

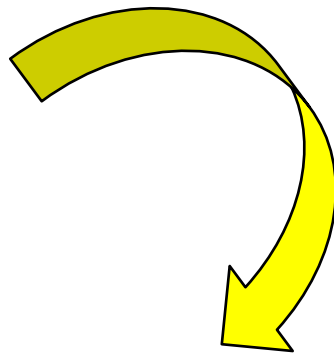
# Principal Component Analysis (PCA)

Jeonghun Yoon

# 데이터의 단순화

데이터를 분석, 사용할 때,

- 사용하기 쉬운 데이터 집합 구축
- 많은 알고리즘의 계산 비용 축소
- 노이즈 제거
- 이해하기 쉬운 결과 도출



데이터의 단순화

① 차원 축소 (dimensional reduction)

- 데이터를 표현하는 속성의 수를 축소

② 요소 분석 (factor analysis)

- 관찰 가능한 데이터 = 잠재적인 변수(latent variable)와 noise의 선형결합

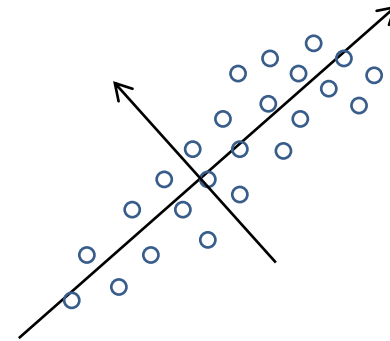
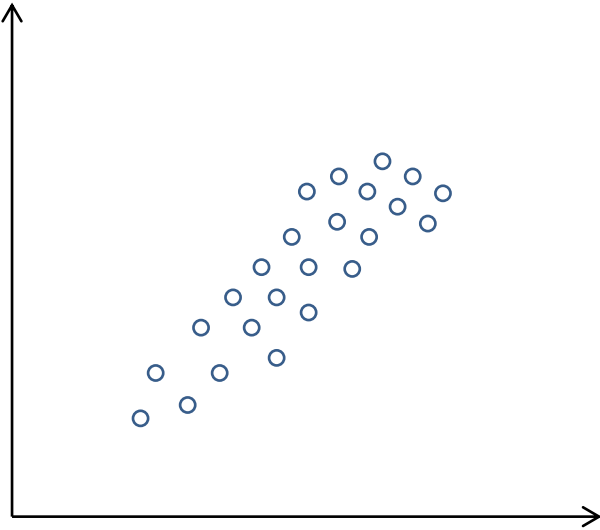
# Motivation

Question 1)

데이터 집합이 다음과 같다.

어떤 것이 더 나은 표현인가?

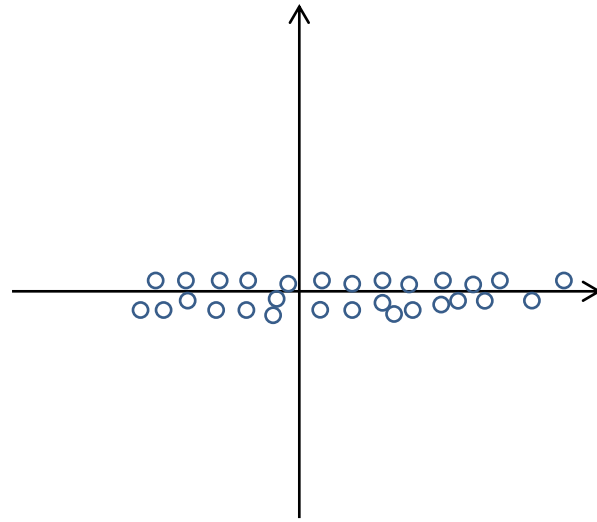
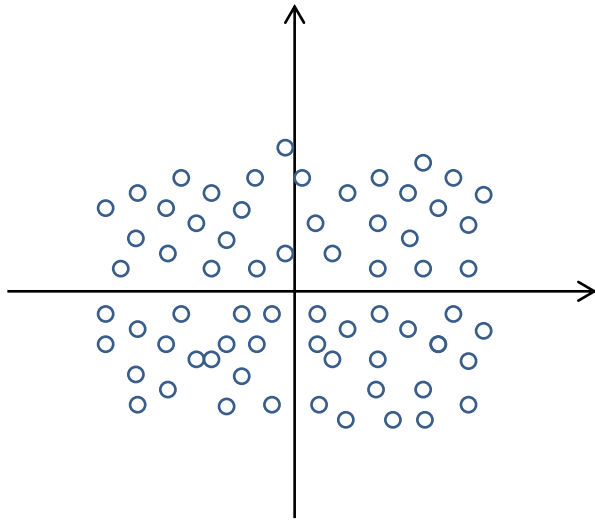
더 나은 표현이라는 것은 무슨 의미인가?



# Motivation

Question 2)

데이터 집합을 표현할 때,  $x$ 축  $y$ 축이 모두 필요한가?



# Dimensional Reduction

## PCA가 사용되는 예 : **Dimensioal Reduction**

우리는 많은 변수(variables)를 가지고 있는 데이터들을 다루게 되는 상황이 많다. 이러한 상황에서, 각각의 변수나 변수의 부분 집합들은 서로 연관되어 있는 경우 (highly correlated)가 많다.

이런 데이터의 집합을 가지고, 예측 모델(prediction model)이나 분류(classification) 모델을 만들 경우 필요가 있을까? 변수들이 서로 연관되어 있고(highly correlated) 우리가 흥미 있어 하는 결과와 상관없는 변수들이 존재할 수 있는 상황임을 상기하자.

# Dimensional Reduction

불필요한(서로 연관되어 있거나, 결과와 상관없는) 변수들은, 변수들을 모으고 분석하는데 드는 비용을 증가시켜서, 예측 모델 또는 분류 모델을 만들 때 전체 비용을 증가시키는 원인이 된다.

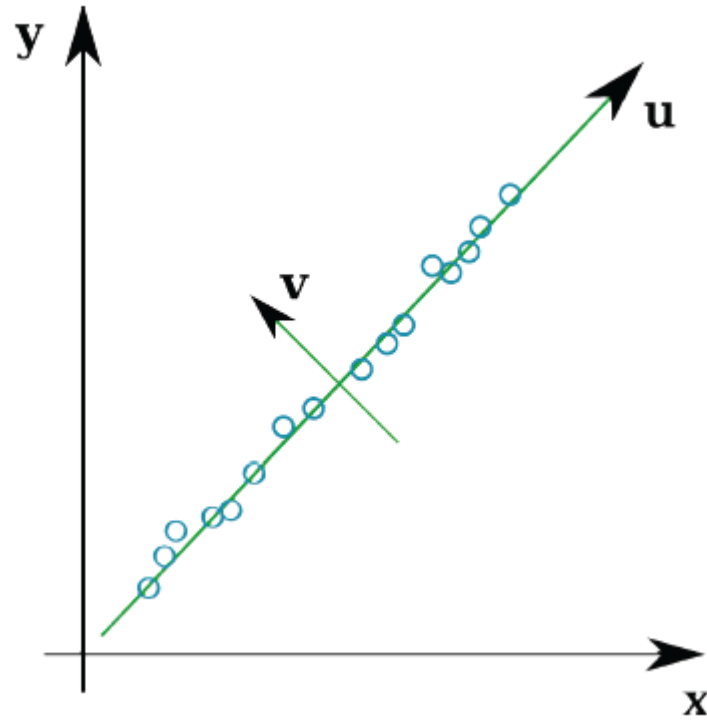
따라서 불필요한 변수들을 줄여 주는 것에 대한 필요성이 제기된다.

모델의 차원(dimensionality)은 모델에 사용되는 독립(independence) 변수 또는 입력(input) 변수의 개수(number)를 의미한다.

