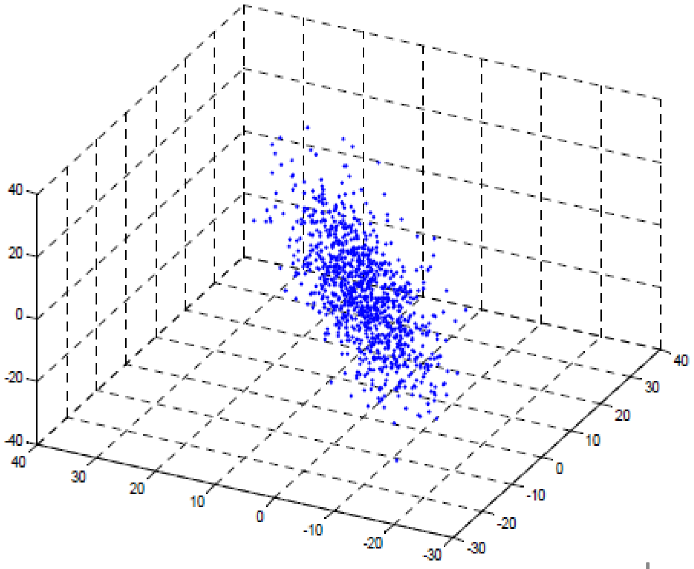
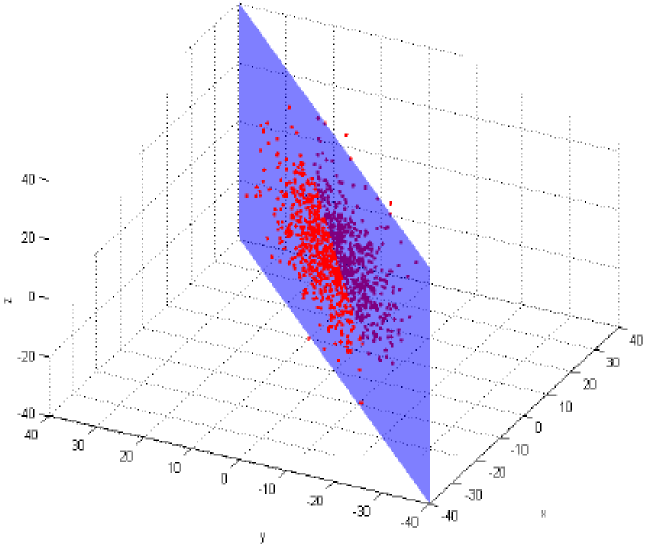
Outline

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Multi-Dimensional Scaling (MDS)
- Non-linear extensions:
  - Kernel PCA
  - Isomap

PCA

- PCA: Principle Component Analysis (closely related to SVD).
- PCA finds a **linear** projection of high dimensional data into a lower dimensional subspace such as:
  - The variance retained is maximized.
  - The least square reconstruction error is minimized.





Some PCA/SVD applications

- LSI: Latent Semantic Indexing.
- Kleinberg/Hits algorithm (compute hubs and authority scores for nodes).
- Google/PageRank algorithm (random walk with restart).
- Image compression (eigen faces)
- Data visualization (by projecting the data on 2D).

PCA

PCA steps: transform an $N \times d$ matrix X into an $N \times m$ matrix Y:

- Centralized the data (subtract the mean).
- Calculate the $d \times d$ covariance matrix: $\mathrm{C} = \frac{1}{N-1} X^T X$ *(different notation from tutorial!!!)*
  - ○ $C_{i,j} = \frac{1}{N-1} \sum_{q=1}^{N} X_{q,i} . X_{q,j}$
  - ○ $C_{i,i}$ (diagonal) is the variance of variable i.
  - ○ $C_{i,j}$ (off-diagonal) is the covariance between variables i and j.
- Calculate the eigenvectors of the covariance matrix (orthonormal).
- Select *m* eigenvectors that correspond to the largest *m* eigenvalues to be the new basis.

Eigenvectors

- If A is a square matrix, a non-zero vector **v** is an eigenvector of A if there is a scalar λ (eigenvalue) such that

$$Av = \lambda v$$

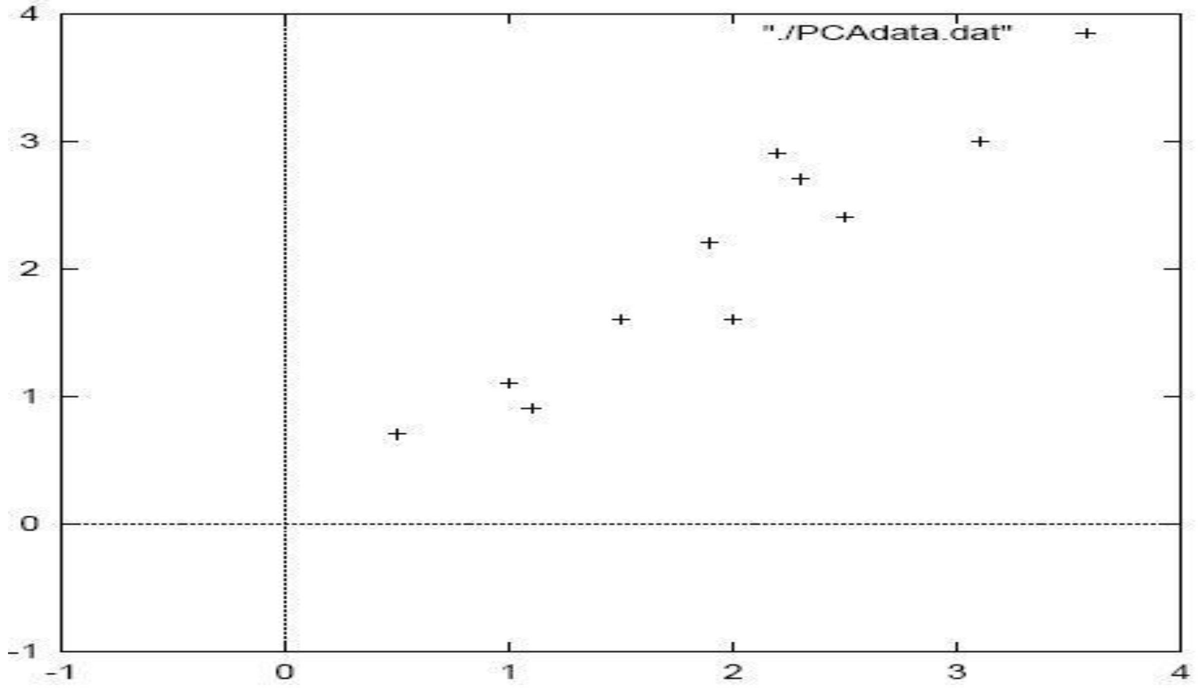
- Example: $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \begin{pmatrix} 3 \\ 2 \end{pmatrix}$

- If we think of the squared matrix as a transformation matrix, then multiply it with the eigenvector do not change its direction.

