[선형대수학 #6] 주성분분석(PCA)의 이해와 활용

수학 이야기 2013. 11. 8. 21:01
주성분 분석, 영어로는 PCA(Principal Component Analysis).

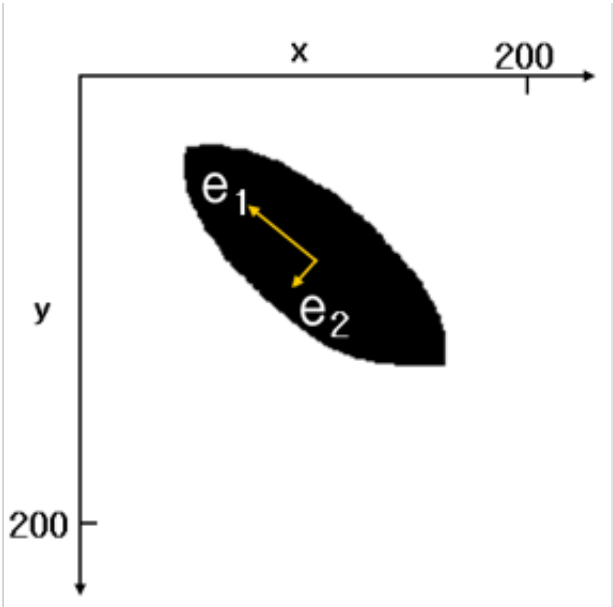
주성분 분석(PCA)은 사람들에게 비교적 널리 알려져 있는 방법으로서, 다른 블로그, 카페 등에 이와 관련된 소개글 또한 굉장히 많다. 그래도 기존에 이미 있는 내용들과 차별성이 있다면 이 글은 주성분 분석(PCA)을 자신의 공부, 연구 또는 개발에 보다 잘 활용할 수 있도록 주성분분석(PCA)의 다양한 활용예를 중심으로 기본 원리 등을 가급적 폭넓게 다뤄보고자 한다.

주성분 분석(PCA)은 사실 선형대수학이라기 보다는 선형대수학의 활용적인 측면이 강하며 영상인식, 통계 데이터 분석(주성분 찾기), 데이터 압축(차원감소), 노이즈 제거 등 다양한 활용을 갖는다.

PCA(Principal Component Analysis)에 대한 계산 방법이나 이론적인 부분은 뒤에 가서 다루고 일단은 PCA에 대한 개념 및 활용적인 측면을 먼저 살펴보도록 하자.

1. PCA(Principal Component Analysis)란?

PCA는 분포된 데이터들의 주성분(Principal Component)를 찾아주는 방법이다. 좀더 구체적으로 보면 아래 그림과 같이 2차원 좌표평면에 n개의 점 데이터 (x1,y1), (x2,y2), ..., (xn,yn)들이 타원형으로 분포되어 있을 때



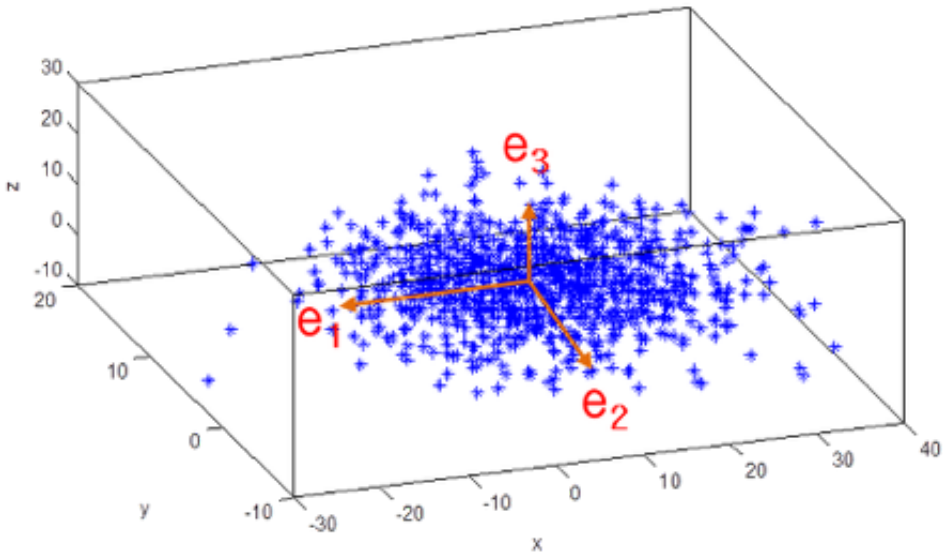
<그림 1> 2D에서의 PCA 예

이 데이터들의 분포 특성을 2개의 벡터로 가장 잘 설명할 수 있는 방법은 무엇일까? 그건 바로, 그림에서와 같이 e1, e2 두 개의 벡터로 데이터 분포를 설명하는 것이다. e1의 방향과 크기, 그리고 e2의 방향과 크기를 알면 이 데이터 분포가 어떤 형태인지를 가장 단순하면서도 효과적으로 파악할 수 있다.

