Principal Component Analysis

Principal Component Analysis (PCA)는 고차원 공간상에 분포하는 데이터들의 주축(major axes)를 찾아주는 역할을 한다. 이는 적은 수의 파라미터를 이용하여 데이터를 표현하는데 유용하다.

■ Algorithm

1. 데이터의 평균(mean), $\bar{\mathbf{x}}$,을 계산한다.

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

2. 데이터의 공분산행렬(covariance matrix), $\mathbf{C}_x$,을 구한다.

$$\mathbf{C}_x = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

3. $\mathbf{C}_x$ 의 고유벡터(eigenvector), $\mathbf{v}_i$,와 고유값(eigenvalue), λ_i ,를 구한다. 이 때, 고유값의 크기가 큰 순서대로 정렬한다 ($\lambda_i \geq \lambda_{i+1}$).
4. 각각의 eigenvalue는 해당하는 eigenvector 방향으로 축을 변환하였을 때, 데이터의 분산(variance)값을 의미한다. 그러므로, 데이터의 전체 분산을 다음과 같이 구한다.

$$V_T = \sum_t \lambda_i$$

5. 구해진 eigenvector와 eivenvalue 중에서 다음의 조건을 만족하는 상위 t 개만을 사용한다.

$$\sum_{i=1}^{t}\lambda \geq f_v V_T$$

여기서 f_v는 표현하고 싶은 데이터의 양을 결정하는 비례상수이다. (예를 들어, 전체 데이터의 98%를 표현하려고 하면 $f_v = 0.98$)

■

■ Example

2차원 공간상에서 6개의 점들이 오른쪽 그림과 같은 형태로 분포되어있다.

$$\mathbf{x}_1=\begin{bmatrix}0\\0\end{bmatrix},\mathbf{x}_2=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{x}_3=\begin{bmatrix}2\\1\end{bmatrix},\mathbf{x}_4=\begin{bmatrix}1\\2\end{bmatrix},\mathbf{x}_5=\begin{bmatrix}2\\2\end{bmatrix},\mathbf{x}_6=\begin{bmatrix}3\\3\end{bmatrix}$$

이 점들의 평균 벡터를 구하면 $\bar{\mathbf{x}}=\begin{bmatrix}1.5 & 1.5\end{bmatrix}^T$ 가 되고, 이에 해당하는 covariance matrix 를 구하면 다음과 같다.

$$\mathbf{C}_x=\begin{bmatrix}1.1 & 0.9\\0.9 & 1.1\end{bmatrix}$$

이 covariance matrix 를 eigen-analysis 를 통하여 eigenvector와 eigenvalue를 구하면 다음과 같이 나온다.

$$\lambda_1=2.0,\mathbf{v}_1=\begin{bmatrix}0.7071\\0.7071\end{bmatrix},\ \lambda_2=0.2,\mathbf{v}_2=\begin{bmatrix}-0.7071\\0.7071\end{bmatrix}$$

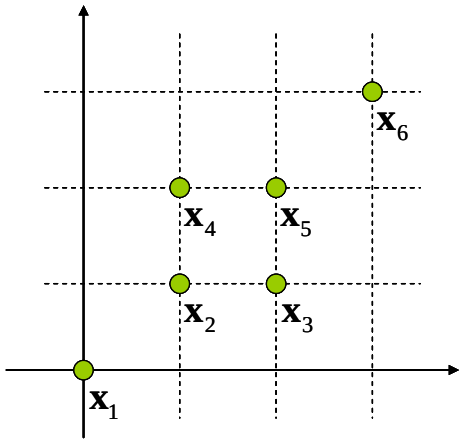

그림 1. 데이터 분포

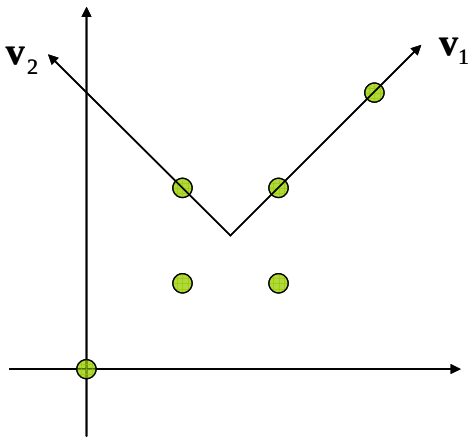

그림 2. 데이터의 주축 벡터

(검증)

주축 $\mathbf{v}_1$ 방향으로 데이터들의 거리를 구하면, $\left(-\frac{3}{2}\sqrt{2},-\frac{1}{2}\sqrt{2},0,0,\frac{1}{2}\sqrt{2},\frac{3}{2}\sqrt{2}\right)$ 이 되고, 이들의 분산을 구하면 다음과 같다.

$$\text{var}=\frac{1}{5}\left(\frac{9}{2}+\frac{1}{2}+0+0+\frac{1}{2}+\frac{9}{2}\right)=2$$

같은 방법으로 $\mathbf{v}_2$ 방향으로의 데이터들의 분산을 구하면 0.2가 나온다.

■ Matlab Code

```
% Principal Component Analysis Test Code
x = [0 0; 1 1; 2 1; 1 2; 2 2; 3 3]';
[m, n] = size(x);
mx = mean(x, 2);
Cx = zeros(m, m);
for i = 1:6
    Cx = Cx + (x(:, i)-mx) * (x(:, i)-mx)';
end
Cx = Cx/(n-1);
[V, D] = eig(Cx)
```

(a) 공식대로 모두 계산한 코드

```
% Principal Component Analysis Test Code
x = [0 0; 1 1; 2 1; 1 2; 2 2; 3 3]';
Cx = cov(x');
[V, D] = eig(Cx)
```

(b) 내장 함수를 이용한 코드

그림 3. Example에 대한 Matlab 코드

실제 코드를 실행하면 V와 D가 다음과 같이 계산되어진다.

V =

-0.7071	0.7071
0.7071	0.7071

D =

0.2000	0
0	2.0000

여기서 D 행렬의 diagonal elements 가 eigenvalue 값을 나타내며, 최대 eigenvalue는 두 번째 벡터에 해당하므로, $\mathbf{v}_1 = \begin{bmatrix} 0.7071 & 0.70701 \end{bmatrix}^T$ 가 된다.

만든이: 황선규
E-mail: kkokkal@vision.hanyang.ac.kr
URL: http://vision.hanyang.ac.kr/~kkokkal/