(1) Spring 2013

Non-Linear Dimension Reduction.

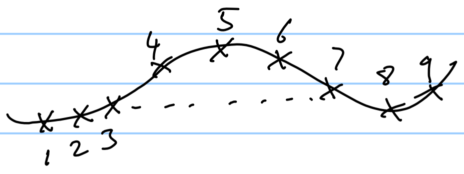
Note Title 11/2/2010

Spectral Methods:

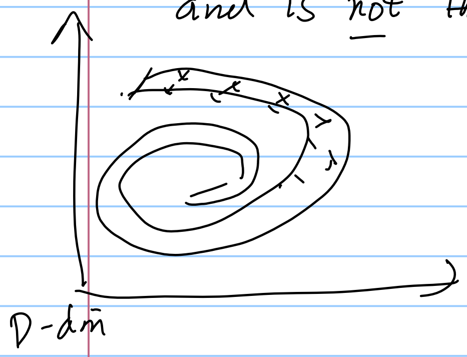
Basic Idea — assume that the data lies on a manifold/surface in D-dim space.

Perform MDS, or other reduction method, using distances calculated on the manifold.

Example:



Distance from x_3 to x_7 is the sum of the distances x_3 to x_4, x_4 to x_5, x_5 to x_6, x_6 to x_7 and is not the direct distance between x_3 and x_7.

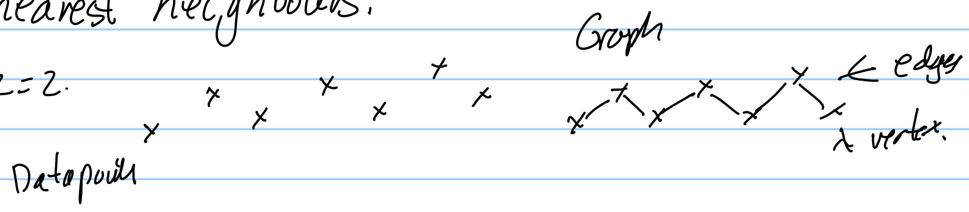


Strategy:
(1) Derive sparse graph using kNN (nearest neighbor)
(2) Derive matrix from the graph weights
(3) Derive embedding from eigenvectors.

First example: ISOMAP.

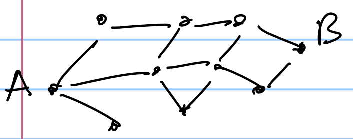
Step 1. Datapoints $\{\underline{x}_i : i = 1 \text{ to } N\}$
Construct an adjacency graph $(V, \mathcal{E})$, where the vertices are the datapoints $\rightarrow$ e.g. $|V| = N$
the edges $\mathcal{E}$ connect each node to its k-nearest neighbours.

E.G. $k=2$.



Spring 2013

(2) Step 2. Estimate geodesics — shortest distance between two nodes i, j of the graph.



Use dynamic programming to calculate the shortest path from A to B

This gives a geodesic distance Δ_{ij} between point x_i and x_j — measured on the manifold.

Step 3. Apply MDS to the Δ_{ij}

Summary : (1) k nearest neighbour — complexity $O(n^2 D)$
(2) shortest path by dynamic programming $O(n^2 \log n + n^2 k)$
(3) MDS — complexity $O(n^2 d)$.

Comment : simple and intuitive algorithm.
⤳ problem if algorithm finds short-cuts e.g.

⤳ only projects the datapoints
∼ needs more work to project new data x.

Results are intuitive.
For example — take two images of a face and interpolate in ISOMAP feature space (y's)

Note: complexity is high. Impractical for large number of datapoints.
Solution → Landmark ISOMAP
identify subset of datapoints as landmarks
do ISOMAP on landmarks.
interpolate for other datapoints.

Theoretical Guarantee if the manifold is isometric to a convex subset of Euclidean space then ISOMAP recovers that subset (up to rotation & translation)

an isometry is a distance-preserving map.

(3) ISOMAP was the first of several methods which used similar strategies.

Next example: Locally Linear Embedding (LLE)

Step 1: As before, construct adjacency graph using kNN.

Step 2: Compute weights ~ characterize the local geometry by weights w_{ij} choosen to minimize

$$\Phi(w) = \sum_i \left| \underline{x}_i - \sum_j w_{ij} \underline{x}_j \right|^2$$

reconstruction error, predict $\underline{x}_i$ from its neighbors.

