행렬을 통해서 본 PCA 가 데이터 분석에서 갖는 의의

류재헌, 박정현, 이상구 교수님, 이시원, 이지용, 이혜연

인공지능을 위한 기초수학의 핵심적인 내용을 담고 있는 주성분분석은 차원 축소를 하면서도 원 데이터의 분포를 최대한 보존한다는 특성을 가지고 있습니다. 이것이 가능한 이유는 우리가 지금까지 배운 행렬의 특성인 정규직교화를 이용했기 때문입니다. 이번 보고서를 통해서 행렬의 데이터 분석에서 '직교'와 '정규화'가 갖는 의의와 PCA 의 과정을 살펴보고 예제를 통한 실습까지 해 보려 합니다.

PCA 에서 정규직교화와 SVD 가 갖는 의미

정규직교벡터란, 직교 벡터(orthogonal vector)이면서 단위 벡터(unit vector)인 벡터를 말합니다. 두 벡터가 90 도의 각도를 이루고, 각 벡터의 길이(크기)는 1 인 방향성분만을 나타내는 벡터를 정규직교벡터(orthonormal vector)라고 합니다.

$$Orthonormal\ \ vectors:$$
$$\boldsymbol{q}_i{}^T\boldsymbol{q}_j=\begin{cases} 0 & if\ \ i\neq j \\ 1 & if\ \ i=j \end{cases} \qquad \cdots(3)$$

<그림 1> 정규직교화 벡터의 수식, 러너게인 티스토리

이러한 정규직교벡터들은 행렬의 정규직교기저(orthonormal basis)를 이루게 됩니다. 정규직교기저를 만드는 이유는 선형대수의 행렬 연산에 있어서 좋은 결과를 보여주기 때문입니다. 벡터를 정규화하면 수치 연산을 하는 과정에서 값이 무한정으로 커지거나 작아져서 발생하는 overflow 나 underflow 문제에 있어서 자유롭습니다.

왜냐하면 정규화를 통해서 벡터들을 동일한 스케일로 만들었고, 이렇게 만들어진 단위 행렬은 연산에서 크기에 영향을 미치지 않기 때문입니다. 직교화는 행렬의 수식을 매우 간단하게 만드는 효과가 있습니다. 이것은 직교행렬이 가지는 성질 덕분입니다.

$$if\ \ Q\ \ is\ \ square,\ \ then$$
$$Q^T=Q^{-1}$$

<그림 2> 직교행렬이 가지는 성질, 러너게인 티스토리

보시다시피 Q가 직교행렬일 경우, Q의 transpose와 Q의 inverse는 같게 됩니다. 어떤 벡터를 행렬 A에 정사영(projection) 시키고 싶을 때를 예로 들면,

$$P=A(A^TA)^{-1}A^T \qquad \cdots(15)$$

$$if\ \ Q\ \ is\ \ orthogonal\ \ matrix\ \ then,$$
$$P=Q(Q^TQ)^{-1}Q^T=\ QQ^T \qquad \cdots(16)$$
$$I$$

<그림 3> 정사영에서 직교행렬을 이용한 연산의 예시, 러너게인 티스토리

보시다시피 직교행렬의 성질로 인해서 연산이 매우 간단해지는 것을 볼 수 있습니다. 여기까지 정리한 내용에서, 이것이 PCA 에 주는 이점과 PCA 의 과정을 따져봅시다.

1. PCA는 여러 차원의 변수에서, 기존의 변수를 일차 결합하여 선형 연관성이 없는 주성분을 만들고 변수를 축소한다.
2. 이렇게 차원을 축소해 원 데이터를 보존하는 3 가지의 변수만 택해서 3 차원 데이터로 차원을 줄이면, 11 차원의 변수를 모두 고려하는 것보다 계산과

시각화가 용이해 쉬운 데이터 분석이 가능하다. (예를 들면, 두 개의 상관관계가 있는 데이터를 생각해보자. 이 데이터들은 하나가 커지면 다른 하나는 거기에 비례해서 커지는 상관관계를 가지고 있다. 이 때 이 두 데이터를 모두 가져오기보단, 하나의 데이터만 가져와도 나머지 하나의 데이터는 자동적으로 알기 때문에 하나만 가져오는 것이 낫다.)

3. 이러한 과정의 핵심은 SVD(특이값분해, singular value decomposition)이며, 일반적인 행렬의 경우에서도 고윳값 분해와 유사한 행렬 분해를 진행하여 p×p 대각선 행렬의 데이터에서 k 개의 데이터를 취사선택해 유의미한 데이터를 갖는 행렬을 만들 수 있다.

요약하자면, 데이터의 나열일 뿐인 n×p 행렬을 유의미한 데이터를 가지는 rank 가 k 인 n×p 행렬로 분석하는 과정이라고 할 수 있겠습니다.

PCA 과정

1. 데이터를 센터링한다.
2. 센터링된 데이터의 행렬 X 의 특이값 분해(SVD)를 구한다.

■ 이제 X의 특이값분해(SVD)가 다음과 같이 주어져 있다고 하자.

$$X = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_k\ \mathbf{u}_{k+1}\ \cdots\ \mathbf{u}_n] \left[\begin{array}{cccc|ccc} s_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & s_k & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & 0 & s_{k+1} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & s_p \\ \hline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{bmatrix}$$

$$= USV^T = U \begin{bmatrix} S_1 \\ O \end{bmatrix} V^T = \sum_{i=1}^{p} s_i \mathbf{u}_i \mathbf{v}_i^T$$

3. V 의 열벡터가 주축(principal axes)이 된다.
4. Z=US 의 열벡터들이 원 데이터를 주축에 정사영하여 얻어진 주성분 점수(PC score)가 된다.
5. 데이터를 p 차원에서 k(<<p)차원으로 줄이기 위하여, U 의 처음 k 개의 열벡터(U_k)와 S 의 k 번째 선행 주 부분행렬(leading principal submatrix)(S_k)을 U_kS_k 는 처음 k 개의 PC 를 포함하는 n×k 행렬이 된다.

아래는 교수님께 질문한 SVD 와 라그랑주 승수법의 상관관계입니다. 특이값 분해를 시행하고, 그 후 라그랑주 승수법을 이용해 PCA 를 진행할 수 있습니다.

PCA 에서 SVD 를 이용한 계산 과정에서 질문이 있습니다.

