Dimensionality Reduction

차원축소

Fereshteh Sadeghi

CSEP 546

Motivation

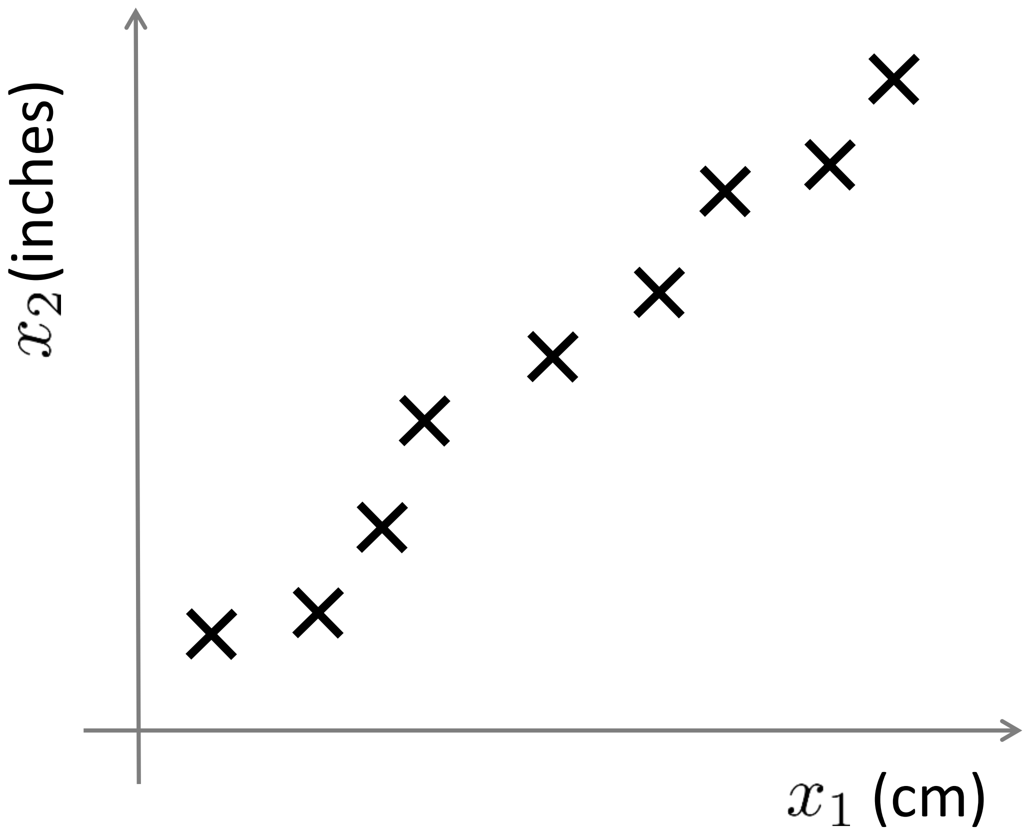
- Clustering
  - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
  - Another way to simplify complex high-dimensional data
  - Summarize data with a lower dimensional real valued vector

Motivation

- Clustering
  - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
  - Another way to simplify complex high-dimensional data
  - Summarize data with a lower dimentional real valued vector

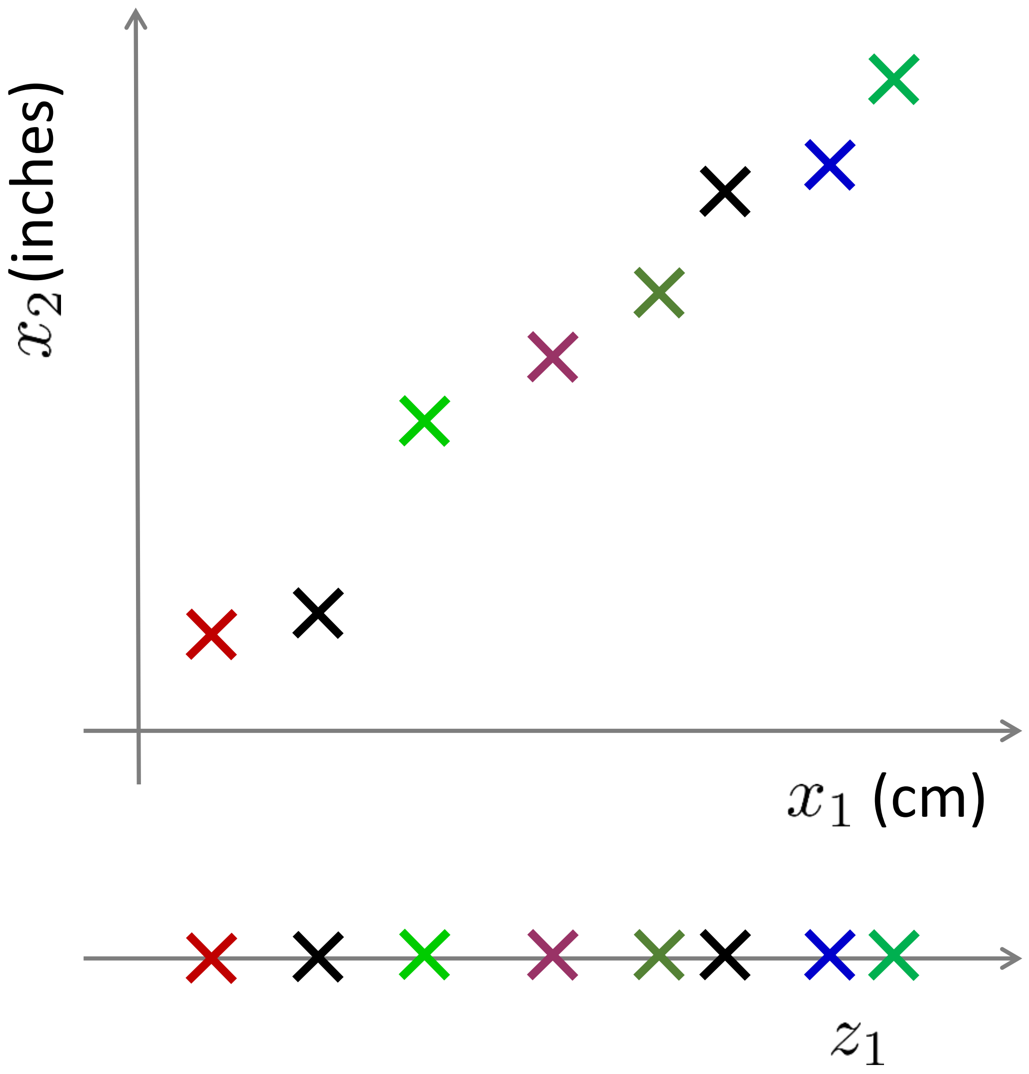
- Given data points in d dimensions
- Convert them to data points in $r<d$ dimensions
  d보다 작은 r차원
- With minimal loss of information

