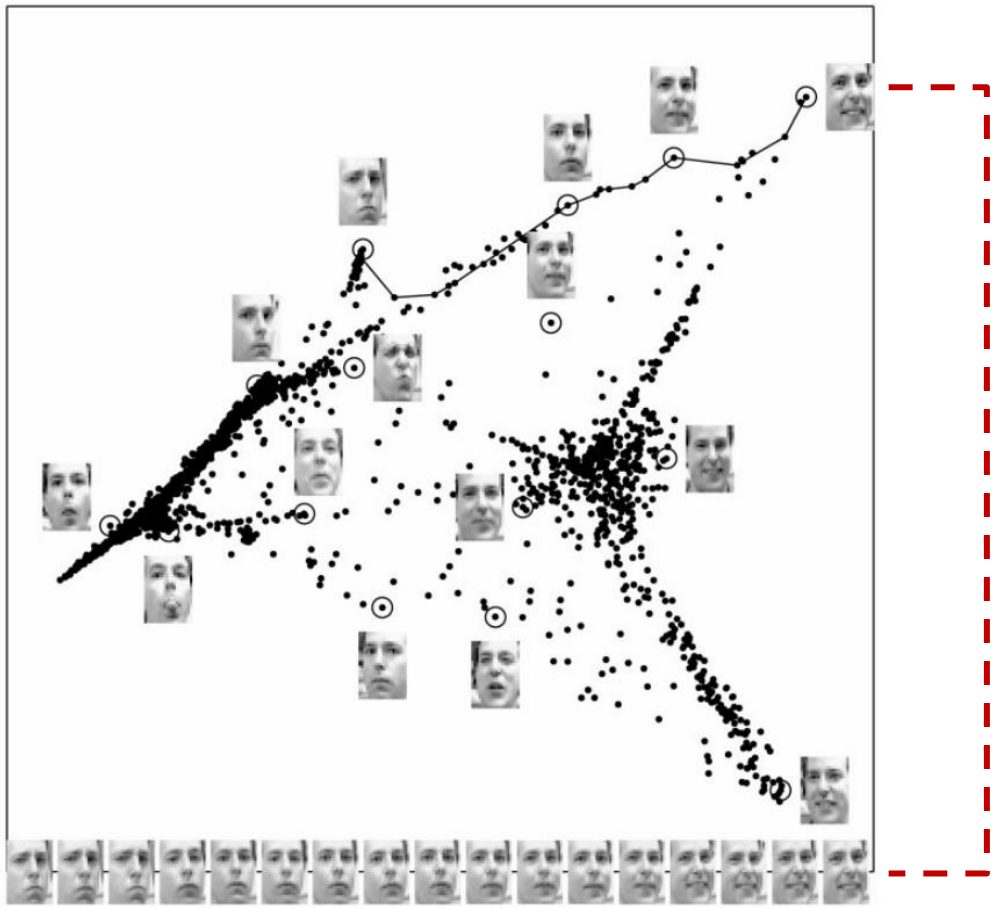
$$\Phi(W) = \sum_i \left| \mathbf{y}_i - \sum_j W_{ij} \mathbf{y}_j \right|^2$$

By Rayleitz-Ritz Theorem (꽤나 심오하여 깊게 다루지 않음)