작성자 : 류재헌(2015####86)작성일 : 8 월 12 일 오후 7:20

■ **이제 X의 특이값분해(SVD)가 다음과 같이 주어져 있다고 하자.**

$$X = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_k\ \mathbf{u}_{k+1}\ \cdots\ \mathbf{u}_n] \left[\begin{array}{cccc|ccc} s_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & s_k & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & 0 & s_{k+1} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & s_p \\ \hline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{bmatrix}$$

$$= USV^T = U\begin{bmatrix} S_1 \\ O \end{bmatrix} V^T = \sum_{i=1}^{p} s_i \mathbf{u}_i \mathbf{v}_i^T$$

여기서 U, V는 직교행렬이고, S_1는 크기 순서대로 배열된 특이값(singular value) $s_1 \geq s_2 \geq \cdots \geq s_p \geq 0$을 주대각선 성분으로 하는 $p \times p$ 대각선행렬 $S_1 = \mathrm{diag}(s_1, \ldots, s_p)$ 이다. 그러면 관계식

$$\Sigma = V\Lambda V^T = \frac{1}{n} X^T X = \frac{1}{n} V\begin{bmatrix} S_1^T & O \end{bmatrix} U^T U \begin{bmatrix} S_1 \\ O \end{bmatrix} V^T = V\left(\frac{S_1^2}{n}\right) V^T$$

으로부터 다음을 얻는다.

(1) $X = USV^T$라 하면 **V의 열벡터**가 **주축(principal axes)**이 된다.

(2) $Z = XV = USV^TV = US$ 이므로, **US의 열벡터들이** PC score (주성분점수, principal component score)**가 된다.**

(3) 특이값 s_i는 Σ의 고윳값 λ_i와 관계식 $\lambda_i = \frac{s_i^2}{n}$을 만족하며, λ_i는 대응하는 PC의 분산이 된다.

(4) 데이터를 p차원에서 k ($\ll p$) 차원으로 줄이기 위하여, U의 처음 k개의 열벡터(U_k)와 S의 k번째 선행 주 부분행렬(leading principal submatrix)(S_k)을 택

이 과정에서, k 는 사용자의 마음대로 p 이하의 값에서 취사선택해도 되는 것인지, 그리고 SVD 를 이용한 이 과정이 라그랑주의 승수법을 이용한 PCA 와 어떤 관계가 있는지 궁금합니다.

이상구(LEE SANGGU) 8 월 12 일 오후 8:37

1. k 가 너무 작아지면 정확도가 떨어지니까 ... 답이 필요한 제한시간과 자신의 계산 능력 및 데이터 처리 능력 고려하여, 자신이 선택한 k 차원으로 줄이면 됩니다. 2. PCA 하려면 먼저 SVD 이용하고, 이어서 라그랑주 승수법 이용 해서 PCA 하는 것 입니다.

라그랑주 승수법

라그랑주 승수법은 최대값 또는 최소값을 찾으려는 문제에서 사용되는 해결방법으로 최적화 문제에 사용되는 수학적 기법입니다..

라그랑주의 승수법	
문제	maximize(또는 minimize) $f(x, y)$ subject to $g(x, y) = 0$
[1단계]	라그랑주 함수 $L(x, y, \lambda) = f(x, y) - \lambda g(x, y)$를 만든다.
[2단계]	$\nabla L(x, y, \lambda) = \mathbf{0}$, 즉 $\begin{cases} \nabla f(x,y) - \lambda \nabla g(x,y) = 0 \\ g(x,y) = 0 \end{cases}$ 을 만족하는 점 (x, y)를 모두 구한다.
[3단계]	[2단계]에서 구한 모든 점에서 함수 f의 값을 구한다. 이 중 가장 큰 값이 최댓값, 가장 작은 값이 최솟값이 된다.

의 과정으로 구해집니다.

(출처: http://matrix.skku.ac.kr/math4ai/part4/)

PCA 의 응용사례

1. 얼굴인식

얼굴을 구성하는 n*m 크기의 이미지 파일들이 존재한다면, 이들은 n*m 차원의 벡터로써 생각할 수 있다. 이때 가지고 있는 여러개의 n*m 차원의 점 데이터들을 토대로 PCA 를 수행한다면 차원 수와 같은 주성분 벡터들을 구할 수 있고, 이를 다시 이미지로 해석한 것을 eigenface 라 하며 분산이 큰 순서대로 나타내면 아래와 같아진다.

이때 뒷부분의 주성분 벡터들은 노이즈성 정보를 나타내기 때문에 전반부 k 개의 주성분 벡터들만으로 원래 데이터를 표현한다면 노이즈가 제거된 상태의 데이터를 얻을 수 있다.

출처:

https://darkpgmr.tistory.com/110

PCA 실습

본 과정은 이상구 교수님의 예제 참고 사이트 PCA example - R -- Sage (skku.ackr)를 참조하였습니다. 사람들이 새 컴퓨터를 선택할 때 관심을 갖는 사항에 대해 척도가 7 점인 4 문항의 설문조사를 실시한 경우입니다.

1. 설문조사를 통하여 얻은 데이터의 행렬 생성

R 코드를 입력한 후, 실행 버튼을 눌러 결과를 확인하세요.

```r
# 데이터 입력, R 코드
Price <- c(6, 7, 6, 5, 7, 6, 5, 6, 3, 1, 2, 5, 2, 3, 1, 2)
Software <- c(5, 3, 4, 7, 7, 4, 7, 5, 5, 3, 6, 7, 4, 5, 6, 3)
Aesthetics <- c(3, 2, 4, 1, 5, 2, 2, 4, 6, 7, 6, 7, 5, 6, 5, 7)
Brand <- c(4, 2, 5, 3, 5, 3, 1, 4, 7, 5, 7, 6, 6, 5, 5, 7)
data <- data.frame(Price, Software, Aesthetics, Brand)
# 데이터가 잘 입력되었는지 확인한다.
data
```

실행(Evaluate) Syntax Highlighting

	Price	Software	Aesthetics	Brand
1	6	5	3	4
2	7	3	2	2
3	6	4	4	5
4	5	7	1	3
5	7	7	5	5
6	6	4	2	3
7	5	7	2	1
8	6	5	4	4
9	3	5	6	7
10	1	3	7	5
11	2	6	6	7
12	5	7	7	6
13	2	4	5	6
14	3	5	6	5
15	1	6	5	5
16	2	3	7	7