Data Compression



Reduce data from 2D to 1D

Data Compression

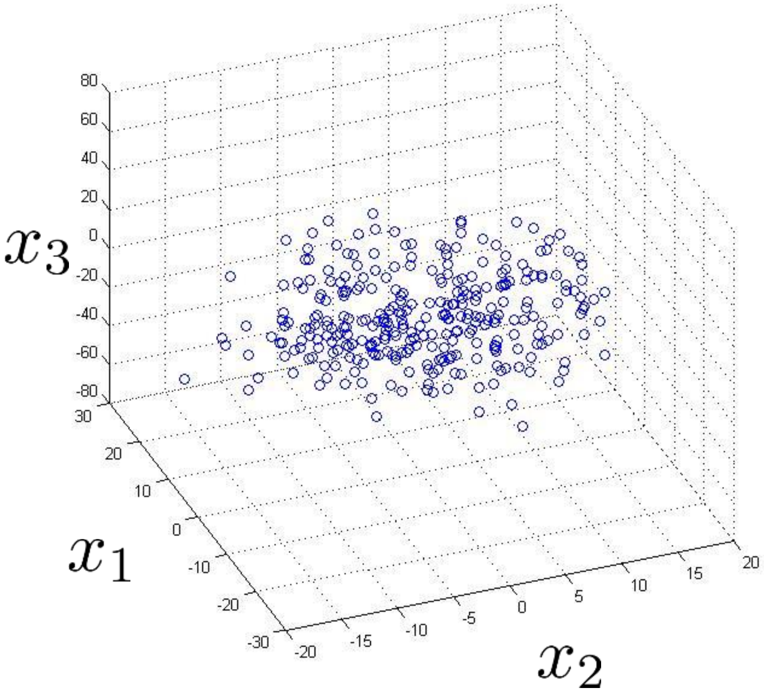


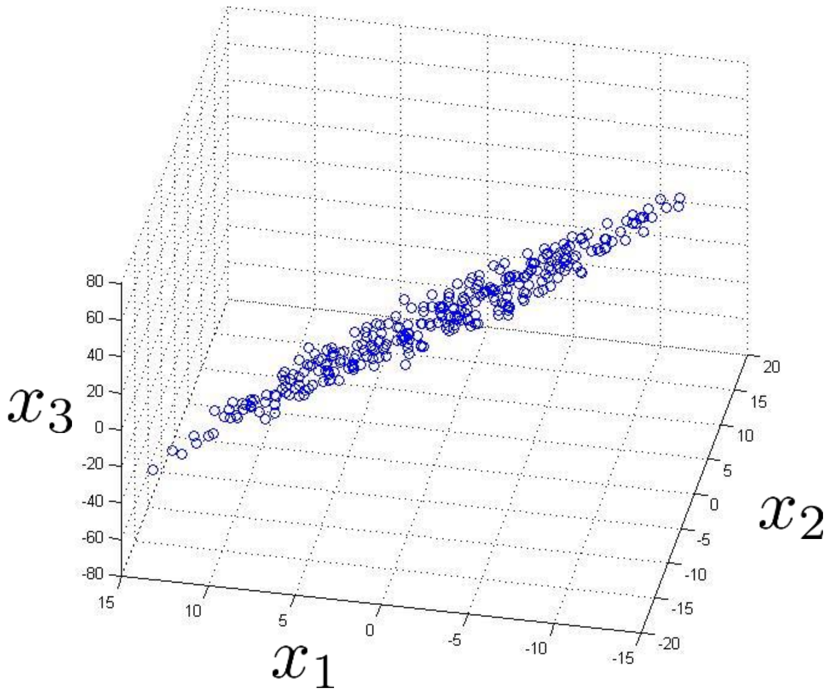
Reduce data from 2D to 1D

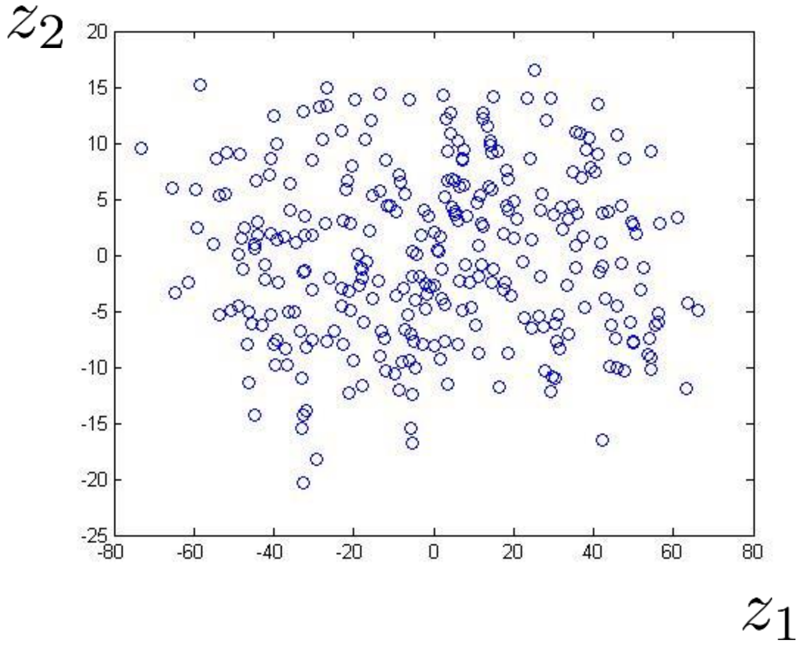
$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