궁금하면 팀장에게 개인적으로 질문

고차원 시각화

LLE 프로세스



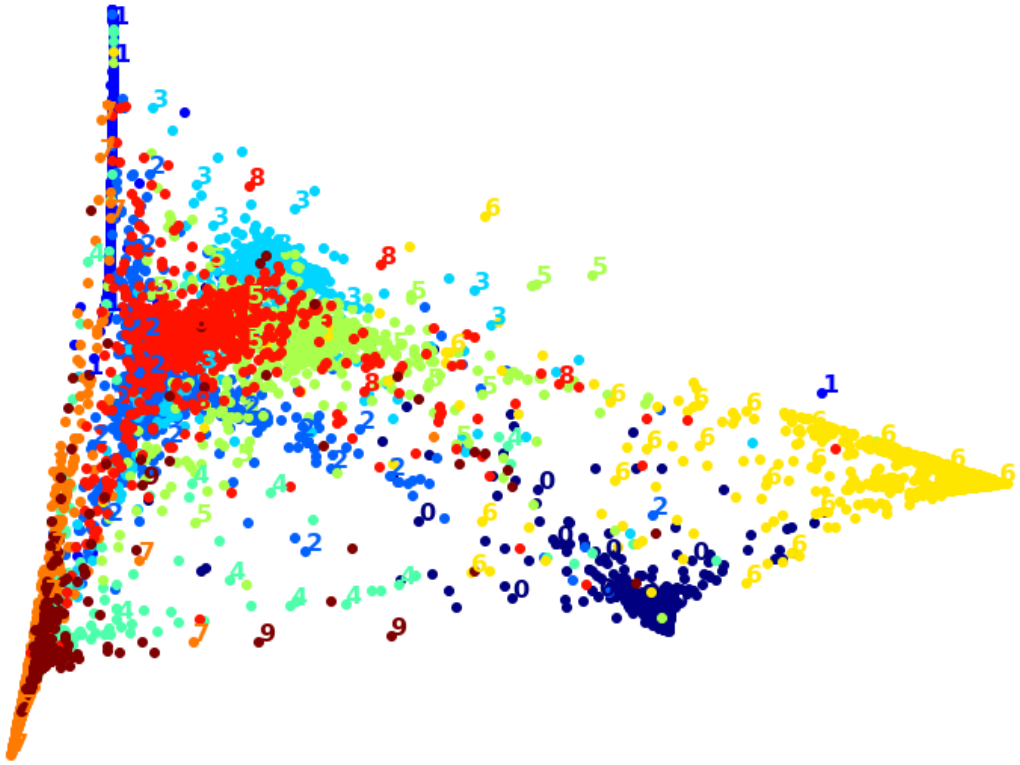
이웃 간의 구조를 보존하였더니

이웃들끼리 한 축으로 이동하며 학습

→ **연속적인 정보를 보존**할 수 있게 됨

고차원 시각화

MNIST 데이터 시각화 with LLE



→ 저차원으로 매핑된 후 직선의 형태들로 임베딩됨

LLE의 특징

장점

Local Minimum 문제가 **발생하지 않음** (SVD로 최적화)

비선형 임베딩 생성이 가능

단점

ISOMAP보다는 매니폴드를 잘 못잡아냄

국소 선형성 가정에 부합하지 않으면 성능 저하

→ 이웃관계 유지가 중요한 경우 사용을 추천함

LLE의 특징

장점

Local Minimum 문제가 **발생하지 않음** (SVD로 최적화)

비선형 임베딩 생성이 가능

단점

ISOMAP보다는 매니폴드를 잘 못잡아냄

국소 선형성 가정에 부합하지 않으면 성능 저하

→ 이웃관계 유지가 중요한 경우 사용을 추천함

고차원 시각화

t-SNE

t-SNE와 LLE의 공통점

고차원에서 **이웃 관계**인 점은
저차원에서도 가깝게 임베딩하는 것이 목표

t-SNE 와 LLE의 차이점

t-SNE는 이웃보다 조금 **더 떨어져 있는 점들**까지 반영함

t-SNE

t-SNE는 t분포를 사용한 SNE 기법이므로 SNE를 먼저 알아보자!

고차원 시각화

t-SNE

SNE *Stochastic Neighbor Embedding*

SNE는 고차원 공간에서의 유클리드 거리를
포인트들 간의 **유사성**을 표현하는 **조건부 확률**로 변환하는 방법

이 때, 확률적으로 이웃을 선택하며 그 확률은 다음과 같음

고차원에서 j번째 데이터를
이웃으로 선택할 확률

$$p_{i|j} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}$$

저차원에서 j번째 데이터를
이웃으로 선택할 확률

$$p_{i|j} = \frac{e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}}{\sum_{k \neq i} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}}$$

고차원 시각화

t-SNE

SNE *Stochastic Neighbor Embedding*

SNE는 고차원 공간에서의 유클리드 거리를
포인트들 간의 **유사성**을 표현하는 **조건부 확률**로 변환하는 방법

이 때, 확률적으로 이웃을 선택하며 그 확률은 다음과 같음

고차원에서 j번째 데이터를
이웃으로 선택할 확률

$$p_{i|j} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}$$

저차원에서 j번째 데이터를
이웃으로 선택할 확률

$$p_{i|j} = \frac{e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}}{\sum_{k \neq i} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}}$$

