고급 소프트웨어 실습(CSE4152)

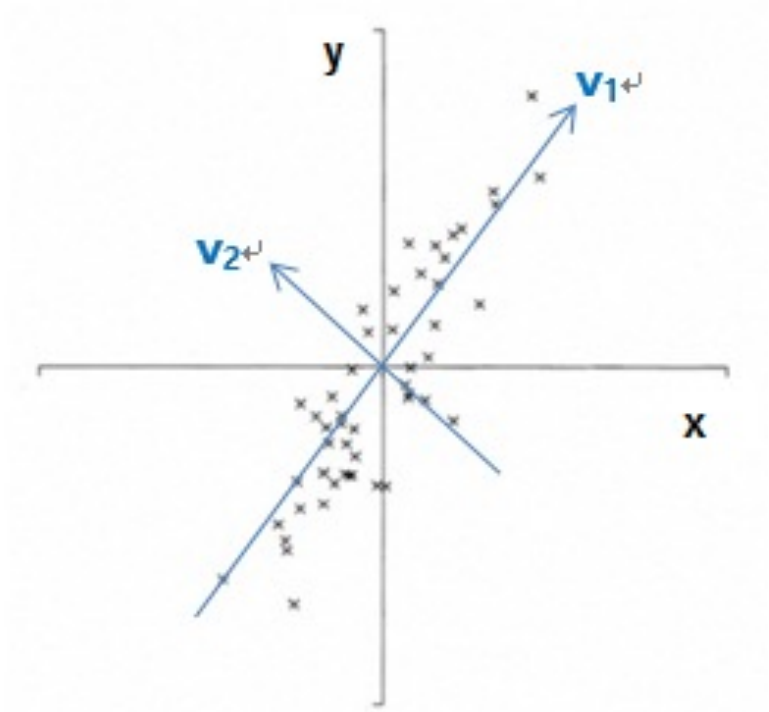
Week 3
Principal Component Analysis

목차

서론

- Principal Component Analysis(PCA, 주성분 분석)는 데이터 집합을 분석하는 기법 가운데 하나
- 분산 값이 큰 방향으로 서로 독립된 좌표 계를 구하여 원래의 데이터를 변환
- PCA를 통해 데이터의 특성 파악 가능
- Karhunen-Loeve Transform (KLT), Hotel transform, 또는 Proper Orthogonal Decomposition (POD)이라고 부르기도 한다.
- PCA는 eigenvalue decomposition 또는 singular value decomposition으로 구할 수 있음
- 본 실습에서는 covariance matrix의 eigenvalue decomposition을 통한 PCA 기법을 알아보도록 한다.

Principal Component Analysis



$$\begin{pmatrix} v_{11}x_1 + v_{12}y_1 & v_{11}x_2 + v_{12}y_2 & \cdots \\ v_{21}x_1 + v_{22}y_1 & v_{21}x_2 + v_{22}y_2 & \cdots \end{pmatrix}$$

$$= \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \times \begin{pmatrix} x_1 & x_2 & x_3 & \cdots \\ y_1 & y_2 & y_3 & \cdots \end{pmatrix}$$

그림 1. PCA의 도식적 표현과 행렬식

- PCA를 통해 v1과 v2를 구할 수 있음
- v1은 원래 데이터의 분산 값이 최대가 되는 방향
- v1과 v2를 새로운 좌표축으로 보고, 원래 데이터 점들을 회전한 것으로 해석 가능
- 새로운 좌표 계에서는 데이터의 분산 값이 최대인 방향이 각 좌표 축과 일치하게 됨

Principal Component Analysis

Covariance matrix의 Eigenvalue decomposition 방법을 통하여 PCA를 수행하는 방법은 아래와 같이 설명할 수 있다.

1. Covariance (공분산) matrix 계산

2. Eigenvalue decomposition

3. Eigenvector 정렬

4. Eigenvector 투영

Principal Component Analysis
- Covariance (공분산) matrix 계산

- Covariance는 2개의 확률변수의 상관 관계를 나타내는 값

- 하나의 변수 값이 증가할 때 다른 값도 증가하는 경우 Covariance는 양수 값을 가지고, 반대의 경우 Covariance는 음수 값을 가짐

- 그림 1에서와 같은 점 p가 x, y의 2차원 데이터($p_i=(x_i\ y_i)^T$)라고 하면 그 Covariance matrix는 식 (1)과 같이 구할 수 있음

$$\rho = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\Sigma_{xy} = Cov(x,y) = \mathrm{E}[(x - \mu_x)(y - \mu_y)]$$

$$\sum \mathrm{E}[(\rho - \mu)(\rho - \mu)^T], \mu = \frac{1}{n}\sum_{i=1}^{n} \rho_i$$