따라서, Machine Learning 영역에서는 본래 모델이 가지고 있던 성질이나 정확도를 최대한 유지하면서 차원을 줄이는 방법을 통하여 위에서 설명된 문제점을 해결하려고 한다.

# Dimensional Reduction

$X, Y?$



$u?$

# Dimensional Reduction

**Dimensional Reduction**의 핵심 아이디어는, 상관도가 높은(interrelated, correlated) 변수들이 많이 존재하는 데이터 집합의 차원(dimensionality)을 줄이면서, 동시에 데이터 집합에 존재하고 있는 차이(variation, 정보)를 최대한 유지하는 것이다. 즉, 차원을 줄이되 “정보 손실을 최소화”하는 것이다.

여기서 말하는 “정보”란 데이터 간의 거리, 상대적인 위치 관계 등을 의미한다.

차원이 줄어들기 때문에 정보의 손실은 어느 정도 감수해야 한다.



# Dimensional Reduction

D.R에서는 원래 공간에서 데이터가 퍼져 있는 정도를 변환된(축소된) 공간에서

얼마나 잘 유지하느냐를 척도로 삼는다. 원래 공간의 정보가 변환된 공간에서

얼마나 잘 유지하는지는 **변환된 공간에서의 데이터의 분산**으로 측정한다.

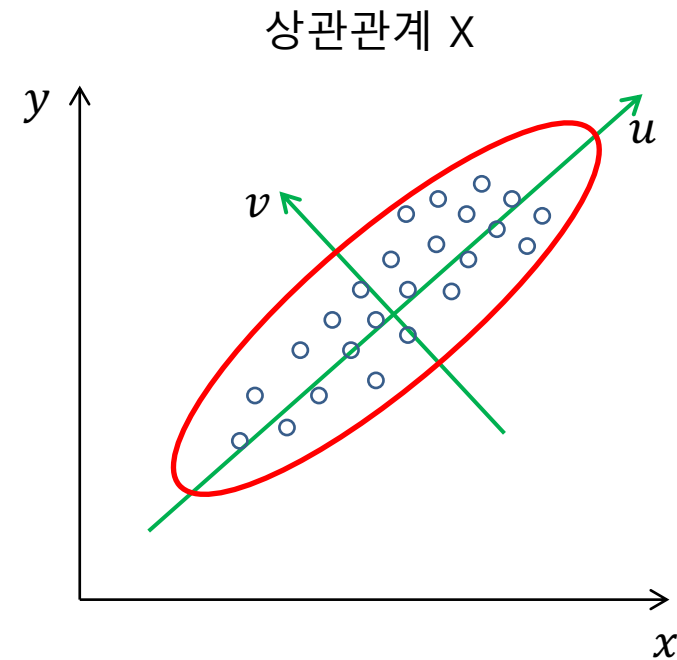
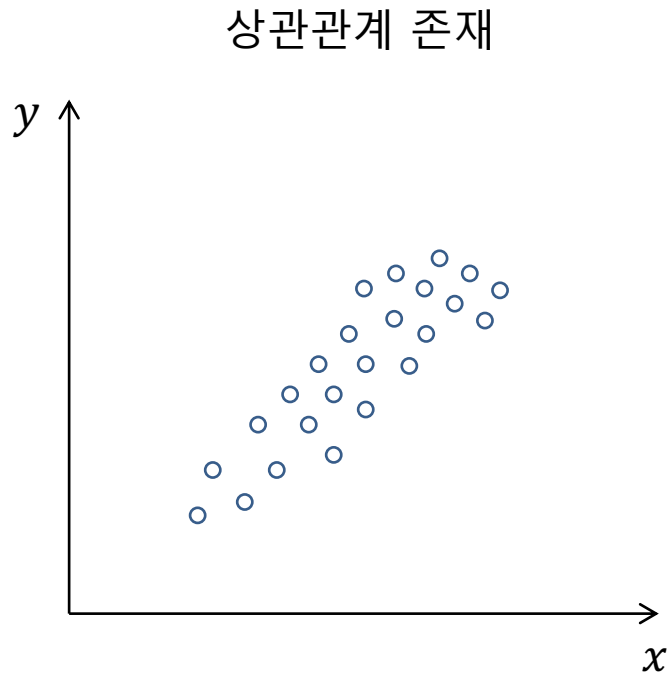
따라서, **변환된 공간에서 데이터의 분산을 최대로 할 수 있는 좌표축**을 찾아야 한다.

원래 공간에서, 데이터가 분산되어 있는 주요한 방향(Principal direction)을 찾자.

## Principal Component Analysis

# Data Representation

PCA가 사용되는 예 : **Data Representation**



Data가 분산 되어 있는 주요한 방향(Principal direction)을 찾아준다.

**Principal Component Analysis**

# 데이터가 분산되어 있는 주요한 방향 찾기(2차원)

Step 1) 데이터를 투영(Projection)하기

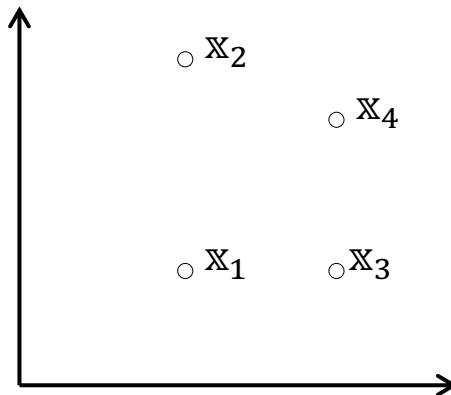
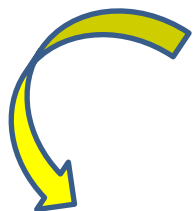
Step 2) 투영된 공간에서 분산 측정하기.

Step 3) 분산의 최대치는 어떻게 찾는가?