2. 데이터 행렬을 센터링해서 mydata 의 행렬 생성

```
# 표준화(centering, scaling) 작업
mydata <- scale(data, center = T, scale = T)  # 데이터 행렬을 센터링, scaling 한다.
mydata
```

```
           Price    Software Aesthetics      Brand
 [1,]  0.8485483 -0.04217745      -0.75 -0.3865961
 [2,]  1.3167129 -1.39185589      -1.25 -1.5112392
 [3,]  0.8485483 -0.71701667      -0.25  0.1757255
 [4,]  0.3803837  1.30750098      -1.75 -0.9489176
 [5,]  1.3167129  1.30750098       0.25  0.1757255
 [6,]  0.8485483 -0.71701667      -1.25 -0.9489176
 [7,]  0.3803837  1.30750098      -1.25 -2.0735607
 [8,]  0.8485483 -0.04217745      -0.25 -0.3865961
 [9,] -0.5559454 -0.04217745       0.75  1.3003686
[10,] -1.4922746 -1.39185589       1.25  0.1757255
[11,] -1.0241100  0.63266177       0.75  1.3003686
[12,]  0.3803837  1.30750098       1.25  0.7380470
[13,] -1.0241100 -0.71701667       0.25  0.7380470
[14,] -0.5559454 -0.04217745       0.75  0.1757255
[15,] -1.4922746  0.63266177       0.25  0.1757255
[16,] -1.0241100 -1.39185589       1.25  1.3003686
attr(,"scaled:center")
     Price   Software Aesthetics      Brand
    4.1875     5.0625     4.5000     4.6875
attr(,"scaled:scale")
     Price   Software Aesthetics      Brand
  2.136001   1.481834   2.000000   1.778342
```

3. Mydata 행렬에 PCA 진행

```
# PCA 진행. 앞에서 이미 센터링 scaling 하였음. retx는 PC score를 제공할지 여부
mypca <- prcomp(mydata, center = F, scale = F, retx = T)
summary(mypca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4
Standard deviation     1.5589 0.9804 0.6817 0.37926
Proportion of Variance 0.6076 0.2403 0.1162 0.03596
Cumulative Proportion  0.6076 0.8479 0.9640 1.00000
```

```
# loadings (V를 의미한다.)
mypca$rotation
```

```
                  PC1        PC2        PC3         PC4
Price      -0.5229138 0.00807487 -0.8483525  0.08242604
Software   -0.1771390 0.97675554  0.1198660  0.01423081
Aesthetics  0.5965260 0.13369503 -0.2950727  0.73431229
Brand       0.5825287 0.16735905 -0.4229212 -0.67363855
```

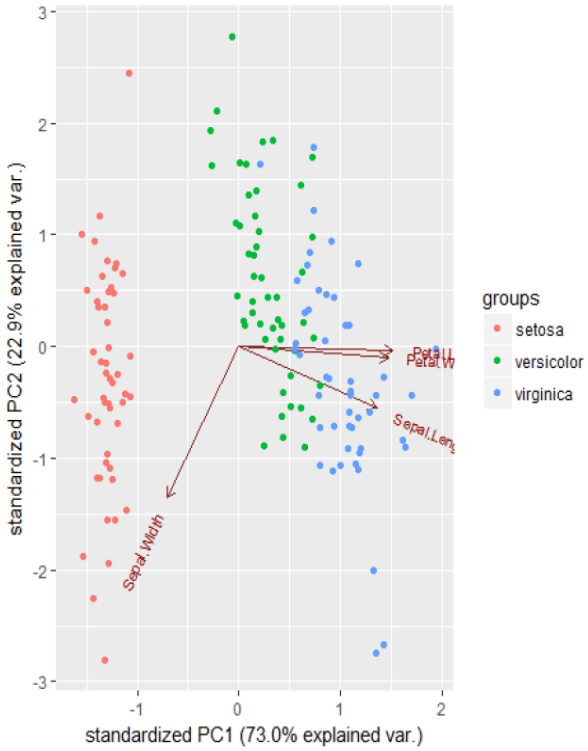
```
# PC1~PC3 score. 1열부터 3열까지
mypca$x[, 1:3]
```

	PC1	PC2	PC3
[1,]	-1.1088441	-0.19931676	-0.34011951
[2,]	-2.0679730	-1.76890900	-0.27589687
[3,]	-0.3634723	-0.69751259	-0.80636384
[4,]	-2.0272096	0.88740412	0.75172002
[5,]	-0.6686402	1.35057412	-1.10839793
[6,]	-1.6151352	-1.01942683	-0.03565573
[7,]	-2.3840835	0.76603244	1.07981912
[8,]	-0.8105812	-0.13246925	-0.48765585
[9,]	1.5030793	0.27221348	-0.30467593
[10,]	1.8749056	-1.17502483	0.65597978
[11,]	1.6283487	0.92758606	0.17338293
[12,]	0.7450737	1.57081802	-0.84695116
[13,]	1.2415980	-0.55167695	0.39695643
[14,]	0.8479424	0.08399428	0.17095950
[15,]	0.9197585	0.66873897	1.19372328
[16,]	2.2852328	-0.98302526	-0.21682424

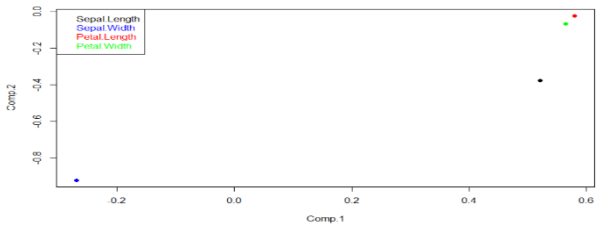
PCA 실습 (2)

구분	PC1	PC2	PC3	PC4
변수 비율	0.7296245	0.2285076	0.03668922	0.005178709

2D Score Plot을 그리면 아래 그림과 같습니다. PC1, PC2 두 개만으로도 전체 데이터 분산의 95.8%를 보존하고 있기 때문에 2차원으로 시각화해도 원 데이터의 모양을 파악하는 데 문제가 없습니다.



loading plot은 아래 그림과 같습니다. PC1 기준으로는 'Petal.Length', 'Petal.Width', 'Sepal.Length'가 비교적 중요한 변수임을 확인할 수 있습니다. PC2 입장에서는 'Sepal.Width'가 중요한 변수입니다.



