Dimension reduction 3: Nonlinear dimension reduction

Ryan Tibshirani
Data Mining: 36-462/36-662

February 12 2013

Optional reading: ESL 14.8–14.9

Reminder: principal component directions and scores

Given a matrix $X \in \mathbb{R}^{n\times p}$, we defined the principal component directions to be orthonormal vectors in $\mathbb{R}^p$ along which the sample variance of X is successively maximized. For centered X, the kth principal component direction is

$(Xcv)^T(Xcv) = c^2 (Xv)^T(Xv)$

$$\underline{v_k} = \underset{\substack{\|v\|_2=1 \\ v^T v_j=0,\ j=1,\ldots k-1}}{\operatorname{argmax}} \quad (Xv)^T(Xv) \leftarrow \text{sample directions}$$

$\leftarrow$ makes orthogonal

The vector $Xv_k \in \mathbb{R}^n$ is called the kth principal component score of X, and $u_k = (Xv_k)/d_k \in \mathbb{R}^n$ is the normalized kth principal component score, with $d_k = \sqrt{(Xv_k)^T(Xv_k)}$. The quantity $\underline{d_k^2/n}$ is the amount of variance explained by v_k

This is all captured by singular value decomposition $\underline{X = UDV^T}$:

- $U \in \mathbb{R}^{n\times p}$ and has columns $u_1, \ldots u_p$
- $D \in \mathbb{R}^{p\times p}$ and $D = \operatorname{diag}(d_1, \ldots d_p)$
- $V \in \mathbb{R}^{p\times p}$ and has columns $v_1, \ldots v_p$

preserve pairwise distances

Reminder: principal component scores and representations

$X \in \mathbb{R}^{n \times p}$

Think of the first k scores $Xv_1 = d_1 u_1, \dots Xv_k = d_k u_k \in \mathbb{R}^n$ as new feature vectors. Write this as $Z = XV_k = (UD)_k \in \underline{\mathbb{R}^{n\times k}}$, and think of Z as are new low-dimensional representation for X

The rows $z_1, \dots z_n \in \mathbb{R}^k$ of Z are the data points in this new low-dimensional representation. Question: how are the pairwise distances related to those of $x_1, \dots x_n \in \mathbb{R}^p$ (rows of X)?

$Z = XV$

Note that $z_i = V_k^T x_i$. Because V_k has orthonormal columns,

$$\underline{\|z_i - z_j\|_2} = \|V_k^T x_i - V_k^T x_j\|_2 = \|V_k V_k^T x_i - V_k V_k^T x_j\|_2$$

Finally, $\|V_k V_k^T x_i - V_k V_k^T x_j\|_2 \approx \underline{\|x_i - x_j\|_2}$, because we saw last time that $XV_kV_k^T$ was the best rank k approximation of X

So answer: they're pretty close!

if O is orthogonal matrix

then $\|Ox\|_2^2 = \|x\|_2^2 \quad \forall x$

$x^T \underline{O^T O} x = x^T x$

The inner-product matrix

$X = \begin{bmatrix} - x_1 - \\ \vdots \\ - x_n - \end{bmatrix}$

Given $X \in \mathbb{R}^{n \times p}$, the matrix $XX^T \in \mathbb{R}^{n \times n}$ is the inner-product matrix. If X has ith row $x_i \in \mathbb{R}^p$, then $(XX^T)_{ij} = \underline{x_i^T x_j}$

Suppose that we wanted the principal component scores of X, but we only had XX^T. Could we still compute them?

Yes! Using the singular value decomposition $\underline{X = UDV^T}$, we have

$$XX^T = UD^2U^T = UD\cancel{V^T V}DU^T$$

This is called an eigendecomposition of XX^T because the columns of U are eigenvectors of XX^T. (Check?) $\quad XX^T u_i = ?$

Hence we can compute the eigendecomposition or "factorize" the inner product matrix XX^T, and then the scores are given by the <u>columns of UD</u>, i.e., $d_j u_j$, $j = 1, \ldots p$

Low-dimensional representation from distances only

Suppose that instead of measuring $X \in \mathbb{R}^{n \times p}$ directly, we only measured the distances between pairs of observations,

$$\Delta_{ij} = \|x_i - x_j\|_2, \quad i,j = 1,\ldots n \qquad x_i \in \mathbb{R}^p$$

This gives us a distance matrix $\Delta \in \mathbb{R}^{n \times n}$ pairwise distance matrix

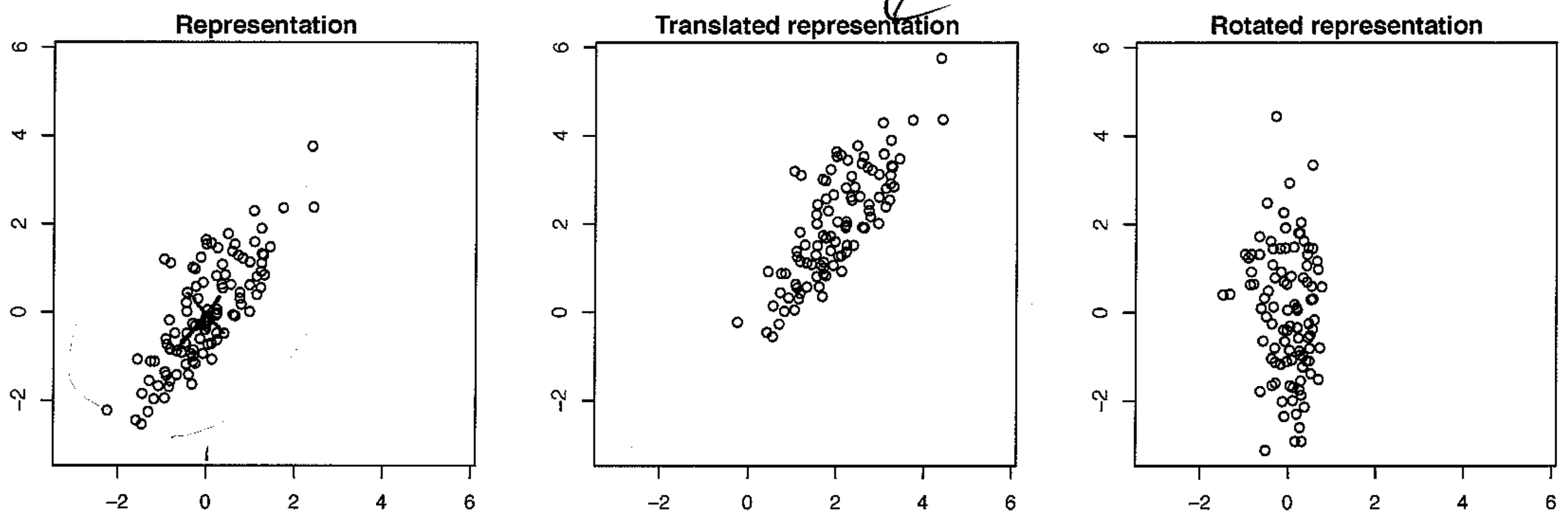
We want a lower-dimensional representation $z_1, \ldots z_n \in \mathbb{R}^k$, for some small k (e.g. $k = 2$ or 3), such that $\|z_i - z_j\|_2 \approx \underline{\Delta_{ij}}$, for every $i,j = 1, \ldots n$

