Dimension Reduction

WEEK 6

Outline

Overview

Random Projection method

Principle component analysis

Nonlinear manifold learning

Dimension Reduction

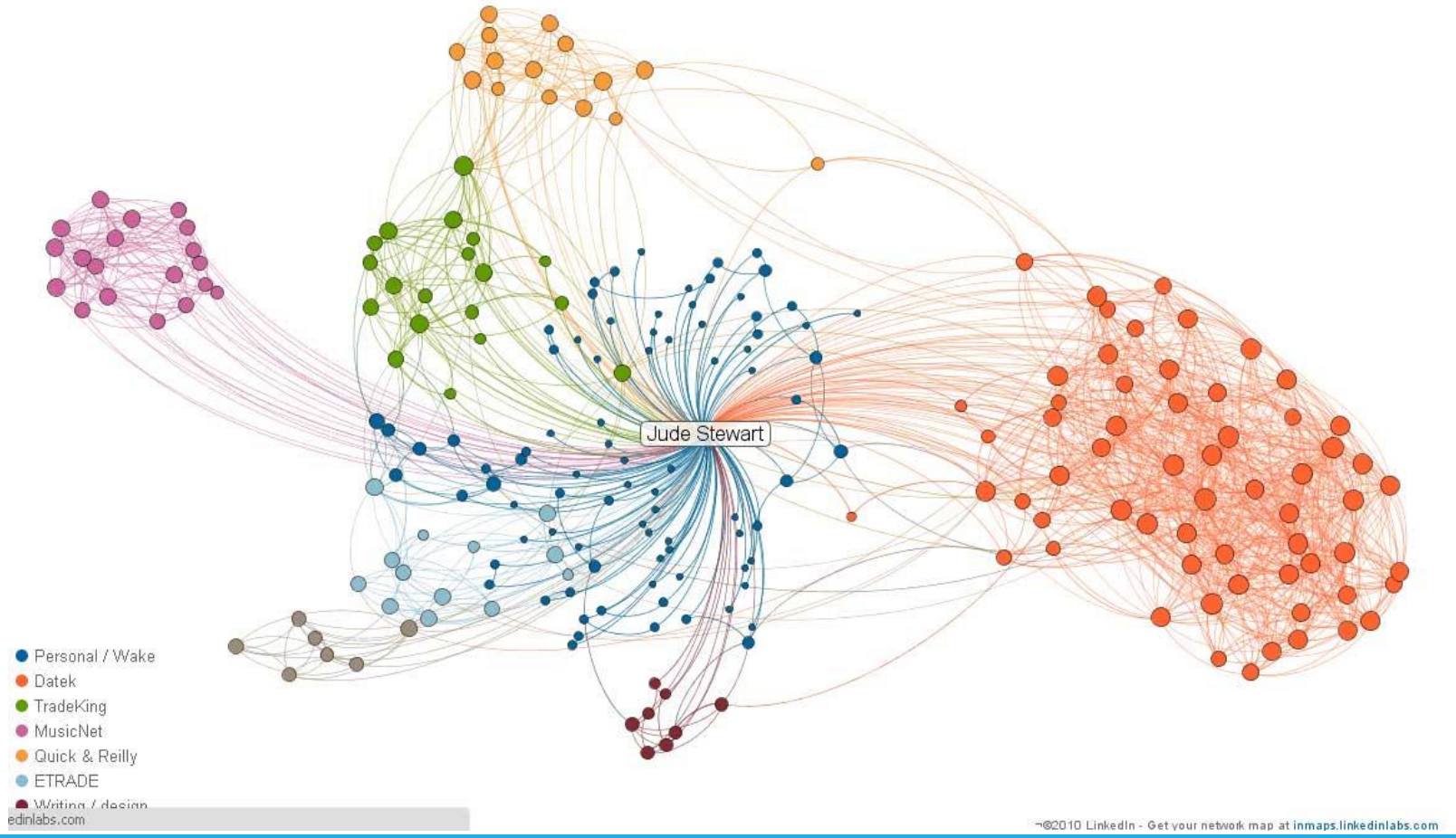
The process to reduce number of random variables under consideration.

