Data Analysis Case Study (2)
- Dimensionality Reduction

ESC 2024 Summer Special Session 2차
발제자: 노희준

Contents

Contents

1

Introduction to Dimensionality Reduction

Why do we need dimensionality reduction in data analysis?

Dimensionality Reduction

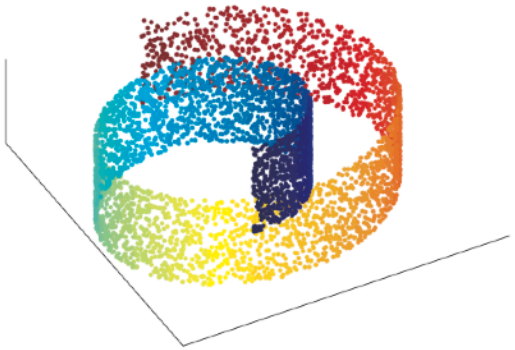
Dimension

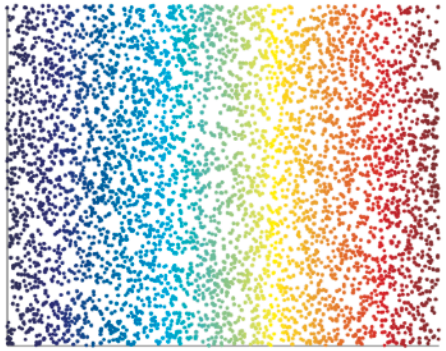
정의 데이터를 표현하는 데 사용하는 축의 개수

목적 Why do we need Dimensionality Reduction?

(1) Computational: 전처리 단계에서 보다 작은 크기로 압축함으로써 이후의 작업에서 computational cost를 줄이기 위함

(2) Visualization: 4차원 이상의 데이터를 2차원, 3차원에서 표현함으로써 시각화를 하기 위함





Dimensionality Reduction

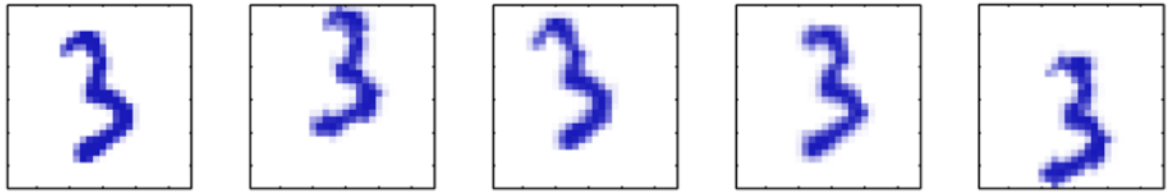
Dimension

정의 데이터를 표현하는 데 사용하는 축의 개수

목적 Why do we need Dimensionality Reduction?

(3) 보다 적은 수의 효과적인 feature만을 사용함으로써 Robust하고 설명력이 좋은 모델을 만들기 위함

e.g. Digit images consist of 28*28 pixels



• Only 3 degrees of freedom in this example: horizontal, vertical translations and rotations.

• All variabilities can be represented with 3 numbers with a nonlinear mapping:

$$[0,1]^{28\times 28} \rightarrow (t_x, t_y, r)$$

2

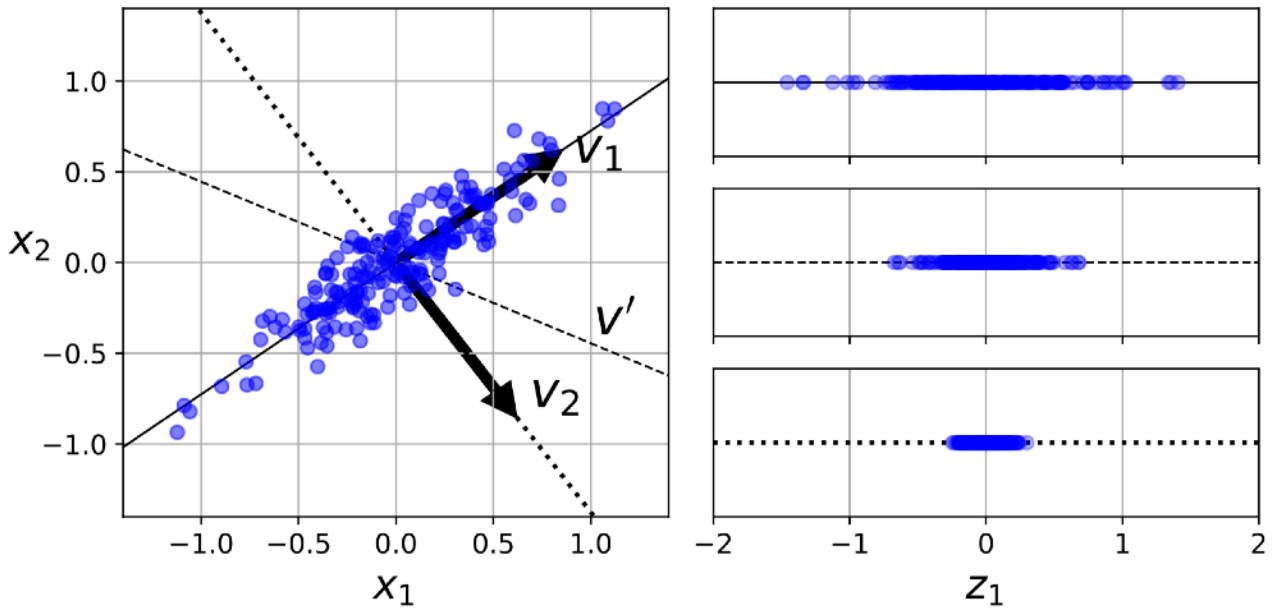
Principal Components Analysis (PCA)

Elementary method for Dimensionality Reduction

Principal Components Analysis (PCA)

PCA can be described as either:

1. Finding an optimal subspace minimizing the projection error w.r.t. L2-norm.
2. Finding orthogonal axes maximizing the variance of the projection.