We saw that the principal component scores do exactly this, but these rely on X or XX^T, which we don't have here. We can only use the distances Δ_{ij}

Is this possible? if took $k=p$, and constructed $z_1, \ldots z_n \in \mathbb{R}^p$ from Δ, could we get $z_i = x_i$ each $i=1,\ldots,n$?

Unidentifiability

don't have to worry about



Distances Δ_{ij} are invariant under any orthonormal transformation $O \in \mathbb{R}^{p \times p}$ of $x_1, \ldots x_n$ (i.e., $O^T O = I$)

Multidimensional scaling

(workhorse)

Assume that $X \in \mathbb{R}^{n \times p}$ has been column centered (this is enough to deal with translation unidentifiability) $\|z_i - z_j\|_2 \approx \Delta_{ij}$

Multidimensional scaling (MDS): given distance matrix $\Delta \in \mathbb{R}^{n \times n}$, we

1. Recover the inner-product matrix $B = XX^T \in \mathbb{R}^{n \times n}$ from Δ
2. Factorize B to get the first k principal component scores

(Called classical multidimensional scaling, there are other flavors, e.g., least squares multidimensional scaling)

We've already seen how to do step 2: remember that we compute the eigendecomposition $B = UD^2U^T$ and then the first k principal component scores are the first k columns of UD

So, how do we do step 1, i.e., how do we recover B?

get back first k PC scores.

Recovering inner-products from distances

The following procedure can be used to recover the inner-products $B = XX^T$ from Δ (Homework 2, bonus):

1. Compute $A_{ij} = -\frac{1}{2}\Delta_{ij}^2$ to form the matrix $A \in \mathbb{R}^{n \times n}$
2. Double center A—i.e., center both the columns and rows of A—to recover the matrix $B \in \mathbb{R}^{n \times n}$. Note that this is the same as:

$$B = (I - M)A(I - M)$$

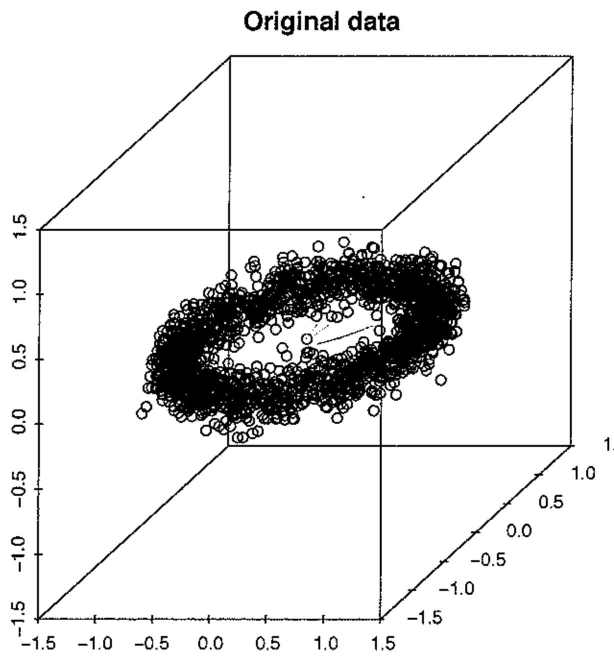
where $M = \frac{1}{n}\mathbb{1}\mathbb{1}^T \in \mathbb{R}^{n \times n}$

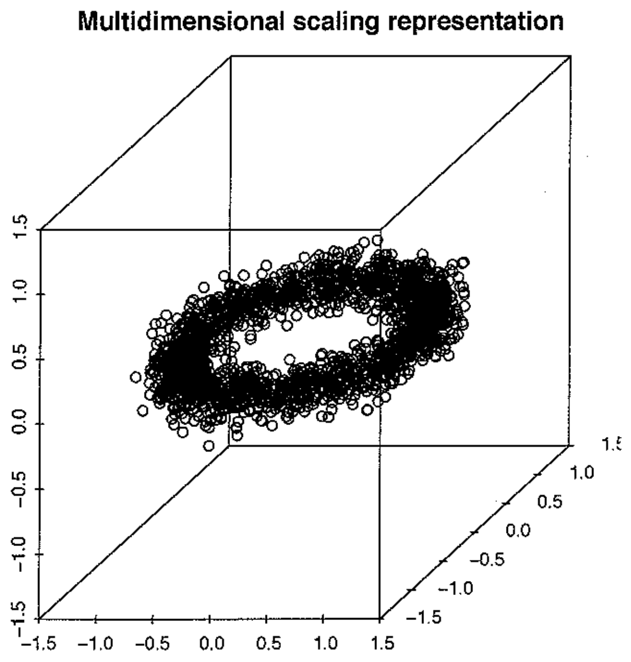
$B = XX^T$

Does it matter whether we first center the columns or the rows?