위 분석에 사용한 R 코드는 아래와 같습니다.

```
library(rgl)
library(devtools)
install_github("vqv/ggbiplot")
library(ggbiplot)

# PCA
# cor = whether the calculation should use the correlation ma
trix or the covariance matrix
# score = whether the score on each principal component shoul
d be calculated
pcadata <- princomp(iris[,1:4], cor=T, scores=T)
# summary
plot(pcadata, type='l')
summary(pcadata)

# 2d score plot
ggbiplot(pcadata,groups = iris$Species)

# loading plot
plot(pcadata$loadings[,1:2], col=c('black','blue','red','gree
n'), pch=16)
legend('topleft',
         c('Sepal.Length','Sepal.Width','Petal.Length','Peta
l.Width'),
     text.col=c('black','blue','red','green'))
```

PCA 요약

주성분분석 = PCA = principal component analysis 는 고차원 데이터의 차원을 줄이는 차원축소의 기법 중 하나입니다. 원래 데이터의 분포를 최대한으로 보전하면서 서로 직교하는 새 축, 기저를 찾아 전차원 공간으로 변환하여 서로 선형연관성이 없는 새로운 변수인 주성분들을 만들어냅니다. 이때 고윳값분해 또는 특이값분해 SVD 를 사용합니다.

PCA 는 n*p 크기의 n=표본갯수, p=확률변수인 데이터 X 에서 센터링을 하는 것에서 시작합니다.

센터링은 xj 평균 = 1/n*(x1j+x2j+...+xnj)

X~ = X - [x1 평균, x2 평균, ... ,xp 평균]

[x1 평균, x2 평균, ... ,xp 평균]

.....

=[x11-x1 평균, x12-x2 평균, ... ,x1p-xp 평균]

[x21-x1 평균, x22-x2 평균, ... ,x2p-xp 평균]

.....

그래서 공분산 행렬은 ∑=[[var(x1), corr(x1,x2), ... ,Corr(x1,xn)], ...] = 1/n *(X~)^T(X~)로 나타낼수 있습니다.

(정사각행렬의 성분을 각 변수의 분산과 공분산으로 채운 것)

확률변수 {x1,x2, ... ,xp}가 센터링이 이미 되어있다고 해봅시다.

v1=열백터[v11,v21,v31, ... ,vp1]이 λ1 의 고유백터일 때 (단 v1 의 크기는 1), 확률변수 {x1,x2, ... ,xp}의 일차결합으로 만든 주성분

z1=x1*v11+x2*v21+x3*v31+ ... +xp*vp1 은 평균이 0 입니다.

그래서 z1 의 분산은 V(z1) = Var(x1*v11+x2*v21+x3*v31+ ... +xp*vp1)

=E((x1*v11+x2*v21+x3*v31+ ... +xp*vp1)²)

=E(∑(i=1->p)∑(j=1->p) vi1*vj1*xi*xj)

=∑(i=1->p)∑(j=1->p) vi1*vj1*E(xi*xj) (E(aX)=aE(X))

=∑(i=1->p)∑(j=1->p) vi1*vj1*σij (σij는 xi와 xj의 공분산)

=v1^T*∑*v1 (이차형식 xTAx)입니다.

왜 v1이 λ1의 고윳값이냐 하면 라그랑주 함수 L(x,λ)=f(x)-λg(x)를 이용하면 됩니다.

f(v1)=v1^T*∑*v1 , g(v1) = v1^T*v1-1=0 이고 L(v1,λ1)=f(v1)-λ1*g(v1)의 그래디언트가 0이므로 2*∑*v1 - 2*v1 =0, det(∑-λ1*I)=0 입니다.

따라서 λ1은 고윳값, v1은 λ1의 고유벡터가 됩니다.

λk는 공분산행렬 ∑의 k번째로 큰 고윳값이 되고 Var(zk)=λk가 됩니다.

또 [z1,z2, ... ,zp]=[x1,x2, ... ,xp]*V 이고

데이터 X를 SVD 분해를 했을때 X=USV라고 할때 (S는 X의 특이값이 큰 순서대로 있는 대각선 성분인 대각선 행렬입니다.)

$$X = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_k\ \mathbf{u}_{k+1}\ \cdots\ \mathbf{u}_n] \left[\begin{array}{cccc|ccc} s_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & s_k & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & 0 & s_{k+1} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & s_p \\ \hline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{array}\right]$$

$$= USV^T = U\begin{bmatrix} S_1 \\ O \end{bmatrix} V^T = \sum_{i=1}^{p} s_i \mathbf{u}_i \mathbf{v}_i^T$$

∑ = V*A*V^T =1/n*X^T*X =1/n*V*[S1^T:O]*U^T*U*[S1:O]*V^T

=V(S1²/n)V^T 로 나타낼 수 있습니다.

여기서 다섯가지를 유추해 낼 수 있습니다.

1) X=USV^T 이면 V 의 열백터가 주축이 됩니다.

2) Z=XV=USV^T*V=US 이므로 Z=US 의 열백터들이 PC score(주성분점수)가 됩니다.

3) 특이값 Si 는 공분산 행렬 ∑의 고윳값 λi 와 관계식 λi=Si²/n 이 됩니다.(이때 λi 는 대응하는 PC 의 분산)

4)데이터를 p 차원에서 k 차원으로 줄이기 위해 S 의 k 번째 선행 주 소행렬식을 택하면, Zk=UkSk=[u1,u2, ...][[s1, ...], ...]으로 처음 k 개의 주성분(PC)를 포함하는 n*k 행렬이 됩니다.

5) Zk 에 행렬 Vk^T 를 곱하면 Xk 를 복원할 수 있다. 이는 truncated SVD 라고 하며 X의 SVD 에서 크기 순서대로 k 개의 큰 특이값 s1 ≥s2≥...≥sk ≥0 과 이에 대응하는 U와 V 의 처음 k 개의 열벡터를 택하는 것과 같습니다.

그래서 sage 에서 실습을 해 보자면 X=[[1,2,3],[6,4,5],[8,9,7],[10,11,12]]를 주성분 분석을 해보았습니다.