Data Compression

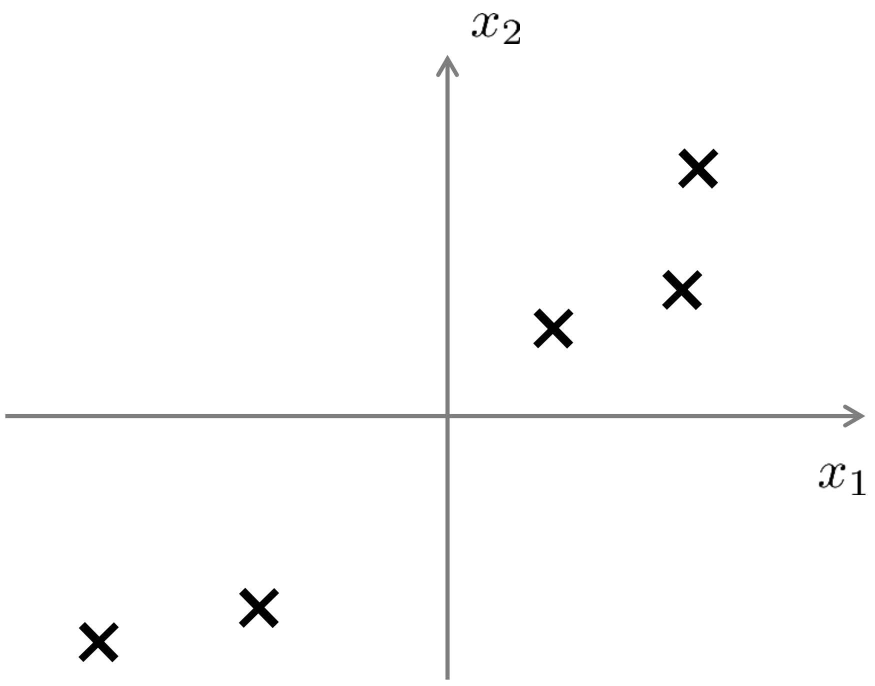
Reduce data from 3D to 2D





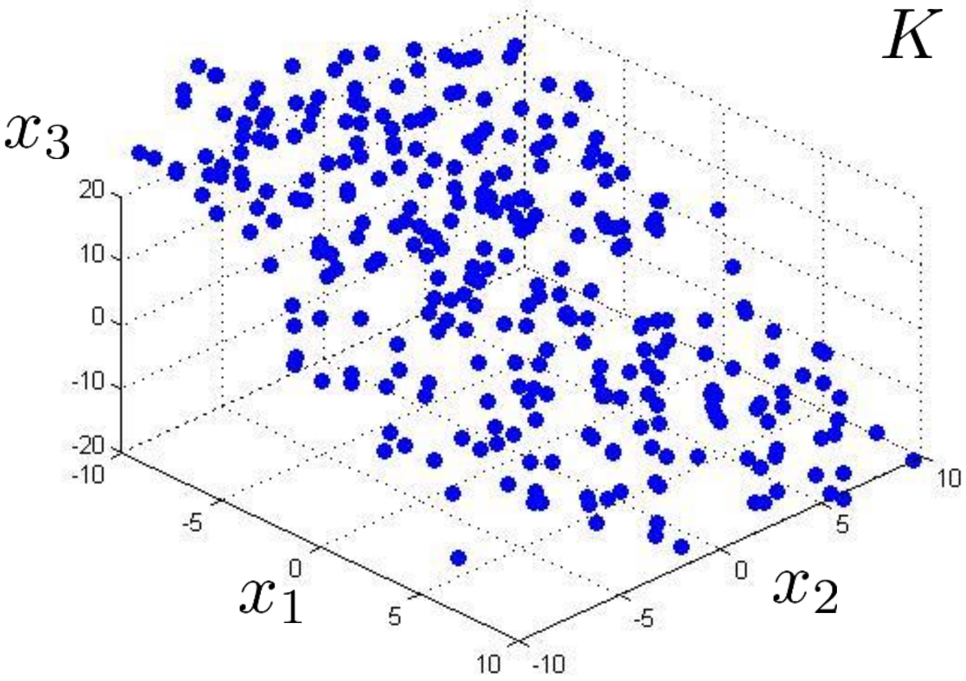


Principal Component Analysis (PCA) problem formulation



$3D \to 2D$

$K = 2$



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

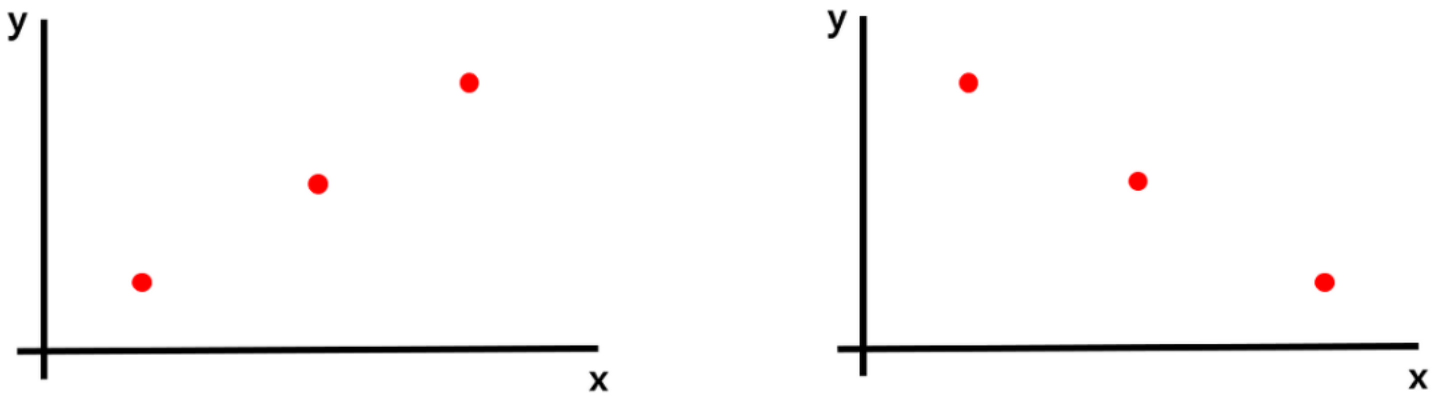
   $$y = A(x - \mu)$$

   where y is the new point, x is the old one, and the rows of A are the eigenvectors

