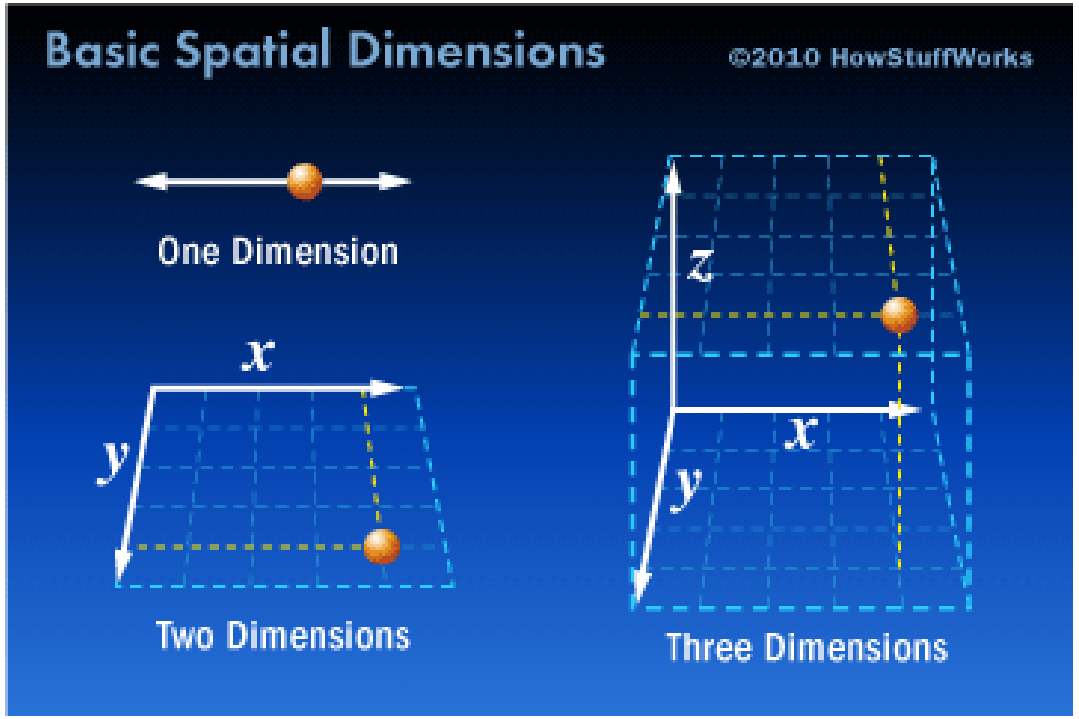
Dimension Reduction

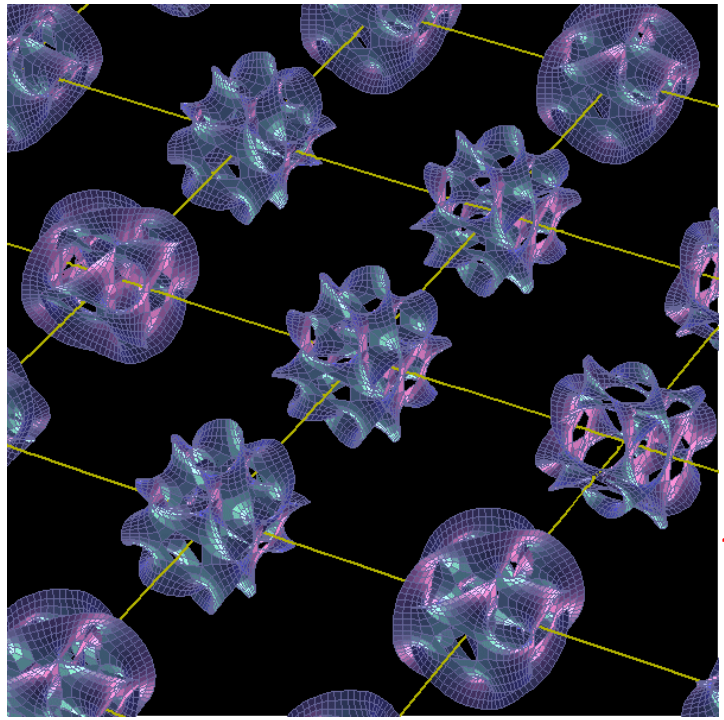
Yanyan Lan

lanyanyan@ict.ac.cn

About Dimension



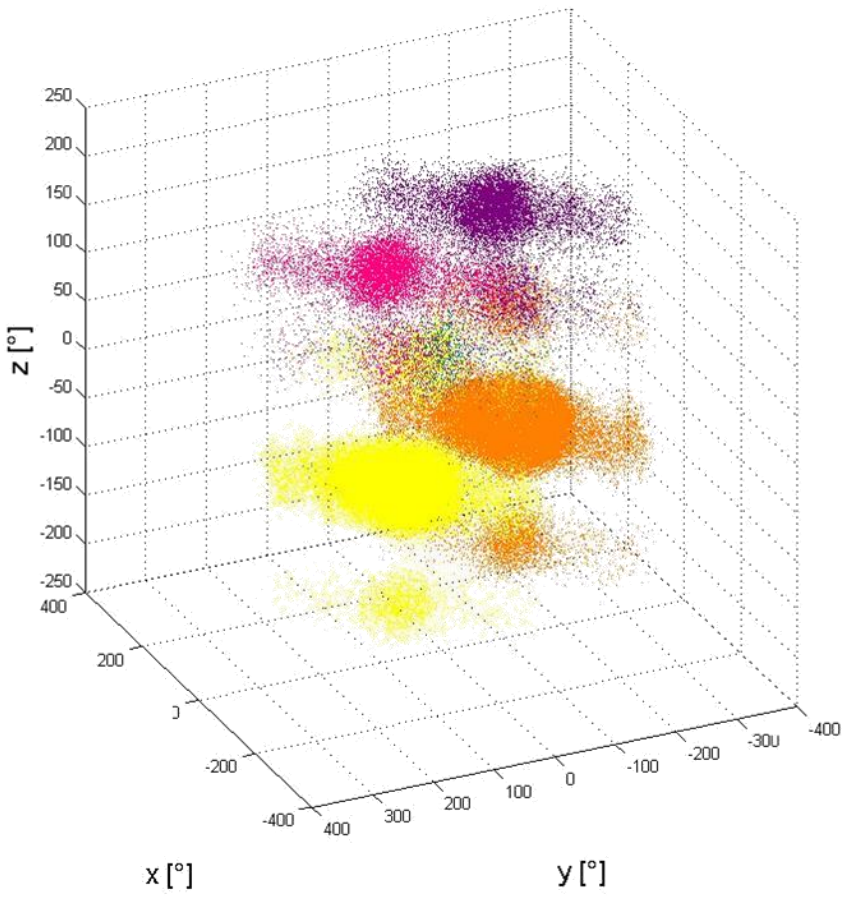
Spatial Dimensions



String Theory ?

Data Space

The dimensionality of data space could vary largely

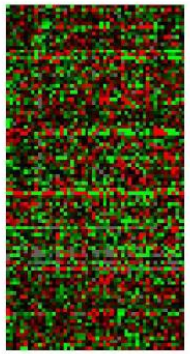


face images

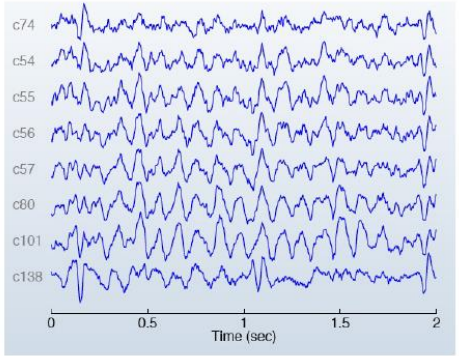
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



gene expression data

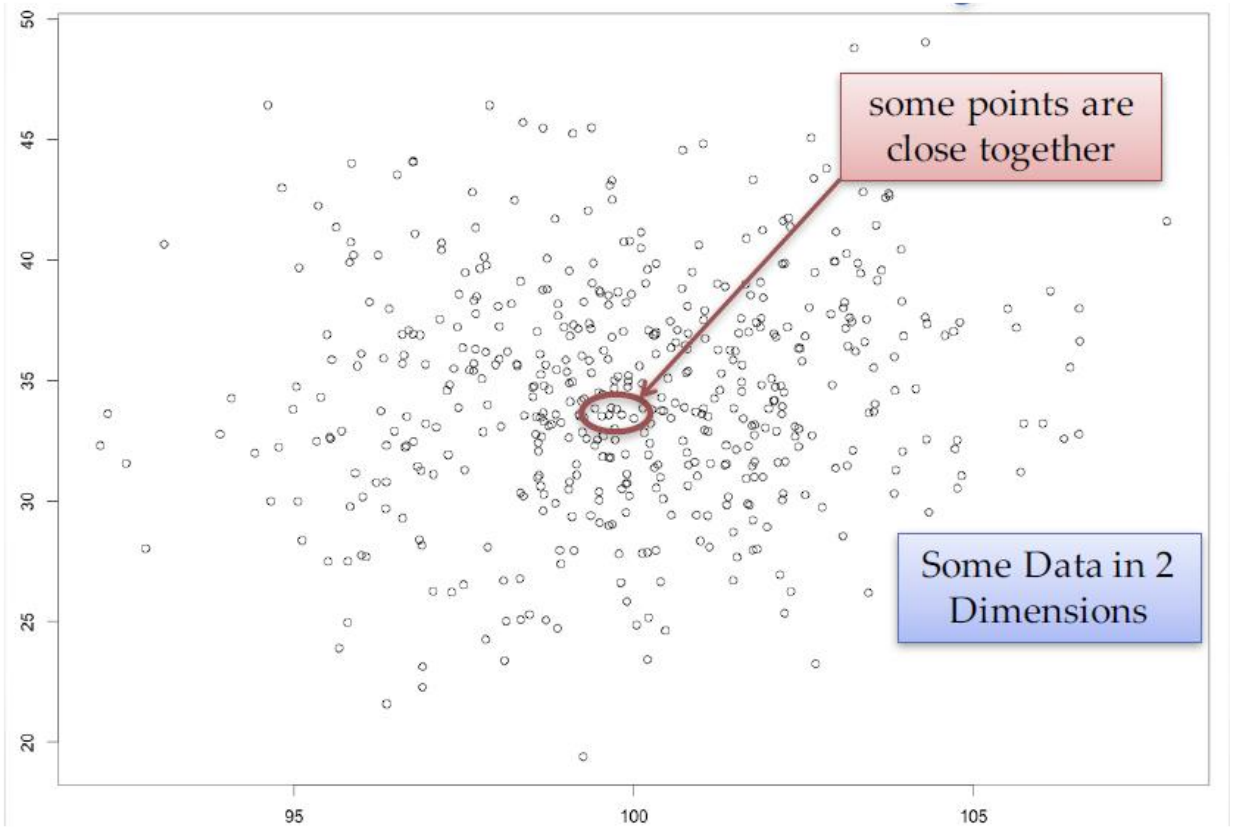


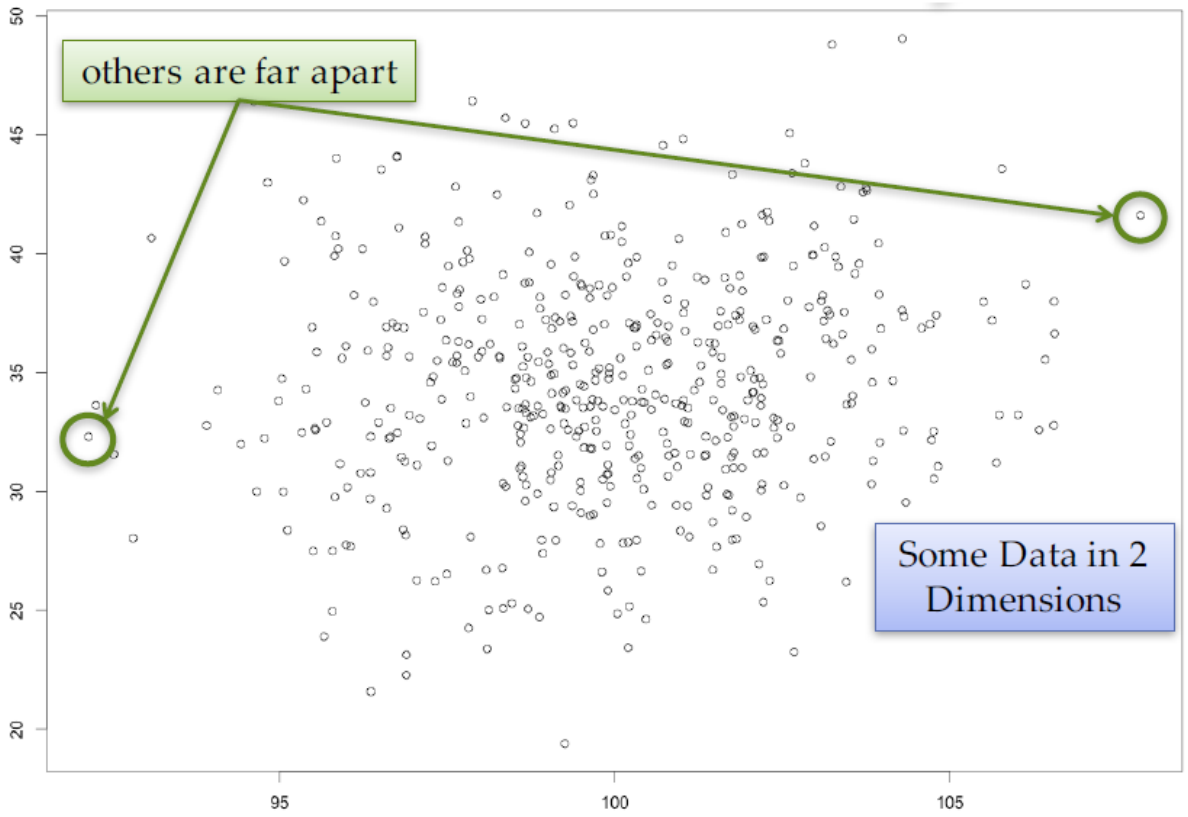
MEG readings

Lots of high-dimensional data...