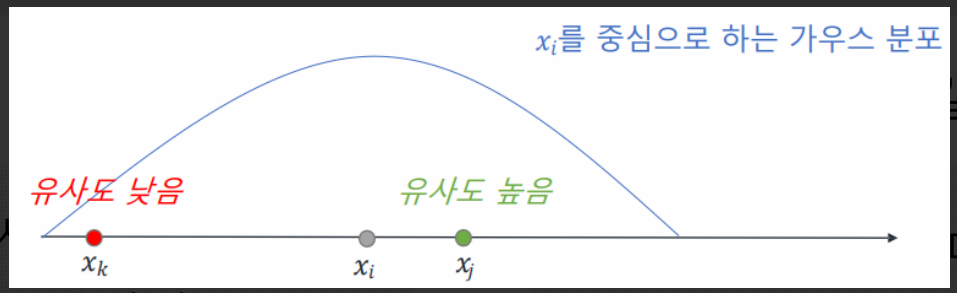
고차원 시각화

t-SNE 고차원에서 근처 점들을 이웃으로 선택하는 기준은?

SNE *Stochastic Neighbor Embedding*

Gaussian radius를 기준으로 사용하기 때문에

더 가까이 있을수록 확률



σ가 포함된 하이퍼파라미터 perplexity는 아래와 같이 계산됨

$$Perplexity(P_i) = 2^{\sum_i p_{j|i} \log_2 p_{j|i}}$$

고차원 시각화

t-SNE

SNE

SNE의 손실함수는 KL divergence!

KL divergence *Kullback-Leibler divergence*

KL divergence는 **두 분포의 유사성을 파악**하게 해주며,
스칼라 값으로 나오기에 유사도 행렬로 사용할 수는 없음

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

고차원 시각화

t-SNE

SNE

SNE의 손실함수는 KL divergence!

KL divergence *Kullback-Leibler divergence*

KL divergence는 **두 분포의 유사성을 파악**하게 해주며,
스칼라 값으로 나오기에 유사도 행렬로 사용할 수는 없음

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

두 분포 p와 q가 **유사**할수록 log의 진수 부분이 **작아짐**

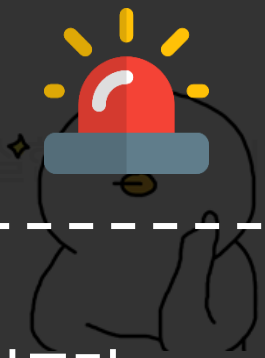
2 고차원 시각화

결국 두 분포가 유사할수록 cost 감소

고차원에서 이웃으로 뽑을 확률과 **저차원에서 이웃으로 뽑을 확률**을

최대한 **일치**시키는 것이 목표

2 고차원 시각화



그러나, SNE는 유사도가 symmetric이 아니므로

최적화가 어렵다는 문제점 존재

고차원에서 이웃으로 뽑을 확률과 저차원에서 이웃으로 뽑을 확률을

최대한 일치시키는 것이 목표

t-SNE 의 필요성!

t-SNE

t-SNE의 필요성1: 최적화의 어려움

symmetric SNE

일반적으로 상호동일하지 않은 조건부 확률을

점들간 상호동일하게 만들어주기 위해 조정함으로써

다음과 같이 **cost function을 간략화**

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

t-SNE

t-SNE의 필요성1: 최적화의 어려움

symmetric SNE

일반적으로 상호동일하지 않은 조건부 확률을

점들간 상호동일하게 만들어주기 위해 조정함으로써

다음과 같이 **cost function을 간략화**

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

유사도를 symmetric으로 만들어주어 최적화가 쉬운 형태!