※ 첫 번째 principal direction을 찾는 것에 초점을 맞추어 설명

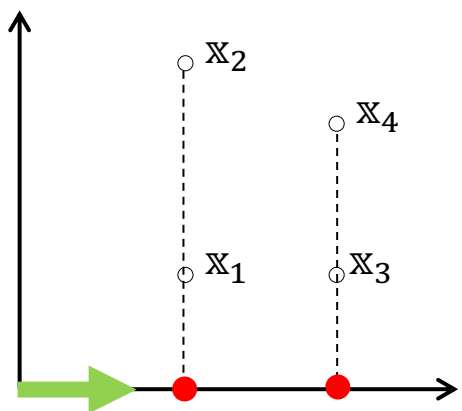
# 데이터를 투영(Projection)하기

1-dimensional  
공간으로  
투영

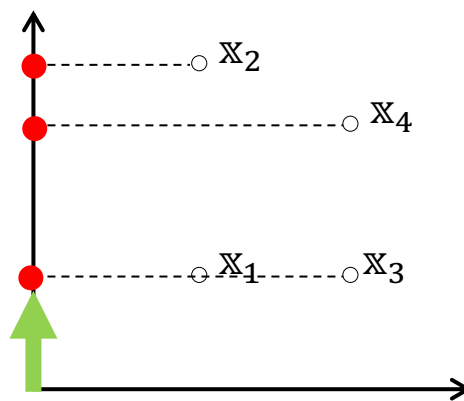


원래의 공간

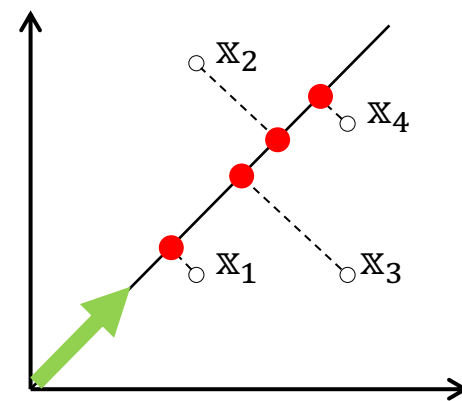
dimensionality : 2



$w^T = (1,0)$ 으로 투영



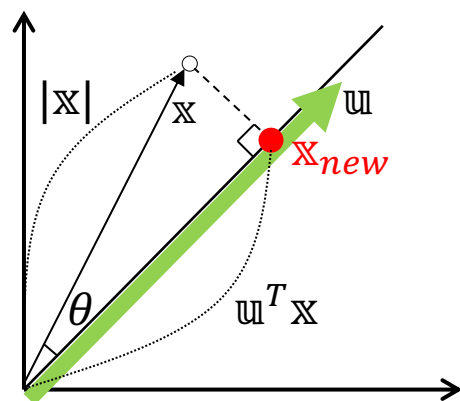
$w^T = (0,1)$ 으로 투영



$w^T = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ 으로 투영

# 데이터를 투영(Projection)하기

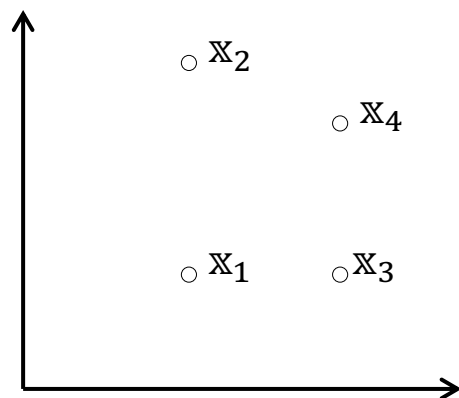
투영된 차원에서 데이터들은 어떤 값을 가질까?



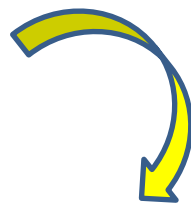
$$\mathbf{x}_{new} = \mathbf{w}^T \mathbf{x}$$

$$(\because \mathbf{w}^T \mathbf{x} = |\mathbf{w}| \cdot |\mathbf{x}| \cos \theta = |\mathbf{x}| \cos \theta)$$

# 데이터를 투영(Projection)하기



원래의 공간 , dimensionality : 2

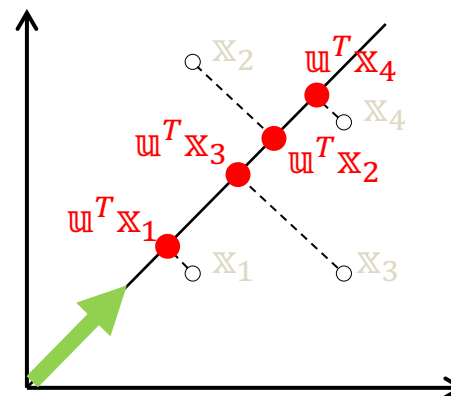


1-dimensional 공간으로 투영

$\mathbf{w}^T = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ 으로 투영

변환된 공간에서의 데이터

:  $\mathbf{w}^T \mathbf{x}_1, \mathbf{w}^T \mathbf{x}_2, \mathbf{w}^T \mathbf{x}_3, \mathbf{w}^T \mathbf{x}_4$



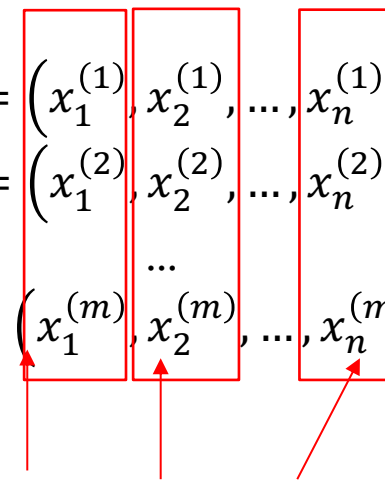
# 투영된 공간에서 분산 측정하기

먼저, PCA를 실행하기 전에 데이터의 평균(mean)과 분산(variance)을 정규화(standardization) 해 준다.

(Pre-process the data)

데이터는 특성 벡터로 표현되는데, 특성 벡터의 각 원소들에서 각각의 평균과 빼 주고, 분산의 제곱근으로 나누어 준다.

예를 들어, 다음과 같은 데이터의 집합이 있을 때,

$$\begin{aligned} \mathbb{X}_1 &= (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}) \\ \mathbb{X}_2 &= (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}) \\ &\dots \\ \mathbb{X}_m &= (x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)}) \end{aligned}$$


이 값들의 평균과 분산을 구한다.  
각각의 값에서 평균을 빼고 분산의 제곱근으로 나눈다.

# 투영된 공간에서 분산 측정하기

정규화 과정에서

- 데이터에서 평균을 빼는 것 : 데이터의 평균이 0이 되도록 만든다.
- 데이터에서 분산의 제곱근을 나누어 주는 것 : 데이터의 값 들이 unit variance를 갖게 해 준다.

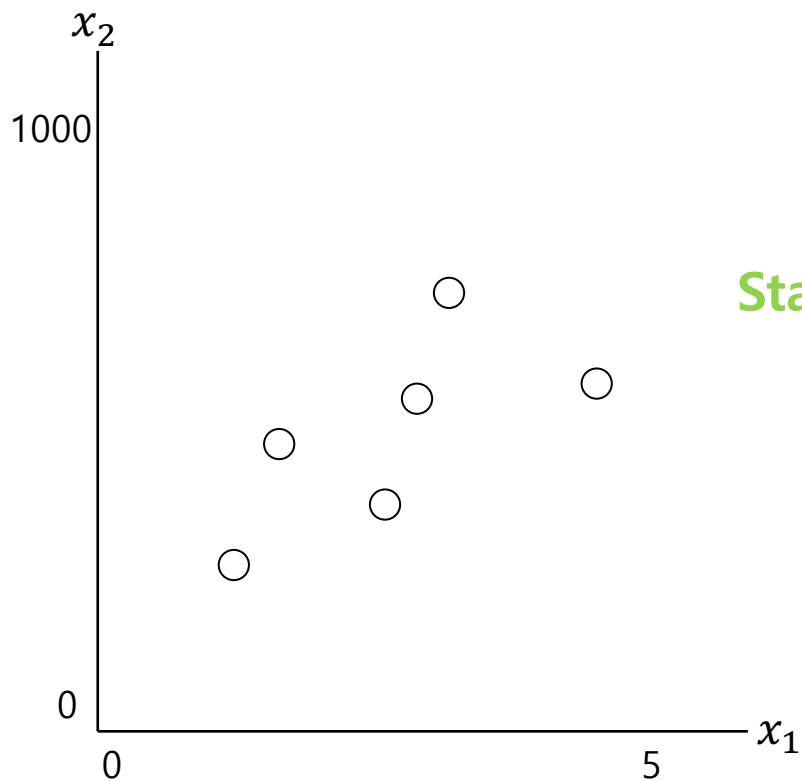
다음의 예를 살펴 보자.

$$\begin{array}{l} \text{분산 : 6} \quad \text{분산 : 60000} \\ \mathbb{x}_1 = (4, 900)^T \\ \mathbb{x}_2 = (7, 1200)^T \\ \mathbb{x}_3 = (10, 1500)^T \end{array} \quad \longrightarrow \quad \begin{array}{l} \text{분산 : 1} \quad \text{분산 : 1} \\ \mathbb{x}_1 = \left(-\frac{3}{\sqrt{6}}, -\frac{3}{\sqrt{6}}\right)^T \\ \mathbb{x}_2 = (0, 0)^T \\ \mathbb{x}_3 = \left(\frac{3}{\sqrt{6}}, \frac{3}{\sqrt{6}}\right)^T \end{array}$$

각각의 attribute의 평균이 0이 되고, 분산이 1이 된다. 즉 같은 “scale”을 가지게 되어, attribute간의 비교가 가능 해진다.

