Dimensionality Reduction

Alexey Zaytsev,
Most slides are from Evgeny Burnaev

Skoltech, Moscow, Russia

Skoltech
Skolkovo Institute of Science and Technology

Outline

Dimensionality Reduction Problem

- Object O is described by a p-dimensional vector $\mathbf{x}(O) \in \mathbb{R}^p$. Components of $\mathbf{x}(O)$ are features of O
- **Example 1: human face**: Image is represented by 1024×1024 pixels — dimension $p = 10^{20} \sim 1000000$

Dimensionality Reduction Problem

- Object O is described by a p-dimensional vector $\mathbf{x}(O) \in \mathbb{R}^p$. Components of $\mathbf{x}(O)$ are features of O
- **Example 1: human face**: Image is represented by 1024×1024 pixels — dimension $p = 10^{20} \sim 1000000$

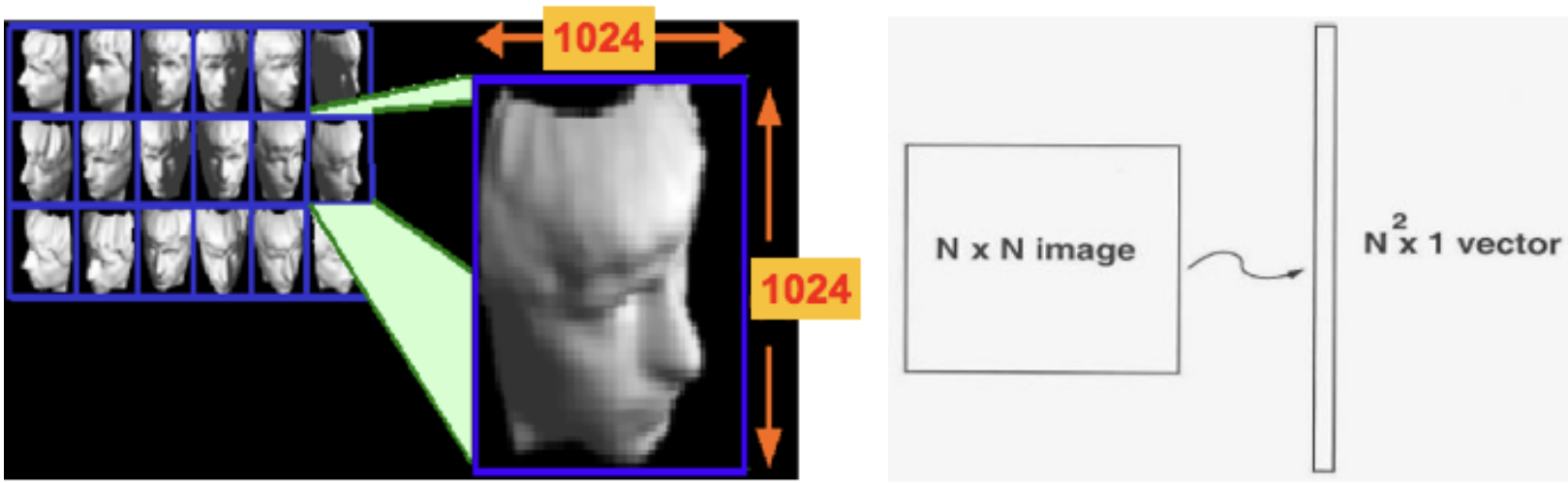


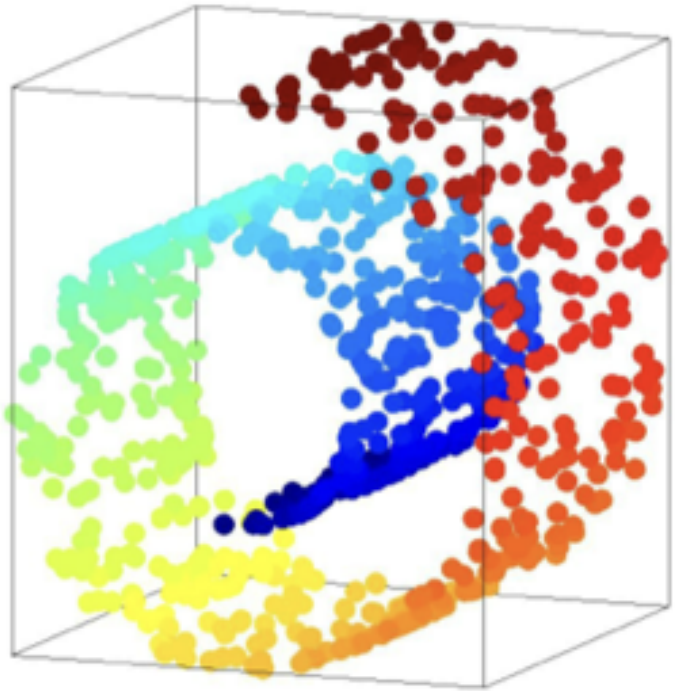
Figure – Grey-scale pixel-wise representation of a human face

- **Example 2: MNIST**

Examples

- **Example 2: MNIST**

- **Example 3: a toy problem "Swiss Roll"**



- **Example 4:** multivariate time-series

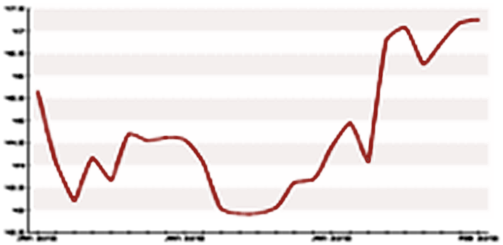
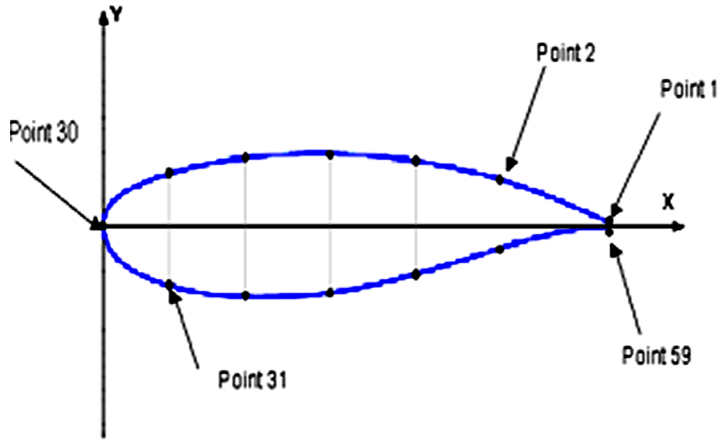


Figure – Electricity price time series

Figure – Speech signal

Time-series segment $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is a p-dimensional vector.

- **Example 5:** wing airfoil description



Airfoil $O \to \mathbf{x}(O) = (x_1, x_2, \ldots, x_p)$ — ordinates of upper and lower contours, $p \sim 50 \div 200$

- **Example 4:** multivariate time-series

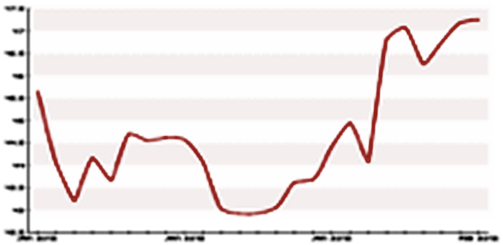
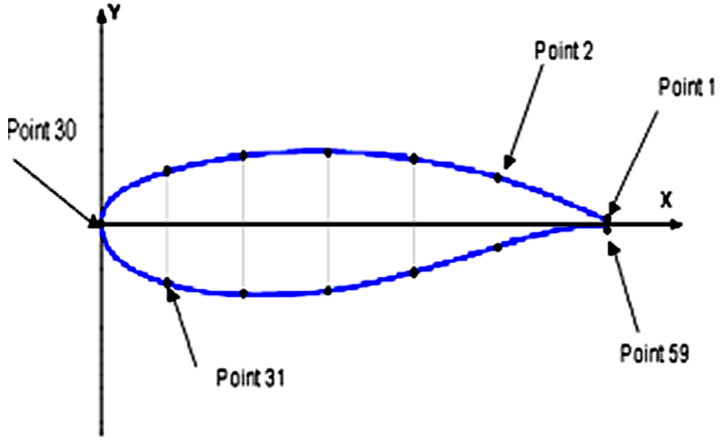


Figure – Electricity price time series

Figure – Speech signal

Time-series segment $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is a p-dimensional vector.

- **Example 5:** wing airfoil description



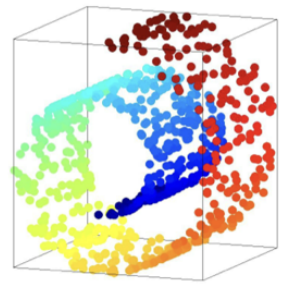
Airfoil $O \to \mathbf{x}(O) = (x_1, x_2, \ldots, x_p)$ — ordinates of upper and lower contours, $p \sim 50 \div 200$

Dimensionality Reduction Problem

- High dimensionality p of $\mathbf{x}(O)$ is critical for efficient learning
- The main idea of **Dimensionality Reduction** is constructing reduced dimension representation $\mathbf{z}(O) \in \mathbb{R}^q, q \ll p$, of O without "significant loss of information"
- **DR** for
  - Visualization (only 2D/3D)
  - Data compression
  - Curse of dimensionality
  - De-noising
  - Reasonable distance metrics
  - Representation learning

$$\mathbf{x} \to \mathbf{z} \quad \text{s.t.}$$

$$\dim(\mathbf{z}) \ll \dim(\mathbf{x})$$



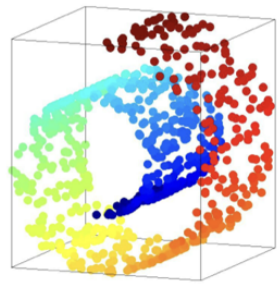
Theory: uncover «intrinsic» dimensionality

Dimensionality Reduction Problem

- High dimensionality p of $\mathbf{x}(O)$ is critical for efficient learning
- The main idea of **Dimensionality Reduction** is constructing reduced dimension representation $\mathbf{z}(O) \in \mathbb{R}^q, q \ll p$, of O without "significant loss of information"
- **DR** for
  - Visualization (only 2D/3D)
  - Data compression
  - Curse of dimensionality
  - De-noising
  - Reasonable distance metrics
  - Representation learning

$$\mathbf{x} \to \mathbf{z} \quad \text{s.t.}$$

$$\dim(\mathbf{z}) \ll \dim(\mathbf{x})$$



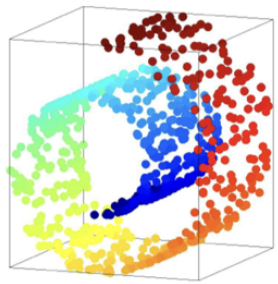
Theory: uncover «intrinsic» dimensionality

Dimensionality Reduction Problem

- High dimensionality p of $\mathbf{x}(O)$ is critical for efficient learning
- The main idea of **Dimensionality Reduction** is constructing reduced dimension representation $\mathbf{z}(O) \in \mathbb{R}^q, q \ll p$, of O without "significant loss of information"
- **DR** for
  - Visualization (only 2D/3D)
  - Data compression
  - Curse of dimensionality
  - De-noising
  - Reasonable distance metrics
  - Representation learning

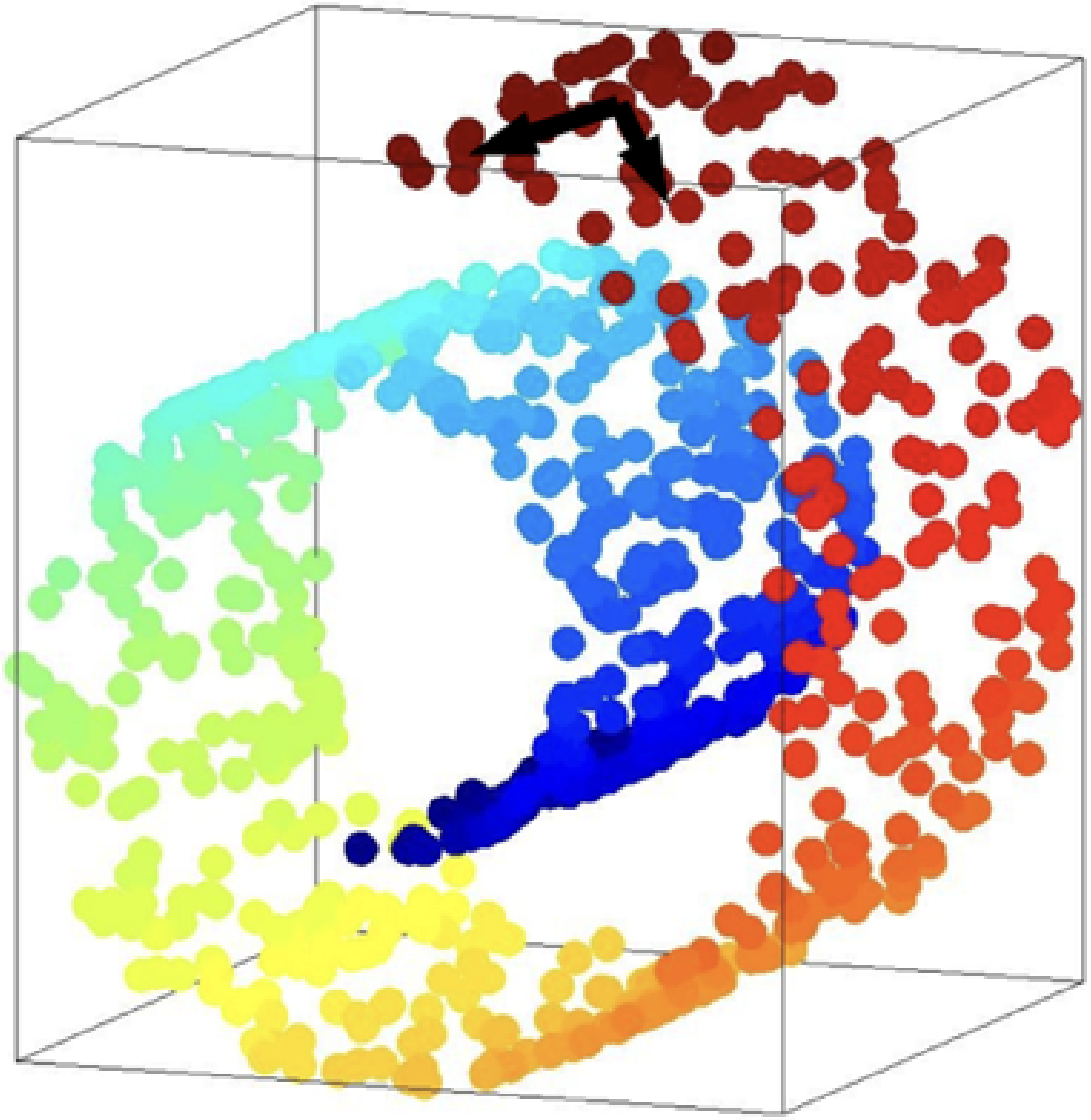
$$\mathbf{x} \to \mathbf{z} \quad \text{s.t.}$$