우선 센터링을 해주자면 x1 평균=6.25, x2 평균=6.5, x3 평균=6.75

X~=X-[[6.25,6.5,6.75],[6.25,6.5,6.75],[6.25,6.5,6.75],[6.25,6.5,6.75]]= [[-5.25,-4.5,-3.75],[-0.25,-2.5,0.25],[1.75,2.5,0.25],[3.75,4.5,5.25]]

sage 에서 공분산행렬 ∑를 구해보면

X=matrix([[-5.25,-4.5,-3.75],[-0.25,-2.5,0.25],[1.75,2.5,0.25],[3.75,4.5,5.25]])

cov_matrix= 1/3 * X.transpose() * X

그래서 ∑=[[14.9166666666667 15.1666666666667 13.2500000000000],[15.1666666666667 17.6666666666667 13.5000000000000],[13.2500000000000 13.5000000000000 13.9166666666667]] 이 나옵니다.

cov_matrix.eigenvalues()

이것의 고유값은 [43.5742232741808, 2.16260385844664, 0.763172867372549] 가 나옵니다.

그리고 cov_matrix.eigenvalues()를 해서 고윳값의 고유벡터를 구해보면 V 를 구할 수 있습니다.

그리고 X=USV^T 로 특이값분해를 해보면

```
X=matrix([[-5.25,-4.5,-3.75],[-0.25,-2.5,0.25],[1.75,2.5,0.25],[3.75,4.5,5.25]])
b= (X.transpose() * X).eigenvalues()
c=[sqrt(i) for i in b]
S=diagonal_matrix(c)
print S 하면
```

S=[[11.4334014983531, 0.000000000000000 0.000000000000000],[0.000000000000000 2.54711828844677 0.000000000000000],[0.000000000000000 0.000000000000000 1.51311552834461]]이 나오는데

$11.4334014983531^2/3=43.5742232741808$, $2.54711828844677^2/3=2.16260385844664$, $1.51311552834461^2/3=0.763172867372549$ 이므로 $\lambda i=Si^2/n$ 인것을 알게 되었습니다.

그리고 그리고 특이값 sk 를 큰 값 순서대로 몇개만 뽑으면 차원데이터를 일부만 손실하면서 차원을 축소할 수 있음을 알게 되었습니다.

"PCA"

1. X변수 표준화 해주어야 $X \Rightarrow X_S$ (평균0) → $X_S^T X_S$ = X의 공분산행렬

2. $X_S^T X_S$ 의 eigenvalue, eigenvector 구하기

a. 왜 eigenvalue / eigenvector 인가?

- 데이터의 분산 (산포)을 최대화하는 α벡터를 찾기

$$= \underset{\alpha,\lambda}{\arg\max}\ \mathrm{Var}(X_S\alpha) - \lambda\|\alpha\|$$

→ α의 크기에 penalty

↳ X의 모든 선형결합

$$= \underset{\alpha,\lambda}{\arg\max}\ \alpha^T \mathrm{Var}(X_S)\alpha - \lambda\alpha^T\alpha$$

$$= \underset{\alpha,\lambda}{\arg\max}\ \alpha^T X_S^T X_S \alpha - \lambda\alpha^T\alpha$$

↳ 미분해서 풀면 $X_S^T X_S \alpha = \lambda\alpha$ 꼴이 됨

→ eigen decomposition

∴ 최대화하는 축 : α, 그때의 분산 : λ

b. eigen value decomposition $A\alpha = \lambda\alpha$ 인 α, λ 구하기

$(\lambda I - A)\alpha = 0$

$\equiv \det(\lambda I - A) = 0$ → λ먼저 구하고 $Aw_j = \lambda w_j$로 대응하는 eigenvector 구하기

(왜 행렬식을 0으로 두는지?)

$\Rightarrow X_S^T X_S = V\Lambda V^T$ 로 decompose (positive definite로 항상가능)

3. let $X_S V = Z$ $V^T\beta = \theta$

$Y = X_S\beta = X_S V V^T \beta = Z\theta$ $\Rightarrow$ $V^T\beta = \theta$ $\left[\begin{matrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{matrix}\right]$