- Feature selection (find subset features)
- Feature extraction (transform features into smaller #)

Modules:

sklearn.decomposition; sklearn.manifold; sklearn.random_projection

The famous Inmaps by LinkedIn



Randomized Projection

Lemma 2.1 *A set of n points $\mathbf{u}_1 \dots \mathbf{u}_n$ in $\mathbb{R}^d$ can be projected down to $\mathbf{v}_1 \dots \mathbf{v}_n$ in $\mathbb{R}^k$ such that all pairwise distances are preserved:*

$$(1-\epsilon)\|\mathbf{u}_i - \mathbf{u}_j\|^2 \leq \|\mathbf{v}_i - \mathbf{v}_j\|^2 \leq (1+\epsilon)\|\mathbf{u}_i - \mathbf{u}_j\|^2$$

if

$$k > \frac{9 \ln n}{\epsilon^2 - \epsilon^3}, \text{ and, } 0 \leq \epsilon \leq 1/2$$

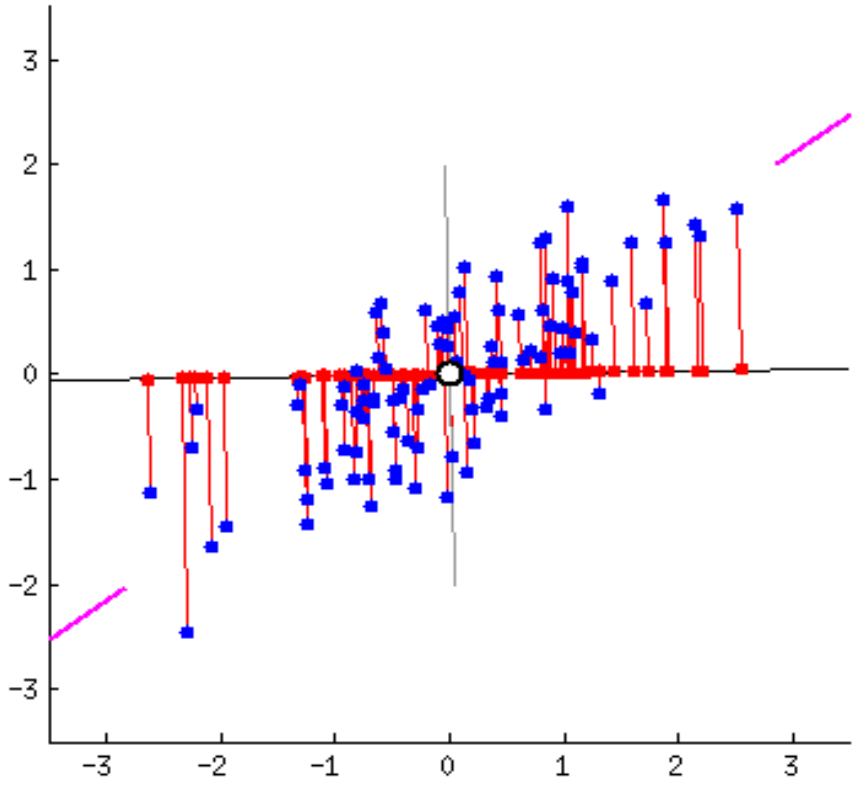
Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation)