$$\dim(\mathbf{z}) \ll \dim(\mathbf{x})$$



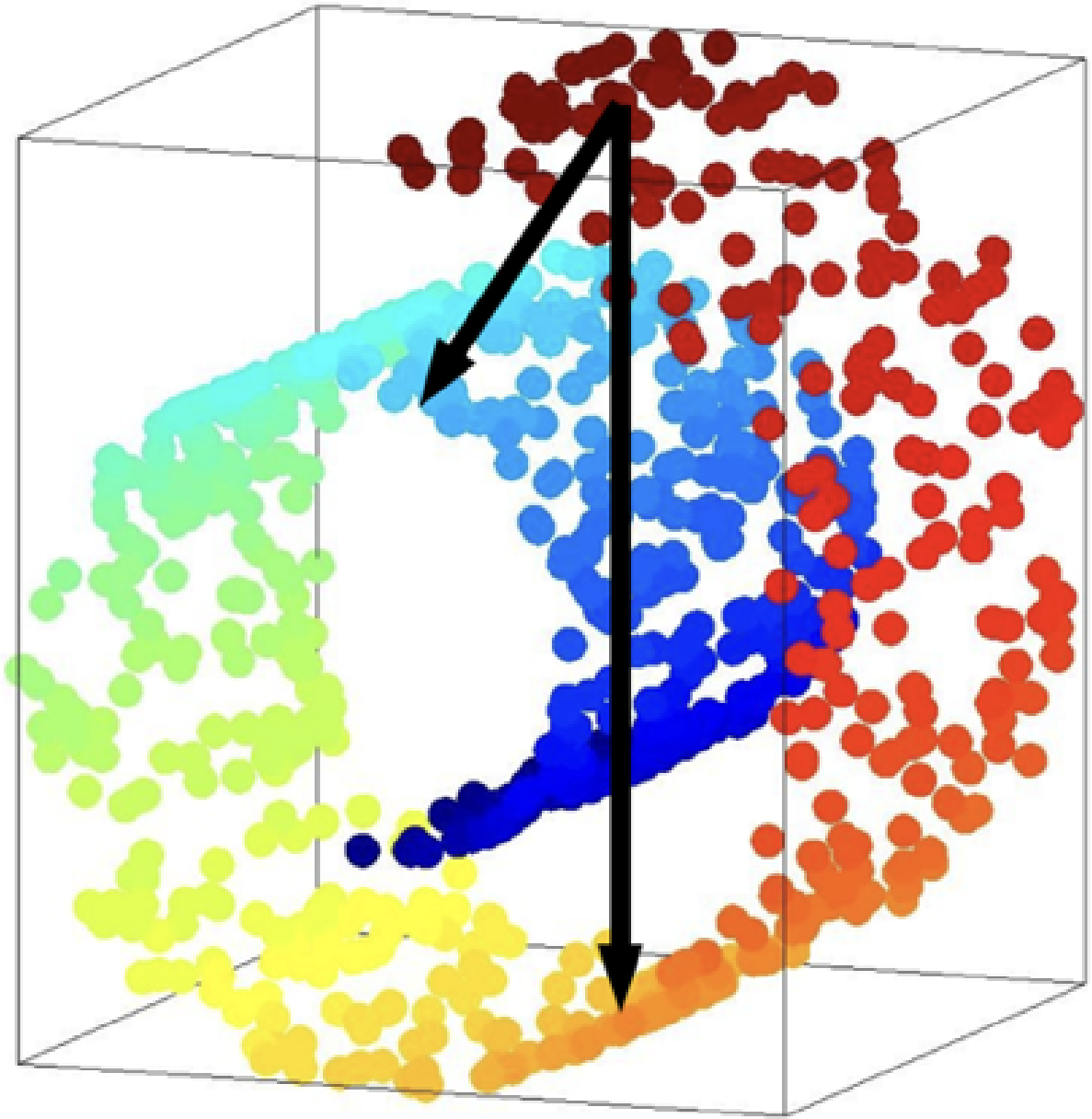
Theory: uncover «intrinsic» dimensionality

What is a reasonable distance metric and Intrinsic dimension?



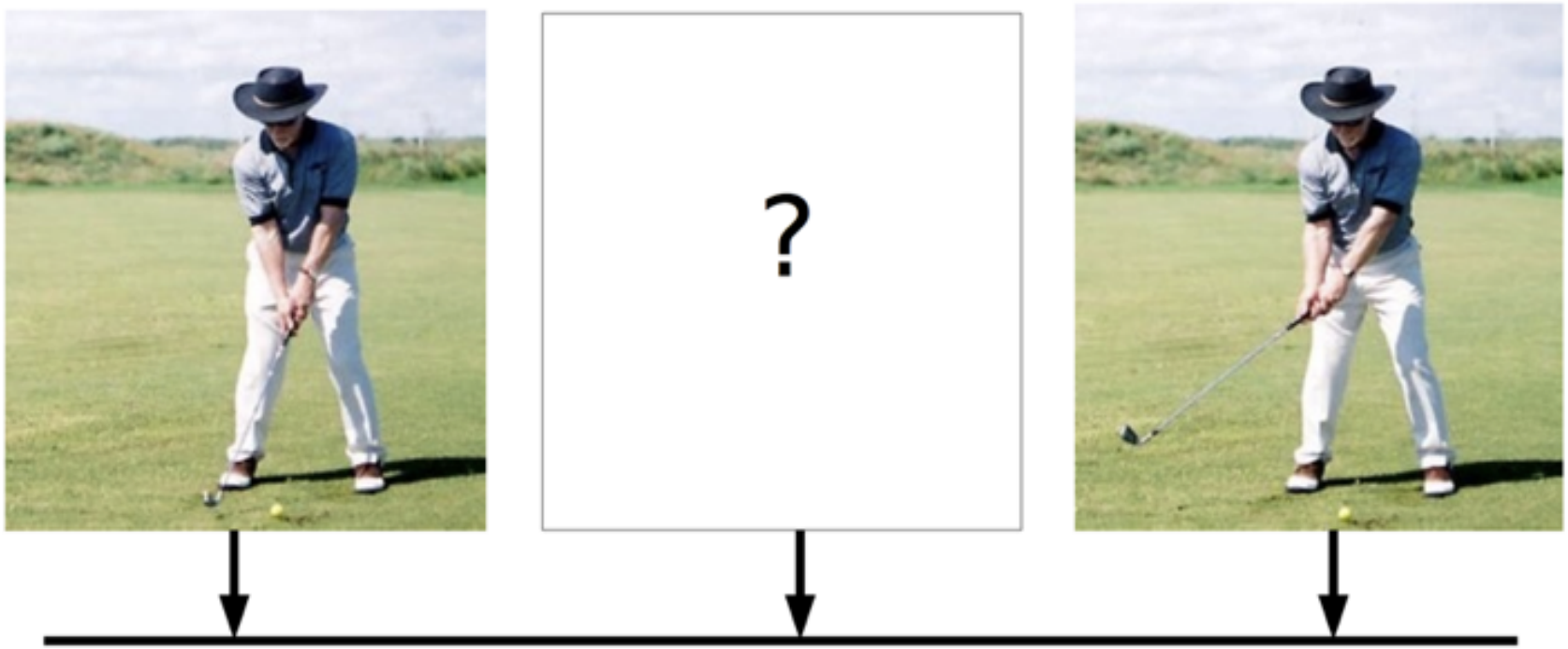
The intrinsic dimension for Swiss Roll is 2

The metric should take structure into account.

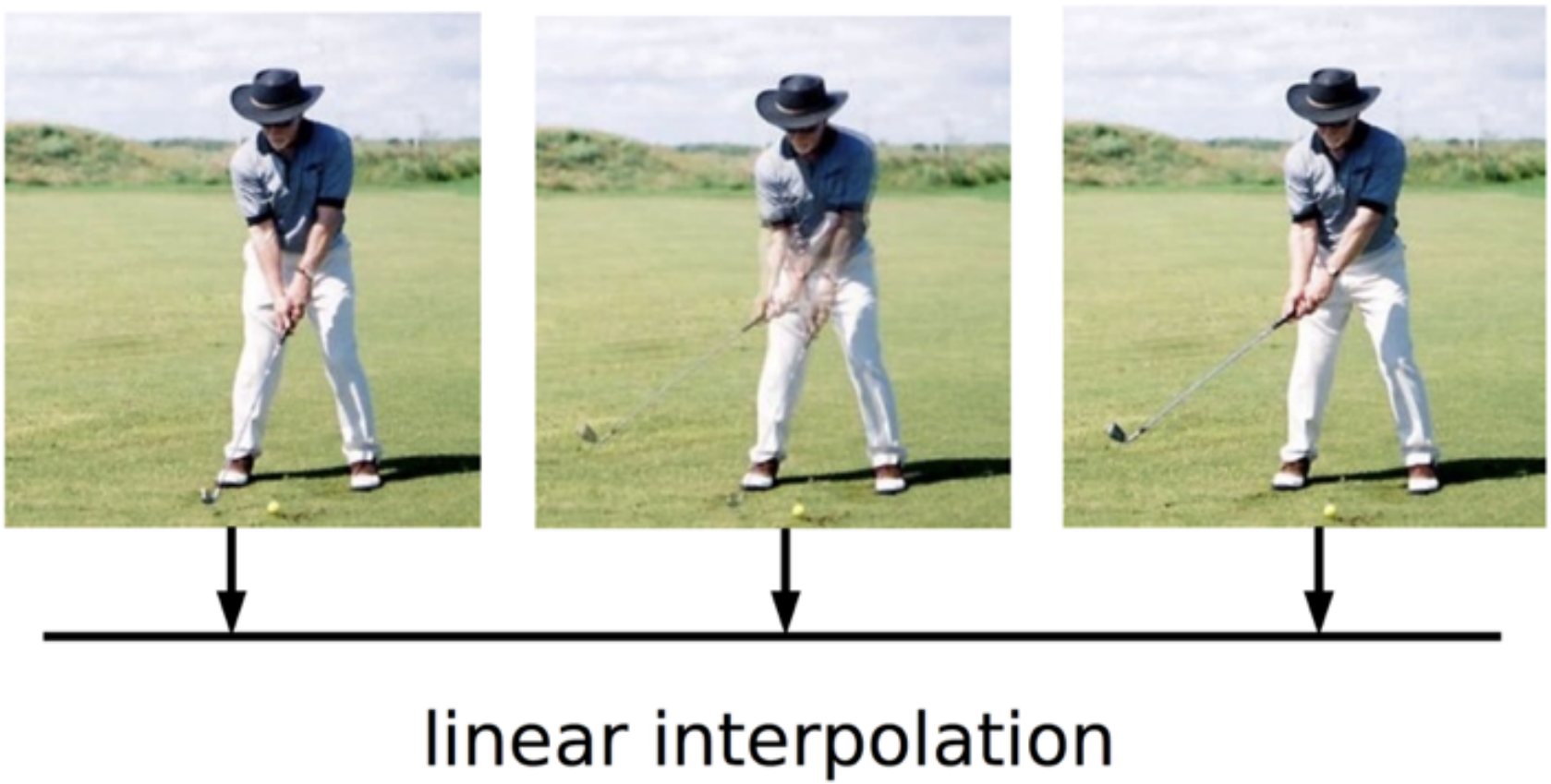


What is "Reasonable distance metrics" in practice?

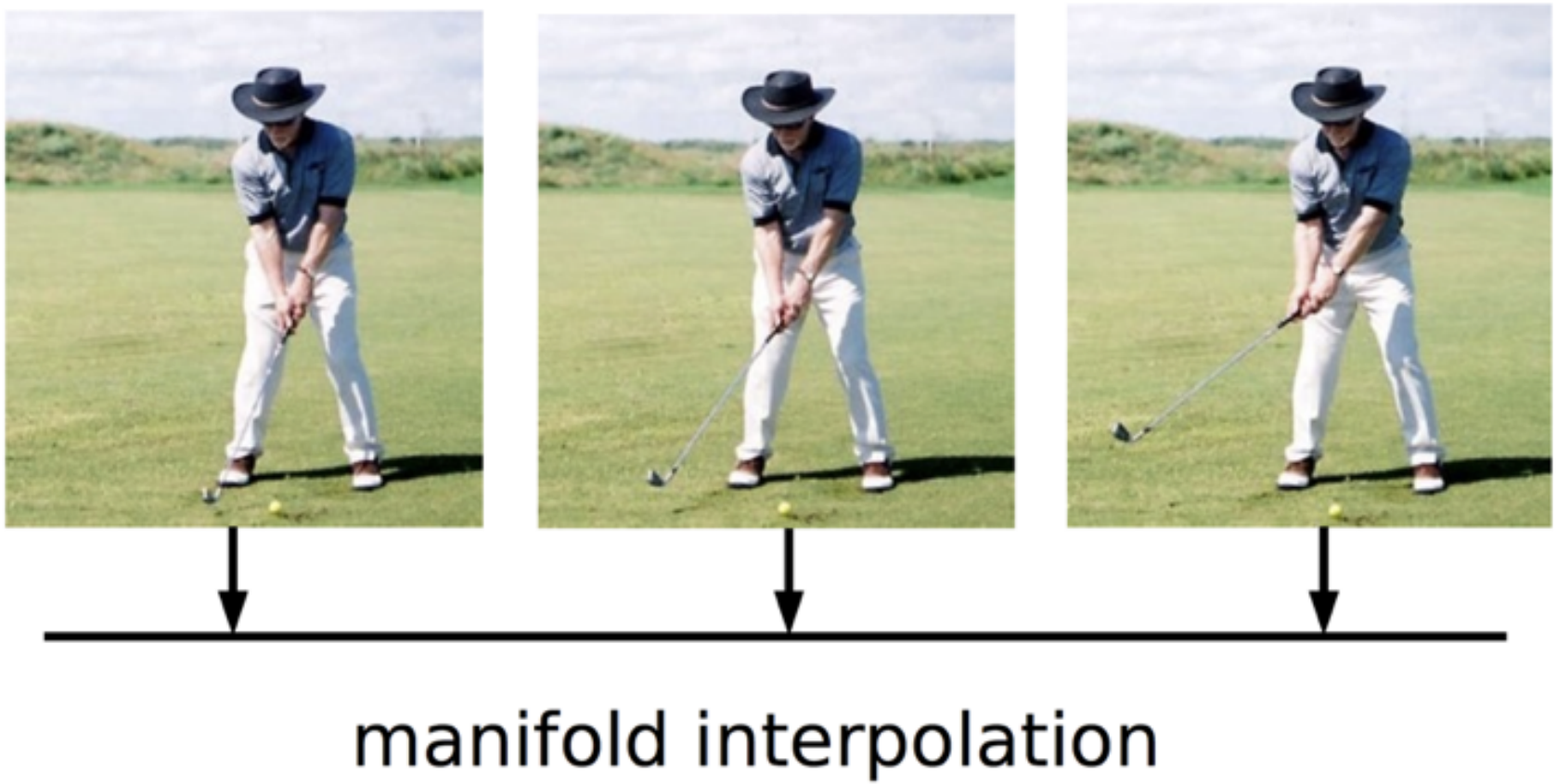
Reasonable distance metrics



Reasonable distance metrics



Reasonable distance metrics



Compressed description $\mathbf{z}(O)$ is constructed from a detailed description $\mathbf{x}(O)$ using training data

$$\mathbf{X}_m = \{\mathbf{x}_i = \mathbf{x}(O_i), i = 1, 2, \ldots, m\} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$$

of features for objects $O_1, O_2, \ldots, O_m$
without using physical information about objects