What are the eigenvectors of the identity matrix?

PCA example

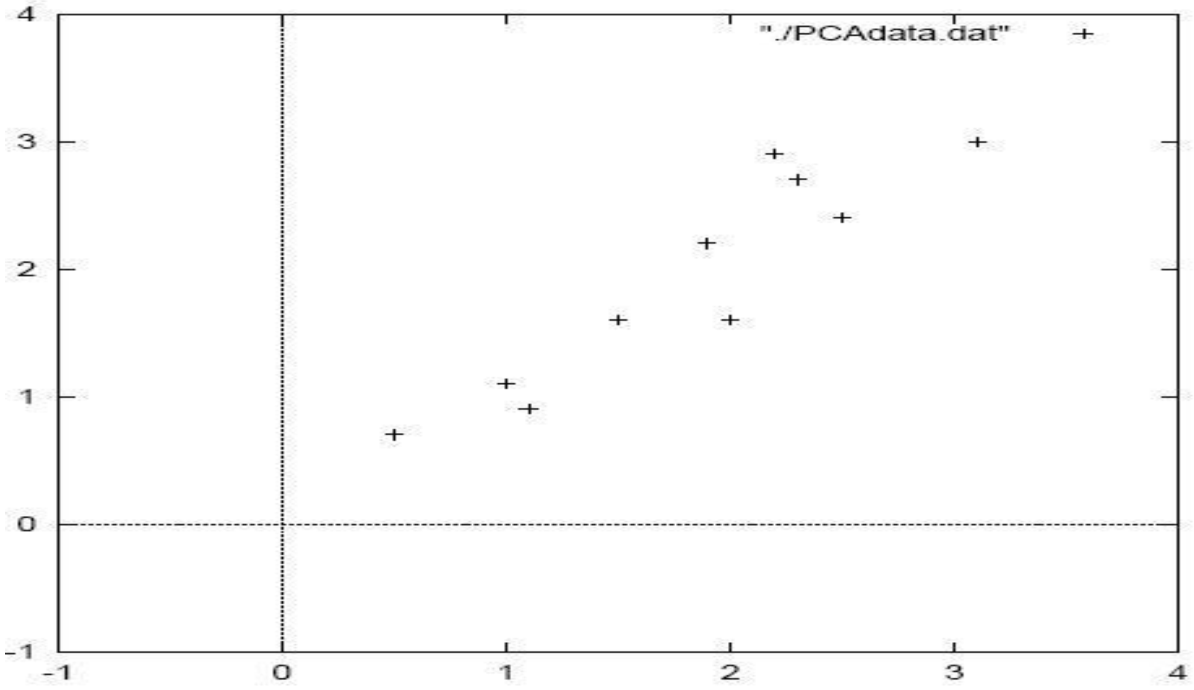
X : the data matrix with $N=11$ objects and $d=2$ dimensions.



PCA example

- *Step 1: subtract the mean and calculate the covariance matrix C.*

$$C = \begin{pmatrix} 0.716 & 0.615 \\ 0.615 & 0.616 \end{pmatrix}$$



PCA example

- *Step 2: Calculate the eigenvectors and eigenvalues of the covariance matrix:*

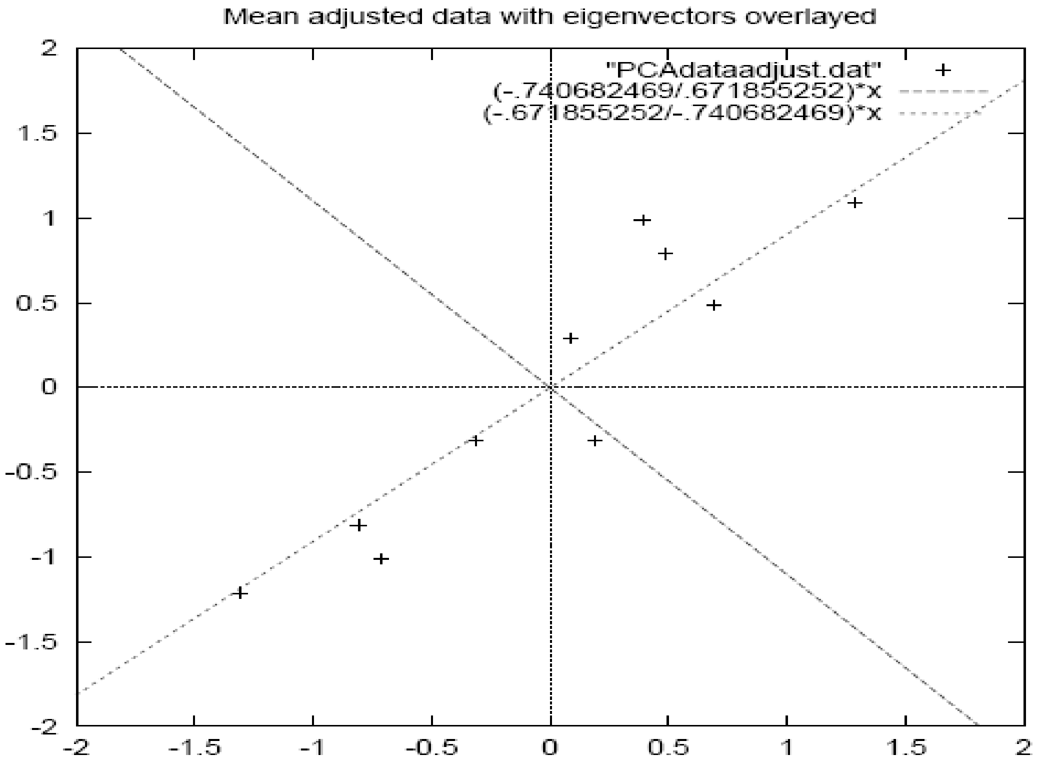
$\lambda_1 \approx 1.28$, $v_1 \approx [-0.677 \;\; -0.735]^T$, $\lambda_2 \approx 0.49$, $v_2 \approx [-0.735 \;\; 0.677]^T$

Notice that v_1 and v_2 are <span>orthonormal</span>:

$|v_1|=1$

$|v_2|=1$

$v_1 \,.\, v_2 = 0$



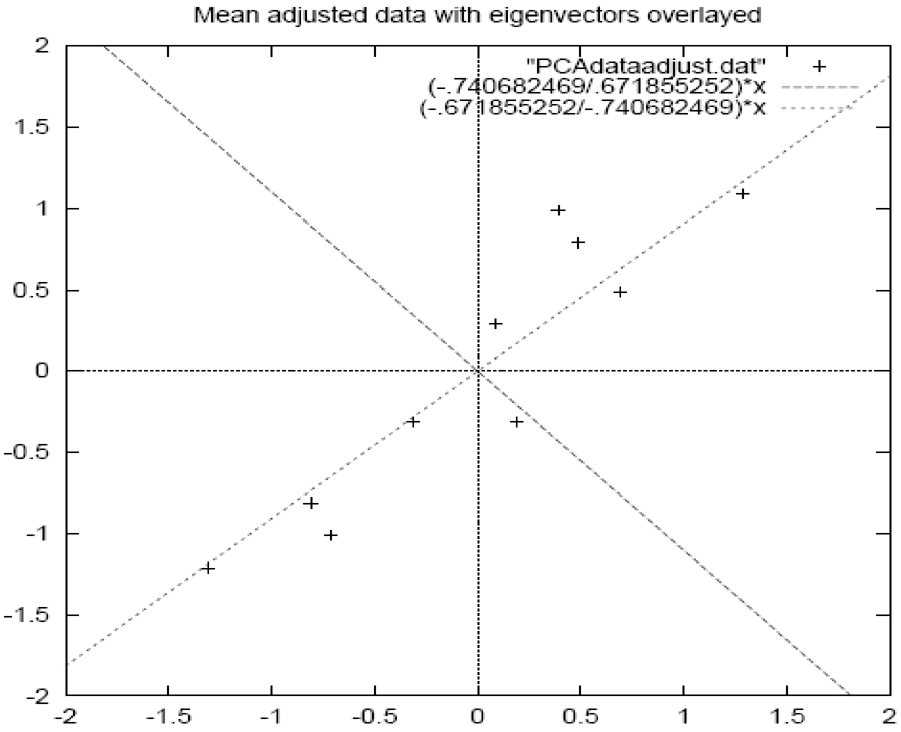
PCA example

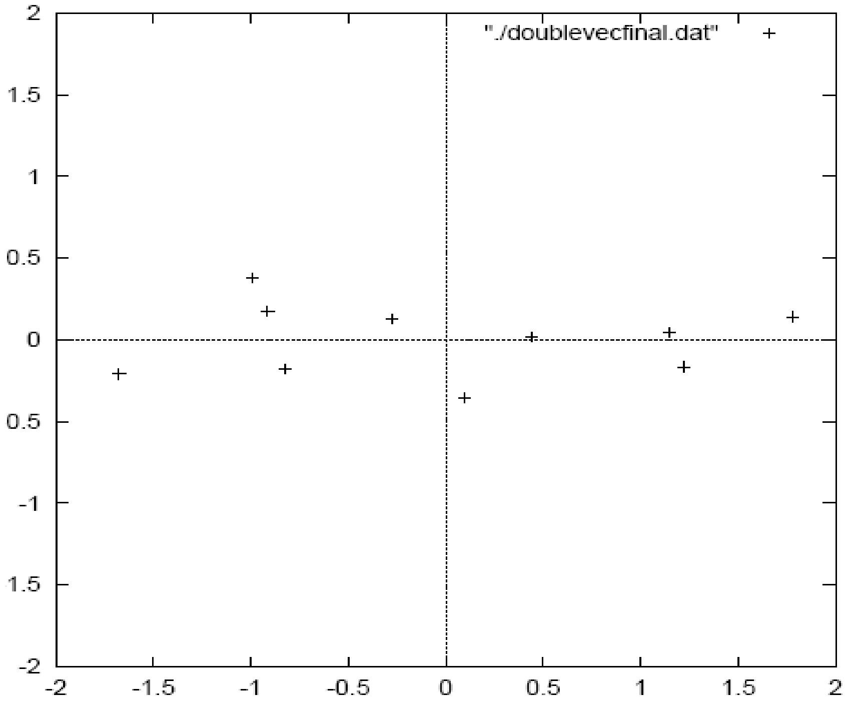
- *Step 3: project the data*

Let $V = [v_1, \dots v_m]$ is $d \times m$ matrix where the columns v_i are the eigenvectors corresponding to the largest m eigenvalues

The projected data: $Y = X\,V$ is $N \times m$ matrix.

If m=d (more precisely rank(X)), then there is no loss of information!





PCA example

- *Step 3: project the data*

$\lambda_1 \approx 1.28$, $v_1 \approx [-0.677 \;\; -0.735]^T$, $\lambda_2 \approx 0.49$, $v_2 \approx [-0.735 \;\; 0.677]^T$

The eigenvector with the highest eigenvalue is the **principle component** of the data.

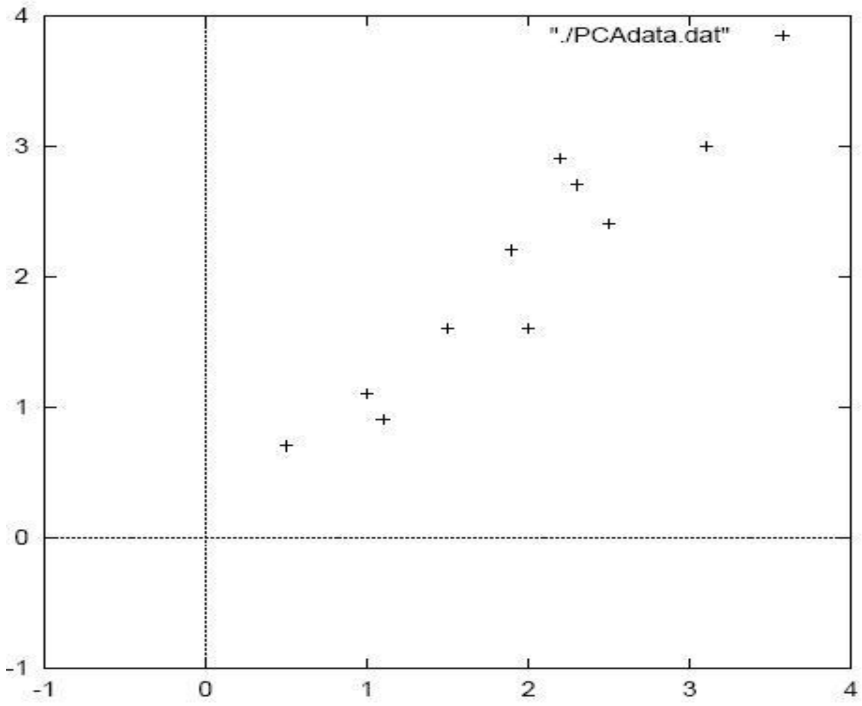
if we are allowed to pick only one dimension, the principle component is the best direction (retain the maximum variance).