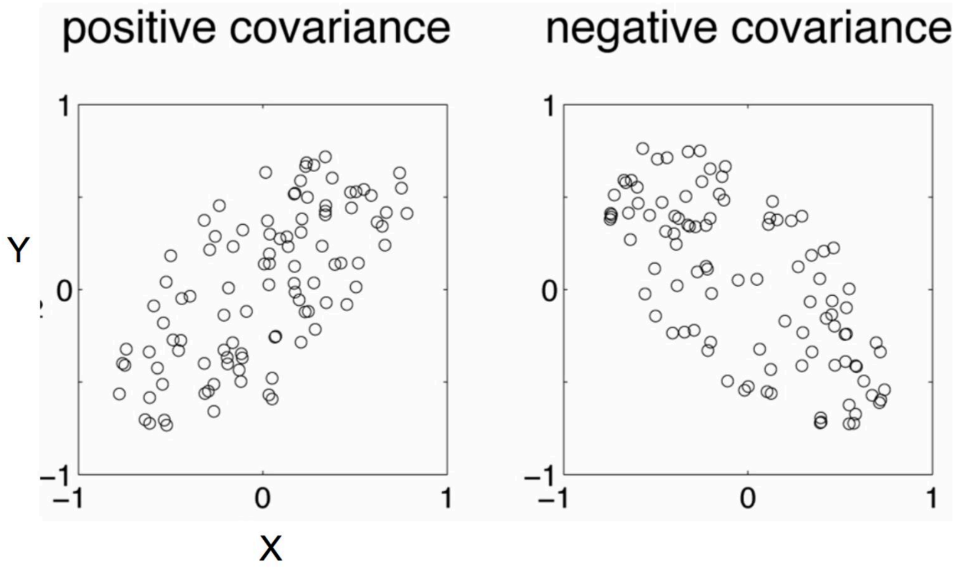
Covariance
공분산

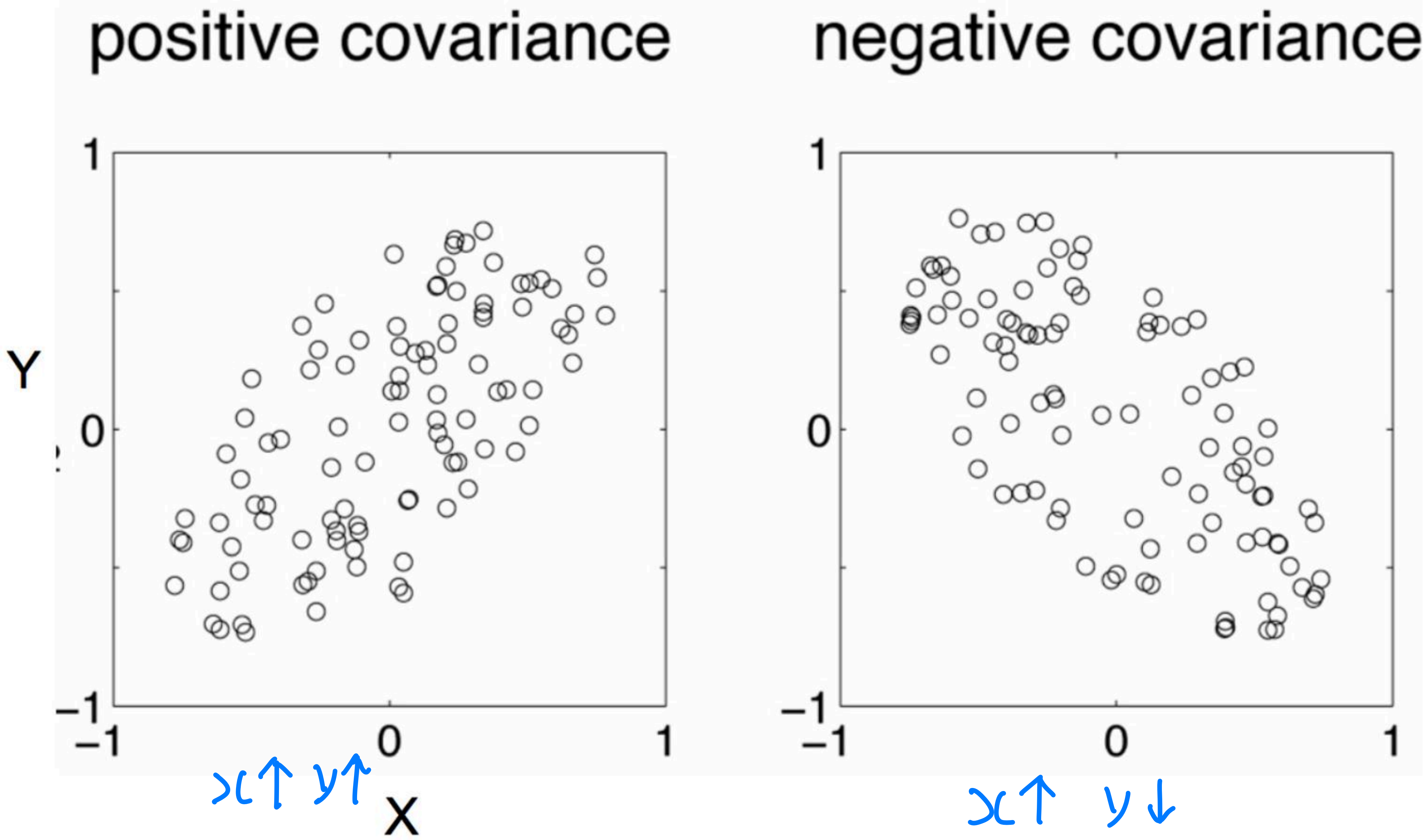
- Variance and Covariance:
  - Measure of the “spread” of a set of points around their center of mass(mean)
- Variance:
  - Measure of the deviation from the mean for points in one dimension
- Covariance:
  - Measure of how much each of the dimensions vary from the mean with **respect to each other**

- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between two dimensions**
- **Covariance between one dimension is the variance** 분산



One dimension (-> 한차원의 variable 이라고 생각) 을 사용한 variance는 데이터의 상관관계를 타낼 수 없음





Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

두 차원간의 관련 정도를 의미하는 행렬 (대칭행렬)

- Used to find relationships between dimensions in high dimensional data sets

$$q_{jk} = \frac{1}{N} \sum_{i=1}^{N} (X_{ij} - \underbrace{E(X_j)}_{\text{The Sample mean}})(X_{ik} - E(X_k))$$

$$\operatorname{corr}(\mathbf{X}) = \begin{bmatrix} 1 & \frac{\mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)]}{\sigma(X_1)\sigma(X_2)} & \cdots & \frac{\mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)]}{\sigma(X_1)\sigma(X_n)} \\ \frac{\mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)]}{\sigma(X_2)\sigma(X_1)} & 1 & \cdots & \frac{\mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)]}{\sigma(X_2)\sigma(X_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)]}{\sigma(X_n)\sigma(X_1)} & \frac{\mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)]}{\sigma(X_n)\sigma(X_2)} & \cdots & 1 \end{bmatrix}.$$

다차원 확장 가능

- covariance
  - $cov(x,y) = E[(x-m_x)(y-m_y)]$
- 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

계산 시

$$C=\begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(x,y) & cov(y,y) \end{pmatrix}$$

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

Eigenvector and Eigenvalue

$$Ax = \lambda x$$

A: Square Matirx

λ: Eigenvector or characteristic vector

X: Eigenvalue or characteristic value

- ***The zero vector can not be an eigenvector***
- ***The value zero can be eigenvalue***