$w_{ii} = 0$

$\sum_j w_{ij} = 1$

local invariance → optimal weights are invariant to rotation, translation, and scaling.

(Can use this technique to reconstruct from landmarks)

Step 3. Linearization

map inputs to outputs y_i to minimize the reconstruction errors

$$\psi(y) = \sum_i \left| \underline{y}_i - \sum_j w_{ij} \underline{y}_j \right|^2$$

Constraint $\sum_j \underline{y}_j = 0$

impose $\frac{1}{N} \sum_i \underline{y}_i \underline{y}_i^T = \underline{\underline{I}}$ identity.

$$\psi(y) = \sum_{ij} \Psi_{ij} \, \underline{y}_i \cdot \underline{y}_j$$

where $\Psi = (I-W)^T (I-W)$

use bottom $d+1$ eigenvectors

Both optimizations $\Phi(w)$ & $\Psi(y)$ can be performed by linear algebra

Properties: polynomial-time optimization (no need to compute geodesics)

does not estimate the dimension.

no theoretical guarantees.

Closely Related to Laplacian Eigenmaps

Step 1: Build adjacency graph.

Step 2: Assign weights $w_{ij} = e^{-\beta \| \underline{x}_i - \underline{x}_j \|^2}$

Step 3: Compute outputs by minimizing

$$\psi(y) = \sum_{ij} \frac{w_{ij} \left| \underline{y}_i - \underline{y}_j \right|^2}{\sqrt{D_{ii} D_{jj}}}$$

$D_{ii} = \sum_j w_{ij}$.

Spring 2013

(4) Questions:

(1) How to estimate dimensionality? ISOMAP does this (eigenvalues) but LLE does not.

(2) Should we preserve distances? will other criteria give us better ways to reduce the dimension.

Two New Algorithms:

Algorithm 1 → preserve local distances. Isometry.
Algorithm 2 → preserve local angles. Conformal Mapping

Isometry → smooth invertible mapping that preserves distances and looks locally like rotation & translation.
Intuition – take sheet of paper, but don't tear.

Algorithm 1   Maximum Variance Unfolding.
generalizes PCA computation of "maximum variance subspace.

Let $K_{ij} = \underline{y}_i \cdot \underline{y}_j$ Gram matrix. $\sum_i K_{ii}$ like variance

Maximize $\sum_i K_{ii}$ subject to:

Semi-Definite Programming (SDP)
(i) $K_{ii} + K_{jj} - 2K_{ij} = \|\underline{x}_i - \underline{x}_j\|^2$ if $\underline{x}_i, \underline{x}_j$ are k-nn
(ii) $\sum_i K_{ij} = 0$
(iii) K is positive semi-definite

Summary: (1) Nearest neighbor to compute adjacency graph,
(2) semi-definite programming (SDP) to compute maximum variance unfolding
(3) Diagonalize Gram matrix – estimate dimension from rank.

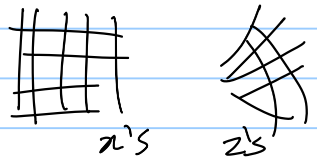
Speed up by using Landmarks.
But is still limited to isometric transformation.

Spring 2013

(5).

Algorithm 2. Preserve local angles (only)

Conformal mapping:
rotation, translation, and scaling.



Construct. k-nearest neighbor graph.
compare edges — are lengths the same up to scaling?

$$D = \sum_{ijk} \eta_{ij}\eta_{ik}\left[\,|\underline{z}_j-\underline{z}_k|^2 - s_i\,|x_j-x_k|^2\right]^2$$

$\times$ scaling at x_i

Given inputs $\langle x_i \rangle$, can we compute outputs $\langle z_i \rangle$ and scaling parameters $\langle s_i \rangle$, such that the angles are well-preserved?

Problem — trivial solution, $s=0$, $\underline{z}=0$.
need to regularize the solution.

Require the z's to be expressed in form $\underline{z}_i = \underline{L}\,\underline{y}_i$
$(L = D - W)$ $D_{ii} = \sum_j W_{ij}$

Only use the m smoothest eigenvectors of L.
(prevents the trivial solution $Tr(L^T L)=1$)

strategy to minimize

$$D(L,s) = \sum_{ijk} \eta_{ij}\eta_{ik}\left[\,\|L(\bar{y}_j-\bar{y}_k)\|^2 - s_i\|\underline{x}_j-\underline{x}_k\|^2\right]^2$$

solve for scaling factor $\frac{\partial D}{\partial s}=0$

Reduce to a Semi-Definite Programming Problem.