고차원 시각화

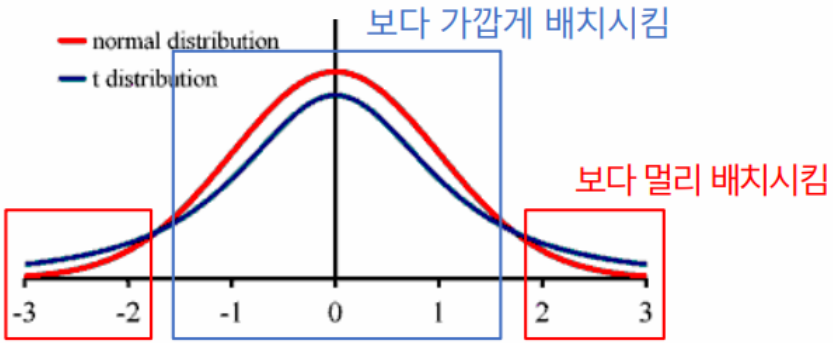
t-SNE

t-SNE의 필요성2: Crowding problem

SNE의 Crowding problem

SNE에서 가정하는 가우시안 분포는 양쪽 꼬리가 충분히 두텁지 않음

따라서 중간정도 가까운 거리와 **비교적 먼 거리를 구별하지 못하는 문제**



가우스 분포와 자유도 1의 t분포의 비교

2 고차원 시각화

이를 완화하기 위해 가우시안 분포와 유사하지만

꼬리가 더 두터운 자유도가 1인 t분포 사용

↓

“ t-SNE ”

t-SNE

t-SNE *t-Stochastic Neighbor Embedding*

SNE의 p_{ij}는 동일하게 사용하되, $\boldsymbol{q_{ij}}$**에만 t분포를 적용**한 방법으로,

t분포를 사용한 임베딩 공간의 두 점 사이 유사도 q_{ij}는 다음과 같이 정의됨

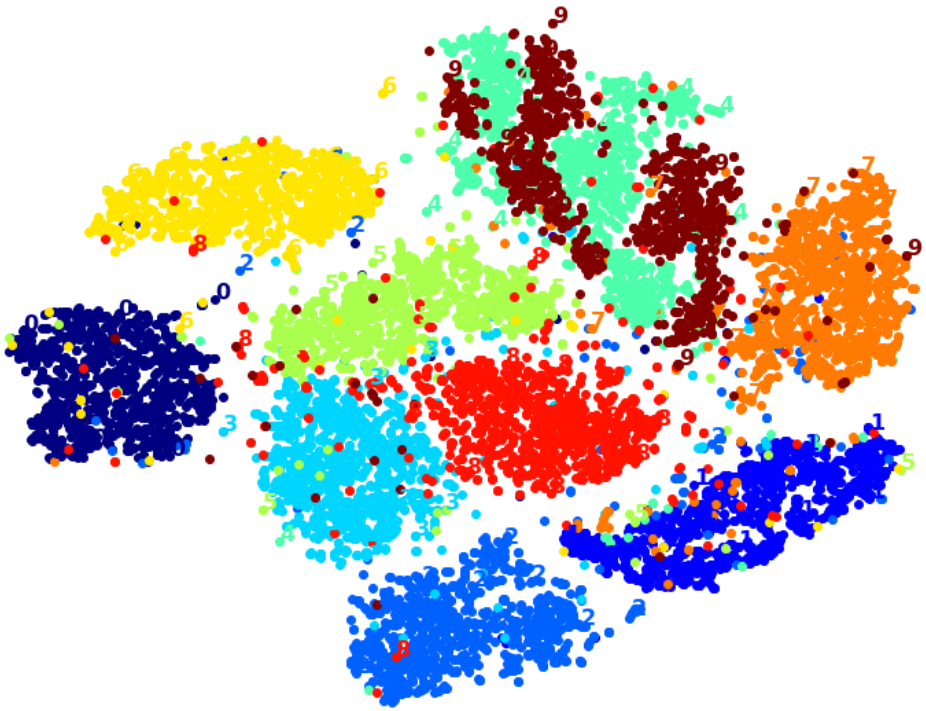
$$q_{ij} = \frac{\left(1 + \left|y_i - y_j\right|^2\right)^{-1}}{\sum_{k \neq l}\left(1 + \left|y_i - y_j\right|^2\right)^{-1}}$$

t-SNE는 **원공간에서 가까운 좌표 값**으로 만든 q와 p가 다르다면

gradient descent를 통해 **더 비슷해지는 방향으로 y의 점들을 이동**

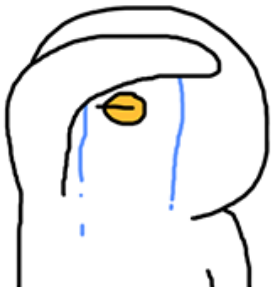
고차원 시각화

MNIST 데이터 시각화 with t-SNE



이거 하려고

지금까지 고생했습니다..



클러스터가 잘 생성되어 있을 뿐만 아니라,

유사한 숫자끼리 인접!