Figure – Appearance variation

Embedding problem. Using a sample

$$\mathbf{X}_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^p$$

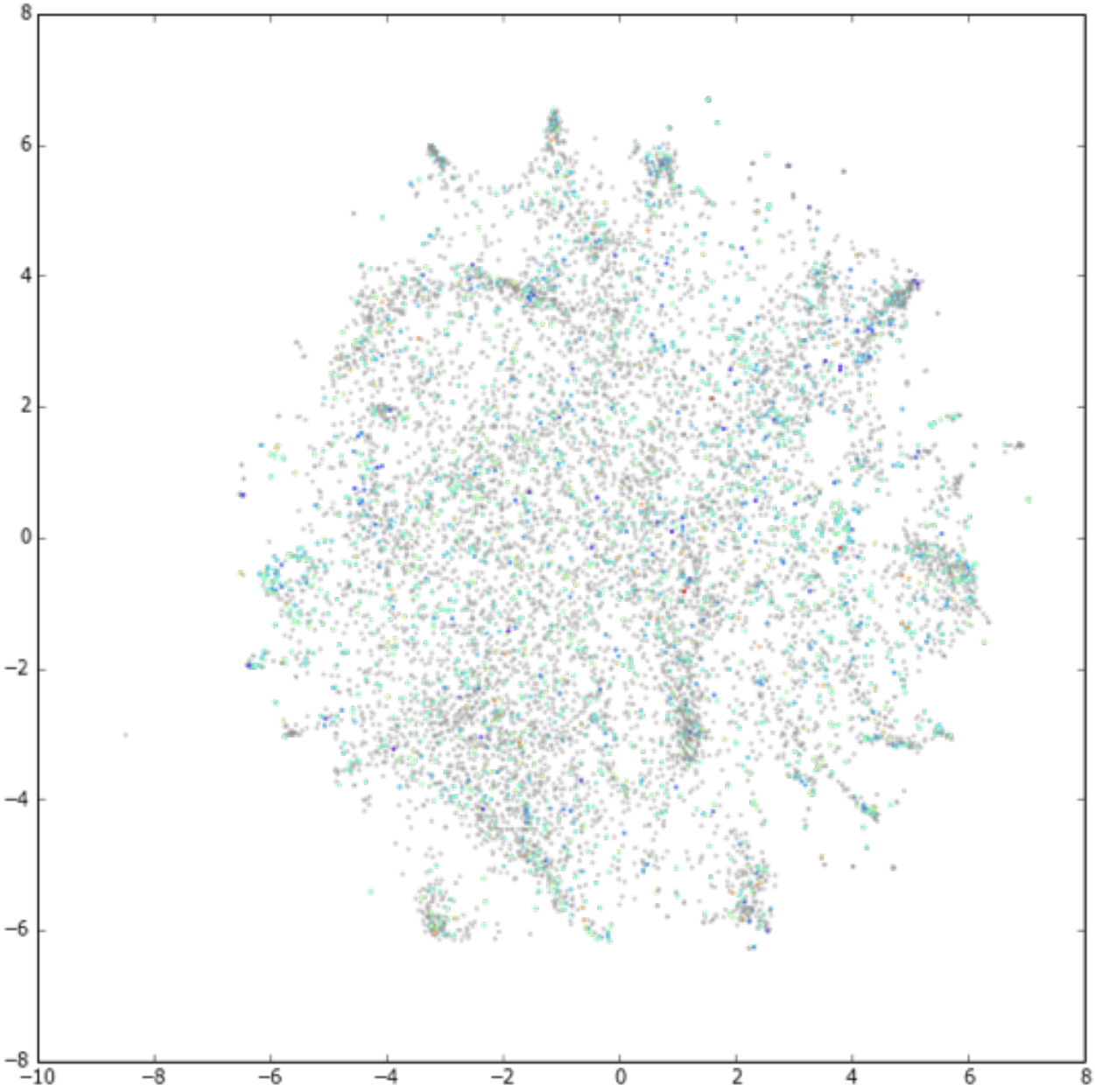
construct an embedding function

$$h : \mathbf{X}_m \to \mathbf{Z}_m = h(\mathbf{X}_m) = \{\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_m\} \subset \mathbb{R}^q$$

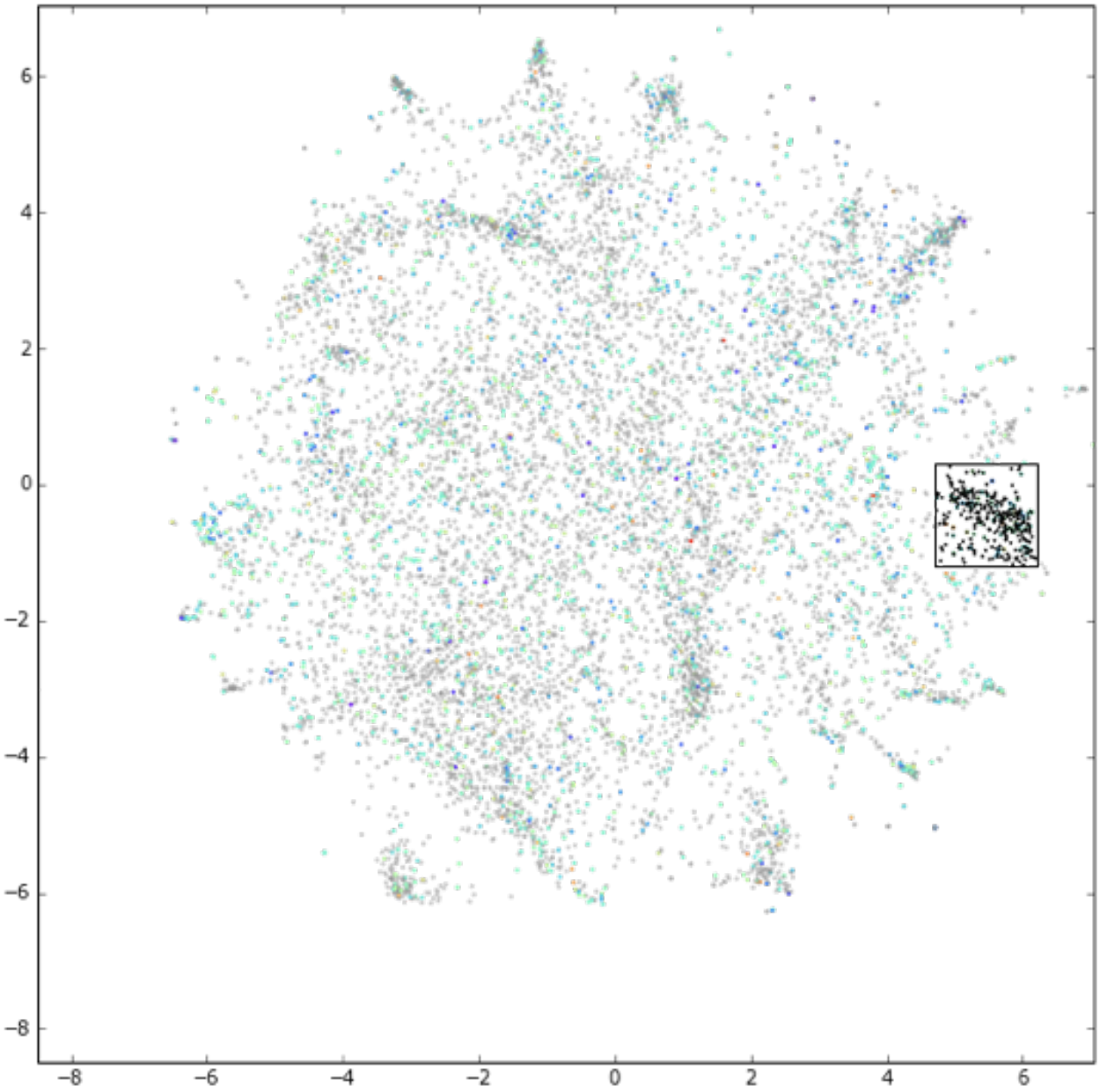
of sample points $\mathbf{X}_m$ in $\mathbb{R}^q$, $q < p$, such that the set $\mathbf{Z}_m$ "consistently represents" the set $\mathbf{X}_m$