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{xy} & \Sigma_{yy} \end{pmatrix} = \mathrm{E}[(\rho - \mu)(\rho - \mu)^T]$$

식 1. Covariance matrix의 표현 식

Principal Component Analysis
- Covariance (공분산) matrix 계산

- 그림 1에 보인 점들의 집합 P=(p1, p2, …, pn)

- Covariance는 식 (2)처럼 n개의 샘플 점들에 대해 각각 구하여 더해 주는 방법으로 계산 가능

- 다른 방법으로는 전체 점들의 집합을 하나의 매트릭스 P로 놓고 식 (3)과 같이 간단히 표현 가능

- 두 방법 모두 전체 계산량은 비슷

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(p_i - \mu)(p_i - \mu)^t$$

식 2. n개의 샘플에 대한 Covariance matrix의 표현 식

$$P = (p_1 \quad p_2 \quad p_3 \quad \cdots \quad p_n) = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{d1} & \cdots & x_{dn} \end{pmatrix}$$

$$U = (p_1 - \mu \quad p_2 - \mu \quad p_2 - \mu \quad \cdots \quad p_n - \mu)$$

$$\Sigma = UU^T$$

식 3. n개의 샘플에 대한 Covariance matrix의 표현 식 (매트릭스 이용)

Principal Component Analysis
- Eigenvalue decomposition

- Covariance matrix를 구한 뒤 Eigenvalue decomposition을 이용하여 각 차원의 분산 값과 분산 방향을 결정
- 하나의 대칭형 matrix Σ 에 대하여 eigenvalue λ와 eigenvector v는 식 (4)와 같이 표현
- 전체 eigenvalue와 eigenvector는 각각 Λ, V와 같이 나타낼 수 있음
- 실습에서는 numpy에서 제공하는 함수를 이용하여 eigenvalue와 eigenvector를 구한다

$$\Sigma V = \Lambda V$$

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

$$V = \begin{pmatrix} v_1 & v_2 & \cdots & v_d \end{pmatrix} = \begin{pmatrix} v_{11} & v_{21} & \cdots & v_{d1} \\ v_{12} & v_{22} & \cdots & v_{d2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1d} & v_{2d} & \cdots & v_{dd} \end{pmatrix}$$

식 4. Eigenvalue equation

Principal Component Analysis
- Eigenvector 정렬 & 투영

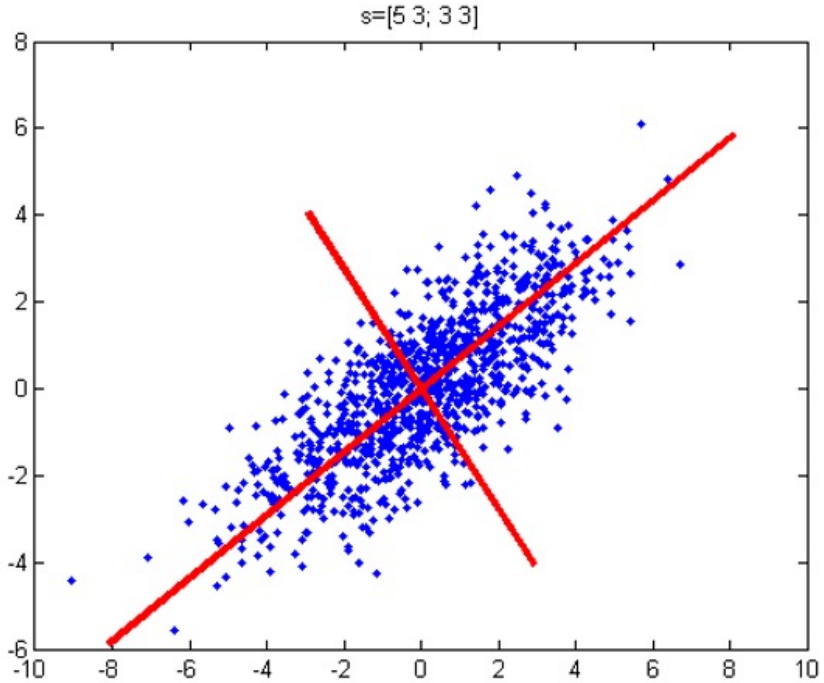
- 각 샘플 차원의 분산량은 eigenvalue와 비례
- Eigenvalue를 크기 순으로 내림차순으로 정렬하고 각 eigenvector를 eigenvalue가 정렬된 방식으로 정렬하면, 분산의 크기에 따라 정렬된 eigenvector를 구할 수 있다.
- 최종적으로 원래 데이터를 eigenvector로 투영시킴으로써 PCA를 완성하게 된다.
- 투영시키는 방법은 식 (5)와 같이 행렬식으로 간단히 표현
- 여기서 Q는 PCA에 의해서 변환(Eigenvector에 투영)된 데이터 값을 나타낸다.
- Eigenvector의 개수를 조정하여 차원 축소를 할 수 있다.

$$Q = V^T U$$

식 5. Eigenvector에 투영

Principal Component Analysis
- Eigenvector 정렬 & 투영

- 그림 2는 2차원의 실제 데이터를 가지고 PCA를 수행한 예를 보여준다. 여기서는 eigenvector를 그림에 빨간 색 선으로 그려서 표현했으며 원래 데이터 점들을 투영시키지는 않았다.