$\boxed{\beta = V\theta}$ 이용해서 β를 구할 수 있음

↳ $\hat{\beta}_{j(M)}^{PC} = \sum_{m=1}^{M} \hat{\theta}_m v_{mj}$

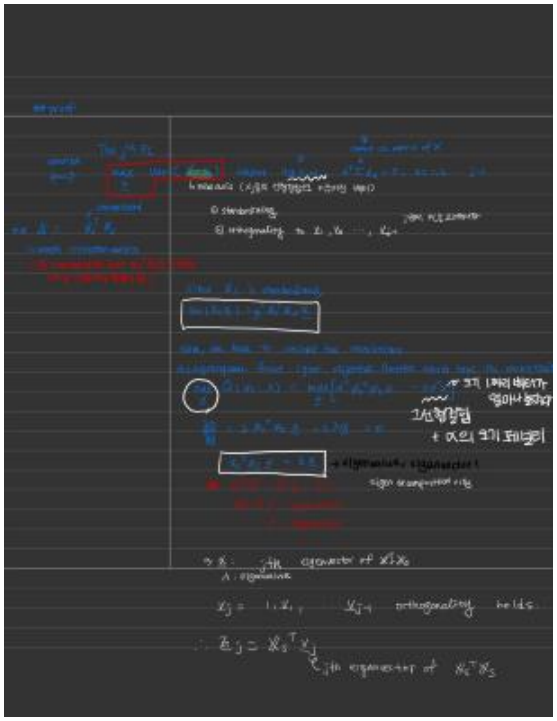
4. computations

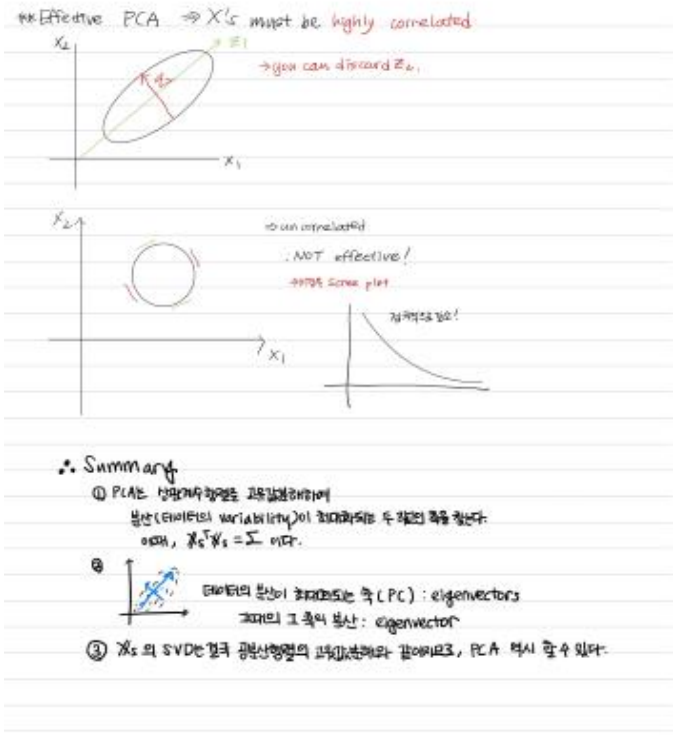
- Z_j : the jth PC

: X_S 와 V_j → 주어진 X를 축변환으로 돌린것

- $\hat{y}^{PCR} = \bar{y} + \sum_{m=1}^{M} \hat{\theta}_m Z_m$, $\hat{\theta}_m = \frac{Z_m^T y}{Z_m^T Z_m}$ (LSE)

- $\hat{\beta}_{j(M)}^{PC} = \sum_{m=1}^{M} \hat{\theta}_m Z_{mj}$ ← m번째 eigenvector의 j번째 원소.





그래서 정리해보면

센터링은 X~

=[x11-x1 평균, x12-x2 평균, ... ,x1p-xp 평균]

[x21-x1 평균, x22-x2 평균, ... ,x2p-xp 평균]

..... 입니다.

확률변수 x1 의 일차결합 z1=x1*v11+x2*v21+x3*v31+ ... +xp*vp1 는 평균이 0, 분산이 v1^T*∑*v1 입니다.

λk 는 공분산행렬 ∑의 k 번째로 큰 고윳값이 되고 Var(zk)=λk 입니다. 그리고 v1 은 λ1 의 고유벡터입니다.

데이터 X를 SVD 분해를 했을때 X=USV 라고 할때

∑ =1/n*X^T*X =V(S1²/n)V^T 입니다. 그래서 V 의 열백터가 주축이 되고 λi=Si²/n 이며 Z=US 의 열백터들이 PC score 가 된다는 것을 알 수 있습니다.

그리고 데이터를 p차원에서 k 차원으로 줄이기 위해 S의 k번째 선행 주 소행렬식을 택하면, Zk=UkSk=[u1,u2, ...][[s1, ...], ...]으로 처음 k개의 주성분(PC)를 포함하는 n*k행렬이 되며, 거기에 Zk에 행렬 Vk^T를 곱하면 Xk를 복원할 수 있습니다. 이는 truncated SVD라고 하며 X의 SVD에서 크기 순서대로 k개의 큰 특이값 s1 ≥s2≥...≥sk ≥0 과 이에 대응하는 U와 V의 처음 k개의 열벡터를 택하는 것과 같습니다.

PCA 가 유독 이해가 잘 가지 않아 하루종일 공부한 후 학우분들이 정리한 관련 파일을 보게 됐습니다.

일목요연하게 정리가 잘 되어 있어 이해하는 데 큰 도움이 되었습니다. 특히 정규직교를 할 수 있기에 차원축소를 하면서 원 데이터를 최대한 보존할 수 있다는 설명에 막혔던 부분이 뚫리는 느낌이었습니다. 제가 교안을 공부하면서 놓친 부분인가 확인해 보았습니다. 하지만 교안에서 찾아볼 수 없는 것으로 보아 제가 강의에서 놓친 것이거나 류재헌 학우가 공부하다 얻은 통찰일 것이라 생각합니다.

명석한 학우들 덕분에 하루종일 매달리며 공부한 PCA 가 직관적으로 이해되서 기분이 좋았습니다.

아래 보고서를 읽다보니 저와 같이 라그랑주 승수법에 대해 잘 모르는 분들은 PCA step 들이 가지는 직관적 부분이 이해가 잘 가지 않을 것 같아, 라그랑주 승수법에 대한 설명을 덧붙였습니다.

또한, 혼자서 요약하며 공부한 필기록을 첨부합니다.

이상구(LEE SANGGU) 8 월 13 일 오전 00:16

Good job^^ 권서영씨, http://matrix.skku.ac.kr/math4ai/PCA/ 을 미리 보셨으면 더 쉽게 이해했을 것입니다.

류재헌(2015####86) 8 월 13 일 오전 00:45

도움이 되어서 다행입니다. 저도 PCA 의 원리와 과정등이 잘 이해가 되지 않아서 교수님의 첨부자료와 인터넷 자료, 영상 강의들을 여럿 읽어봤습니다. 아직도 100 퍼센트 이해가 됐다고 말할 수 없는 것 같습니다. https://www.youtube.com/watch?v=YEdscCNsinU 이 자료가 시각적으로 PCA 의 원리를 이해하는데 도움이 되었습니다. 참고가 되었으면 좋겠습니다.

(추가) PCA에 대한 추가적인 이해

PCA를 공부하면서, 큰 고유값을 가지는 고유벡터를 선택하는 이유가 궁금했고, 그것에 대해서 조사했습니다. 결론은 고유값에 해당되는 분산이 클수록 데이터의 특성을 잘 나타내기 때문에, 그 고유값에 해당되는 고유벡터를 선택한다는 것이었습니다.

고유값과 고유벡터에 대해서 다시 한 번 복습해 보겠습니다.

정의. 고윳값(eigenvalue), 고유벡터(eigenvector)

A를 n차의 정사각행렬이라 하자. $\mathbf{0}$ 아닌 벡터 $\mathbf{x}\in\mathbb{R}^n$가 적당한 스칼라 λ에 대하여 다음을 만족하면 λ를 A의 고윳값(eigenvalue)이라 하고, $\mathbf{x}$를 λ에 대응하는 A의 고유벡터(eigenvector)라고 한다.

$$A\mathbf{x}=\lambda\mathbf{x}$$

- $\mathbf{x}\in\mathbb{R}^n$가 고윳값 λ에 대응하는 A의 고유벡터이면 영 아닌 임의의 스칼라 k에 대하여 $k\mathbf{x}$도 λ에 대응하는 A의 고유벡터가 된다.

$$A\mathbf{x}=\lambda\mathbf{x} \Rightarrow A(k\mathbf{x})=k(A\mathbf{x})=k(\lambda\mathbf{x})=\lambda(k\mathbf{x})$$

우리는 6주동안 선형대수학에 대해 공부했기 때문에, 이 공식에 대해서는 아주 잘 알고 있다고 말할 수 있습니다. 하지만 이것이 기하학적으로 갖는 의미는 무엇일까요?