PCA는 데이터 하나 하나에 대한 성분을 분석하는 것이 아니라, 여러 데이터들이 모여 하나의 분포를 이룰 때 이 분포의 주 성분을 분석해 주는 방법이다.

여기서 주성분이라 함은 그 방향으로 데이터들의 분산이 가장 큰 방향벡터를 의미한다. <그림 1>에서 e1 방향을 따라 데이터들의 분산(흩어진 정도)이 가장 크다. 그리고 e1에 수직이면서 그 다음으로 데이터들의 분산이 가장 큰 방향은 e2 이다.

PCA는 2차원 데이터 집합에 대해 PCA를 수행하면 2개의 서로 수직인 주성분 벡터를 반환하고, 3차원 점들에 대해 PCA를 수행하면 3개의 서로 수직인 주성분 벡터들을 반환한다. 예를 들어 3차원 데이터의 경우는 아래 그림과 같이 3개의 서로 수직인 주성분 벡터를 찾아준다.

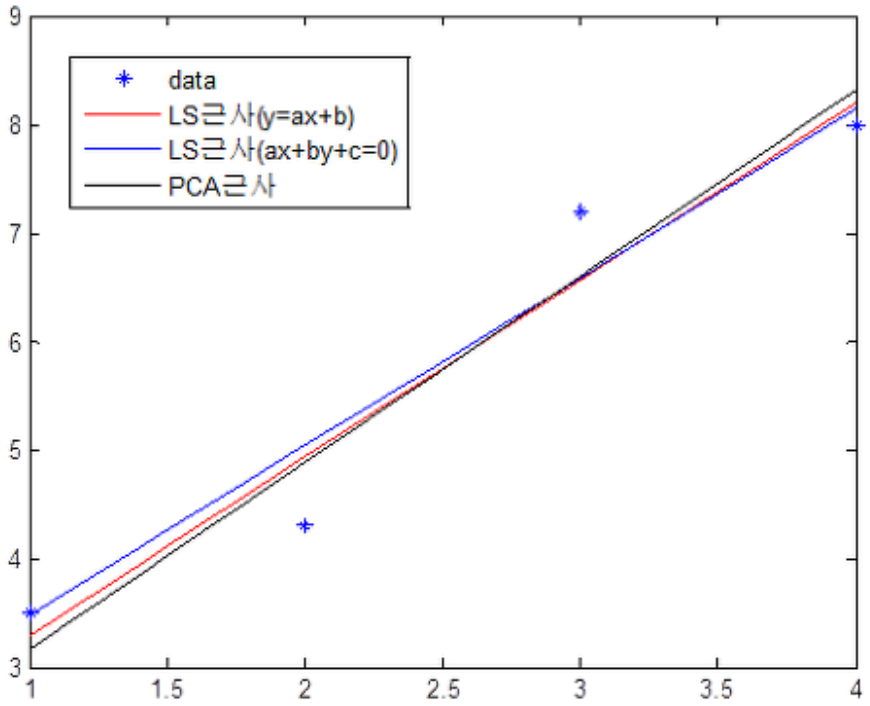


<그림 2> 3D에서의 PCA

2. PCA(Principal Component Analysis) 활용

A. 2D 점들의 직선 근사

이전 글 [선형대수학 #5] 선형연립방정식 풀이에서 예로 들었던 네 점 (1, 3.5), (2, 4.3), (3, 7.2), (4, 8)을 가장 잘 근사하는 직선의 방정식을 PCA로 구해보자.



<그림 3> PCA 라인 근사

PCA로 직선을 근사하는 방법은 데이터들의 평균 위치를 지나면서 PCA로 나온 제 1 주성분 벡터와 평행인 직선을 구하면 된다.

실제로 위 네 점을 지나는 직선을 PCA로 구해보면 y = 1.7194x + 1.4515가 나오며 그 그래프는 <그림 3>과 같다. 그림에서 보듯이 PCA로 구한 직선과 최소자승법(LS 방법)으로 구한 직선이 모두 다름을 볼 수 있다. 그 이유는 최소자승법은 직선과 데이터와의 거리를 최소화하는 반면 PCA는 데이터의 분산이 가장 큰 방향을 구하기 때문이다.