Eigenvector and Eigenvalue

Ax = λx

A: Square Matrix

λ: Eigenvector or characteristic vector

X: Eigenvalue or characteristic value

Example

$$Show\ \ x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \ is\, an\, eigenvector\ for\ \ A = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}$$

$$Solution : Ax = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$But\quad for\quad \lambda = 0,\quad \lambda x = 0 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$Thus, x\, is\, an\, eigenvector\, of\ \ A, and\, \lambda = 0\ is\, an\, eigenvalue.$$

Eigenvector and Eigenvalue

$$Ax = \lambda x \longrightarrow \begin{array}{l} Ax - \lambda x = 0 \\ (A - \lambda I)x = 0 \end{array}$$

If we define a new matrix B: $\longrightarrow$

$$B = A - \lambda I$$
$$Bx = 0$$

If B has an inverse: $\longrightarrow$ $x = B^{-1}0 = 0$ ✗ BUT! an eigenvector cannot be zero!!

x will be an eigenvector of A if and only if B does not have an inverse, or equivalently det(B)=0 :

$$\boxed{\det(A - \lambda I) = 0}$$

Eigenvector and Eigenvalue

Example 1: Find the eigenvalues of

$$A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{vmatrix} = (\lambda - 2)(\lambda + 5) + 12$$

$$= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

two eigenvalues: $-1, -2$

Note: The roots of the characteristic equation can be repeated. That is, $\lambda_1 = \lambda_2 = \ldots = \lambda_k$. If that happens, the eigenvalue is said to be of multiplicity k.

Example 2: Find the eigenvalues of

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ 0 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 2 \end{vmatrix} = (\lambda - 2)^3 = 0$$

$\lambda = 2$ is an eigenvector of multiplicity 3.

고유 벡터 (eigen vector) 구하기

2차 정방행렬 $A = \begin{pmatrix} 4 & 2 \\ 3 & 5 \end{pmatrix}$ 에 대해 Ax는

$$\begin{pmatrix} 4 & 2 \\ 3 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

이며, 성분 별로 풀어 쓰면

$$\begin{aligned} 4x_1 + 2x_2 &= \lambda x_1 \\ 3x_1 + 5x_2 &= \lambda x_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} (4-\lambda)x_1 \quad + 2x_2 &= 0 \\ 3x_1 + (5-\lambda)x_2 &= x_2 \end{aligned}$$

이를 행렬로 표기하면, $Ax = \lambda x$

$$Ax - \lambda x = 0$$

$$Ax - \lambda I x = 0$$ (I는 단위행렬(Identity matrix))

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

[R 분석과 프로그래밍] http://rfriend.tistory.com

고유 벡터 (eigen vector) 구하기

$A = \begin{pmatrix} 4 & 2 \\ 3 & 5 \end{pmatrix}$ 에 대해 Ax-λx=(A-λI)x=0 을 만족하는 필요충분조건으로 Cramer 정리에 의해 이 식의 계수행렬의 행렬식은 0이 됨

$$D(\lambda) = \det(A - \lambda I) = \begin{vmatrix} 4-\lambda & 2 \\ 3 & 5-\lambda \end{vmatrix}$$

$$= (4-\lambda)(5-\lambda) - 2 \times 3$$

$$= (\lambda-4)(\lambda-5) - 2 \times 3$$

$$= \lambda^2 - 9\lambda + 20 - 6$$

$$= (\lambda-7)(\lambda-2) = 0$$

$$\therefore \lambda = 7,\ 2$$

[R 분석과 프로그래밍] http://rfriend.tistory.com

고유 벡터 (eigen vector) 구하기

λ=7 에 대응하는 고유벡터 x는

$$\begin{pmatrix}4-\lambda & 2\\ 3 & 5-\lambda\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}4-7 & 2\\ 3 & 5-7\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

$$=\begin{pmatrix}-3 & 2\\ 3 & -2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

$$=\begin{pmatrix}-3x_1+2x_2\\ 3x_1-2x_2\end{pmatrix}$$

$$=-3\begin{pmatrix}x_1-\frac{2}{3}x_2\\ -x_1+\frac{2}{3}x_2\end{pmatrix}$$

$$=\left[x_1-\frac{2}{3}x_2\right]\begin{pmatrix}-3\\ 3\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

$$\therefore x_1=\frac{2}{3}x_2$$

λ=7 에 대응하는 고유벡터 x는

$$\therefore \text{고유벡터}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}\frac{2}{3}c_1\\ c_1\end{pmatrix}=3c_1\begin{pmatrix}2\\ 3\end{pmatrix}$$

[R 분석과 프로그래밍] http://rfriend.tistory.com

λ=2 에 대응하는 고유벡터 x는

$$\begin{pmatrix}4-\lambda & 2\\ 3 & 5-\lambda\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}4-2 & 2\\ 3 & 5-2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

$$=\begin{pmatrix}2 & 2\\ 3 & 3\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

$$=\begin{pmatrix}2x_1+2x_2\\ 3x_1+3x_2\end{pmatrix}$$

$$=[x_1+x_2]\begin{pmatrix}2\\ 3\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

$$\therefore x_1=-x_2$$

λ=2 에 대응하는 고유벡터 x는

$$\therefore \text{고유벡터}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}-c_2\\ c_2\end{pmatrix}=c_2\begin{pmatrix}-1\\ 1\end{pmatrix}$$

[R 분석과 프로그래밍] http://rfriend.tistory.com

Principal Component Analysis

Input: $\mathbf{x} \in \mathbb{R}^D \colon \ \mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Set of basis vectors: $\mathbf{u}_1, \ldots, \mathbf{u}_K$

Summarize a *D* dimensional vector *X* with *K* dimensional feature vector *h(x)*

$$h(\mathbf{x}) = \begin{bmatrix} \mathbf{u}_1 \cdot \mathbf{x} \\ \mathbf{u}_2 \cdot \mathbf{x} \\ \cdots \\ \mathbf{u}_K \cdot \mathbf{x} \end{bmatrix}$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

Basis vectors are orthonormal

$$\mathbf{u}_i^T \mathbf{u}_j = 0$$
$$||\mathbf{u}_j|| = 1$$

왜? 직교할까?
-> symmetric matrix의 eigen vector는 직교함

New data representation *h(x)*

$$z_j = \mathbf{u}_j \cdot \mathbf{x}$$
$$h(\mathbf{x}) = [z_1, \ldots, z_K]^T$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

