# 손바닥ML

## With Scikit-Learn & Tensorflow

2018. 2. 7

Part I. The Fundamentals of ML

# Ch.8 Dimensionality Reduction

# Vector Spaces

체 $K$에 대한 **벡터 공간** $(V, +, \cdot)$은 $K$에 대한 가군이다. 즉, 다음과 같은 튜플이다.

- $V$는 집합이다. 이 집합의 원소를 **벡터**라고 한다.
- $+ \ : V \times V \to V$는 함수이다. 이 연산을 **벡터 덧셈**이라고 한다.
- $\cdot \ : K \times V \to V$는 함수이다. 이 연산을 **스칼라 곱셈**이라고 한다.

이 데이터는 다음과 같은 공리들을 만족시켜야 한다.

- $(V, +)$는 아벨 군을 이룬다. 즉, 다음 성질들이 성립한다.
  - (벡터 덧셈의 결합 법칙) 임의의 $u, v, w \in V$에 대하여, $(u + v) + w = u + (v + w)$
  - (벡터 덧셈의 교환 법칙) 임의의 $u, v \in V$에 대하여, $u + v = v + u$
  - (벡터 덧셈의 항등원) 임의의 $u \in V$에 대하여 $u + 0 = u$인 원소 $0 \in V$가 존재한다.
  - (역원의 존재) 임의의 $u \in V$에 대하여, $-u + u = 0$인 원소 $-u \in V$가 존재한다.
- $(V, +, \cdot)$는 $K$의 가군을 이룬다. 즉, 다음 성질들이 성립한다.
  - 임의의 $a, b \in K$ 및 $v \in V$에 대하여, $a \cdot (b \cdot v) = (ab) \cdot v$
  - 임의의 $v \in V$에 대하여, $1 \cdot v = v$. 여기서 $1 \in K$는 $K$의 곱셈 항등원이다.
  - (분배 법칙) 임의의 $a, b \in K$ 및 $u, v \in V$에 대하여, $(a + b) \cdot (u + v) = a \cdot u + b \cdot u + a \cdot v + b \cdot v$

# Vector Spaces

- A ***basis*** B of a vector space V over a field F is a linearly independent subset of V that spans V.

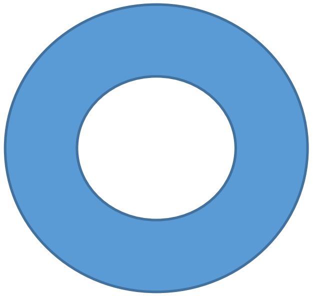
- In more detail, suppose that $B = \{v_1, \cdots, v_n\}$ is a finite subset of a vector space V over a field F (such as the real or complex numbers R or C). Then B is a basis if it satisfies the following conditions:

- the linear independence property,
  for all $a_1, \cdots, a_n \in F$, if $a_1 v_1 + \cdots + a_n v_n = 0$, then necessarily $a_1 = \cdots = a_n = 0$ ; and
- the spanning property,
  for every (vector) $x \in V$ it is possible to choose $a_1, \cdots, a_n \in F$ such that $x = a_1 v_1 + \cdots + a_n v_n$.

- The dimension of a vector space V is the number of elements of a basis B of V.

# The Curse of Dimensionality

- 반지름이 1인 $n$ 차원 구에서 중심을 기준으로 0.5만큼 떨어져 있는 면적(or 부피)와 안쪽의 비율은 차원이 커지면 커질수록 boundary 근처에 대부분이 존재함.



- $n = 2 : \pi - \frac{1}{2}^{2}\pi = \frac{3}{4}\pi$
- $n = 10 : \pi - \frac{1}{2}^{10}\pi = \frac{1023}{1024}\pi$

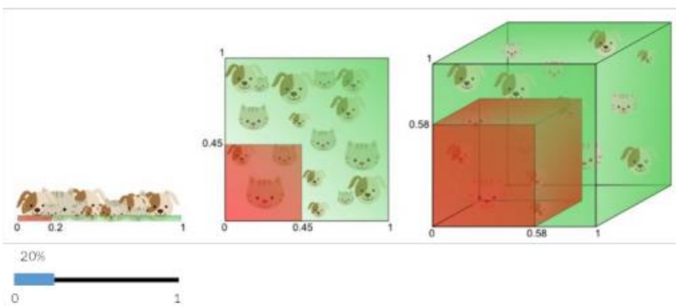
∴ 대다수의 점이 중심으로 부터 멀리 떨어져있다.

- 대부분 ML의 cost function 이 Euclidean distance 를 이용해서 정의되므로 차원이 높아지면서 이와 같은 현상이 생기는건 좋지 않다!