고차원 시각화

t-SNE

t-SNE 기반 시각화 특징

장점

PCA와 달리 군집이 중복되지 않아 시각화에 유용

전역적 특징도 잘 잡아내 클러스터링 구조를 잘 보존

단점

계산 시간이 매우 많이 걸리며, 랜덤 프로세스에 의존적

→보통 50차원 이내로 압축 뒤 t-SNE

고차원 시각화

t-SNE

t-SNE 기반 시각화 특징

장점

PCA와 달리 군집이 중복되지 않아 시각화에 유용

전역적 특징도 잘 잡아내 클러스터링 구조를 잘 보존

단점

계산 시간이 매우 많이 걸리며, **랜덤 프로세스**에 의존적

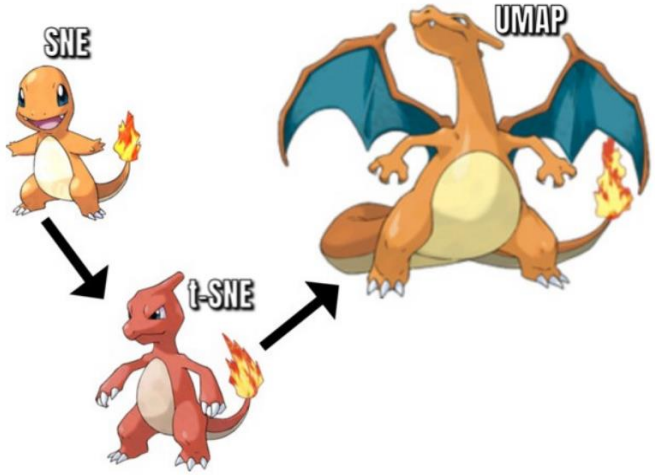
→보통 50차원 이내로 압축 뒤 t-SNE

UMAP

UMAP 소개

고차원 시각화 기법 중 가장 **최신**에 개발된 알고리즘

대규모 데이터셋에 대해 효과적



SNE, t-SNE, UMAP의 관계

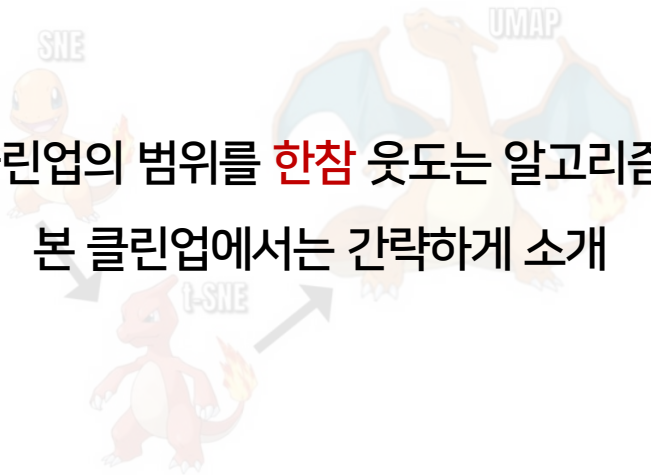
고차원 시각화

UMAP

UMAP 소개

고차원 시각화 기법 중 가장 **최신**에 개발된 알고리즘

대규모 데이터셋에 대해 효과적



SNE, t-SNE, UMAP의 관계

클린업의 범위를 한참 웃도는 알고리즘

본 클린업에서는 간략하게 소개

고차원 시각화

UMAP

UMAP 알고리즘의 직관적 이해

> **고차원 공간**에서의 데이터를 **그래프**로 만들고 저차원으로 projection

⋮

Fuzzy simplical complex

고차원 그래프를 만드는 단계에서 고안된 개념으로,

두 점이 연결될 가능도를 나타내는 **가중치가 있는 그래프** 표현

수학적 이론으로 들어가면 상당히 어려우니

직관적인 이해만 해보자!

고차원 시각화

UMAP

UMAP 알고리즘의 직관적 이해

두 포인트의 연결 여부를 판단하기 위해

각 포인트를 중심으로 원을 그리고,

원이 겹쳐지는 점들에 대해 **연결**되어 있다고 생각

원의 반지름이 커짐에 따라 그래프를 fuzzy하게 만들어

점 사이의 연결 정도를 줄임

고차원 시각화

UMAP

UMAP 알고리즘의 직관적 이해

두 포인트의 연결 여부를 판단하기 위해

각 포인트를 중심으로 원을 그리고,

원이 겹쳐지는 점들에 대해 **연결**되어 있다고 생각

원의 크기

- 원이 너무 **작으면** 크기가 작고 **고립된 클러스터**가 생성
- 원이 너무 **크면** 모든 점들이 **다 연결**되도록 만듬

N-th nearest neighbor로 해결!