Principal Components Analysis (PCA)

Let $\mathbf{X}$ be a mean-centered $N \times p$ matrix, that is $\sum_{i=1}^{N} \mathbf{x}_i = \mathbf{0}$.

Define P_k as the set of N-dimensional rank-k orthogonal projection matrices.

Objective function : $\min_{\mathbf{P} \in \mathcal{P}_k} \|\mathbf{PX} - \mathbf{X}\|_F^2$

$\mathbf{P}^*$ minimizes reconstruction error,

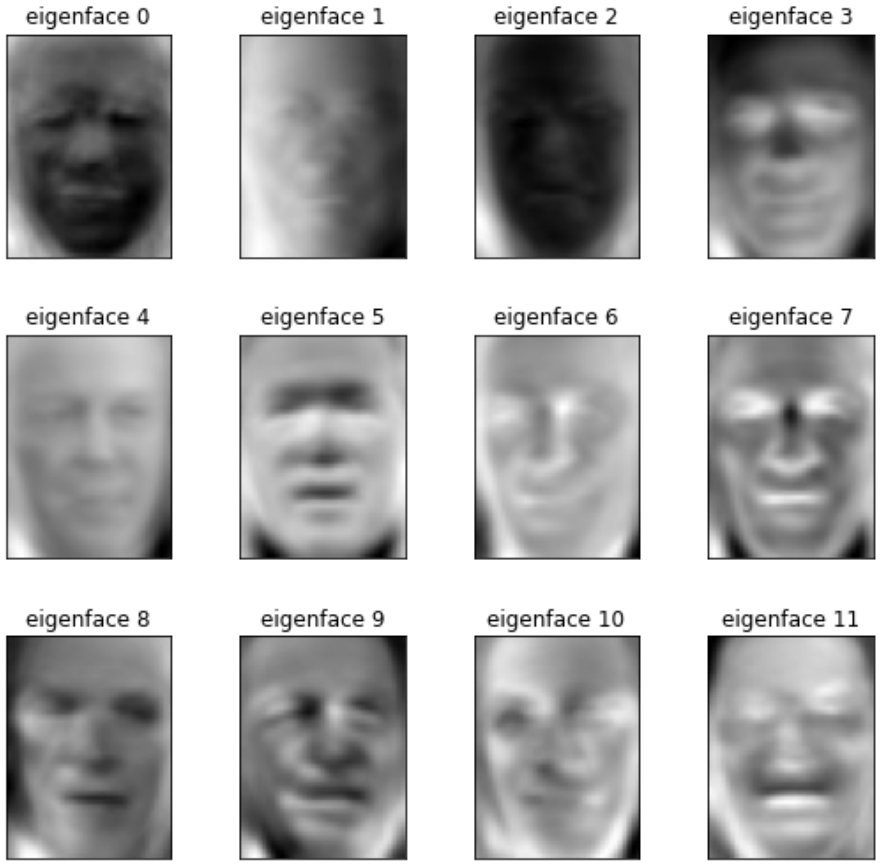
that is the sum of the squared L_2-distances between the original data and the projected data.

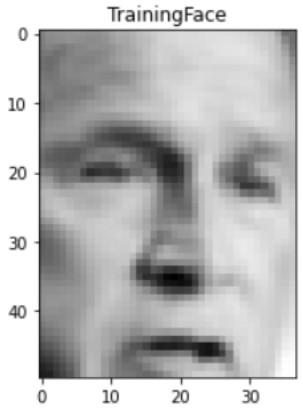
And $\mathbf{P}^* = \mathbf{V}_k \mathbf{V}_k^T$, where $\mathbf{V}_k \in \mathbb{R}^{N \times k}$ is the matrix formed by the top k singular vectors of $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$,

where $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$.

Moreover, the associated k-dimensional representation of $\mathbf{X}$ is given by $\mathbf{X}_{\text{reconstructed}} = \mathbf{V}_k^\top \mathbf{X}$

PCA Example: Eigenfaces





=

3

Low-rank Approximation via SVD

How can we compress data linearly?

Low-rank Approximation via SVD

Finding Top-k eigenvectors for XX^T

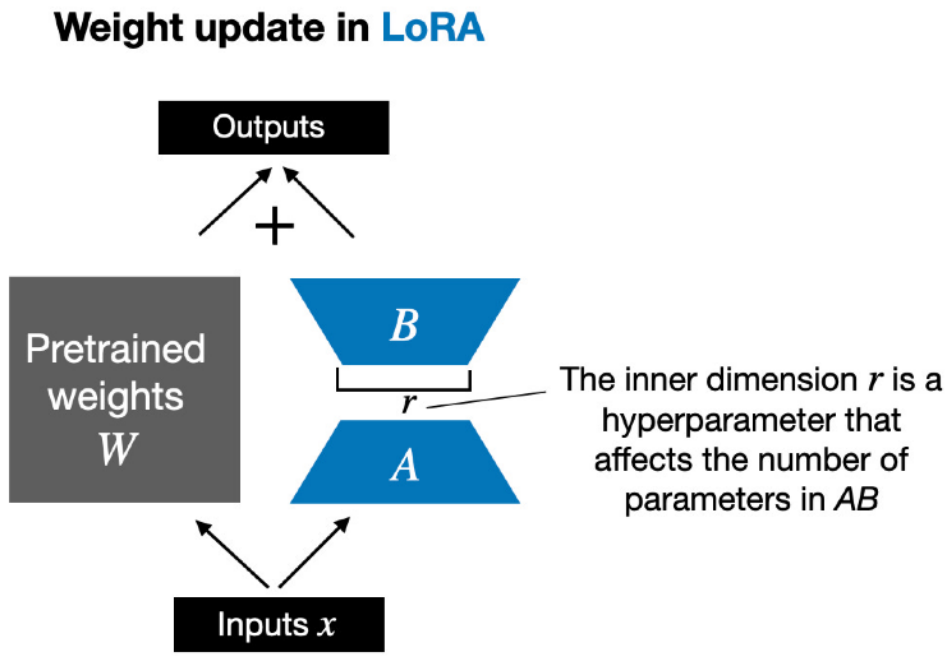
= Low-rank Approximation

$rank(\mathbf{X}) = p$라 할때, $rank(\mathbf{X}_k) = k$ where $k << p$인 $\mathbf{X}_k$로 approximation.