Mu = [2, 1]

Covariance matrix = $\begin{bmatrix} 5 & 3 \\ 3 & 3 \end{bmatrix}$

Eigenvector : $\begin{bmatrix} 0.5863 & -0.8101 \\ -0.8101 & -0.5863 \end{bmatrix}$

Eigenvalue : $\begin{bmatrix} 834.4 & 0 \\ 0 & 6975.3 \end{bmatrix}$

그림 2. 2차원 데이터의 PCA 예제.

응용 분야

- 데이터의 통계적 분석

- 영상 처리 분야에서 다루기 힘든 고차원 데이터의 압축
  - 저장공간 절약, 연산량 감소
  - 초기 얼굴 인식 연구에 많이 사용됨

실습: PCA 구현

- 이번 실습에서는 Python 환경에서 P=(p1, p2, …, pn)으로 주어진 데이터를 읽어들여 PCA를 수행하는 프로그램을 작성해본다.
- 아래 단계들을 완성하면, 입력 값의 분산이 가장 큰 방향을 나타내는 eigenvector에 투영된 값들을 좌표계에 그리게 된다.

실습: PCA 구현

- **Covariance matrix (공분산 행렬) 생성**
  - 식 (2) 또는 (3)을 이용하여 입력된 데이터의 Covariance matrix를 구한다. n 개의 샘플의 평균값을 빼주는 것에 주의한다.
- **Covariance matrix의 eigenvalue와 eigenvector를 구하고 eigenvalue 의 크기로 정렬하기**
  - numpy에서 제공하는 함수를 사용하여 Covariance matrix로부터 eigenvalue와 eigenvector를 구한다. 구해진 eigenvalue를 내림차순으로 정렬하고, eigenvector를 대응하는 eigenvalue의 순으로 정렬한다.
- **원래 데이터를 eigenvector에 투영시키기**
  - 공식 (5)와 같이 원래 데이터를 eigenvector에 투영시킨다.
- **투영된 데이터를 이차원 공간에 표현하기**
  - 최고 분산 방향을 나타내는 앞의 두 eigenvector에 투영된 데이터 값을 이차원 공간에 그려본다.

t-SNE (t-distribution Stochastic Neighbor Embedding)

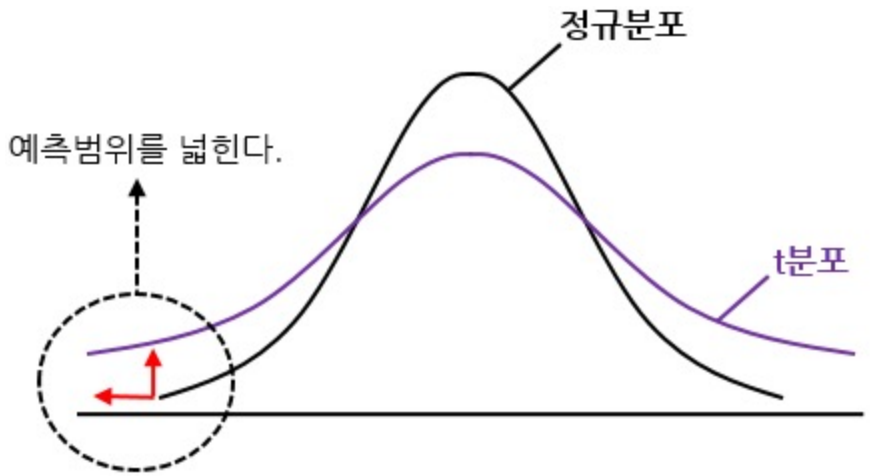
- Stochastic Neighbor Embedding(SNE) 란 고차원 공간에 존재하는 데이터 x의 이웃 간의 거리를 최대한 보존하는 저차원의 y 를 학습하는 방법론이다.
- $p_{j|i} = \frac{e^{-\frac{|x_i - x_j|^2}{2\sigma_i^2}}}{\sum_k e^{-\frac{|x_i - x_k|^2}{2\sigma_i^2}}}$: 고차원 공간에 존재하는 i 번째 개채 x_i 가 주어졌을 때 j 번째 이웃인 x_j 가 선택될 확률
- $q_{j|i} = \frac{e^{-|y_i - y_j|^2}}{\sum_k e^{-|y_i - y_k|^2}}$: 저차원에 임베딩된 i 번째 개체 y 가 주어졌을 때 j 번째 이웃인 y_j 가 선택될 확률

t-SNE (t-distribution Stochastic Neighbor Embedding)