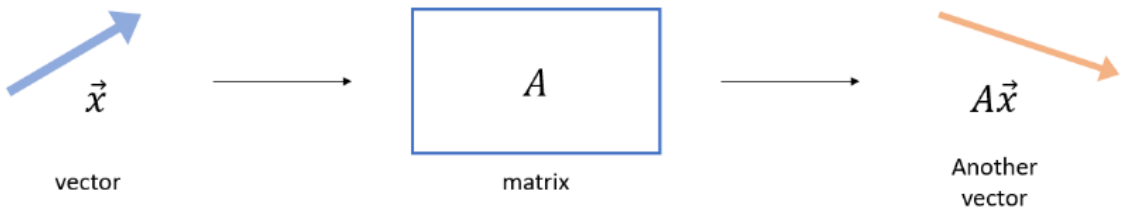
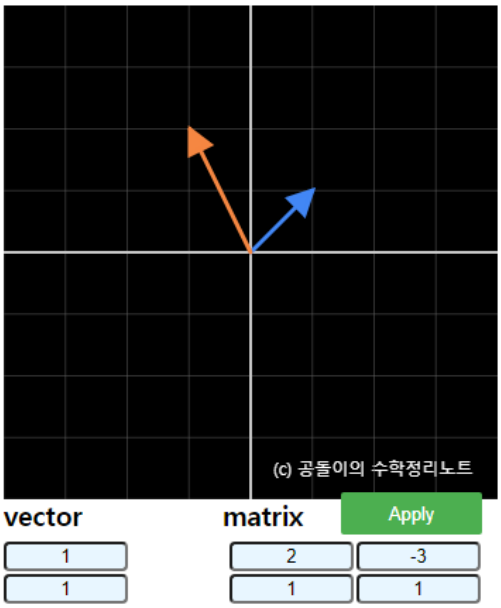


그림 1. 행렬은 벡터를 변환시켜주는 연산자이다.

그림 1에서 볼 수 있듯이 행렬을 이용해 벡터를 변환 시켜 주면, 변환 후의 벡터($A\vec{x}$)는 변환 전의 벡터($\vec{x}$)와 비교했을 때, 크기도 방향도 모두 변할 수 있다. 아래의 애플릿을 이용해 임의의 벡터와 행렬을 이용한 선형 변환 결과를 확인해보자.



<그림 1> 선형대수학에서 행렬이 갖는 의미, 공돌이의 수학정리노트

행렬은 벡터를 변환시켜주는 역할을 합니다. 하지만 어떤 벡터의 경우에는, 이 벡터에 행렬을 곱한 값이 상수배를 곱한 값과 같은 경우가 있을 겁니다.

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

을 입력으로 넣어주고, matrix에는

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

를 입력으로 넣어준 뒤 결과를 확인해보자.

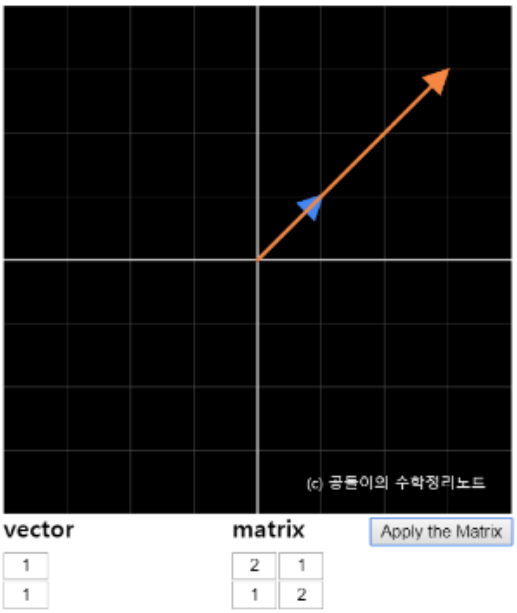


그림 2. 어떤 벡터와 행렬은 변환시키면 평행하지만 크기만 바뀐 벡터를 출력한다.

즉 입력 벡터 $\vec{x}$를 A로 선형변환 시킨 결과($A\vec{x}$)가 상수배라는 것이다.

$$A\vec{x} = \lambda\vec{x}$$

<그림 2> 고유값과 고유벡터의 기하학적 의미, 공돌이의 수학정리노트

이 경우 상수가 고유값을, 벡터가 고유벡터를 의미합니다. 여기서 중요한 것은, 고유벡터가 이 행렬의 축(Axis)를 담당한다는 것입니다.

고유 벡터(eigenvector)의 의미를 잘 생각해보면, 고유 벡터는 그 행렬이 벡터에 작용하는 주축(principal axis)의 방향을 나타내므로 공분산 행렬의 고유 벡터는 데이터가 어떤 방향으로 분산되어 있는지를 나타내준다고 할 수 있다.

고유 값은 고유벡터 방향으로 얼마만큼의 크기로 벡터공간이 늘려지는 지를 얘기한다[1]. 따라서 고유 값이 큰 순서대로 고유 벡터를 정렬하면 결과적으로 중요한 순서대로 주성분을 구하는 것이 된다. 아래의 그림 5에서 고유벡터 두 개를 볼 수 있고 각각의 벡터의 크기가 각 벡터의 고윳값을 의미한다.