~~계산적인 측면에서 보면 PCA는 최소자승법(LS)에 비해 훨씬 효율적이다. 왜냐하면 데이터의 차원이 n, 데이터의 개수가 m개일 때 LS는 n×m 행렬의 의사역행렬(pseudo inverse)를 계산해야하지만 PCA는 n×n 행렬의 고유값 분해만 계산하면 되기 때문이다 (2차원 평면에서 1,000개의 점을 근사하는 경우를 생각해 보면 LS는 2 x 1,000의 역행렬을 계산해야 하고 PCA는 2 x 2 행렬의 고유값 분해만 하면 된다).~~ => 잘못된 설명으로 삭제함 (2015.12.29 댓글 참조)

☞ 그림 3에서 LS근사(y=ax+b)는 직선과의 y축 거리를 최소화시키고, LS근사(ax+by+c=0)는 평면 z = ax+by+c와의 z축 거리를 최소화시킨다 (평면 z = ax+by+c과 xy평면과의 교선이 ax+by+c=0). PCA는 데이터들의 평균점을 지나는 직선들 중에서 데이터들을 직선에 투영(projection)시켰을 때 해당 직선을 따라서 데이터의 분산이 최대가 되는 방향의 직선을 구한다.

B. 3D 점들의 직선 또는 평면 근사

3차원 공간에서 점들의 집합을 직선으로 근사하는 문제는 최소자승법(LS, Least Square Method)으로는 쉽지가 않다. 하지만 주성분분석(PCA)을 이용하면 이를 손쉽고 효율적으로 구할 수 있다. 만일 주어진 데이터 점 (x1,y1,z1), (x2,y2,z2), ... (xn,yn,zn)들의 평균이 (mx,my,mz)이고 PCA로 구한 제 1 주성분 벡터가 e1 = (a,b,c)라면 이들 데이터 점들을 근사하는 직선식은 다음과 같다.

$$\frac{x-m_x}{a}=\frac{y-m_y}{b}=\frac{z-m_z}{c} \quad \text{--- (1)}$$

이들 점들을 평면 방정식으로 근사하는 경우에도 PCA를 적용할 수 있다. 만일 PCA로 나온 제 1 주성분 벡터가 e1, 제 2 주성분 벡터가 e2라면 근사 평면 방정식은 다음과 같다.

$$(e_1 \times e_2) \cdot (\mathbf{x}-\mathbf{m})=0 \quad \text{--- (2)}$$

단, ×는 벡터의 외적, **x** = (x,y,z), **m** = (mx,my,mz).

C. eigenface와 영상인식 응용

PCA가 영상인식에 활용되는 대표적인 예는 얼굴인식(face recognition)이다. 그리고 이와 관련된 개념 혹은 용어로서 eigenface(아이겐페이스)라는게 있다.

다음과 같은 20개의 45x40 얼굴 이미지들이 있다고 하자.

<그림 4> 45x40 얼굴 이미지 20장

이미지에서 픽셀 밝기값을 일렬로 연결하여 벡터로 만들면 이들 각각의 얼굴 이미지는 45x40 = 1,800 차원의 벡터로 생각할 수 있다 (즉, 각각의 이미지는 1,800 차원 공간에서 한 점(좌표)에 대응).

이제 이 20개의 1,800차원 점 데이터들을 가지고 PCA를 수행하면 데이터의 차원 수와 동일한 개수의 주성분 벡터들을 얻을 수 있다. 이렇게 얻어진 주성분 벡터들을 다시 이미지로 해석한 것이 eigenface이다 (얼굴 이미지를 가지고 얻은 벡터이기에 eigenface라 부른다). 실제 위 이미지에 대해 얻어진 1,800개의 eigenface들 중 분산이 큰 순서대로 처음 20개를 나열하면 아래 그림과 같다.

<그림 5> 처음 20개의 eigenface들

위 그림에서 볼 수 있듯이 앞부분 eigenface들은 데이터들에 공통된 요소(얼굴의 전반적인 형태)를 나타내고 뒤로 갈수록 세부적인 차이 정보를 나타낸다. 그리고 더 뒤로 가면 거의 노이즈(noise)성 정보를 나타낸다.

앞서 PCA를 통해 얻어진 주성분 벡터들은 서로 수직인 관계에 있다고 말한 바 있다. 이 말은 주성분 벡터들이 n차원 공간을 생성하는 기저(basis) 역할을 할 수 있음을 의미한다. 즉, PCA로 얻은 주성분 벡터들을 e1, e2, ..., en라면 임의의 n차원 데이터 x는 x = c1e1 + c2e2 + ... + cnen과 같이 ei들의 일차결합으로 표현될 수 있다 (이 때, 상수계수 ci들은 x와 ei의 내적 즉, ci = x·ei로 계산할 수 있으며 이와 같이 어떤 데이터 집합의 데이터들을 그들의 주성분 벡터들의 일차결합으로 표현하는 것을 Karhunen–Loève transform (KLT) 또는 Hotelling transform이라 부른다).