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

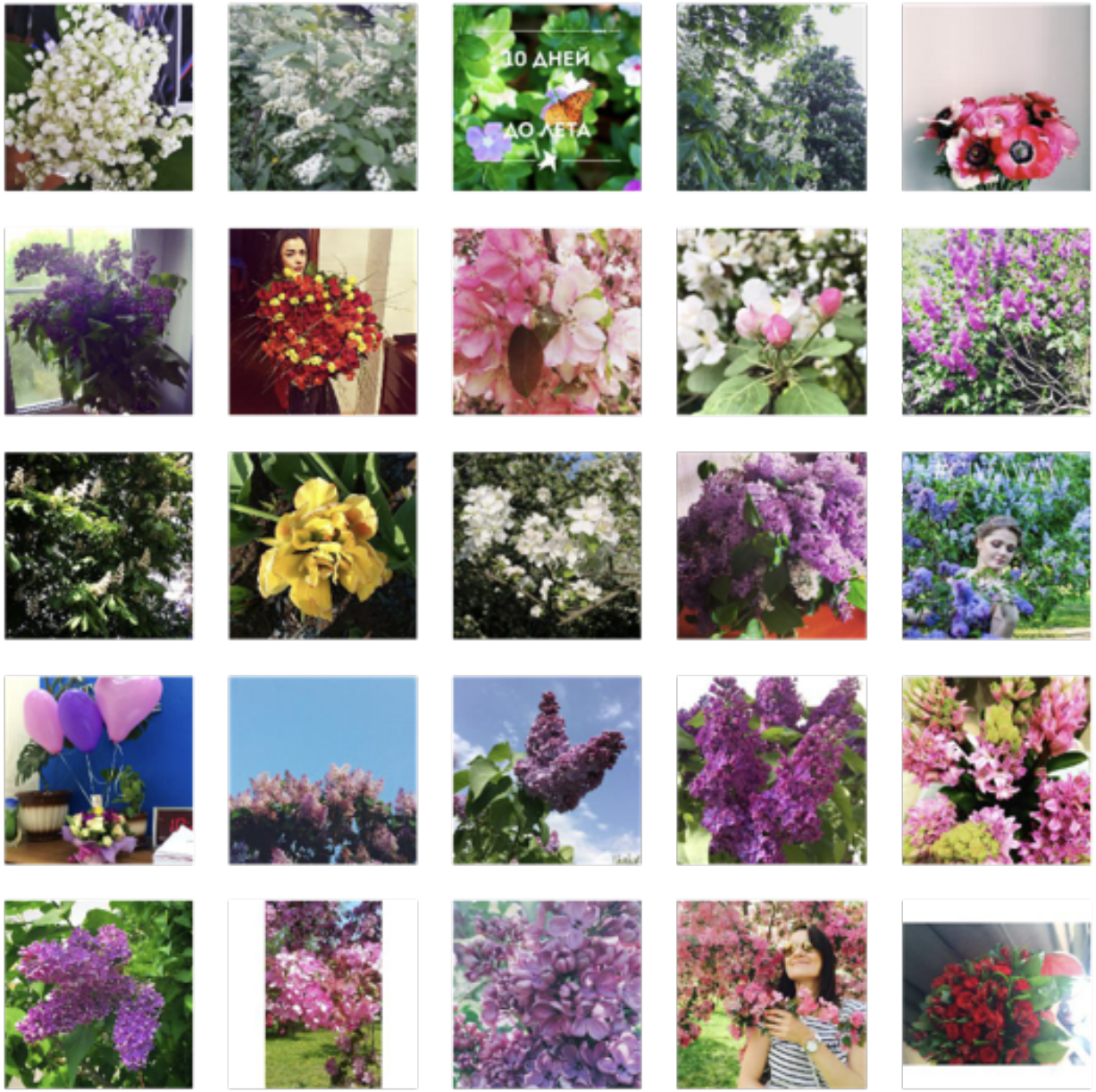
Embedding of Images



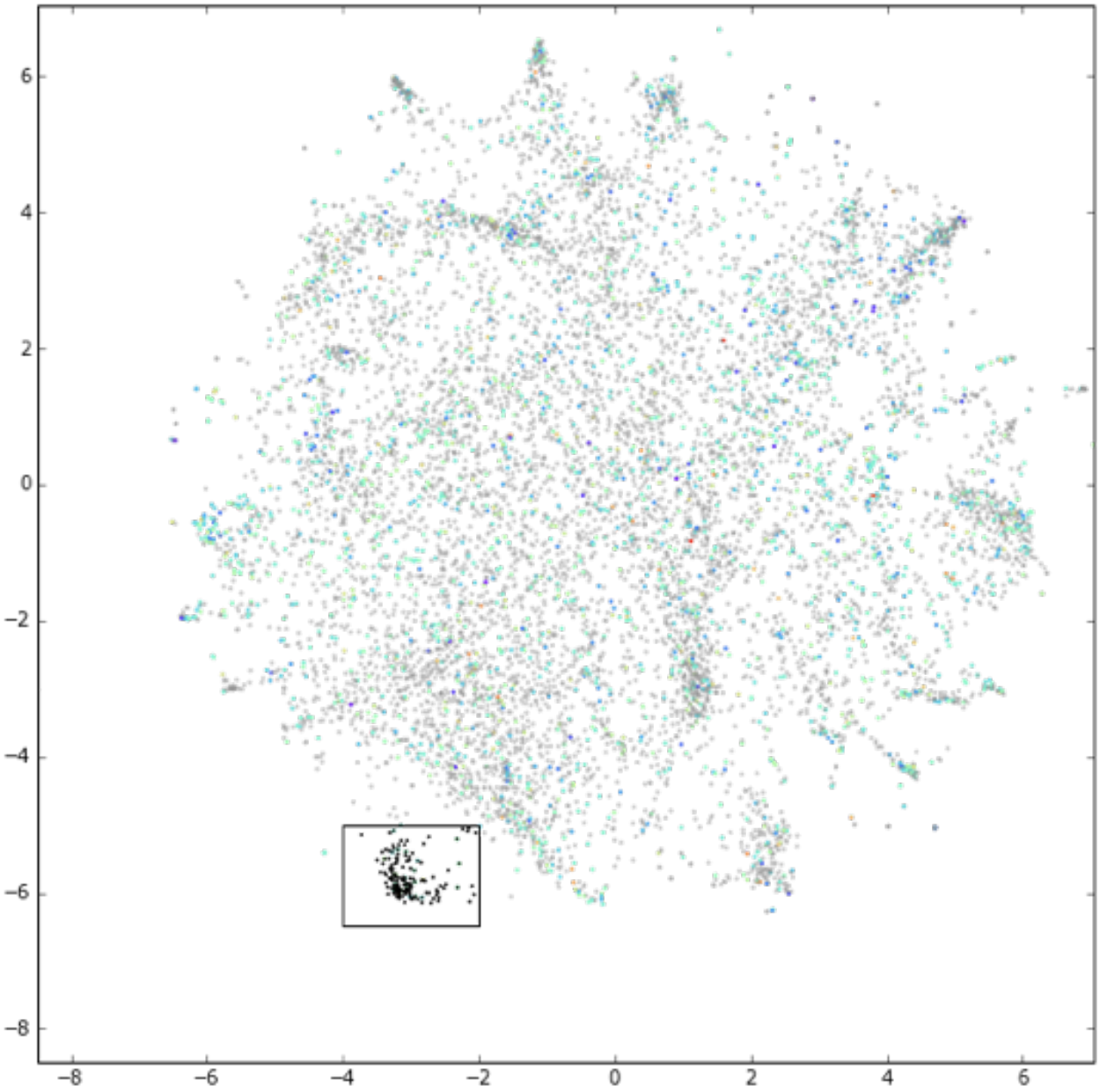
Embedding of Images



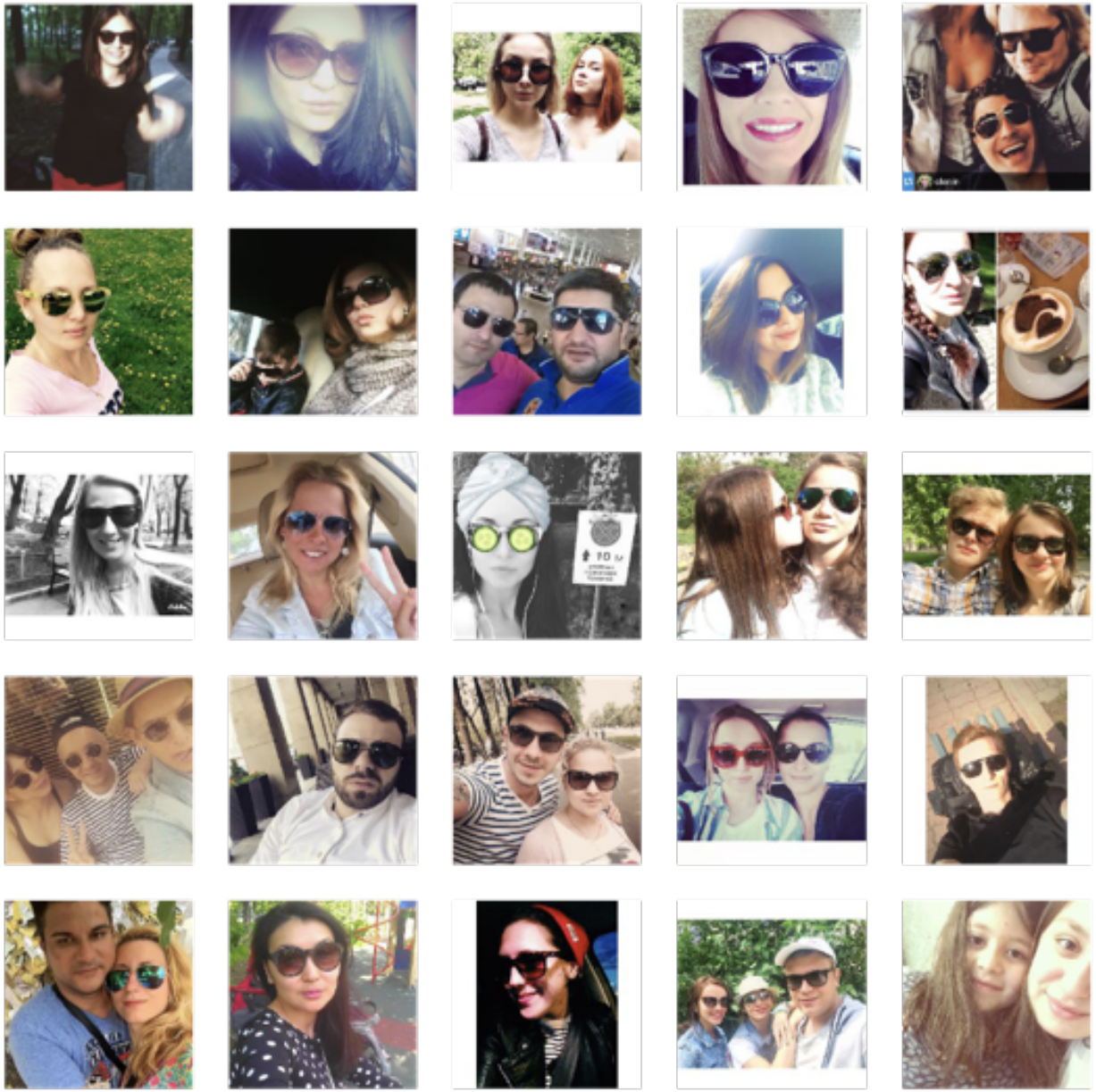
Embedding of Images



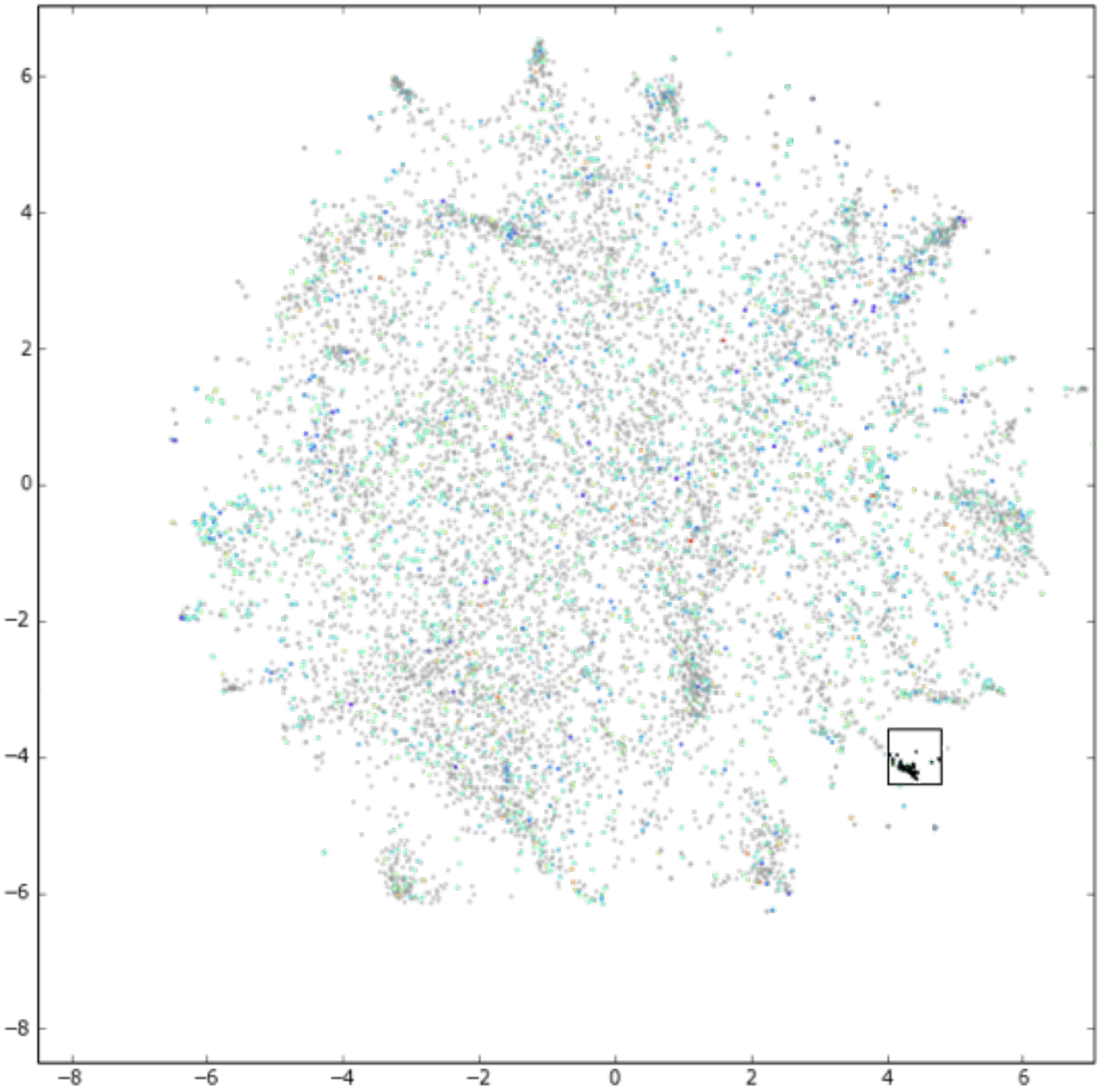
Embedding of Images



Embedding of Images



Embedding of Images



Embedding of Images

- **Typical Scheme:**
  - construct some cost function $L(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)$. E.g. for Multi Dimensional Scaling (**MDS**)

$$L(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) = \sum_{i,j} \left(\rho(\mathbf{x}_i, \mathbf{x}_j) - \|\mathbf{z}_i - \mathbf{z}_j\|\right)^2$$

  - optimize cost function
  - visually analyze results(Swiss Roll, Spiral, . . .)
- **Popular methods to solve embedding problem:**
  - Pursuit Projection; Principal Component Analysis
  - Kernel PCA (KPCA)
  - Locally Linear Embedding (LLE); Conformal Eigenmaps
  - Laplacian Eigenmaps (LE)
  - Hessian Eigenmaps (HE)
  - ISOmetric MAPping (ISOMAP); Landmark ISOMAP

- **Typical Scheme:**
  - construct some cost function $L(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)$. E.g. for Multi Dimensional Scaling (**MDS**)

$$L(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) = \sum_{i,j} \left(\rho(\mathbf{x}_i, \mathbf{x}_j) - \|\mathbf{z}_i - \mathbf{z}_j\|\right)^2$$

  - optimize cost function
  - visually analyze results(Swiss Roll, Spiral, ...)
- **Popular methods to solve embedding problem:**
  - Pursuit Projection; Principal Component Analysis
  - Kernel PCA (KPCA)
  - Locally Linear Embedding (LLE); Conformal Eigenmaps
  - Laplacian Eigenmaps (LE)
  - Hessian Eigenmaps (HE)
  - ISOmetric MAPping (ISOMAP); Landmark ISOMAP

Using a sample

$$\mathbf{X}_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^p$$

construct an embedding transformation

$$h : \mathbf{X} \subset \mathbb{R}^p \to \mathbf{Z} = h(\mathbf{X}) \subset \mathbb{R}^q$$

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

$$\mathbf{x}_{\text{new}} \in \mathbf{X}/\mathbf{X}_m$$

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

Example: face recognition

Extended Embedding Problem II

- For the extended embedding problem we need a Data Model $\mathbf{X}$: description of $\mathbf{X}$ and of a mechanism, which generates points $\mathbf{X}_m$ and new points $\mathbf{x}_{\text{new}}$ from $\mathbf{X}$
- Usually we assume that real data is well approximated by some manifold, i.e.

  Manifold Data Model:

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

  — open set $\mathbf{B}$ (inner coordinate space)
  — smooth bijective transformation $f : \mathbf{B} \to \mathbb{R}^p$

- **Manifold Learning Problem**

Extended Embedding Problem II

- For the extended embedding problem we need a Data Model $\mathbf{X}$: description of $\mathbf{X}$ and of a mechanism, which generates points $\mathbf{X}_m$ and new points $\mathbf{x}_{\text{new}}$ from $\mathbf{X}$
- Usually we assume that real data is well approximated by some manifold, i.e.

  Manifold Data Model:

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

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

- Manifold Learning Problem

Extended Embedding Problem II

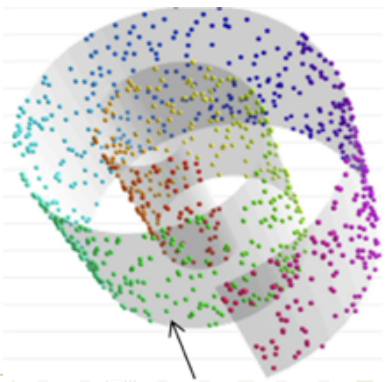
- For the extended embedding problem we need a Data Model $\mathbf{X}$: description of $\mathbf{X}$ and of a mechanism, which generates points $\mathbf{X}_m$ and new points $\mathbf{x}_{\text{new}}$ from $\mathbf{X}$
- Usually we assume that real data is well approximated by some manifold, i.e.

  Manifold Data Model:

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

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

- **Manifold Learning Problem**



Using a sample

$$\mathbf{X}_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^p$$

construct embedding transformation

$$h : \mathbf{X} \subset \mathbb{R}^p \to \mathbf{Z} = h(\mathbf{X}) \subset \mathbb{R}^q$$

and **Reconstruction transformation**

$$g : \mathbf{Z} \subset \mathbb{R}^q \to \mathbf{X} \subset \mathbb{R}^p$$

such that

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

Problem: find $\mathbb{R}^p$ an affine subspace

$$L(q) = \left\{\mathbf{x} \in \mathbb{R}^p : \mathbf{x} = \mathbf{x}_0 + \sum_{j=1}^{q} z_j \times \mathbf{e}_j,\ (z_1, z_2, \ldots, z_q) \in \mathbb{R}^q\right\}$$

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

$$\mathbf{X}_m = \left\{\mathbf{x}_i, i = 1, 2, \ldots, m\right\} \subset \mathbb{R}^p$$

in PCA: "the best" = minimize w.r.t. $\mathbf{x}_0, \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q\} \subset \mathbb{R}^p$

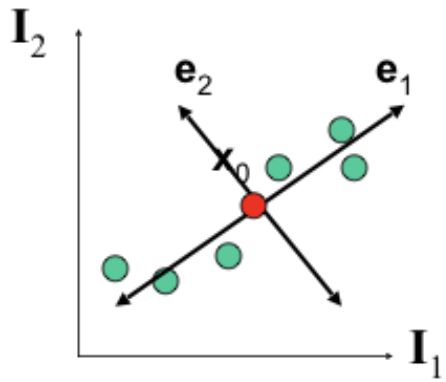
$$\frac{1}{m}\sum_{j=1}^{m} \left\| \mathbf{x}_j - P_{L(q)}\mathbf{x}_j \right\|^2$$

$$\mathrm{Pr}_{L(q)}(\mathbf{x}) = \mathbf{x}_0 + \sum_{j=1}^{q} z_j(\mathbf{x}) \times \mathbf{e}_j, \quad z_j(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_j)$$

$\mathbf{x}_{\text{mean}}$ — empirical mean of $\{\mathbf{x}_i, i = 1, 2, \ldots, m\}$
$\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$ — eigenvectors of an empirical covariance $(p \times p)$-matrix

$$\Sigma = \frac{1}{m} \sum_{j=1}^{m} (\mathbf{x}_j - \mathbf{x}_{\text{mean}}) \times (\mathbf{x}_j - \mathbf{x}_{\text{mean}})^\top$$

providing orthonormal basis in $\mathbb{R}^p$



$$L(1) = \{\mathbf{x} \in R^2 : \mathbf{x} = \mathbf{x}_0 + z_1 \times \mathbf{e}_1, z_1 \in R^1\},\ p = 2,\ q = 1$$

Solution: $\mathbf{x}_0 = \mathbf{x}_{\text{mean}},\ L(q) = \mathbf{x}_0 \bigoplus \text{Span}(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q)$,

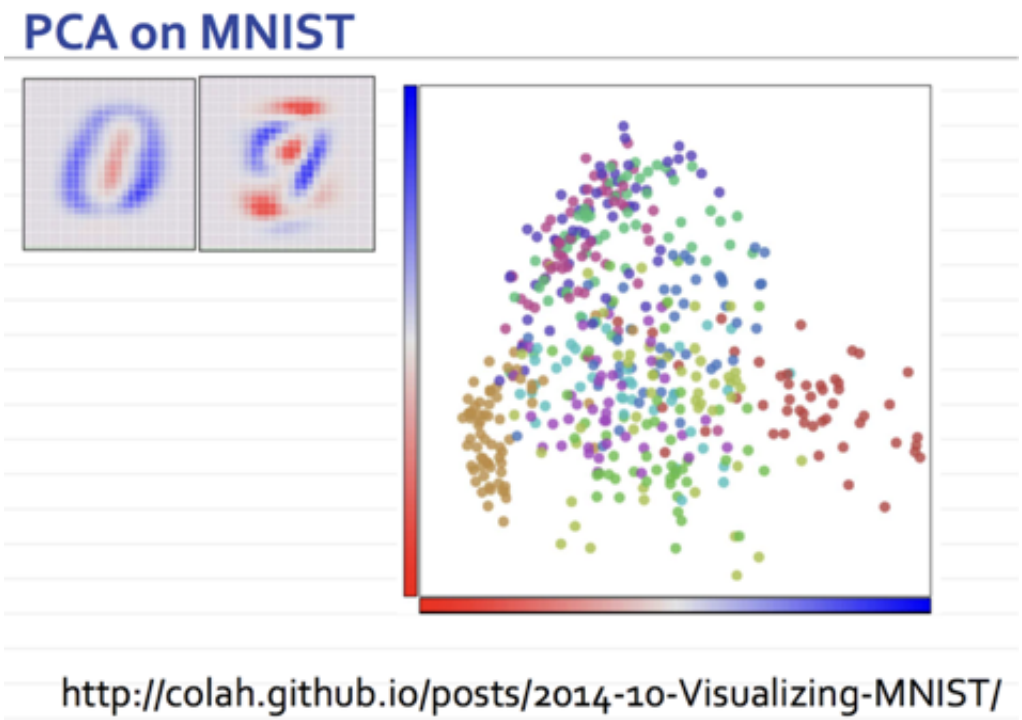
where orthonormal eigenvectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q\}$ of Σ correspond to q highest eigenvalues of this matrix

embedding procedure:

$$\mathbf{x} \in \mathbb{R}^p \to h(\mathbf{x}) = \big(z_1(\mathbf{x}), z_2(\mathbf{x}), \ldots, z_q(\mathbf{x})\big)^\top \in \mathbb{R}^q$$

reconstruction procedure:

$$\mathbf{z} = (z_1, z_2, \ldots, z_q)^\top \in \mathbb{R}^q \to g(\mathbf{z}) = \mathbf{x}_0 + z_1 \times \mathbf{e}_1 + z_2 \times \mathbf{e}_2 + \ldots + z_q \times \mathbf{e}_q$$



PCA for Face Recognition

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

$$\mathbf{x} \to \big((\mathbf{x} - \mathbf{x}_0, \mathbf{e}_1), (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_2), \ldots, (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_p)\big)^\top \in \mathbf{R}^p$$

where

$$z_1 = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_1),\ z_2 = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_2), \ldots, z_p = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_p)$$

