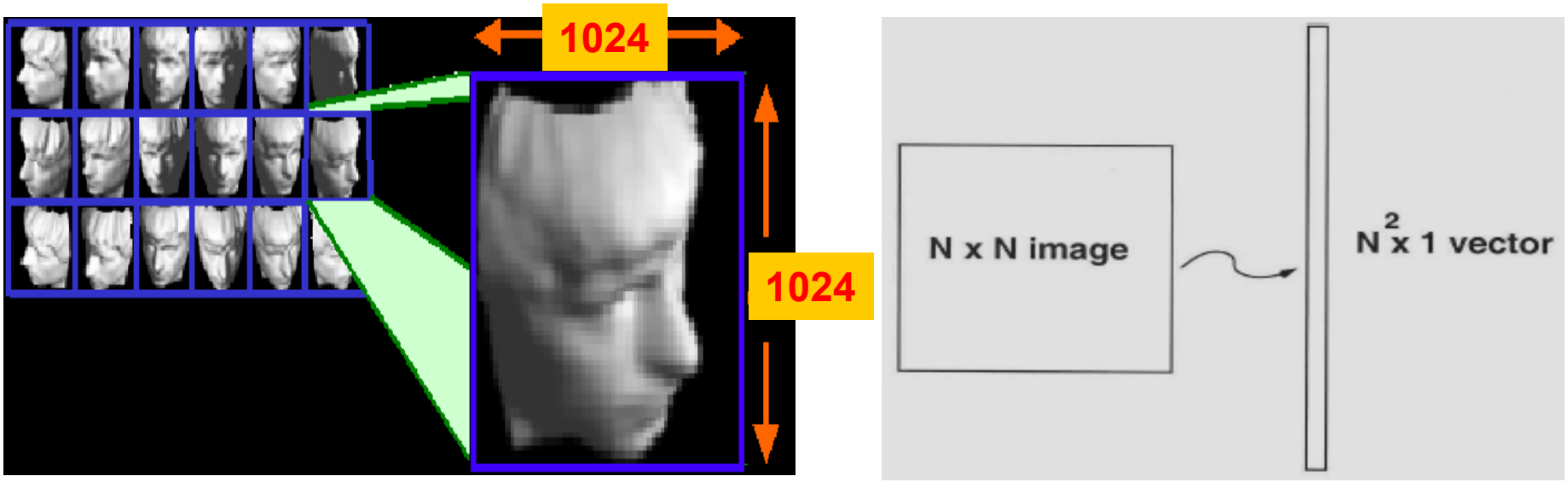
Dimensionality Reduction

Evgeny Burnaev
Skoltech

Dimensionality Reduction Problem

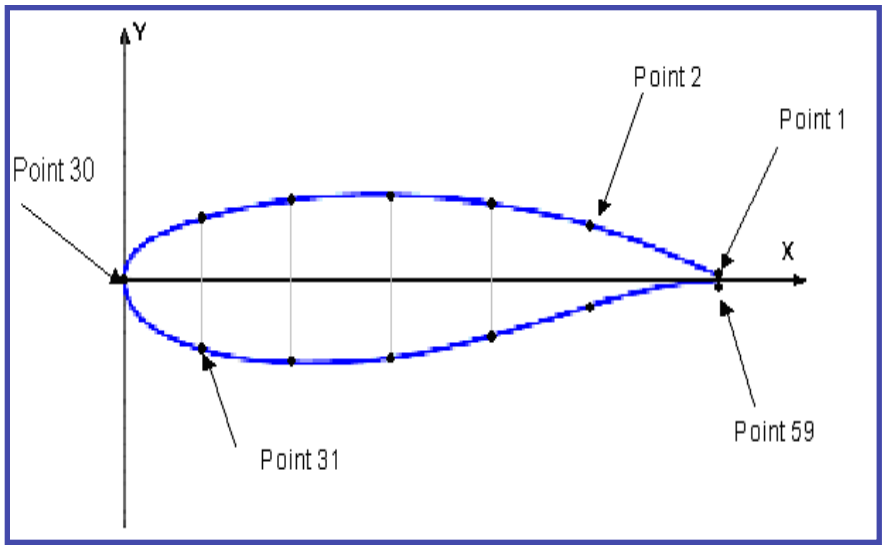
- object **O** is described by a **p**-dimensional vector **X(O)** $\in R^p$. Components of **X(O)** are features of **O**

Example 1 (human face): grey-scale pixel-wise representation



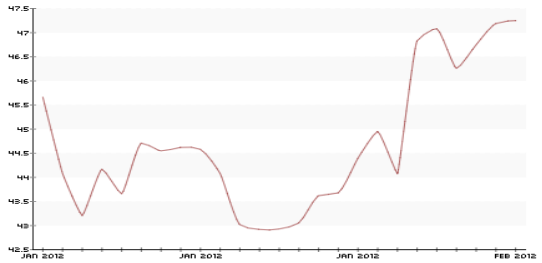
Face is represented by 1024×1024 pixels – dimension p = 10^{20} ~ 1 000 000

Example 2 (wing airfoil description).

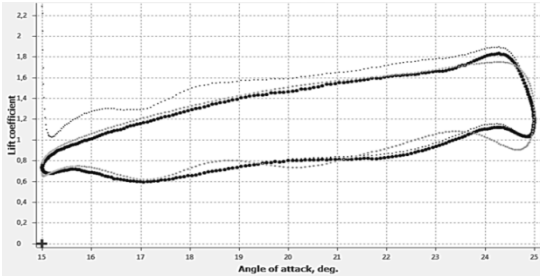


Airfoil **O** $\rightarrow$ **X(O)** = $(x_1, x_2, \dots, x_p)^T$ - ordinates of upper and lower contours, **p ~ 50 ÷ 200**

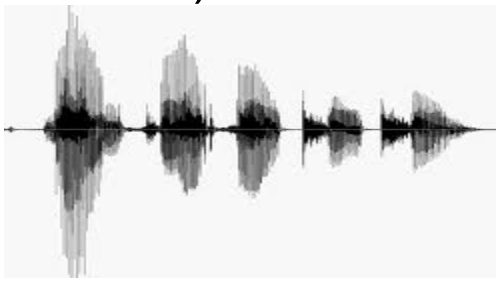
Example 3 (plots of dependences, multidimensional time-series)



Electricity price curve



Lift force vs. AoA curve

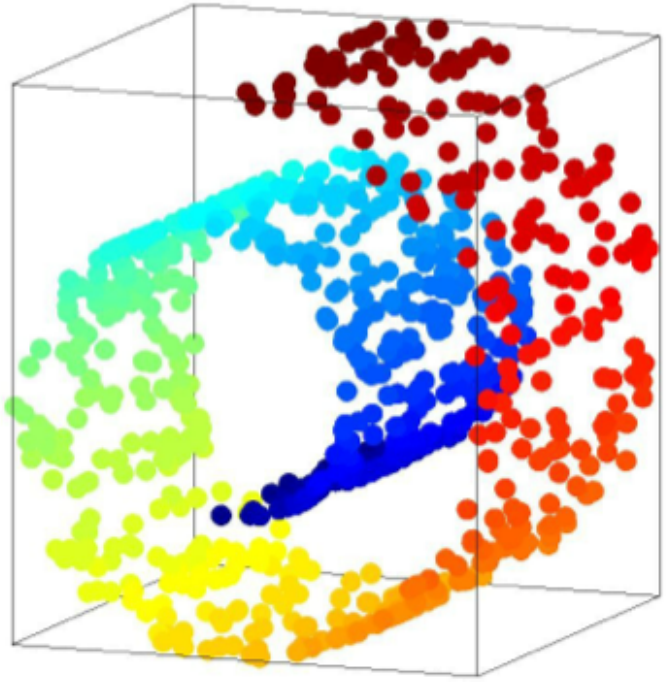


Speech recognition

Curve f(x) is described by the vector $f = (f_1, f_2, \dots, f_p)^T \in R^p$, $f_j = f(t_j)$; time-series segment $(x_1, x_2, \dots, x_p)$ is considered as a **p-**dimensional vector

Example 4 (MNIST)

Example 5 Famous Toy Problem
"Swiss Roll"



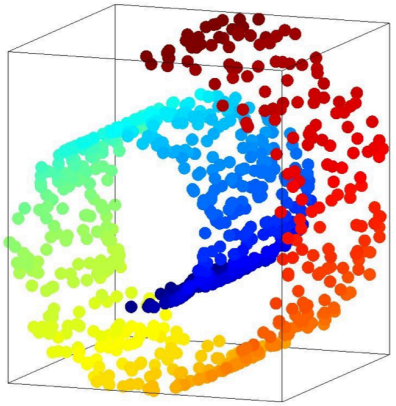
- High dimensionality **p** of **X(O)** is critical for efficient learning
- We can visualize only in 2D/3D

- Dimension Reduction = construct reduced dimension representation **y(O)** $\in$ **R^q**, **q << p**, of **O** without "significant loss of information"

- DR for
  - ✓ Visualization
  - ✓ Data compression
  - ✓ "curse of dimensionality"
  - ✓ De-noising
  - ✓ Reasonable distance metrics

X → X' S.T.
dim(X') << dim(X)

uncovers the intrinsic dimensionality (invertible)



compressed description **y(O)** is constructed for a detailed description **X(O)** using a learning sample

$$\mathbf{X}_n = \{X_i = X(O_i),\ i = 1, 2, \ldots, n\} = \{X_1, X_2, \ldots, X_n\}$$

of features for objects $O_1, O_2, \ldots, O_n$,

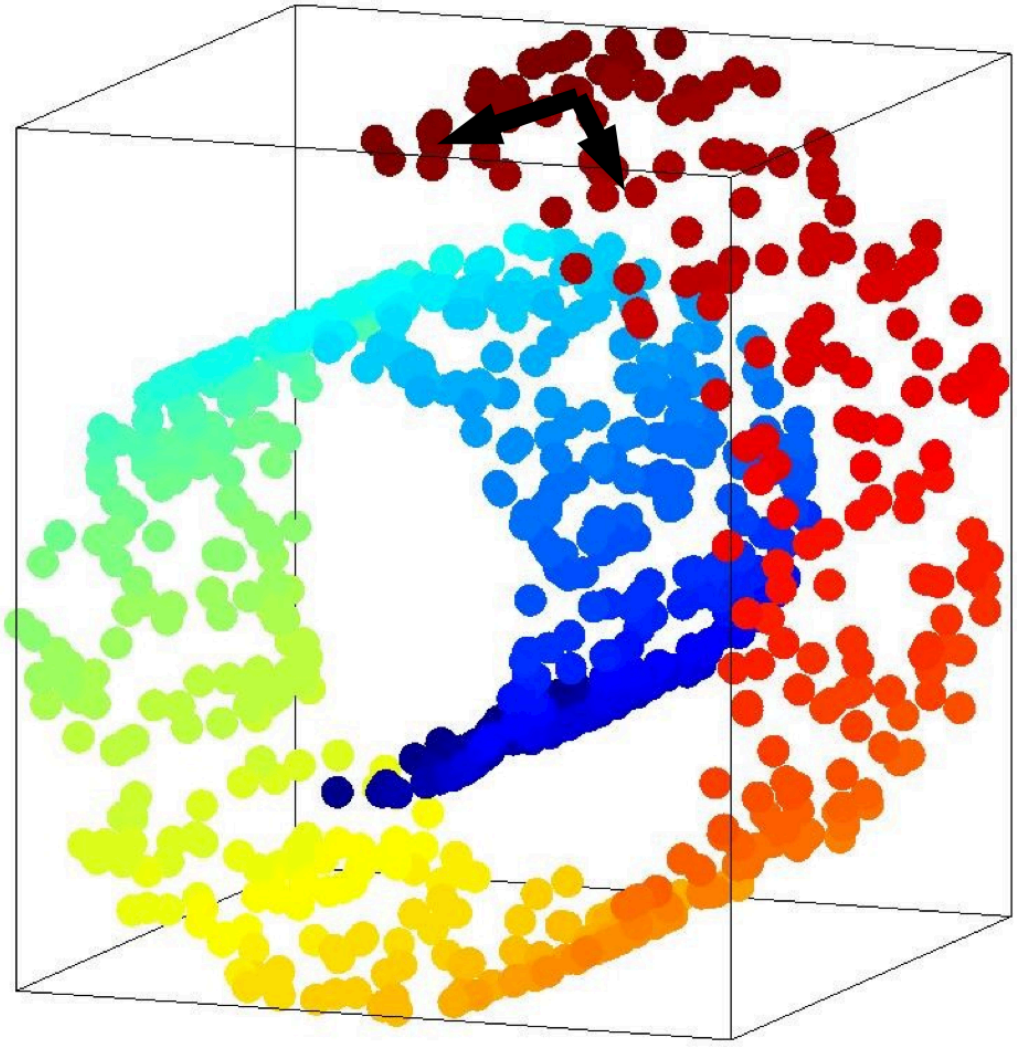
without using physical information about objects

appearance variation

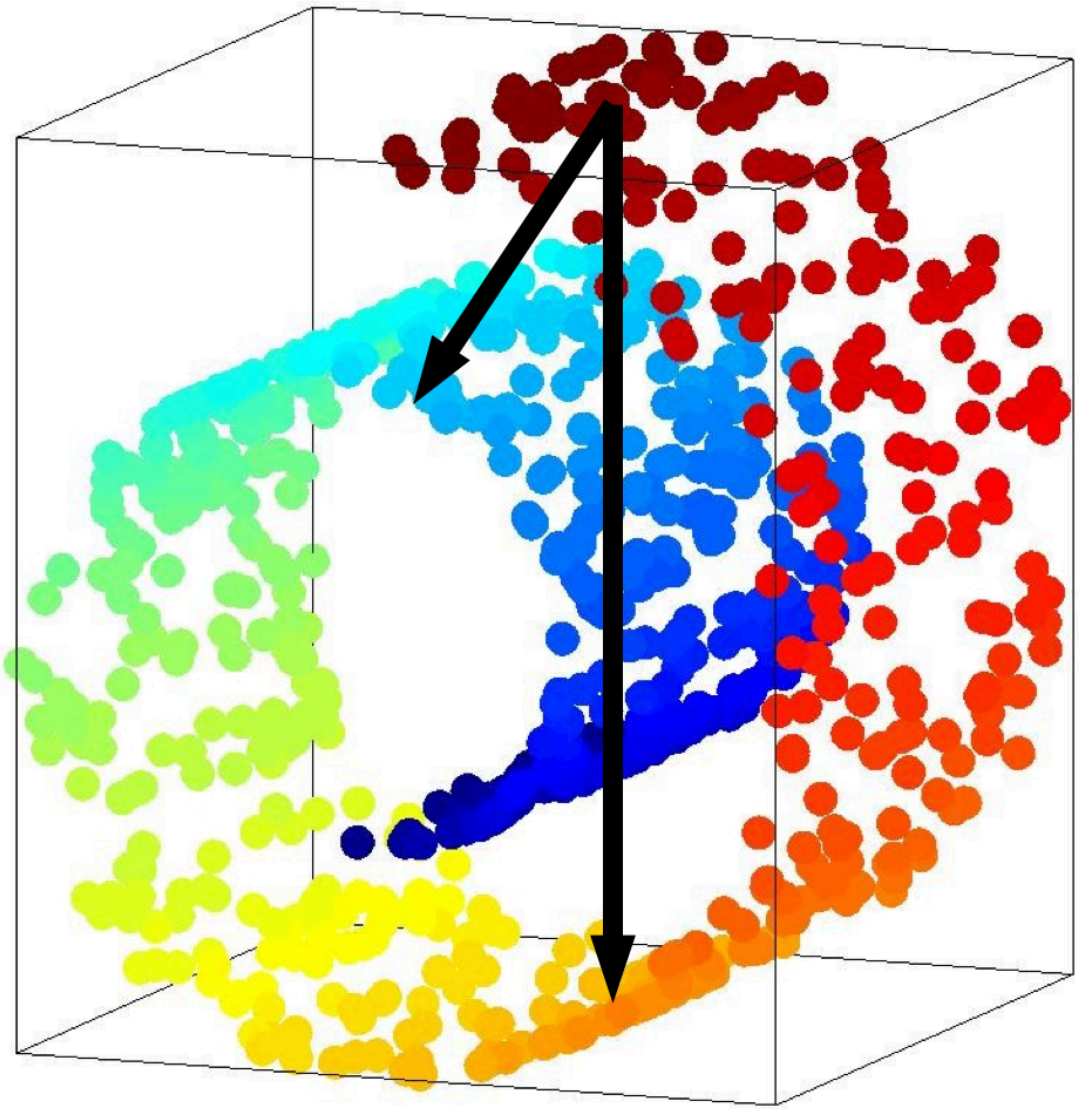
What is “Reasonable distance metrics”?

deformation

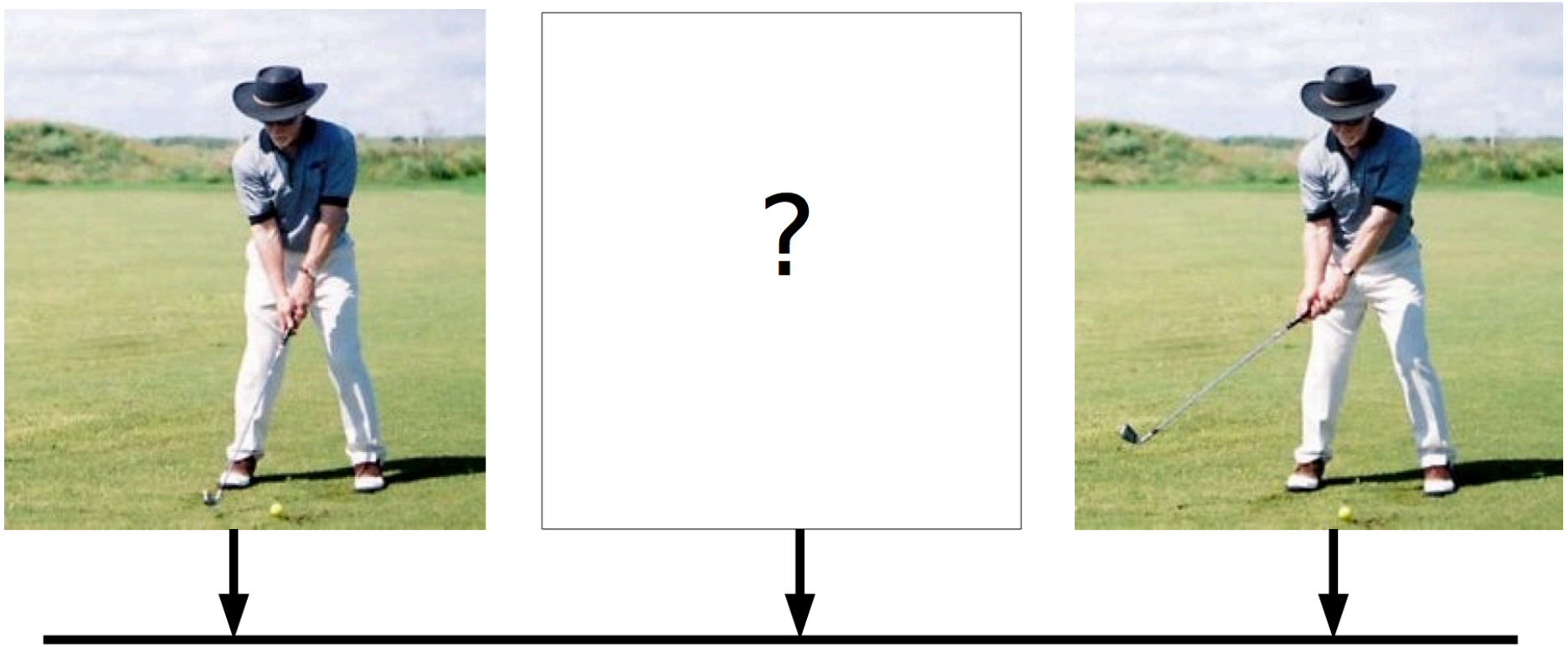
What is “Reasonable distance metrics”?