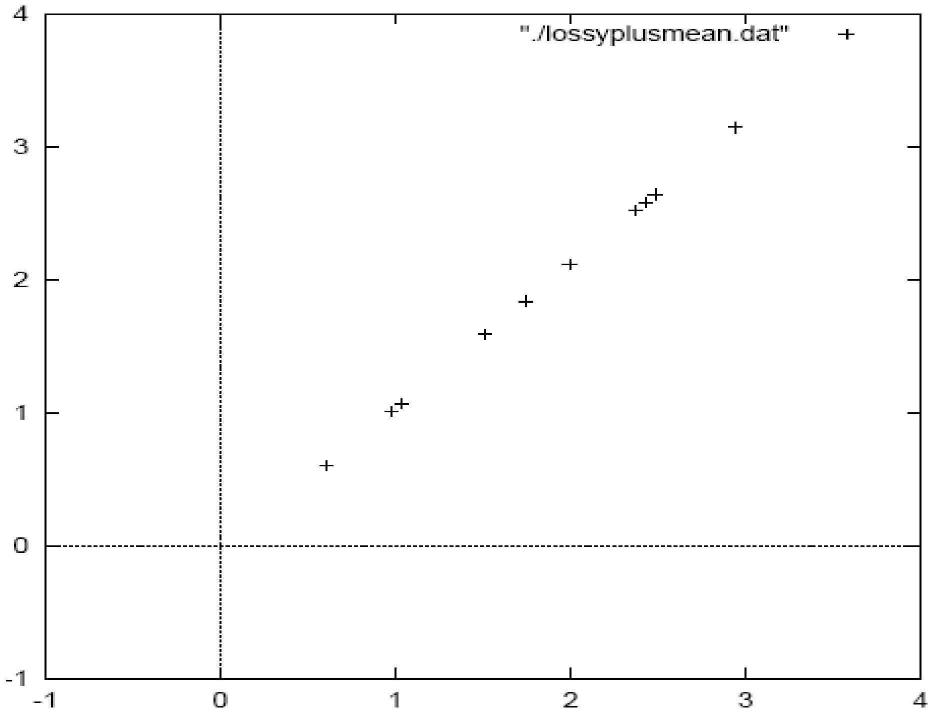
Our PC is $v_1 \approx [-0.677 \;\; -0.735]^T$

PCA example

- *Step 3: project the data*

If we select the first PC and reconstruct the data, this is what we get:





We lost variance along the other component (lossy compression!)

Useful properties

- The covariance matrix is always symmetric

$$\mathrm{C}^T = \left(\frac{1}{N-1} X^T X\right)^T = \frac{1}{N-1} X^T {X^T}^T = C$$

- The principal components of X are orthonormal

$$v_i{}^T v_j = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases}$$

- V=[v_1, … v_m], then $V^T = V^{-1}$, i.e $V^T V = I$

Useful properties

Theorem 1: if square $d \times d$ matrix S is a real and symmetric matrix ($S=S^T$) then

$$\boldsymbol{S = V \Lambda V^T}$$

Where $V = [v_1, \ldots\ v_d]$ are the eigenvectors of S and $\Lambda = diag\ (\lambda_1, \ldots\ \lambda_d)$ are the eigenvalues.

Proof:

$S\, V = V\, \Lambda$

$[S\, v_1\ \ldots\ S\, v_d] = [\lambda_1 . v_1\ \ldots\ \lambda_d . vd]$: the definition of eigenvectors.

$S = V \Lambda V^{-1}$

$S = V \Lambda V^T$ because V is orthonormal $V^{-1} = V^T$

Useful properties

The projected data: $Y = X\ V$

The covariance matrix of Y is

$$C_Y = \frac{1}{N-1} Y^T Y = \frac{1}{N-1} V^T X^T X\ V = V^T C_X V$$

$= \mathrm{V}^{\mathrm{T}}\mathrm{V}\ \Lambda\ \mathrm{V}^{\mathrm{T}}\mathrm{V}$ because the covariance matrix C_X is symmetric

$= \mathrm{V}^{-1}\mathrm{V}\ \Lambda\ \mathrm{V}^{-1}\mathrm{V}$ because V is orthonormal

$= \Lambda$

After the transformation, the covariance matrix becomes diagonal!

PCA (derivation)

- Find the direction for which the variance is maximized:

$$v_1 = argmax_{v1}\ var(Xv_1)$$

Subject to: $v_1{}^T v_1$=1

- Rewrite in terms of the covariance matrix:

$$var(Xv_1) = \frac{1}{N-1}(Xv_1)^T(Xv_1) = v_1{}^T \frac{1}{N-1} X^T X\ v_1 = v_1{}^T C\ v_1$$

- Solve via constrained optimization:

$$L(v_1, \lambda_1) = v_1{}^T C\ v_1 + \lambda_1(1 - v_1{}^T v_1)$$

PCA (derivation)

- Constrained optimization:

$$L(v_1, \lambda_1) = {v_1}^T C\, v_1 + \lambda_1(1 - {v_1}^T v_1)$$

- Gradient with respect to v_1:

$$\frac{dL(v_1, \lambda_1)}{dv_1} = 2Cv_1 - 2\lambda_1 v_1 \Rightarrow Cv_1 = \lambda_1 v_1$$

This is the eigenvector problem!