그런데, 뒷부분의 주성분 벡터들은 데이터 분포에 포함된 노이즈(noise)성 정보를 나타내기 때문에 뒷부분은 버리고 전반부 k개의 주성분 벡터들만을 가지고 원래 데이터를 표현하면 노이즈가 제거된 데이터를 얻을 수 있다. 즉, 원래의 x가 x = c1e1 + c2e2 + ... + cnen일 때 xk = c1e1 + ... +ckek로 x를 근사하는 것이다. 위 얼굴 이미지들에 대해 전반부의 일부(k = 20, 10, 5, 2) eigenface들만을 이용한 근사 이미지들은 아래 그림과 같다 (클릭시 확대 이미지).

<그림 6> k개의 eigenface만을 이용한 데이터 복원(reconstruction)

그림에서 볼 수 있듯이 많은 수의 eigenface를 이용하면 원본 얼굴과 거의 유사한 근사(복원) 결과를 볼 수 있지만 k가 작아질수록 개인 고유의 얼굴 특성은 사라지고 공통의 얼굴 특성이 남게 된다 (k=20인 경우 원래 얼굴이 그대로 살아나지만 k=2인 경우 개인간의 구분이 거의 사라짐을 볼 수 있다).

☞ k개의 주성분 벡터만을 이용하여 원래 데이터를 표현하는 것은 통상적으로 근사라는 용어보다는 복원(reconstruction)이라는 용어가 주로 사용된다.

☞ 노이즈(noise)에 대해 좀더 생각해 보면, 앞서 말했듯이 PCA는 개별 데이터에 대한 분석이 아니라 전체 데이터에 대한 집합적 분석 도구이다. 만일 강아지 100마리에 대한 PCA 분석 결과와 고양이 100마리에 대한 PCA 분석 결과가 있다고 하자. 이 때, 강아지 데이터에서 얻어진 eigenface들 중 앞의 것들은 (고양이와 구분되는) 강아지 고유의 형태 정보를 나타내고 뒤로 갈수록 강아지들 내부에서 강아지들 사이의 차이점을 표현할 수 있는 정보를 나타낸다. 그리고 더 뒤로 나아가면 노이즈성 정보를 표현한다. 마찬가지로 고양이 데이터에 대한 eigenface들은 주요한 성분일수록 고양이 공통의 성분, 뒤로 갈수록 고양이 개체 사이의 차이를 가르는 요소를 나타낸다. 그런데, 어디서 어디까지가 데이터 공통 성분이고 어디까지가 데이터의 차이인지, 그리고 어디부터 노이즈 성분인지 그 구분은 명확하지 않다. 그 경계를 이론적으로 계산하는 방법론 등도 있긴 하지만 대부분은 응용에 따라서, 그리고 데이터에 따라서 주관적으로 또는 실험적으로 결정하는 것이 통상적이다.

☞ 위에서 설명한 k개의 주성분 벡터만을 이용하여 원래 데이터를 표현하는 것은 관점에 따라서 차원 감소(dimension reduction), 데이터 압축(compression), 노이즈 제거 등으로 다양하게 해석될 수 있다. 먼저, 차원감소라 함은 n차원의 데이터를 xk = c1e1 + ... + ckek로 표현했을 때 e1, ..., ek를 새로운 좌표축으로 하는 공간에서 x를 (c1, c2, ..., ck)와 같이 k차원의 점으로 표현한다는 의미이다. 둘째, 데이터 압축의 의미는 {x}들을 그대로 저장하지 않고 k개의 주성분 벡터들과 계수 (c1, .., ck)들만을 저장하면 저장용량을 크게 줄일 수 있다는 의미이다. 참고로 SVD(특이값분해)를 이용한 데이터 압축은 데이터를 개별적으로 압축하지만 PCA는 데이터를 집합적으로 압축한다는 점이 다르다. 마지막으로 노이즈 제거란 의미는 k개의 주성분만을 이용해서 데이터를 복원함으로써 의미없는 노이즈 부분을 날린다는 의미이다.

☞ 참고용으로 위에서 사용한 20개의 샘플 얼굴 이미지들과 PCA, eigenface 및 k reconstruction 과정에 대한 matlab 코드를 첨부로 올립니다:

eigenface_test.zip

D. PCA를 이용한 얼굴검출과 얼굴인식

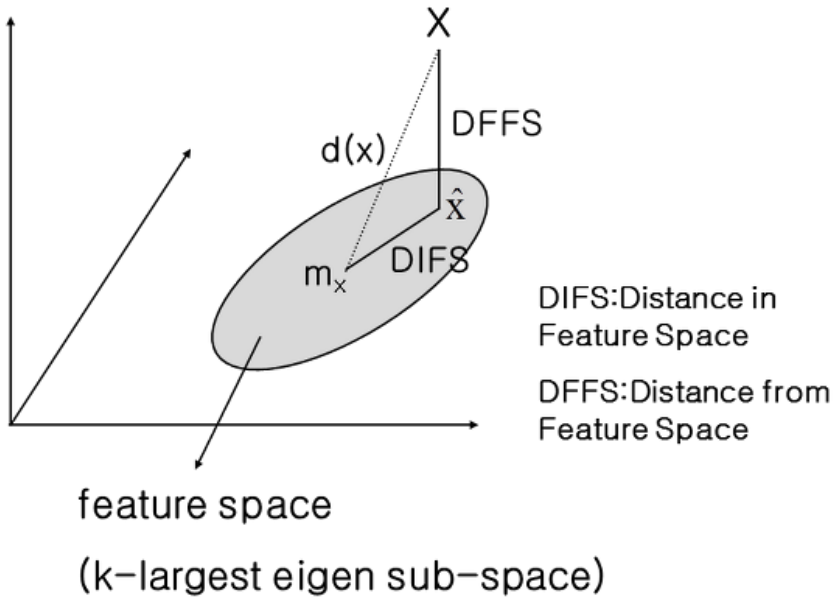
먼저, 컴퓨터 비전에서 사용하는 detection과 recognition의 차이를 살펴보면 face detection은 사람 구분없이 그냥 얼굴을 찾는 것이고, face recognition은 이 얼굴이 누구 얼굴인지를 알아내는 것을 말한다.

i) face detection 응용

PCA를 얼굴검출에 응용하기 위해서는 먼저 수많은(최소 1,000개 이상) 얼굴 샘플들을 모아서 eigenface들을 구한 후 얼굴로서 의미가 있다고 생각되는 전반부 k개의 eigenface들만을 선택한다. 이후 테스트할 입력 영상(윈도우 영역) x가 들어오면 x를 k개의 eigenface들만을 이용하여 복원(reconstruction) 했을 때 원래 영상 x와 얼마나 가까운지를 살펴

본다. 만일 x가 k개의 eigenface를 조합해서 완벽히 근사된다면 x는 얼굴일 확률이 매우 높다. 또 하나의 판단 기준은 이렇게 근사된 xk가 평균적인 얼굴(average face)과 얼마나 가까운가이다. x가 아무리 eigenface들로 근사가 잘 되어도 실제 평균 face와 동떨어져 있다면 face로 보기 힘들다. 따라서, x에 대한 최종 판단은 얼마나 근사가 잘 되는지와 근사된 얼굴이 실제 얼굴 이미지 평균과 얼마나 차이가 있는지를 종합적으로 고려하여 판단한다.

이러한 두 평가기준의 차이를 그림으로 나타내면 아래 그림과 같다. 그림에서 DFFS는 얼마나 근사가 잘 되는지를 나타내고 DIFS는 근사된 얼굴이 얼굴 평균과 얼마나 가까운지를 나타낸다.



<그림 7> PCA의 detection 응용

☞ 위 방법은 PCA가 어떻게 활용될 수 있는지를 설명하기 위한 예일 뿐이며 실제로 위 방법이 face detection에 있어서 매우 효과적인 방법이라는 말은 아니다. 위 방법은 예전에 봤던 1998년도 논문 방법인데, 당시로서는 대표적인 face detection 논문 중 하나였다 (논문 제목은 기억이 잘 안남). 예전에 위 방법을 실제로 구현해 본 적이 있었는데 그다지 결과가 좋지는 않았다.

ii) face recognition 응용

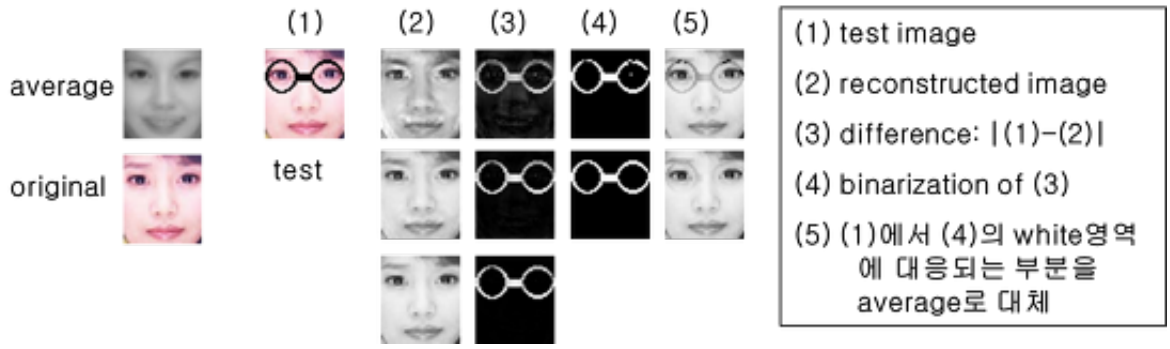
PCA를 recognition에 응용할 때에는 조금 방법이 다르다. 먼저, 모든 사람의 얼굴 샘플을 모을 필요가 없으며 인식 대상이 되는 사람들의 얼굴 샘플들만을 모은다 (예를 들어 보안시스템의 경우 출입이 허가된 사람들의 얼굴 샘플). 이들 샘플들에 대해 PCA를 통해 k개의 주요 eigenface들을 구한 후 각 개인들을 eigenface로 근사했을 때의 근사계수를 저장한다. 즉, xk = c1e1 + ... + ckek일 때 (c1, ..., ck)를 개인의 고유 feature로 저장한다. 이후 입력 데이터 x가 들어왔을 때 이를 k개의 eigenface로 근사한 근사계수가 미리 저장된 개인별 근사계수들 중 누구와 가장 가까운지를 조사하여 x를 식별한다.