- We select first q, $q \sim 10^2$ eigenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_q} \in \mathbb{R}^q$

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

$$\mathbf{x} \to \big((\mathbf{x} - \mathbf{x}_0, \mathbf{e}_1), (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_2), \ldots, (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_p)\big)^\top \in \mathbf{R}^p$$

where

$$z_1 = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_1),\ z_2 = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_2), \ldots, z_p = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_p)$$

- We select first q, $q \sim 10^2$ eigenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_q} \in \mathbb{R}^q$

$$\mathbf{x}^* \approx \mathbf{x}_0 + z_1 \times \mathbf{e}_1 + z_2 \times \mathbf{e}_2 + \ldots + z_q \times \mathbf{e}_q$$

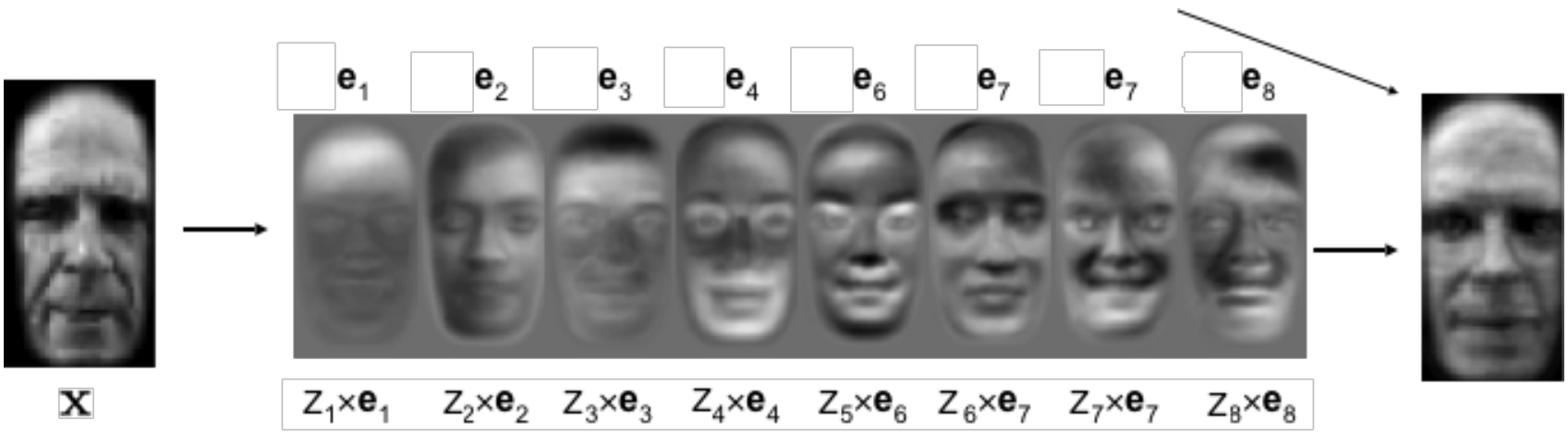
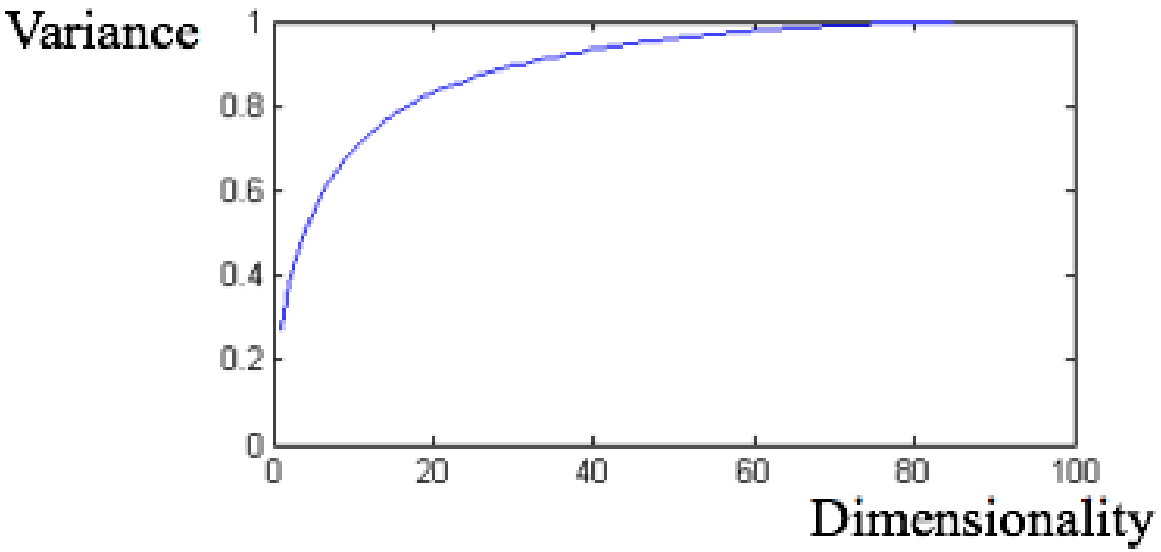


Figure – Left to right: original, reconstruction from 84, 40, 20, 3, 2 and 1 dimensions.



Minimize quadratic form

$$\sum_{i,j} \big(\rho(O_i, O_j) - \|\mathbf{z}_i - \mathbf{z}_j\|\big)^2$$

w.r.t. $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m \in \mathbb{R}^q$,
where ρ is some metric in the object feature space

$\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m \in \mathbb{R}^q$ are defined **except for shift and rotation**, thus we use normalization, e.g.

$$\mathbf{Z}^\top \times \mathbf{Z} = I_q \text{ and } \mathbf{Z}^\top \times \mathbf{1} = \mathbf{0},$$

where $\mathbf{Z}^\top = (\mathbf{z}_1 : \mathbf{z}_2 : \ldots : \mathbf{z}_m)$ — $(q \times m)$-matrix, $\mathbf{1} \in \mathbb{R}^m$ — vector of ones

$$\text{If } \rho(O_i, O_j) = \|\mathbf{x}(O_i) - \mathbf{x}(O_j)\|, \text{ then MDS} = \text{PCA}$$

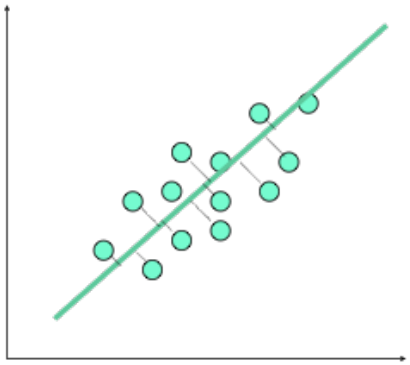
- We can use eigenvalue decomposition and keep eigenvectors that correspond to the largest eigenvalues
- This approach minimizes the squared error
- Cutting dimension of $\mathbf{z}$ leads to the embedding of smaller dimension
- Looking at eigenvalues leads to guess on the true intrinsic dimension

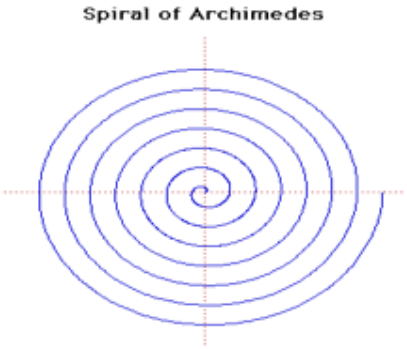
Note: the analytical solution is not possible, if the space is not Euclidean.

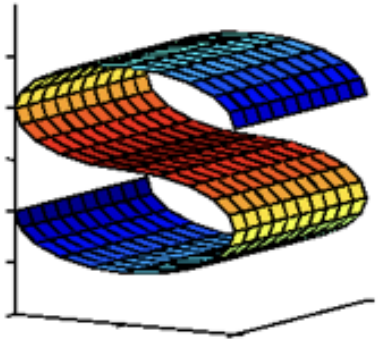
Sammon mapping idea: local distances are often more important to preserve

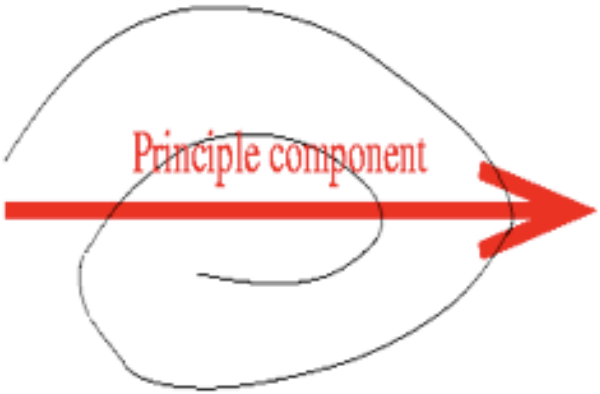
$$\sum_{i,j} \frac{\big(\rho(O_i, O_j) - \|\mathbf{z}_i - \mathbf{z}_j\|\big)^2}{\big(\rho(O_i, O_j)\big)^2}$$

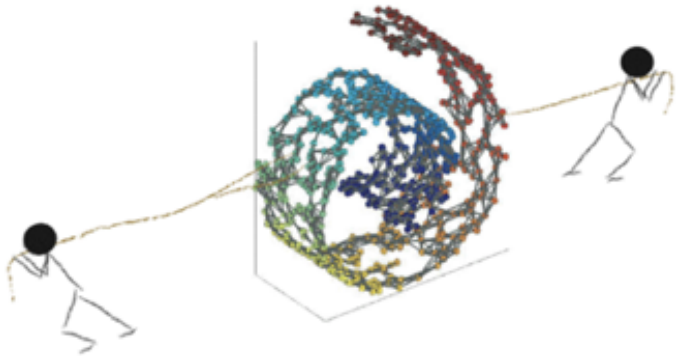
PCA, MDS (for "Euclidean proximity") — linear methods, do not work for nonlinear data distribution











Curse of dimensionality

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

$$\frac{V(r) - V\big(r(1-\epsilon)\big)}{V(r)} = \frac{1 - (1-\epsilon)^d}{1} \to 1$$

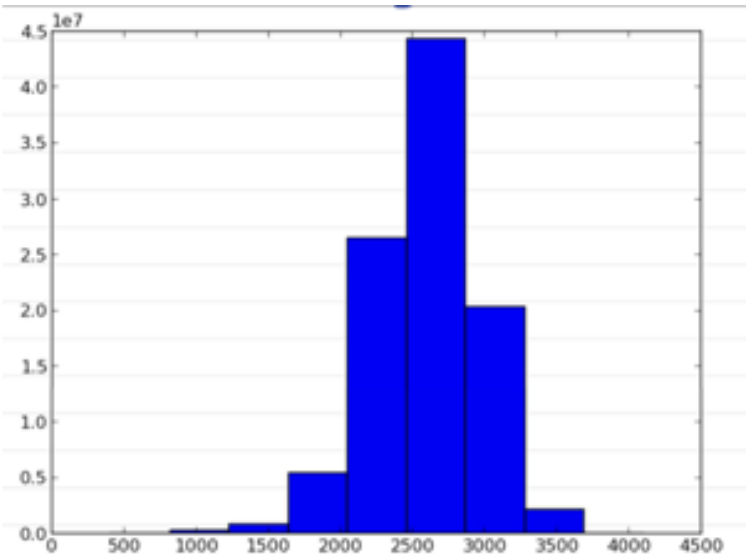


Figure – Histogram of pairwise Euclidean distances for MNIST

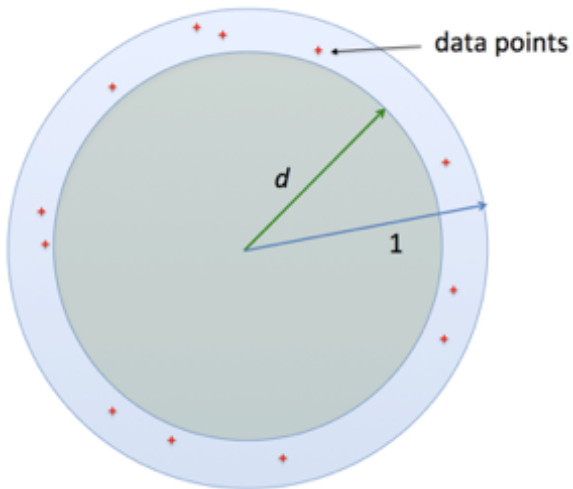


Figure – All volume of a ball is close to its cover

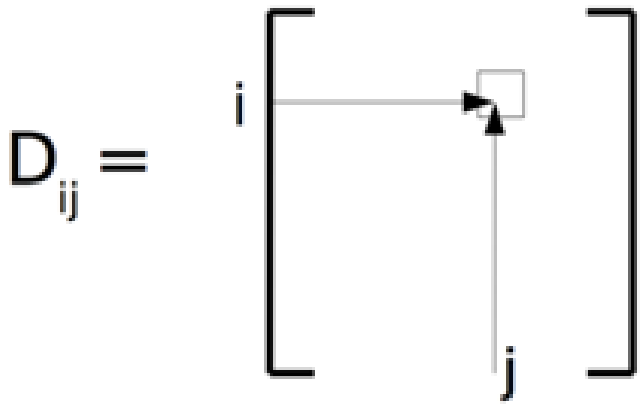
Consider smarter distance!

Nonlinear (local) DR methods

- **Locally Linear Embedding, LLE**
- **Laplacian Eigenmaps, LE**
- **Hessian Eigenmaps, HE**
- **ISOmetric MAPing, ISOMAP**
- Kernel PCA, KPCA - Spectral Embedding Algorithm, SEA
- Riemannian Manifold Learning, RML
- Local Tangent Space Alignment, LTSA ...

Usual MDS

For m data points, and a distance matrix D



. . . we can construct an q-dimensional space to preserve inter-points distances by using the top eigenvectors of D scaled by their eigenvalues

$$\mathbf{z}_i = [\sqrt{\lambda_1} v_1^i, \sqrt{\lambda_2} v_2^i, \dots \sqrt{\lambda_q} v_q^i]$$

Idea: local distance matrix

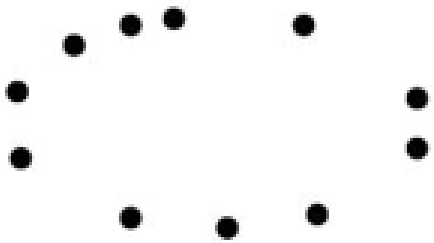
Infer a distance matrix using distances along the manifold.