Low rank의 장점

1. Compression
    1. 조금의 정보로 모두 표현 가능. missing entries
2. Updating Huge ML models
    1. LoRA
    2. Weight matrix를 low rank weight matrix로
3. De-noising

    Over-fitting 방지. 오히려 Robust해짐



Low-rank Approximation via SVD

• A singular value decomposition (SVD) of an m × n matrix A expresses the matrix as the product of three “simple” matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}$$

where

(1) U is an m × m orthogonal matrix

(2) V is an n × n orthogonal matrix

(3) S is an m × n diagonal matrix with nonnegative entries, and with the diagonal entries sorted from high to low (as one goes “northwest” to “southeast).

Low-rank Approximation via SVD

- $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ is equivalent to the expression

$$\mathbf{A} = \sum_{i=1}^{\min\{m,n\}} s_i \cdot \mathbf{u}_i \mathbf{v}_i^\top$$

Low Rank Approximation!

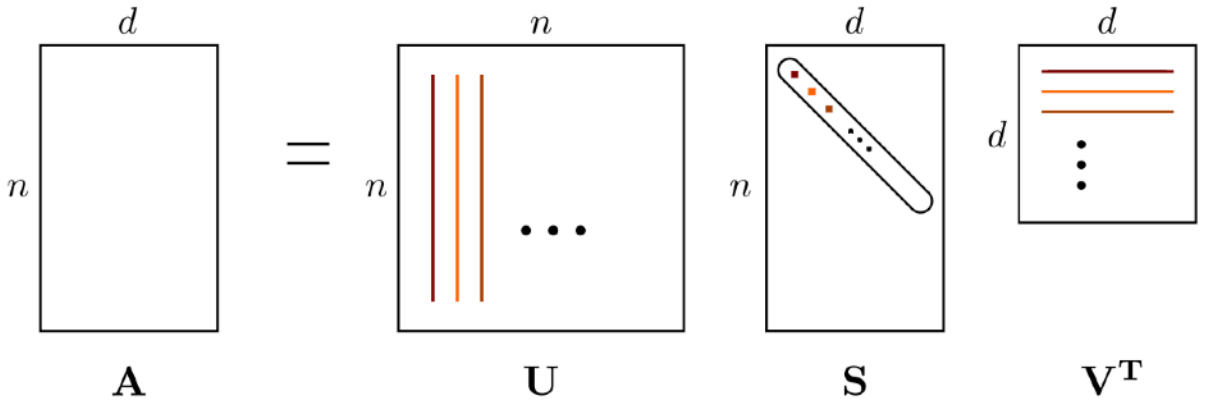


Figure 2: The singular value decomposition (SVD). Each singular value in $\mathbf{S}$ has an associated left singular vector in $\mathbf{U}$, and right singular vector in $\mathbf{V}$.

Low-rank Approximation via SVD

- The proposed rank-k approximation is

$$\hat{\mathbf{A}} = \sum_{i=1}^{k} s_i \cdot \mathbf{u}_i \mathbf{v}_i^\top$$

Low Rank Approximation!

where as usual we assume that the singular values have been sorted:

$$s_1 \geq s_2 \geq \cdots s_{\min\{m,n\}} \geq 0$$

- We re-represent A using the SVD, which provides a list of A's "ingredients," ordered by "importance," and we retain only the k most important ingredients.

Low-rank Approximation via SVD

- An equivalent way to think about the proposed rank-k approximation

1. Compute the SVD $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, $\mathbf{S}$ is a nonnegative $m \times n$ diagonal matrix with diagonal entries sorted from high to low, and $\mathbf{V}^\top$ is a $n \times n$ orthogonal matrix.

2. Keep only the top k right singular vectors: set $\mathbf{V}_k^\top$ equal to the first k rows of $\mathbf{V}^\top$ (a $k \times n$ matrix).

3. Keep only the top k left singular vectors: set $\mathbf{U}_k$ equal to the first k columns of $\mathbf{U}$ (an $m \times k$ matrix).

4. Keep only the top k singular values: set $\mathbf{S}_k$ equal to the first k rows and columns of $\mathbf{S}$ (a $k \times k$ matrix), corresponding to the k largest singular values of $\mathbf{A}$.