Relation to Kernel PCA

KPCA (kernel PCA) – linear in feature space.
– non-linear in input space.

But can kernel PCA perform isometric transformation?

Answer, No although KPCA with RBF kernel is approximately a local isometry.

But spectral methods – Isomap, etc – can be seen as ways to construct kernel matrices.
i.e not generic kernels, but data-driven kernels

Spring 2013

(7)

Note Title 11/2/2010

Multidimensional Scaling (MDS)

MDS is a linear projection method. It is related to PCA. MDS and PCA can be used for non-linear projection – next lecture.

Key Idea of MDS: project to preserve the distances $|\underline{x}_i - \underline{x}_j|$ between the datapoints $\{\underline{x}_i : i = 1 \text{ to } N\}$,

i.e. $\underline{x}_i \to \underline{y}_i$ such that $|\underline{x}_i - \underline{x}_j| \approx |\underline{y}_i - \underline{y}_j|$,

but the $\underline{y}$'s have lower dimension.

This projection constraint is imposed on the dot products $\underline{x}_i \cdot \underline{x}_j \approx \underline{y}_i \cdot \underline{y}_j$

this will imply ~~that~~ $|\underline{x}_i - \underline{x}_j| \approx |\underline{y}_i - \underline{y}_j|$.

Important Property: we only need to know $|\underline{x}_i - \underline{x}_j| \triangleq \Delta_{ij}$ in order to calculate $\underline{x}_i \cdot \underline{x}_j$. This will be useful for non-linear applications later. Also, sometimes only $|\underline{x}_i - \underline{x}_j|$ is specified.

Result: $\underline{x}_i \cdot \underline{x}_j = \frac{1}{2N} \sum_k \Delta_{ik}^2 + \frac{1}{2N} \sum_l \Delta_{lj}^2 - \frac{1}{2N^2} \sum_{lk} \Delta_{lk}^2 - \frac{1}{2} \Delta_{ij}^2$

provided $\sum_i \underline{x}_i = 0$. $\Rightarrow$ subtract $\frac{1}{N}\sum_i \underline{x}_i$ from data to ensure this.

Proof: $\Delta_{ij}^2 = |\underline{x}_i|^2 + |\underline{x}_j|^2 - 2\underline{x}_i \cdot \underline{x}_j$

Let $T = \sum_i |\underline{x}_i|^2$, Note that $\sum_i \underline{x}_i \cdot \underline{x}_j = 0 = \sum_j \underline{x}_i \cdot \underline{x}_j$.

Then $\sum_k \Delta_{ik}^2 = N|\underline{x}_i|^2 + T$, $\sum_l \Delta_{lj}^2 = N|\underline{x}_j|^2 + T$, $\sum_{lk} \Delta_{lk}^2 = 2NT$.

the result follows by substitution.

Define the Gram matrix

$$G_{ij} = \underline{x}_i \cdot \underline{x}_j = \sum_k \Delta_{ik}^2 + \frac{1}{2N} \sum_l \Delta_{lj}^2 - \frac{1}{2N^2} \sum_{lk} \Delta_{lk}^2 - \frac{1}{2} \Delta_{ij}^2 .$$

Define an error function

$$err(\underline{y}) = \sum_{ij} (G_{ij} - \underline{y}_i \cdot \underline{y}_j)^2$$

$\underline{x}_i$ lies in D-dim space, G_{ij} is $N \times N$ matrix

$\underline{y}_i$ is a vector in d-dim space $\quad d << D$

$d < N$

(8) Spring 2013

Minimize $\text{err}(\underline{y}) = \sum_{ij} (G_{ij} - \underline{y}_i \cdot \underline{y}_j)^2$.