Key

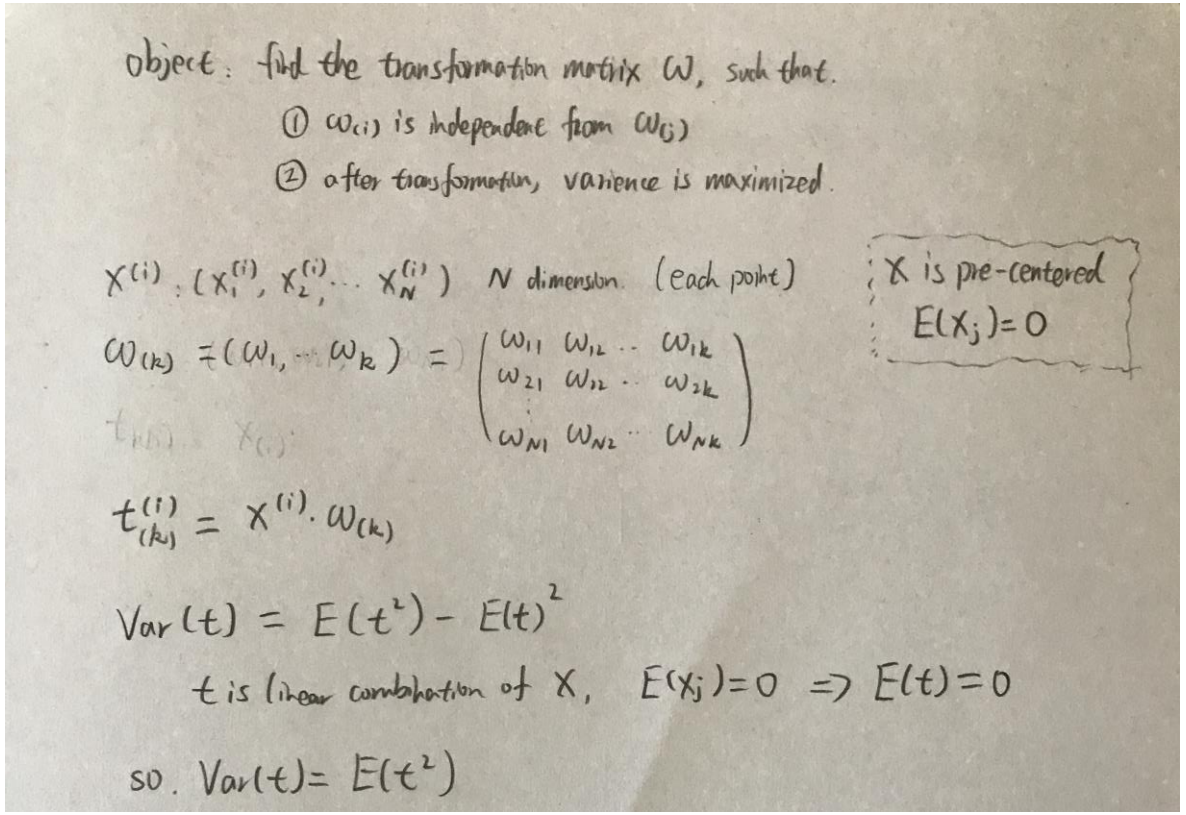
1. Orthogonal transformation
2. Linearly uncorrelated components