Example: donut data

Recall donut example: $X \in \mathbb{R}^{2000\times 3}$. The left plot shows X and the right plot shows the multidimensional scaling representation $Z \in \mathbb{R}^{2000\times 3}$ computed from the distance matrix $\Delta \in \mathbb{R}^{2000\times 2000}$





The multidimensional scaling recovery is the same as the principal component scores $UD \in \mathbb{R}^{2000\times 3}$. This is the same as X up to an orthonormal transformation—recall that $UD = \underline{XV}$ $\qquad X$

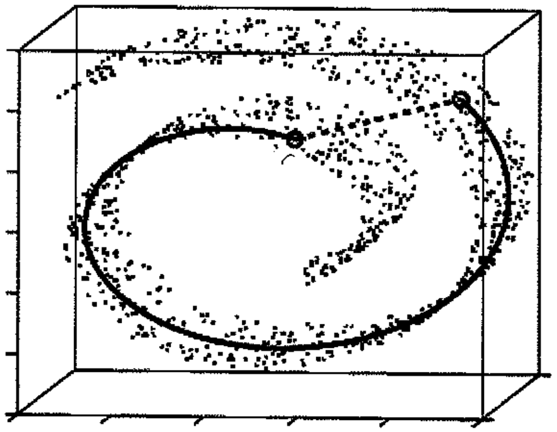
Beyond Euclidean distance

If the Δ_{ij} were actual Euclidean distances between the rows of a centered matrix $X \in \mathbb{R}^{n \times p}$, we get back the first k principal component scores exactly. Importantly, multidimensional scaling can be applied to any Δ_{ij}, not just Euclidean distances

There is a class of methods which construct a fancier metric Δ_{ij} between high-dimensional points $x_1, \ldots x_n \in \mathbb{R}^p$, and then they feed these Δ_{ij} through multidimensional scaling to get a low-dimensional representation $z_1, \ldots z_n \in \mathbb{R}^k$. In this case, we don't just get principal component scores, and our low-dimensional representation can end up being a nonlinear function of the data

Why would we want to use non-Euclidean distances?

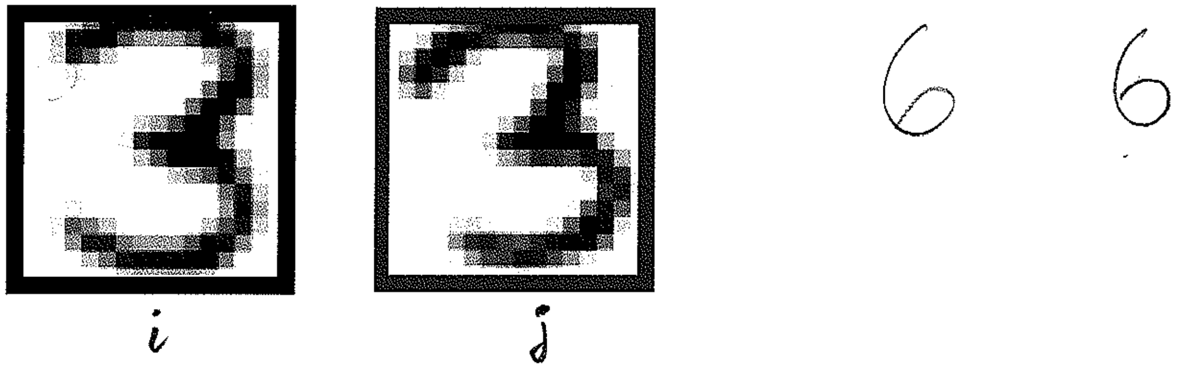
for Δ ij



Tangent distance

Tangent distance is an example of a fancier metric that we can run through multidimensional scaling (though used elsewhere too)

A motivating example is the digit data. Here, we have 16×16 images, treated as points $x_i \in \mathbb{R}^{256}$ (i.e., they are unraveled into vectors). If we take, e.g., a "3" and rotate it through a small angle, we would like for the rotated image to be considered close to the original image. This is not neccessarily true of Euclidean distance

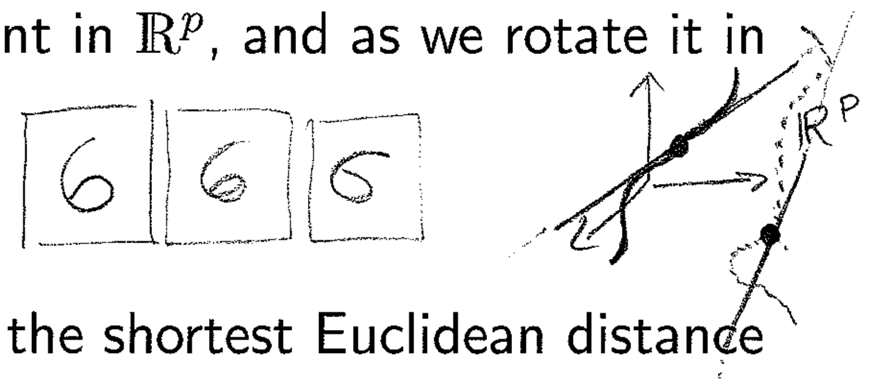


We could define $\Delta_{ij}^{\text{rotation}}$ to be the shortest Euclidean distance between a rotated version of x_i and rotated version of x_j. What's the problem with this? (Hint: think about rotating a "6")

Rotations define a curve

Two problems with $\Delta_{ij}^{\text{rotation}}$: hard to calculate, and allows for too large of a transformation. We need something easier to calculate, and that restricts attention to small rotations

It helps to think of a set of rotations of an image as defining a curve in $\mathbb{R}^p$—an image x_i is a point in $\mathbb{R}^p$, and as we rotate it in either directions, we get a curve