Do spectral decomposition:

$$\underline{\underline{G}} = \sum_{\alpha=1}^{N} \lambda_\alpha \underline{v}_\alpha \underline{v}_\alpha^T$$

$\lambda_1 \geq \ldots \geq \lambda_n \geq 0$ eigenvalues of $\underline{\underline{G}}$

$\underline{v}_\alpha \cdot \underline{v}_\beta = \delta_{\alpha\beta}$ eigenvectors

Claim: optimal minimization is $y^i_\alpha = \sqrt{\lambda_\alpha}\, v^\alpha_i$

i.e. $\underline{y}_i = (\sqrt{\lambda_1}\, v^1_i, \sqrt{\lambda_2}\, v^2_i, \ldots, \sqrt{\lambda_n}\, v^N_i)$

N-dimension

Proof. $\underline{y}_i \cdot \underline{y}_j = \sum_\alpha \sqrt{\lambda_\alpha}\, v^\alpha_i \sqrt{\lambda_\alpha}\, v^\alpha_j = \sum_\alpha \lambda_\alpha \underline{v}_\alpha \underline{v}_\alpha^T = G_{ij}$

This gives $\text{err} = 0$.

But we can reduce the dimension further by truncating $\underline{y}_i$ to $\underline{y}_i = (\sqrt{\lambda_1}\, v^1_i, \ldots, \sqrt{\lambda_d}\, v^d_i)$ for $d < N$

In this case $G_{ij} \neq \underline{y}_i \cdot \underline{y}_j$

$\underline{y}_i \underline{y}_j = \sum_{\alpha=1}^{d} \lambda_\alpha v^\alpha_i v^\alpha_j$, $G_{ij} = \sum_{\alpha=1}^{N} \lambda_\alpha v^\alpha_i v^\alpha_j$

Hence $G_{ij} - \underline{y}_i \cdot \underline{y}_j = \sum_{\alpha=d+1}^{N} \lambda_\alpha v^\alpha_i v^\alpha_j$

Claim $\sum_{ij} (G_{ij} - \underline{y}_i \cdot \underline{y}_j)^2 = \sum_{\alpha=d+1}^{N} \lambda_\alpha^2$

Proof. $\sum_{ij} \left\{ \sum_{\alpha=d+1}^{N} \sum_{\beta=d+1}^{N} \lambda_\alpha \lambda_\beta v^\alpha_i v^\alpha_j v^\beta_i v^\beta_j \right)$,

$\sum_i v^\alpha_i v^\beta_i = \delta^{\alpha\beta}$, $\sum_j v^\alpha_j v^\beta_j = \delta^{\alpha\beta}$

$$\sum_{\alpha=d+1}^{N} \sum_{\beta=d+1}^{N} \lambda_\alpha \lambda_\beta \delta^{\alpha\beta} \delta^{\alpha\beta} = \sum_{\alpha=d+1}^{N} \lambda_\alpha^2.$$

Hence we project to d-dimension provided $\sum_{\alpha=d+1}^{N} \lambda_\alpha^2$ is small, or $\dfrac{\sum_{\alpha=d+1}^{N} \lambda_\alpha^2}{\sum_{\alpha=1}^{N} \lambda_\alpha^2}$ is small.

$\underline{y}_i = (\sqrt{\lambda_1}\, v^1_i, \ldots, \sqrt{\lambda_d}\, v^d_i)$

Spring 2013 ?

Relation between MDS & PCA ?

Both linear. Both depend on eigenvectors/eigenvalue.

Recall PCA (subtract mean to ensure $\sum_i \underline{x}_i = 0$)

$\underline{x}_p = (\underline{x}.\underline{e}_1, \ldots, \underline{x}.\underline{e}_d)$

the $\underline{e}$'s are eigenvectors of $K_{ab} = \frac{1}{N}\sum_{i=1}^{N} \underline{x}_i \underline{x}_i^T$

MDS $\underline{y}_i = (\sqrt{\lambda_1} v_i^1, \ldots, \sqrt{\lambda_d} v_i^d)$

the $\underline{v}$'s are eigenvectors of $G_{ij} = \sum_{i=1}^{N} \underline{x}_i \cdot \underline{x}_j$

Claim: the eigenvalues of $\underline{\underline{G}}$ and $\underline{\underline{K}}$ are the same. The eigenvectors are closely related.

Proof Let $\underline{\underline{X}}$ be an $N \times D$ matrix $i = 1$ to N no. of points, $a = 1$ to D space dimension

with elements X_{ia} ath component of ith datapoint

Consider $\underline{\underline{X}}\,\underline{\underline{X}}^T$ $N \times N$ matrix, $(\underline{\underline{X}}\,\underline{\underline{X}}^T)_{ij} = \sum_a X_{ia} X_{ja}$

$\underline{\underline{X}}^T\underline{\underline{X}}$ $D \times D$ matrix, $(\underline{\underline{X}}^T\underline{\underline{X}})_{ab} = \sum_i X_{ia} X_{ib}$

Both are square matrices and both are positive definite, so they have positive eigenvalues.