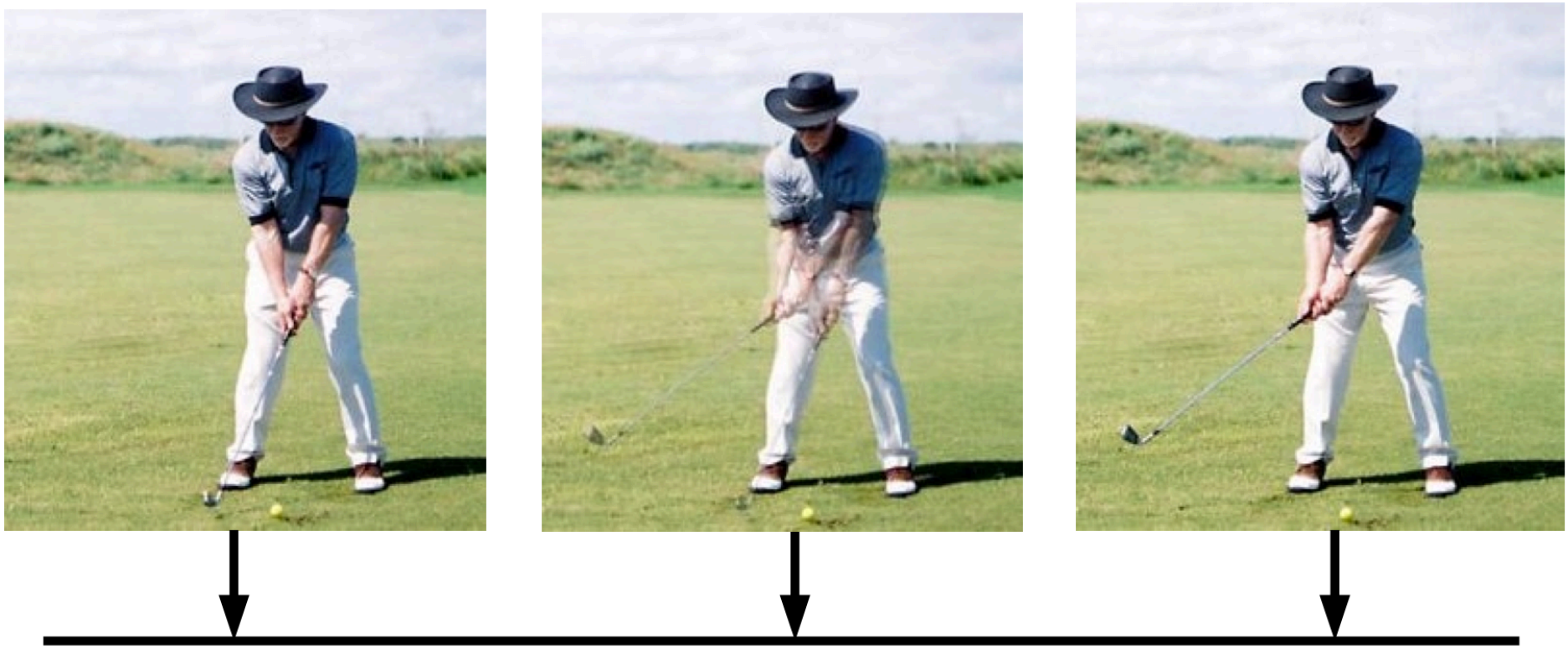
What is “Reasonable distance metrics”?



reasonable distance metrics

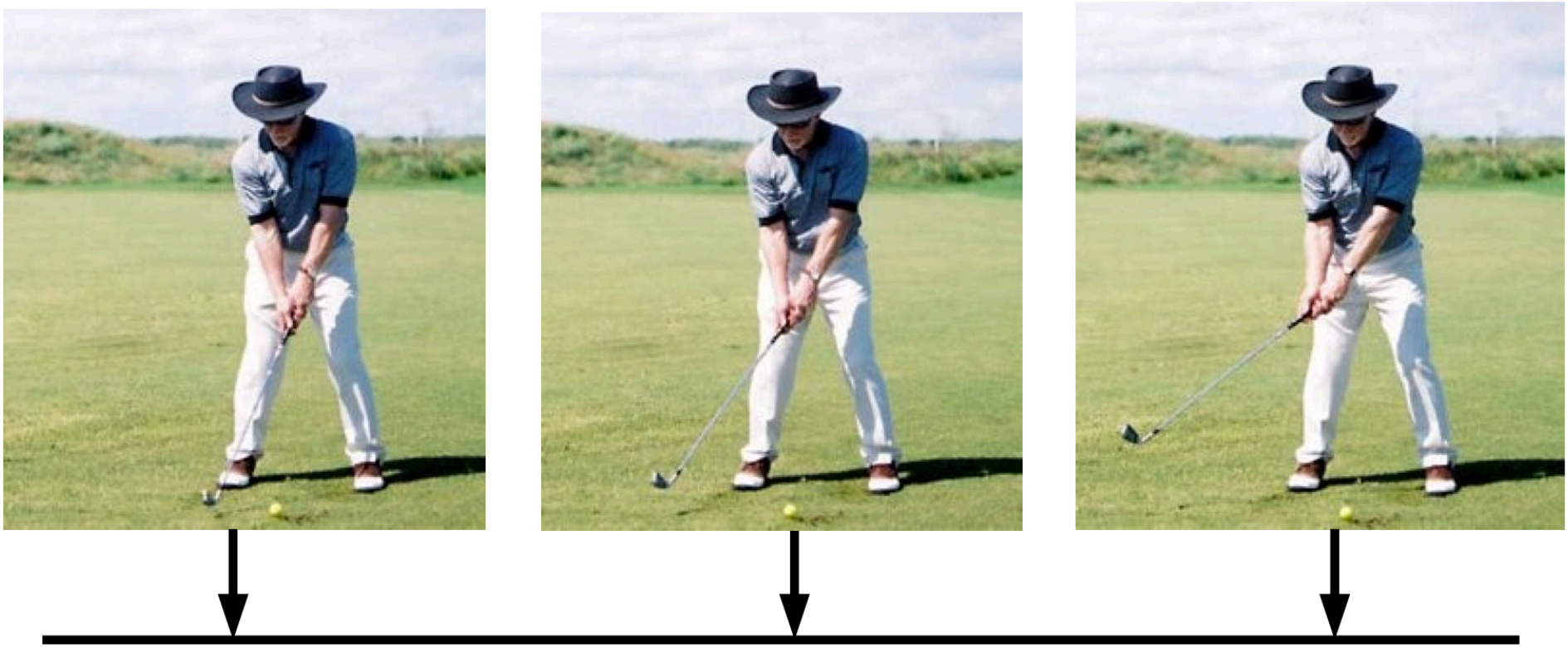


reasonable distance metrics



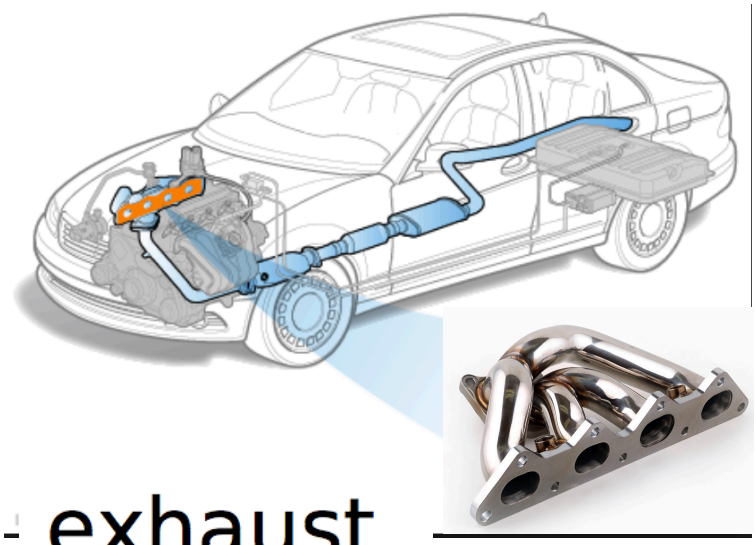
linear interpolation

reasonable distance metrics

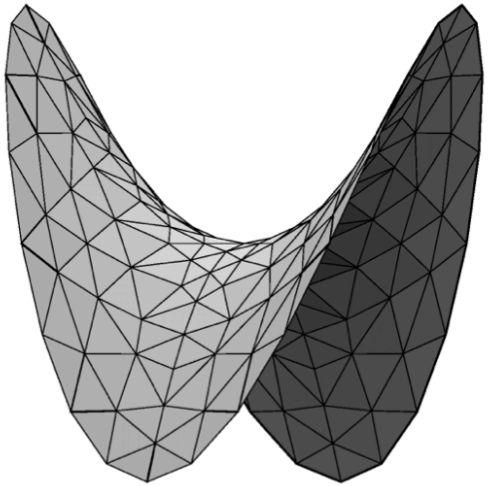


manifold interpolation

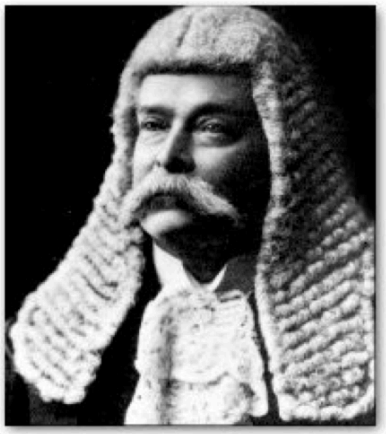
Types of manifold



- exhaust manifold

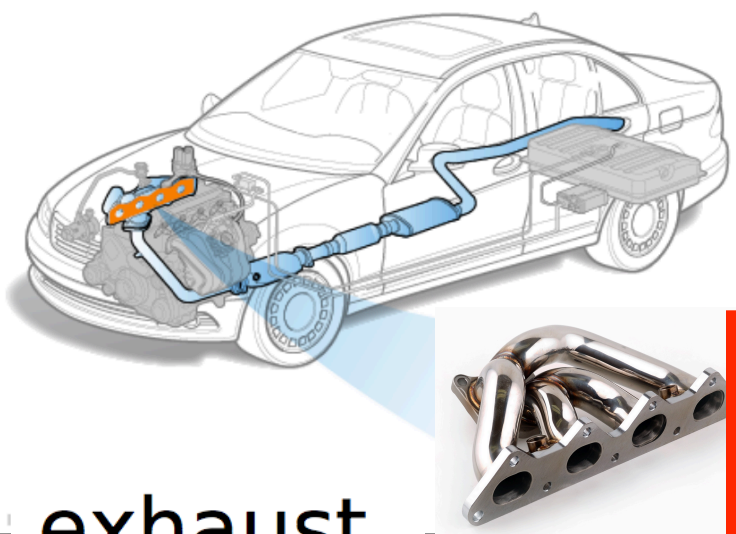


low-D surface embedded in high-D space

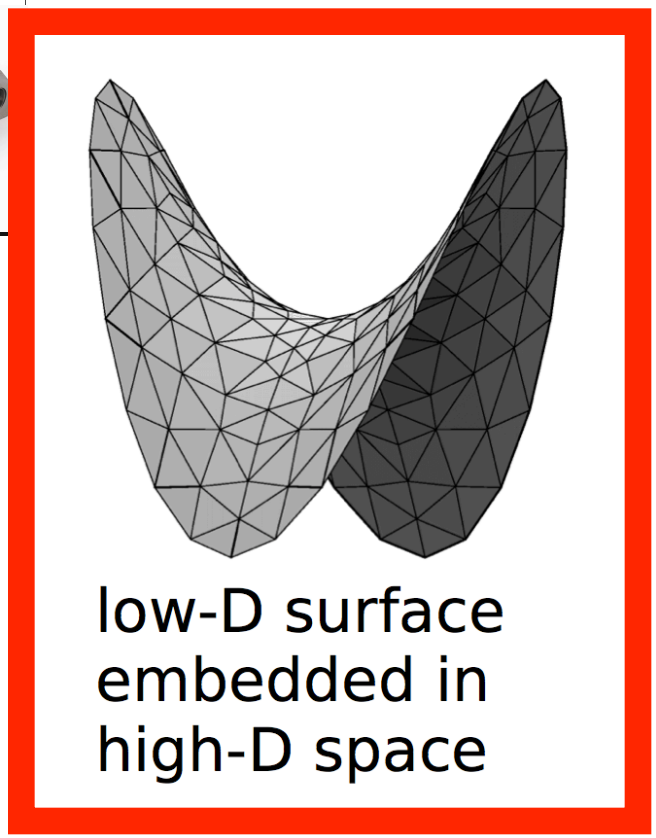


Sir Walter Synnot Manifold 1849-1928

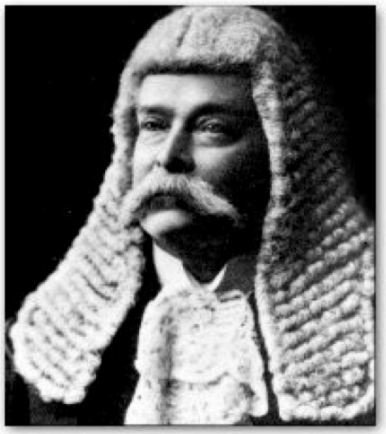
Types of manifold



- exhaust manifold



low-D surface embedded in high-D space



Sir Walter Synnot Manifold 1849-1928

Dimension Reduction as an Embedding Problem

Embedding Problem. Using a sample

$$\mathbf{X}_n = \{X_1, X_2, \dots, X_n\} \subset R^p$$

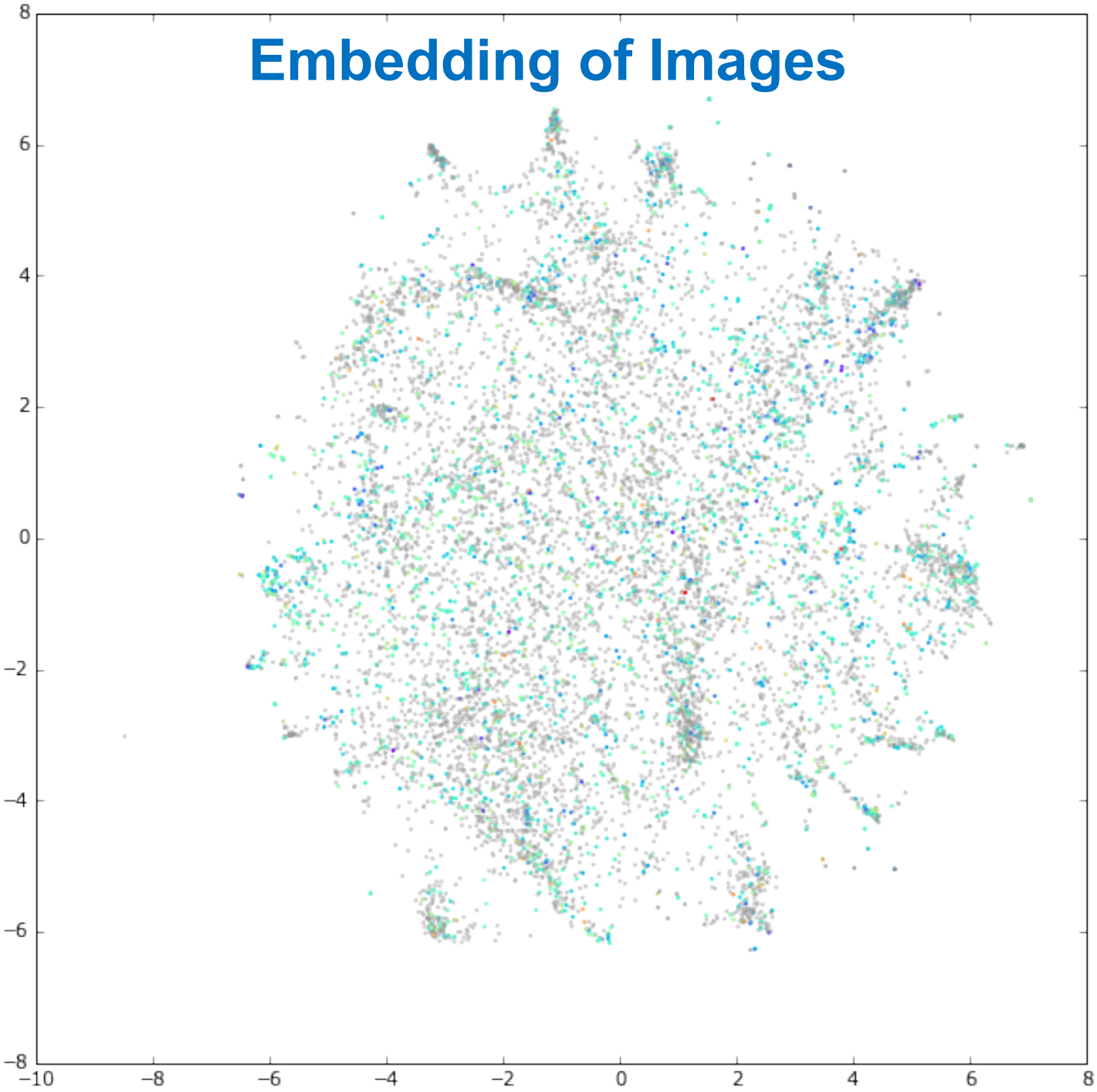
construct an embedding

$$h: \mathbf{X}_n \to \mathbf{Y}_n = h(\mathbf{X}_n) = \{y_1, y_2, \dots, y_n\} \subset R^q$$