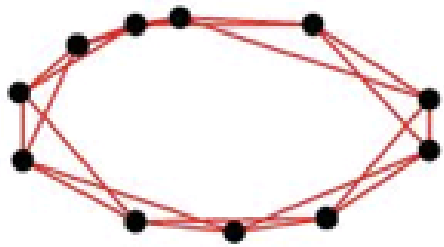
- **Step 1. Construct neighbourhoods** (ϵ—Neighbourhoods, or using k—Nearest Neighbours)

$$U(\mathbf{x}) = U(\mathbf{x}|\rho, \epsilon) = \{\mathbf{x}_i \in \mathbf{X}_m : \ \rho(\mathbf{x}, \mathbf{x}_i) \leq \epsilon\}$$

$$\Rightarrow \textbf{Graph } \Gamma(\mathbf{X}_m) = (\mathbf{X}_m, V) : (\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{V} \Leftrightarrow$$

$$\mathbf{x}_j \in U(\mathbf{x}_i) \text{ and } \mathbf{x}_i \in U(\mathbf{x}_j)$$





- **Step 2. Construct Weights**

$$w_{ij} = w(\mathbf{x}_i, \mathbf{x}_j) = w(v), v = (\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{V}$$

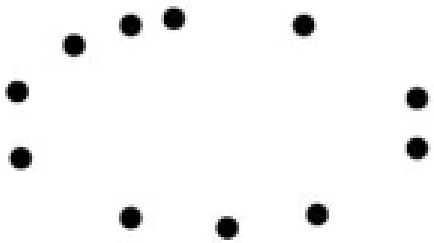
$$w_{0,ij} = 1 \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \in V, \ w_{0,ij} = 0 \text{ in other cases}$$

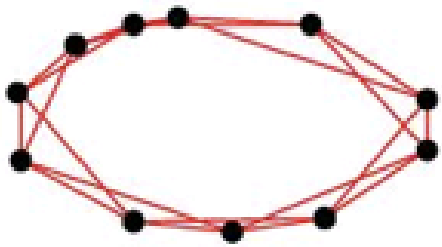
- **Step 1. Construct neighbourhoods** (ϵ—Neighbourhoods, or using k—Nearest Neighbours)

$$U(\mathbf{x}) = U(\mathbf{x}|\rho, \epsilon) = \{\mathbf{x}_i \in \mathbf{X}_m : \ \rho(\mathbf{x}, \mathbf{x}_i) \leq \epsilon\}$$

$$\Rightarrow \textbf{Graph } \Gamma(\mathbf{X}_m) = (\mathbf{X}_m, V) : (\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{V} \Leftrightarrow$$

$$\mathbf{x}_j \in U(\mathbf{x}_i) \text{ and } \mathbf{x}_i \in U(\mathbf{x}_j)$$





- **Step 2. Construct Weights**

$w_{ij} = w(\mathbf{x}_i, \mathbf{x}_j) = w(v), v = (\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{V}$

$$w_{0,ij} = 1 \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \in V, \ w_{0,ij} = 0 \text{ in other cases}$$

- Build a sparse graph with k-nearest neighbours

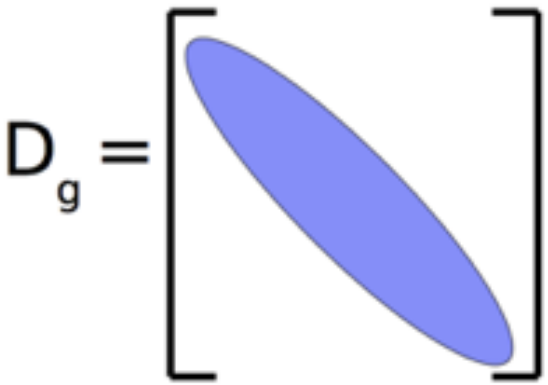
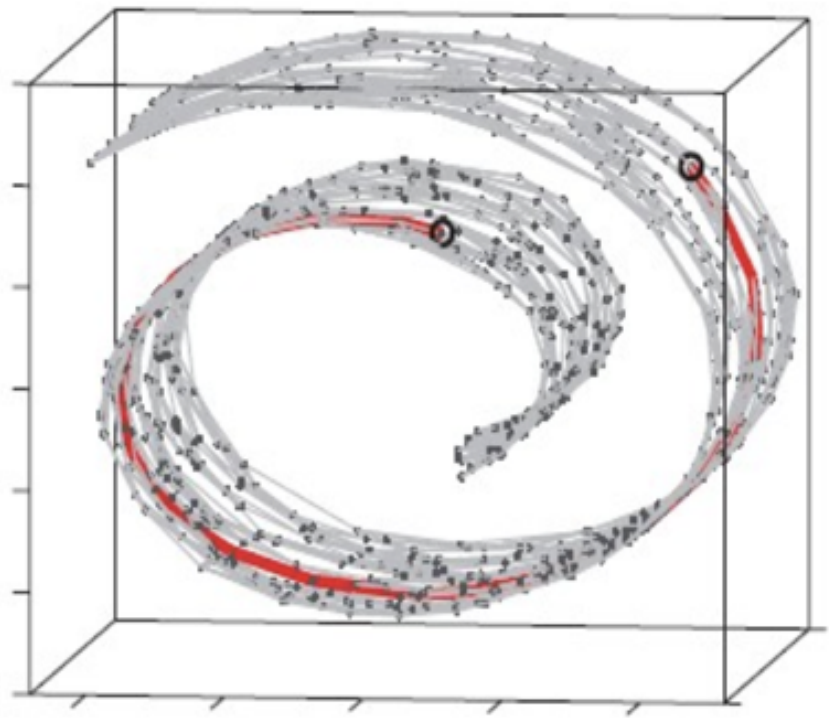
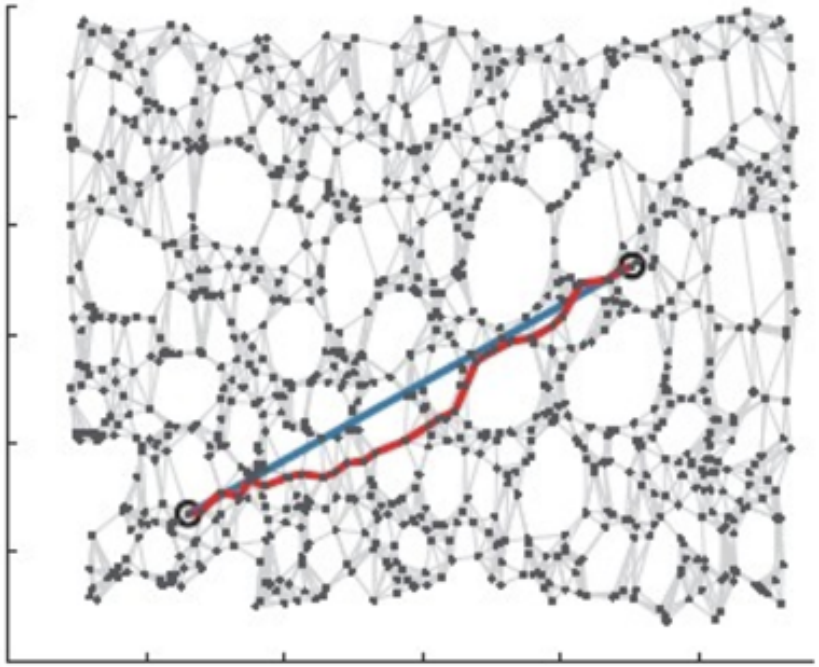


Figure – Distance matrix is sparse



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

$$D_g = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$



Usual MDS

- Build a low-D embedded space to best preserve the complete distance martix. Error function:

$$E = \|\tau(D_G) - \tau(D_Z)\|_{L^2}$$

where D_G — inner product distances in graph, D_Z — inner product distances in a new coordinate system

- Solution — set points Z to top eigenvectors D_G

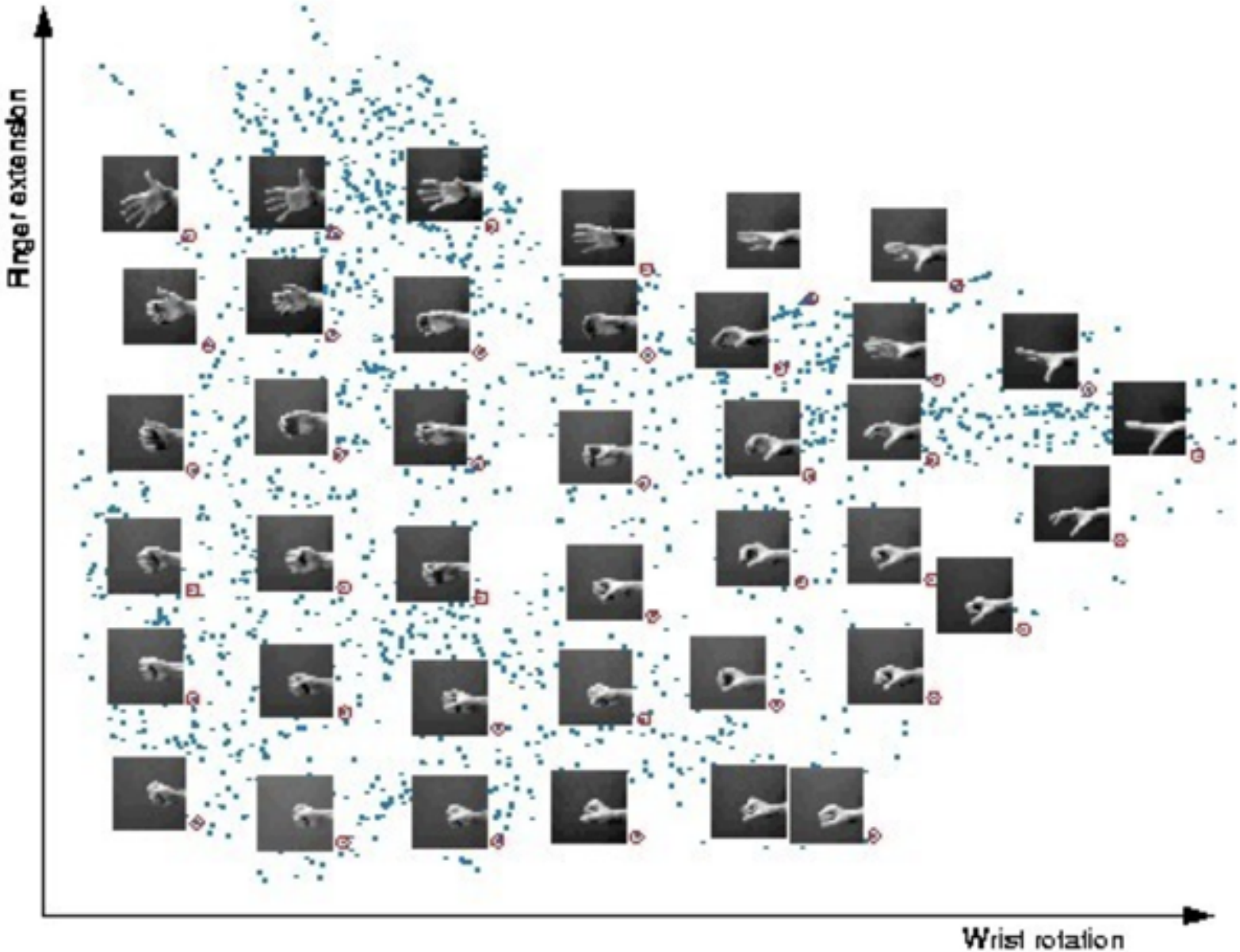
Usual MDS

- Build a low-D embedded space to best preserve the complete distance martix. Error function:

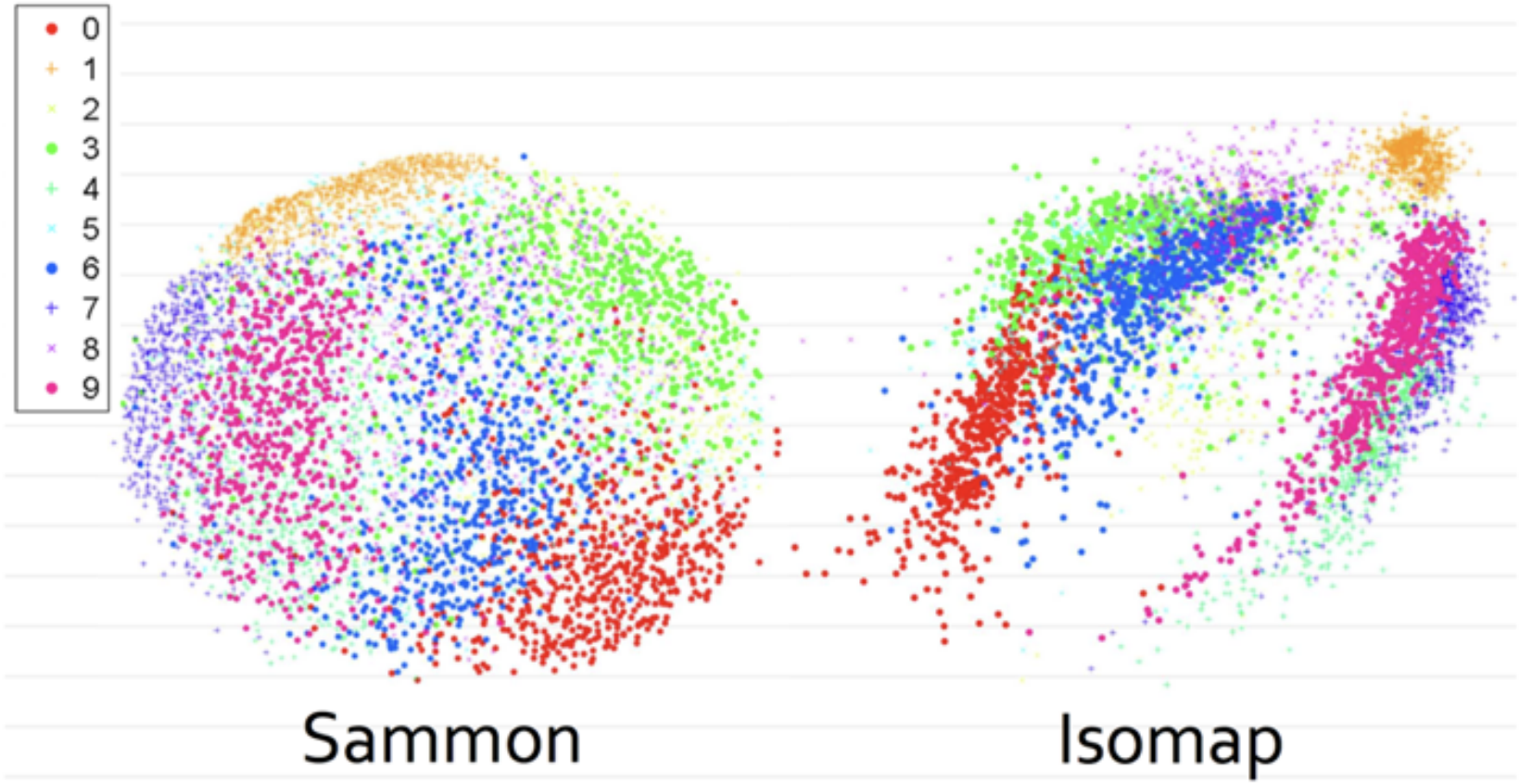
$$E = \|\tau(D_G) - \tau(D_Z)\|_{L^2}$$

  where D_G — inner product distances in graph, D_Z — inner product distances in a new coordinate system
- Solution — set points Z to top eigenvectors D_G

ISOMAP results: hands



MNIST: MDS variant Sammon vs. ISOMAP



Theoretical guarantees for ISOMAP

- If points are sampled uniformly from a "nice" manifold, then the shortest path distances approximate manifold geodesic distances for $m \to \infty$, $k \approx \log m$.
- For some assumptions about the manifold, ISOMAP recovers the embedding with the distortion converging to 0.

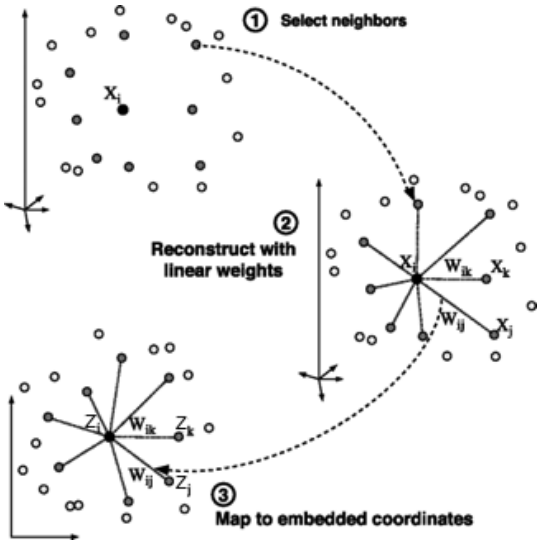
Note: the dimension is an issue. One should select the dimension larger, than the true dimension of the manifold.