The Curse of Dimensionality

As the dimensionality (i.e. number of variables) of a space grows, data points become so spread out that the ideas of distance and density become murky.

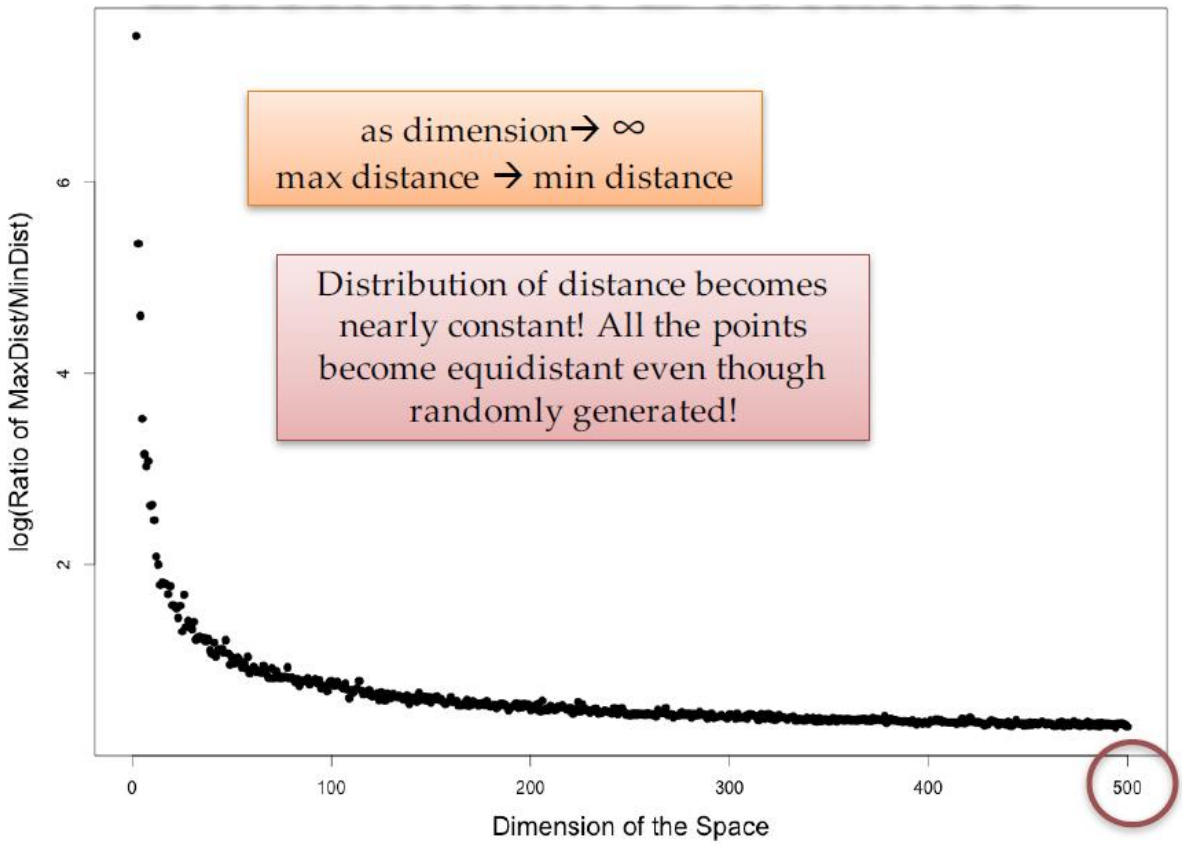




max/min = 30/0.02 = 1500. The max distance is 1500 times larger than the min distance.

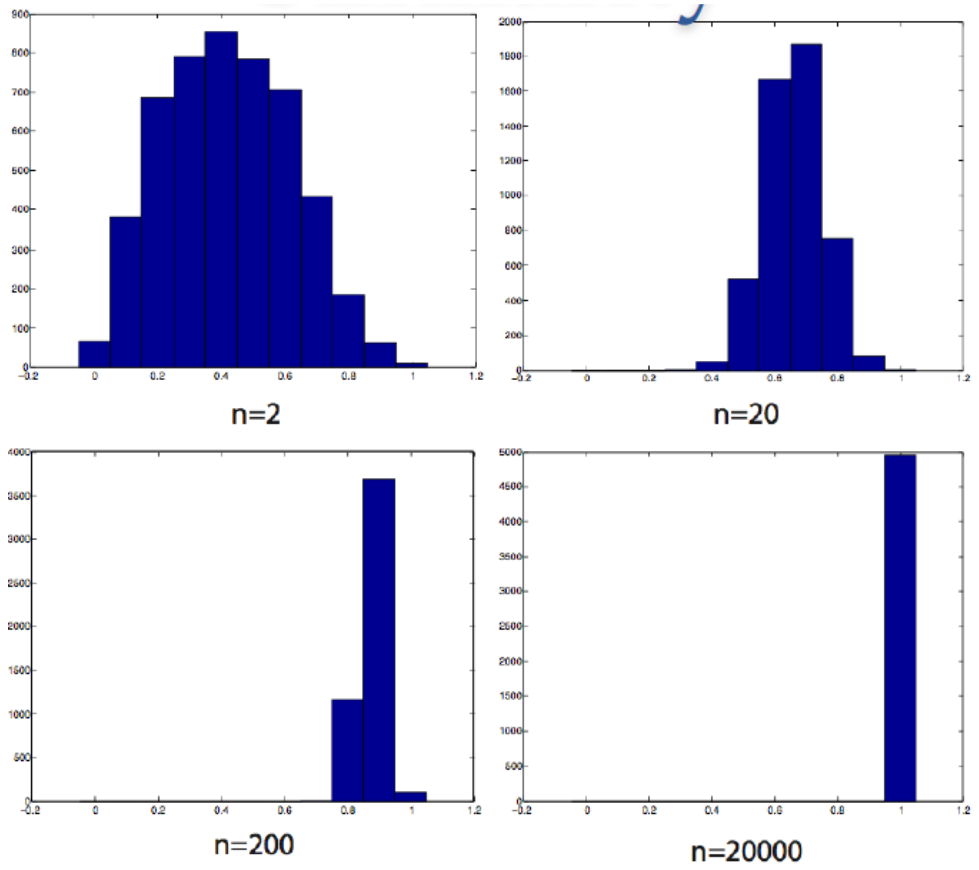
The Curse of Dimensionality

- Now lets generate those 500 points in 3-space, 4-space, … , 50-space.
- We'll compute that same metric, the ratio of the maximum distance to the minimum distance
- See how it changes as the number of dimensions grows…



The Curse of Dimensionality

- No distance/similarity metric is immune to the vastness of high dimensional space.
- One more. Let's look at the distribution (or lack thereof) of cosine similarity.
- Compute the cosine similarity between each pair of points, and divide that similarity by the maximum.



When is this a problem?

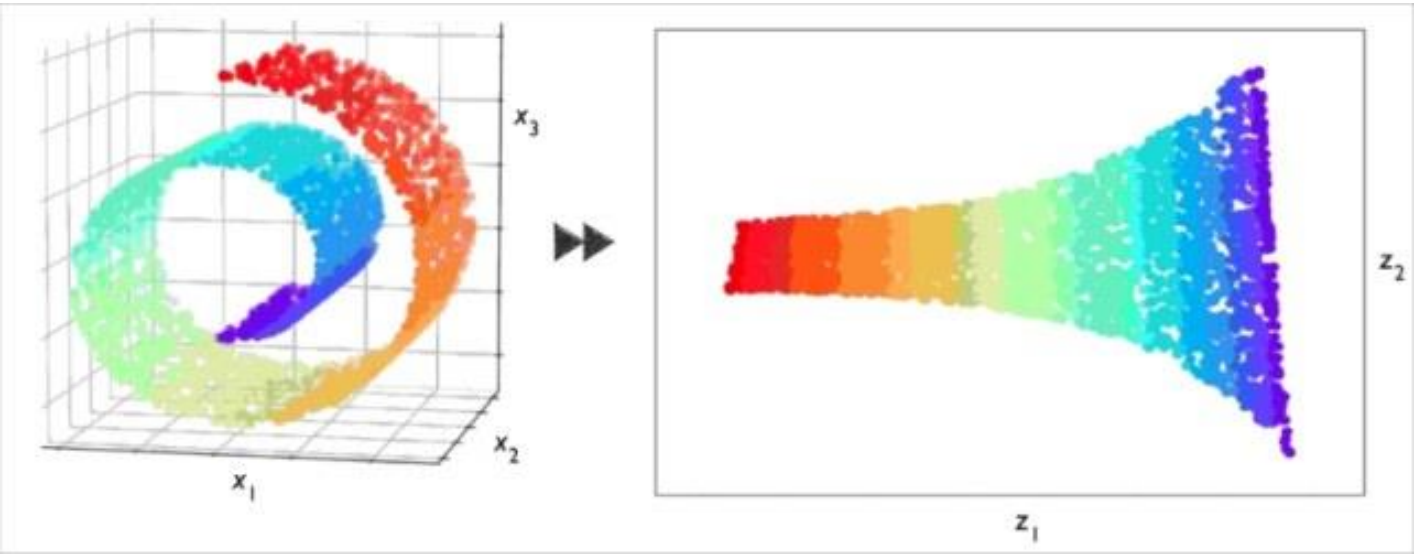
- Primarily when using algorithms which rely on distance or similarity
  - Particularly for clustering and k nearest neighbor methods
- Secondarily on all models due to collinearity and a desire for model simplicity.
- Computational/storage complexity can be problematic in all algorithms.

What can we do about it?

Dimension Reduction / Embedding

Dimension Reduction

- Dimension reduction/Embedding refers to the mapping of the original high-dim data into a lower-dim space.
- Essential idea: highly redundant data are likely to be compressible, i.e., the intrinsic dimension may be small, or the dimension related to the learning task may be small



Dimension Reduction Overview

- Consider dataset $\mathbf{X}$ consisting of n points in a d-dimensional (feature) space
- Data point x in $\mathbf{X}$ is a vector in $\mathbb{R}^d$
- Data can be seen as an $d \times m$ matrix

$$\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{m1} \\ \vdots & \vdots & \vdots \\ x_{1\mathrm{d}} & \cdots & x_{md} \end{pmatrix}$$

- Dimension Reduction methods:
  - map each vector x in $\mathbb{R}^d$ to a vector z in $\mathbb{R}^{d'}$
- Mapping: $f: \mathbb{R}^d \to \mathbb{R}^{d'}$
- For the idea to be useful we want: $d' \ll d$