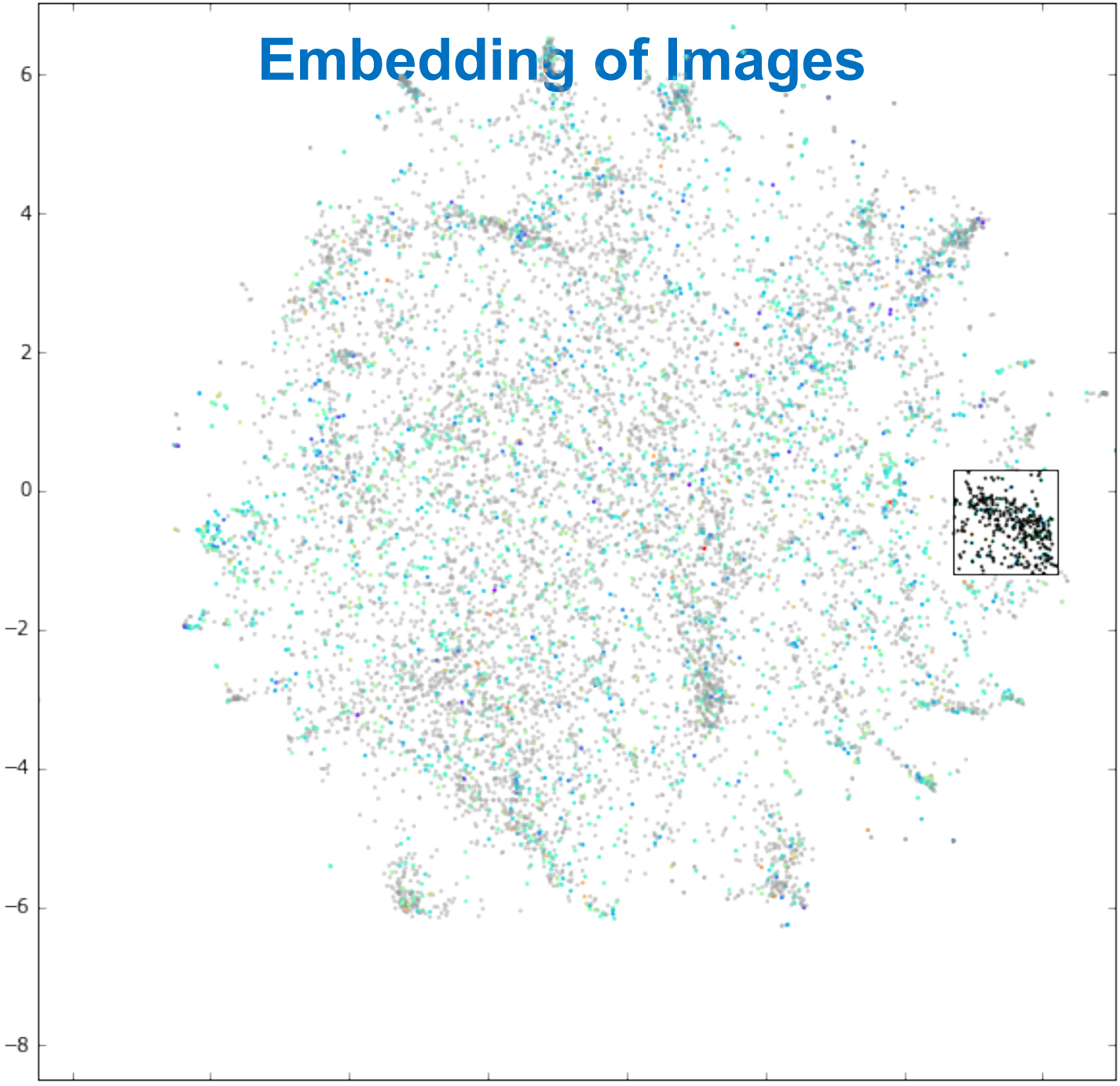
of sample points $\mathbf{X}_n$ в R^q, $\mathbf{q < p}$, such that the set $\mathbf{Y}_n$ **«consistently represents»** the set $\mathbf{X}_n$

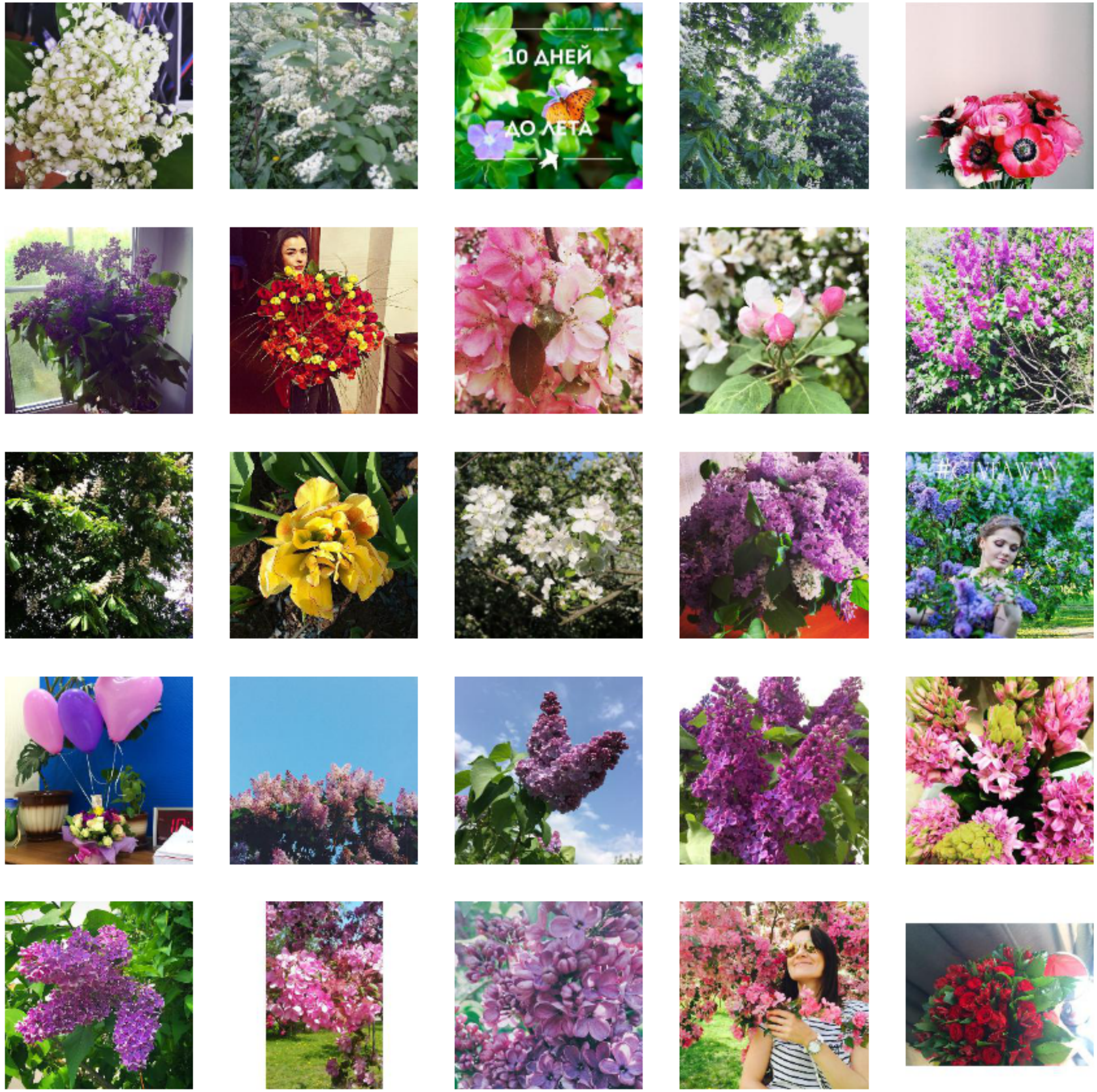
(e.g., the set $\mathbf{Y}_n$ should preserve some geometric structure of $\mathbf{X}_n$, etc.)

Embedding of Images

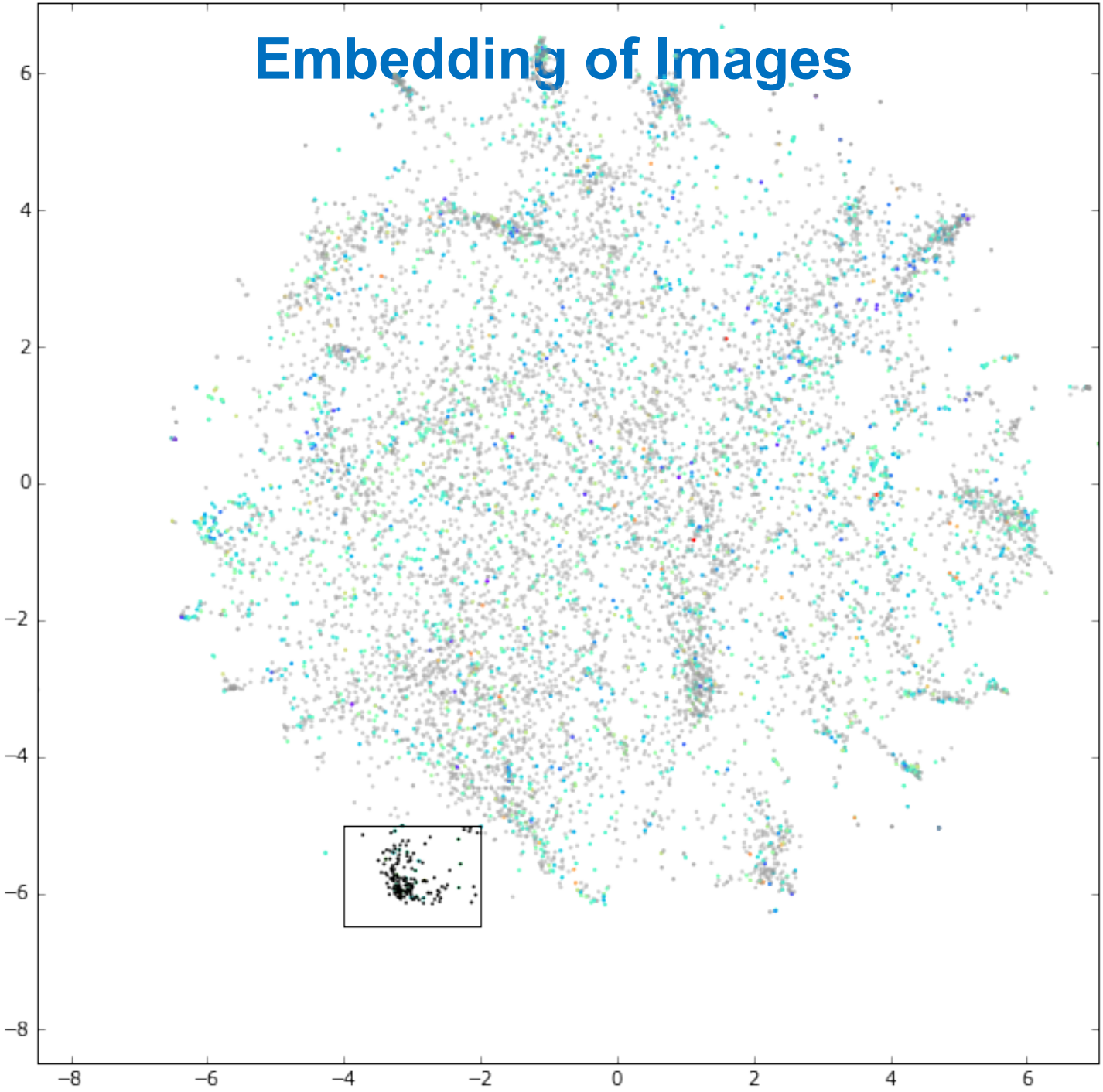


Embedding of Images




10 ДНЕЙ
ДО ЛЕТА

Embedding of Images









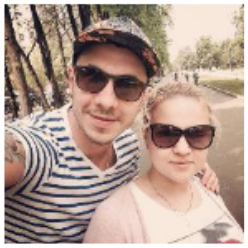







10 м

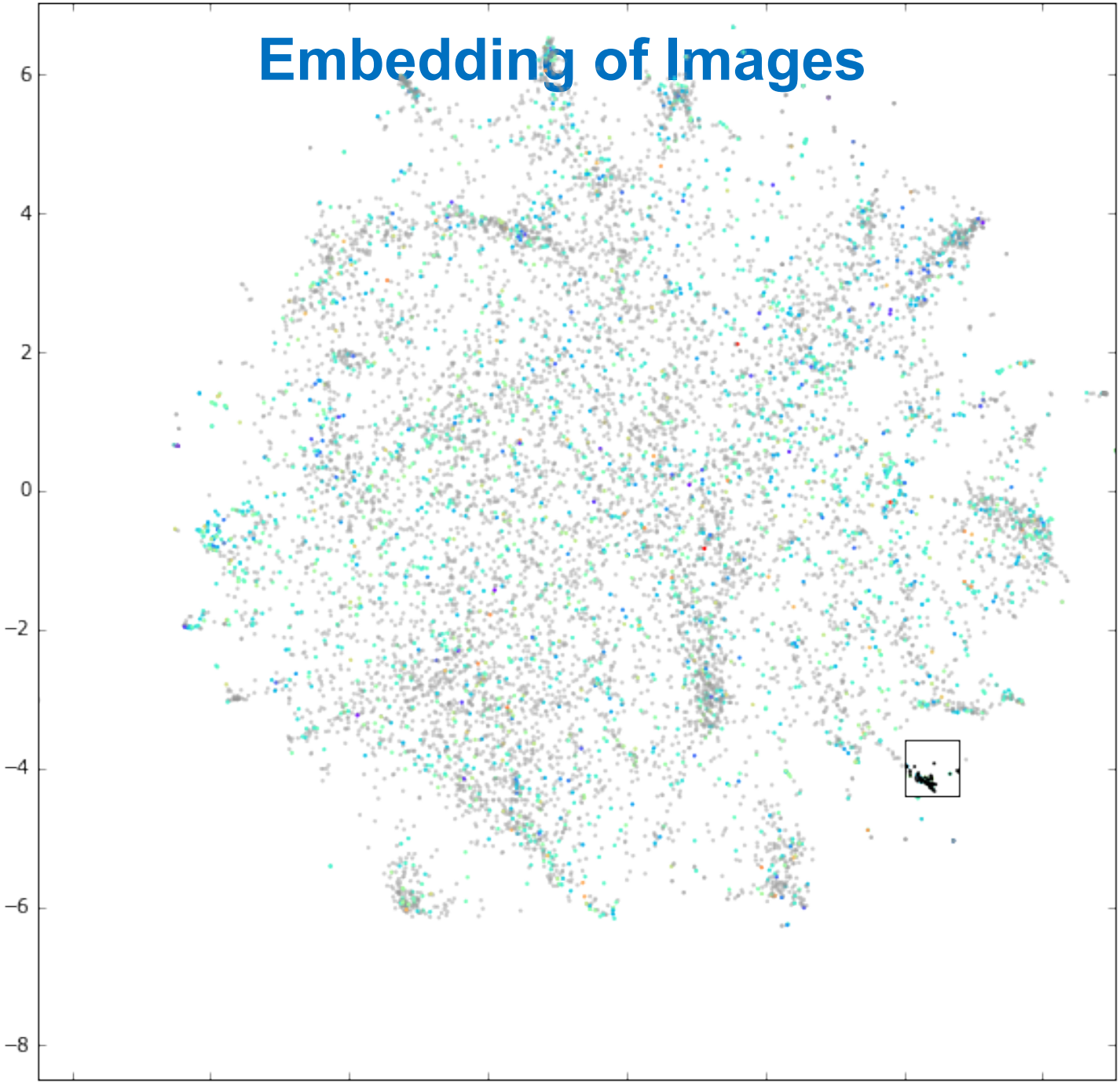


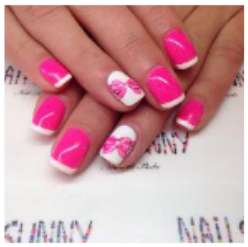




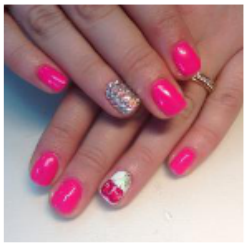

Embedding of Images

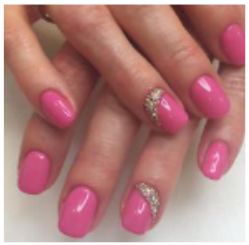


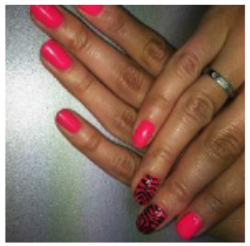
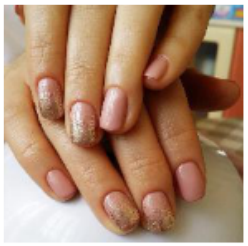




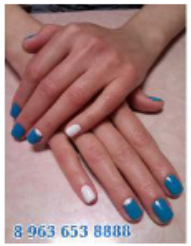






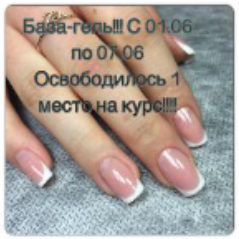
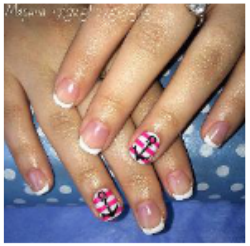
8 963 653 8888






8 963 653 8888







База-гель!!! С 01.06
по 07.06
Освободилось 1
место на курс!!!