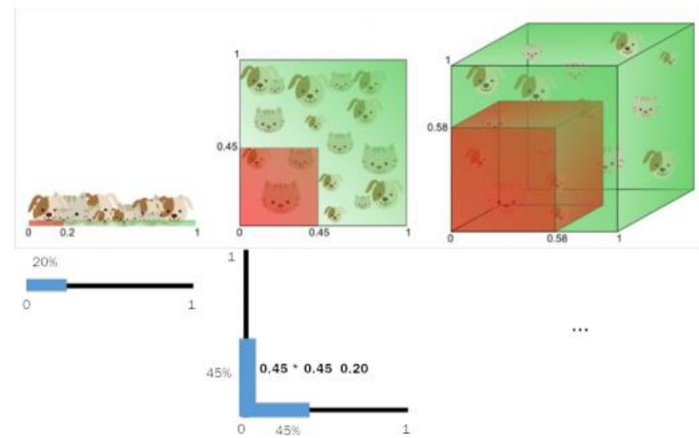
http://norman3.github.io/prml/docs/chapter01/4.html

# The Curse of Dimensionality

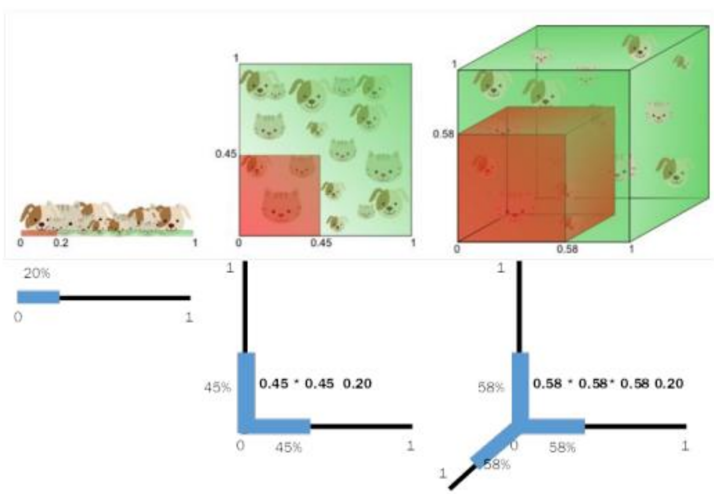
1차원에서 20%의 공간을 채우기 위해서는, 변수 1개당 20%의 데이터만 있으면 된다.




2차원에서 20%의 공간을 채우기 위해서는, 변수 1개당 45%의 데이터가 필요하다.

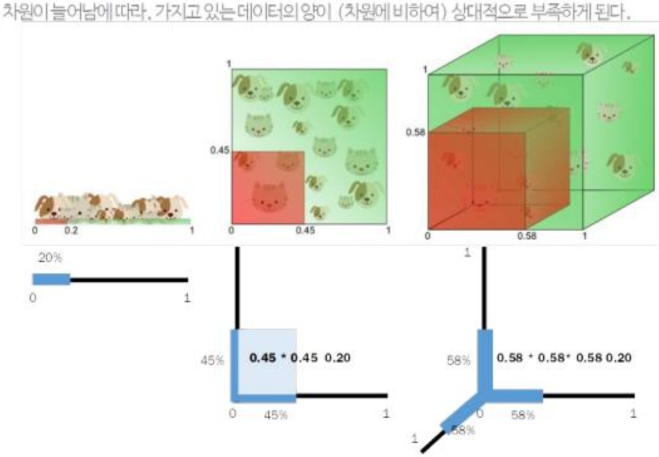



3차원에서 20%의 공간을 채우기 위해서는, 변수 1개당 58%의 데이터가 필요하다.




따라서 차원이 늘어날 수록, 같은 비율(%)의 공간을 채우기 위해, 변수 1개당 필요한 데이터의 양이 급격히 증가한다.

그러므로, 차원이 늘어남에 따라, 가지고 있는 데이터의 양이 (차원에 비하여) 상대적으로 부족하게 된다.