☞ 만일 인식할 대상이 DB에 저장된 사람들 중 한명이라면 이 방법으로 등록된 사람들 중 누구인지 쉽게 식별할 수 있다. 그러나 DB에 등록되지 않은 사람이라면 이를 식별하는 것은 별도의 문제이다. 실제로 이 방법이 얼마나 효과적인지는 잘 알지 못하며 그냥 PCA 특성을 이런 식으로도 활용 가능하다는 정도로 참고하기 바랍니다.

E. PCA를 이용한 안경제거

이게 딱히 유용한 응용인지는 잘 모르겠지만 예전에 한 논문에서 봤던 방법으로 PCA를 이용하여 얼굴영상에서 안경쓴 사람의 안경 영역을 찾아서 안경이 제거된 얼굴 영상을 얻을 수 있는 방법이 있다 (역시 논문 제목은 잘 기억이 안남..)

그 방법을 간략히 설명하면 먼저 안경을 쓰지 않은 사람들로만 구성된 얼굴 이미지 집합을 구성한 후, PCA를 통해 eigenface들을 추출한다. 이후 테스트 영상 x에 대해 이를 k개의 eigenface들로 근사한 근사 영상 xk를 구하여 |x - xk| 의 차이가 임계치 이상인 영역(즉, 안경 영역)을 평균 얼굴 이미지로 대체한다. 이 과정을 그림으로 나타내면 아래와 같다.



<그림 8> PCA의 안경제거 응용

☞ 이상 몇몇 PCA 응용 예들을 설명했지만 PCA의 기본 원리 및 특성만 잘 이해하고 있으면 이 외에도 다양한 활용이 가능할 것입니다. 예를 들어, 영상 쪽이 아닌 음성인식, 기타 신호처리 쪽으로도 유사한 방식의 활용이 가능합니다.

3. PCA의 계산

PCA를 알기 위해서는 먼저 공분산 행렬(covariance matrix)에 대해 알아야 한다.

먼저, x와 y의 공분산(covariance)이란 아래 식과 같이 정의된다.

$$cov(x,y)=E[(x-m_x)(y-m_y)]$$
$$=E[xy]-m_xm_y \quad \text{--- (3)}$$

단, mx는 x의 평균, my는 y의 평균, E[]는 기대값(평균).

x의 분산은 x들이 평균을 중심으로 얼마나 흩어져 있는지를 나타내고, x와 y의 공분산은 x, y의 흩어진 정도가 얼마나 서로 상관관계를 가지고 흩어졌는지를 나타낸다. 예를 들어, x와 y 각각의 분산은 일정한데 x가 mx보다 클때 y도 my보다 크면 공분산은 최대가 되고, x가 mx보다 커질때 y는 my보다 작아지면 공분산은 최소(음수가 됨), 서로 상관관계가 없으면 공분산은 0이 된다.

공분산 행렬(covariance matrix)이란 데이터의 좌표 성분들 사이의 공분산 값을 원소로 하는 행렬로서 데이터의 i번째 좌표 성분과 j번째 좌표 성분의 공분산 값을 행렬의 i행 j열 원소값으로 하는 행렬이다.