Typical scheme:

• construct some cost function $L(y_1, y_2, \ldots, y_n)$. E.g. for Multi Dimensional Scaling, MDS

$$L(y_1, y_2, \ldots, y_n) = \sum_{i,j} \left(\rho(X_i, X_j) - \| y_i - y_j \| \right)^2$$

• optimize cost function

• visually analyze results (Swiss Roll, Spiral, …)

Popular methods to solve embedding problem:

- Pursuit Projection; Principal Component Analysis
- Locally Linear Embedding, LLE; Conformal Eigenmaps
- Laplacian Eigenmaps LE
- Hessian Eigenmaps, HE
- ISOmetric MAPing, ISOMAP; Landmark ISOMAP
- Kernel PCA, KPCA
- Local Tangent Space Alignment, LTSA

Extended Embedding Problem

Using a sample

$$\mathbf{X}_n = \{X_1, X_2, \ldots, X_n\} \subset R^p$$

construct an embedding transformation

$$h: \mathbf{X} \subset R^p \rightarrow \mathbf{Y} = h(\mathbf{X}) \subset R^q,$$

both for sample points $\mathbf{X}_n$, and for new (out-of-sample) points

$$X_{new} \in \mathbf{X} / \mathbf{X}_n$$

without solving embedding problem for $\mathbf{X}_n \cup \{X\}$ anew

Example – face recognition

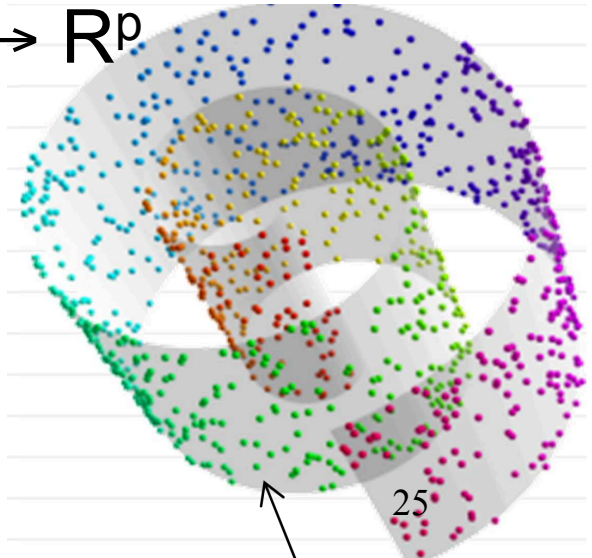
For extended embedding problem we need a Data Model **X:** description of **X**, and of a mechanism, which generates points $\mathbf{X}_n$ and new points X_{new} from **X**

Usually we assume that real data is well approximated by some manifold, i.e.

Manifold Data Model:

$$\mathbf{X} = \{X = f(b) \in R^p : b \in \mathbf{B} \subset R^q\} \subset R^p$$

- open set **B** (inner coordinate space)
- smooth bijective transformation $f: \mathbf{B} \rightarrow R^p$



Manifold Learning Problem!!!

Full Dimension Reduction problem

Using a sample

$$\mathbf{X}_n = \{X_1, X_2, \ldots, X_n\} \subset R^p$$

construct embedding transformation:

$$h: \mathbf{X} \subset R^p \rightarrow \mathbf{Y} = h(\mathbf{X}) \subset R^q$$

and **Reconstruction transformation**

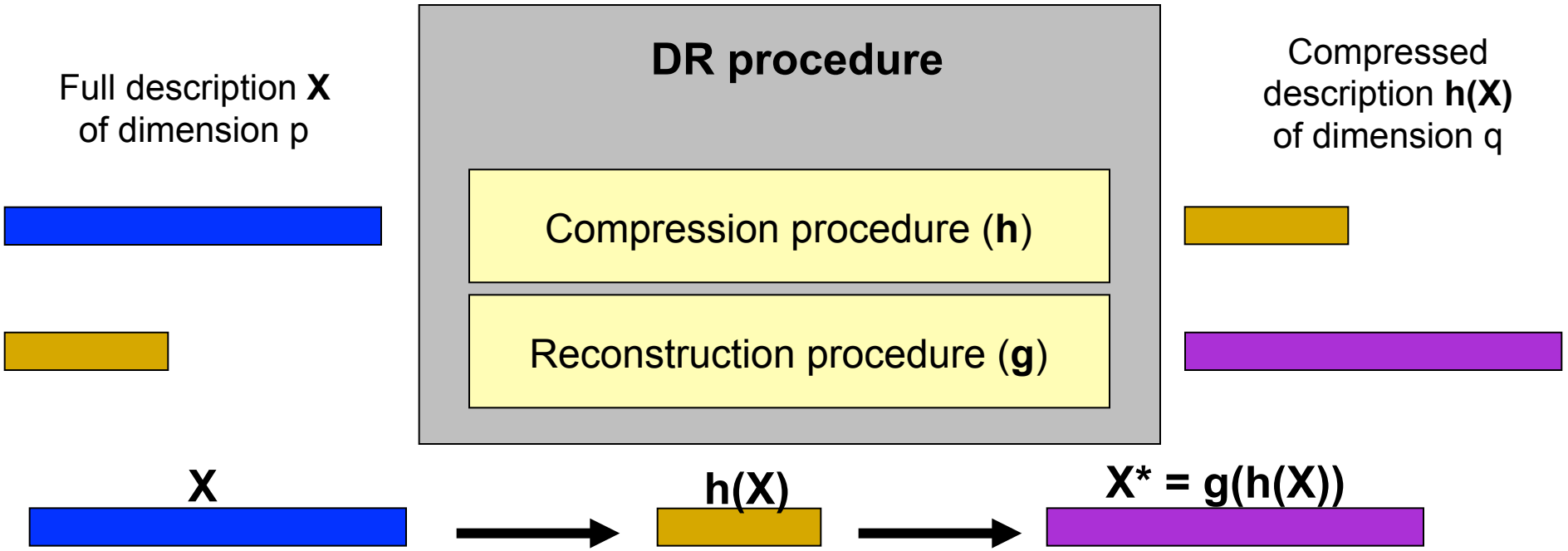
$$g: \mathbf{Y} \subset R^q \rightarrow \mathbf{X} \subset R^p$$

such that

$$g(h(X)) \approx X \quad \text{for } \mathbf{all}\ X \in \mathbf{X}$$

Example – dimension reduction of airfoils

Dimension Reduction



Requirements:

minimal dimension **q = Dim h(X)** of compressed description **h(X)**, providing required proximity between **X* = g(h(X))**) and **X**

or:

maximal proximity between **X** and **X* = g(h(X))** for a fixed dimension **q = Dim h(X)** of compressed description

Why do we need reconstruction?

Optimize $\upsilon(X)$ w.r.t. p-dimensional vector $X \in \mathbf{X} \subset R^p$. If $\theta = (h, g)$ is such that $X \approx g(h(X))$, and

$$\upsilon(X) \approx \upsilon(g(h(X))),$$

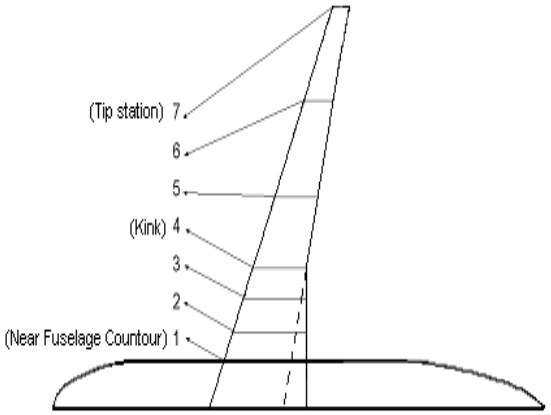
then instead we can optimize the functional

$$\upsilon_\theta(y) = \upsilon(g(y))$$

w.r.t. q-dimensional vector y

For an optimal y* we have to recover optimal $X^* = g(y^*)$

Example – aircraft wing shape optimization



- airfoil description dimension p = 60
- compressed description dimension q = 6
- dimension for the initial problem

$$X = (X_1, X_2, \dots, X_7) \in R^{420}$$

- dimension of the reduced problem

$$y = (y_1, y_2, \dots, y_7) \in R^{42}$$

- For optimal y we have to recover a real wing description

Principal Component Analysis, PCA
(K. Pearson, 1899)

Problem: find in R^p an affine subspace

$$L(q) = \left\{ x \in R^p : x = x_0 + \sum_{j=1}^{q} y_j \times e_j,\ y_1, y_2, \ \ldots, y_q \in R^1 \right\}$$

of dimension q < p, **which best approximates the set of points**

$$\mathbf{X}_n = \{X_i,\ i = 1, 2, \ldots, n\} \subset R^p.$$

in PCA: "the best" = minimize w.r.t. $\mathbf{x_0}$, $\mathbf{\{e_1, e_2, \ldots, e_q\} \subset R^p}$

$$\frac{1}{n} \sum_{j=1}^{n} \left\| X_j - P_{L(q)} X_j \right\|^2$$

$$\Pr_{L(q)}(X) = x_0 + \sum_{j=1}^{q} y_j(X) \times e_j, \qquad y_j(X) = (X - x_0, e_j)$$

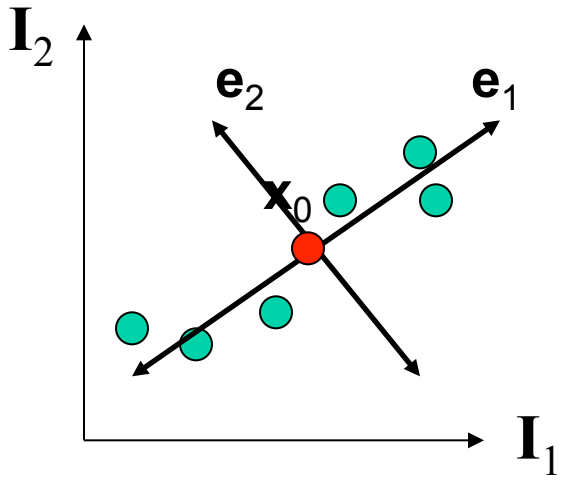
x_{mean} – empirical mean of $\{X_i, i = 1, 2, \dots, n\}$,

$\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_p\}$ – eigenvalues of an empirical covariance (p×p)-matrix

$$\Sigma = \frac{1}{n}\sum_{j=1}^{n}(X_j - x_{mean}) \times (X_j - x_{mean})^T$$

providing orthonormal basis in R^p

p = 2
q = 1



$L(1) = \{X \in R^2: X = x_0 + y \times \mathbf{e}_1, y \in R^1\}$

Solution: $x_0 = x_{mean}$, $L(q) = x_0 \oplus \text{Span}(\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_q)$,

where orthonormal eigenvectors $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_q\}$ of Σ

correspond to q highest eigenvalues of this matrix

PCA solves E-Problem, EE-Problem and FULL DR problem:

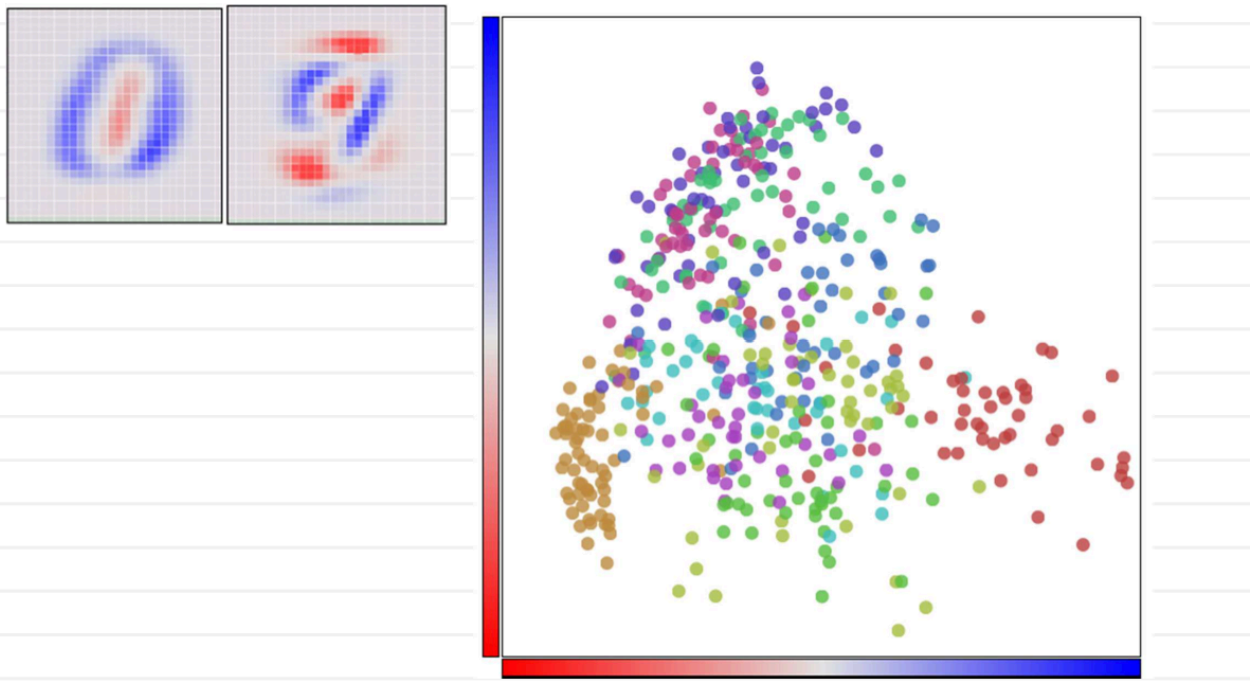
embedding procedure:

$$X \in R^p \rightarrow h(X) = (y_1(X), y_2(X), \ldots, y_q(X))^T \in R^q$$

reconstruction procedure:

$$\mathbf{y} = (y_1, y_2, \ldots, y_q)^T \in R^q \rightarrow g(y) = x_0 + y_1 \times \mathbf{e}_1 + y_2 \times \mathbf{e}_2 + \ldots + y_q \times \mathbf{e}_q$$

PCA on MNIST



http://colah.github.io/posts/2014-10-Visualizing-MNIST/

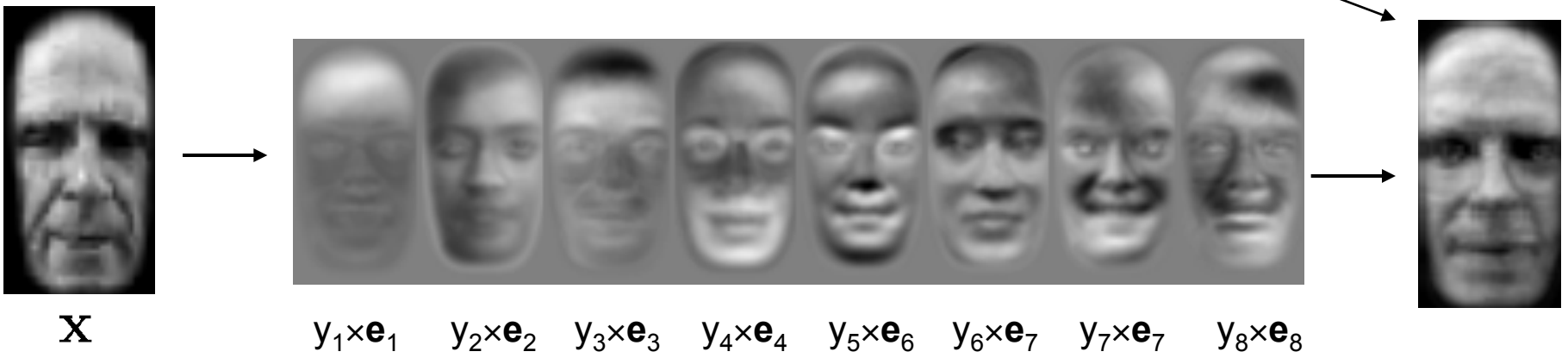
PCA for Face Recognition

- Eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p \in R^p$ in a space of faces: $p \sim 10^6$