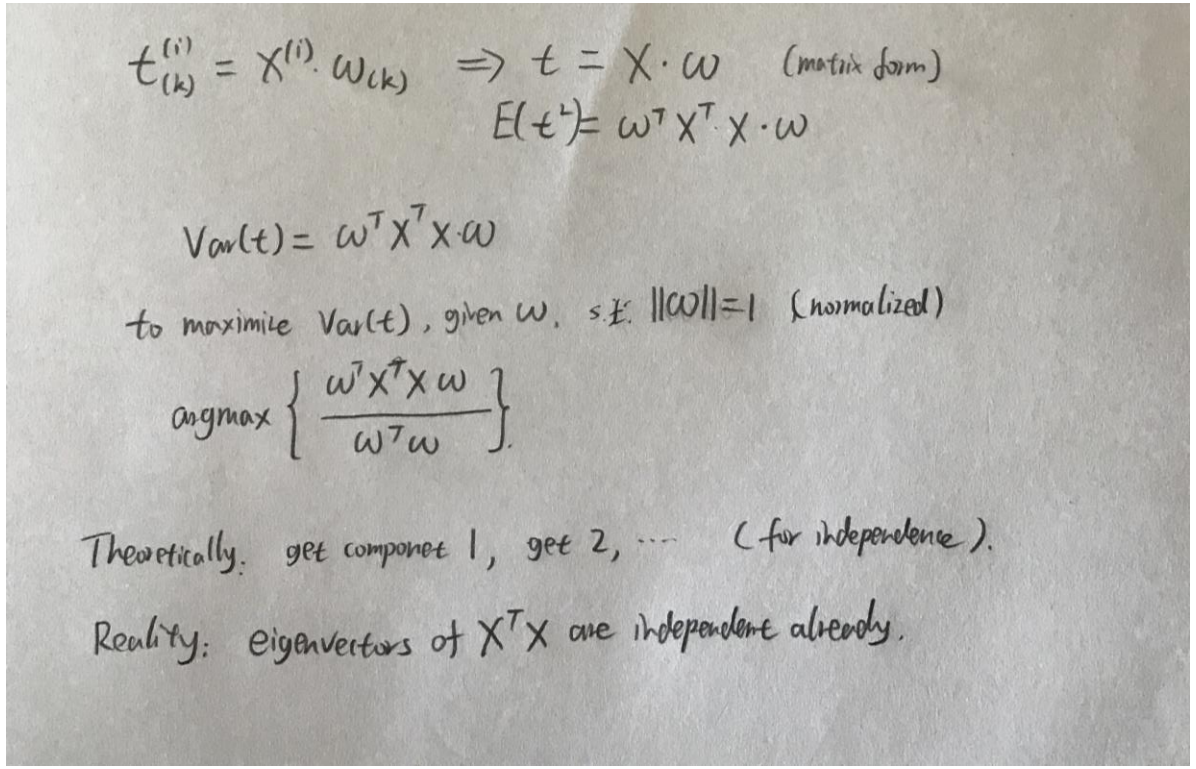
Principal Component Analysis



the best, in the mean-square error sense, linear dimension reduction technique

PCA (formula)


object: find the transformation matrix W, such that.
① $w_{(i)}$ is independent from $w_{(j)}$
② after transformation, varience is maximized.
$X^{(i)} : (X_1^{(i)}, X_2^{(i)}, \cdots X_N^{(i)})$ N dimension (each point)
X is pre-centered
$E(X_j) = 0$
$w_{(k)} = (w_1, \cdots w_k) = \begin{pmatrix} w_{11} & w_{12} \cdots & w_{1k} \\ w_{21} & w_{22} \cdots & w_{2k} \\ \vdots & & \\ w_{N1} & w_{N2} \cdots & w_{Nk} \end{pmatrix}$
$t_{(k)}^{(i)} = X^{(i)} \cdot w_{(k)}$
$Var(t) = E(t^2) - E(t)^2$
t is linear combination of X, $E(X_j) = 0 \Rightarrow E(t) = 0$
so. $Var(t) = E(t^2)$


$t_{(k)}^{(i)} = X^{(i)} \cdot w_{(k)} \Rightarrow t = X \cdot w$ (matrix form)
$E(t^2) = w^T X^T X \cdot w$
$Var(t) = w^T X^T X \cdot w$
to maximize Var(t), given w. s.t. $\|w\| = 1$ (normalized)
$\text{argmax} \left\{ \frac{w^T X^T X w}{w^T w} \right\}$.
Theoretically: get componet 1, get 2, ··· (for independence).
Reality: eigenvectors of $X^T X$ are independent already.

PCA (calculation)

Two way to gee PCA result.

① covariance matrix eigenvector decomposition

a. calculate X^TX (A)

b. $A = Q \Lambda Q^{-1}$

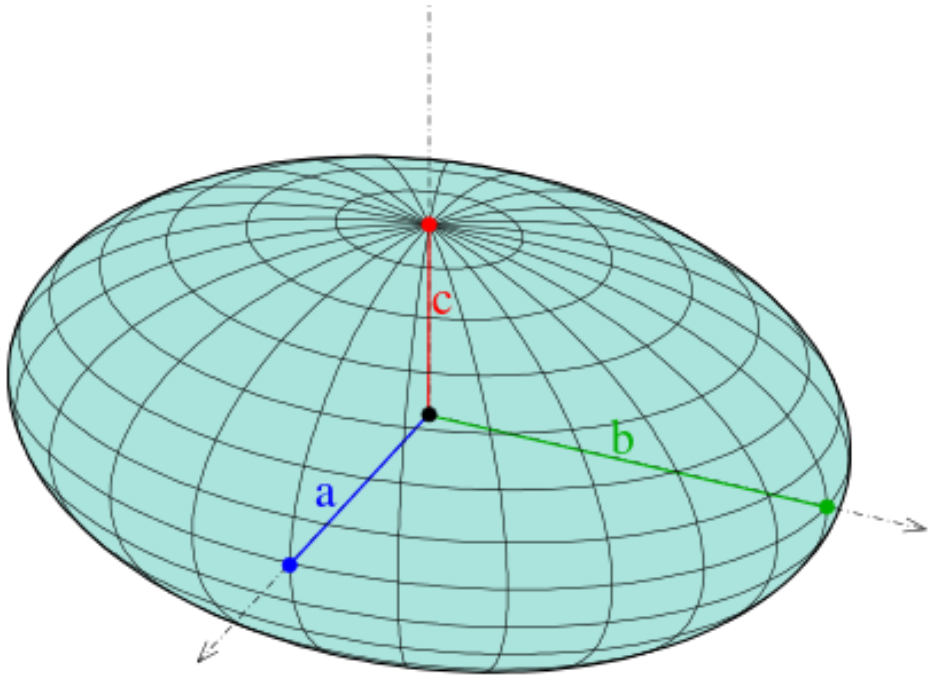
c. Λ contains the eigenvalues, Q contains the eigenvector.