고차원 시각화

UMAP

UMAP 알고리즘의 직관적 이해

두 포인트의 연결 여부를 판단하기 위해

각 포인트를 중심으로 원을 그리고,

원이 겹쳐지는 점들에 대해 **연결**되어 있다고 생각

? N-th nearest neighbor

n번째로 가까운 점까지 반지름을 확장하는 방식

데이터가 **충분**한 지역은 **반지름을 작게**,

희소한 지역은 **반지름을 크게** 설정 가능

고차원 시각화

UMAP

UMAP 알고리즘의 직관적 이해

두 포인트의 연결 여부를 판단하기 위해
각 포인트를 중심으로 원을 그리고,
원이 겹쳐지는 점들에 대해 **연결**되어 있다고 생각

원의 반지름이 커짐에 따라 그래프를 fuzzy하게 만들어
점 사이의 연결 정도를 줄임

UMAP

UMAP 알고리즘의 직관적 이해

각 포인트들이 **가장 가까운 점들과는 연결**되어야 한다고
규정함으로써 local strucutre 보존,
global structure와 밸런스를 맞추도록 함

이렇게 생성된 **고차원 그래프**를 가지고,
저차원 그래프의 레이아웃을 최대한 비슷하게 최적화하여 매핑

고차원 시각화

UMAP

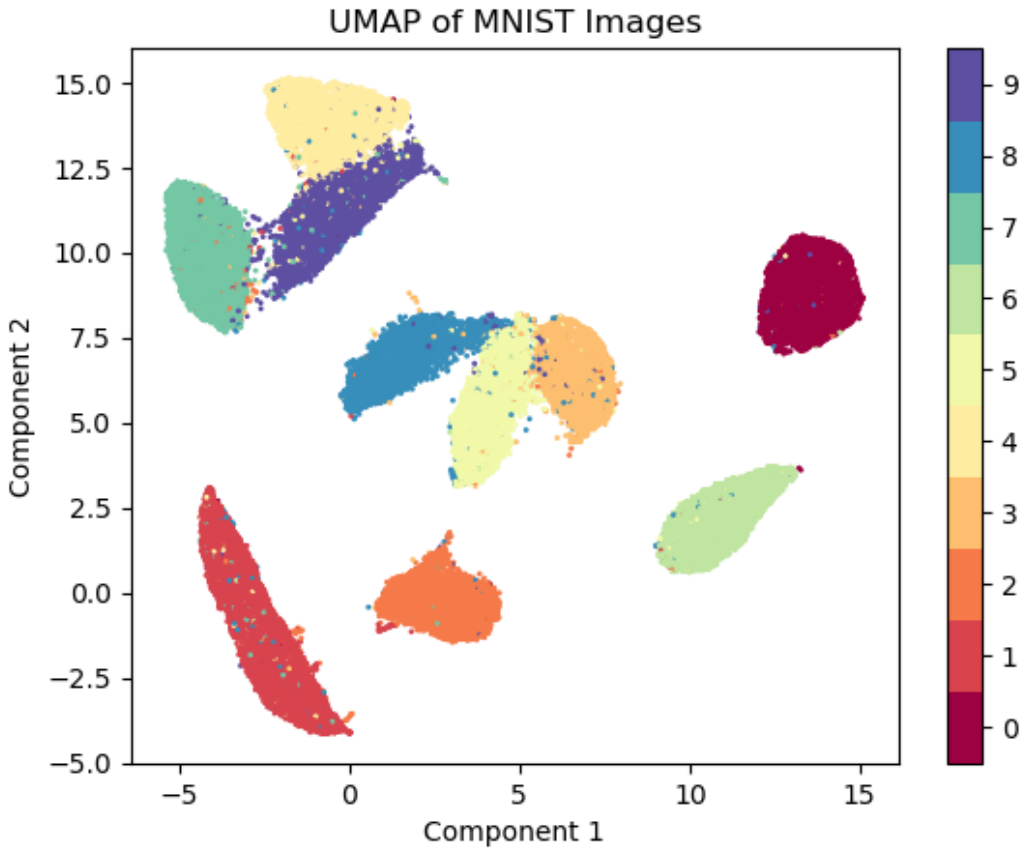
UMAP 알고리즘의 직관적 이해

각 포인트들이 **가장 가까운 점들과는 연결**되어야 한다고
규정함으로써 local strucutre 보존,
global structure와 밸런스를 맞추도록 함