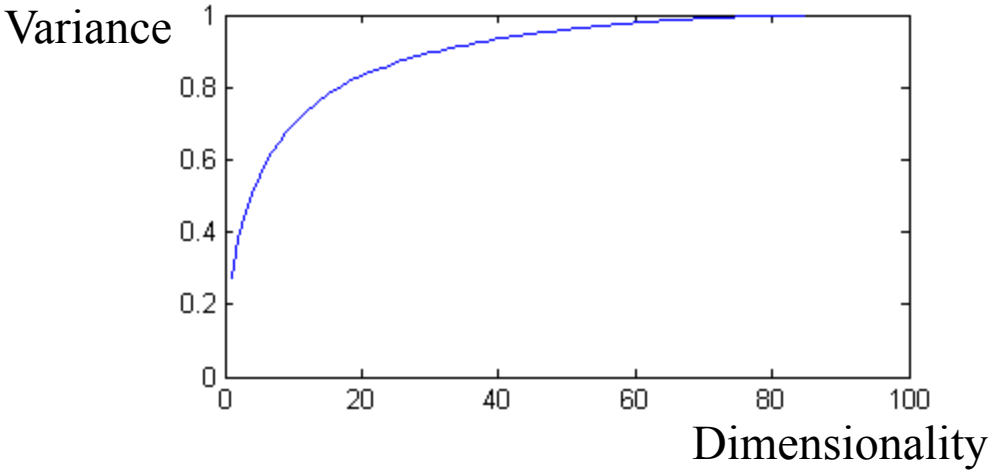
$$X \rightarrow (\underbrace{(X - x_0, \mathbf{e}_1)}_{y_1}, \underbrace{(X - x_0, \mathbf{e}_2)}_{y_2}, \ldots, \underbrace{(X - x_0, \mathbf{e}_p)}_{y_p})^T \in R^p$$

- We select first q, $q \sim 10^2$ eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q \in R^q$

$$X^* \approx x_0 + y_1 \times \mathbf{e}_1 + y_2 \times \mathbf{e}_2 + \ldots + y_q \times \mathbf{e}_q$$



Left to right: original, reconstruction from 84, 40, 20, 3, 2, and 1 dimensions.



Pursuit Projection, PP

Similar to PCA, constructs **«the best» affine hyperplane** of smaller dimension

but for some other performance criteria

E.g. to better highlight clusters in data

One of the first papers is by J.H. Friedman and J. W. Tukey

“A Projection Pursuit Algorithm for Exploratory Data Analysis” (1974)

Multi Dimensional Scaling, MDS

Minimize quadratic form $\sum_{i,j} \left(\rho(O_i, O_j) - \| y_i - y_j \| \right)^2$

w.r.t. $y_1, y_2, \ldots, y_n \in R^q$, ρ is a some metrics in the object feature space

$y_1, y_2, \ldots, y_n \in R^q$ are defined **except for shift and rotation,** thus we use normalization, e.g.

$$\mathbf{Y}^T \times \mathbf{Y} = \mathbf{I_q} \quad \text{and} \quad \mathbf{Y}^T \times \mathbf{1} = \mathbf{0}$$

where $\mathbf{Y}^T = (y_1 : y_2 : \ldots : y_n)$ – (q×n)-matrix, $\mathbf{1} \in R^n$ – vectors of ones

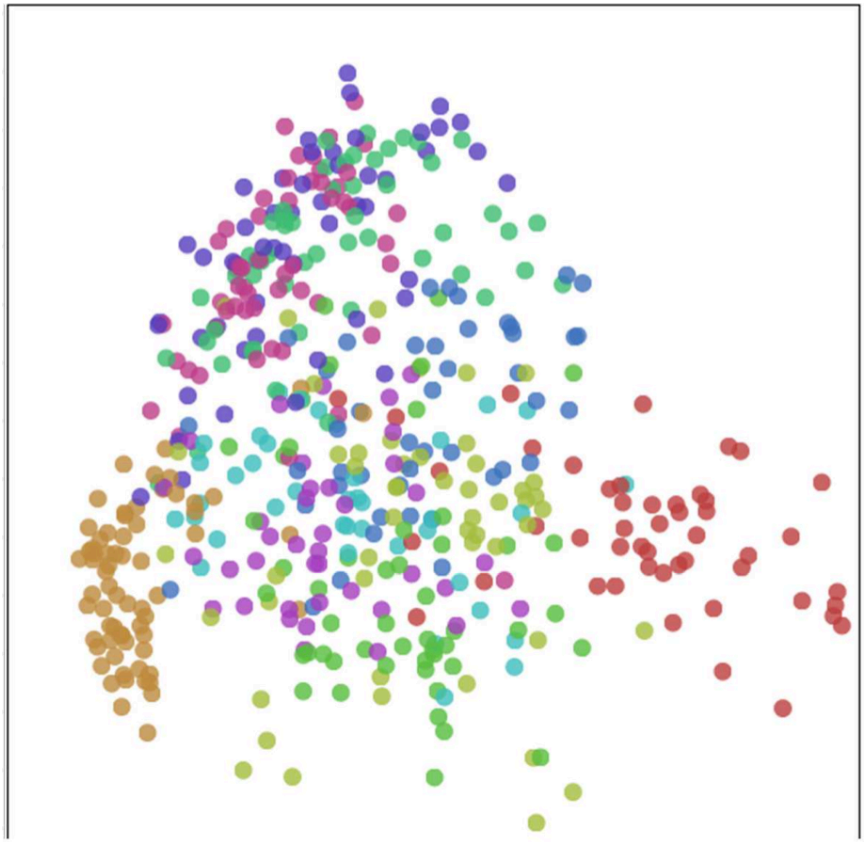
If $\rho(O_i, O_j) = \|X(O_i) - X(O_j)\|$, then MDS = PCA

Sammon mapping

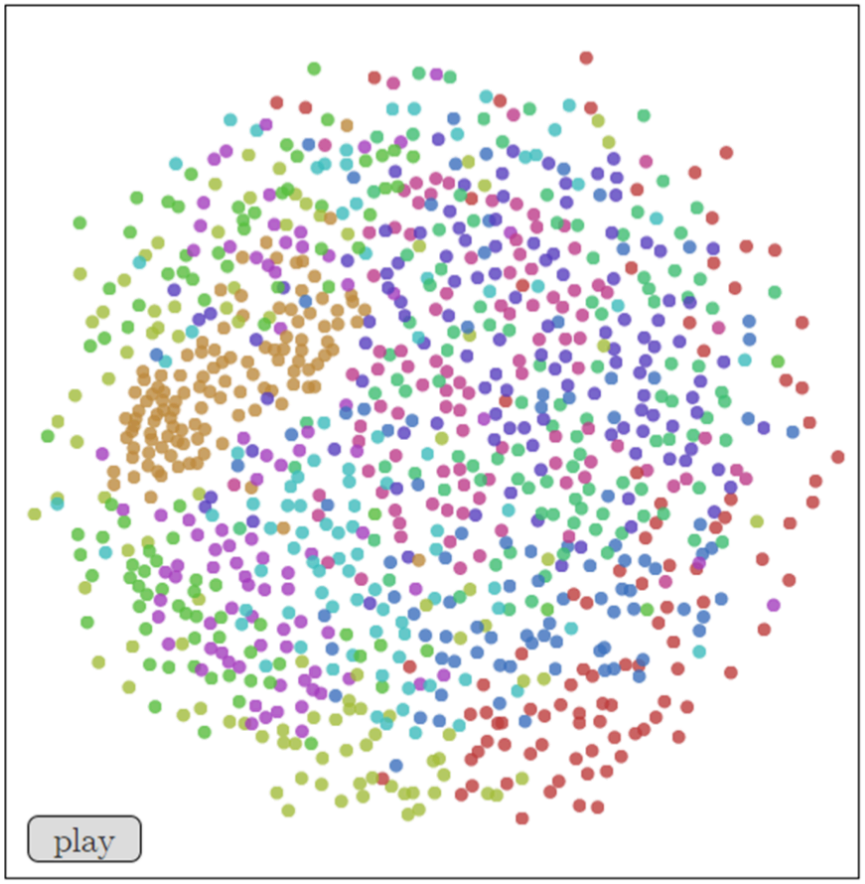
Common useful idea: local distances are often more important to preserve

$$\sum_{i,j} \frac{\left(\rho(O_i, O_j) - \left\| y_i - y_j \right\|\right)^2}{\left(\rho(O_i, O_j)\right)^2}$$

PCA vs. MDS

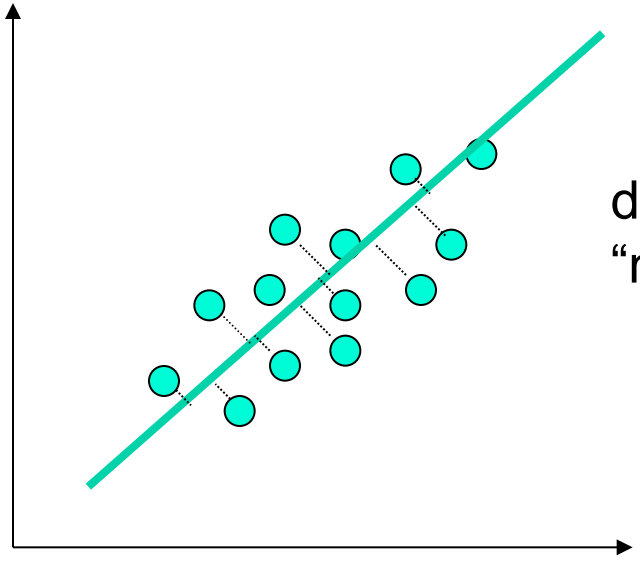


PCA

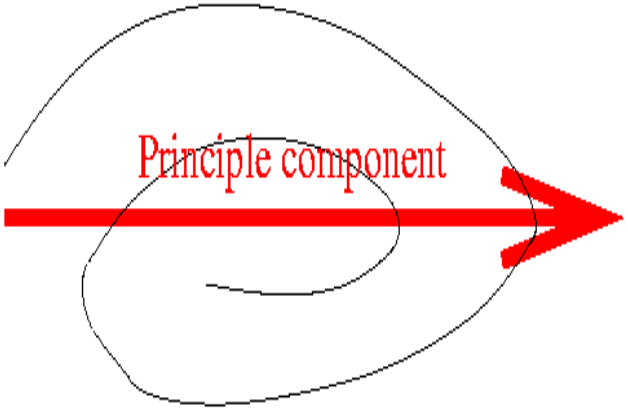


Sammon

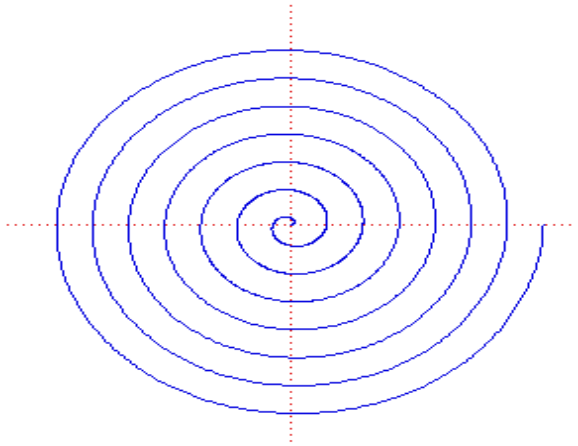
MDS, PCA, PP, MDS (for «Euclidean proximity») – linear methods,

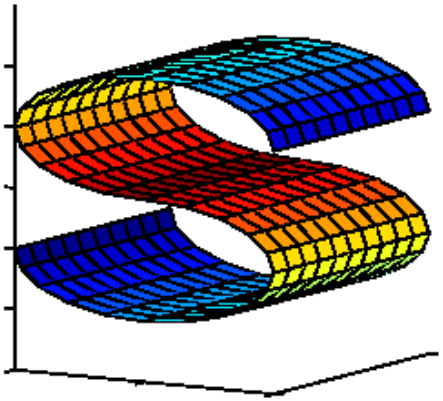


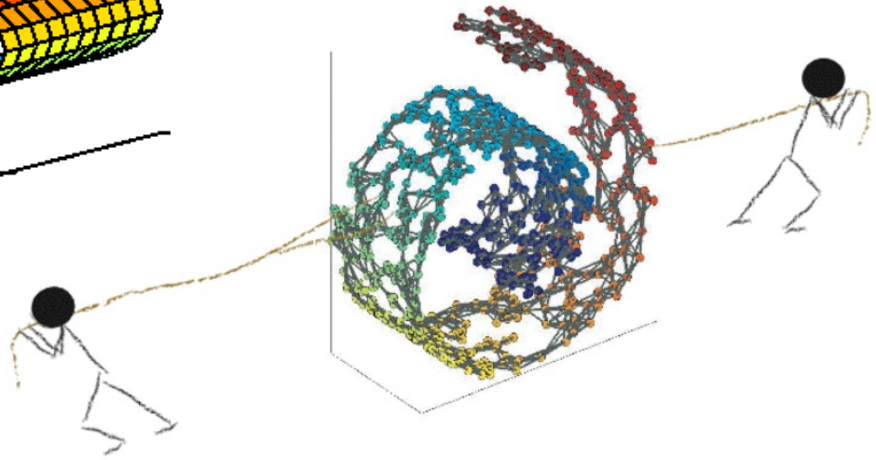
do not work for
“nonlinear data”:



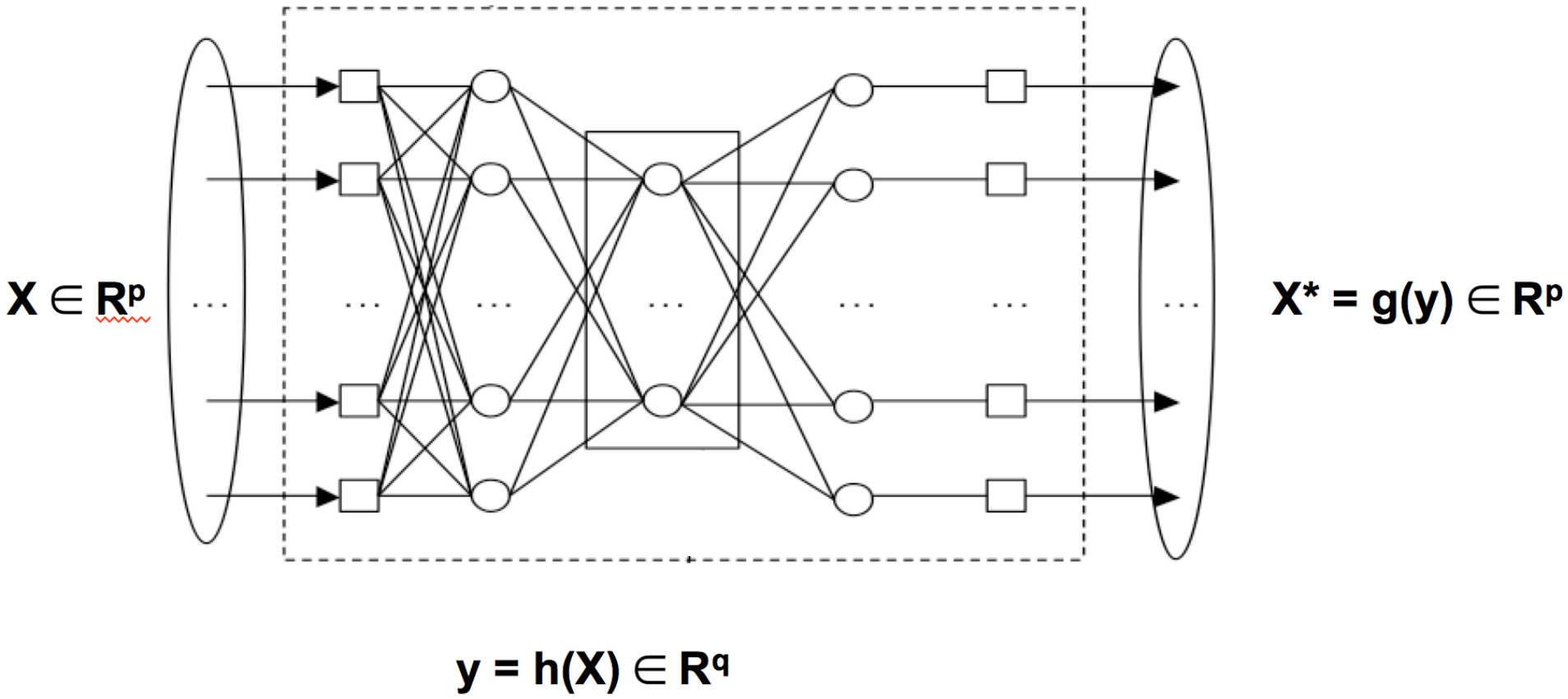
Spiral of Archimedes

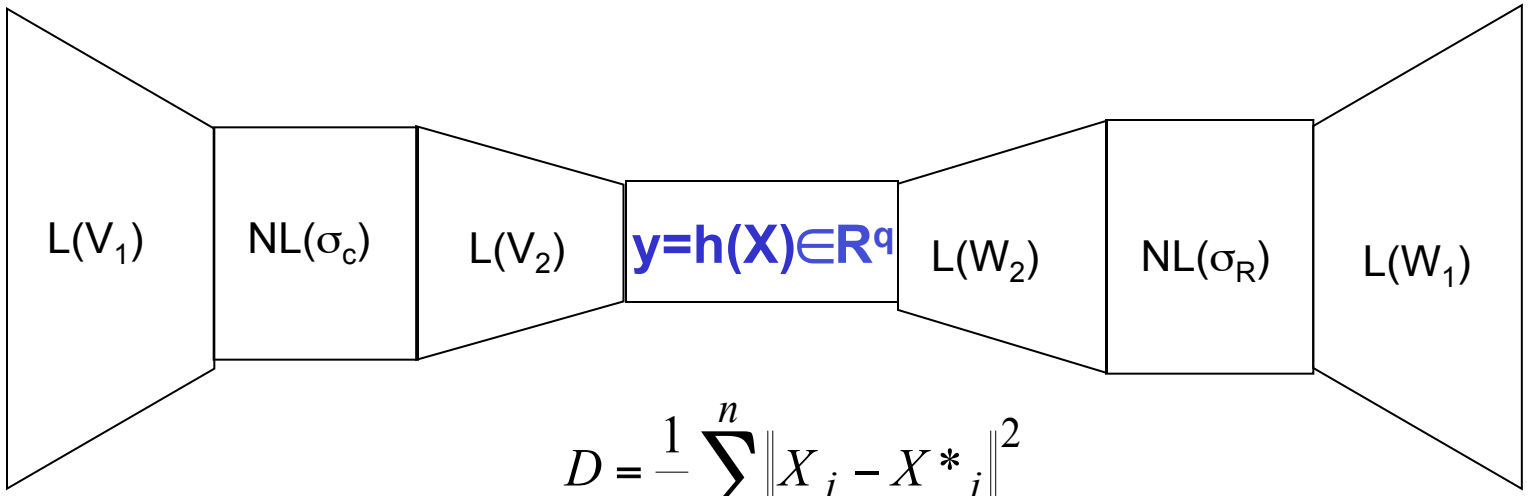






Replicative Neural Networks





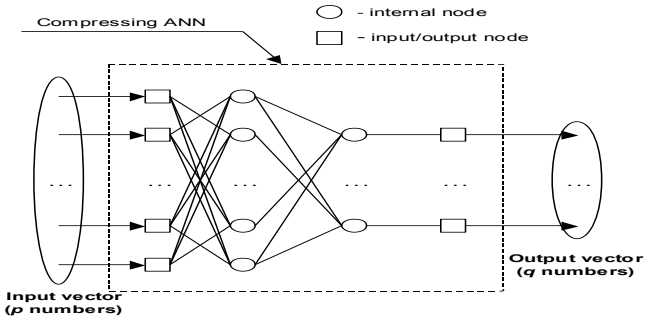
$$D = \frac{1}{n}\sum_{j=1}^{n}\left\|X_j - X^*_j\right\|^2$$

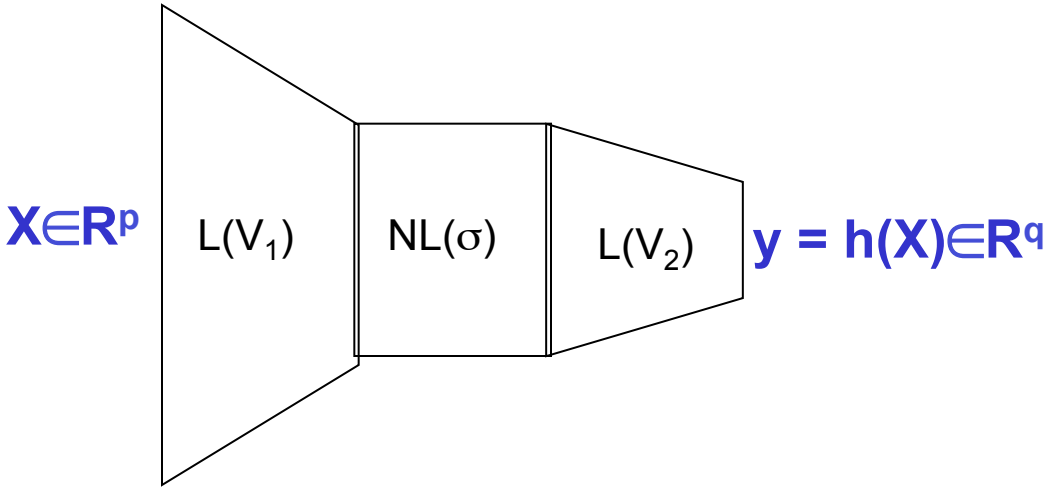
- 4 linear transformation with matrices V_1, V_2, W_1 and W_2
- 2 **fixed nonlinear transformations**

$\sigma(x)$ – **sigmoid functions**, e.g., $\sigma(x) = (1 + e^{-x})^{-1}$

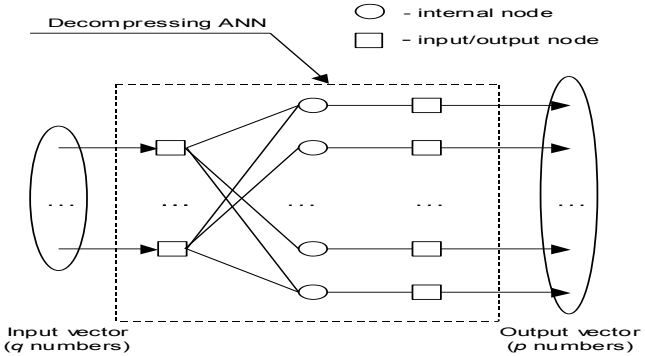
- Back-Propagation to optimize w.r.t. to parameters V_1, V_2, W_1 and W_2 the reconstruction error D

Compression transformation





Reconstruction transformation



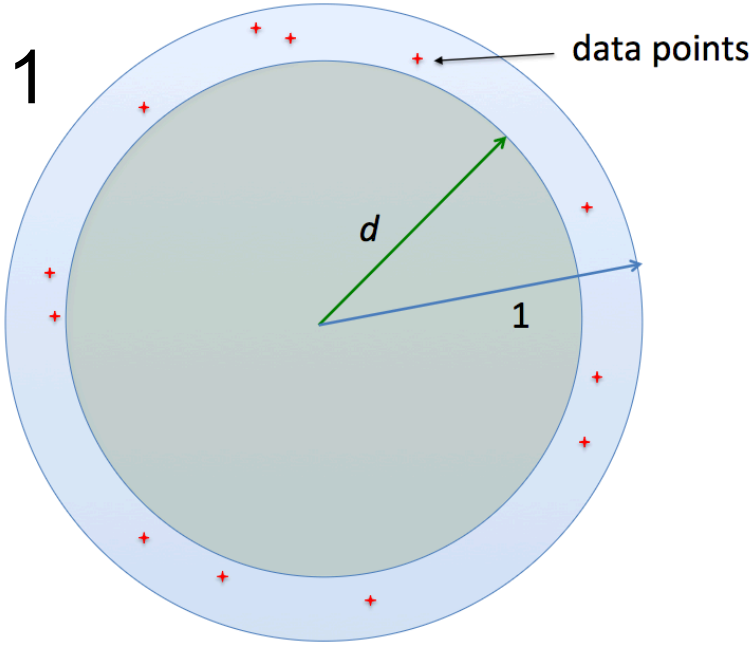
$y \in R^q$ $L(W_2)$ $NL(\sigma_R)$ $L(W_1)$ $X^* = g(y) \in R^p$

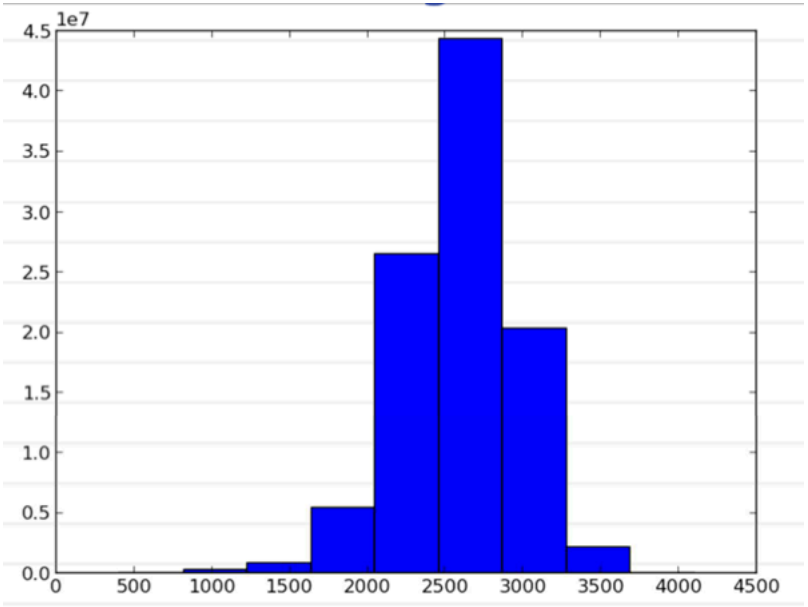
Curse of dimensionality

$$V(r) \sim C(d) * r^d$$

$$\frac{V(r) - V(r(1-\varepsilon))}{V(r)} = \frac{1 - (1-\varepsilon)^d}{1} \rightarrow 1$$

All volume of a ball is close to its cover





Histogram of pairwise Euclidian distances for MNIST

=> Consider more clever distance!!!!

Nonlinear (local) DR methods

5) **Locally Linear Embedding, LLE**
6) **Laplacian Eigenmaps LE**
7) **Hessian Eigenmaps, HE**
8) **ISOmetric MAPing, ISOMAP**
9) Kernel PCA, KPCA - Spectral Embedding Algorithm, SEA
10) Riemannian Manifold Learning, RML
11) Local Tangent Space Alignment, LTSA, **…**

GRAPH: standard step for many “local” DR methods

Step 1. Construct neighborhoods (ε-Neighborhoods, k Nearest Neighbors)

$$U(X) = U(X|\rho, \varepsilon) = \{X_i \in \mathbf{X}_n: \rho(X, X_i) \le \varepsilon\}$$

$\Rightarrow$ **Graph** $\Gamma(\mathbf{X}_n) = (\mathbf{X}_n, \mathbf{V})$: $(X_i, X_j) \in \mathbf{V} \Leftrightarrow X_j \in U(X_i)$ и $X_i \in U(X_j)$

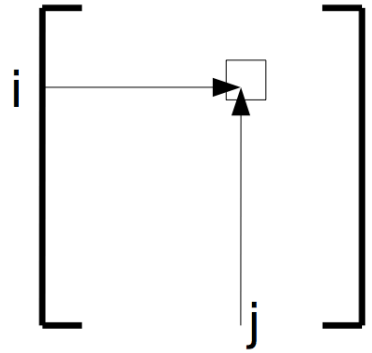
Step 2. Construct weights $w_{ij} = w(X_i, X_j) = w(v)$, $v = (X_i, X_j) \in \mathbf{V}$

$w_{0,ij} = 1$ for all $(X_i, X_j) \in V$, $\quad w_{0,ij} = 0$ in other cases

ISOMAP

Usual MDS

For *n* data points, and a distance matrix D,

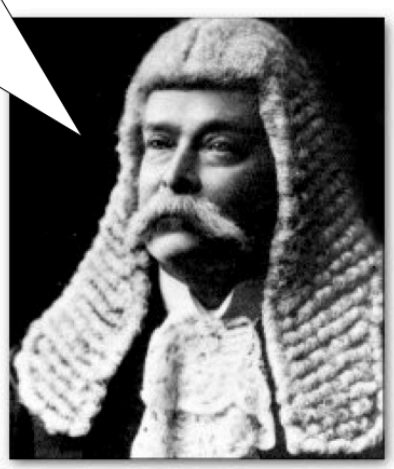
$D_{ij} =$ 

...we can construct a *m*-dimensional space to preserve inter-point distances by using the top eigenvectors of D scaled by their eigenvalues.

$$y_i = [\ \sqrt{\lambda_1} v_1^i\ ,\ \ \sqrt{\lambda_2} v_2^i\ ,\ \ldots\ \ ,\sqrt{\lambda_m} v_m^i\ ]$$

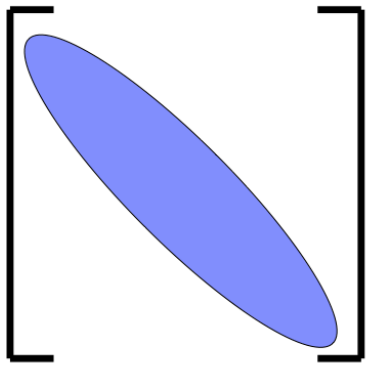
ISOMAP

Infer a distance matrix using distances along the manifold.

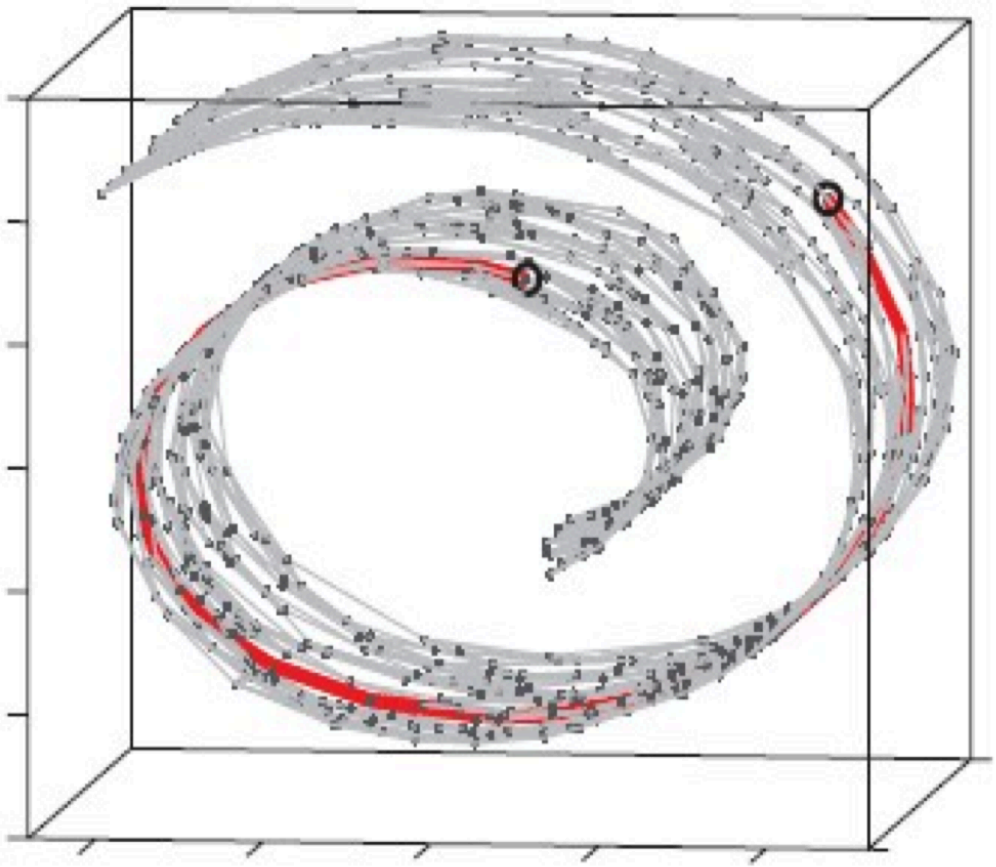


ISOMAP

1. Build a sparse graph with K-nearest neighbors

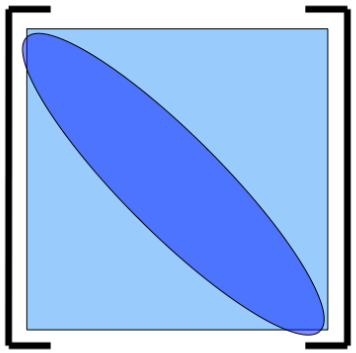
$D_g =$ 

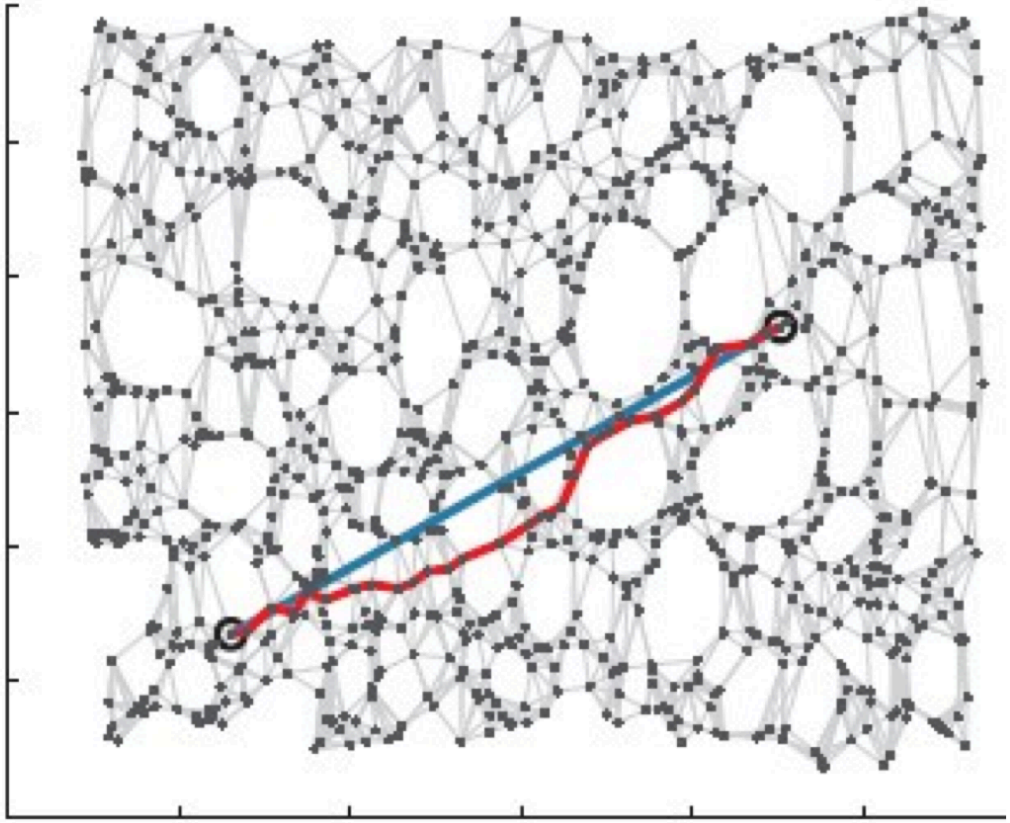
(distance matrix is sparse)