# Main approaches for dimensionality reduction

- Projection

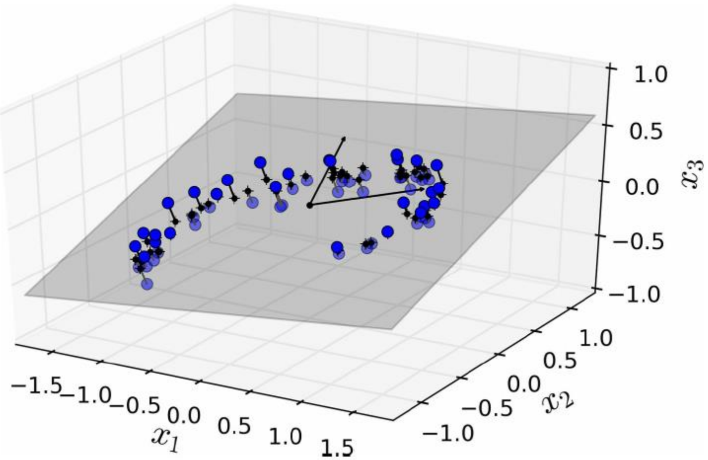



Figure 8-2. A 3D dataset lying close to a 2D subspace

⇒

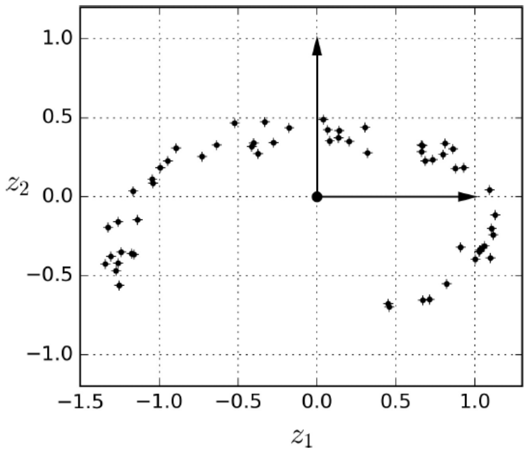



Figure 8-3. The new 2D dataset after projection

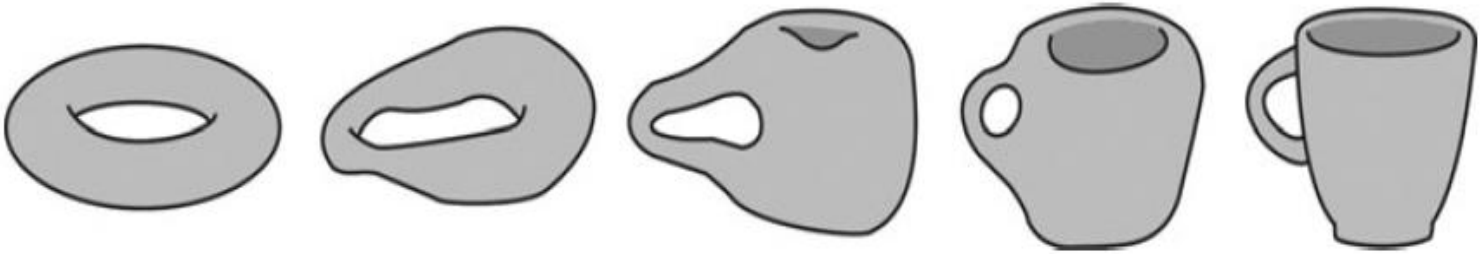
# Dimension reducing methods

□ Two approaches to reducing from dim D to dim L

- Feature selection
  - Choose $L < D$ important features, and throw away remaining features.
  - e.g. using correlation
- Feature extraction
  - Represent all features in the original data $\overrightarrow{x_i} \in \mathbb{R}^D$ with vector $\overrightarrow{z_i} \in \mathbb{R}^L$ where $L < D$

# Manifold Learning

A ***topological space*** X is called locally Euclidean if there is a non-negative integer n such that every point in X has a neighborhood which is homeomorphic to the Euclidean space $R^n$. A ***topological manifold*** is a locally Euclidean Hausdorff space.