5. The rank-k approximation is then

$$\mathbf{A}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\top. \tag{6}$$

Low-rank Approximation via SVD

- An equivalent way to think about the proposed rank-k approximation

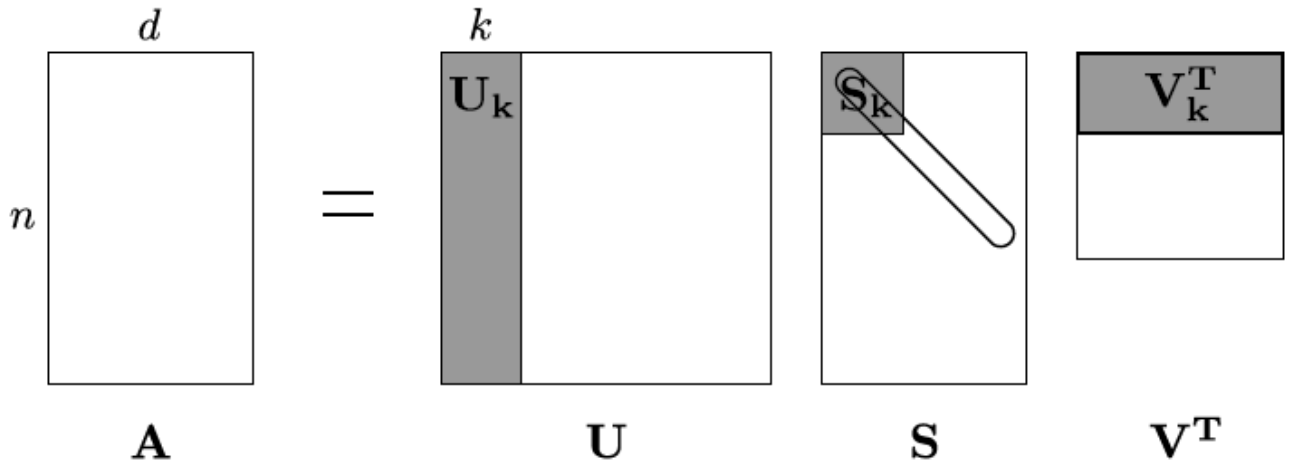


Figure 3: Low rank approximation via SVD. Recall that $\mathbf{S}$ is non-zero only on its diagonal, and the diagonal entries of S are sorted from high to low. Our low rank approximation is $\mathbf{A}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\top$.

Low-rank Approximation via SVD

- The following facts justifies this approach:

For every $m \times n$ *matrix* $\mathbf{A}$, *rank target* $k \geq 1$, *and rank-*k $m \times n$ *matrix* $\mathbf{B}$,

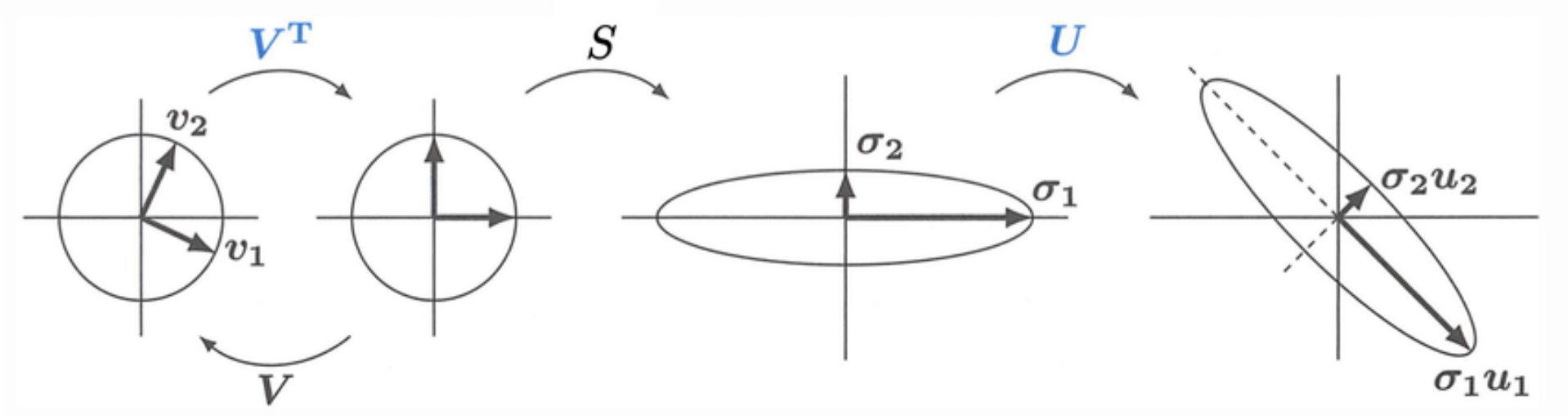
$$\|\mathbf{A} - \mathbf{A}_k\|_F \leq \|\mathbf{A} - \mathbf{B}\|_F,$$

where $\mathbf{A}_k$ *is the rank-*k *approximation* (6) *derived from the SVD of* $\mathbf{A}$.

Low-rank Approximation via SVD

Geometric view:

1. Rotation in the domain
2. Scaling + Adding or Deleting dimensions
3. Rotation in the range



PCA derivation via SVD

4

Practical PCA: randomized PCA, incremental PCA

Dealing computational issues in using PCA

Randomized PCA

Problems in high-dimensional data.

e.g. $2048 \times 3240 \times 3$ 사이즈의 컬러 이미지, 동영상, 음성 기록 등..

1. Computing $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$

2. Find top-k eigenvectors of $\mathbf{S}$

$\Longrightarrow$ 약 $O(p^3)$의 FLOP 발생, where $p = 2048 * 3240 * 3 = 19906560.$

TOO MUCH!

$\Longrightarrow$ 현재 많이 쓰이는 PCA 방법 : Randomized PCA

randomized PCA

Randomized PCA:

1. Using Truncated SVD
2. Initialize random $d \times k$ matrix Q_k
3. Repeat $Q_k = AQ_k$, Power iteration method
4. Normalizing 및 linearly independent 유지. Numerical Stability
5. Projected small matrix $B = Q_k^T A$를 SVD로 Factorization. $B = U_B \Sigma V^T$
6. 다시 A의 column space로 reconstruction. $A \approx Q_k U_B \Sigma V^T = Q \Sigma V^T$

```
for it = 1:its

  Q = A*Q;

  if(it < its)
    [Q,R] = lu(Q);
  end

  if(it == its)
    [Q,R,E] = qr(Q,0);
  end

end
```

Incremental PCA

데이터를 새로 수집하면, 매번 다시 $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$, $p \times p$ matrix를 구해서 계산해야하는가?

e.g. 쇼핑몰 고객 데이터, chatGPT 등 거의 모든 회사 업무

해결책:

한번 구해놓은 principal component에, 새로 들어온 데이터만 PCA를 적용시켜서 업데이트하자.

= Incremental PCA

Initialization을 Randomized PCA로 수행 후, Incremental PCA로 업데이트.