Basic Criterion for dimension reduction

- Minimize the information loss:
  - Given D, find $\{z_i\}_{i=1}^m$ such that

$$||z_i - z_j|| = D_{ij}, 1 \leq i, j \leq m$$

MDS

where $D \in R^{m \times m}$ denotes the distance matrix in original space

  - Given $x_i, 1 \leq i \leq m$, find $f: \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, such that:

$$min \sum_{i=1} ||f^{-1}(z_i) - x_i||$$

- Maximize the class discrimination
  - maximize covariance of $\{z_i\}_{i=1}^m$:

$$max \sum_{i=1}^{m} cov(z_i, z_i)$$

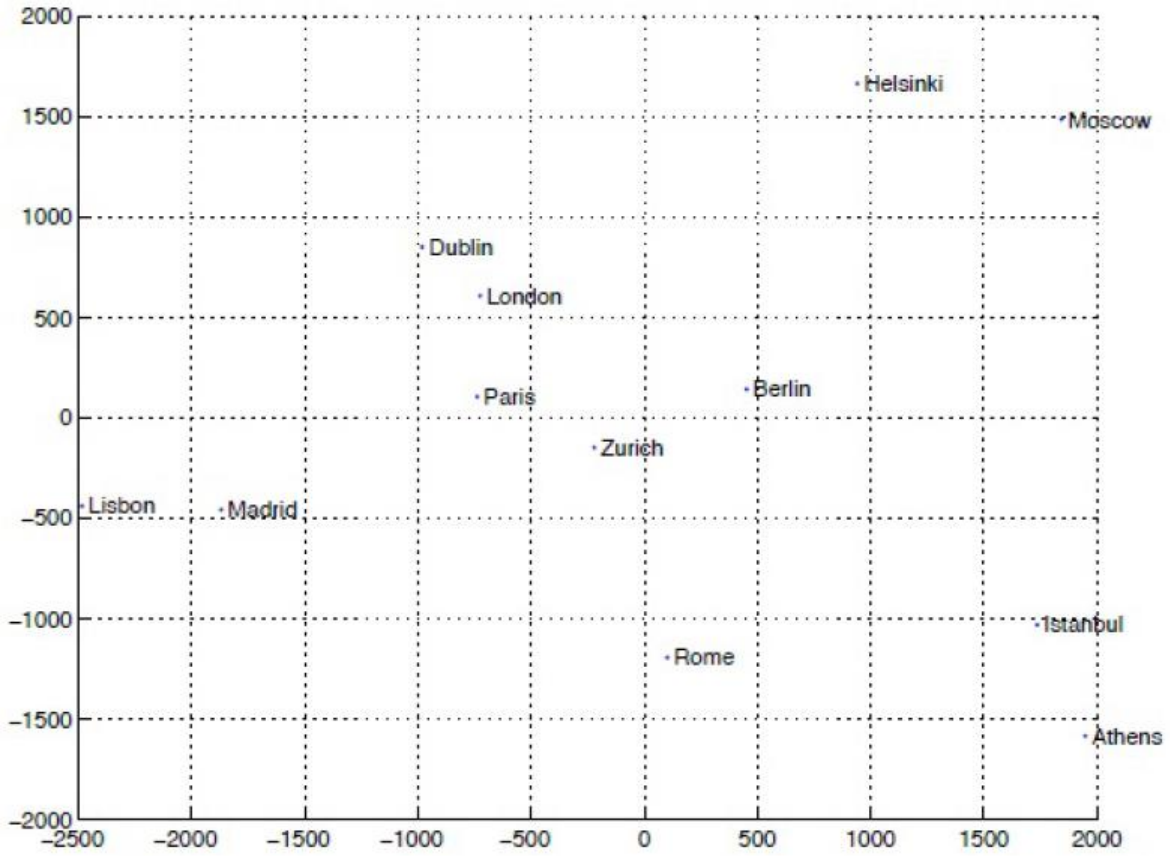
PCA

Multiple Dimensional Scaling (MDS)

Multiple Dimensional Scaling (MDS)

- Problem formulation:
  - Given the pairwise distances between pairs of points in some space (but the exact coordinates of the points and their dimensionality are unknown).
  - We want to embed the points in a lower-dimensional space such that the pairwise distances in this space are as close as possible to those in the original space.
- The projection to the lower-dimensional space is not unique because the pairwise distances are invariant to such operations as translation, rotation and reflection.

Embedding of Cities



x-axis: North/South; y-axis: East/West

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

Original Distance Matrix

Formulation of MDS

- Basic idea: the Euclidean distance in the d’-space equals to the distance in the original d-space, i.e.,

$$\|z_i - z_j\| = dist_{ij}, dist_{ij} = D_{ij}$$

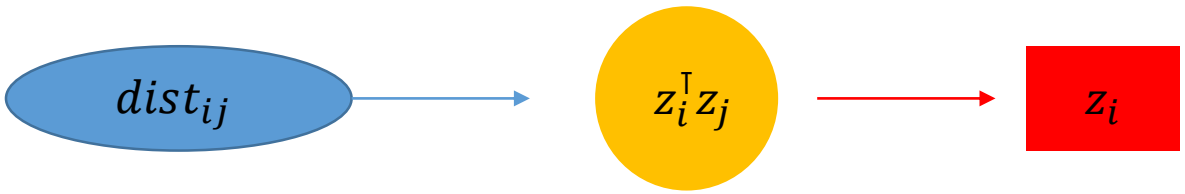
- How to compute z_i?

Formulation of MDS

- Basic idea: the Euclidean distance in the d'-space equals to the distance in the original d-space, i.e.,

$$\|z_i - z_j\| = dist_{ij}, dist_{ij} = D_{ij}$$

- How to compute z_i?



Deviation for B

Let $B = Z^\intercal Z \in \mathbb{R}^{m\times m}$, $b_{ij} = z_i^\intercal z_j$

$$dist_{ij}^2 = \|z_i\|^2 + \left\|z_j\right\|^2 \text{-} 2z_i^\intercal z_j = b_{ii} + b_{jj} - 2b_{ij}$$

Suppose *Z* is normalized (zero centered), i.e., $\sum_{i=1}^{m} z_i = 0$, Represent *B* by *D*

$$b_{ij} = -\frac{1}{2}\left(dist_{ij}{}^2 - b_{ii} - b_{jj}\right)$$

$$= -\frac{1}{2}\left(dist_{ij}{}^2 - \frac{1}{m}\left(\sum_{j=1}^{m} dist_{ij}{}^2 - tr(B)\right) - \frac{1}{m}\left(\sum_{i=1}^{m} dist_{ij}{}^2 - tr(B)\right)\right)$$

$$= -\frac{1}{2}\left(dist_{ij}{}^2 - \frac{1}{m}\sum_{j=1}^{m} dist_{ij}{}^2 - \frac{1}{m}\sum_{i=1}^{m} dist_{ij}{}^2 + \frac{2}{m}tr(B)\right)$$

$$= -\frac{1}{2}\left(dist_{ij}{}^2 - \frac{1}{m}\sum_{j=1}^{m} dist_{ij}{}^2 - \frac{1}{m}\sum_{i=1}^{m} dist_{ij}{}^2 + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}{}^2\right)$$

$$= -\frac{1}{2}\left(dist_{ij}{}^2 - dist_{i\cdot}{}^2 - dist_{j\cdot}{}^2 + dist_{\cdot\cdot}{}^2\right)$$

Deviation for Z

Do eigenvalue decomposing for B

$$\mathbf{B} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$$

$$\boldsymbol{\Sigma} = diag(\lambda_1, \lambda_2 \dots \dots \lambda_d)$$

$$\boldsymbol{\Sigma}_* = diag(\lambda_1, \lambda_2 \dots \dots \lambda_{d^*}), \lambda_1 > \lambda_2 > \cdots \dots \lambda_{d^*} > 0$$

$\mathbf{V}_*$ is the responding eigenvector matrix, then

$$\mathbf{Z} = \boldsymbol{\Sigma}_*^{1/2}\mathbf{V}_*^{\intercal} \in R^{d^* \times \mathrm{m}}$$

In practice, we can use $d' \ll d$ eigenvalue matrix $\widetilde{\boldsymbol{\Sigma}} = diag(\lambda_1, \lambda_2 \dots \dots \lambda_{d'})$

$$\mathbf{Z} = \widetilde{\boldsymbol{\Sigma}}^{1/2}\widetilde{\mathbf{V}}^{\intercal} \in R^{d' \times \mathrm{m}}$$

MDS Algorithm

Input: distance matrix $D \in R^{m \times m}$

the dimension of low dimensional space: d'

Process:

- $dist_{i\cdot}^{\ 2}, dist_{\cdot j}^{\ 2}, dist_{\cdot\cdot}^{\ 2}$
- Calculate $B = (b_{ij})$
- Do eigenvalue decomposition of *B*
- $\widetilde{\boldsymbol{\Sigma}}$ is the diagonal matrix with d' max eigenvalues, $\tilde{V}$ is the eigenvector matrix

Output: $\widetilde{\boldsymbol{\Sigma}}^{1/2}\widetilde{\mathbf{V}}^{\mathsf{T}} \in R^{d' \times m}$

MDS Examples: Tetrahedron

Pairwise distance matrix for tetrahedron (with distance 1)

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$D^2 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$dist_{i.}{}^2 = \frac{1}{m}\sum_{j=1}^{m} dist_{ij}{}^2 = (0.75, 0.75, 0.75, 0.75)$$

$$dist_{.j}{}^2 = \frac{1}{m}\sum_{j=1}^{m} dist_{ij}{}^2 = (0.75, 0.75, 0.75, 0.75)$$

$$dist_{..}{}^2 = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}{}^2 = 0.75$$

MDS Examples: Tetrahedron

$$B = \begin{pmatrix} 0.375 & -0.125 & -0.125 & -0.125 \\ -0.125 & 0.375 & -0.125 & -0.125 \\ -0.125 & -0.125 & 0.375 & -0.125 \\ -0.125 & -0.125 & -0.125 & 0.375 \end{pmatrix}$$

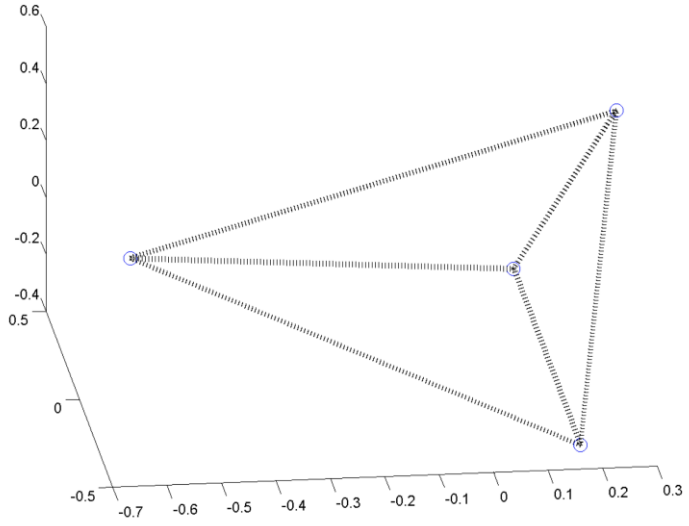
$$\mathbf{\Sigma} = diag(0.5, 0.5, 0.5, 5.55e^{-17}) \qquad \mathbf{V} = \begin{pmatrix} -0.5800 & 0.6426 & 0.0252 & -0.5 \\ 0.1599 & -0.2123 & -0.8242 & -0.5 \\ -0.3143 & -0.6892 & 0.4198 & -0.5 \\ 0.7343 & 0.2588 & 0.3791 & -0.5 \end{pmatrix}$$

$$\widetilde{\mathbf{\Sigma}} = diag(0.5, 0.5, 0.5) \qquad \widetilde{\mathbf{V}} = \begin{pmatrix} -0.5800 & 0.6426 & 0.0252 \\ 0.1599 & -0.2123 & -0.8242 \\ -0.3143 & -0.6892 & 0.4198 \\ 0.7343 & 0.2588 & 0.3791 \end{pmatrix}$$

MDS Examples: Tetrahedron

$$\mathbf{Z} = \widetilde{\mathbf{\Sigma}}^{1/2}\widetilde{\mathbf{V}}^{\intercal} = \begin{pmatrix} -0.4101 & 0.4544 & 0.0179 \\ 0.1131 & -0.1501 & -0.5828 \\ -0.2222 & -0.4873 & 0.2969 \\ 0.5193 & 0.1830 & 0.2681 \end{pmatrix}$$

Leading to the gram matrix $B_{4\times4}$ with eigenvalues (.5, .5, .5, 0). Using dimension p = 3, we have perfectly retrieved the tetrahedron.



Principal Components Analysis

Basic idea of linear dimensionality reduction

- Data points: $\mathbf{X} = (x_1, x_2 \ldots x_m) \in \mathbb{R}^{d \times m}$
- Data points in new space $\mathbf{Z} = (z_1, z_2 \ldots z_m) \in \mathbb{R}^{d' \times m}$
- Choose d' directions $w_1, w_2 \ldots w_{d'}$

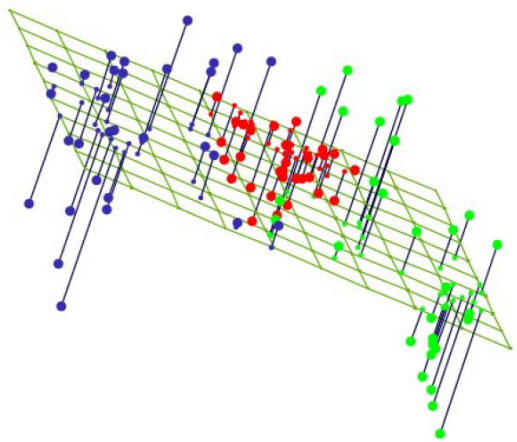
$$\mathbf{W} = \begin{pmatrix} | & & | \\ w_1 & \cdots & w_{d'} \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times d'}$$

- Project x down to : $z = \mathbf{W}^\intercal x$

How to choose $\mathbf{W}$?