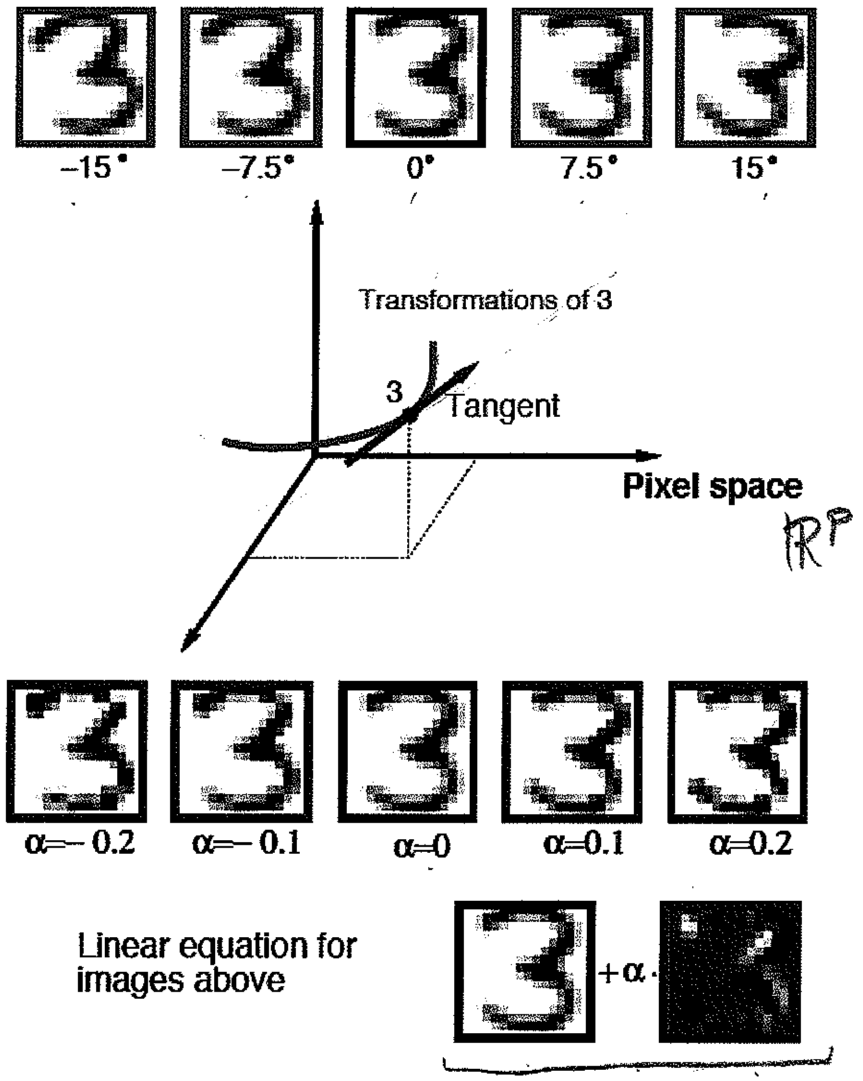


In this perspective:

- Rotation distance $\Delta_{ij}^{\text{rotation}}$ is the shortest Euclidean distance between the two curves generated by rotating x_i and x_j
- Tangent distance $\Delta_{ij}^{\text{tangent}}$ is defined by first computing the tangent line to each curve at the observed image, and then using the shortest Euclidean distance between tangent lines

Using tangent distance remedies both problems above. (Why?)

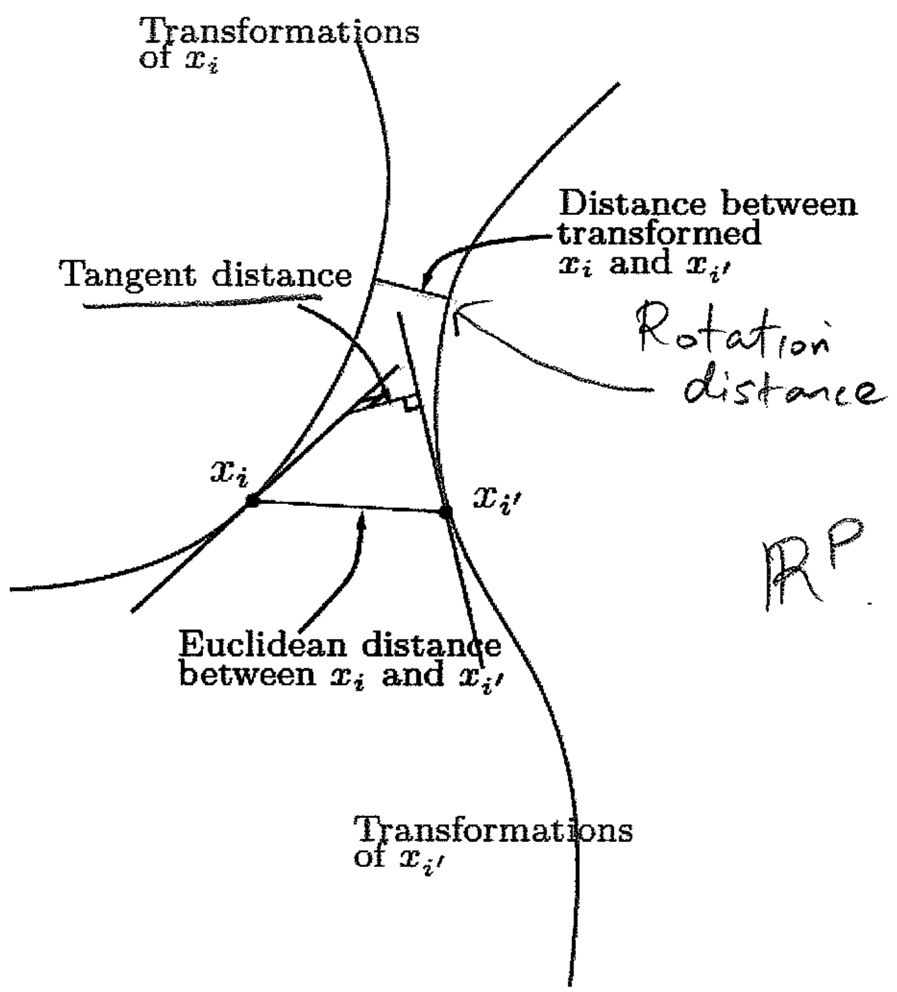
Illustration of rotation curve and tangent



what happens for big α?

(From ESL page 473)

Illustration of tangent distance


Transformations of x_i
Distance between transformed x_i and $x_{i'}$
Tangent distance
Rotation distance
x_i
$x_{i'}$
$\mathbb{R}^P$
Euclidean distance between x_i and $x_{i'}$
Transformations of $x_{i'}$

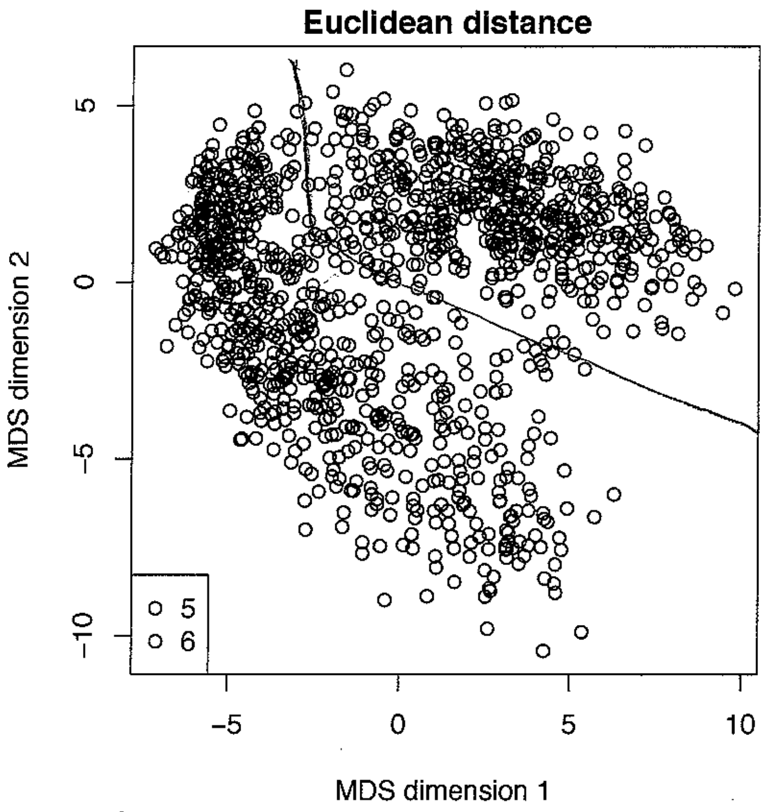
(From ESL page 474)