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

https://scikit-learn.org/0.15/modules/generated/sklearn.decomposition.RandomizedPCA.html

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html#

5

Linear Dimensionality Reduction for specific tasks

Several important methods for specific tasks

Linear Dimensionality Reduction for specific tasks

모든 데이터에 일단 PCA만 하면 될까? -> 당연히 아님.

e.g. Non-linear cases, Regression, Classification, 특수한 데이터, Orthogonality 없었으면 좋겠음

Fisher Linear Discriminant (FLD)

For classification

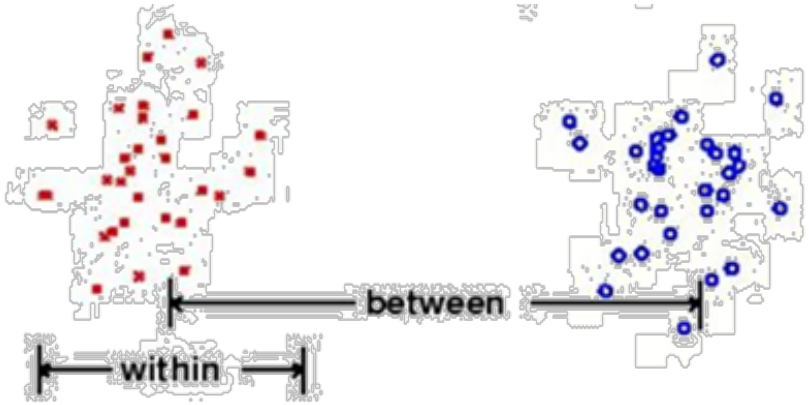
Fisher's Linear Discriminant

- Use $\mathbf{w}$ to project $\mathbf{x}$ onto one dimension, and then threshold the value by $-w_0$.

  C_1 if $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \geq -w_0$

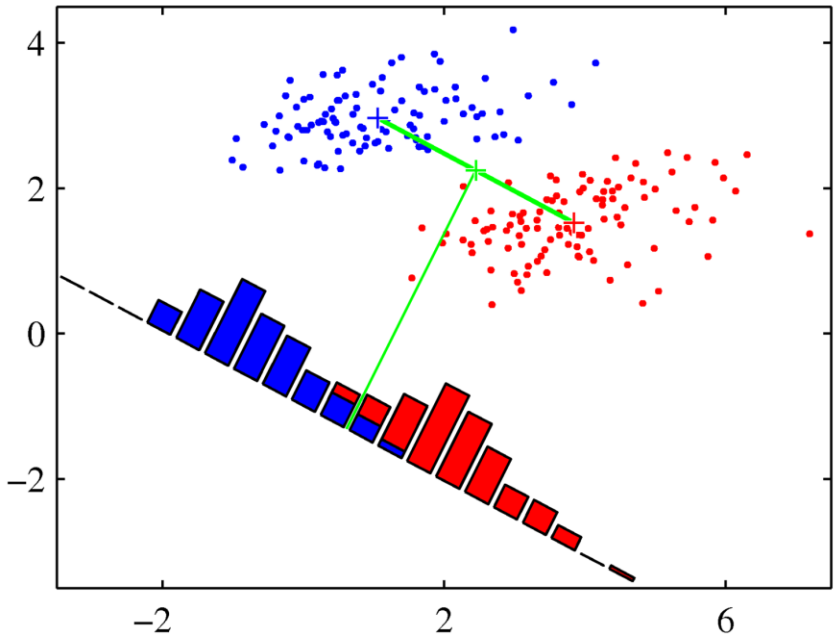
  C_0 otherwise.

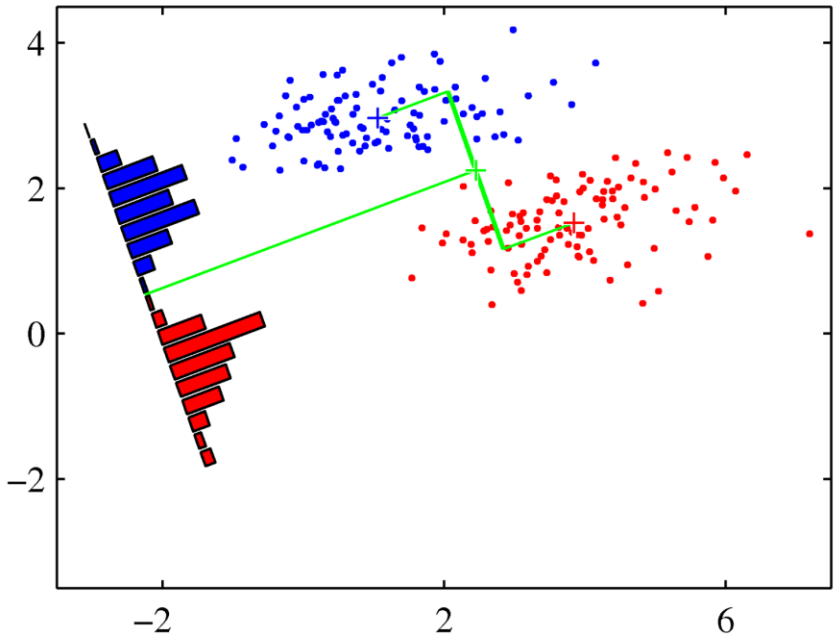
- Select $\mathbf{w}$ that best **separates** the classes.
- The learning objective should simultaneously
  - Maximize between-class variances
  - Minimize within-class variances



Fisher's Linear Discriminant

- The learning objective should simultaneously
  - Maximize between-class variances
  - Minimize within-class variances
- Because maximizing separation alone doesn't work.
  - Minimizing class variance is a big help.





FLD: Formulation

- We want to maximize the distance between classes

$$m_2 - m_1 \equiv \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) \qquad \text{where } \mathbf{m}_k = \frac{1}{N_k}\sum_{n \in C_k} \mathbf{x}_n$$

Projected mean Mean

- While minimizing the distance within each class

$$s_1^2 + s_2^2 \equiv \sum_{n \in C_1} (\mathbf{w}^T\mathbf{x}_n - m_1)^2 + \sum_{n \in C_2} (\mathbf{w}^T\mathbf{x}_n - m_2)^2$$

- Objective function: $J(\mathbf{w}) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$

  - The bias w_0 is not in the learning objective; it should be determined separately after finding $\mathbf{w}$.