PCA objective 1: reconstruction error

- Assume that $\|w_i\| = 1, {w_i}^{\mathsf{T}} w_j = 0 (i \neq j)$, new coordinates
- **W** serves two functions:
  - Encode: $z = \mathbf{W}^{\mathsf{T}} x, \quad z_j = {w_j}^{\mathsf{T}} x$
  - Decode: $\tilde{x} = \mathbf{W} z = \sum_{j=1}^{d'} z_j w_j$
- Want reconstruction error $\| x - \tilde{x} \|$ to be small
- Objective: minimize total squared reconstruction error



$$\min_{\mathbf{W} \in R^{d \times d'}} \sum_{i=1}^{m} \| x_i - \mathbf{W} z \|^2$$

PCA objective 1: Derivation

$$\sum_{i=1}^{m} \|x_i - \mathbf{W}z\|_2^2 = \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{d'} z_{ij} w_j \right\|_2^2$$

$$= \sum_{i=1}^{m} {z_i}^T z_i - 2 \sum_{i=1}^{m} {z_i}^T W^T x_i + const$$

$$\propto -tr\left(W^T \left(\sum_{i=1}^{m} x_i \, {x_i}^T\right) W\right) = -tr(W^T X X^T W)$$

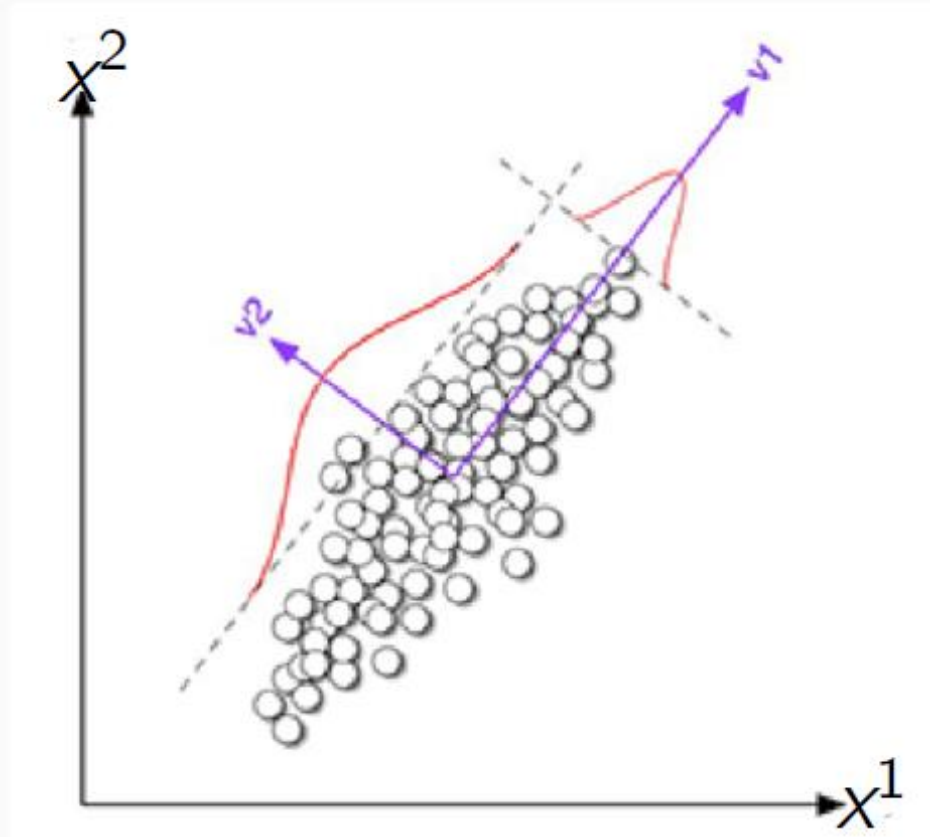
$$\max_W tr\left(\mathbf{W^T X X^T W}\right)$$
$$s.t. \mathbf{W^T W} = 1$$

PCA objective 2: Maximal variance

- idea: look for a direction that the data projected onto it has maximal variance

- Maximize sample variance of projection

$$max \sum_i z_i^\top z_i = \max_w tr(W^T X X^T W)$$



PCA

- Find $W \in R^{d \times d'}\ such\ that$:

$$\max_{W} tr\left(\mathbf{W^T X X^T W}\right)$$
$$s.t. \mathbf{W^T W} = 1$$

- Lagrange methods:
  - introduce Lagrange factor $\lambda_i, 1 \leq i \leq d$ and obtain the Lagrange function:

$$L = tr\left(\mathbf{W^T X X^T W}\right) - \sum_{i=1}^{d} \lambda_i (w_i{}^T w_i - 1)$$

  - the property of matrix's trace $\frac{\partial tr(\mathbf{WAW^T})}{\partial \mathbf{W}} = 2\mathbf{WA}, \mathbf{A} = \mathbf{A}^T$
  - let $\frac{\partial L}{\partial \mathbf{W}^T} = 0, then\ \mathbf{X X^T W} = \lambda \mathbf{W}$
  - W is the eigenvectors of matrix $\mathbf{X X^T}$

Solving PCA

Input: $\mathbf{X} = \{x_1, x_2, \dots x_m\}$

the dimension of low dimensional space: d'

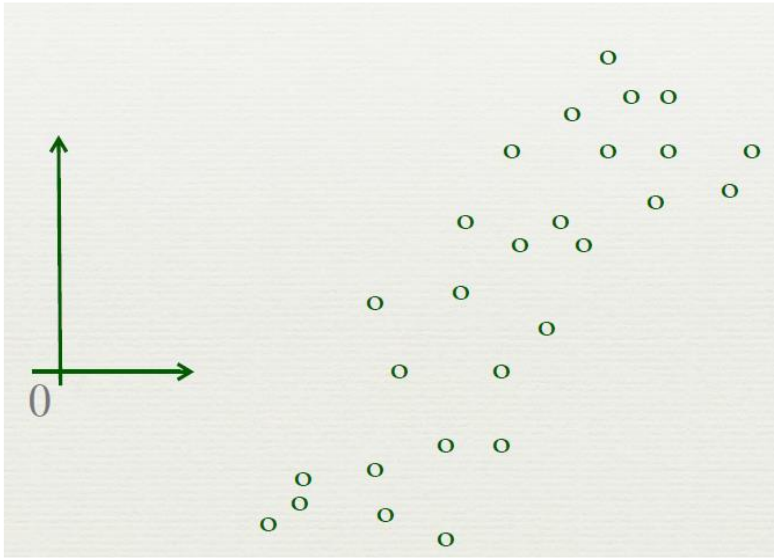
Process:

- $x_i = x_i - \frac{1}{m}\sum_{j=1}^{m} x_j$
- calculate $\mathbf{X}\mathbf{X}^{\mathbf{T}}$
- Do eigenvalue decomposition of $\mathbf{X}\mathbf{X}^{\mathbf{T}}$
- The eigenvectors of d'-max eigenvalue: $w_1, w_2, \dots w_{d'}$

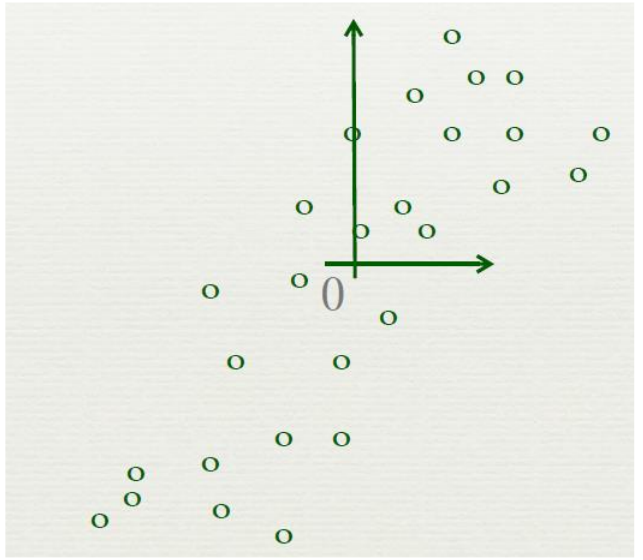
Output: $\mathbf{W} = (w_1, w_2, \dots w_{d'})$

Solving PCA

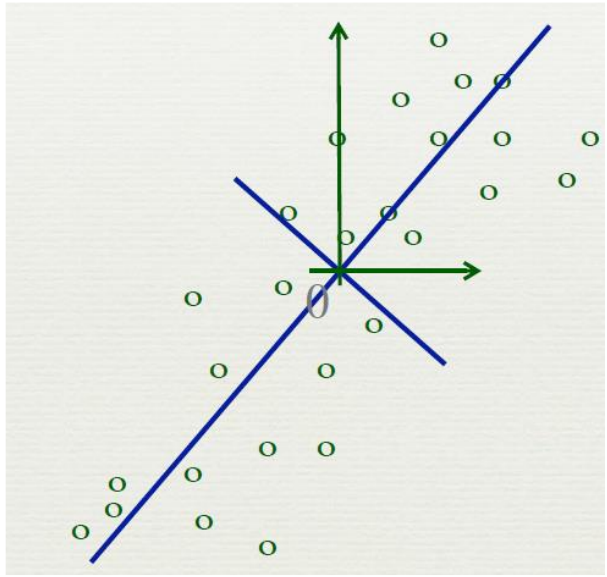
Original Data



Zero-centered data



Principle components



Very Nice When Initial Dimension Not Too Big

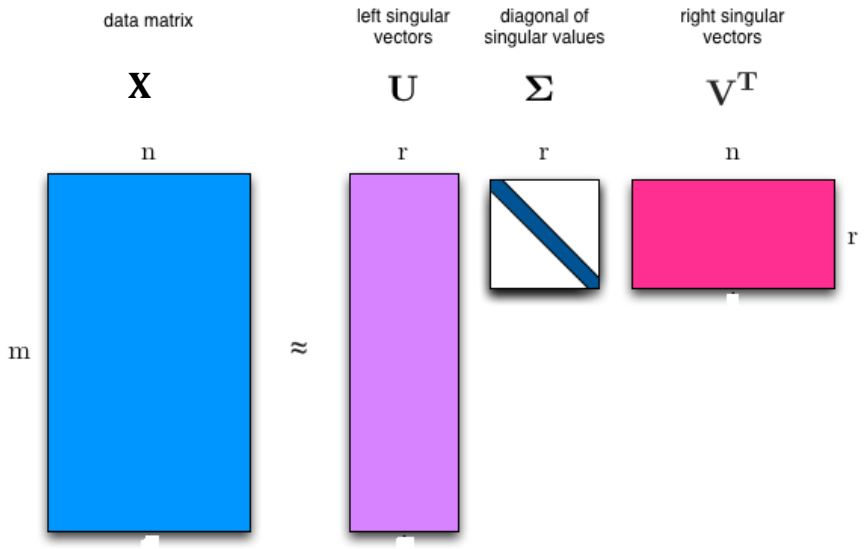
- What if very large dimensional data?
  - e.g., Images (d ≥10^4)
- Problem:
  - Covariance matrix $\boldsymbol{\Sigma}$ is size (d x d)
  - d=10^4 $\quad$ | $\boldsymbol{\Sigma}$ | = 10^8

- Singular Value Decomposition (SVD) to the rescue!
  - pretty efficient algorithms available, including Matlab SVD
  - some implementations find just top N eigenvectors

Recall: Singular Value Decomposition (SVD)

- We showed that the principal components are the singular values of $\mathbf{X}$
- In particular:
  - i-th principal component of $\mathbf{X}^{\boldsymbol{T}}\boldsymbol{X}$ is the i-th left singular vector of $\mathbf{X}$
  - the i-th eigenvalue of $\mathbf{X}^{\boldsymbol{T}}\boldsymbol{X}$ is exactly the i-th singular value squared (σ_i^2)

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}}$$



Solving PCA （2）

Input: $D = \{x_1, x_2, \dots x_m\}$

the dimension of low dimensional space: d'

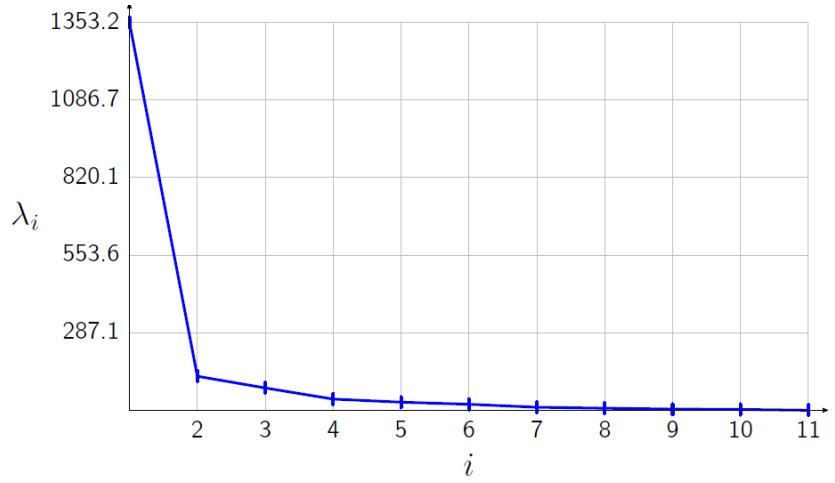
Process:

- $x_i = x_i - \frac{1}{m}\sum_{j=1}^{m} x_j$
- Do SVD over $X \approx U_{d\times d}\Sigma_{d\times m}{V_{m\times m}}^{\mathsf{T}}$
- The left singular vectors of d'-max singular value: $u_1, u_2, \dots u_{d'}$

Output: $U = (u_1, u_2, \dots u_{d'})$

Parameter d’

- User assign
- Select using simple classifier (e.g. kNN) with cross-validation
- Or use the reconstruction threshold: $\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{j=1}^{d} \lambda_j} \geq t$ (e.g. 85%)



Eigenvalues typically drop sharply

Application of PCA (1): Face Recognition

- Eigenfaces: a set of eigenvectors as basis features for face images
- Computing Eigenfaces:
    1. A set of m face images, each being represented as a D-dimensional vector $x_i, S = \{x_1, x_2 \dots x_m\}$

PCA for Face Recognition: Eigenface

- Computing Eigenfaces:
  2. Compute the mean face image: $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$

  3. Centralize the face images by subtracting the mean face $\hat{x_i} = x_i - \overline{x_i}$, $\hat{S} = \{\hat{x}_1, \hat{x}_2 \ldots \hat{x}_m\}$

PCA for Face Recognition: Eigenface

- Computing Eigenfaces:
  4. Compute the eigenvectors and eigenvalues of the covariance matrix $C = \hat{S}\hat{S}^T$ The eigenvectors are therefore called Eigenfaces. They are the directions in which the images differ from the mean face.