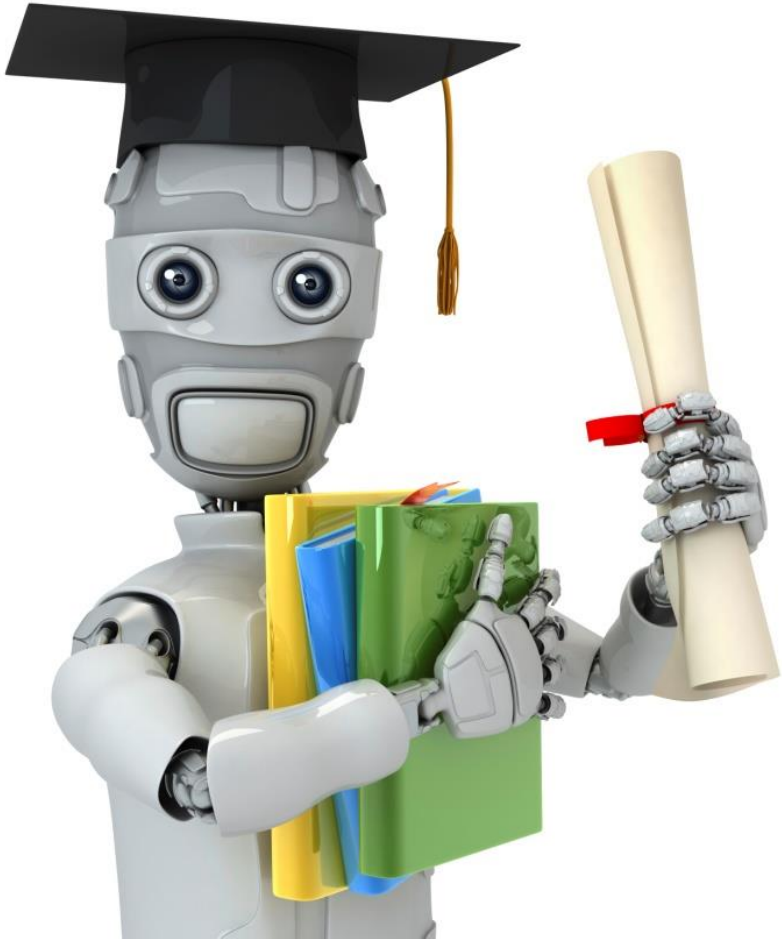
New data representation *h(x)*

$$h(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$$

$$h(\mathbf{x}) = \mathbf{U}^T (\mathbf{x} - \mu_0)$$

Empirical mean of the data

$$\mu_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$



Dimensionality Reduction

Principal Component Analysis algorithm

Machine Learning

Data preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ $\leftarrow$

Preprocessing (feature scaling/mean normalization):

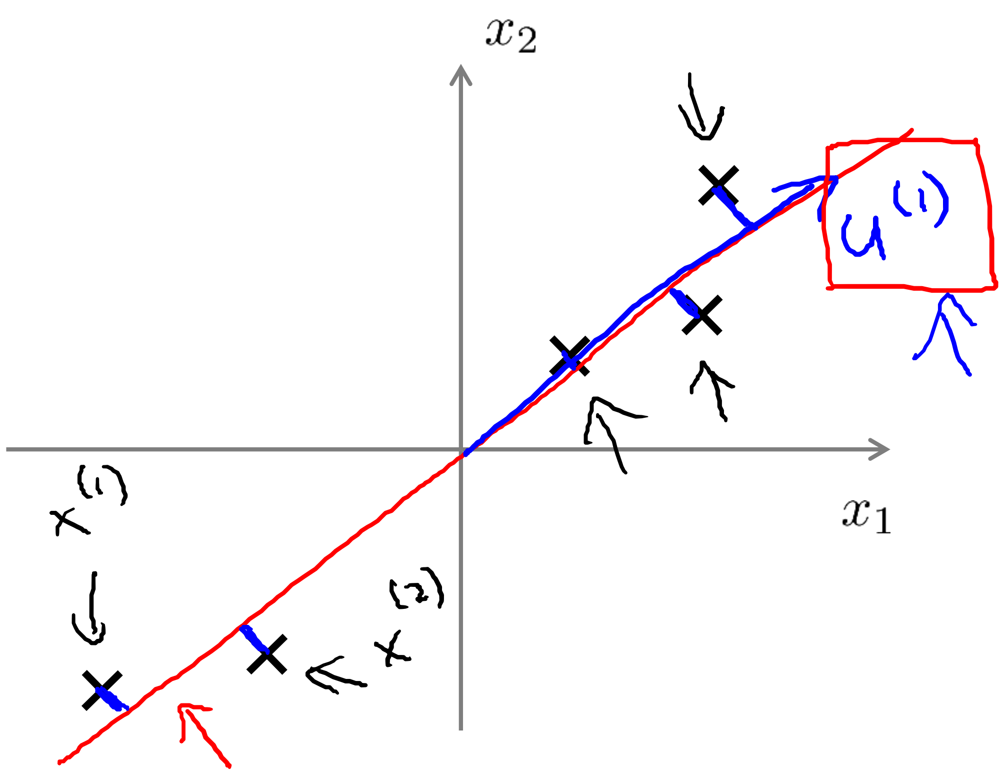
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$

If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

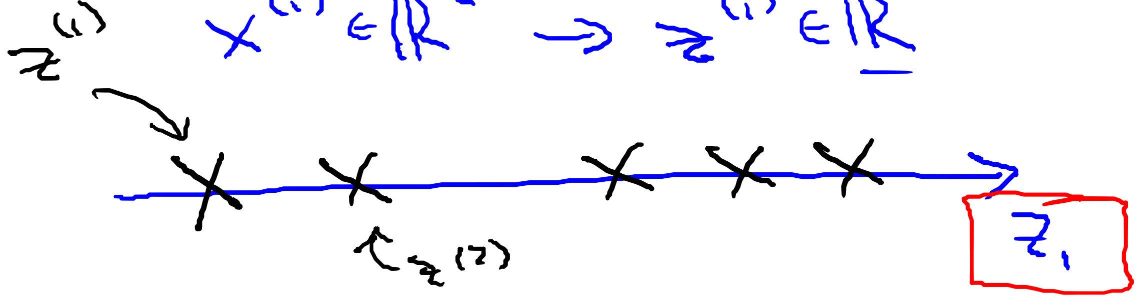
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