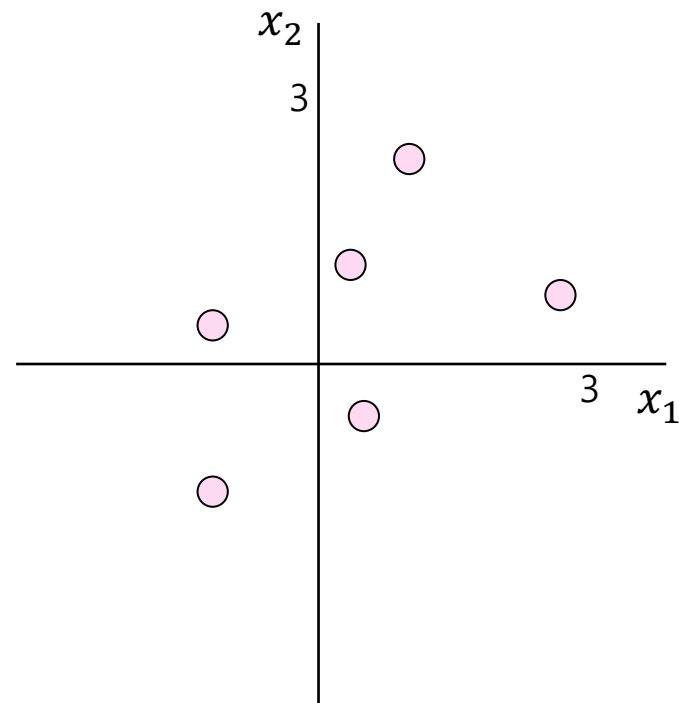


# 투영된 공간에서 분산 측정하기



attribute  $x_1$ 과  $x_2$ 가  
서로 다른 scale을 가질 수 있다.

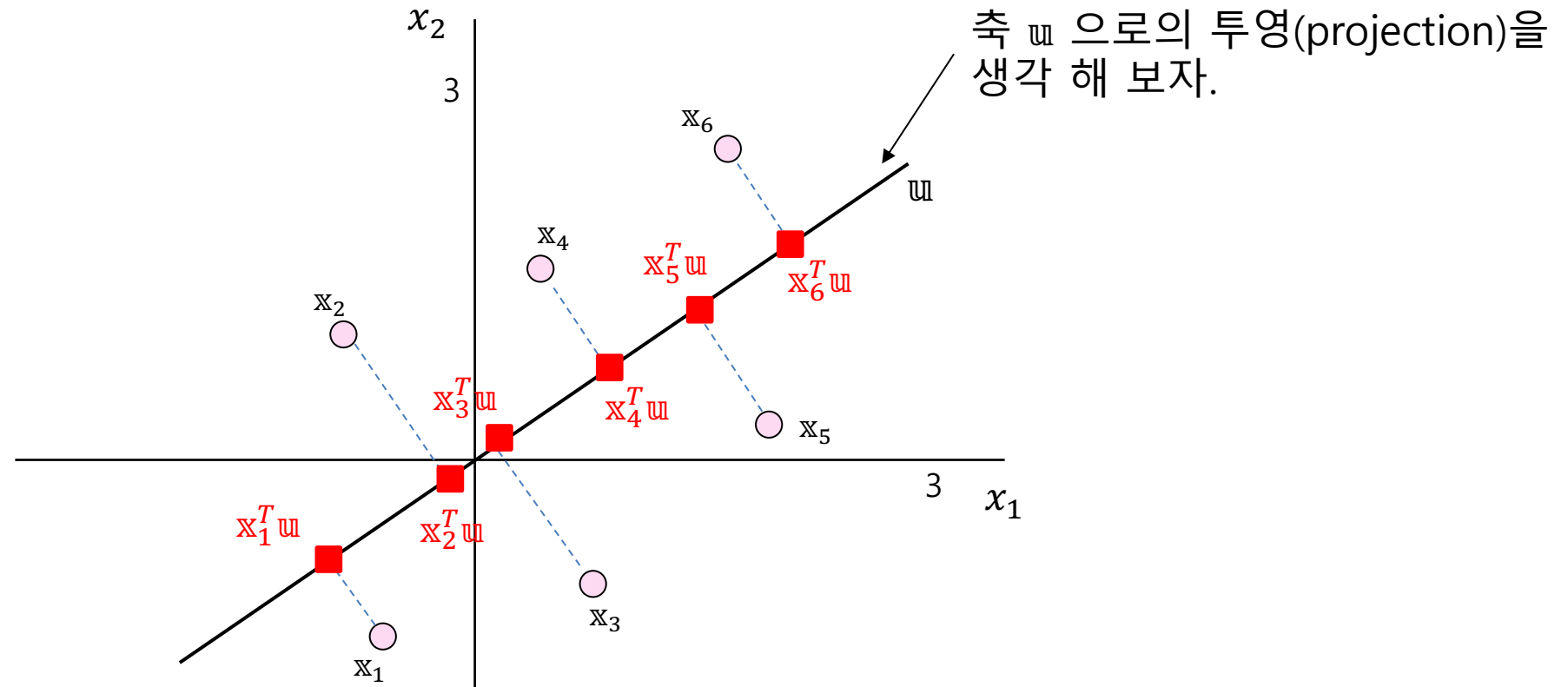
Standardizing



attribute  $x_1$ 과  $x_2$ 가  
같은 scale을 가진다.

# 투영된 공간에서 분산 측정하기

정규화 된 데이터 집합에서, 투영 된 공간(원래의 공간에 존재하는 축으로의 투영)에서의 분산을 구해 보자.



※  $x_1$ 과  $x_2$ 는 정규화 됨.

$x_1^T u, x_2^T u, x_3^T u, x_4^T u, x_5^T u, x_6^T u$ 의 분산을 구하자 !

# 투영된 공간에서 분산 측정하기

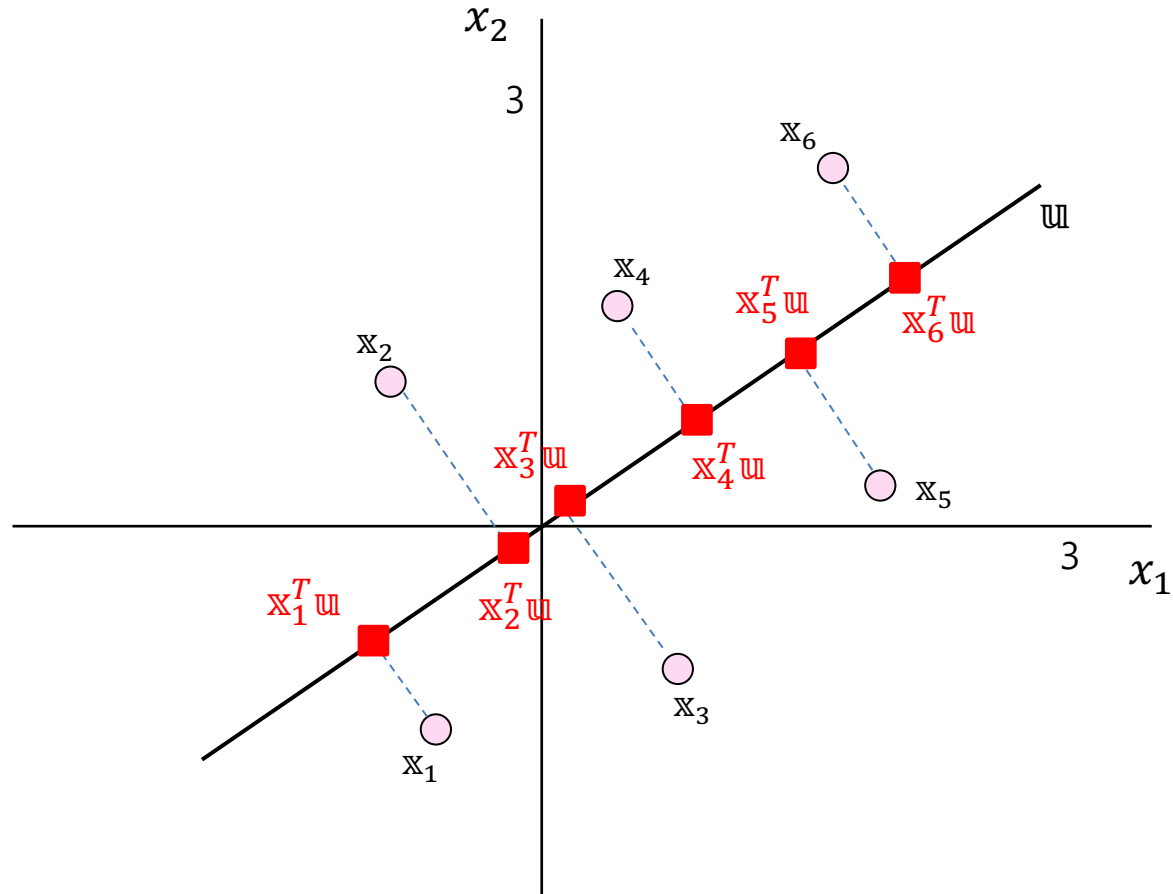
$\mathbf{x}_1^T \mathbf{u}, \mathbf{x}_2^T \mathbf{u}, \mathbf{x}_3^T \mathbf{u}, \dots, \mathbf{x}_m^T \mathbf{u}$  의 분산을 구하자 !