PCA for Face Recognition: Eigenface

- Computing Eigenfaces:
  5. choose the principal components E.g. choose d' principal components according to

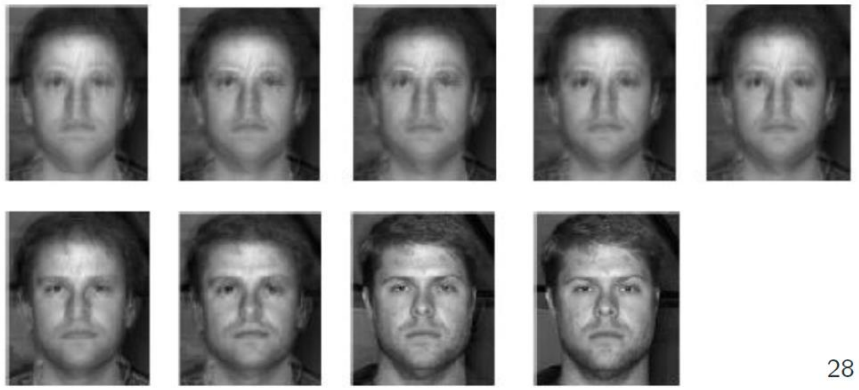
$$\frac{\lambda_1+\lambda_2+\cdots+\lambda'_d}{\lambda_1+\lambda_2+\cdots+\lambda_d} \geq \epsilon$$

  6. the Eigenfaces can be used to represent both existing and new faces. A new centralized face image can be projected on the Eigenfaces.

PCA for Face Recognition: Eigenface

$$\hat{x} = \bar{x} + z_1\vec{w}_1 + z_2\vec{w}_2 + z_3\vec{w}_3 + \cdots$$

Reconstruction Procedure



Application of PCA: Topic Analysis

- LSI (latent semantic indexing) aims to discover something about the meaning behind the words; about the topics in the documents.

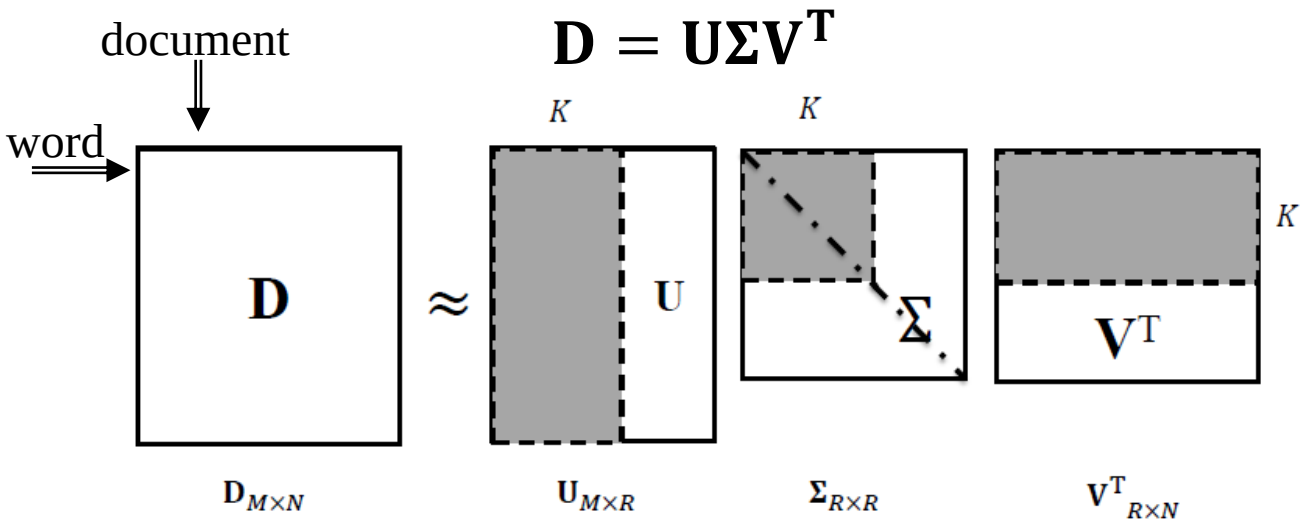
Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

LSI

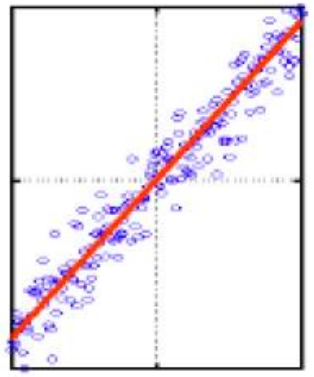


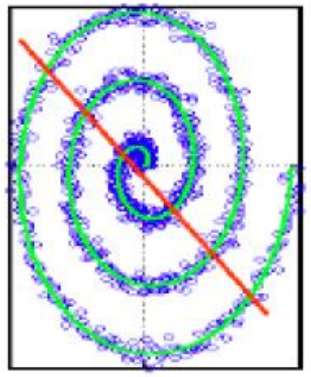
- rank($\mathbf{U\Sigma_k V^T}$) = k ≤rank($\mathbf{D}$)
- K is the latent topic number
- $\mathbf{u_i}$ denotes the k-th topic
- $\mathbf{v_i}^T$denotes the i-th document represented by the k topics.

Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	Topic10
OPEC	Africa	contra	school	Noriega	firefight	plane	Saturday	Iran	senate
oil	South	Sandinista	student	Panama	ACR	crash	coastal	Iranian	Reagan
cent	African	rebel	teacher	Panamanian	forest	flight	estimate	Iraq	billion
barrel	Angola	Nicaragua	education	Delval	park	air	western	hostage	budget
price	apartheid	Nicaraguan	college	canal	blaze	airline	Minsch	Iraqi	Trade

PCA Summary

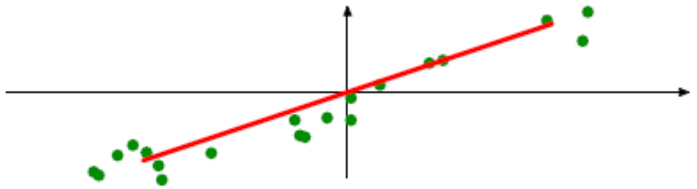
- **Strengths**:
  - Eigenvector method
  - No tuning parameters
  - Non-iterative
  - No local optima
- **Weaknesses**:
  - Limited to second order statistics
  - Limited to linear projections



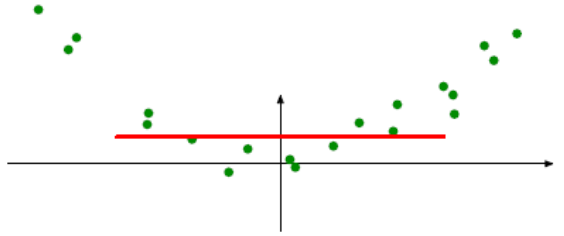


Non-linear Methods

Limitations of Linearity



PCA is effective

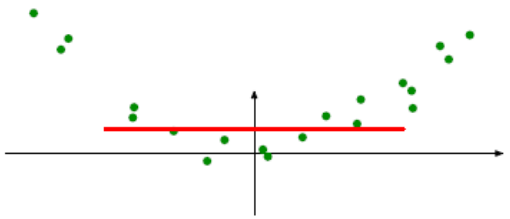


PCA is ineffective

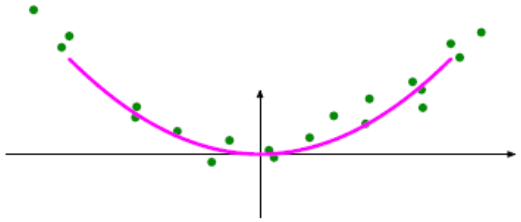
- Problem is that PCA subspace is linear:

$$S = \{x \in \mathbb{R}^d : x = \mathbf{W}z : z \in R^{d'}\}$$
$$= \{(x_1, x_2) : x_1 = w_1 z, x_2 = w_2 z\}$$
$$= \{(x_1, x_2) : x_2 = \frac{w_1}{w_2} x_1\}$$

Going beyond linearity: quick solution



Broken solution



Desired solution

- We want desired solution $S = \{(x_1, x_2): x_2 = \frac{u_2}{u_1} {x_1}^2\}$
- We can get this: $S = \{\Phi(x) = Wz\}\ with\ \Phi(X) = (x_1^2, x_2)^T$

linear dimensionality reduction in $\Phi(x)$ space

Nonlinear dimensionality reduction in x space

- In general, can set $\Phi(x) = (x_1, x_1^2, x_1, x_2, \sin(x_1) \ldots)^T$
- Problems: (1) ad-hoc and tedious

  (2) $\Phi(x)$ large, computationally expensive

Kernel PCA

- Kernel function: $k(x_1, x_2)$ such that

  K, the kernel matrix formed by $K_{ij} = k(x_i, x_j)$ is positive semi-definite

- Examples:
  - Linear kernel: $k(x_1, x_2) = x_1^{\top} x_2$
  - Polynomial kernel: $k(x_1, x_2) = (1 + x_1^{\top} x_2)^2$
  - Gaussian(RBF) kernel: $k(x_1, x_2) = e^{-||x_1 - x_2||^2}$

  Treat data points x as black boxes, only access via K .

  K intuitively measures similarity between two inputs.

- Mercer's theorem (using kernels is sound)

  Exists high-dimensional feature space Φ such that $k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2)$

KPCA Algorithm

- KPCA
  - Calculate K, $K_{ij} = k(x_i, x_j)$
  - Get centralized K: $\overline{K}$
  - Do eigenvalue decomposition of $\overline{K}$
  - Select eigenvectors corresponding to the d' max eigenvalues
- For new sample x the j-th component of z^j
  - $z^j = {w_j}^T \Phi(x) = \sum_{i=1}^{m} \alpha_i^j \Phi(x_i)^T \Phi(x) = \sum_{i=1}^{m} {\alpha_i}^j k(x_i, x)$

Manifold Learning

Euclidean distance in high dimensional input space may not accurately reflect the intrinsic similarity