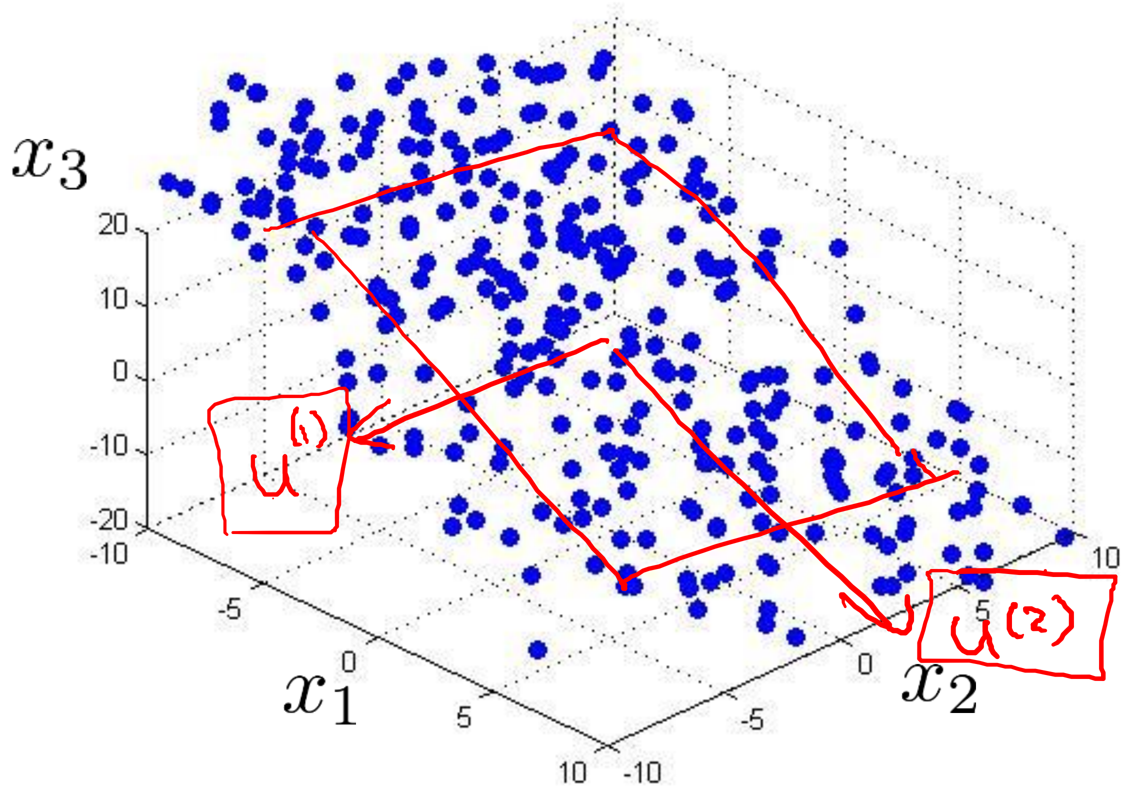
Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D

$x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$





Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n} \boxed{\underbrace{(x^{(i)})}_{n\times 1}\underbrace{(x^{(i)})^T}_{1\times n}} \quad n\times n$$

Sigma

Compute "eigenvectors" of matrix Σ:

→ Singular value decomposition

```
[U,S,V] = svd(Sigma);
```

eig(Sigma)

n×n matrix.

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

k

$U \in \mathbb{R}^{n\times n}$

$u^{(1)}, \ldots, u^{(k)}$

Principal Component Analysis (PCA) algorithm

```
[U,S,V] = svd(Sigma
)
```

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

k

$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}}_{n \times k}^{T} x^{(i)} = \underbrace{\underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix}}_{k \times n} \underbrace{x^{(i)}}_{n \times 1}}_{k \times 1}$$

U_{reduce}

$z \in \mathbb{R}^k$

Principal Component Analysis (PCA) algorithm summary

- After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

```
Sigma =
```

$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$$

$$X = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$$

$\boxed{\text{Sigma} = (1/m) * X' * X;}$

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

$x \in \mathbb{R}^n$ $\cancel{x_0 = 1}$

PCA가 뭔지는 알겠다
근데,
왜 분산이 커지는 방향을 찾을 수 있는거지?

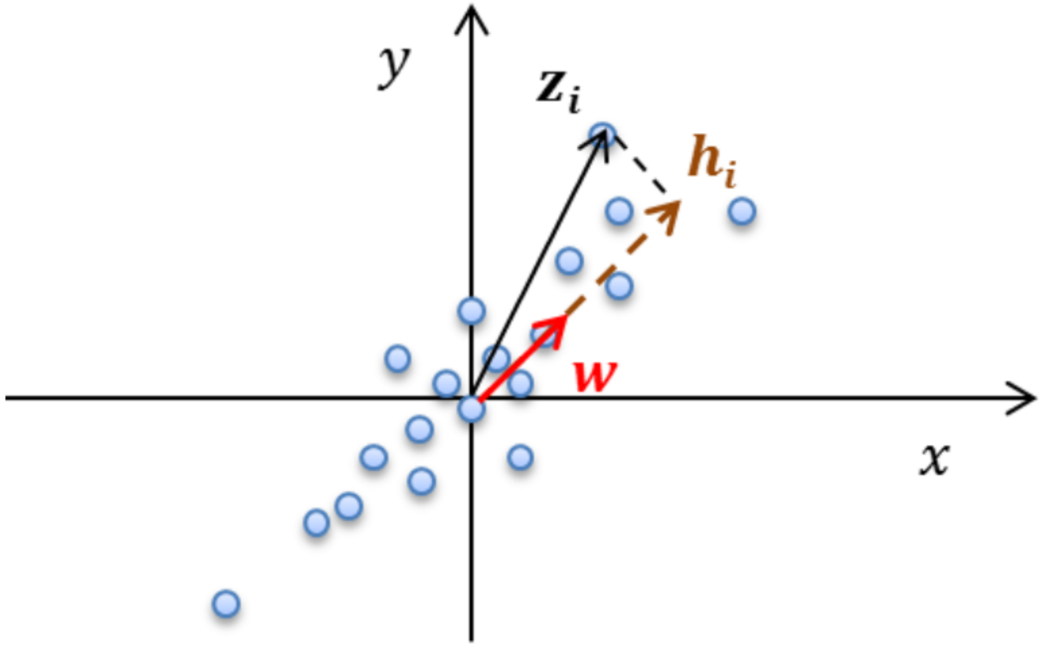
PCA의 목적 ?   -> 데이터 Compression
해결 하고자 하는 원리 -> 데이터가 고루고루 퍼진 분산값이 높은 축을 찾아서 그 방향으로 projection.