ISOMAP

2. Infer other interpoint distances by finding shortest paths on the graph (Dijkstra's algorithm).

$D_g =$ 

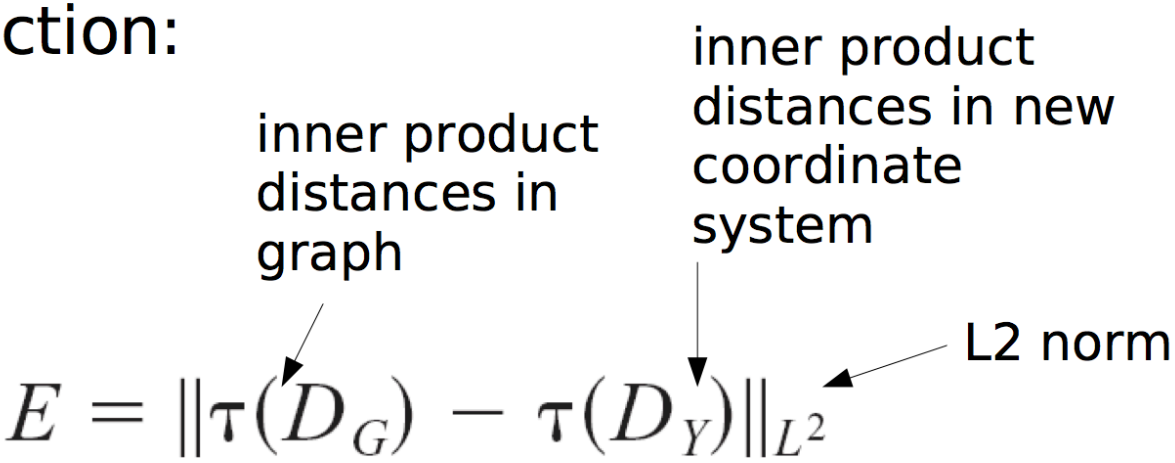


ISOMAP

Usual MDS

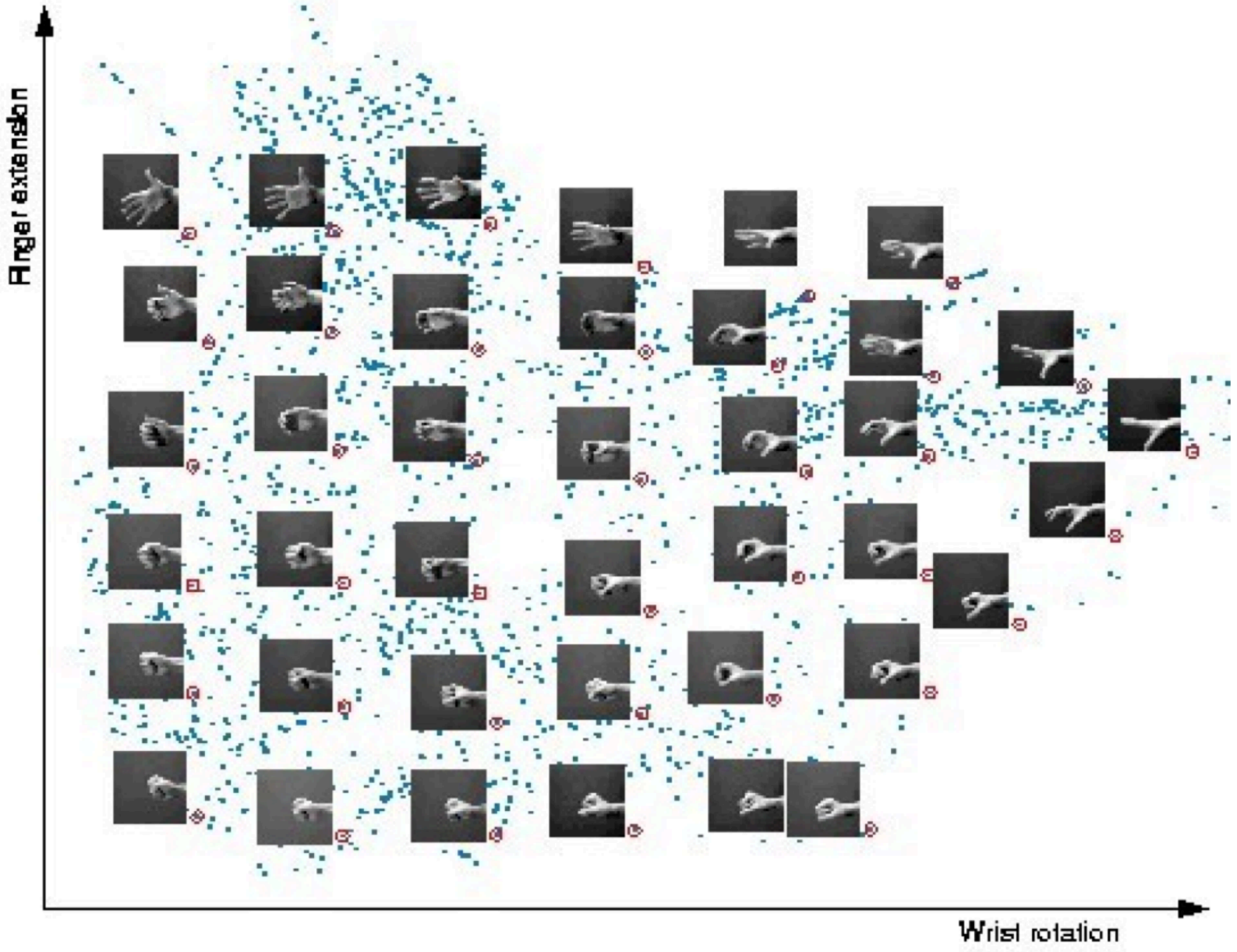
3. Build a low-D embedded space to best preserve the complete distance matrix.

Error function:

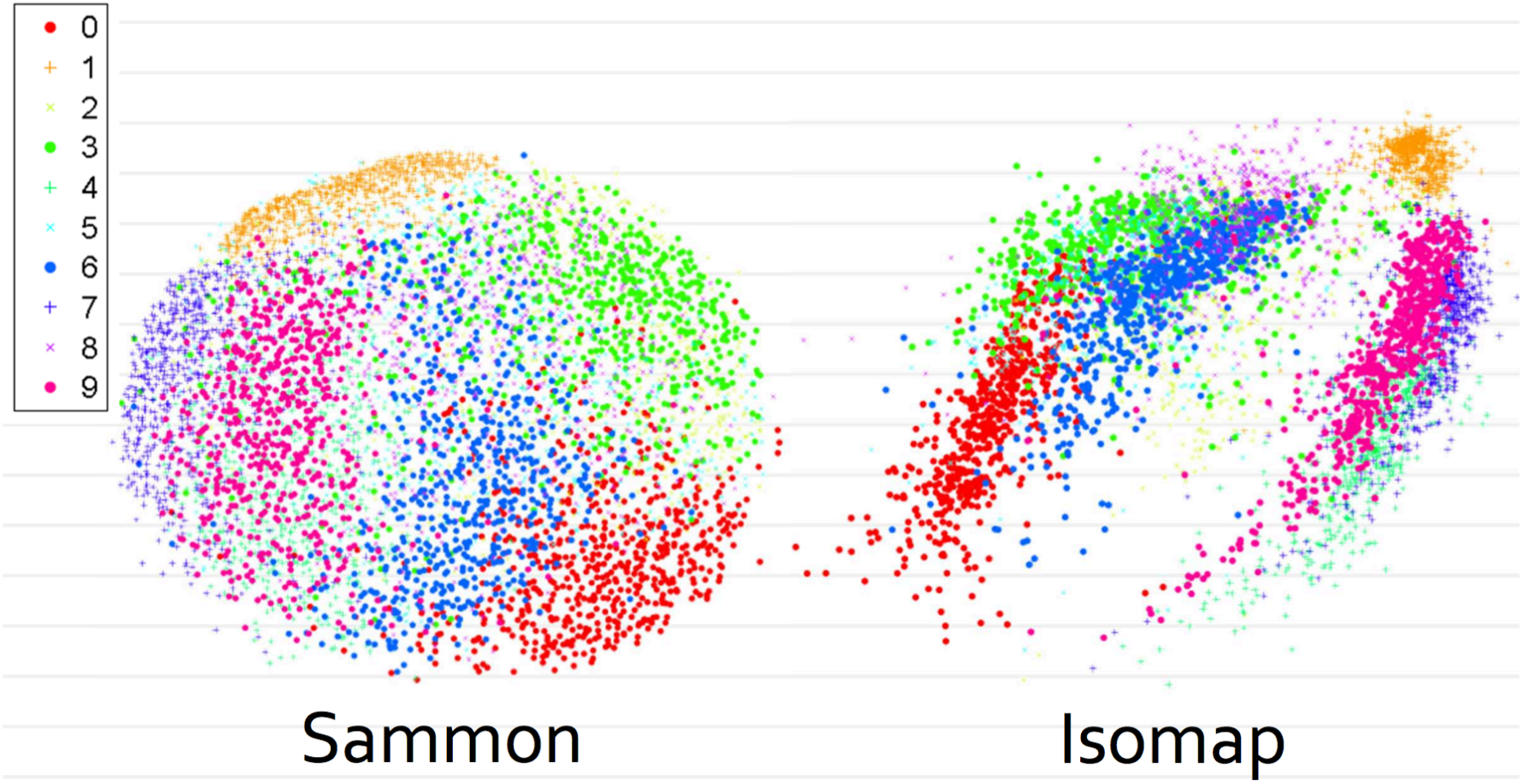


Solution – set points Y to top eigenvectors of D_g

Isomap results: hands



MNIST: Sammon vs. ISOMAP



Isomap: pro and con

- preserves global structure

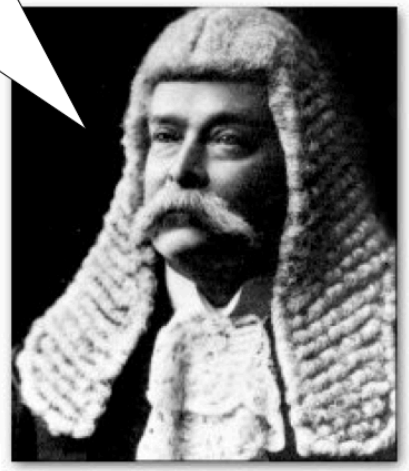
- few free parameters

- sensitive to noise, noise edges

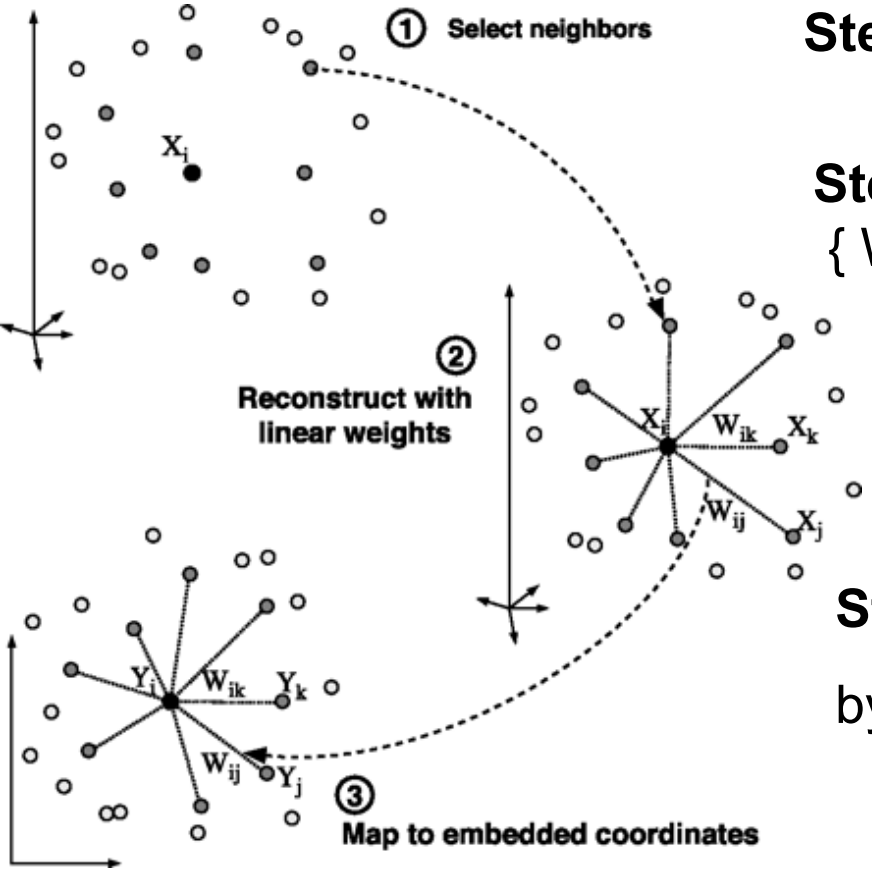
- computationally expensive (dense matrix eigen-reduction)

Locally Linear Embedding

Find a mapping to preserve local linear relationships between neighbors



Locally Linear Embedding, LLE (L.K. Saul, et al., 2000)



Step 1. K nearest neighbors

Step 2. Get "Baricentric" coordinates $\{ W_{ji} \}$ by minimizing

$$J_1(\mathbf{W}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j=1}^{K} W_{ji} \mathbf{x}_{j(i)} \right\|^2$$

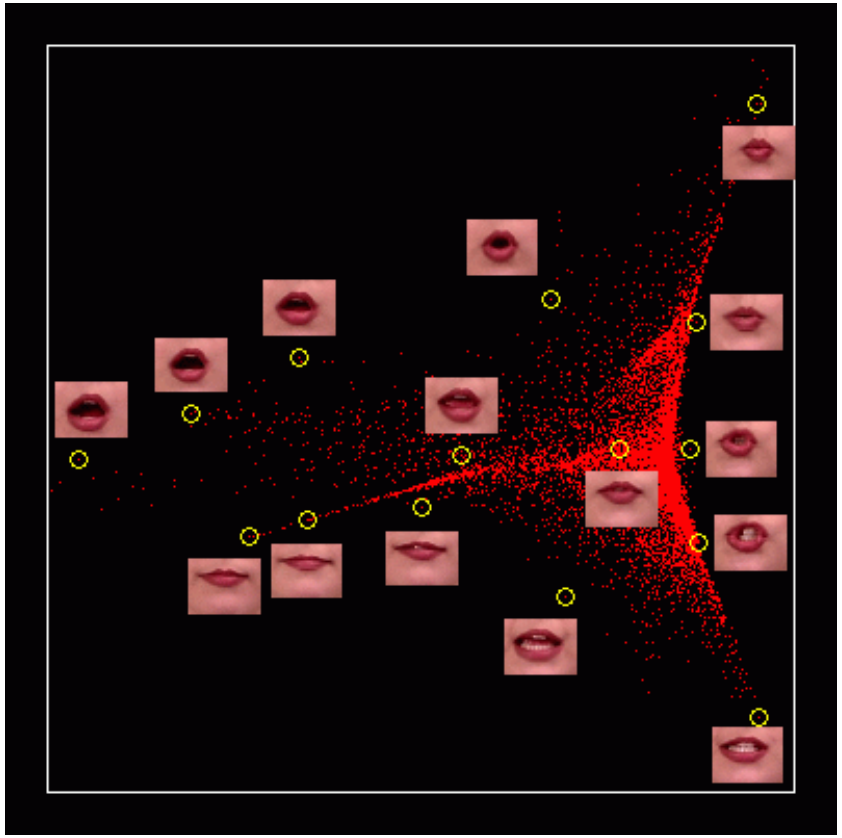
Step 3. Get $\{y_1, y_2, \ldots, y_n\} \in R^q$