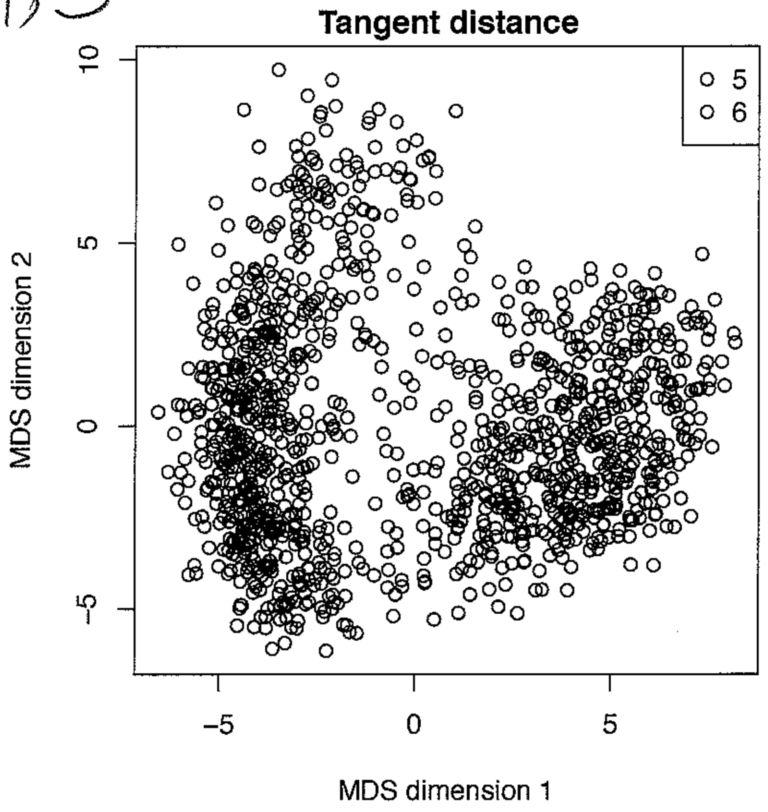
Example: handwritten digits

Example: $n = 1220$ images of handwritten digits and $p = 256$, i.e., each image is 16 $\times$ 16 pixels. Left plot: first two multidimensional scaling dimensions using Euclidean distances (principal component directions), right plot: using tangent distances

3 3

4,5



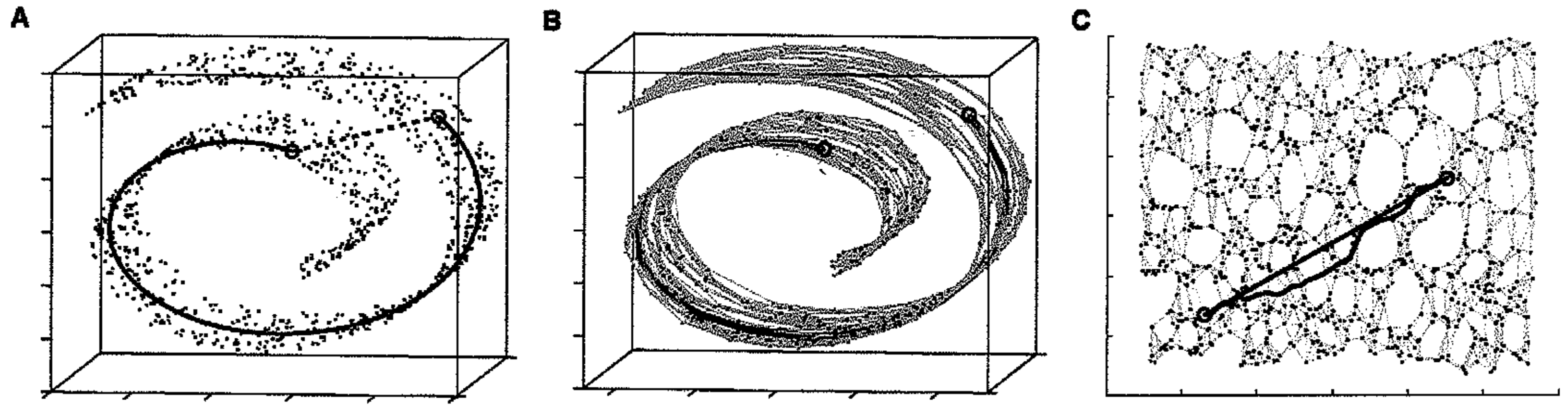


Same as first 2 PC scores

think about separation
RE: clustering

Isometric feature mapping

Isometric feature mapping[1] (Isomap) learns structure in a more general setting to define distances. The basic idea is to construct a graph $G = (V, E)$, i.e., construct edges E between vertices $V = \{1, \ldots n\}$, based on the structure between $x_1, \ldots x_n \in \mathbb{R}^p$. Then we define a graph distance $\Delta_{ij}^{\text{Isomap}}$ between i and j, and use multidimensional scaling for our low-dimensional representation



(From Tenenbaum et al. (2000))

[1]Tenenbaum et al. (2000), "A global geometric framework for nonlinear dimensionality reduction"

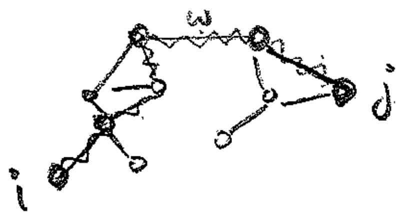
The Isomap graph distances

Constructing the graph: for each pair i, j, we connect i, j with an edge if either:

- x_i is one of x_j's m nearest neighbors, or   e.g., $m=5$
- x_j is one of x_i's m nearest neighbors

The weight of this edge $e = \{i, j\}$ is then $w_e = \|x_i - x_j\|_2$

Defining graph distances: now that we have built a graph, i.e., we have built an edge set E, we define the graph distance $\Delta_{ij}^{\text{Isomap}}$ between x_i and x_j to be the shortest path in our graph from i to j:



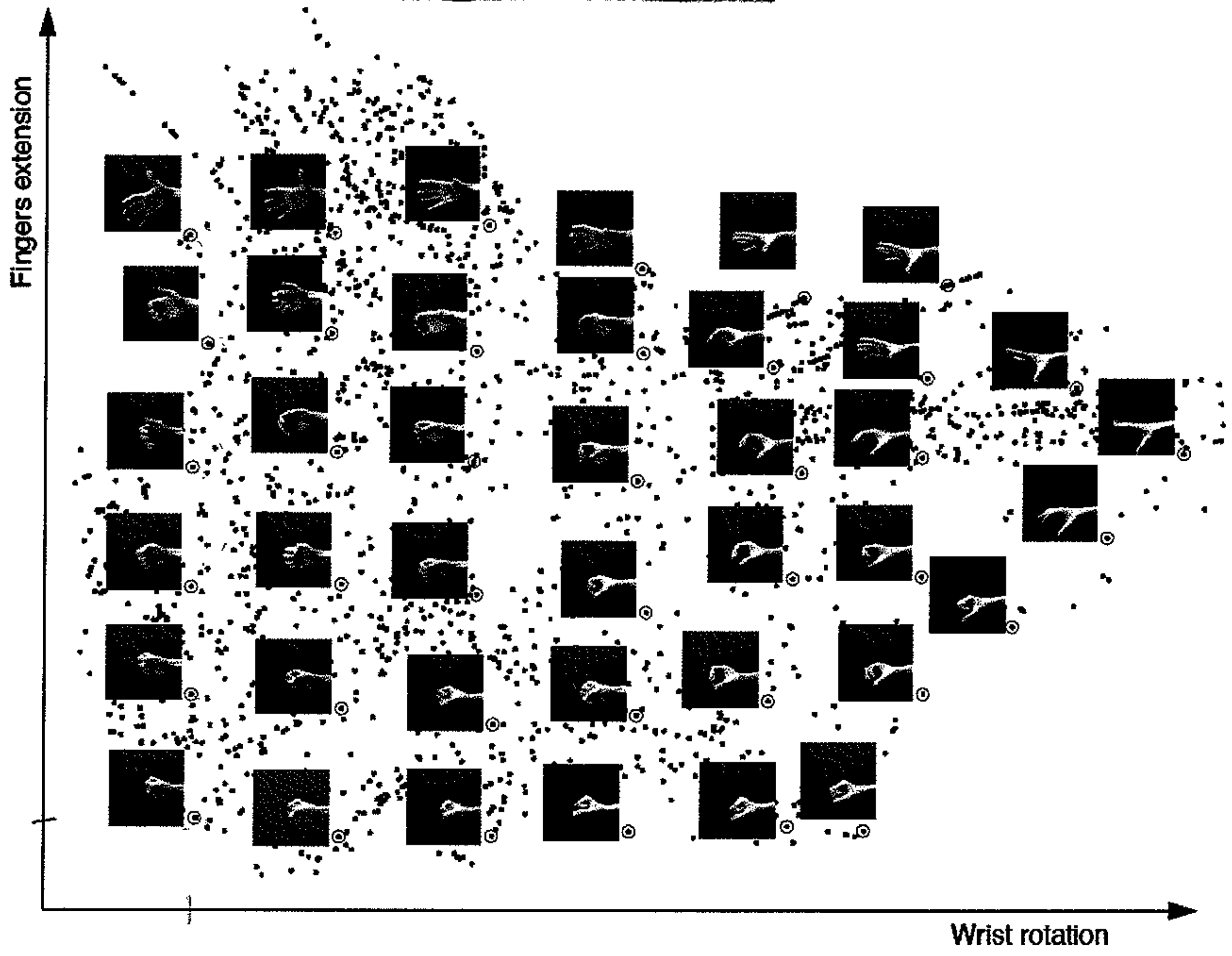
$$\Delta_{ij}^{\text{Isomap}} = \min_{\substack{\text{paths } P \subseteq E \\ \text{from } i \text{ to } j}} \sum_{e \in P} w_e$$

(This can be computed by, e.g., Dijkstra's algorithm or Floyd's algorithm)

we have to choose m ...

Example: hand positions

Example: $n = 3000$ images of hands and $p = 4096$, i.e., each image is 64×64 pixels. Isomap applied with $m = 6$ nearest neighbors, reduced to $k = 2$ dimensions:



(From `http://isomap.stanford.edu/handfig.html`)

Local linear embedding

Local linear embedding[2] (LLE) is a similar method in spirit but its details are very different. It doesn't use multidimensional scaling

The basic idea has two steps:

1. Learn a bunch of local approximations to the structure between $x_1, \ldots x_n \in \mathbb{R}^p$
2. Learn a low-dimensional representation $z_1, \ldots z_n \in \mathbb{R}^k$ that best matches these local approximations

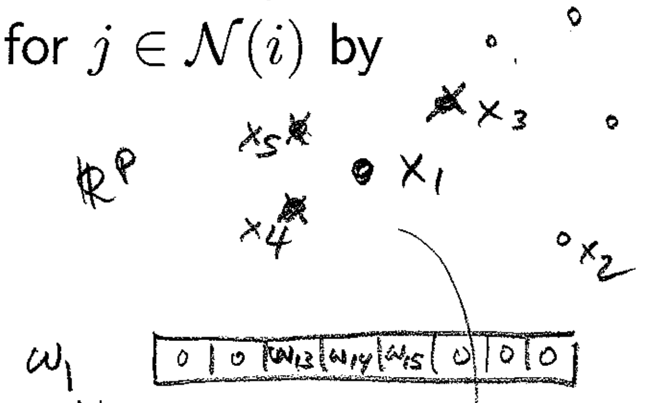
What is meant by such local approximations? We simply try to predict each x_i by a linear function of nearby points x_j (hence the name *local linear* embedding)

[2] Roweis et al. (2000), "Nonlinear dimensionality reduction by locally linear embedding"