- covariance
  - $\mathrm{cov}(x,y) = E[(x-m_x)(y-m_y)]$
- covariance matrix
  - $x=[x_1,\ldots,x_n]^T$: sample data, n차원 열벡터
  - $C = E[(x-m_x)(x-m_x)^T]$: n×n 행렬
  - $\langle C\rangle_{ij} = E[(x_i-m_{xi})(x_j-m_{xj})^T]$: i번째 성분과 j번째 성분의 공분산
  - C is real and symmetric $C=\begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$

<그림 9> 공분산 행렬

예를 들어, 2차원 데이터 n개가 (x1, y1), (x2, y2), ..., (xn, yn)와 같이 있다면 이 데이터들의 공분산 행렬은 다음과 같이 계산된다.

$$C=\begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(x,y) & cov(y,y) \end{pmatrix}$$
$$=\begin{pmatrix} \frac{1}{n}\sum(x_i-m_x)^2 & \frac{1}{n}\sum(x_i-m_x)(y_i-m_y) \\ \frac{1}{n}\sum(x_i-m_x)(y_i-m_y) & \frac{1}{n}\sum(y_i-m_y)^2 \end{pmatrix} \quad \text{--- (4)}$$

PCA란 한마디로 말하면 입력 데이터들의 공분산 행렬(covariance matrix)에 대한 고유값분해(eigendecomposition)로 볼 수 있다(고유값 분해에 대해서는 [선형대수학 #3] 고유값과 고유벡터 글 참조). 이 때 나오는 고유벡터가 주성분 벡터로서 데이터의 분포에서 분산이 큰 방향을 나타내고, 대응되는 고유값(eigenvalue)이 그 분산의 크기를 나타낸다.

PCA
- C : covariance matrix of x
- $C = P\Sigma P^T$ (P: orthogonal, Σ: diagonal)

$$C = \begin{pmatrix} e_1 & \cdots & e_n \end{pmatrix} \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} \begin{pmatrix} e_1^T \\ \vdots \\ e_n^T \end{pmatrix}$$

- P : n×n orthogonal matrix
- Σ : n×n diagonal matrix
- $Ce_i = \lambda_i e_i$
  - e_i : eigenvector of C, direction of variance
  - λ_i : eigenvalue, e_i 방향으로의 분산
  - $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$
- e_1: 가장 분산이 큰 방향
- e_2: e_1에 수직이면서 다음으로 가장 분산이 큰 방향
- e_k: $e_1, \ldots, e_{k-1}$에 모두 수직이면서 가장 분산이 큰 방향

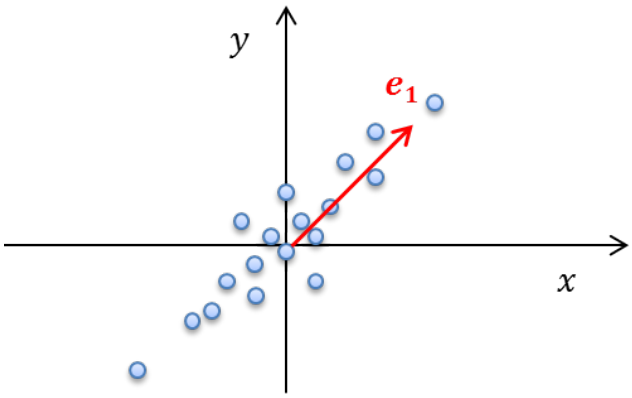
<그림 10> PCA 계산 과정

4. 생각할 점

마지막으로 생각할 점 2가지를 남기며 글을 마무리합니다. 아직은 정리되지 않은.. 제 스스로 가지고 있는 의문이기도 합니다.

Q1. PCA에서 찾은 제 1 주성분 벡터는 데이터 분포의 분산이 가장 큰 방향을 나타낸다. 그리고 eigenface 응용에서는 첫번째 eigenface가 얼굴 공통의 형태 정보를 나타내고 뒤로 갈수록 세부적인 차이를 나타낸다. 그런데, eigenface가 바로 주성분 벡터이므로 분포의 분산방향일텐데 이게 왜 공통의 형태 정보가 되는 것일까?

=> 아래 댓글들로 좋은 의견들을 주셨고, 댓글들을 보면서 나름 정리한 설명을 적어봅니다.

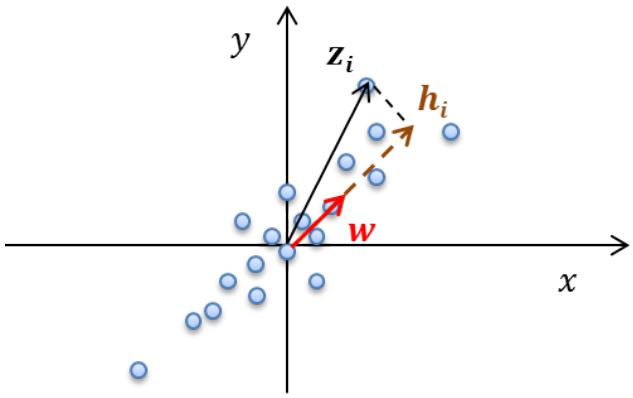


A. 위 그림과 같이 PCA에서 구한 첫번째 주성분 벡터 e1은 데이터들의 분산이 가장 큰 방향을 나타낸다. 그런데, 이것을 데이터들의 좌표값 (xi, yi) 입장에서 생각해 보면 e1은 데이터들의 값을 가장 잘 대표하는 (공통의) 값이 된다. 위 그림 예에서 e1은 x=y인 단위벡터이고 데이터들도 x좌표와 y좌표가 거의 유사한 값을 갖는다. 즉, 데이터들을 (xi,yi) = kie1과 같이 e1의 상수배로 근사해 보면 e1이 데이터들의 가장 공통적인 좌표값을 나타냄을 알 수 있다. ◇

Q2. 왜 공분산행렬의 고유벡터가 데이터 분포의 분산 방향이 되고, 고유값이 그 분산의 크기가 되는 것일까?