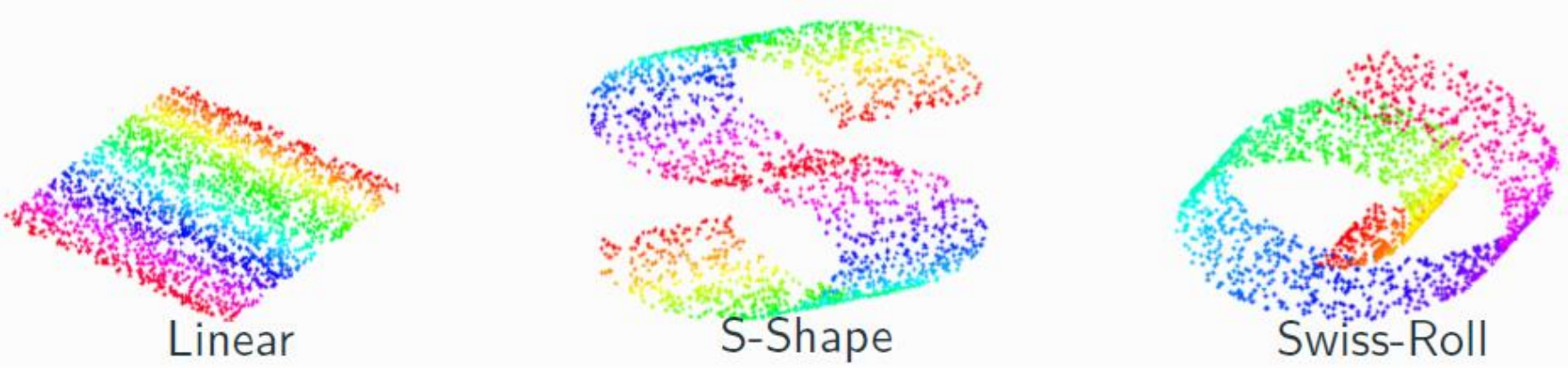
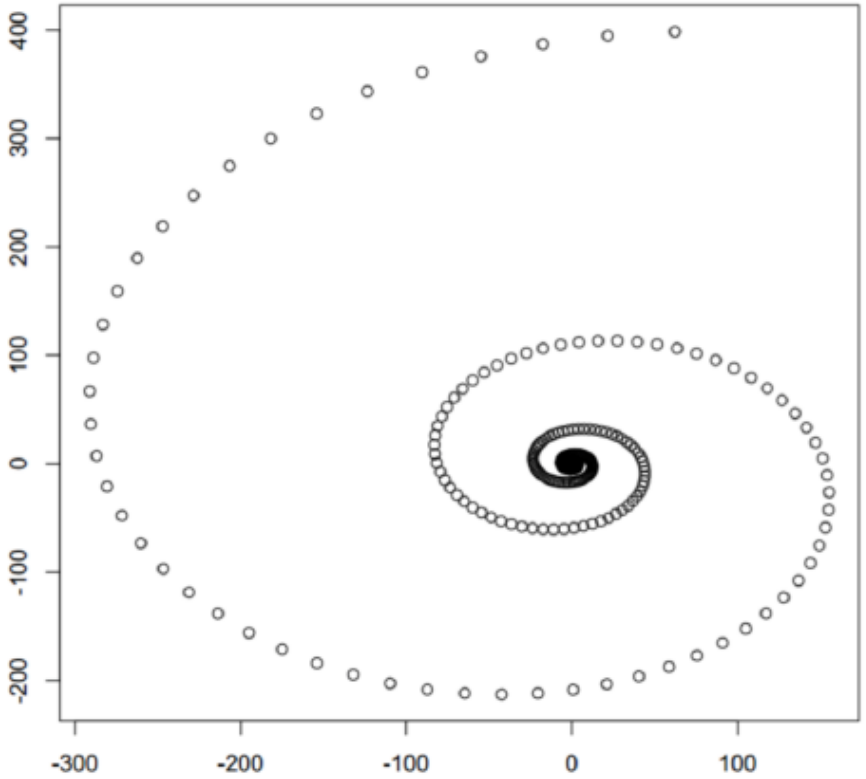


Figure 3: 2D data embedded into 3D space

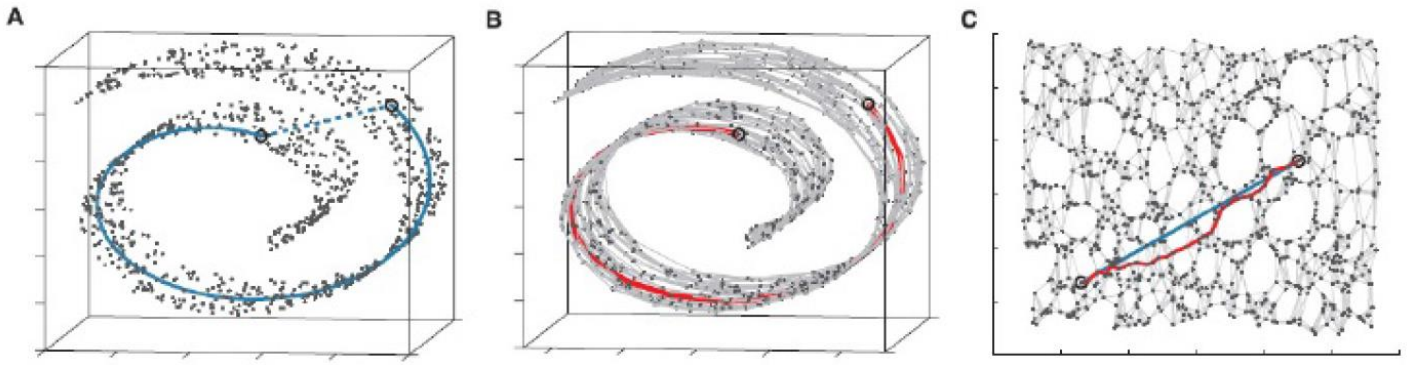
Manifold Learning

- Manifold is a topological space that is locally Euclidean.
- A manifold can arbitrarily well-approximated the subspace of a high space.



Geodesic Distance

- How to measure the distance in high-dimensional space?



- The geodesic ("shortest path") distance should be used instead.
- Geodesic distance:
  - Neighboring points: input space Euclidean distance provided a good approximation of the geodesic distance.
  - Faraway points: geodesic distance can be approximated by adding up a sequence of short hops between neighboring points based on Euclidean distance.

ISOMAP Algorithm

- ISOMAP is a nonlinear dimensionality reduction method that is based on metric MDS but seeks to preserve the intrinsic geometry of the data as captured in the geodesic distances between data points.

- Three steps of the Isomap algorithm:
  - Construct the neighborhood graph
  - Compute the shortest paths
  - Construct the low-dimensional embedding

ISOMAP Algorithm

Input: dataset $D = \{x_1, x_2, \dots, x_n\}$

number of the nearest neighbors: k

lower dimension: d'

procedure:

1. **for** $i = 1,2, \dots, m$ **do**
2. find k nearest neighbors $N = \{n_1, \dots, n_k\}$ of x_i
3. $\text{dist}(x_i, n_j) = \text{euclidean\_dist}(x_i, n_j), \forall n_j \in N$; $\text{dist}(x_i, n_j) = \infty, \forall n_j \notin N$
4. **end for**
5. use the shortest path algorithm to calculate the distance between any two points $\text{dist}(x_i, x_j)$
6. Input $\text{dist}(x_i, x_j)$ to MDS
7. **return** the output of MDS

output: the projection of D on low-dimensional space $Z = \{z_1, \dots, z_m\}$

ISOMAP Algorithm

- There are two bottlenecks in the lsomap algorithm

- Shortest path computation:
  - Floyd's algorithm: $O(N^3)$
  - Dijkstra's algorithm(with Fibonacci heaps): $O(KN^2 logN)$ where K is the neighborhood size
- Eigen decomposition: $O(N^3)$

Global vs. Local Embedding Methods

- Metric MDS and Isomap compute embeddings that seek to preserve inter-point straight-line (Euclidean) distances or geodesic distances between all pairs of points. Hence they are global methods.
- Both locally linear embedding (LLE) and Laplacian eigenmap try to recover the global nonlinear structure from local geometric properties. They are local methods.
- Overlapping local neighborhoods, collectively analyzed, can provide information about the global geometry.

Problem Setting of Locally Linear Embedding (LLE)

- Let $X = \{x^{(1)}, x^{(2)} \ldots . x^{(m)}\}$ be a set of m points in a high dimensional input space $\mathbb{R}^d$
- The N data points are assumed to lie on or near a nonlinear manifold of intrinsic dimensionality $d' < d (\text{typically } d' \ll d)$
- Provided that sufficient data are available by sampling well from the manifold, the goal of LLE is to find a low-dimensional embedding of X by mapping the *d*-dimensinal data into a single global coordinate system in $R^{d'}$
- Let us denote the set of N points in the embedding space $R^{d'}$ by $Z = \{z^{(1)}, z^{(2)} \ldots z^{(N)}\}$

LLE Algorithm

- For each data point $x^{(i)} \in X$:
  - Find the set $N_i \; of \; K$ nearest neighbors of $x^{(i)}$
  - Compute the reconstruction weights of the neighbors that minimize the error of reconstructing $x^{(i)}$
- Compute the low-dimensional embedding z that best preserves the local geometry represented by the reconstruction weights.

LLE Algorithm

First find the nearest neighbors coordinates Q_i for each x_i , and calculate the refactoring weights w_i:

$$\min_{w_1,\dots,w_n} \sum_{i=1}^{m} \left\| x_i - \sum_{j\in Q_i} w_{ij} x_j \right\|_2^2, \ s.t. \ \sum_{j\in Q_i} w_{ij} = 1 \tag{1}$$

Let $C_{jk} = (x_i - x_j)^T (x_i - x_k)$, w_{ij} has closed-form solution:

$$w_{ij} = \frac{\sum_{k\in Q_i} C_{jk}^{-1}}{\sum_{l,s\in Q_i} C_{ls}^{-1}} \tag{2}$$

Since w_i remain unchanged, we can get coordinate z_i in low-dimension of x_i:

$$\min_{z_1,\dots,z_n} \sum_{i=1}^{m} \left\| z_i - \sum_{j\in Q_i} w_{ij} z_j \right\|_2^2 \tag{3}$$

Let $Z = (z_1, \dots, z_m) \in \mathbb{R}^{d'\times m}$, $(W)_{ij} = w_{ij}$,

$$M = (I - W)^T (I - W) \tag{4}$$

So eq(3) can be rewritten and Z^T can be solved through SVD, Z consists of the eigenvector of M corresponding to the smallest d' eigenvalues:

$$\min_{Z} tr(ZMZ^T), \ s.t. ZZ^T = I \tag{5}$$

LLE Algorithm

Input: dataset $D = \{x_1, x_2, \dots, x_n\}$

number of the nearest neighbors: k

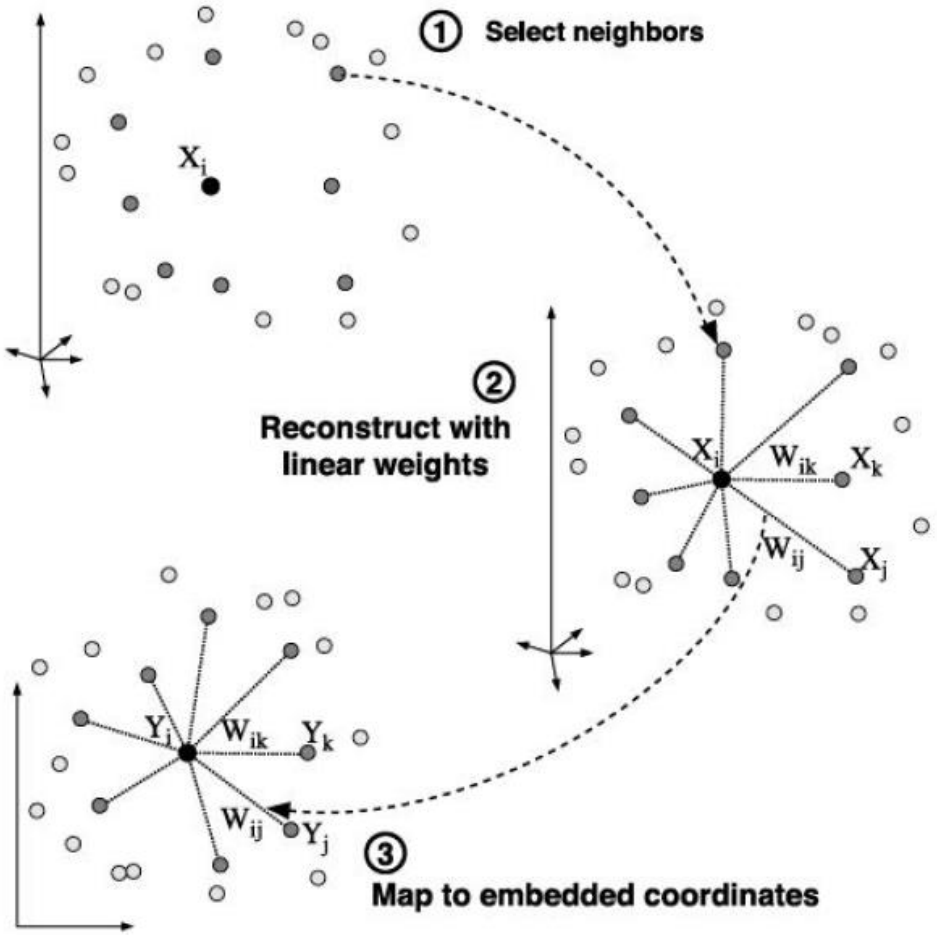
lower dimension: d'

procedure:

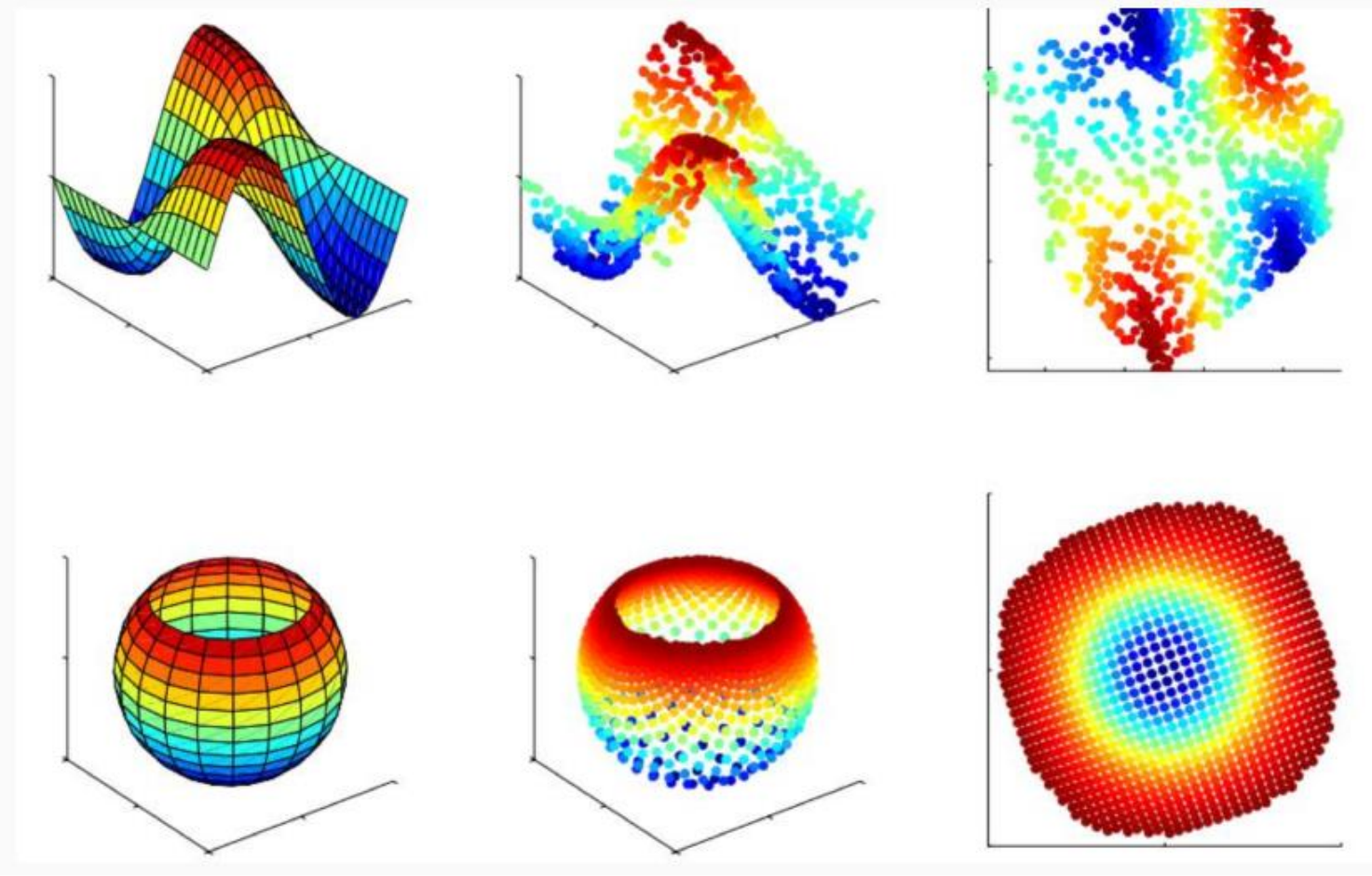
1. **for** $i = 1,2, \dots, m$ **do**
2. find k nearest neighbors $N = \{n_1, \dots, n_k\}$ of x_i
3. calculate $w_{ij},\ j \in Q_i$ from eq(1)
4. $w_{ij} = 0,\ \forall j \notin Q_i$
5. **end for**
6. calculate W from eq(4)
7. do SVD
8. **return** the eigenvector of M corresponding to the smallest d' eigenvalues

output: the projection of D on low-dimensional space $Z = \{z_1, \dots, z_m\}$

Algorithm Steps



LLE Example



[Roweis and Saul, Science, Vol 290, 2000; Saul and Roweis, JMLR 2003]

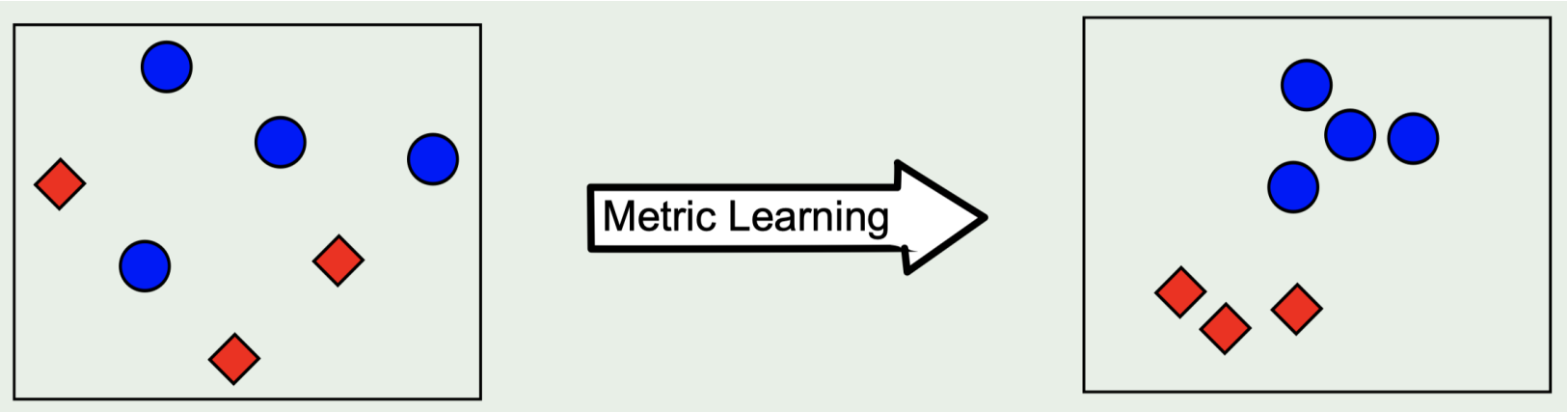
Metric Learning

Motivation

- One purpose of dimension reduction for high-dimensional data is to find a suitable low-dimensional space
- Learning in low-dimensional space is better than original space
- Each space corresponds to a metric distance defined on the sample attributes
- Finding the right space is essentially looking for a suitable distance metric
- Why not try to learn a suitable distance metric directly?

Metric Learning

- The notion of good metric is problem-dependent: Each problem has its own semantic notion of similarity, which is often badly captured by standard metrics (e.g., Euclidean distance)
- Solution: learn the metric from data
- Basic idea: learn a metric that assigns small (resp. large) distance to pairs of examples that are semantically similar (resp. dissimilar)



Metric Learning

- Distance Metric learning is to learn a distance metric for the input space of data
- Data is from a given collection of pair of similar/dissimilar points that preserves the distance relation among the training data
- Learned metric can significantly improve the performance of algorithms such as classification, clustering and many other tasks

Distance Metric

- Function *dist*() is a distance metric
  - $dist(x_i, x_j) \geq 0$(non-negativity)
  - $dist(x_i, x_j) = 0 \; iff \; x_i = x_j$ (identity of indiscernibles)
  - $dist(x_i, x_j) = dist(x_j, x_i)$(symmetry)
  - $dist(x_i, x_j) \leq dist(x_i, x_k)$+$dist(x_k, x_j)$(triangle inequality)
- Minkowski distance: $dist_{mk} = (\sum_{u=1}^{n} |x_{iu} - x_{ju}|^p)^{\frac{1}{p}}$
  - p=2 Euclidean distance: $dist_{mk} \overset{\text{def}}{=} dist_{ed}$
  - p=1 Manhattan distance: $dist_{mk} \overset{\text{def}}{=} dist_{man}$

Mahanalobis Distance Metric

- For two samples x_i, x_j, square of Euclidean distance is,

$$\mathrm{dist}_{ed}^2(x_i, x_j) = ||x_i - x_j||_2^2 = \mathrm{dist}_{ij,1}^2 + \mathrm{dist}_{ij,2}^2 + \cdots + \mathrm{dist}_{ij,d}^2$$

- Where $\mathrm{dist}_{ij,k}^2$ is distance between x_i and x_i in the k-th dimension
- Assume that the importance of different attributes is different, we can introduce attribute weights w:

$$\mathrm{dist}_{wed}^2(x_i, x_j) = ||x_i - x_j||_2^2 = w_1 \cdot \mathrm{dist}_{ij,1}^2 + w_2 \cdot \mathrm{dist}_{ij,2}^2 + \cdots + w_{\mathrm{n}} \cdot \mathrm{dist}_{ij,d}^2$$

$$= (x_i - x_j)^T W (x_i - x_j)$$

- $w_i \geq 0, W = \mathrm{diag}(w), (W)_{ii} = w_i$, W is learnable

Mahanalobis Distance Metric

- The off-diagonal elements of W are all 0, which means that the axes are orthogonal, that is, the attributes are irrelevant
- In real problems, there is a correlation between attributes
- Replace W by a positive semi-definite matric M,

$$\mathrm{dist}_{mah}^2(x_i, x_j) = (x_i - x_j)^T M(x_i - x_j) = ||x_i - x_j||_M^2$$

Why is M positive semi-definite (PSD)?

- If M is not PSD, then dist_{mah}^2 could be negative
- Suppose $v = x_i - x_j$ is an eigenvector corresponding to a negative eigenvalue λ of M

$$\begin{aligned}\text{dist}_{mah}^2(x_i, x_j) &= (x_i - x_j)^T M(x_i - x_j) \\ &= v^T M v \\ &= \lambda v^T v = \lambda < 0\end{aligned}$$

How to learn M ? – Neighborhood Component Analysis

- Main idea: Optimize leave-one-out error of a stochastic nearest neighbor classifier in the projection space induced by $\text{dist}_{mah}^2(x_i, x_j)$
- Define the probability of x_j being the neighbor of x_i as

$$p_{ij} = \frac{\exp\left(-||x_i - x_j||_M^2\right)}{\sum_l \exp(-||x_i - x_l||_M^2)}$$

- p_{ij} is the largest when $i = j$
- The expected number of correctly classification points:

$$\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} \sum_{j \in \Omega_i} p_{ij}$$

Neighborhood Component Analysis (NCA)

- Since M is PSD, there must be an orthogonal base P so that m can be written as $M = PP^T$
- The optimization goal of NCA is,

$$\min_{\mathrm{P}} 1 - \sum_{i=1}^{m} \sum_{j \in \Omega_i} \frac{\exp\left(-||P^T x_i - P^T x_j||_2^2\right)}{\sum_l \exp\left(-||P^T x_i - P^T x_l||_2^2\right)}$$

- Solve the above formula to get the distance metric matrix M that maximizes the LOO correct rate of the neighbor classifier

Applications of Dimension Reduction

- Customer relationship management
- Text mining
- Image retrieval
- Microarray data analysis
- Protein classification
- Face recognition
- Handwritten digit recognition
- Intrusion detection

Application – Character Recognition

- Representation: a high-dimensional vector (e.g., 28 x 28 = 784) where each dimension represents the brightness of one pixel

- Essential idea: highly redundant data are likely to be compressible, i.e., the intrinsic dimension may be small, or the dimension related to the learning task may be small

Application – Text document analysis

- Representation: a high-dimensional vector (e.g., 10K) of term frequency over the vocabulary of the word

Term	D1	D2
game	1	0
decision	0	0
theory	2	0
probability	0	3
analysis	0	2
…		

- Underlying structure parameters: topic proportions, clustering structure

Word2Vec

Word Embedding

- One-hot coding
  - In vector space terms, this is a vector with one 1 and a lot of zeros

Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

- Distributed similarity based representations
  - representing a word by means of its neighbors
  - “You shall know a word by the company it keeps” (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical

government debt problems turning into **banking** crises as has happened in

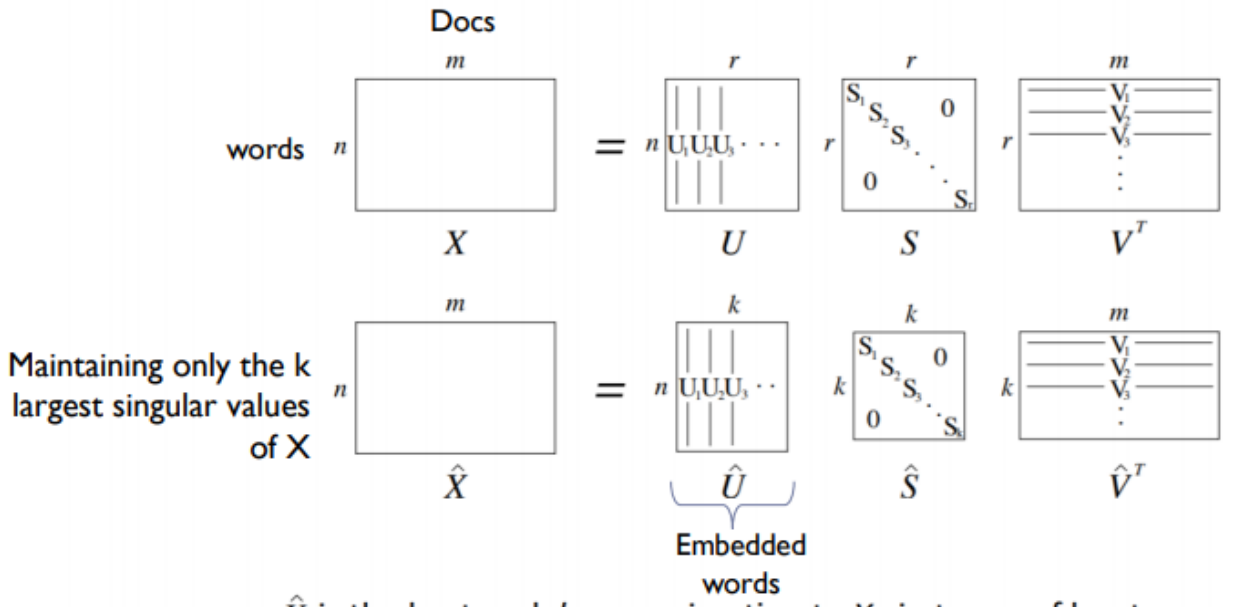
saying that Europe needs unified **banking** regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Word Embedding

- How to make neighbors represent words?
  - Store “most” of the important information in a fixed, small number of dimensions: a dense vector
  - With a co-occurrence matrix X
  - LSA: Dimensionality Reduction based on word-doc matrix

Singular Value Decomposition of cooccurrence matrix *X*.