Using weights to linearly predict the local structure

For each $x_i \in \mathbb{R}^p$, we first find its m nearest neighbors, and collect their indices as $\mathcal{N}(i)$. Then we build a weight vector $w_i \in \mathbb{R}^n$, setting $w_{ij} = 0$ for $j \notin \mathcal{N}(i)$ and fitting w_{ij} for $j \in \mathcal{N}(i)$ by minimizing

$$\Big\| x_i - \sum_{j \in \mathcal{N}(i)} w_{ij} x_j \Big\|_2^2$$



$w_2 \ldots w_n$

Finally, we take these weights $w_1, \ldots w_n \in \mathbb{R}^n$ and we fit the low-dimensional representation $z_1, \ldots z_n \in \mathbb{R}^k$, by minimizing

$$\sum_{i=1}^{n} \Big\| z_i - \sum_{j=1}^{n} w_{ij} z_j \Big\|_2^2$$

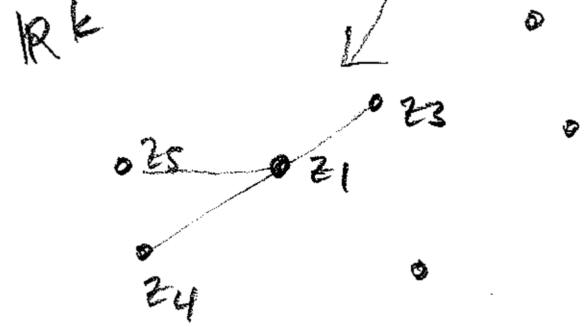
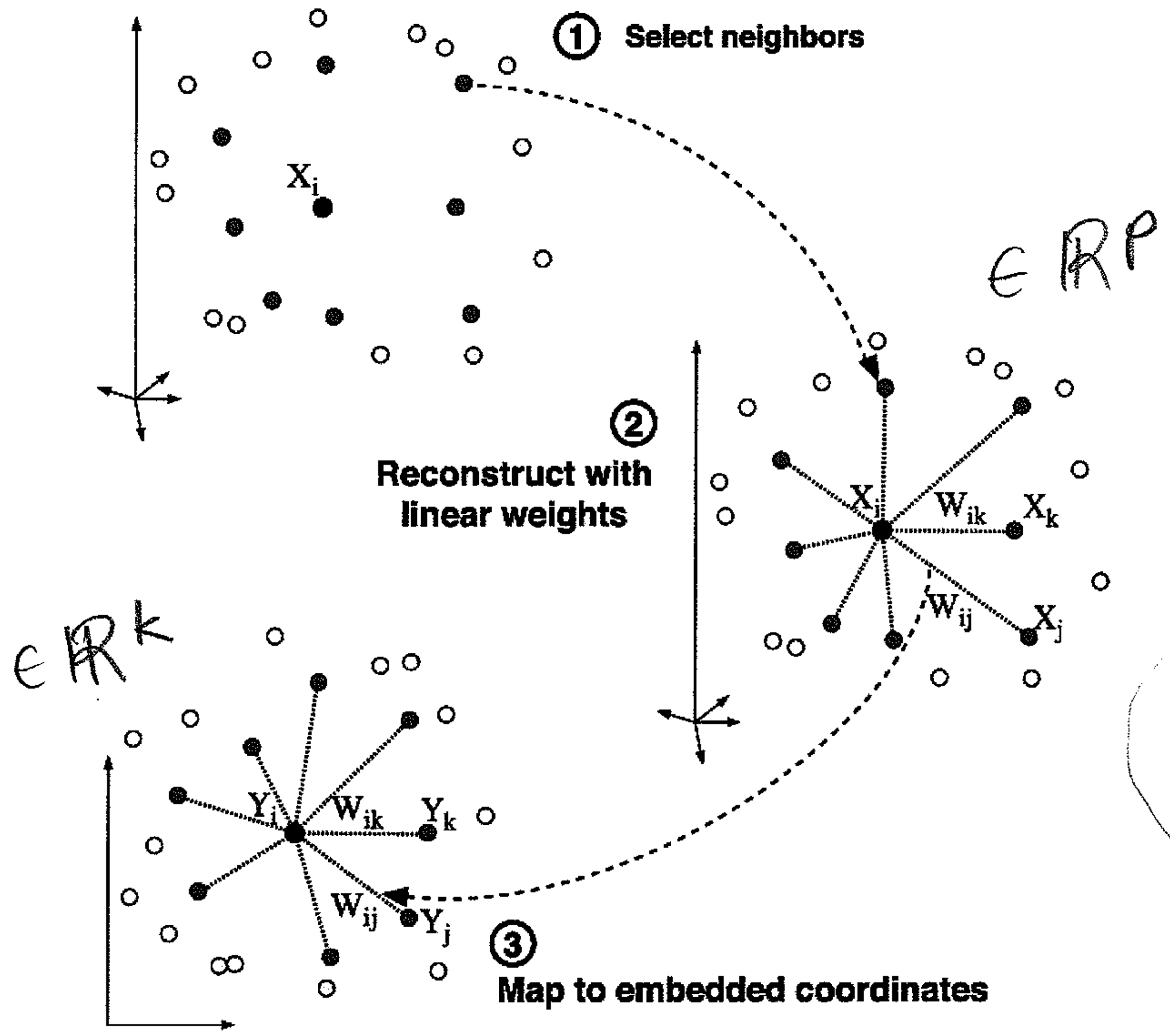
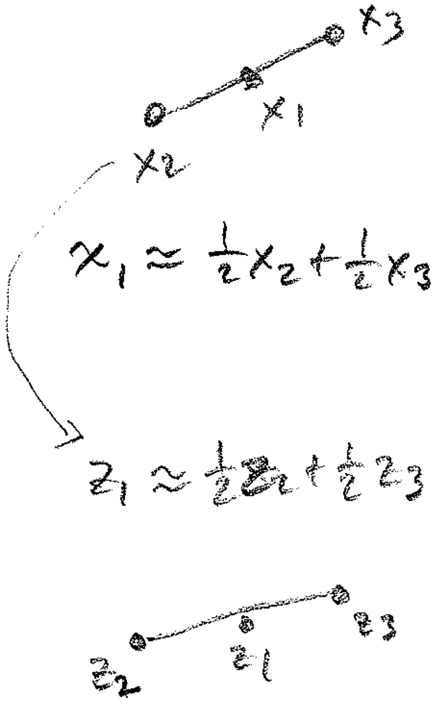


