Week 7. Principal Component Analysis

실습 목표: Principal Component Analysis(PCA, 주성분 분석)를 이용하여 이미지 데이터를 압축하여 처리하는 방법을 알아보고, 이에 관한 프로그램을 작성함으로써 행렬 연산을 통한 이미지 처리 기법에 관하여 학습한다.

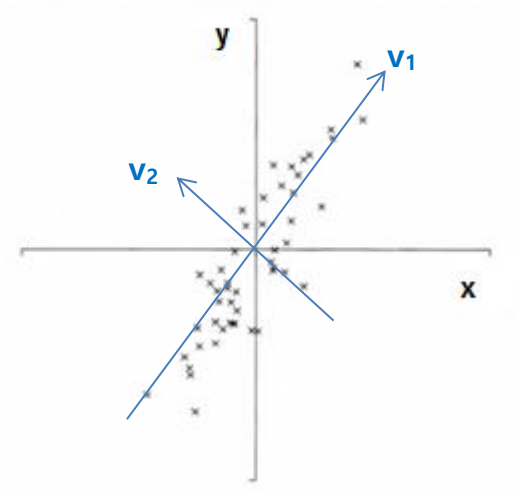
목차

서론

Principal Component Analysis(PCA, 주성분 분석)는 데이터 집합을 분석하는 기법 가운데 하나로, 데이터의 좌표 축을 변환시켜 각 좌표축에 대하여 원래 데이터를 사상시켰을 때 분산 값이 커지는 순으로 서로 독립된 좌표 계를 구하여 원래의 데이터를 변환 시키는 방법이다. 이와 같이 PCA는 데이터의 분산 상태를 분석함으로써 그 내부적 구조를 파악할 수 있게 해준다. PCA는 또한 Karhunen-Loeve Transform (KLT), Hotel transform, 또는 Proper Orthogonal Decomposition (POD)이라고 부르기도 한다. PCA는 eigenvalue decomposition 또는 singular value decomposition으로 구할 수 있다. 여기서는 covariance matrix의 eigenvalue decomposition을 통한 PCA 기법을 알아보도록 한다.

Principal Component Analysis

그림 1과 같이 분산된 데이터가 있을 경우 PCA를 통해서 v_1과 v_2를 구할 수 있으며 v_1은 원래 데이터의 분산 값이 최대가 되는 방향이 된다. PCA는 종종 원래 데이터를 특정 좌표계에 대하여 회전시키는 변환으로도 이해된다. 이 때 v_1과 v_2를 새로운 좌표축으로 보고, x, y 축에 겹쳐지게 회전시키고, 원래 데이터 점들도 함께 회전시키면 전체 데이터를 회전 시킨 것과 같은 효과가 생김을 알 수 있다. 새로운 좌표 계에서는 데이터의 분산 값이 최대인 방향이 각 좌표 축과 일치하게 된다.



$$\begin{pmatrix} V_{11}X_1 + V_{12}Y_1 & V_{11}X_2 + V_{12}Y_2 & \cdots \\ V_{21}X_1 + V_{22}Y_1 & V_{21}X_2 + V_{22}Y_2 & \cdots \end{pmatrix} = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \times \begin{pmatrix} X_1 & X_2 & X_3 & \cdots \\ Y_1 & Y_2 & Y_3 & \cdots \end{pmatrix}$$

그림 1. PCA의 도식적 표현과 행렬식

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

1. Covariance (공분산) matrix 계산

Covariance는 2개의 확률변수의 상관 관계를 나타내는 값으로, 하나의 변수 값이 증가할 때 다른 값도 증가하는 경우 Covariance는 양수 값이 되며 반대의 경우는 음수 값을 가지게 된다. 그림 1에서와 같은 점 p가 x, y의 2차원 데이터($p_i=(x_i\ y_i)^T$)라고 하면 그 Covariance matrix는 공식 (1) 같이 구할 수 있다.