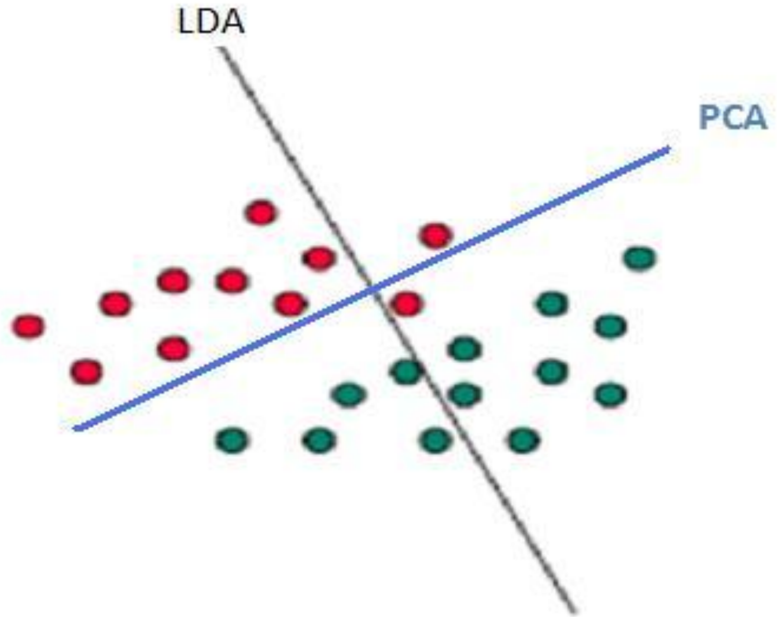
- Multiply by ${v_1}^T$:

$$\lambda_1 = {v_1}^T C\, v_1$$

The projection variance is the eigenvalue

PCA

Unsupervised: maybe bad for classification!



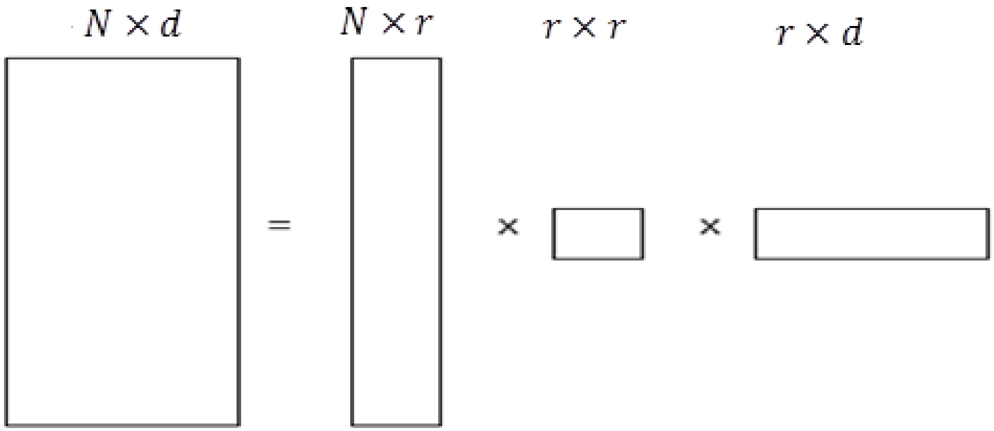
Outline

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Multi-Dimensional Scaling (MDS)
- Non-linear extensions:
  - Kernel PCA
  - Isomap

SVD

Any $N \times d$ matrix X can be uniquely expressed as:

X = U x **Σ** x **V**$^{\mathbf{T}}$



- r is the rank of the matrix X (# of linearly independent columns/rows).
- U is a column-orthonormal $N \times r$ matrix.
- Σ is a diagonal $r \times r$ matrix where the singular values σ_i are sorted in descending order.
- V is a column-orthonormal $d \times r$ matrix.

SVD example

doc-to-concept similarity matrix

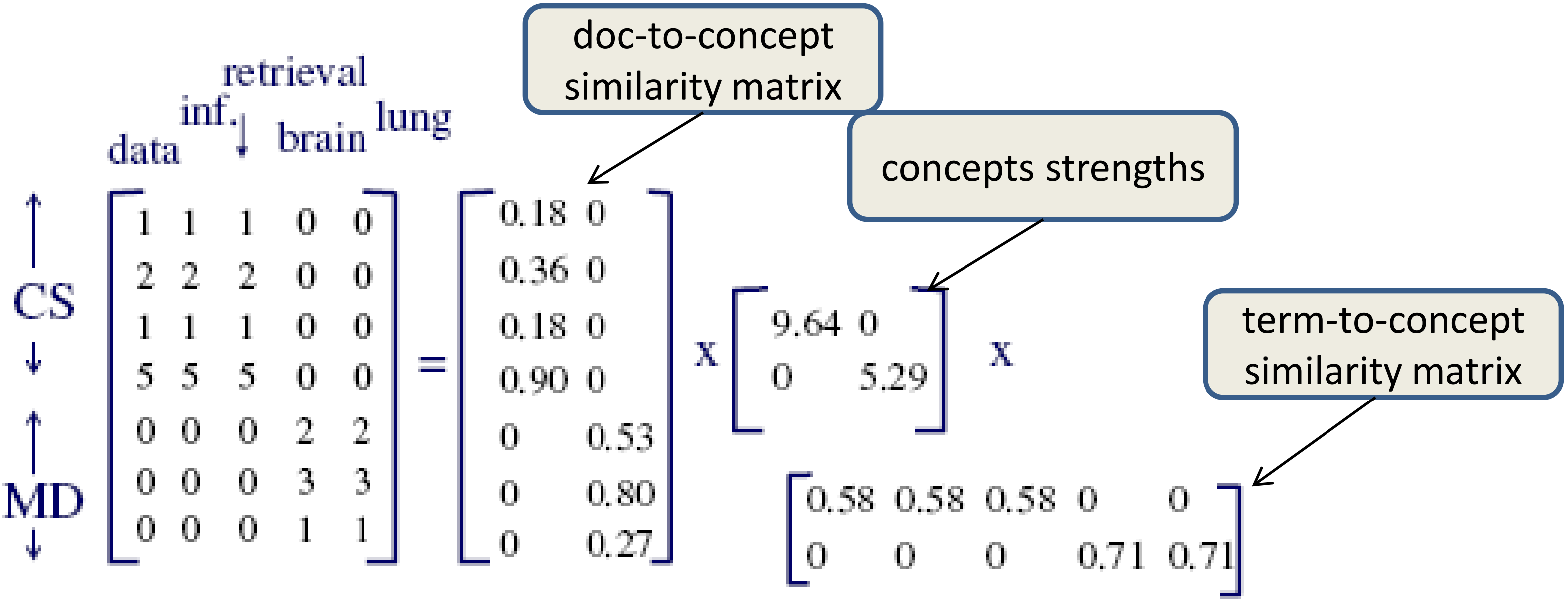
concepts strengths

term-to-concept similarity matrix

$$\begin{array}{c|ccccc} & \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \\ \hline \text{CS} & 1 & 1 & 1 & 0 & 0 \\ & 2 & 2 & 2 & 0 & 0 \\ & 1 & 1 & 1 & 0 & 0 \\ & 5 & 5 & 5 & 0 & 0 \\ \text{MD} & 0 & 0 & 0 & 2 & 2 \\ & 0 & 0 & 0 & 3 & 3 \\ & 0 & 0 & 0 & 1 & 1 \end{array} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \mathrm{x} \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \mathrm{x} \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

The rank of this matrix r=2 because we have 2 types of documents (CS and Medical documents), i.e. 2 concepts.