# Manifold Learning

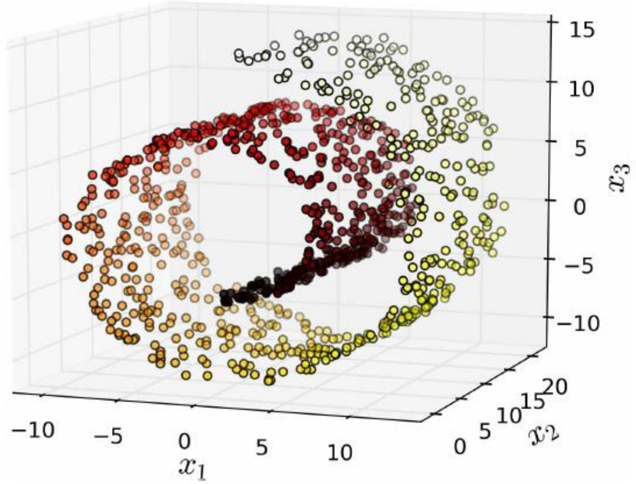



*Figure 8-4. Swiss roll dataset*

⇓

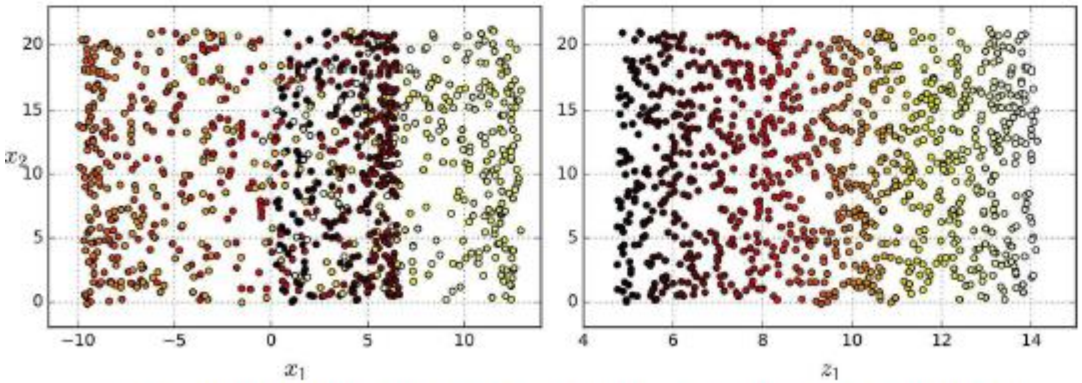



*Figure 8-5. Squashing by projecting onto a plane (left) versus unrolling the Swiss roll (right)*

# Singular Value Decomposition(SVD)

a rectangular matrix $A$ can be broken into product of 3 matrices - an orthogonal matrix $U$, a diagonal matrix $S$, and the transpose of an orthogonal matrix $V$

$$A_{mn} = U_{mm} S_{mn} V_{nn}^T$$

- 단, $U^T U = I$, $V^T V = I$;
- $U$ 행렬의 column들은 orthonormal eigenvectors of $AA^T$,
- $V$ 행렬의 column들은 orthonormal eigenvectors of $A^T A$,
- $S$ 는 a diagonal matrix containing square roots of eigenvalues from $U$ or $V$ in descending order.

http://www.openwith.net/wp-content/uploads/2016/10/SVDoverview.pdf

# Singular Value Decomposition(SVD)

Suppose M is a m × n matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization, called a ***singular value decomposition*** of M, of the form

$$M = U \sum V^*$$

where

- U is an m × m unitary matrix over K (if $k = \mathbb{R}$, unitary matrices are orthogonal matrices),
- Σ is a diagonal m × n matrix with non-negative real numbers on the diagonal,
- V is an n × n unitary matrix over K, and
- $V^*$ is the conjugate transpose of V.

# Singular Value Decomposition(SVD)

$$\underset{m \times n}{A} = \underset{m \times m}{U} \; \underset{m \times n}{\Sigma} \; \underset{n \times n}{V^T} \tag{1}$$


There are several facts about SVD:

(a) $\text{rank}(A) = \text{rank}(\Sigma) = r$.

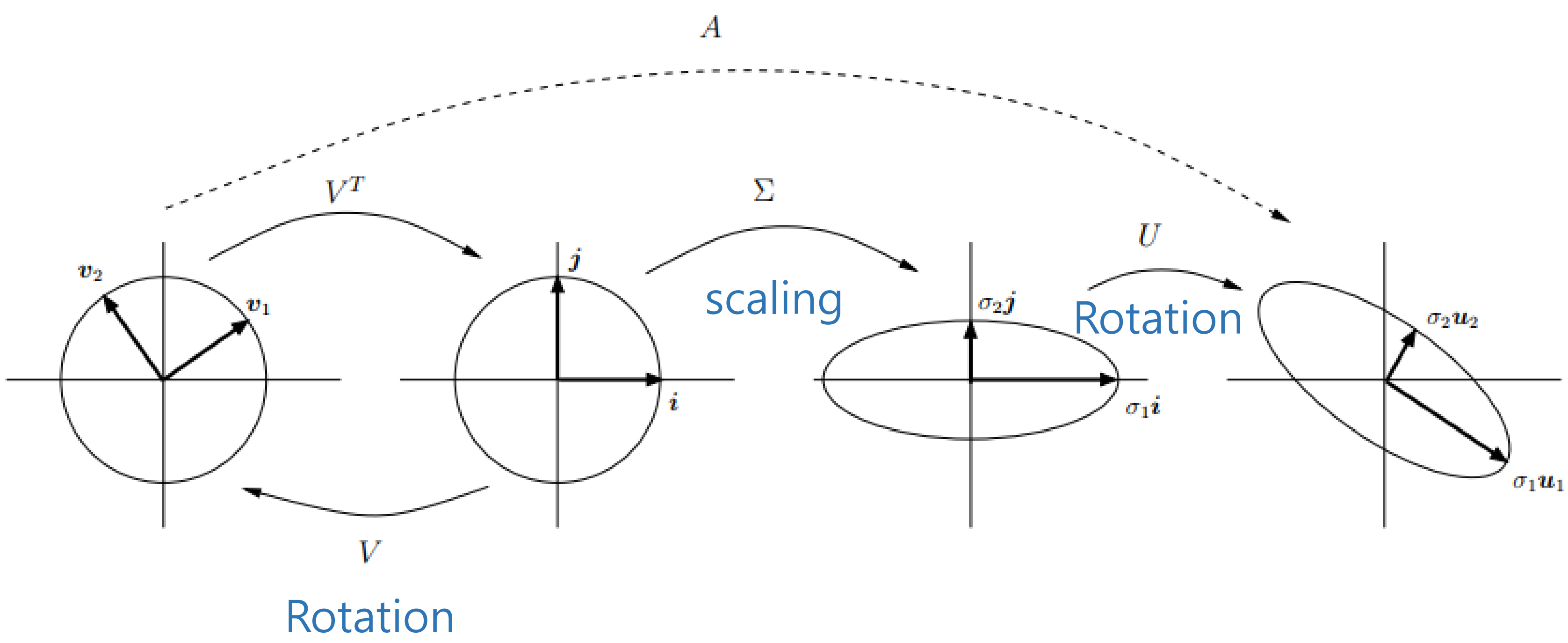
(b) The column space of $A$ is spanned by the first $r$ columns of $U$.