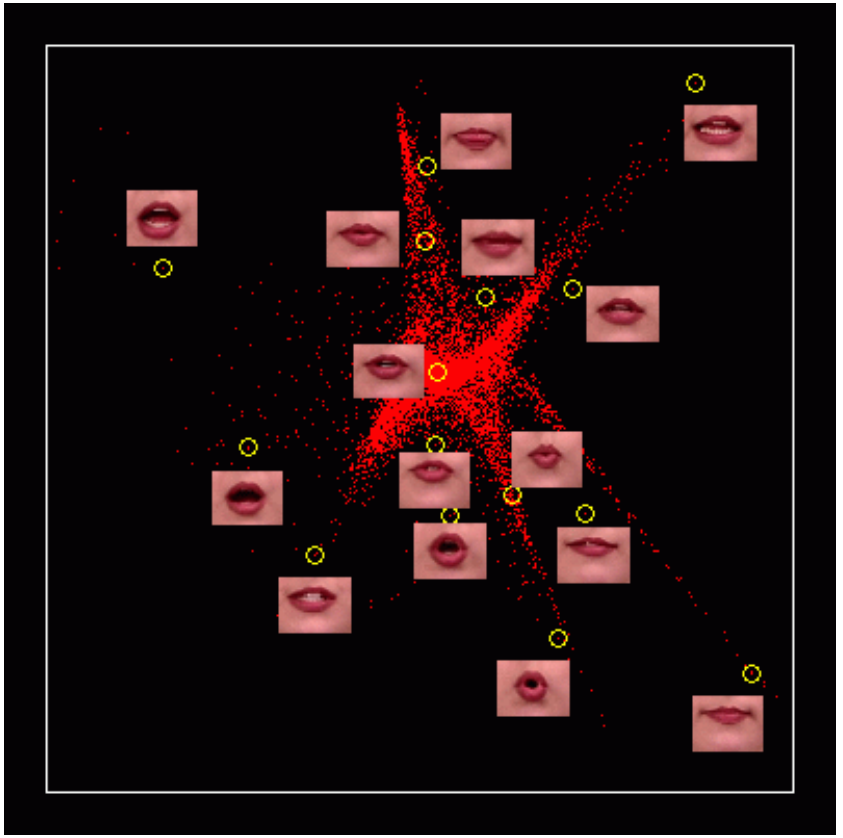
by minimizing

$$J_2(\mathbf{Y}) = \sum_{i=1}^{n} \left\| \mathbf{y}_i - \sum_{j=1}^{N} W_{ji} \mathbf{y}_{j(i)} \right\|^2$$

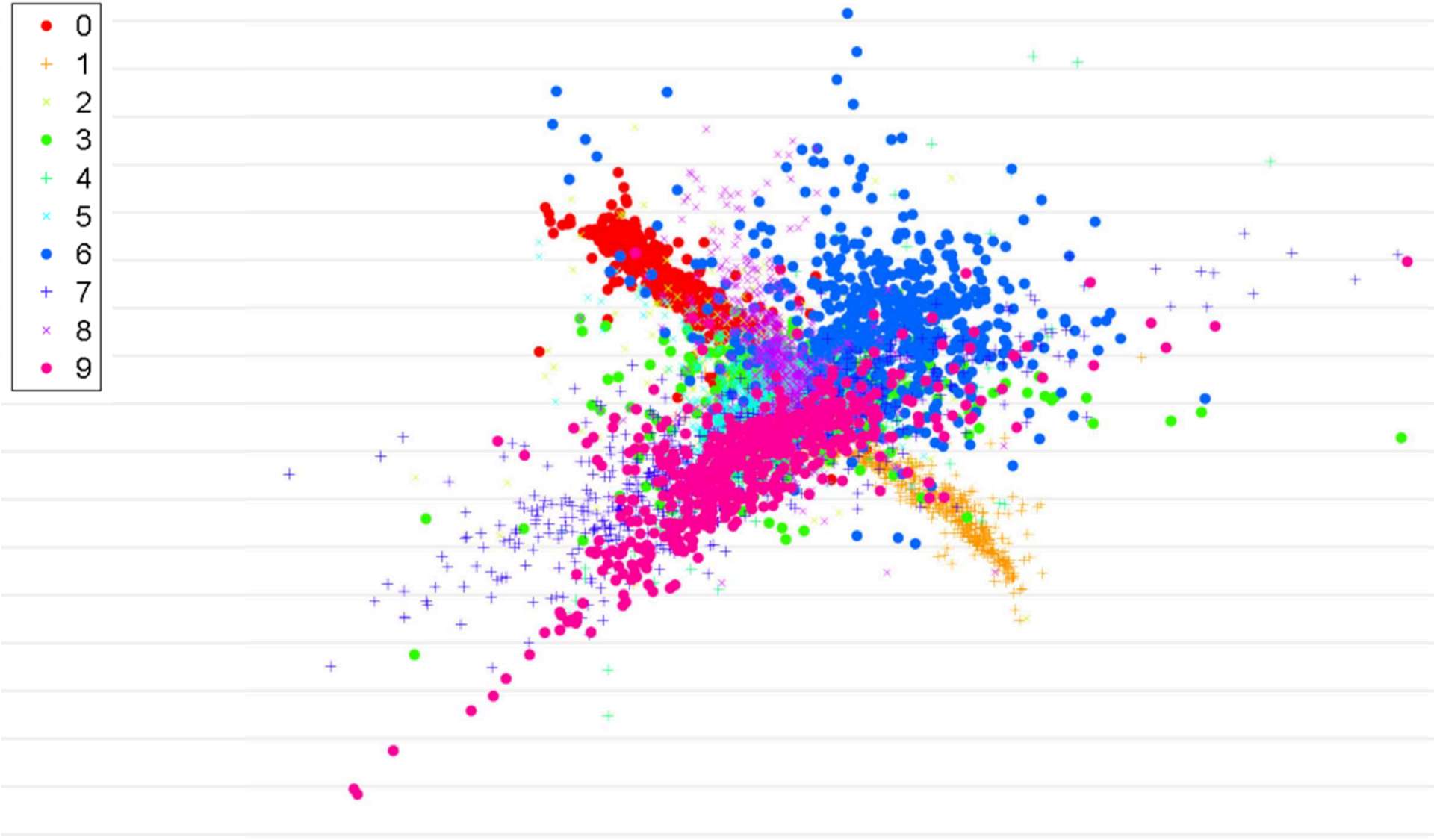
with standard normalizing conditions

LLE: Lips pictures





LLE: MNIST



LLE: pro and con

- no local minima, one free parameter

- incremental & fast

- simple linear algebra operations

- can distort global structure

Laplacian Eigenmaps (LE) (M. Belkin, P. Niyogi, 2002)

Step 1 is standard

Step 2: $w_{ij} = w_{0,ij} \times \exp(- t^{-1} \times \|X_i - X_j\|^2)$ for all $(X_i, X_j) \in V$

Step LE3. Construct embedding. Values $y_i = h(X_i)$ minimize quadratic form

$$\sum_{i,j=1}^{n} W_{ij} \times \|y_i - y_j\|^2 = \mathrm{Tr}(\mathbf{Y}^T \times R_{LE} \times \mathbf{Y})$$

s.t. $Y^T \times D \times Y = I_q$

$D = \mathrm{Diag}(\sum_{j=1}^{n} W_{ij})$, where $R_{LE} = D - W$, $W = \|w_{ij}\|$,

$\mathbf{Y}^T$ is $(q \times n)$-matrix with columns $\{y_1, y_2, \ldots, y_n\} \in R^q$

Solution: Y = $(Y_1, Y_2, \dots, Y_q)$, $Y_i \in R^n$ is a solution of a system of n linear equations

$$R_{LE}\, Y = \lambda_i \times DY$$

for q smallest **positive** roots $\lambda_1, \lambda_2, \dots, \lambda_q > 0$ of the equation

$$Det(R_{LE} - \lambda \times D) = 0.$$

We do not consider $\lambda_0 = 0$, since it corresponds to $Y_0 = \mathbf{1} \in R^n$

and contradicts our normalization $\mathbf{Y \times 1 = 0}$

Justification: why Laplacian

- Manifold **X** $\subset R^p$ is smooth with Riemannian structure

- Let **q = 1** and h is an embedding transformation of **X** in **Y** = h(**X**) $\subset R^1$, $\nabla h(X)$ is a gradient (vector field on a manifold **X**):

$$\|h(x) - h(z)\| \le \|\nabla h(x)\| \times \|x - z\| + o\|x - z\|;$$

- "we see that if $\|\nabla h(x)\|$ provides us with an estimate of how far apart h maps nearby points. We therefore **look a map that best preserves locality on average by trying to find**

$$\arg\min\{I(h) : \|h\|_{L^2(\mathbf{X})} = 1\} \text{ , where } I(h) = \int_{\mathbf{X}} \|\nabla h(x)\|^2 mes(dx)$$
;

- $L_{LB}h = -\operatorname{div}(\nabla h)$, then by Stocks theorem

$$\int_{\mathbf{X}} \|\nabla h\|^2 = \int_{\mathbf{X}} L_{LB}(h) \times h$$

$\Rightarrow$ minimization of I(h) <=> minimization of $\int_{\mathbf{X}} L_{LB}(h) \times h$

- operator L_{LB} has discrete spectra $0 = \lambda_0 \le \lambda_1 \le \lambda_2 \le \dots$, and $h_0 \equiv \text{const}$

$\Rightarrow h_1 \perp h_0$ minimizes I(h), h_i are eigenfunctions of L_{LB};

- **minimization** of ***I(h)*** = $\int_{\mathbf{X}} \|\nabla h\|^2$ <=> **minimization of a sample operator**

$$Lh = \sum_{i,j=1}^{n} W_{ij} \times \left(h(X_i) - h(X_j)\right)^2$$

defined on nodes of a graph графа $\Gamma(\mathbf{X}_n)$; it is a discrete analogue of Laplace-Beltrami operator

Hessian Eigenmaps, **HE** (D.L. Donoho, C. Grimes, 2002).

Hessian Eigenmaps is "LE only with the Hessian replacing the Laplacian"

$$\int_{\mathbf{X}} \left\| \nabla h(x) \right\|^2 mes(dx) \quad \Rightarrow \quad H(h) = \int_{\mathbf{X}} \left\| H_h(x) \right\|_F^2 mes(dx)$$

Two more methods (not covered here):

Riemannian Manifold Learning, RML

Local Tangent Space Alignment, LTSA

stochastic neighbor embedding

"Stochastic" similarity in the initial space

$$p_{j|i} = \frac{exp(- \parallel x_i - x_j \parallel^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(- \parallel x_i - x_k \parallel^2 / 2\sigma_i^2)}$$

Similarity in the compressed space

$$q_{j|i} = \frac{exp(- \parallel y_i - y_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel y_i - y_k \parallel^2)}$$

stochastic neighbor embedding

"Stochastic" similarity in the initial space

$$p_{j|i} = \frac{exp(- \parallel x_i - x_j \parallel^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(- \parallel x_i - x_k \parallel^2 / 2\sigma_i^2)}$$

Similarity in the compressed space

$$q_{j|i} = \frac{exp(- \parallel y_i - y_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel y_i - y_k \parallel^2)}$$

$$Cost = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

stochastic neighbor embedding

$$\frac{\partial Cost}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

To escape poor minima:
- Initialize with Gaussian noise
- Add diminishing Gaussian noise on each stage

t-sne: Similarity in the compressed space in case ot t(1) distribution

$$q_{ij} = \frac{(1+\parallel y_i - y_j \parallel^2)^{-1}}{\sum_{k \neq l}(1+\parallel y_k - y_l \parallel^2)^{-1}}$$

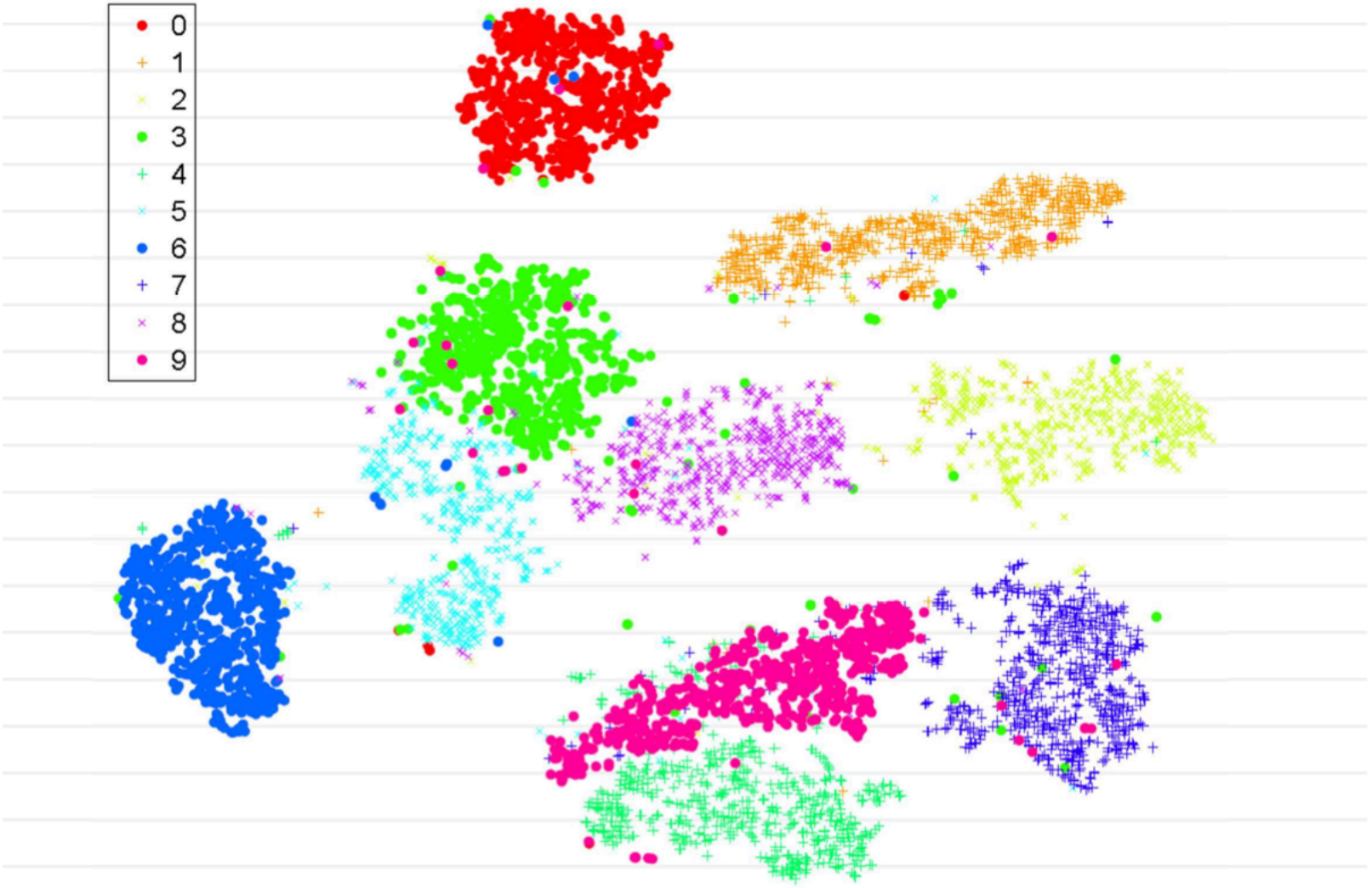
stochastic neighbor embedding

t-sne: minimize KL between the whole densities P and Q

$$Cost = KL(P \parallel Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

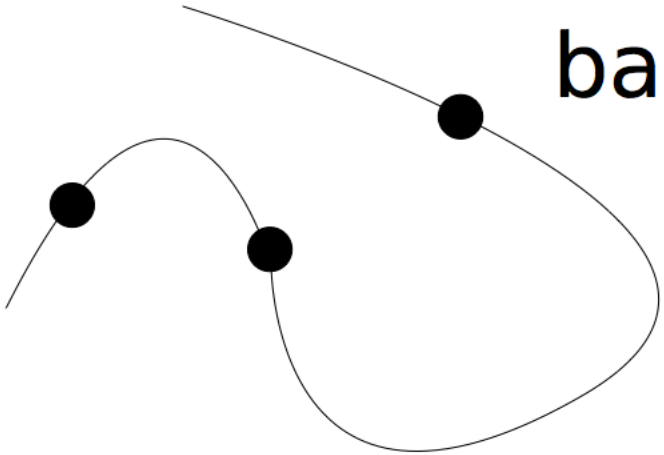
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

t-stochastic neighbor embedding: MNIST

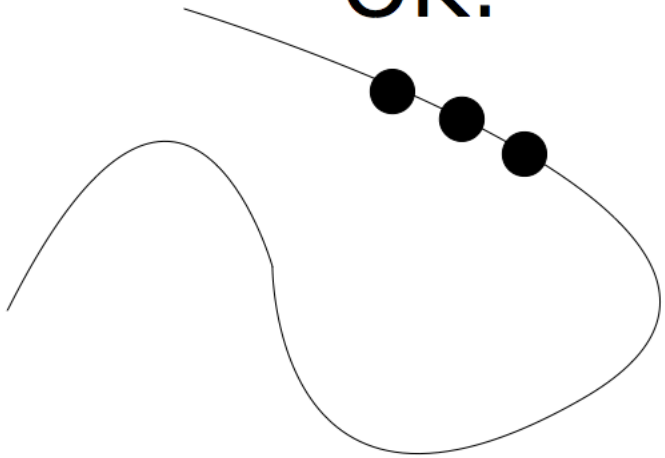


No Free Lunch

the “curvier” your manifold, the denser your data must be



bad



OK!