② singular vector decomposition. (SVD)

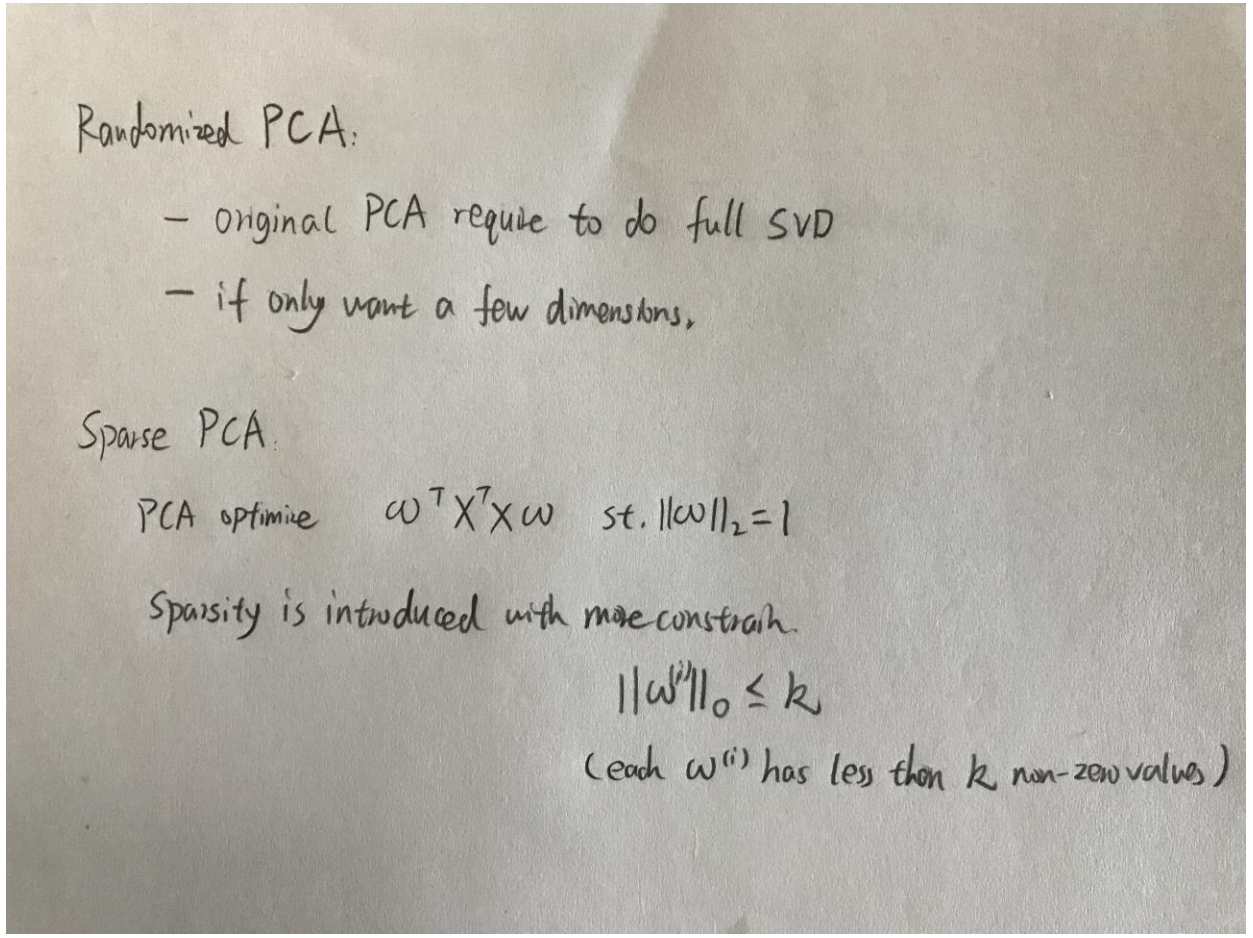
a. $X = U \Sigma W^T$ $\quad X^TX = W \Sigma^2 W^T$ (same!)