SVD example



$$\begin{array}{c|ccccc} & \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \\ \hline & 1 & 1 & 1 & 0 & 0 \\ \text{CS} & 2 & 2 & 2 & 0 & 0 \\ & 1 & 1 & 1 & 0 & 0 \\ & 5 & 5 & 5 & 0 & 0 \\ & 0 & 0 & 0 & 2 & 2 \\ \text{MD} & 0 & 0 & 0 & 3 & 3 \\ & 0 & 0 & 0 & 1 & 1 \end{array} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

doc-to-concept similarity matrix

concepts strengths

term-to-concept similarity matrix

U: document-to-concept similarity matrix

V: term-to-concept similarity matrix.

Example: $U_{1,1}$ is the weight of CS concept in document d_1, σ_1 is the strength of the CS concept, $V_{1,1}$ is the weight of ‘data’ in the CS concept.

$V_{1,2}=0$ means ‘data’ has zero similarity with the 2nd concept (Medical).

What does $U_{4,1}$ means?

PCA and SVD relation

Theorem: Let X = U Σ V^T be the SVD of an $N \times d$ matrix X and $C = \frac{1}{N-1} X^T X$ be the $d \times d$ covariance matrix. The eigenvectors of C are the same as the right singular vectors of X.

Proof:

$$X^T X = V \Sigma U^T U \Sigma VT = V \Sigma \Sigma V^T \overset{=}{} V \Sigma^2 VT$$

$$C = V \frac{\Sigma^2}{N-1} V^T$$

But C is symmetric, hence C = **V Λ $\mathbf{V^T}$** (according to theorem1).

Therefore, the eigenvectors of the covariance matrix are the same as matrix V (right singular vectors) and the eigenvalues of C can be computed from the singular values $\lambda_i = \frac{\sigma_i{}^2}{N-1}$

Summary for PCA and SVD

Objective: project an $N \times d$ data matrix X using the largest m principal components $V = [v_1, \dots v_m]$.

1. zero mean the columns of X.
2. Apply PCA or SVD to find the principle components of X.

PCA:

I. Calculate the covariance matrix $\mathrm{C} = \frac{1}{N-1} X^T X$.

II. V corresponds to the eigenvectors of C.

SVD:

I. Calculate the SVD of $X = U \Sigma V^T$.

II. V corresponds to the right singular vectors.

3. Project the data in an *m* dimensional space: Y = X V

Outline

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Multi-Dimensional Scaling (MDS)
- Non-linear extensions:
  - Kernel PCA
  - Isomap

MDS

- Multi-Dimensional Scaling [Cox and Cox, 1994] .
- MDS give points in a low dimensional space such that the Euclidean distances between them best approximate the original distance matrix.

  Given distance matrix

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix}.$$

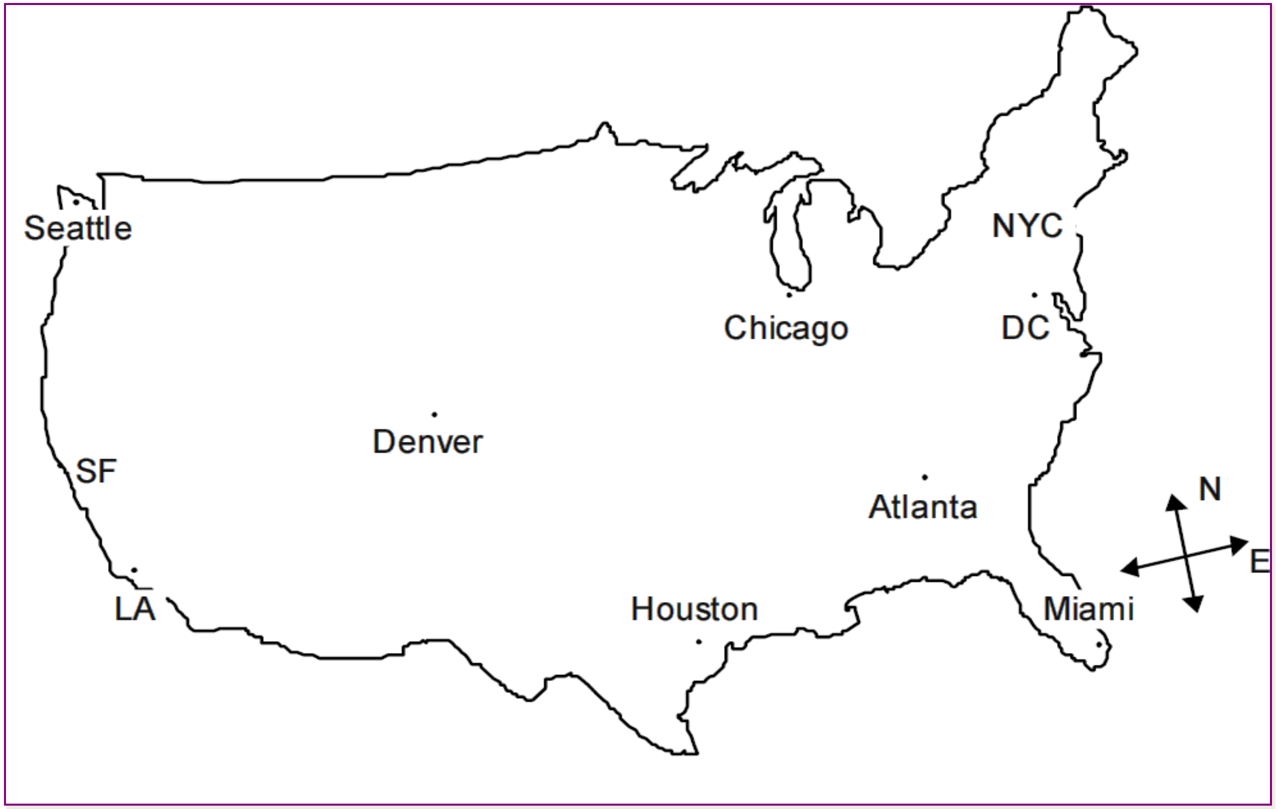
  Map input points x_i to z_i such as $||z_i - z_i|| \approx \delta_{i,j}$

- Classical MDS: the norm || . || is the Euclidean distance.
- Distances → inner products (Gram matrix) → embedding

  There is a formula to obtain Gram matrix G from distance matrix Δ.