(c) The null space of $A$ is spanned by the last $n - r$ columns of $V$.

(d) The row space of $A$ is spanned by the first $r$ columns of $V$.

(e) The null space of $A^T$ is spanned by the last $m - r$ columns of $U$.

# Singular Value Decomposition(SVD)



# Singular Value Decomposition(SVD)

## Computing PCA

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing $C$ already takes $O(nd^2)$ time (very expensive)

Method 2: singular value decomposition (SVD)

Find $\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}_{n\times n}^\top$

where $\mathbf{U}^\top\mathbf{U} = I_{d\times d}$, $\mathbf{V}^\top\mathbf{V} = I_{n\times n}$, $\Sigma$ is diagonal

Computing top $k$ singular vectors takes only $O(ndk)$

Relationship between eigendecomposition and SVD:

Left singular vectors are principal components ($C = \mathbf{U}\Sigma^2\mathbf{U}^\top$)

# Principal Components Analysis

- Main idea

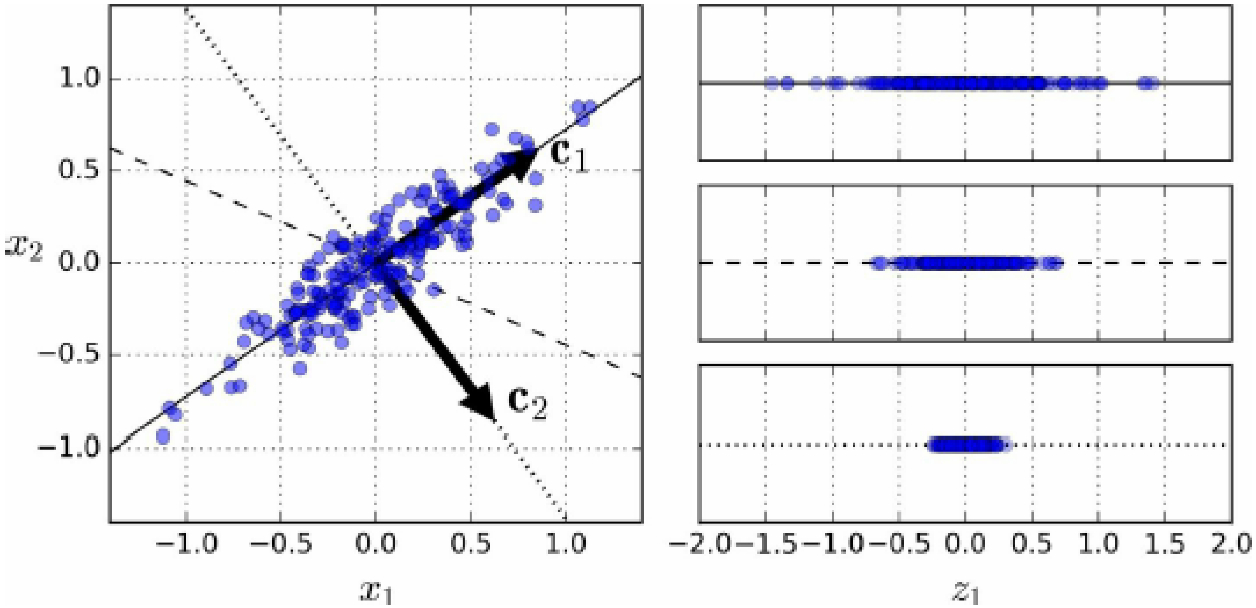



*Figure 8-7. Selecting the subspace onto which to project*

결론 : 차원을 줄일 때 가장 잘 흐트러뜨리는 방향으로 줄여보자!
즉, 분산이 가장 크게끔 회전(?)시킨 후 나머지 축을 제거하자.