ISOMAP: pros and cons

- Science paper from 2000
- 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



- **Step 1.** k-nearest neighbours
- **Step 2.** Get "Baricentric" coordinates $\{W_{ji}\}$ by minimizing

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

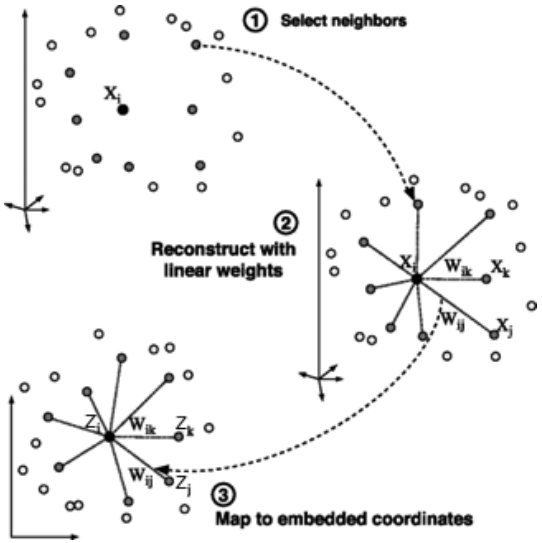
where $\mathbf{x}_{j(i)}$ is the j-th nearest neighbour of $\mathbf{x}_i$.

- Get $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m \in \mathbb{R}^q$ by minimizing

$$J_2(\mathbf{Z}_m) = \sum_{i=1}^{m} \left\| \mathbf{z}_i - \sum_{j=1}^{m} W_{ji} \mathbf{z}_{j(i)} \right\|^2$$

with standard normalizing conditions

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



- **Step 1.** k-nearest neighbours
- **Step 2.** Get "Baricentric" coordinates $\{W_{ji}\}$ by minimizing

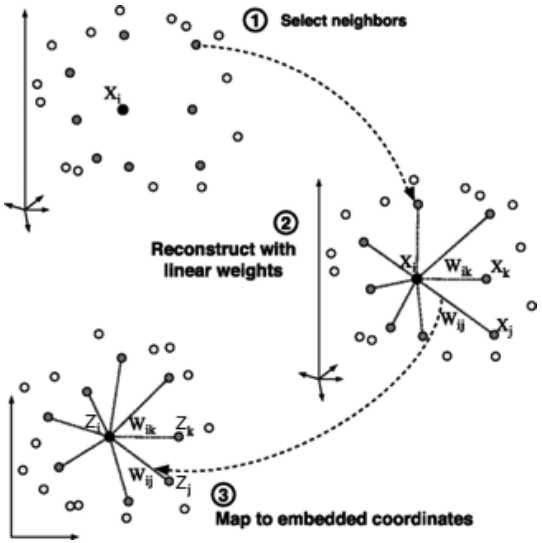
$$J_1(\mathbf{W}) = \sum_{i=1}^{m} \left\| \mathbf{x}_i - \sum_{j=1}^{K} W_{ji} \mathbf{x}_{j(i)} \right\|^2,$$

where $\mathbf{x}_{j(i)}$ is the j-th nearest neighbour of $\mathbf{x}_i$.

- Get $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m \in \mathbb{R}^q$ by minimizing

$$J_2(\mathbf{Z}_m) = \sum_{i=1}^{m} \left\| \mathbf{z}_i - \sum_{j=1}^{m} W_{ji} \mathbf{z}_{j(i)} \right\|^2$$

with standard normalizing conditions



- **Step 1.** k-nearest neighbours
- **Step 2.** Get "Baricentric" coordinates $\{W_{ji}\}$ by minimizing

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

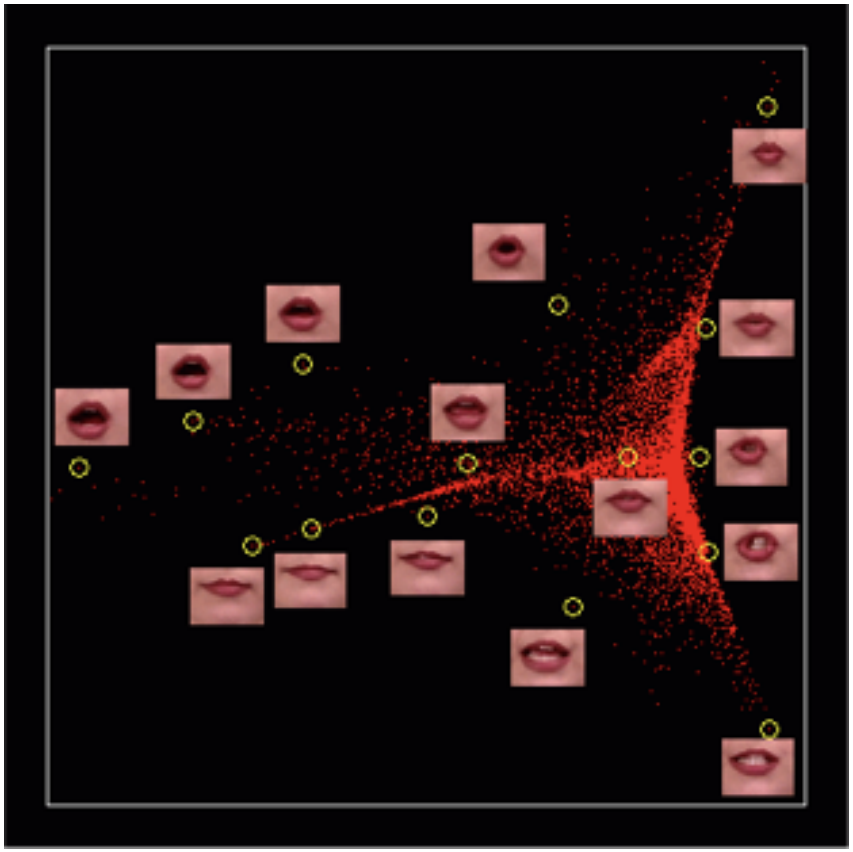
where $\mathbf{x}_{j(i)}$ is the j-th nearest neighbour of $\mathbf{x}_i$.

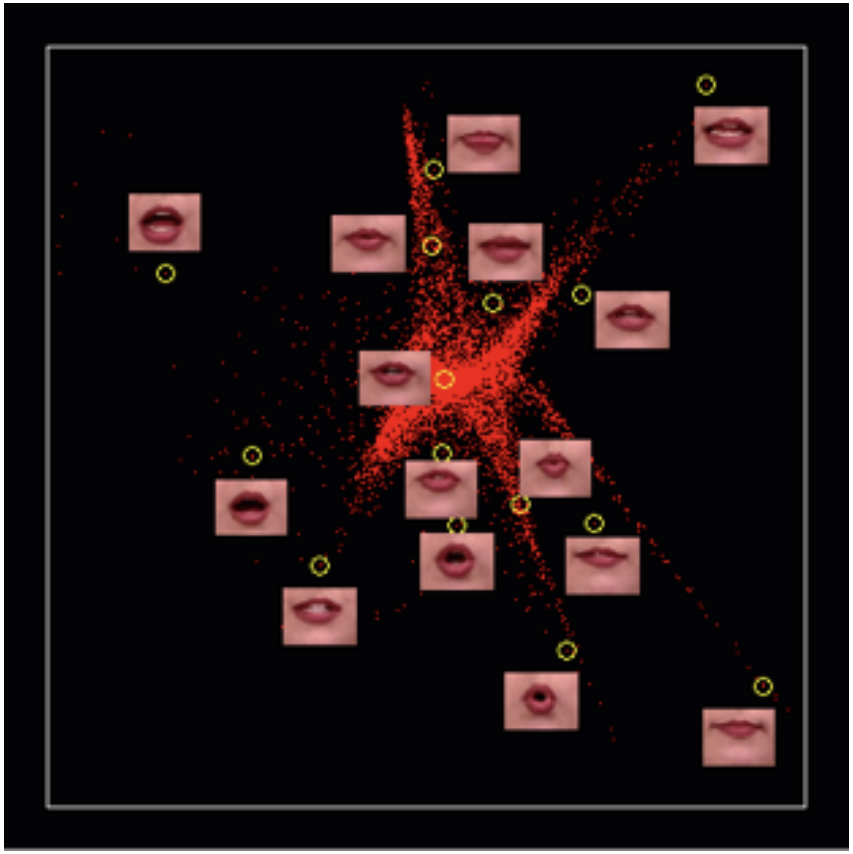
- Get $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m \in \mathbb{R}^q$ by minimizing

$$J_2(\mathbf{Z}_m) = \sum_{i=1}^{m} \left\| \mathbf{z}_i - \sum_{j=1}^{m} W_{ji} \mathbf{z}_{j(i)} \right\|^2$$

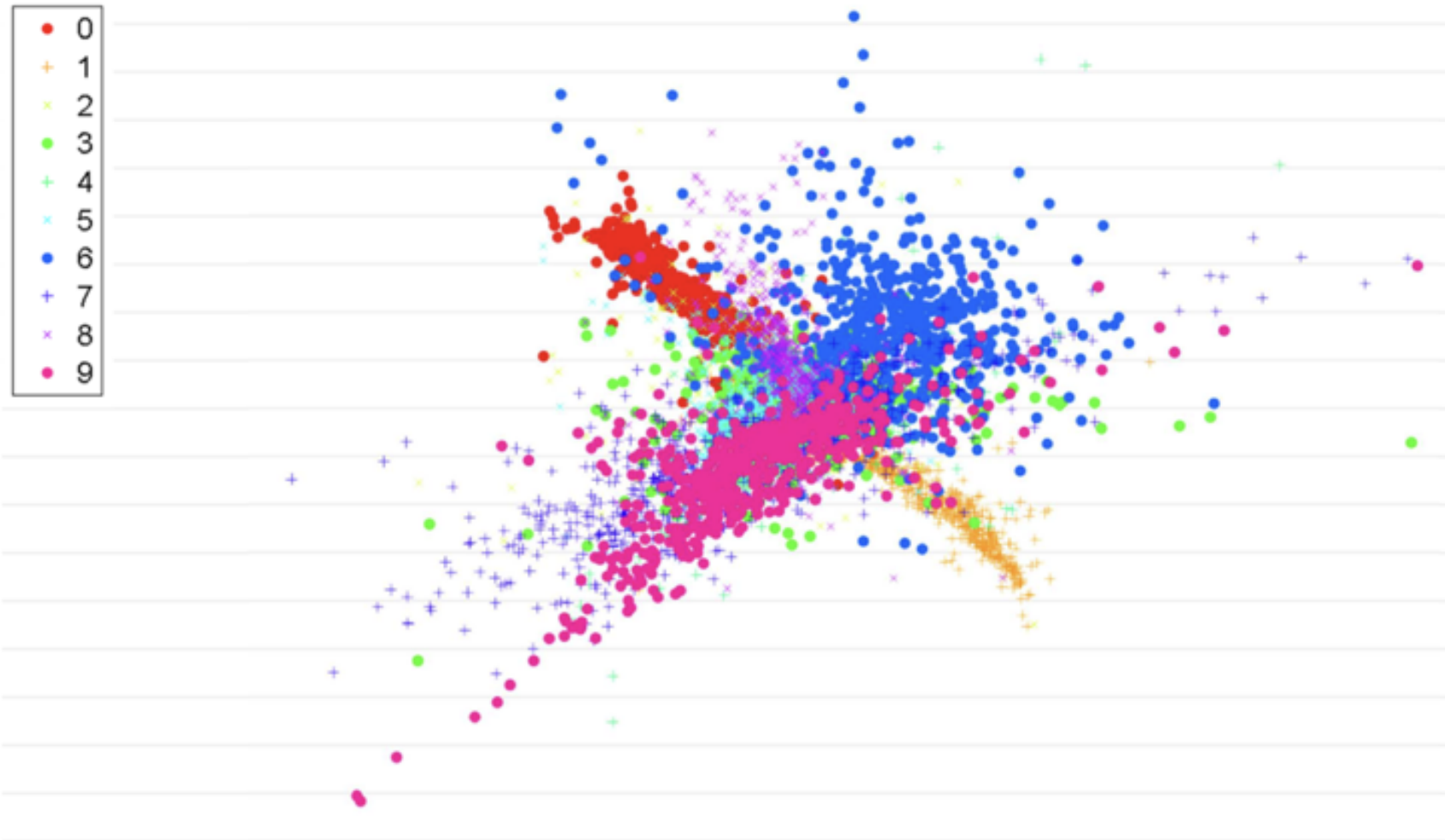
with standard normalizing conditions

LLE: Lips pictures





LLE: MNIST



LLE summary

Pros and cons

- no local minima, one free parameter
- incremental and fast
- simple linear algebra operations
- can distort global structure

Two more methods not covered here:

- **Riemannian Manifold Learning, RML**
- **Local Tangent Space Alignment, LTSA**

Stochastic Neighbour Embedding

- "Stochastic" similarity in the initial space, the probability to pick j-th object as a neighbour [Van der Maaten and Hinton]:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

- Similarity in the compressed space

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Cost function

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

Stochastic Neighbour Embedding

- "Stochastic" similarity in the initial space, the probability to pick j-th object as a neighbour [Van der Maaten and Hinton]:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

- Similarity in the compressed space

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Cost function

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

Stochastic Neighbour Embedding

- "Stochastic" similarity in the initial space, the probability to pick j-th object as a neighbour [Van der Maaten and Hinton]:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

- Similarity in the compressed space

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Cost function

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

$$\frac{\partial \text{Cost}}{\partial \mathbf{z}_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(\mathbf{z}_i - \mathbf{z}_j)$$

To escape poor minima:

- Initialize with Gaussian noise
- Add diminishing Gaussian noise on each stage

t-SNE: Similarity in the compressed space in case of a heavy-tailed Student t-distribution $t(1)$

$$q_{j|i} = \frac{(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{z}_k - \mathbf{z}_l\|^2)^{-1}}$$

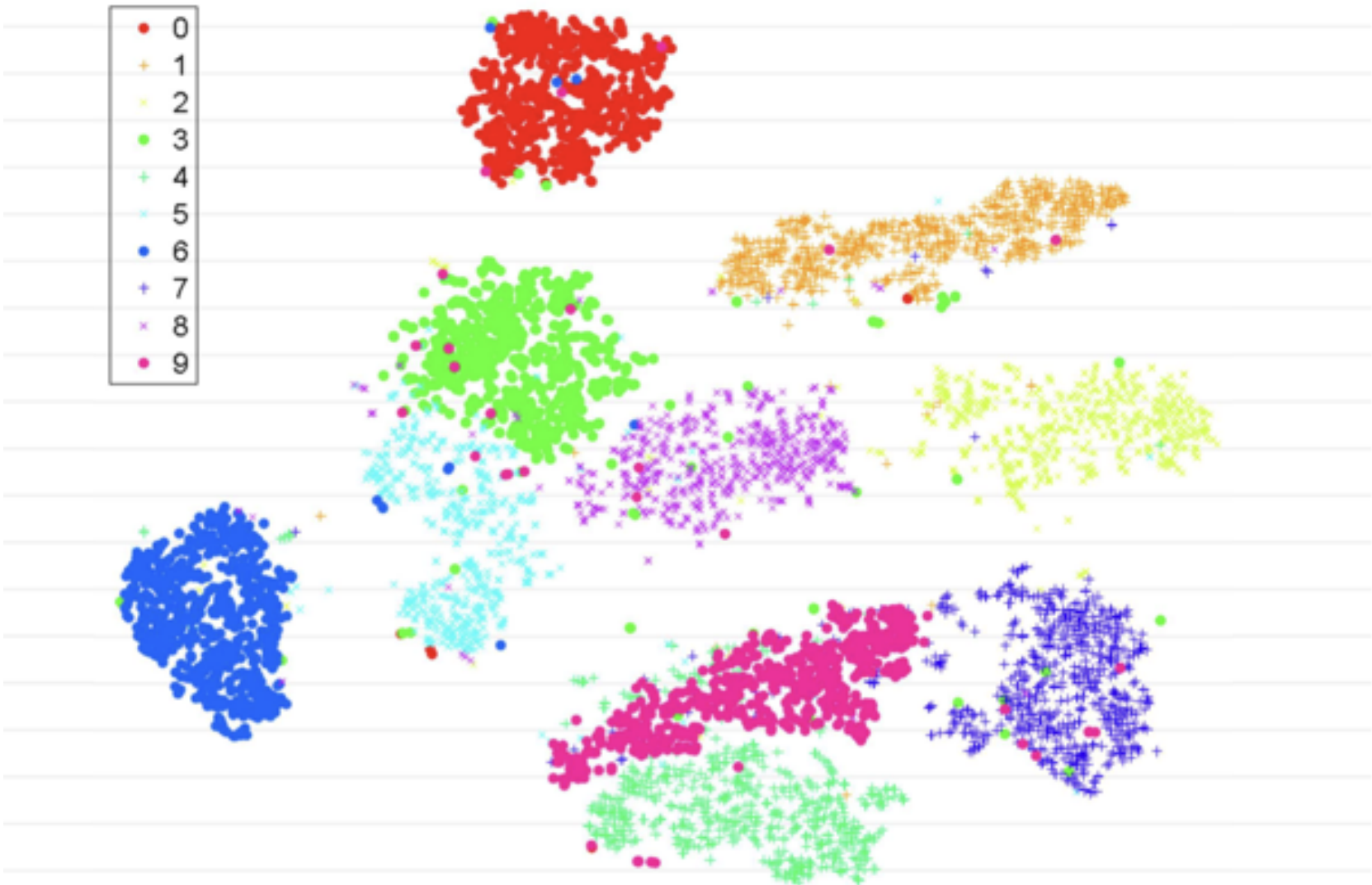
t-SNE: minimize KL between the whole densities P and Q