$$\mu = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^T \mathbf{u} = 0 \quad \mathbf{0} = (0, \dots, 0)^T$$

$$\begin{aligned} \sigma^2 &= \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i^T \mathbf{u} - \mu)^2 = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i^T \mathbf{u})^2 = \frac{1}{m} \sum_{i=1}^m \mathbf{u}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{u} = \mathbf{u}^T \left( \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{u} \\ &= \mathbf{u}^T \left( \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i - \mathbf{0})(\mathbf{x}_i - \mathbf{0})^T \right) \mathbf{u} = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u} \\ &\quad \text{= } \mathbf{\Sigma} \text{ (공 분산 행렬)} \end{aligned}$$

$$\mathbf{x} = (x_1, \dots, x_n)^T$$

# 투영된 공간에서 분산 측정하기

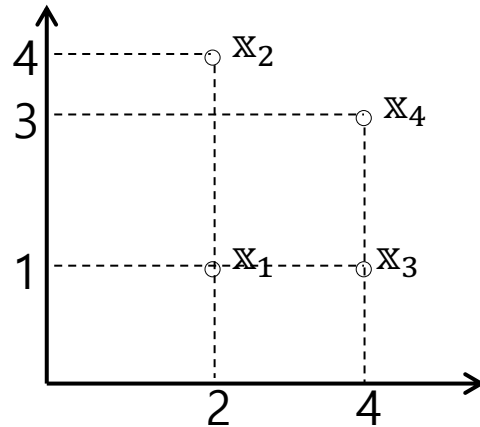


※  $x_1$ 과  $x_2$ 는 정규화 됨.

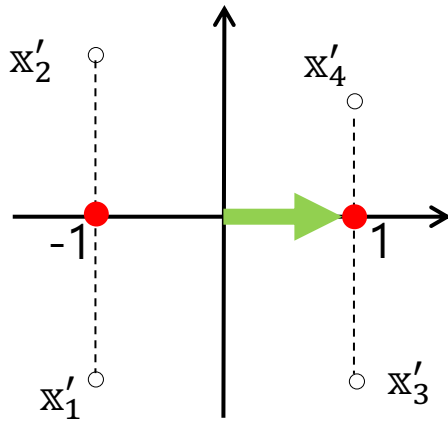
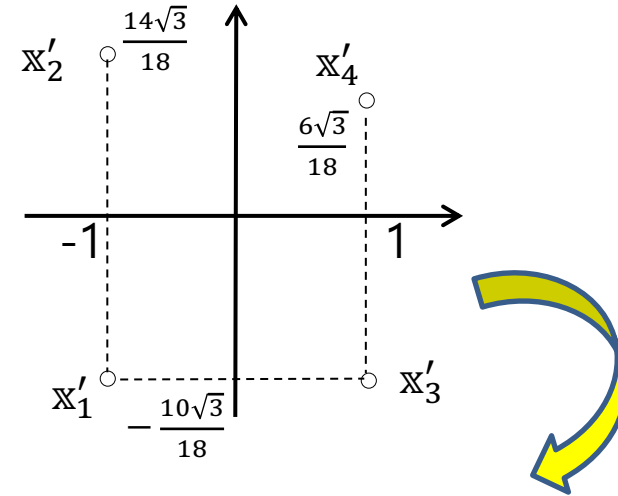
$x_1^T u, x_2^T u, x_3^T u, x_4^T u, x_5^T u, x_6^T u$  의 분산 !  $u^T \Sigma u$

# 투영된 공간에서 분산 측정하기

원래의 공간  
dimensionality : 2

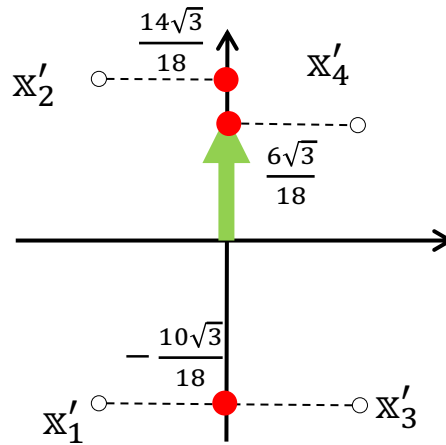


Normalization



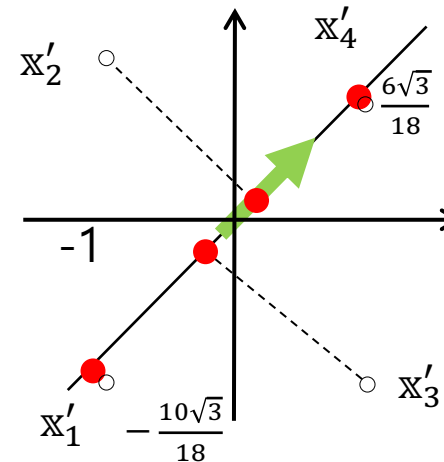
$\mathbf{w}^T = (1, 0)$ 으로 투영

분산 : 1



$\mathbf{w}^T = (0, 1)$ 으로 투영

분산 : 1



$\mathbf{w}^T = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ 으로 투영

분산 : 3.23018

# 분산의 최대치는 어떻게 찾는가?

변환된(투영된) 공간에서 분산을 최대화 해 줄 수 있는 벡터  $\mathbf{w}$ 를 찾자.  $\mathbf{w}$ 는 unit vector라고 생각하자.

우리가 구하고자 하는  $\mathbf{w}$ 는 방향이 중요하기 때문이다. 즉,  $\mathbf{w}^T \mathbf{w} = 1$ 이다.

따라서 문제는  $\mathbf{w}$ 가 unit vector일 때의  $\mathbf{w}^T \Sigma \mathbf{w}$ 의 최대값을 구하는

조건부 최적화 문제가 된다.

$$\text{Maximize } \mathbf{w}^T \Sigma \mathbf{w} \quad \text{s.t } \mathbf{w}^T \mathbf{w} = 1$$

이것은 라그랑지 승수(Lagrange multiplicity)를 사용하여 해결 할 수 있다.

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \Sigma \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1)$$

# 분산의 최대치는 어떻게 찾는가?

$\mathcal{L}(\mathbf{w}, \lambda)$ 를 미분하여  $\mathbf{w}$ 의 최대치를 구해 보자.