FLD: Derivation

- Numerator: $m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$
  $\Rightarrow \|m_2 - m_1\|^2 = \mathbf{w}^T\underline{(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T}\mathbf{w}$

  S_B: Between-class scatter

- Denominator:

$$s_k^2 = \sum_{n \in C_k} (\mathbf{w}^T\mathbf{x}_n - \mathbf{w}^T\mathbf{m}_k)^2 = \sum_{n \in C_k} \mathbf{w}^T(\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{w}$$

$$\Rightarrow s_1^2 + s_2^2 = \mathbf{w}^T\underline{\left[\sum_{k=1}^{2}\sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T\right]}\mathbf{w}$$

S_W: Within-class scatter

- Objective function:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- Solution: $\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Generalized FLD

- Two-class
  - Learning objective: $J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$
  - Solution: $\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

- Multi-class
  - Learning objective: $J(W) = \mathrm{tr}(W^T S_W W)^{-1}(W^T S_B W)$
  - Solution: $S_B W = S_W W \Lambda$
    - where $S_W = \sum_k \sum_{y^{(i)}=k} (\mathbf{x}^{(i)} - \mathbf{m}_k)(\mathbf{x}^{(i)} - \mathbf{m}_k)^T$

$$S_B = \sum_k N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

$$\mathbf{m} = \frac{1}{N} \sum_i x^{(i)} = \frac{1}{N} \sum_k N_k \mathbf{m}_k$$

Generalized FLD

- Multi-class
  - Learning objective: $J(W) = \mathrm{tr}(W^T S_W W)^{-1}(W^T S_B W)$
  - Solution: $S_B W = S_W W \Lambda$
    - where $S_W = \sum_k \sum_{y^{(i)}=k} (\mathbf{x}^{(i)} - \mathbf{m}_k)(\mathbf{x}^{(i)} - \mathbf{m}_k)^T$

$$S_B = \sum_k N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

$$\mathbf{m} = \frac{1}{N} \sum_i x^{(i)} = \frac{1}{N} \sum_k N_k \mathbf{m}_k$$

    - Derived by generalized eigenvalue decomposition
      - Not directly usable for classification, as each C_k does not get its own function; need to build another classifier
      - Up to $(K-1)$ linearly independent $\mathbf{w}_k$'s ($\mathrm{rank}(W) \leq K-1$)
      - Not unique: WZ is also a solution for nonsingular Z

Generalized FLD

- Multi-class
  - Learning objective: $J(W) = \text{tr}(W^T S_T W)^{-1}(W^T S_B W)$
  - Solution: $S_B W = S_T W \Lambda$
    - where $S_T = \sum_i (\mathbf{x}^{(i)} - \mathbf{m})(\mathbf{x}^{(i)} - \mathbf{m})^T = S_W + S_B$

$$S_B = \sum_k N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

$$\mathbf{m} = \frac{1}{N} \sum_i x^{(i)} = \frac{1}{N} \sum_k N_k \mathbf{m}_k$$

  - This is equivalent to the within-scatter version.
    - Proof sketch: Intuitively,

$$\text{argmax}_W \frac{f(W)}{g(W)} = \text{argmax}_W \frac{f(W)}{f(W) + g(W)}$$

Generalized FLD

- Often called linear discriminant analysis (LDA)
  - Or Fisher's LDA (FLDA)
  - Cf. For two-class classification, **GDA with shared covariance** gives us the same weight vector $\mathbf{w}$, and is also often called LDA.

- Mainly used for **dimensionality reduction**
  - From an arbitrary D to D' ($\leq K-1$)
    - Choose D' eigenvectors with largest eigenvalues
  - This is supervised dimensionality reduction; we will cover other (un)supervised dimensionality reduction methods later.

Recap: GDA with shared covariance

- Maximum likelihood estimation (MLE):

$$\phi = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 0\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 0\}}$$

$\mathbf{m}_2$ in FLD formulation

$$\mu_1 = \frac{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}\mathbf{x}^{(\mathrm{i})}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}}$$

$\mathbf{m}_1$ in FLD formulation

$$\sum = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \mu_{y^{(i)}})(\mathbf{x}^{(i)} - \mu_{y^{(i)}})^T$$

$N \cdot S_W$ in FLD formulation

Generalized FLD (Out of Scope)

- Variations of LDA
  - Regularized LDA: $S_T \leftarrow S_T + \lambda I$
  - Uncorrelated LDA: $W^T S_T W = I$
  - Orthogonal LDA: $W \leftarrow U$ where $W = U\Sigma V^T$ is SVD
  - Kernel LDA: $\mathbf{x} \leftarrow \phi(\mathbf{x})$ and use kernel trick
    - (We will discuss the kernel method later)
  - Combination of some of the above

Generalized FLD (Out of Scope)

- Equivalent to **least squares** from **centered data** to **a family of targets** [Lee et al., 2015]
  - if and only if $\text{rank}(S_B) = K - 1$
  - i.e., their solutions map data to the same latent space
    - $W^T(X - \mathbf{m}\mathbf{1}^T) \approx Y \Leftrightarrow \underset{W}{\text{argmax}}\, \text{tr}(W^T S_T W)^{-1}(W^T S_B W)$
  - A possible $Y = [I_{K-1}\ \mathbf{0}_K]L \in \mathbb{R}^{(K-1)\times N}$
    - $L \in \mathbb{R}^{K\times N}$ is a one-hot encoded label matrix
    - A collection of $(K-1)$ one-hot vectors and a zero vector
  - A possible $Y = B \in \mathbb{R}^{(K-1)\times N}$ where $S_B = XB^T BX^T$
    - $X \in \mathbb{R}^{D\times N}$ is the data matrix and B is a full rank factorization

[Lee et al.] On the Equivalence of Linear Discriminant Analysis and Least Squares. In *AAAI*, 2015.