MDS example

Given pairwise distances between different cities (Δ matrix), plot the cities on a 2D plane (recover location)!!



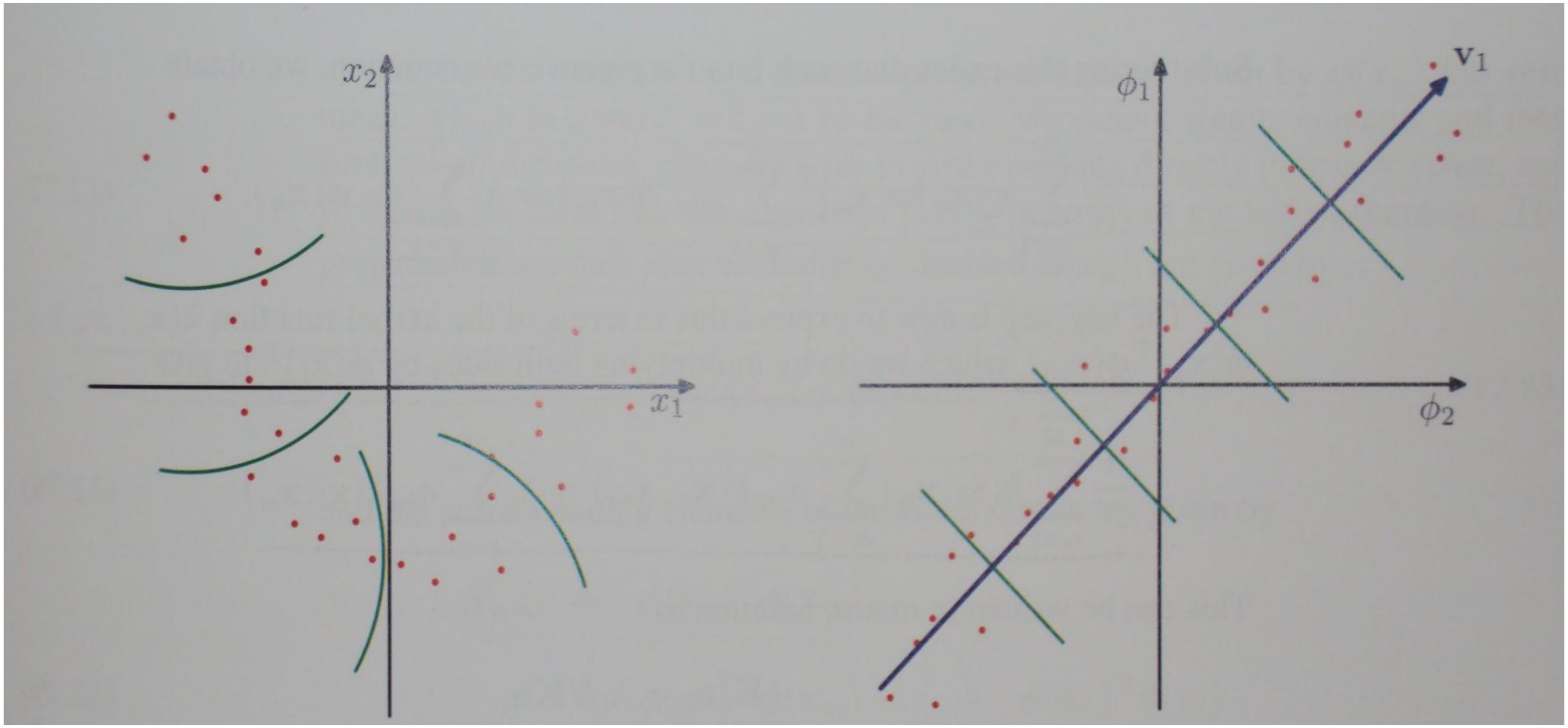
PCA and MDS relation

- Preserve Euclidean distances = retaining the maximum variance.
- *Classical MDS is equivalent to PCA when the distances in the input space are the Euclidean distance.*
- PCA uses the $d \times d$ covariance matrix: $\mathrm{C} = \frac{1}{N-1} X^T X$
- MDS uses the $N \times N$ Gram (inner product) matrix: $G = X\, X^T$
- If we have only a distance matrix (we don't know the points in the original space), we cannot perform PCA!
- Both PCA and MDS are invariant to space rotation!

Kernel PCA

- Kernel PCA [Scholkopf et al. 1998] performs nonlinear projection.
- Given input $(x_1, \ldots\ x_N)$, kernel PCA computes the principal components in the feature space $(\varphi(x_1), \ldots\ \varphi(x_N))$.
- Avoid explicitly constructing the covariance matrix in feature space.
- The kernel trick: formulate the problem in terms of the kernel function $k(x, x') = \varphi(x).\varphi(x')$ without explicitly doing the mapping.
- Kernel PCA is non-linear version of MDS use Gram matrix in the feature space (a.k.a Kernel matrix) instead of Gram matrix in the input space.

Kernel PCA



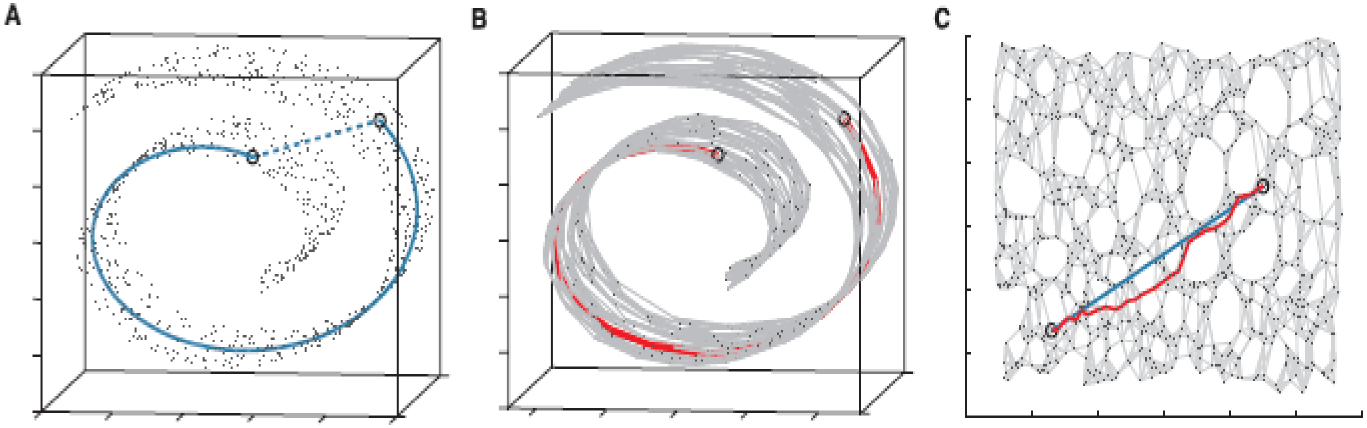
Original space

A non-linear feature space

Isomap

- Isomap [Tenenbaum et al. 2000] tries to preserve the distances along the data Manifold (Geodesic distance).