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \Sigma \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1)$$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = \Sigma \mathbf{w} - \lambda \mathbf{w} = 0$$

$$\Leftrightarrow \Sigma \mathbf{w} = \lambda \mathbf{w}$$

$$\Leftrightarrow \mathbf{w} : \text{eigenvector of } \Sigma$$

eigenvector  $\mathbf{w}$ 를 분산 식에 대입하자.

$$\mathbf{w}^T \Sigma \mathbf{w} = \mathbf{w}^T \lambda \mathbf{w} = \lambda \mathbf{w}^T \mathbf{w} = \lambda$$

즉, 분산을 최대화 하는 문제는  $\Sigma$ 의 eigenvalue를 최대화 하는 문제가 된다.

$$\arg \max_{\mathbf{w}} \mathbf{w}^T \Sigma \mathbf{w} = \arg \max_{\lambda} \lambda$$

# 분산의 최대치는 어떻게 찾는가?

$$\arg \max_{\mathbf{u}} \mathbf{u}^T \Sigma \mathbf{u} = \arg \max_{\lambda} \lambda$$

따라서, 변환된(축소된) 공간에서 분산의 최대값은  $\Sigma$ 의 eigenvalue의 최대값이다.

분산의 최대값은,  $\mathbf{u}$ 가  $\Sigma$ 의 eigenvalue 중 가장 큰 값을 가지는 eigenvalue에

대응되는 eigenvector일 때 달성된다. 우리는 이것을 주성분이라고도 부른다.

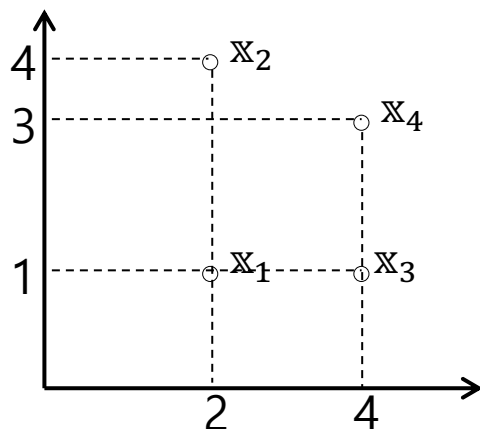
principal eigenvector

principal component

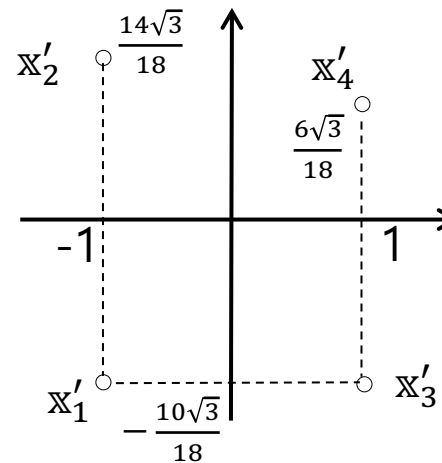


# 분산의 최대치는 어떻게 찾는가?

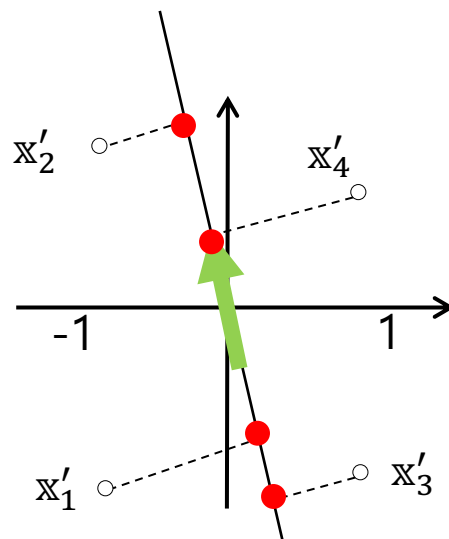
원래의 공간  
dimensionality : 2



Normalization



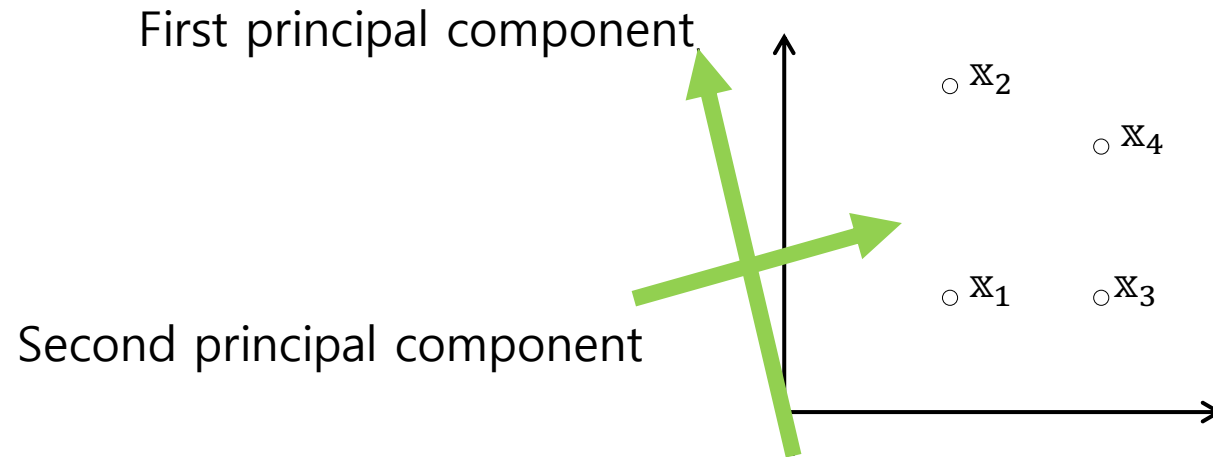
1-dimensional  
공간으로  
투영



$\mathbf{u}$  : 공 분산 행렬의 principal eigenvector(principal component)

분산 > 3.23018

# 분산의 최대치는 어떻게 찾는가?



Second principal component는 어떻게 구하는가? 다음 슬라이드에서....

# 데이터가 분산되어 있는 주요한 방향 찾기(D차원)

지금까지 우리는 2차원에서의 주성분(principal components)를 구했다.

그러면 이제 D차원에서의 주성분들을 구할 것이다.

2차원에서 데이터의 첫번째 주성분(principal component)은, 데이터의 공 분산 행렬의 가장 큰 eigenvalue에 대응되는 eigenvector였다.

**D차원에서 주성분은 데이터 공분산 행렬의 가장 큰 eigenvalue에서 부터 D번째로 큰 eigenvalue까지에 대응되는 D개의 eigenvector가 될 것이다.**

# 데이터가 분산되어 있는 주요한 방향 찾기(D차원)

(증명!)

D개의 주성분 중 가장 첫 번째 주성분( $\mathbb{p}_1$ )은 데이터의 공 분산 행렬의 가장 큰 eigenvalue에 대응하는 eigenvector  $\mathbb{w}_1$  이었다.

두 번째 주성분( $\mathbb{p}_2$ ) 되기 위해서는, 우선 데이터  $\mathbb{x}$ 의  $\mathbb{w}_1$ 로의 투영과, 두 번째 주성분이 될 벡터로의 투영의 상관성이 없어야 한다(uncorrelated). 즉,  $cov(\mathbb{w}_1^T \mathbb{x}, \mathbb{p}_2^T \mathbb{x}) = 0$  이어야 한다.