이렇게 생성된 **고차원 그래프**를 가지고,
저차원 그래프의 레이아웃을 최대한 비슷하게 최적화하여 매핑

고차원 시각화

MNIST 데이터 시각화 with UMAP



2 고차원 시각화

고차원에서의 **이웃 관계**가 잘 **유지**됨

UMAP 사용 가이드

하이퍼파라미터

n_neighbors

초기 고차원 그래프 생성 시에 사용되는 nearest neighbor의 수

- 크면 global structure에 집중
- 작으면 local structure에 집중

min_dist

저차원 공간에서 포인트들 간의 최소 거리

UMAP이 얼마나 점들을 촘촘하게 묶을 지 조절

- 크면 포인트들이 상대적으로 느슨하게 퍼지게 됨
- 작으면 포인트들이 촘촘하게 무리지어 있게 됨

고차원 시각화

UMAP 사용 가이드

하이퍼파라미터

n_neighbors

초기 고차원 그래프 생성 시에 사용되는 nearest neighbor의 수

- 크면 global structure에 집중
- 작으면 local structure에 집중

min_dist

저차원 공간에서 포인트들 간의 최소 거리

UMAP이 얼마나 점들을 **촘촘하게** 묶을 지 조절

- 크면 포인트들이 상대적으로 느슨하게 퍼지게 됨
- 작으면 포인트들이 촘촘하게 무리지어 있게 됨

고차원 시각화

UMAP

장점

t-SNE보다 **더 빠른 속도**

퍼지 위상학과 리만 기하학에 기반한 **강력한 수학적 기반**

지역적(local) 정보를 넘어 **전역적(global) 정보**까지 잘 보존

단점

하이퍼파라미터의 영향을 크게 받음

정보손실에 의한 **데이터 왜곡**

→ **다양한 하이퍼 파라미터 조합**을 시도해봐야 함

고차원 시각화

UMAP

장점

t-SNE보다 **더 빠른 속도**

퍼지 위상학과 리만 기하학에 기반한 **강력한 수학적 기반**

지역적(local) 정보를 넘어 **전역적(global) 정보**까지 잘 보존

단점

하이퍼파라미터의 영향을 크게 받음

정보손실에 의한 **데이터 왜곡**

→ **다양한 하이퍼 파라미터 조합**을 시도해봐야 함

고차원 시각화

UMAP

UMAP 시연

UMAP 시연에 관심 있는 분들은 다음 링크를 들어가보세요

https://pair-code.github.io/understanding-umap/

고차원 시각화

고차원 시각화 총정리

변수들이 전반적으로 **정규분포 형태**를 띄고, **해석력**을 조금이나마 챙기고 싶다.
비교적 저차원이다. 데이터들이 선형일 것이다. 최대한 **빨라야** 한다.

PCA

데이터 포인트 간의 **거리 정보만이** 중요하다. 비교적 저차원이다.

MDS

고차원 시각화

고차원 시각화 총정리

데이터셋에 **매니폴드가 내재**하는 것 같다. 단순 유클리드 거리가 아닌,
실제 매니폴드 상의 거리 정보만을 반영하여 시각화하고 싶다.

ISOMAP

데이터셋에 매니폴드가 내재하는 것 같다.
데이터 포인트 간의 **이웃 관계를 저차원에서도 유지**하고 싶다.

LLE

고차원 시각화

고차원 시각화 총정리

고차원 데이터 시각화가 하고싶고 **시간은 많다.**

군집별로 구별이 잘 되어있으면 좋겠다.

t-SNE

용도에 맞는 전문가용 모델이 필요하다.

군집별로 구별이 잘 되어있으면 좋겠고, **시간이 적게** 들었으면 좋겠다.

UMAP

선대팀 클린업

고차원 시각화 총정리

고차원 데이터 시각화가 하고싶고 **시간은 많다.**

군집별로 구별이 잘 되어있으면 좋겠다.

t-SNE

용도에 맞는 전문가용 모델이 필요하다.

군집별로 구별이 잘 되어있으면 좋겠고, **시간이 적게** 들었으면 좋겠다.

UMAP

THANK YOU