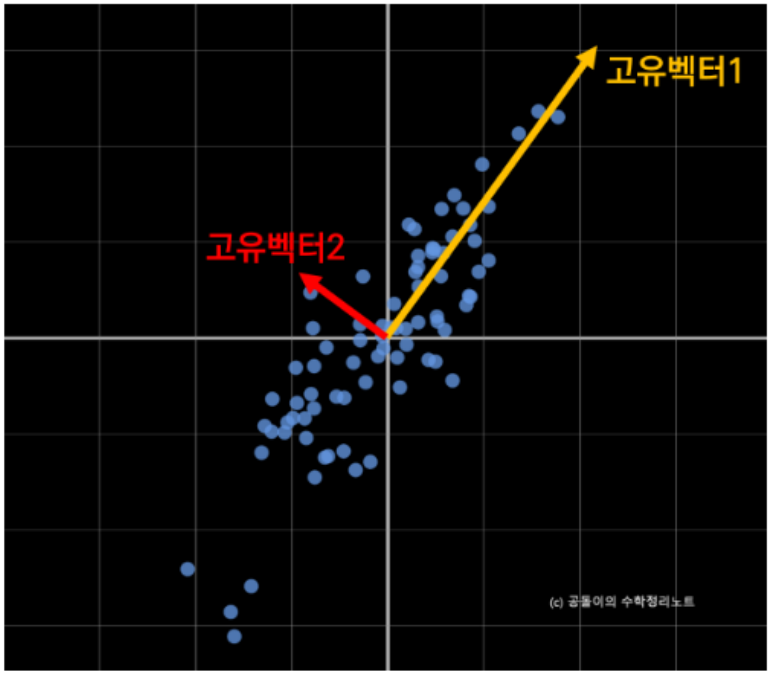
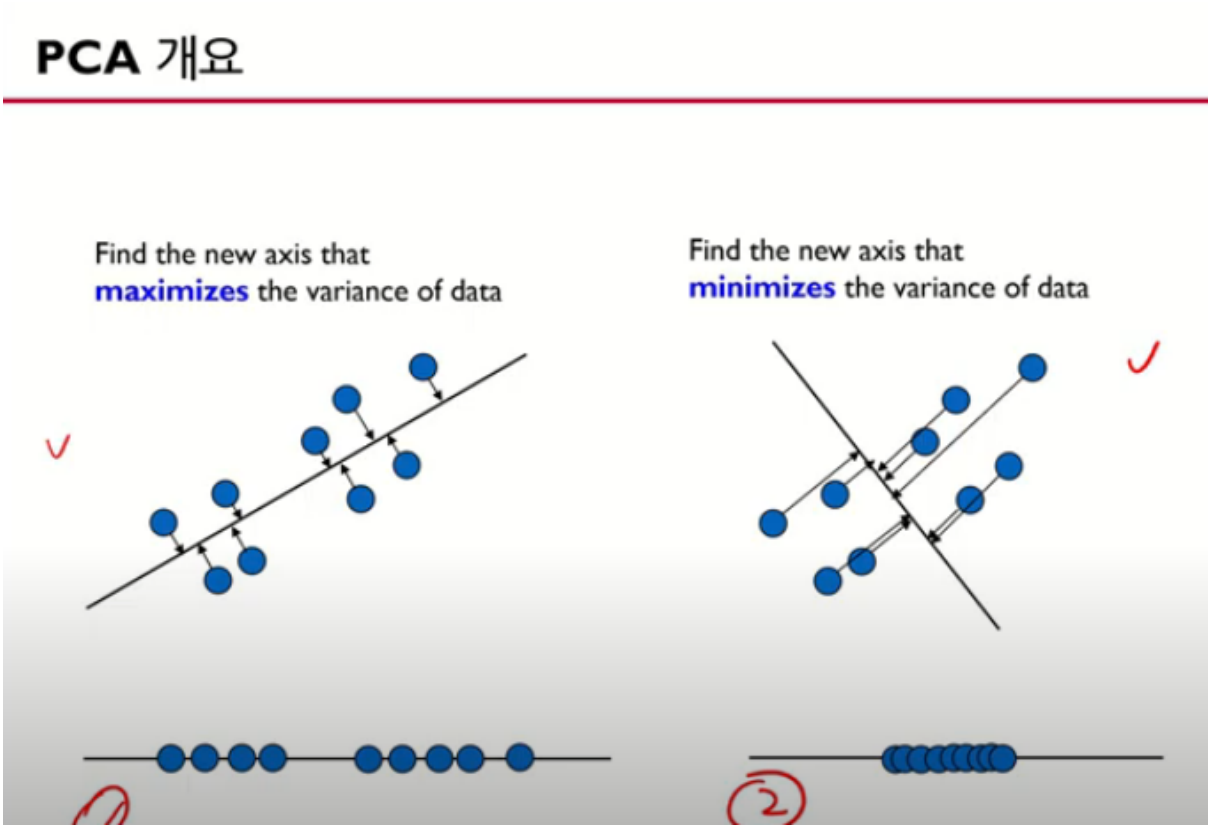


그림 5. 공분산행렬 Matrix 1의 고유벡터

즉, 우리가 고민했던 문제는 무엇인가? 바로 '데이터 벡터를 어떤 벡터에 내적(혹은 정사영)하는 것이 최적의 결과를 내주는가?' 이다. 이 문제에 대한 해결은 바로 공분산 행렬의 고윳값과 고유 벡터를 구함으로써 가능하다.

<그림 3> 행렬의 주축, 공돌이의 수학정리노트

그렇다면 여러 개의 고유벡터가 있을 때, 어떤 벡터에 데이터를 정사영하는 것이 최적의 결과를 내줄까요? 그림 4를 통해 알아보겠습니다.



<그림 4> 주축의 분산, 김성범[소장/인공지능연구소] 유튜브

그림 4는 동일한 데이터에서 축을 어떻게 설정할 지에 대한 문제를 나타낸 그림입니다. 왼쪽 그림과 오른쪽 그림을 비교해 보면, 왼쪽 그림의 축에 나타난 데이터가 오른쪽 그림에 비해서 넓게 퍼져있는 것을 볼 수 있습니다. 이것은 데이터의 특성을 왼쪽의 축이 오른쪽의 축보다 잘 나타낸다고 할 수 있습니다. 동시에 왼쪽의 데이터가 더 퍼져있기 때문에 분산이 크다는 것 역시 알 수 있습니다. 정리하자면, 분산이 클수록 데이터의 특성을 더 잘 나타내는 축이며, 이것이 주성분 벡터를 찾을 때 분산이 큰(고유값이 큰) 벡터를 선택하는 이유라고 할 수 있겠습니다.

< 참고문헌 >

1. 이상구, MATH4AI, BigBook
2. 주성분 분석 (skku.ac.kr)
3. [Linear Algebra] Lecture 17-(1) 직교행렬(Orthogonal Matrices)과 그람 슈미트 과정(Gram-Schmidt Process) :: Learn Again! 러너게인 (tistory.com)

4. Loner의 학습노트 :: 공분산 행렬과 주성분 분석(PCA) (tistory.com)
5. [핵심 머신러닝] Principal Component Analysis (PCA, 주성분 분석) - YouTube
6. 주성분 분석(PCA) - 공돌이의 수학정리노트 (angeloyeo.github.io)