# Covariance matrix

- 확률변수 $X, \mathrm{Y}$에 대해서 공분산(***covariance***)은 아래와 같이 정의된다

$$
\begin{aligned}
cov(X,Y) &= E[(X - E[X]) \cdot (Y - E[Y])] \\
&= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\
&= E[XY] - E[X]E[Y]
\end{aligned}
$$

- The ***covariance matrix*** of $X = \{X_1, \cdots, X_n\}$ is

$$
\begin{pmatrix}
\mathrm{var}(x_1) & \mathrm{cov}(x_1, x_2) & \cdots & \mathrm{cov}(x_1, x_n) \\
\mathrm{cov}(x_2, x_1) & \mathrm{var}(x_2) & \cdots & \mathrm{cov}(x_2, x_n) \\
\vdots & \vdots & \cdots & \vdots \\
\mathrm{cov}(x_n, x_1) & \mathrm{cov}(x_n, x_2) & \cdots & \mathrm{var}(x_n)
\end{pmatrix}
$$

# Covariance matrix

- Some properties of covariance matrix
  - Symmetric
  - Positive semi definite

- A matrix $M$ is called ***positive semi definite*** if
$$x^T M x \geq 0$$
for all $x \in \mathbb{R}^n$.

- In statistics, the covariance matrix of a multivariate probability distribution is always positive semi-definite; and it is positive definite unless one variable is an exact linear function of the others. Conversely, every positive semi-definite matrix is the covariance matrix of some multivariate distribution.

# Principal Component Analysis (PCA)

- 우리의 D차원의 data가 N개 있고, L개의 feature 로 축소시키려고 한다. 이 때, L 차원으로 축소한 데이터와 원래의 데이터의 차이를 최소화 하는것이 우리의 목표이므로 L차원으로 축소한 데이터를 다시 D차원으로 복귀시켜주는 행렬 W를 이용하여 reconstruction error를 우리의 cost function으로 둘 수 있다.

$$J(W,Z) = \sum_{i=1}^{N} ||\vec{x_i} - W\vec{z_i}||^2 = ||X - WZ^T||_F^2$$

With low-dimensional encoding $Z \in \mathbb{R}^{N \times L}$ with rows $\vec{z_i} \in \mathbb{R}^L$ and orthonormal matrix $W \in \mathbb{R}^{D \times L}$.

- The optimal solution is $\widehat{W_L} = V_L(L$ eigenvectors with largest eigenvalues of $\hat{\sum} = \frac{1}{N}\sum_{i=1}^{N} \vec{x_i}\,\vec{x_i}^T$, and $\widehat{\vec{z_i}} = \widehat{W}^T\vec{x_i}$.

  즉, $X$의 covariance matrix의 eigenvalue를 구해서 높은 순서대로 $L$개를 뽑아서 그 eigenvalue의 eigenvector들을 column vector로 만든 행렬이 $V_L$이다. 주의할 점은 $X$의 평균을 0벡터 $(0, \cdots, 0)$으로 만들어야 한다.

# Principal Component Analysis (PCA)

- Minimize reconstruction error!

$$J(w_1, z_1) = \sum_i ||x_i - z_{i1} w_1||^2$$
$$= \sum_i (x_i^T x_i - 2 z_{i1} w_1^T x_i + z_{i1}^2 w_1^T w_1)$$
$$= \sum_i (x_i^T x_i - 2 z_{i1} w_1^T x_i + z_{i1}^2)$$

- 최소값을 가지면 미분해서 0 이 되어야 하므로

$$\frac{\partial J(w_1, z_1)}{\partial z_{i1}} = 0 \;=>\; \boldsymbol{z_{i1} = w_1^T x_i} \quad \textbf{(A)}$$

이 식을 위에 대입하면

$$J(w_1, z_1) = \sum_i (x_i^T x_i - 2 z_{i1}^2 + z_{i1}^2)$$
$$= constant \; - \sum_i z_{i1}^2$$

- Since $E[z_{i1}] = E[w_1^T x_1] = w_1^T E[x_i] = 0$(∵PCA를 하기 위해 평균을 0 벡터로 맞추고 시작하였음), we have

$$Var[z_{i1}] = E\left[z_{i1}^2\right] - E[z_{i1}]^2 = \frac{1}{N} \sum_i z_{i1}^2 \, . \quad \textbf{(B)}$$

- Thus we have

$$J(w_1, z_1) = constant \; - N \cdot Var[z_{i1}].$$

즉, Minimize reconstruction error = maximize variance

# Principal Component Analysis (PCA)

- Using $\mathrm{z_{i1}} = \mathrm{w_1^T} x_1$ $and$ $\hat{\Sigma} = \frac{1}{N} x_i x_i^T$, we have

$$\frac{1}{N}\sum_i z_{i1}^2 = \frac{1}{N}\sum_i w_1^T x_i x_i^T w_1 = w_1^T \hat{\Sigma} w_1 \qquad \textbf{(C)}$$

where $\hat{\Sigma} = \frac{1}{N} x_i x_i^T$ is the sample covariance matrix of $X$.