Illustration of local linear embedding





$x_1 \approx \frac{1}{2}x_2 + \frac{1}{2}x_3$

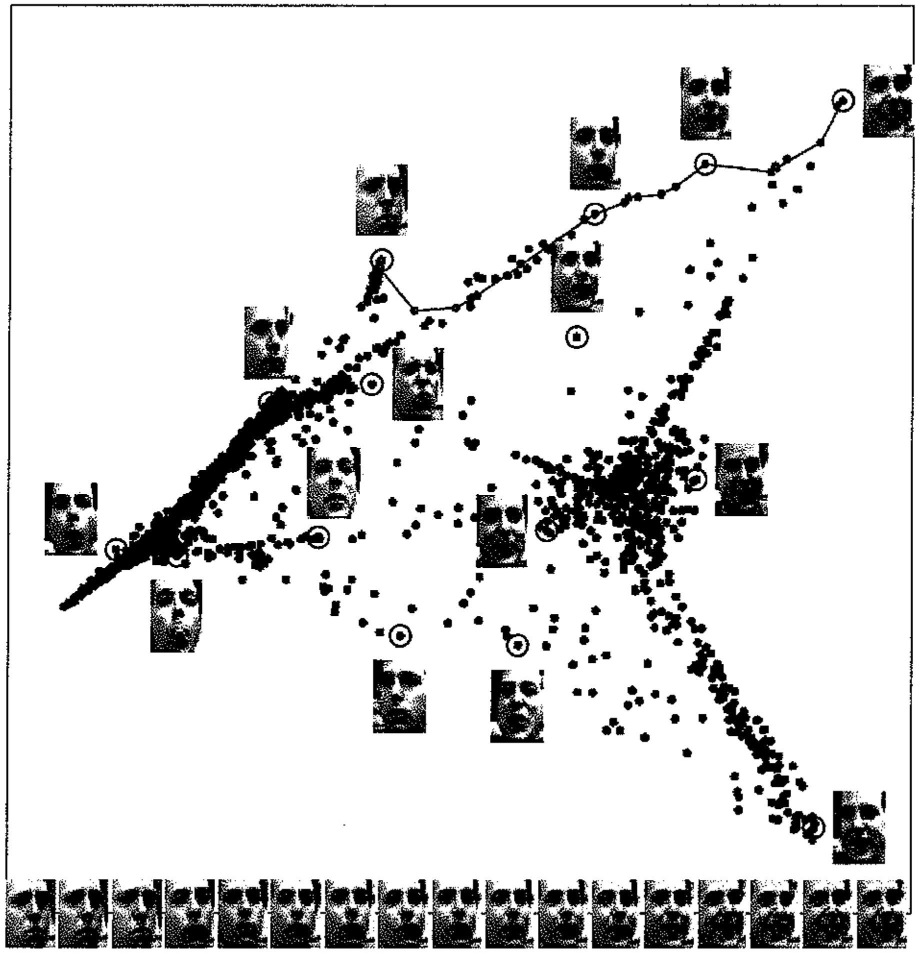
$z_1 \approx \frac{1}{2}z_2 + \frac{1}{2}z_3$

(From Roweis et al. (2000))

we have to choose m

Example: facial expressions

Example: $n = 2000$ images of faces and $p = 560$, i.e., each image is 20×28 pixels. Local linear embedding applied with $m = 12$ nearest neighbors, reduced to $k = 2$ dimensions:



(From Roweis et al. (2000))

Multidimensional scaling and Isomap in R

Recall that multidimensional scaling can be computed directly from the distances matrix Δ, and an eigendecomposition. E.g.,

```
A = -1/2*Delta^2
B = scale(A-rowMeans(A), center=T, scale=F)
e = eigen(B)
Z = e$vectors %*% diag(sqrt(e$values))
```

Isomap is implemented by the function `isomap` in the package `vegan`. E.g.,

```
k = 2 # Number of dimensions in reduced representation
m = 8 # Number of nearest neighbors to consider
iso = isomap(Delta, ndim=k, k=m)
```

Recap: nonlinear dimension reduction

We learned (classical) multidimensional scaling which, given a distance matrix $\Delta \in \mathbb{R}^{n\times n}$ between unknown points $x_1, \ldots x_n$, computes a low-dimensional representation of these points $z_1, \ldots z_n \in \mathbb{R}^k$

If $x_1, \ldots x_n \in \mathbb{R}^p$ and $\Delta_{ij} = \|x_i - x_j\|_2$, then the low-dimensional representation returned by multidimensional scaling exactly corresponds to the first k principal component scores of $X \in \mathbb{R}^{n\times p}$ (whose ith row is x_i)

But multidimensional scaling can be applied to more general distance measures Δ_{ij} than just Euclidean distance, therefore achieving a nonlinear dimension reduction. Two such examples are tangent distance and the graph distance defined by Isomap

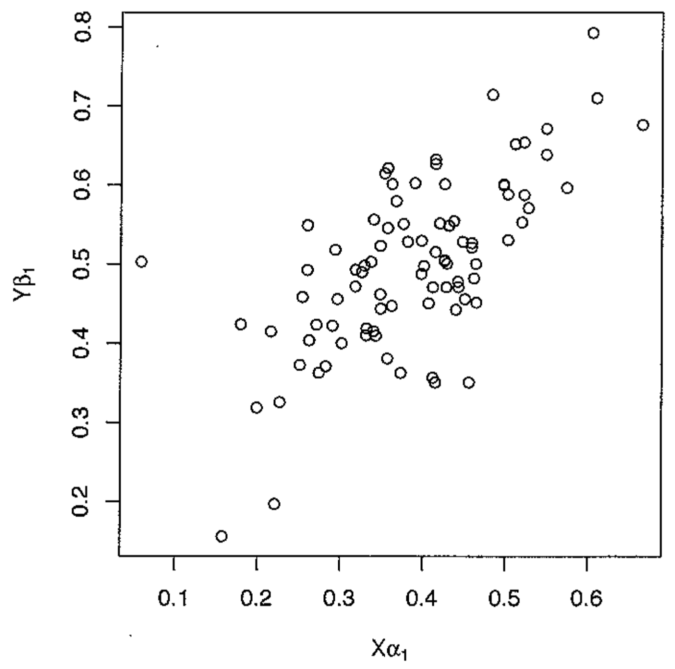
Local linear embedding is a different nonlinear technique in which we locally approximate the structure, and find a low-dimensional representation that fits these approximations

Next time: canonical correlation analysis

Finding pairs of directions that explain covariance between two sets of variables

$$\alpha_1 = \begin{pmatrix} 2.770 \\ 5.517 \end{pmatrix} \begin{array}{c} \text{mechanics} \\ \text{vectors} \end{array}, \quad \beta_1 = \begin{pmatrix} 8.782 \\ 0.860 \\ 0.370 \end{pmatrix} \begin{array}{c} \text{algebra} \\ \text{analysis} \\ \text{statistics} \end{array},$$

$$\rho_1 = 0.663$$


$Y\beta_1$
$X\alpha_1$