Factor Analysis (FA)

by GPT-4o ^^;;

Factor Analysis (FA)는 관측된 다변량 데이터를 설명하기 위해 잠재 변수(잠재 요인)를 사용하는 통계 기법입니다. 이 방법은 주로 심리학, 사회학, 경제학 등에서 사용되며, 변수 간의 관계를 이해하고 데이터의 구조를 간단히 표현하는 데 도움을 줍니다.

주요 개념 및 절차

1. **목적**:
   - 데이터의 상관 구조를 설명하는 잠재 요인(factors)을 식별.
   - 변수 간의 관계를 이해하고, 데이터의 차원을 축소하여 분석을 용이하게 함.
2. **모델**:
   - 수식: $\mathbf{X} = \mathbf{LF} + \epsilon$
     - $\mathbf{X}$: 관측된 변수들의 벡터 (n 차원).
     - $\mathbf{L}$: 로딩 행렬(loading matrix) (n x m 크기), 변수와 요인 간의 관계를 나타냄.
     - $\mathbf{F}$: 잠재 요인들(factors)의 벡터 (m 차원, m < n).
     - ϵ: 잔차(오차) 벡터, 독립적이고 동일한 분포를 따름.
3. **추정 방법**:
   - **최대우도법 (Maximum Likelihood)**: 요인과 로딩 행렬을 추정하는 가장 일반적인 방법.
   - **주축법 (Principal Axis Factoring)**: 공분산 행렬의 고유값 분해를 통해 요인을 추정.

예제

예를 들어, 심리학 연구에서 여러 심리적 특성(예: 스트레스, 행복, 불안 등)을 측정하는 설문 조사가 있다고 가정합니다. Factor Analysis를 사용하여 이들 측정값이 몇 가지 잠재적인 심리적 요인(예: 정신적 안정성, 외향성 등)에 의해 설명될 수 있는지 분석할 수 있습니다.

장점

- 데이터의 복잡성을 줄이고, 요약된 정보 제공.
- 변수 간의 잠재적인 관계를 이해하는 데 도움.
- 차원 축소를 통해 시각화 및 해석 용이.

단점

- 요인 수 선택의 주관성.
- 데이터가 정규 분포를 따른다는 가정이 필요.
- 과적합(overfitting) 가능성.

Factor Analysis는 데이터를 요약하고 설명하는 강력한 도구이지만, 적절한 가정과 해석이 필요합니다.

Factor Analysis (FA)

Generalized ver. of PCA

장점

1. Objective function 변경
2. EM for FA/PCA
3. Some variations PPCA, ···

CCA, ICA, MDS (1)

• Canonical Correlation Analysis (CCS)

목적 두 개의 데이터셋 간의 상관관계를 최대화하는 선형 변환을 찾기

적용 분야 멀티 뷰 데이터 분석, 예를 들어 텍스트와 이미지의 상관관계 분석

• Independent Component Analysis (ICA)

목적 혼합 신호에서 원래의 독립적인 소스를 추출(복원)

적용 분야

(1) 신호 처리: EEG, MEG 등 생물학적 신호의 잡음 제거 및 신호 분리

(2) 이미지 처리: 혼합된 이미지 데이터에서 원본 이미지를 복원

(3) 금융 데이터 분석: 여러 금융 지표의 혼합 신호에서 개별적인 요인을 분리

CCA, ICA, MDS (2)

• Multidimensional Scaling (MDS)