- Cannot compute Geodesic distances without knowing the Manifold!

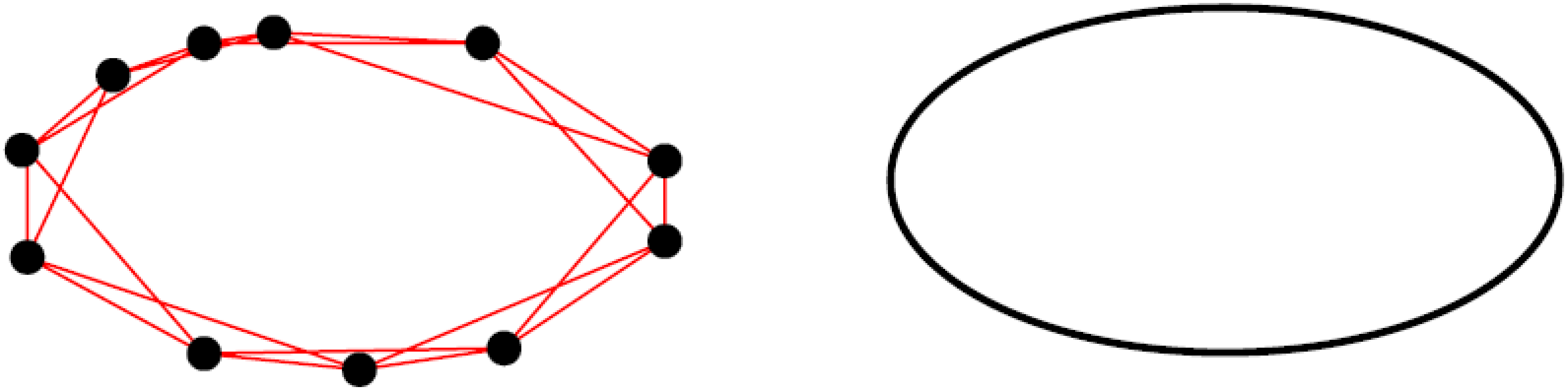


Blue: true manifold distance, red: approximated shortest path distance

- Approximate the Geodesic distance by the shortest path in the adjacency graph

Isomap

- Construct the neighborhood graph (connect only k-nearest neighbors): the edge weight is the Euclidean distance.

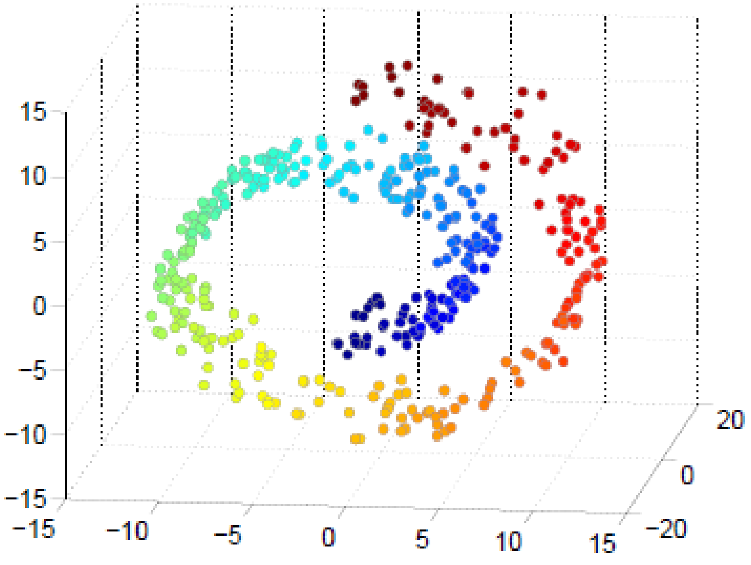


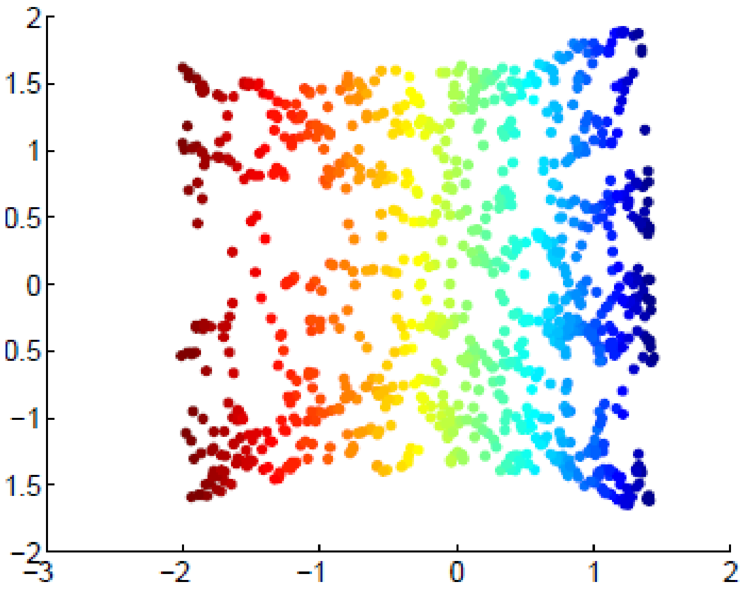
- Estimate the pairwise Geodesic distances by the shortest path (use Dijkstra algorithm).

- Feed the distance matrix to MDS.

Isomap

- Euclidean distances between outputs match the geodesic distances between inputs on the Manifold from which they are sampled.





Related Feature Extraction Techniques

Linear projections:

- Probabilistic PCA [Tipping and Bishop 1999]
- Independent Component Analysis (ICA) [Comon , 1994]
- Random Projections

Nonlinear projection (manifold learning):

- Locally Linear Embedding (LLE) [Roweis and Saul, 2000]
- Laplacian Eigenmaps [Belkin and Niyogi, 2003]
- Hessian Eigenmaps [Donoho and Grimes, 2003]
- Maximum Variance Unfolding [Weinberger and Saul, 2005]