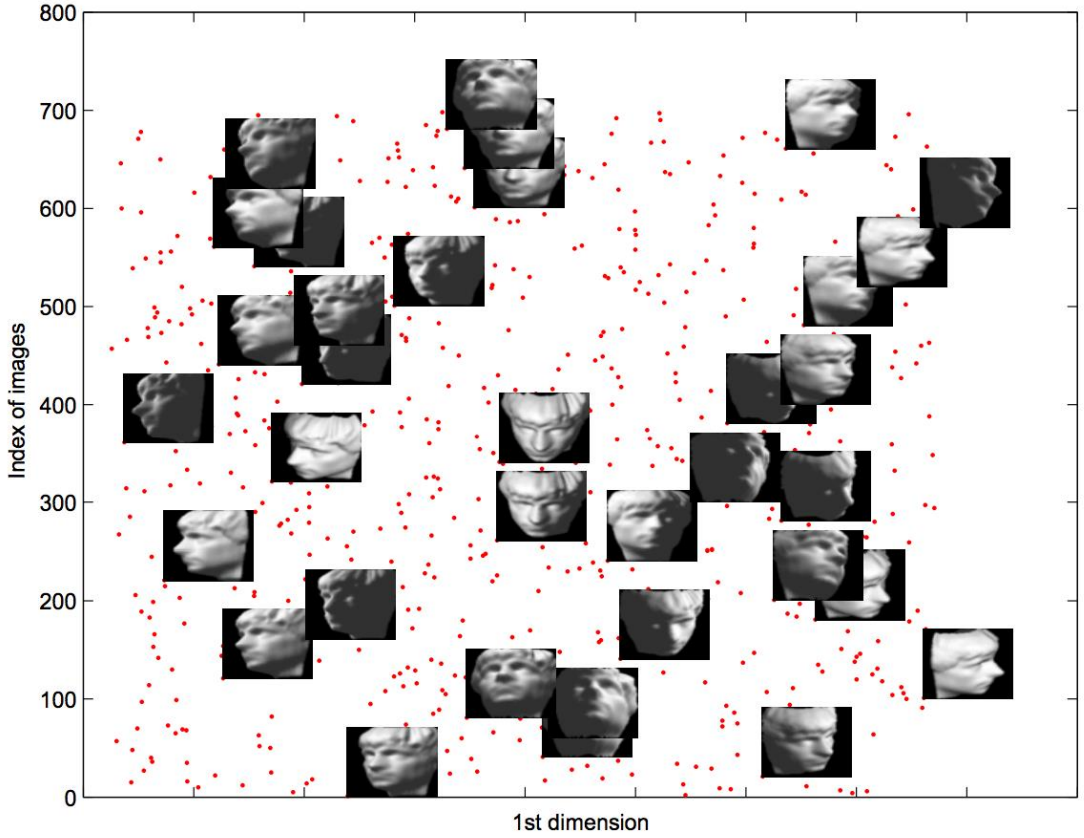
b. $T = X \cdot W = U \Sigma$ (done ~).

SVD is usually prefered because of efficient algorithm.



PCA (extension)





Manifold learning

"the dimensionality of many data sets is only artificially high"

Multidimensional Scaling

$$Strain_D(x_1, x_2, ..., x_N) = \left(1 - \frac{\left(\sum_{i,j} d_{ij} \cdot \langle x_i, x_j \rangle\right)^2}{\sum_{i,j} d_{ij}^2 \cdot \sum_{i,j} \langle x_i, x_j \rangle^2}\right)^{1/2}$$

Isomap

Algorithm [edit]

A very high-level description of **Isomap** algorithm given below.

- Determine the neighbors of each point.
  - All points in some fixed radius.
  - *K* nearest neighbors.
- Construct a neighborhood graph.
  - Each point is connected to other if it is a *K* nearest neighbor.
  - Edge length equal to Euclidean distance.
- Compute shortest path between two node.
  - Dijkstra's algorithm
  - Floyd–Warshall algorithm
- Compute lower-dimensional embedding.
  - Multidimensional scaling