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

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

$$\sum = E[(p-\mu)(p-\mu)^T], \quad \mu = \frac{1}{n}\sum_{i=1}^{n} p_i$$

식 1. 하나의 샘플에 대한 Covariance matrix의 표현 식

그림 1에 보인 점들의 집합 $P=(p_1, p_2, \ldots, p_n)$ 을 생각해 보면, Covariance matrix 는 공식 (2)와 같이 n개의 샘플 점들에 대하여 각각 구하여 더해 주거나,

$$\sum = \sum_{i=1}^{n} (p_i-\mu)(p_i-\mu)$$

식 2. n개의 샘플에 대한 Covariance matrix의 표현 식

또는 전체 점들의 집합 P를 하나의 매트릭스로 보고, 각 열을 평균값으로 빼준 행렬식(U)을 이용하여 공식 (3)과 같이 간단한 행렬 연산으로 표현 할 수도 있다. (표현 식은 간단하지만 전체 연산 량은 비슷해 지게 된다)

$$U = (p_1-\mu \quad p_2-\mu \quad p_3-\mu \quad \cdots \quad p_n-\mu)$$

$$\sum = UU^T$$

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

여기서 n개의 변수가 d-차원의 벡터 값을 가지는 것으로 일반화 시키면, 우리가 다루는 데이터는 공식 (4)와 같이 표현되며, 공식 (2) 또는 (3)을 이용하여 Covariance matrix를 구할 수 있다.

$$P = \begin{pmatrix} p_1 & p_2 & p_3 & \cdots & p_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \cdots & x_{dn} \end{pmatrix}$$

식 4. d차원, n개의 샘플의 표현 식

2. Eigenvalue decomposition

Covariance matrix를 구한 뒤에는 Eigenvalue decomposition을 이용하여 각 차원의 분산 값과 분산 방향을 구한다. Eigenvalue decomposition는 대칭형 matrix에 대하여 적용하며, 비대칭 매트릭스에 대해서는 Singular Value Decomposition (SVD)를 이용한다. 하나의 대칭형 matrix U에 대하여 eigenvalue, λ와 eigenvector v는 공식 (5)와 같이 표현된다. 전체 eigenvalue와 eigenvector는 각각 Λ, V와 같이 나타낼 수 있다.

$$UV = \Lambda V$$

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_d \end{pmatrix} \qquad V = \begin{pmatrix} V_1 & V_2 & \cdots & V_d \end{pmatrix} = \begin{pmatrix} V_{11} & V_{21} & \cdots & V_{d1} \\ V_{12} & V_{22} & \cdots & V_{d2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1d} & V_{2d} & \cdots & V_{dd} \end{pmatrix}$$

식 5. Eigenvalue equation

Eignevalue equation은 간단한 행렬식에 대해서는 characteristic equation을 이용하여 정확한 해를 구할 수 있으나, 보통 행렬식의 차원이 5 이상이 되면 정확한 해를 구하기가 어려워지며 근사 해를 구하게 된다. 실습에서는 OpenCV에서 제공하는 함수를 이용하여 eigenvalue와 eigenvector를 구하는 방법을 연습해 본다.

3. Eigenvector 정렬

각 샘플 차원의 분산량은 eigenvalue와 비례하게 된다. Eigenvalue를 크기 순으로 내림차순으로 정렬하고 각 eigenvector를 eigenvalue가 정렬된 방식으로 정렬하면 분산의 크기에 따라 정렬된 eigenvector를 구할 수 있다.