$\underline{\underline{X}}\,\underline{\underline{X}}^T$ is used for MDS, $\underline{\underline{X}}^T\underline{\underline{X}}$ is used for PCA

Suppose $\underline{e}, \lambda$ are an eigenvector, eigenvalue of $\underline{\underline{X}}^T\underline{\underline{X}}$

$$\underline{\underline{X}}^T\underline{\underline{X}}\,\underline{e} = \lambda \underline{e}$$

So $$\underline{\underline{X}}\,\underline{\underline{X}}^T\underline{\underline{X}}\,\underline{e} = \lambda \underline{\underline{X}}\,\underline{e}$$

$$(\underline{\underline{X}}\,\underline{\underline{X}}^T)(\underline{\underline{X}}\,\underline{e}) = \lambda(\underline{\underline{X}}\,\underline{e})$$

So $(\underline{\underline{X}}\,\underline{e})$ is an eigenvector (un-normalized) of $\underline{\underline{X}}\,\underline{\underline{X}}^T$ with eigenvalue λ.

Similarly if $\underline{\underline{X}}\,\underline{\underline{X}}^T\underline{v} = \lambda \underline{v}$ then $(\underline{\underline{X}}^T\underline{v})$ is an eigenvector (un-normalized) of $\underline{\underline{X}}^T\underline{\underline{X}}$ with eigenvalue λ.

Conclusion $\rightarrow$ the two matrices have the same eigenvalues and related eigenvectors.

Spring 2013

Result → the truncation conditions for MDS and PCA are similar $\frac{\sum_{i=1}^{d} \lambda_i^2}{\sum_{i=1} \lambda_i^2} >$ Threshold.

MDS projects $\underline{x}_i$ to $\underline{y}_i = (\sqrt{\lambda_1} v_i^1, \dots, \sqrt{\lambda_d} v_i^d)$

PCA projects $\underline{x}$ to $\underline{y} = (\underline{x}\cdot\underline{e}_1, \dots \underline{x}\cdot\underline{e}_d)$
where the $\underline{e}$'s and the $\underline{v}$'s are related (see previous page).

Note: The equivalence between the eigenvalues of $\underline{X}\underline{X}^T$ and $\underline{X}^T\underline{X}$ has computational importance
If $N \ll D$, faster to compute eigenvalues/vectors for $(\underline{X}\underline{X}^T)$, then convert to eigenvalues/vectors of $\underline{X}^T\underline{X}$

Note: to do PCA we have to know the data $\{\underline{x}_i\}$, but to do MDS we only need to know Δ_{ij}.
In some applications it is possible to specify Δ_{ij} but not the $\{\underline{x}_i\}$.

Deeper Understanding: This relationship between $\underline{X}\underline{X}^T$ and $\underline{X}^T\underline{X}$ can be used to proove SVD:

$\underline{X} = \underline{F}\,\underline{D}\,\underline{E}$ where $\underline{F}\underline{F} = \underline{I}$ ✗ Identity.
$\underline{D} = \begin{pmatrix} d_1 & 0 \\ 0 & \ddots d_D \end{pmatrix}$ $\quad \underline{E}\underline{E}^T = \underline{I}$ ✗

This is a generalization of the spectral decomposition
$\underline{G} = \sum \lambda_a \underline{v}_a \underline{v}_a^T$ $\quad \underline{X}$ is $N \times D$ $\quad N \neq D$
to any matrix $\underline{X}$ → so $\underline{X}$ is not square.

It follows that $\underline{X}^T\underline{X} = (\underline{E}^T\underline{D}\underline{F}^T)(\underline{F}\underline{D}\underline{E})$
$= \underline{E}^T\underline{D}^2\underline{E}$
spectral decomposition → with $\lambda_1 = d_1^2, \lambda_2 = d_2^2 \dots$
and $\underline{X}\underline{X}^T = \underline{F}\underline{D}\underline{E}\,\underline{E}^T\underline{D}\underline{F}^T = \underline{F}\underline{D}^2\underline{F}^T$

So SVD is like the square root **of spectral decomposition.**