$$\text{Cost} = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

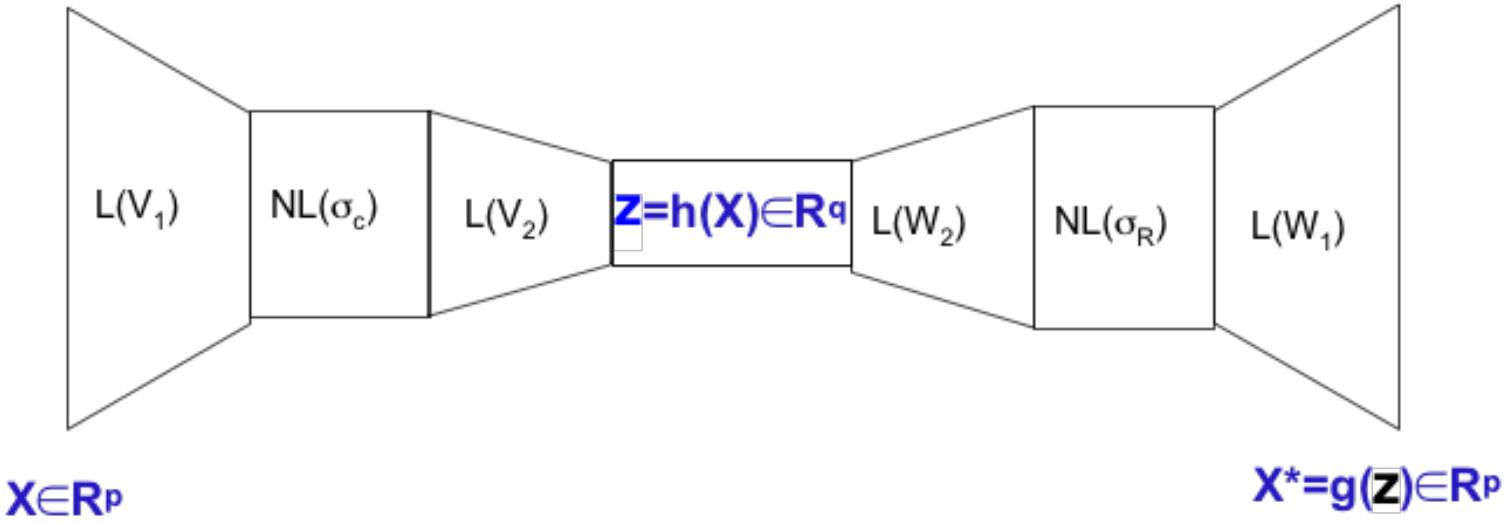
$$p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2n}$$

Minimization is done via gradient descent.

t-Stochastic Neighbor Embedding for MNIST



Autoencoder Neural Networks: encoder and decoder



The loss function is the reconstruction error:

$$R = \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{x}_j - \mathbf{x}_j^*\|^2.$$

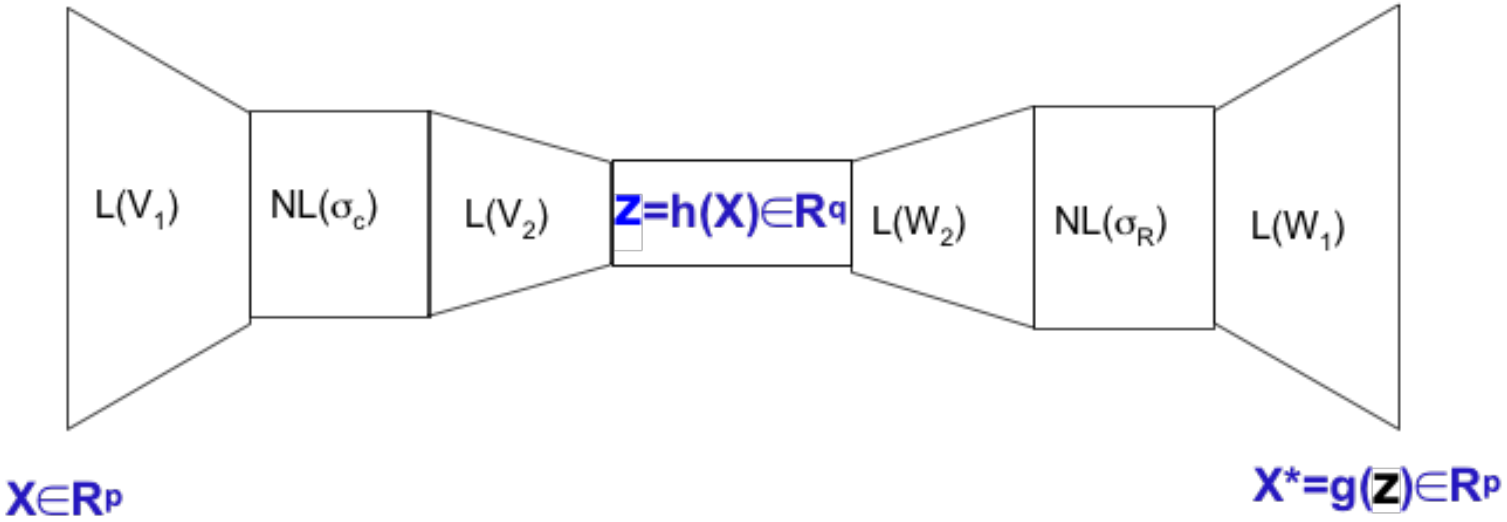
An architecture example:

- Four linear transformations with matrices V_1, V_2, W_1 and W_2
- Tow fixed **nonlinear** transformations

$$\sigma(x) - \textbf{sigmoid function}, \text{e.g.}, \sigma(x) = \frac{1}{(1+e^{-x})}$$

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

Autoencoder Neural Networks: encoder and decoder



The loss function is the reconstruction error:

$$R = \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{x}_j - \mathbf{x}_j^*\|^2.$$

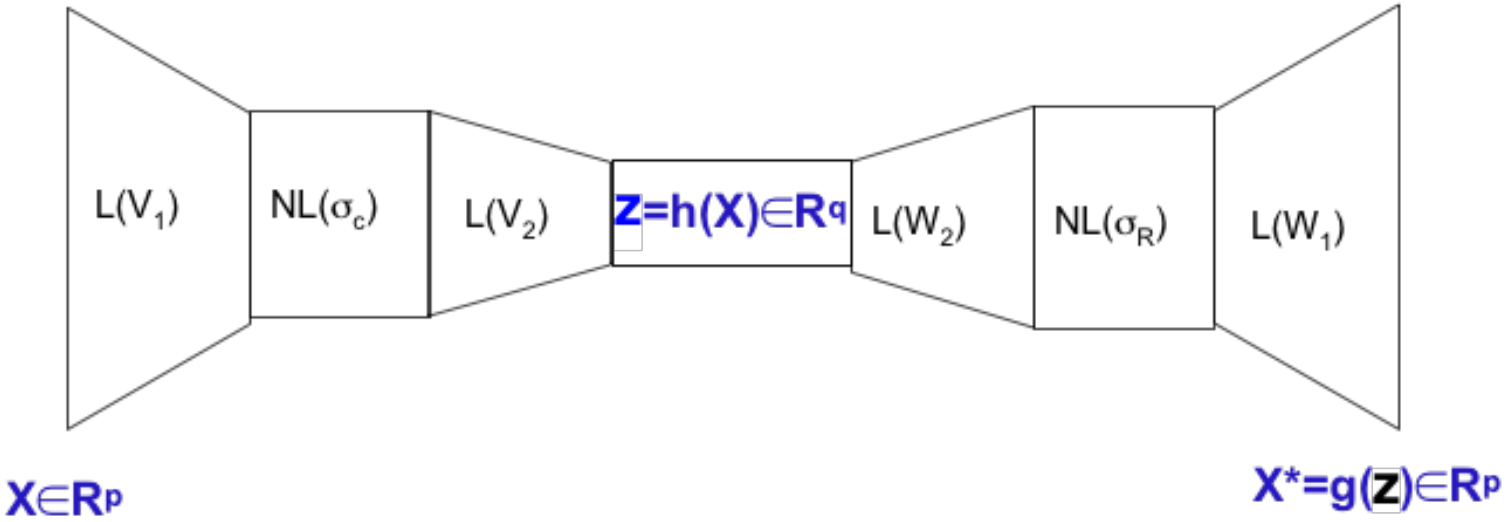
An architecture example:

- Four linear transformations with matrices V_1, V_2, W_1 and W_2
- Tow fixed **nonlinear** transformations

$$\sigma(x) - \textbf{ sigmoid function}, \text{ e.g., } \sigma(x) = \frac{1}{(1+e^{-x})}$$

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

Autoencoder Neural Networks: encoder and decoder



The loss function is the reconstruction error:

$$R = \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{x}_j - \mathbf{x}_j^*\|^2.$$

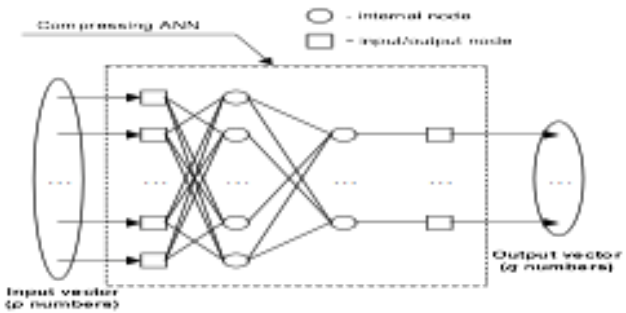
An architecture example:

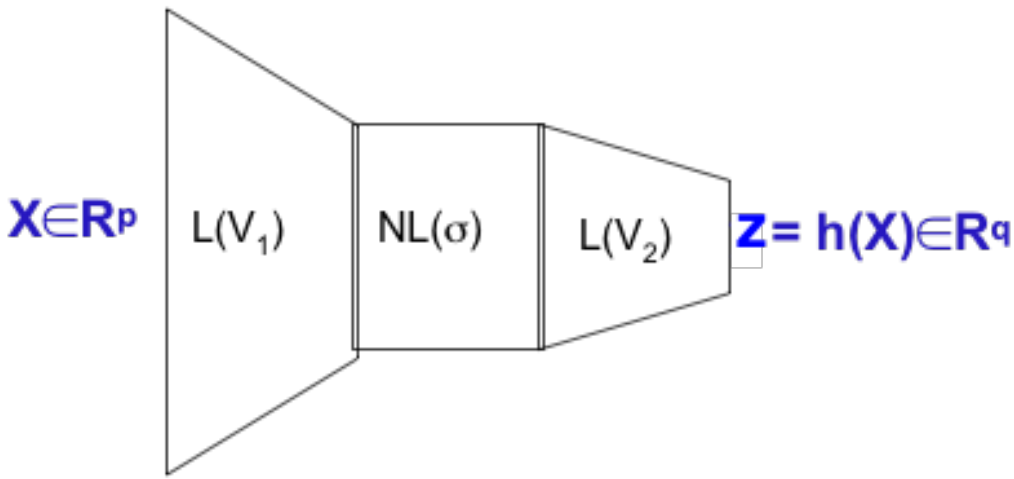
- Four linear transformations with matrices V_1, V_2, W_1 and W_2
- Tow fixed **nonlinear** transformations

$$\sigma(x) - \textbf{ sigmoid function}, \text{ e.g., } \sigma(x) = \frac{1}{(1 + e^{-x})}$$

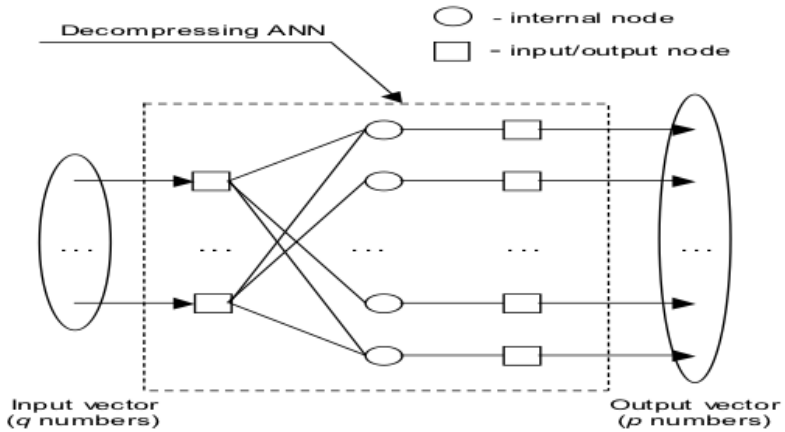
- Back-propagation to optimize w.r.t. to parameters V_1, V_2, W_1 and W_2 the reconstruction error R

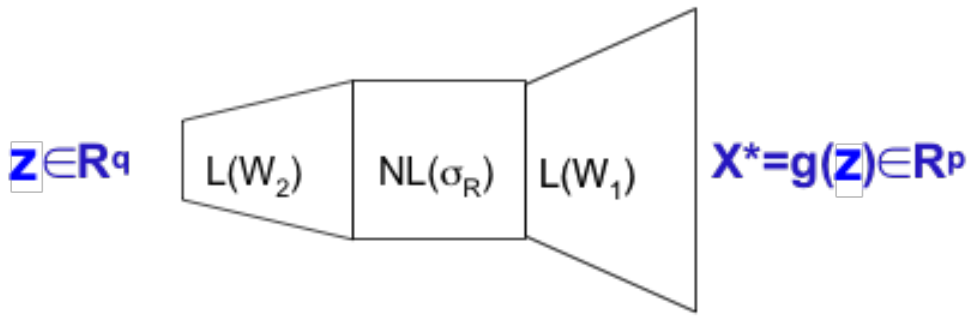
Compression Transformation





Reconstruction Transformation





Extensions of autoencoders

- Regularization: a variational autoencoder uses a Gaussian distribution as an embedding instead of the point embedding $\mathbf{z}$.
- Masking: mask some part of input and define the loss function as the reconstruction quality of the masked part

Contrastive learning loss function

Suppose that for each pair of objects $\mathbf{x}_i, \mathbf{x}_j$ we have a label y_{ij}:

- $y_{ij} = 1$, if this objects should be close in the embedding space
- $y_{ij} = 0$ otherwise.

The Triplet loss function uses a triple of objects i, j, k such that $y_{ij} = 1$ and $y_{ik} = 0$:

$$Loss(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = \max(d(\mathbf{z}_i, \mathbf{z}_j) - d(\mathbf{z}_i, \mathbf{z}_k) + m, 0),$$

m is the margin hyperparameter, $d(\mathbf{z}_i, \mathbf{z}_j)$ is the distance in the embedding space.

Example: Face recognition

Conclusion

- Interpretable *linear* methods: PCA, kernel PCA
- Fast and nice: t-SNE
- Take into account everything: neural networks-based