또한,  $\mathbb{p}_2$ 는 d차원으로 투영 된 데이터의 분산인  $\mathbb{w}^T \Sigma \mathbb{w}$ 를 최대화 시켜야 한다.

$$\ast cov(\mathbb{w}_1^T \mathbb{x}, \mathbb{w}^T \mathbb{x}) = \mathbb{w}_1^T \Sigma \mathbb{w} = \mathbb{w}^T \Sigma \mathbb{w}_1 = \mathbb{w}^T \lambda_1 \mathbb{w}_1 = \lambda_1 \mathbb{w}^T \mathbb{w}_1$$

$$cov(\mathbb{w}_1^T \mathbb{x}, \mathbb{w}^T \mathbb{x}) = 0 \quad \Leftrightarrow \quad \lambda_1 \mathbb{w}^T \mathbb{w}_1 = 0 \quad \Leftrightarrow \quad \mathbb{w}^T \mathbb{w}_1 = 0$$

# 데이터가 분산되어 있는 주요한 방향 찾기(D차원)

따라서  $\mathbb{p}_2$ 를 구하기 위한 조건부 최대화 식은 다음과 같다.

$$\max_{\mathbb{w}} \mathbb{w}^T \Sigma \mathbb{w} \quad s.t. \quad \mathbb{w}^T \mathbb{w} = 1 \quad \& \quad cov(\mathbb{w}_1^T \mathbb{X}, \mathbb{w}^T \mathbb{X}) = 0$$
$$\mathbb{w}^T \mathbb{w}_1 = 0$$

라그랑지 승수는 다음과 같다.

$$\mathcal{L}(\mathbb{w}, \lambda, \delta) = \mathbb{w}^T \Sigma \mathbb{w} - \lambda(\mathbb{w}^T \mathbb{w} - 1) - \delta(\mathbb{w}^T \mathbb{w}_1)$$

# 데이터가 분산되어 있는 주요한 방향 찾기(D차원)

$\mathbf{w}$ 에 대하여 미분을 하자.

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda, \delta) = 2\Sigma\mathbf{w} - 2\lambda\mathbf{w} - \delta\mathbf{w}_1 = 0$$

양변에  $\mathbf{w}_1^T$ 를 곱하자.

$$2\mathbf{w}_1^T \Sigma \mathbf{w} - 2\lambda \mathbf{w}_1^T \mathbf{w} - \delta \mathbf{w}_1^T \mathbf{w}_1 = 0$$

위 식은,  $0 - 0 - \delta 1 = 0$  과 동치이다. 즉  $\delta = 0$ 이 되어야 한다.

따라서

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda, \delta) = 2\Sigma\mathbf{w} - 2\lambda\mathbf{w} - \delta\mathbf{w}_1 = 2\Sigma\mathbf{w} - 2\lambda\mathbf{w} = 0$$

이 되어야 한다.

# 데이터가 분산되어 있는 주요한 방향 찾기(D차원)

$\Sigma \mathbf{w} - \lambda \mathbf{w} = 0$  이기 때문에,  $\mathbf{w}^T \Sigma \mathbf{w} = \lambda$ 이다.

이미 첫 번째 주성분은 찾은 상태이고, 첫 번째 주성분과 uncorrelated 관계에 있는 주성분을 찾기 위한 것이므로  $\lambda$ 는  $\lambda_1$ 이 아니다. 가장 큰 eigenvalue 값을 제외하고, 가장 큰 값을 갖는 eigenvalue는  $\Sigma$ 의 두 번째로 큰 eigenvalue가 될 것이다.

따라서, 두 번째로 큰 eigenvalue에 대응되는 eigenvector가 우리가 찾는 두 번째 주 성분이 될 것이다.

이와 같이, 3 번째, 4 번째, ..... , D 번째 주성분을 찾는다.

즉, D개의 주성분은,  $\Sigma$ 의 eigenvalue 를 크기 순서로 배열하여 가장 큰 것에서부터 D 번째로 큰 eigenvalue  $e_1, e_2, \dots, e_D$ 에 각각 대응하는  $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_D$  이다.

# PCA 작동 예제

다음과 같이 데이터가 주어졌을 때, PCA의 작동을 살펴보자.

	$x$	$y$
1	0.72	0.14
2	0.18	0.23
3	2.50	2.30
4	0.45	0.17
5	0.03	0.44
6	0.13	0.24
7	0.30	0.03
8	2.65	2.10
9	0.91	0.92
10	0.46	0.33
평균	0.83	0.69



# PCA와 SVD의 연결

PCA를 통하여 얻어지는 주성분들은, data들의(normalized points) 공분산 행렬의 값이 큰 eigenvalue들에 대응되는 eigenvector들이다.

주성분을 구할 수 있는 다른 방법은 없을까?

Singular Value Decomposition! SVD!

What is SVD?

- SVD는 행렬을 분해하는 기법중에 하나이다.
- SVD는 모든(any) 행렬에 대하여 적용가능한 분해(정수와 비교하면 인수분해) 기법이기 때문에, 행렬의 분해에 가장 널리 사용되는 기법이다.

# Singular Value Decomposition

- $X : n \times p$  matrix
- $U : n \times n$  matrix whose columns are **orthonormal** ( $U^T U = I$ )
- $\Sigma : n \times p$  matrix containing the  $r = \min(n, p)$ , rank,  $\sigma_i \geq 0$  on the **main diagonal**
- $V : p \times p$  matrix whose rows and columns are **orthonormal** ( $V^T V = V V^T = I$ )

모든 (any) matrix  $X$ 에 대하여,

$$X = U \Sigma V^T$$

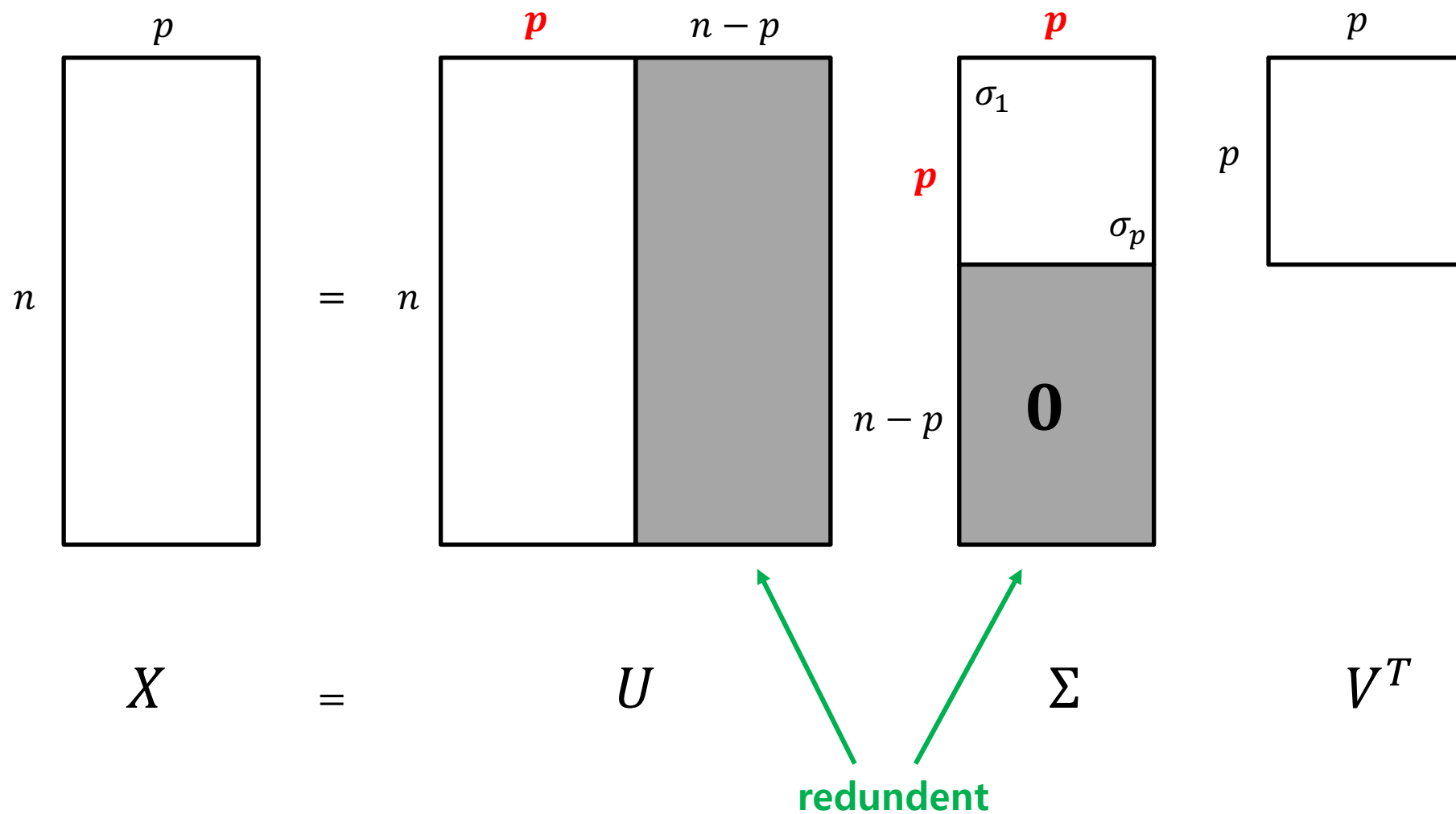
와 같이 분해될 수 있다.  $\sigma_i$ 를 singular value(특이값)이라고 한다.