(원래 sample covariance matrix 는 $\hat{\Sigma} = \frac{1}{N}(x_i - \bar{x})(x_i - \bar{x})$^$T$ 인데

우리는 PCA의 전처리로 $\bar{x} = 0$으로 만들기 때문에 위 식을 얻는다)

- SVM에서 처럼 Lagrange dual function을 사용해 보면, 우리는 함수

$$\tilde{J}(w_1) = w_1^T \hat{\Sigma} w_1 + \lambda_1 (w_1^T w_1 - 1)$$

를 maximize 하면 된다.

- 최대값을 가지면 미분해서 0이 되어야 하므로

$$\frac{\partial \tilde{J}(w_1)}{\partial w_1} = 2\hat{\Sigma} w_1 - 2\lambda_1 w_1 = 0 \Rightarrow \hat{\Sigma} w_1 = \lambda_1 w_1.$$

**즉, $\lambda_1$은 sample covariance matrix $\hat{\Sigma}$ 의 eigenvector이고,**
**$w_1$은 $\lambda_1$의 eigenvector 이다.**

# Principal Component Analysis (PCA)

- Using the equation $\hat{\Sigma} w_1 = \lambda_1 w_1$, we have
$$w_1^T \hat{\Sigma} w_1 = w_1^T \lambda_1 w_1 = \lambda_1 = \mathrm{Var}[\mathrm{z}_{\mathrm{i1}}]$$
by (B) and (C).

- 정리해보면
minimize reconstruction error
= maximize variance
= choose maximum eigenvalues

# Principal Component Analysis (PCA)

## □ Choosing the number of dimensions

- Reconstruction error with $L$ dimensios

$$E(D,L) = \frac{1}{N}\sum_i \left|\left|x_i - Wz_i\right|\right|^2$$

- Theorem. $E(D,L) = \sum_{k=L+1}^{D} \lambda_k$

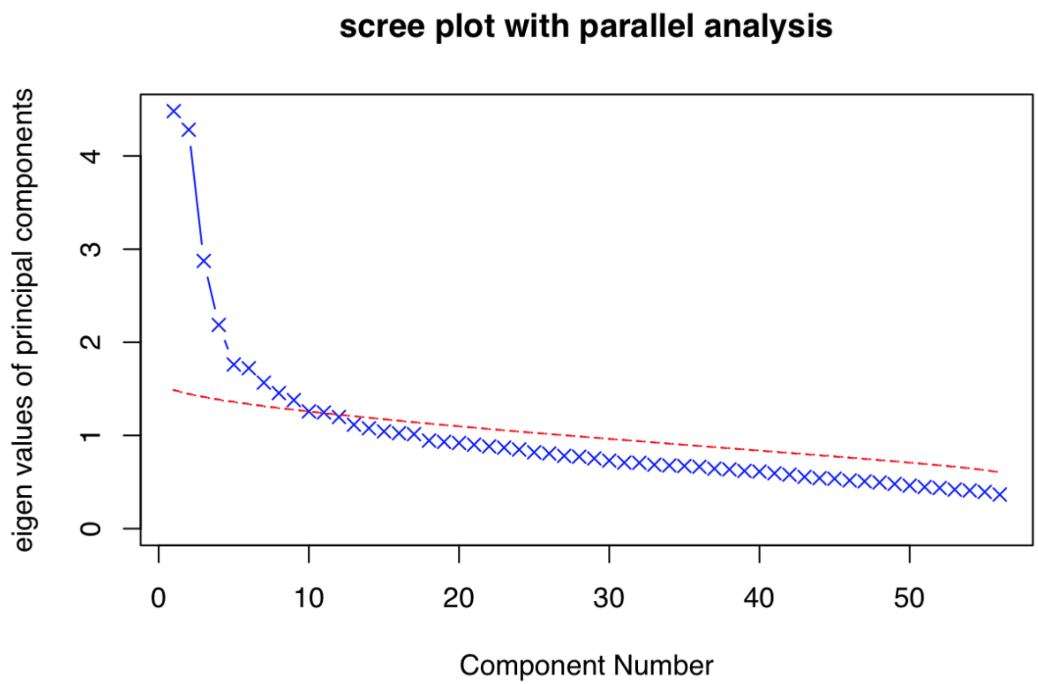
(proof) $\left|\left|x_i - w_1 z_1 - \cdots - w_D z_D\right|\right|^2$