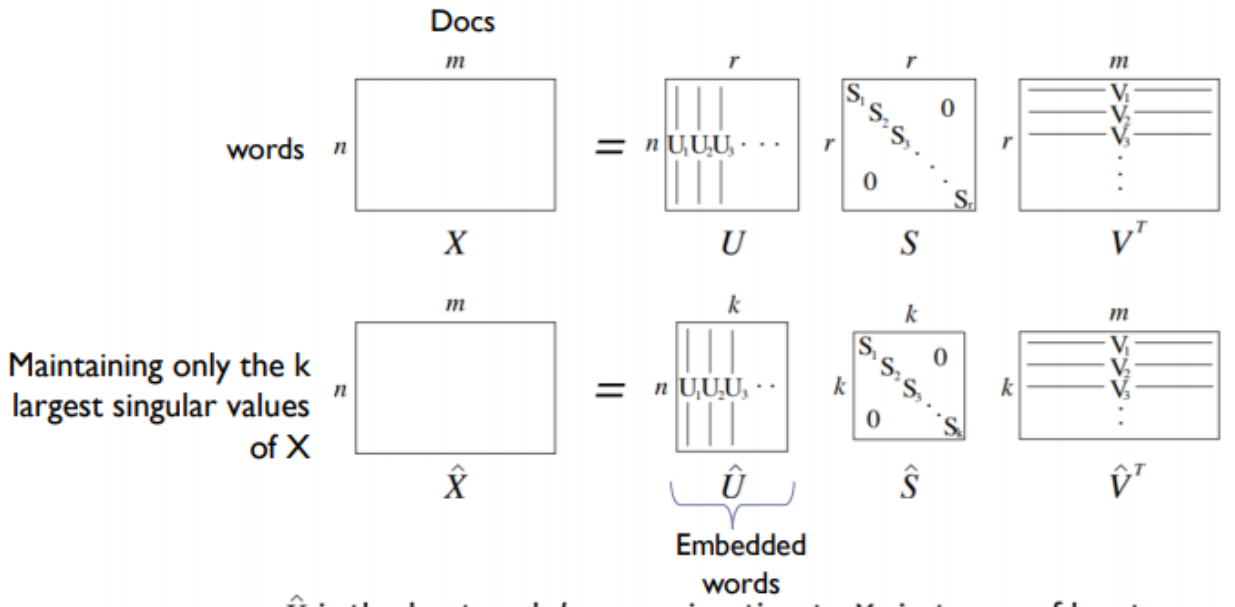


$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

Word Embedding

- How to make neighbors represent words?
  - Store “most” of the important information in a fixed, small number of dimensions: a dense vector
  - With a co-occurrence matrix X
  - LSA: Dimensionality Reduction based on word-doc matrix

Singular Value Decomposition of cooccurrence matrix *X*.



$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

Word Embedding

- Problems with SVD
  - Its computational cost scales quadratically for n x m matrix: $O(mn^2)$ (when n<m)
    - Bad for millions of words or documents
  - Hard to incorporate new words or documents
  - Does not consider order of words in the documents

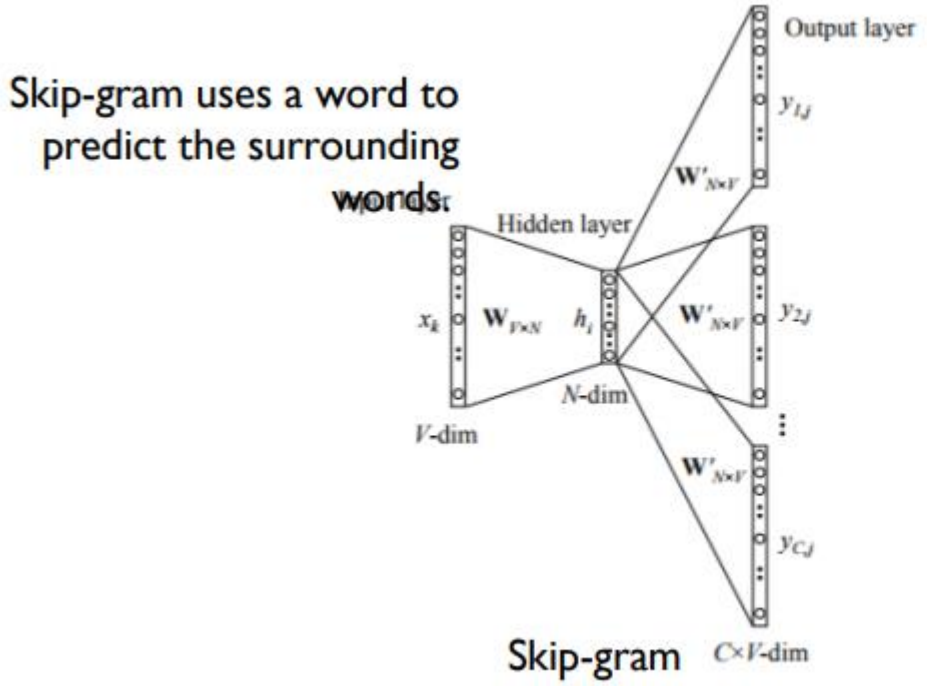
Word2Vec

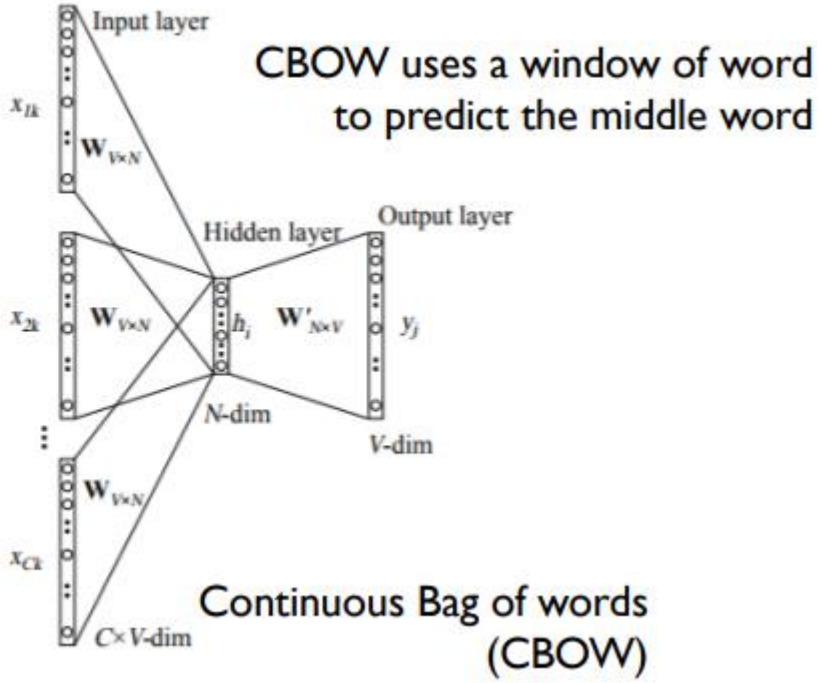
- Directly learn low-dimensional word vectors
- Key idea:The word vector can predict surrounding words
- Originally described (Mikolov et al 2013), a NN model using a two-layer network (i.e., not deep!) to perform dimensionality reduction.
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary
- Very computationally efficient, good all-round model (good hyper-parameters already selected)

Skip-gram vs. Continuous bag-of-words (CBOW)

- Two possible architectures:
  - given some context words, predict the center (CBOW)

    Predict center word from sum of surrounding word vectors
  - given a center word, predict the contexts (Skip-gram)





Objective of Word2Vec (use skip-gram as example)

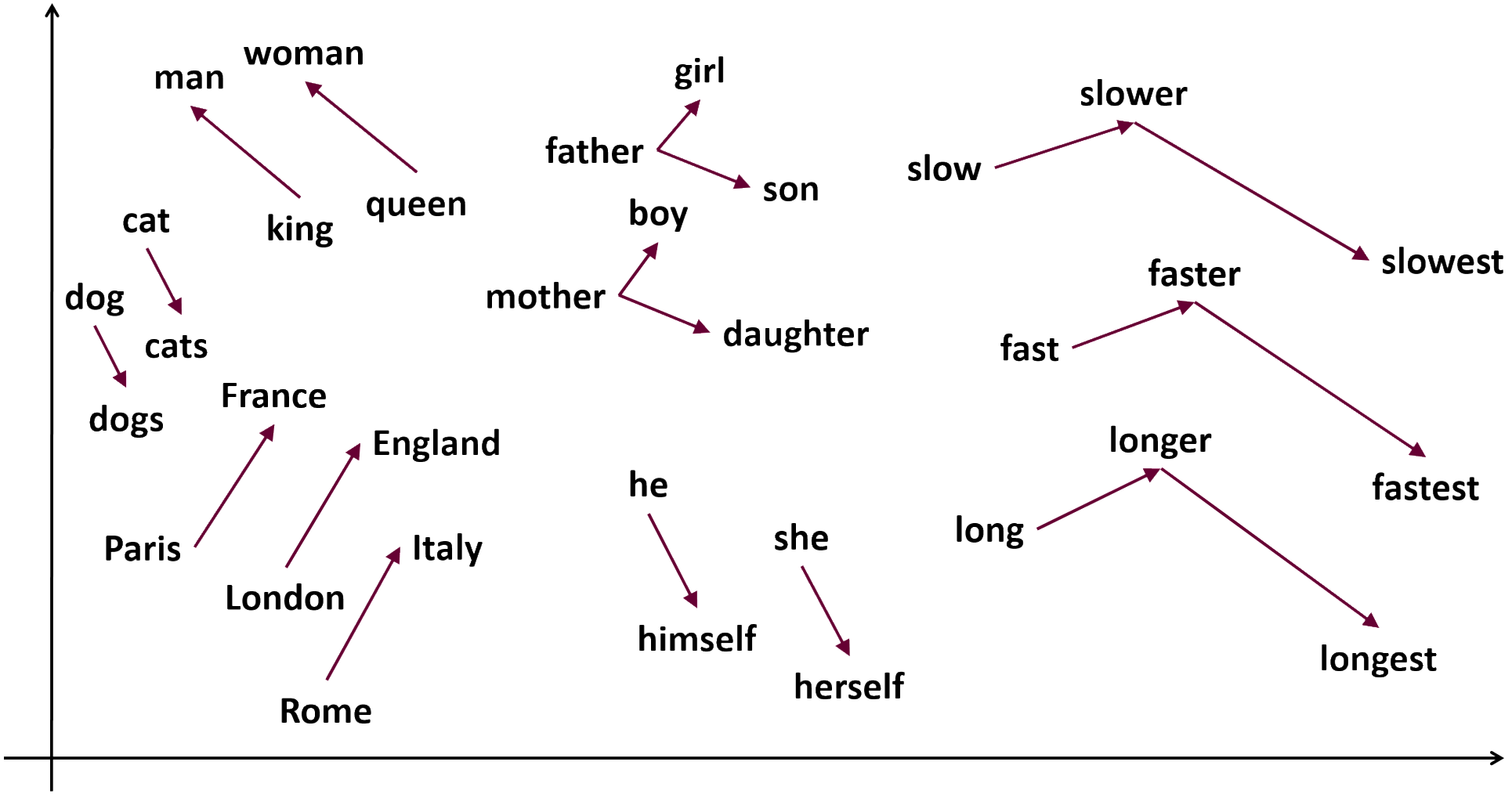
- Maximize the log likelihood of context word

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

$$p(w_{t+j}|w_t) = \frac{\exp(u_{w_{t+j}} \cdot v_{w_t})}{\sum_{w'} \exp(u_{w'} \cdot v_{w_t})}$$

- Every word has 2 vectors
  - v_w: when w is the center word
  - u_w: when w is the outside word (context word)
- Use gradient descent to minimize J

Word2Vec


man
woman
king
queen
girl
father
son
boy
mother
daughter
slow
slower
slowest
cat
cats
dog
dogs
fast
faster
fastest
France
England
Paris
London
Italy
Rome
he
she
himself
herself
long
longer
longest

Summary

- Dimension Reduction: mapping of the original high-dim data into a lower-dim space
- Main idea:
  - Minimization of reconstruction error
  - Maximization of variance
- Algorithms:
  - Linear (MDS, PCA)
  - Non-linear (KPCA, Manifold Learning, Metric Learning)