$X^T X, X X^T$ 의 고유값을  $\lambda_i$  ( $i = 1, \dots, r$ )이라고 할 때,

$$\sigma_i = \sqrt{\lambda_i}$$

for  $i = 1, \dots, r$  이다.

# Singular Value Decomposition



# Singular Value Decomposition

앞의 그림을 참고해보면,  $U$ 의 마지막  $(n - p)$  column은  $\Sigma$ 의 0인 부분과 곱해지므로, 행렬 A에서 미치는 영향력이 없다. (irrelevant , redundant)

따라서 아래와 같은 행렬의 곱으로 바꾸어 생각할 수 있다.

The diagram illustrates the decomposition of matrix  $X$  into three components. On the left, matrix  $X$  is represented by a rectangle with height  $n$  and width  $p$ . This is followed by an equals sign. To the right of the equals sign is matrix  $U_{suf}$ , a rectangle with height  $n$  and width  $p$ . This is followed by matrix  $\Sigma_{suf}$ , a square with side length  $p$  containing diagonal elements  $\sigma_1$  and  $\sigma_p$ . Finally, there is matrix  $V_{suf}^T$ , a square with side length  $p$ . The labels  $U_{suf}$ ,  $\Sigma_{suf}$ , and  $V_{suf}^T$  are written in blue, while the dimensions  $n$  and  $p$  are in black.

$$\begin{matrix} p \\ \boxed{\phantom{X}} \\ n \\ X \end{matrix} = \begin{matrix} p \\ \boxed{\phantom{U_{suf}}} \\ n \\ U_{suf} \end{matrix} \begin{matrix} p \\ \boxed{\begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{matrix}} \\ p \\ \Sigma_{suf} \end{matrix} \begin{matrix} p \\ \boxed{\phantom{V_{suf}^T}} \\ p \\ V_{suf}^T \end{matrix}$$

즉, SVD를 사용하여 데이터에서(matrix) 불필요한 정보나 noise를 제거할 수 있다.

# PCA - SVD

## PCA와 SVD의 연관성

- $X_{n \times p}$  : 데이터 샘플  $\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^p) \in \mathbb{R}^p$  for  $i = 1, \dots, n$ 를 row로 하는 matrix
- SVD를 통하여 얻어진,  $X = U\Sigma V^T$ 에서,  $V$ 의 column들이  $X$ 의 공분산 행렬  $X^T X$ 의 eigenvector이다. 즉,  $X$ 의 주성분들이다.
- eigenvectors of  $XX^T$  :  $U$
- eigenvalues of  $XX^T$  :  $\Sigma$ 의 대각성분의 제곱값
- eigenvectors of  $X^T X$  :  $V$
- eigenvalues of  $X^T X$  :  $\Sigma$ 의 대각성분의 제곱값

# PCA - SVD

$X^T X$  (임의의 행렬  $X$ )

- $X^T X = V \Sigma^T U^T U \Sigma V^T = V (\Sigma^T \Sigma) V^T = V D V^T$ 
  - $D = \Sigma^T \Sigma$  이기 때문에  $D$ 의 대각성분은 singular value의 제곱
- $(X^T X)V = V D V^T V = V D$
- 위의 식에 의하여  $X^T X$ 의 고유값은  $V$ 의 대각성분들이 되고,  $V$ 의 columns은  $X^T X$ 의 고유벡터가 된다.

$XX^T$  (임의의 행렬  $X$ )

- $XX^T = U \Sigma V^T V \Sigma^T U^T = U \Sigma^T \Sigma U^T = U D U^T$ 
  - $D = \Sigma^T \Sigma$  이기 때문에  $D$ 의 대각성분은 singular value의 제곱
- $(XX^T)U = U D U^T U = U D$
- 위의 식에 의하여  $XX^T$ 의 고유값은  $U$ 의 대각성분들이 되고,  $U$ 의 columns은  $XX^T$ 의 고유벡터가 된다.

따라서,  $X$ 의 공분산 행렬인  $X^T X$ 의 eigenvector를 쉽게 도출해 낼 수 있다.

# PCA가 사용되는 예

## 특징 추출

- 얼굴 인식이 대표적인 예이다.
- 원래 훈련 집합에 있는 샘플 신호(얼굴 영상)  $\mathbf{s}$  에서 특징 벡터  $\mathbf{x}$ 를 추출한다.
  - 주성분으로 이루어진 변환행렬  $U \Rightarrow \mathbf{x} = U\mathbf{s}$
- 추출된 특징 벡터  $\mathbf{x}$ 의 집합들을 이용하여 새로운 훈련 집합을 구성하고 다음 단계의 지도학습에 사용한다.

## 특징 벡터의 차원 축소

- 특징 벡터  $\mathbf{x}$ 가 있을 때 이것의 차원을 줄여서 새로운 특징 벡터  $\mathbf{x}^{new}$ 를 생성할 수 있다.
- 수백 차원의 데이터 구조는 이해하기 힘들므로 2차원이나 3차원으로 관측값의 차원을 축소해 시각적으로 이해하기 쉽도록 도와준다.

# PCA가 사용되는 예

차원의 저주를 완화

정보 손실을 최소화하면서 데이터를 압축



# PCA가 사용되는 예

얼굴 인식 분야에서 널리 쓰이는 특징 추출 방법(Eigenface)

- 얼굴 영상 집합 :  $Y = \{I_1, I_2, \dots, I_N\}$ ,  $I_i$ 는  $n \times n$  크기의 영상
- $I_i$ 를 1차원 신호로 변환
  - $s_i = (I_i[1][1], I_i[1][2], \dots, I_i[1][n], \dots, I_i[2][1], \dots, I_i[2][n], \dots, I_i[n][n])^T$
  - $Y = \{I_1, \dots, I_N\}$ 은  $S = \{s_1, \dots, s_n\}$ 으로 변환.
- $S$ 를 훈련 집합으로 하고, 주성분 분석(PCA)를 사용하여, 주성분으로 이루어진 변환 행렬을 생성
- 변환 행렬을 이용하여 훈련 집합  $S$ 에서, 새로운 특징 벡터로 이루어진 훈련 집합  $S^{new}$ 를 생성
- $S^{new}$ 로 분류기를 훈련
  - SVM, K-NN, 신경망(perceptron) 등을 이용
- 인식을 해야하는 새로운 얼굴 영상이 입력되면 2번째 과정을 이용하여 신호  $s_{new}$ 를 구하고, 변환 행렬을 이용하여 특징벡터  $x_{new}$ 를 추출. 분류기에 넣어서 인식.