방법 ? -> 공분산 행렬 계산 후, eigen vector 랑 eigen value 구해서, value 값이 큰 것에 해당하는 vector로 Projection.

방법은 알겠는데… 왜 공분산 행렬 계산 후 eigen vector 방향이 분산 값이 큰 걸까?

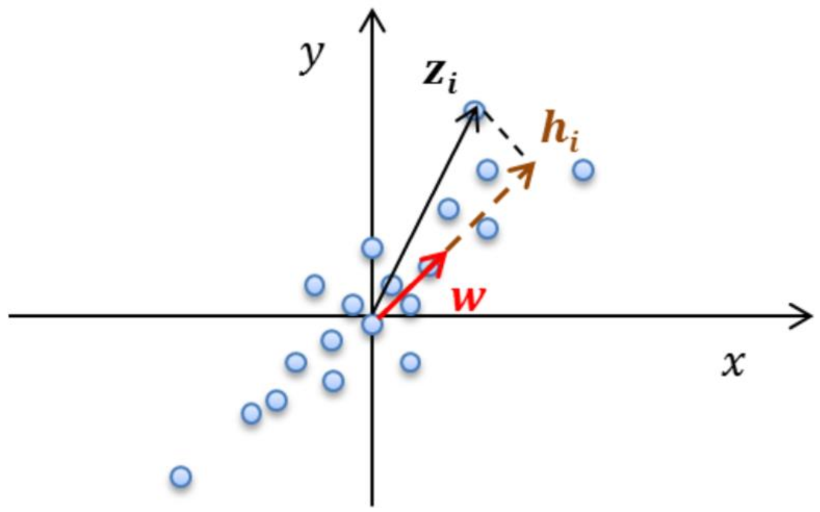
(증명은 시험에 안 나옴)

A. 데이터들을 zi = (x1, ..., xp), i = 1, ..., n라 하자 (데이터의 차원은 p, 개수는 n). 이때, 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.



$(z_i \cdot w)$ 들의 분산을 σ_w^2라 놓고, 원래의 입력 데이터들을 행벡터로 쌓아서 생성한 n x p 행렬을 Z라 하면

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



와 같이 정리된다 (z_i들의 평균이 0이 되도록 centering을 한 후라고 생각하면 ($z_i \cdot w$)의 평균은 0). 이 때, $C = Z^T Z/n$ 으로 잡은 C는 z_i들의 공분산 행렬이 된다.

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

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

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, z_i에 대한 공분산 행렬 C의 eigenvector가 z_i의 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. ◇

The space of all face images

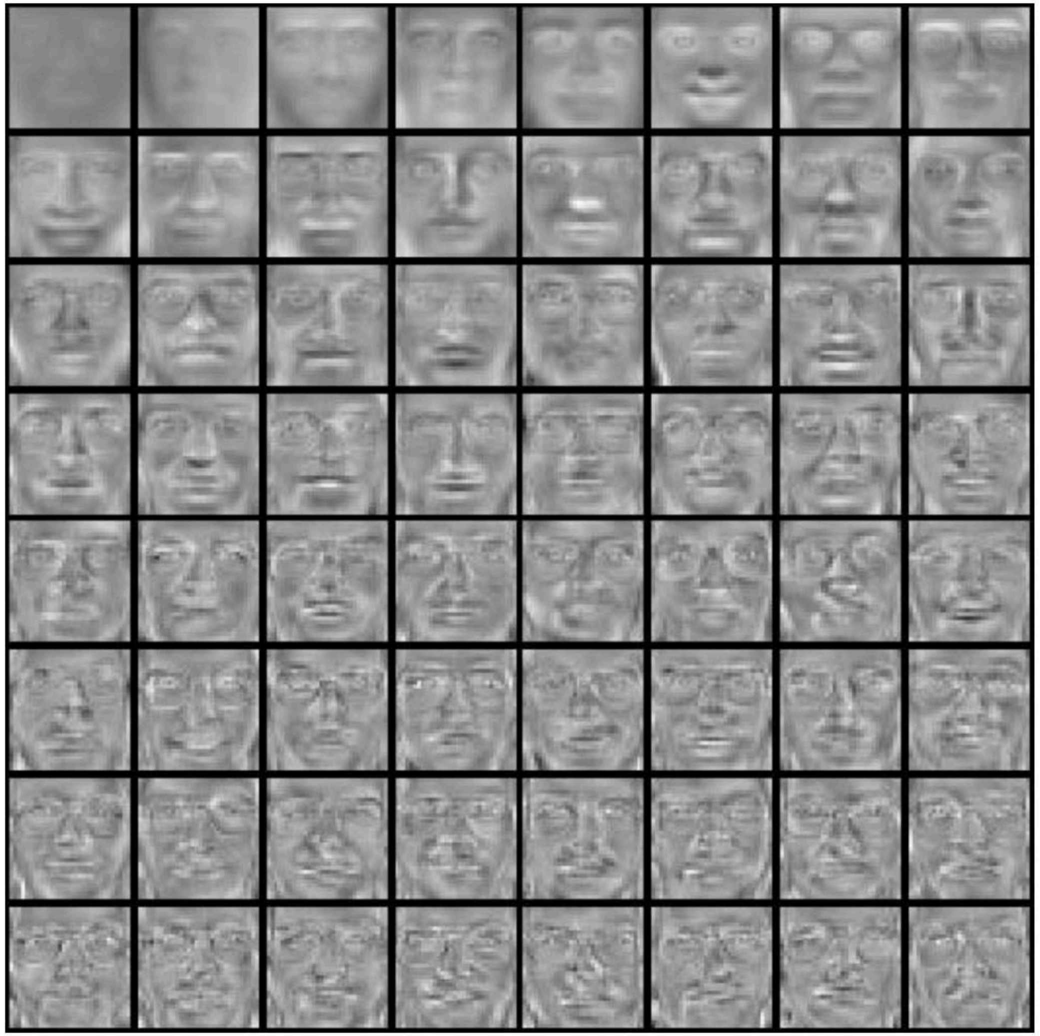
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

slide by Derek Hoiem

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1, \ldots u_k$



slide by Derek Hoiem

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

 =  + 

$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

slide by Derek Hoiem

Reconstruction

P = 4

P = 200

P = 400

After computing eigenfaces using 400 face images from ORL face database

Dimensionality reduction

- PCA (Principal Component Analysis):
  - Find projection that maximize the variance
- ICA (Independent Component Analysis):
  - Very similar to PCA except that it assumes non-Guassian features
- Multidimensional Scaling:
  - Find projection that best preserves inter-point distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the component axes for class-separation
- …
- …