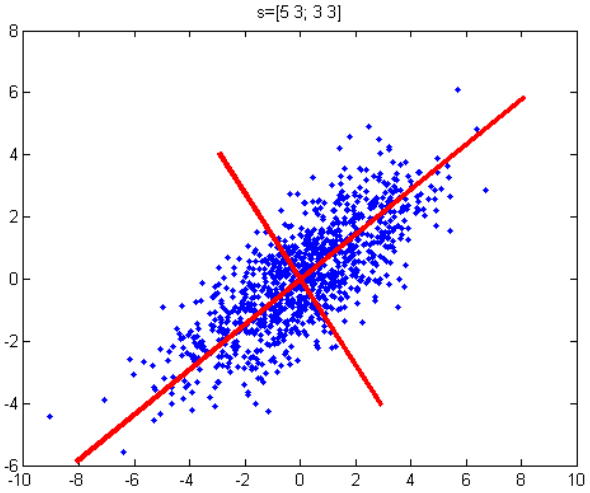
4. Eigenvector로 투영

최종적으로 원래 데이터를 eigenvector로 투영시킴으로써 PCA를 완성하게 된다. 원본 데이터를 eigenvector에 투영시키는 방법은 공식 (6)와 같이 행렬식으로 간단히 표현된다. 여기서 Q는 PCA에 의해서 변환된 데이터 값을 나타낸다.

$$Q = V^T U$$

식 6. Eigenvalue equation

그림 2는 2차원의 실제 데이터를 가지고 PCA를 수행한 예제를 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

PCA는 데이터의 통계적 분석을 위하여 널리 쓰이고 있지만, 영상 처리 분야에서는 다루기 힘든 고차원의 데이터를 압축하는 용도로 많이 쓰인다. 특히 얼굴 이미지 데이터의 압축을 통해 연산량을 줄이는 외에도, 얼굴에 내재된 신원 정보 외의 조명 변화와 같은 부차적인 정보들이 PCA 과정을 거치며 제거되는 효과를 보여, 초기 얼굴 인식 연구에 많이 사용되었다.

실습: PCA 구현

이번 실습에서는 $P=(p_1, p_2, ..., p_n)$으로 주어진 데이터(dimension=10, n=200)를 읽어 들여 PCA를 수행하는 프로그램을 작성해본다. 실습 프로그램을 실행 시키면 윈도우에 x, y 축이 그려지고, 메뉴 Problem1을 클릭하면 입력 데이터가 좌표 계에 검은색 점으로 그려진다. 아래 단계들을 완성하면, 입력 값의 분산이 가장 큰 방향을 나타내는 eigenvector에 투영된 값들을 파란색 점으로 원래의 좌표 계에 그리게 된다.

1) Covariance matrix (공분산 행렬) 생성

공식 (2) 또는 (3)을 이용하여 입력된 데이터의 Covariance matrix를 구한다. n개의 샘플의 평균값을 빼주는 것에 주의한다.

2) Covariance matrix의 eigenvalue와 eigenvector를 구하고 eigenvalue의 크기로 정렬하기

OpenCV에서 제공하는 함수를 사용하여 Covariance matrix로부터 eigenvalue와 eignevector를 구한다. 구해진 eigenvalue를 내림차순으로 정렬하고, eigenvector를 대응하는 eigenvalue의 순으로 정렬한다.

3) 원래 데이터를 eigenvector에 투영시키기

공식 (6)과 같이 원래 데이터를 eigenvector에 투영시킨다.

4) 투영된 데이터를 이차원 공간에 표현하기

최고 분산 방향을 나타내는 앞의 두 eigenvector에 투영된 데이터 값을 이차원 공간에 그려본다.

숙제: PCA에서 압축을 위한 최적 eigenvector 개수 구하기

1) Eigenvector의 개수는 원래 데이터의 차원(dimension)과 같으므로 모든 eigenvector를 이용하여 원래 데이터를 투영하면 데이터 압축의 효과를 볼 수 없다. 또한 데이터의 분산 특성을 고려하지 않고 특정 몇 개의 eigenvector를 이용하여 투영시키면 데이터의 유용한 정보가 손실될 수 있다. 유용한 정보를 손실하지 않으면서 최대로 압축 효과를 얻을 수 있도록 eigenvector의 개수를 설정할 수 있는 방법이 있는 지 설명해 보자.

참고 문헌

M. Petrou and C. Petrou, Image Processing: The Fundamentals, 2nd ed., Wiley.

R. Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing Using MATLAB, 2nd ed., Gatesmark Publishing.

J. E. Jackson, User's Guide to Principal Components, Wiley.

OpenCV Reference Manual, http://docs.opencv.org/opencv2refman.pdf