$$= (x_i - z_1 w_1 - \cdots - z_D w_D)^{\mathrm{T}}(x_i - z_1 w_1 - \cdots - z_D w_D)$$
$$= \left(x_i^T - w_1^T z_1^T - \cdots - w_D^T z_D^T\right)(x_i - z_1 w_1 - \cdots - z_D w_D)$$
$$= \left(x_i^T x_i - x_i^T z_1 w_1 - \cdots - x_i^T z_D w_D\right)$$
$$+(-w_1^T z_1^T x_i + w_1^T z_1^T w_1 z_1 + \cdots + w_1^T z_1^T z_D w_D)$$
$$+\cdots$$
$$+(-w_D^T z_D^T x_i + w_D^T z_D^T w_1 z_1 + \cdots + w_D^T z_D^T w_D z_D)$$
$$= x_i^T x_i - z_1^T z_1 - \cdots - z_D^T z_D \ (\because z_i = W x_i)$$
$$+(-z_1^T z_1 + 0 + \cdots + 0)$$
$$+\cdots$$
$$+(-z_D^T z_D + 0 + \cdots + 0)$$

# Principal Component Analysis (PCA)

- 차원을 정하는 방법
  - $\lambda_1 + \cdots + \lambda_D \geq 0.8(or\ 0.9)$ 이 되는 최소의 $D$ 를 찾자!

- **Scree plot**



# Dual problem

- Constrained optimization :

$$\min_{x} f(x) \text{ s.t. } g(x) \leq 0, h(x) = 0.$$

- Lagrange method:

  Lagrange prime function : $L(x, \alpha, \beta) = f(x) + \alpha g(x) + \beta h(x)$

  Lagrange multiplier : $\alpha \geq 0, \beta$

  Lagrange dual function :

$$d(\alpha, \beta) = \inf_{x \in X} L(x, \alpha, \beta) = \min_{x \in X} L(x, \alpha, \beta).$$

  Then we have

$$\boxed{\max_{\alpha \geq 0, \beta} L(x, \alpha, \beta) = \begin{cases} f(x) : if \;\; x\, is\, feasible \\ \qquad \infty : \;\; o.w. \end{cases}}$$

우리의 optimization problem

$$\min_{w,b} ||w|| \text{ such that } (w \cdot x_i + b)y_i \geq 1, \; \forall i$$

을 대입해보면 아래와 같다.

# Kernel PCA

Fig. 1 demonstrates the basic idea behind nonlinear kernel PCA. Consider a random field $\mathbf{y} = (y_1, y_2)^T \in \mathbb{R}^2$. If $\mathbf{y}$ is non-Gaussian, $y_1$ and $y_2$ can be nonlinearly related to each other in $\mathbb{R}^2$. In this case, linear PCA or K–L expansion attempt to fit

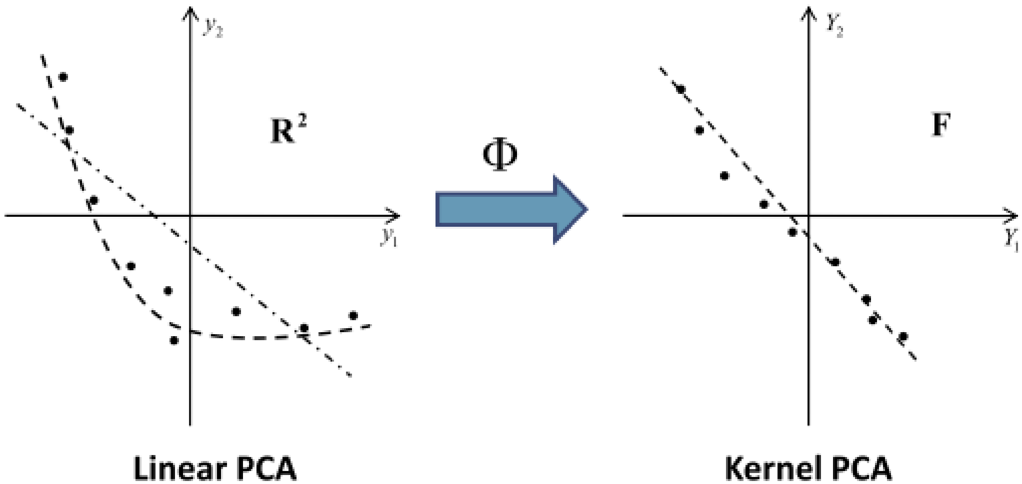



**Fig. 1.** Basic idea of KPCA. Left: in this non-Gaussian case, the linear PCA is not able to capture the nonlinear relationship among the realizations in the original space. Right: after the nonlinear mapping $\Phi$, the realizations becomes linearly related in the feature space $F$. Linear PCA or K–L expansion can now be performed in $F$.

# Other Methods

- Locally linear embedding(LLE)
- Multidimensional scaling(MDS)
- Isomap
- t-Distributed Stochastic neighbor embedding
- Linear discriminant analysis(LDA)