목적 데이터의 유사성을 유지하면서 저차원 공간에 매핑

적용 분야 데이터 시각화, 심리학, 사회과학 등

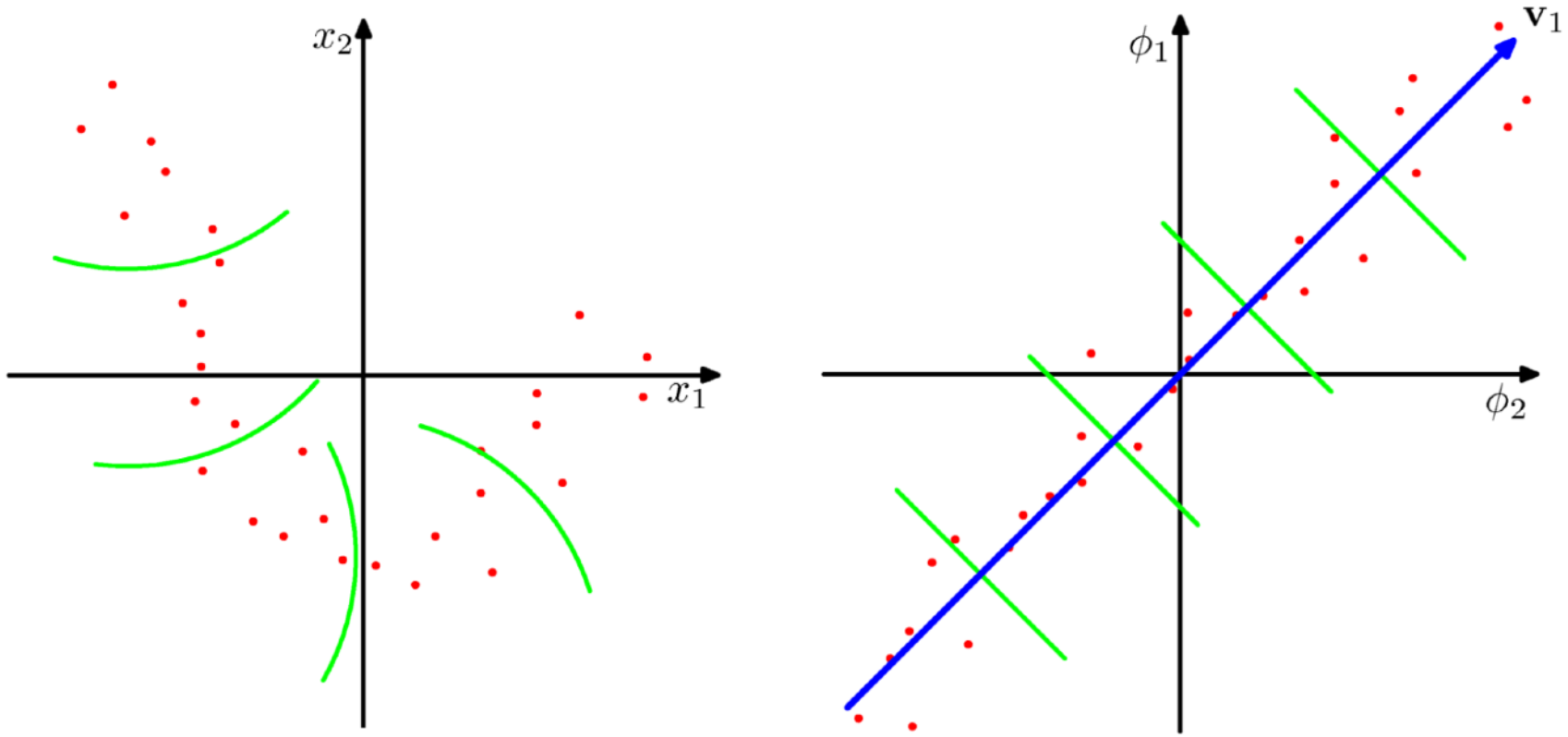
6

Non-Linear Dimensionality Reduction

Adapting superior representation by using non-linear methods

Kernel PCA (1)

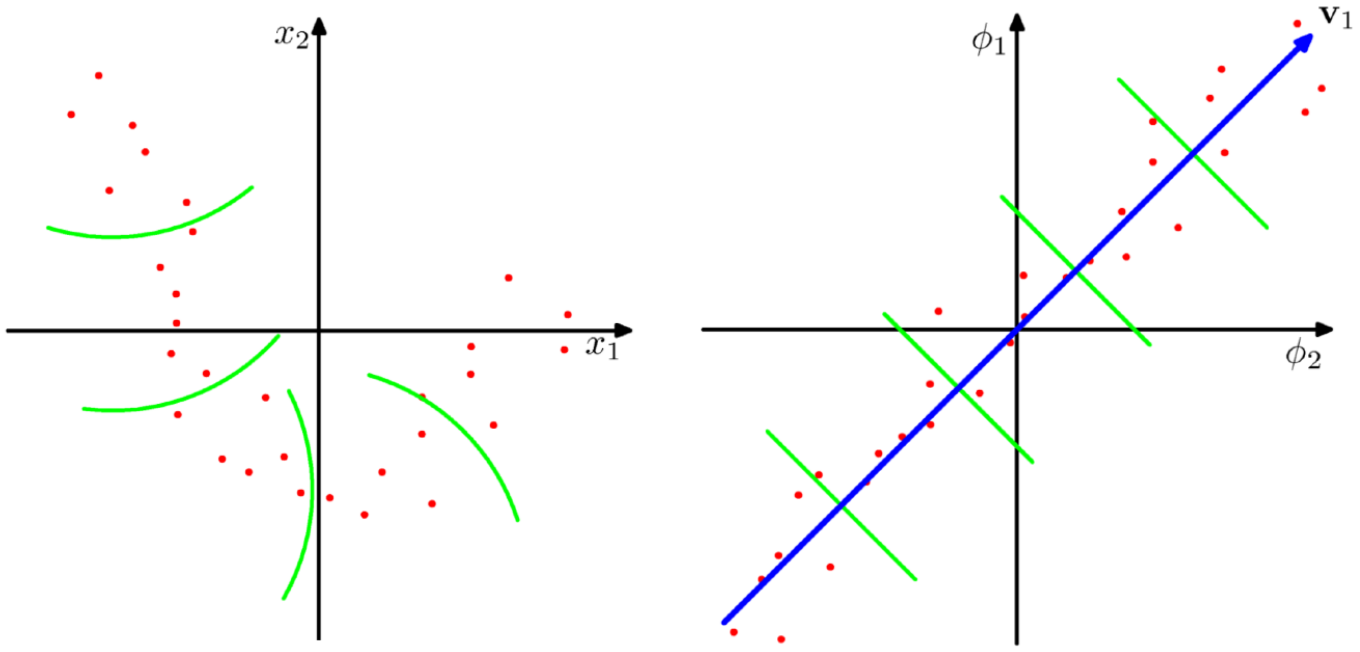
- Suppose the regularity that allows dimensionality reduction is nonlinear.



Kernel PCA (2)

- (Linear) PCA can be performed on a feature space: $\left\{\mathbf{x}^{(n)}\right\}_{n=1}^{N} \rightarrow \left\{\phi\left(\mathbf{x}^{(n)}\right)\right\}_{n=1}^{N}$
- Together with a nonlinear feature mapping ϕ,

PCA performs a nonlinear dimensionality reduction.



Kernel PCA (3)

- Define a kernel to avoid having to evaluate the feature vectors explicitly

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

- PCA can be expressed in terms of the kernel.
  - Need centering the kernel matrix (to zero mean)

$$K' = K - LK - KL + LKL$$

  - Where $L = \frac{1}{N} 1_N 1_N^T$ is the $N \times N$ matrix with all $1/N$

- Solved by eigenvalue decomposition
- FLD can be kernelized in a similar way.

Auto-Encoder, t-SNE, UMAP

https://projector.tensorflow.org/

https://distill.pub/2016/misread-tsne/

https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

설명:

https://velog.io/@swan9405/머신러닝-T-SNE-T-distributed-Stochastic-Neighbor-Embedding

https://m.blog.naver.com/myohyun/222421460444

감사합니다