=> 아래의 ankh님이 댓글로 알려주신 링크(http://www.stat.cmu.edu/~cshalizi/350/lectures/10/lecture-10.pdf)의 문서를 보고 어느정도 수식적으로 이해를 하게 되었습니다 (감사드립니다). 그 내용을 정리해 보면 다음과 같습니다 (수학적인 지식이 필요한 내용이긴 한데, 그냥 배경 설명없이 적도록 하겠습니다).

A. 데이터들을 zi = (x1, ..., xp), i = 1, ..., n라 하자 (데이터의 차원은 p, 개수는 n). 이 때, 크기가 1인 임의의 단위벡터 w에 대해 zi들을 w에 투영시킨(projection) 벡터 hi = (zi·w)w 들을 생각해 보면 입력 데이터 zi들의 분산을 최대화하는 방향은 결국 프로젝트된 벡터 hi의 크기인 (zi·w)의 분산을 최대화시키는 w를 찾는 문제와 동일하다.



(zi·w) 들의 분산을 σ_w^2라 놓고, 원래의 입력 데이터들을 행벡터로 쌓아서 생성한 n x p 행렬을 Z라 하면

$$\sigma_w^2 = \frac{1}{n}\sum_i (z_i \cdot w)^2 - \left(\frac{1}{n}\sum_i (z_i \cdot w)\right)^2$$
$$= \frac{1}{n}\sum_i (z_i \cdot w)^2$$
$$= \frac{1}{n}(Zw)^T(Zw)$$
$$= \frac{1}{n} w^T Z^T Z w$$
$$= w^T \frac{Z^T Z}{n} w$$
$$= w^T C w \quad \text{--- (5)}$$

와 같이 정리된다 (zi들의 평균이 0이 되도록 centering을 한 후라고 생각하면 (zi·w)의 평균은 0). 이 때, C = Z^TZ/n으로 잡은 C는 zi들의 공분산 행렬이 된다.

따라서 구하고자 하는 문제는 w가 단위벡터($w^Tw=1$)라는 조건을 만족하면서 w^TCw를 최대로 하는 w를 구하는 constrained optimization 문제로 볼 수 있으며 Lagrange multiplier λ를 도입하여 다음과 같이 최적화 문제로 식을 세울 수 있다.

$$u=w^TCw-\lambda(w^Tw-1) \quad \text{--- (6)}$$

이 때, u를 최대로 하는 w는 u를 w로 편미분한 ∂u/∂w 를 0 으로 하는 값이다.

$$\frac{\partial u}{\partial w}=2Cw-2\lambda w=0$$
$$Cw=\lambda w \quad \text{--- (7)}$$

즉, zi에 대한 공분산 행렬 C의 eigenvector가 zi의 분산을 최대로 하는 방향벡터임을 알 수 있다. 또한 여기서 구한 w를 식 (5)에 대입하면 $\sigma_w^2 = w^T\lambda w = \lambda$ 가 되므로 w에 대응하는 eigenvalue λ가 w 방향으로의 분산의 크기임을 알 수 있다. ◇

[선형대수학 #1] 주요용어 및 기본공식
[선형대수학 #2] 역행렬과 행렬식(determinant)
[선형대수학 #3] 고유값과 고유벡터 (eigenvalue & eigenvector)
[선형대수학 #4] 특이값 분해(Singular Value Decomposition, SVD)의 활용
[선형대수학 #5] 선형연립방정식 풀이
[선형대수학 #6] 주성분분석(PCA)의 이해와 활용

by 다크 프로그래머

'수학 이야기' 카테고리의 다른 글

두 직선, 벡터의 관계(사이각,회전각) 구하기 (9)	2014.01.28
[선형대수학 #6] 주성분분석(PCA)의 이해와 활용 (184)	2013.11.08
[선형대수학 #5] 선형연립방정식 풀이 (31)	2013.10.29
[선형대수학 #4] 특이값 분해(Singular Value Decomposition, SVD)의 활용 (136)	2013.10.23

IT, PCA, principal component analysis, 선형대수학, 수학, 영상인식, 주성분분석

설정

트랙백

댓글