- 두 확률분포가 얼마나 비슷한지 측정하는 지표 : Kullback-Leibler divergence.
- SNE 는 아래의 비용 함수를 최소화하는 방향으로 학습을 진행함
- $Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
- 실험상 σ 를 고정된 값을 써도 성능에 차이를 보이지 않음.
- 실험 상 i 번째 개체가 주어졌을때 j 번째 개체가 뽑힐 확률과 j 번째 개체가 주어졌을 때 i 번째 개체가 선택될 확률이 동일하다고 놓고 풀어도 성능이 비슷함.
- $\frac{\partial C}{\partial y_i} = 4 \sum_j (y_j - y_i)(p_{ij} - q_{ij})$

t-SNE (t-distribution Stochastic Neighbor Embedding)

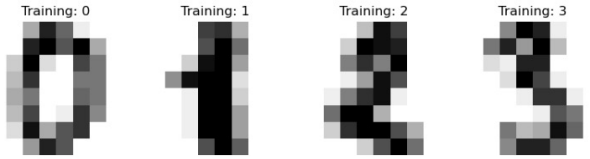
- SNE 가 전제하는 확률은 가우시안 분포이다.
- 가우시안 분포는 꼬리가 두텁지 않아서 i번째 객체에서 적당히 떨어져있는 이웃 j 와 많이 떨어져있는 이웃 k 가 선택될 확률이 크게 차이가 나지 않는다.
- 이들 구분을 잘 하기위해 가우시안 분포보다는 꼬리가 두터운 t 분포를 이용한다.



t-SNE (t-distribution Stochastic Neighbor Embedding)

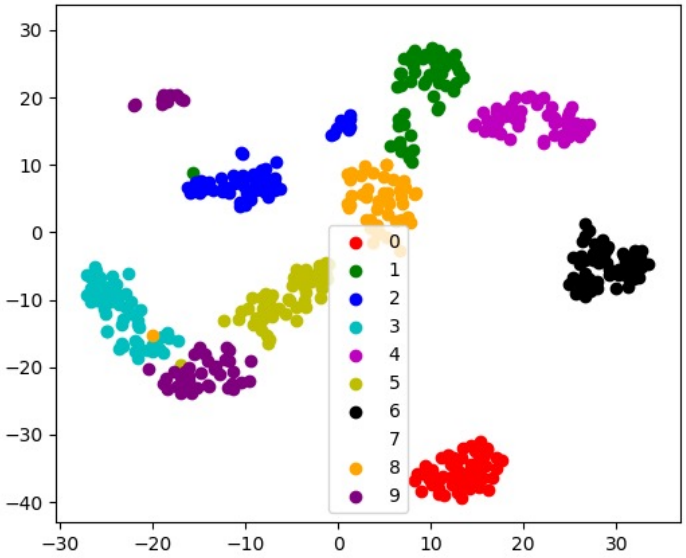
- **t-SNE 는 q_{ij}**에만 아래와 같이 t분포를 적용하고 p_{ij}는 SNE 와 동일
- $q_{ij} = \frac{(1+|y_i - y_j|^2)^{-1}}{\sum_{k \neq l}(1+|y_k - y_l|^2)^{-1}}$

실습: t-SNE 이용



- Digits 데이터는 0~9 의 숫자의 이미지를 64차원으로 변환한 데이터이다.
- Digits 데이터를 이용하여 t-SNE 한 결과물을 보이시오.

결과 예시



과제: PCA reconstruction

- PCA 로 차원축소한 데이터를 다시 원래의 데이터로 복원할 수 있다.
- $\hat{P} = V \times Q + \mu$
- $\hat{P}$: 복원된 데이터
- Q: PCA 로 투영된 데이터
- V: Eigenvector
- μ: 원 데이터의 평균

과제: PCA reconstruction

- Digits 데이터를 2차원, 3차원, 4차원, 32차원으로 축소한 결과를 복원했을 때 원래 데이터와의 MSE(Mean Square Error) 을 확인 및 복원데이터를 시각화하고 오차율이 다른 이유를 분석하여 보고서를 제출하시오. (코드, 분석내용 첨부)

- 제출형식: {분반}_{학번}_{이름}.{확장자} (ex: 1_20210928_홍길동.ipynb)
- 제출파일: 보고서({분반}_{학번}_{이름}.pdf), 주피터노트북 파일({분반}_{학번}_{이름}.ipynb)
- 제출기한: 일주일 (사이버캠퍼스 참고)