Dimension reduction for data visualization

Alexey Zaytsev,

Skoltech, CDISE

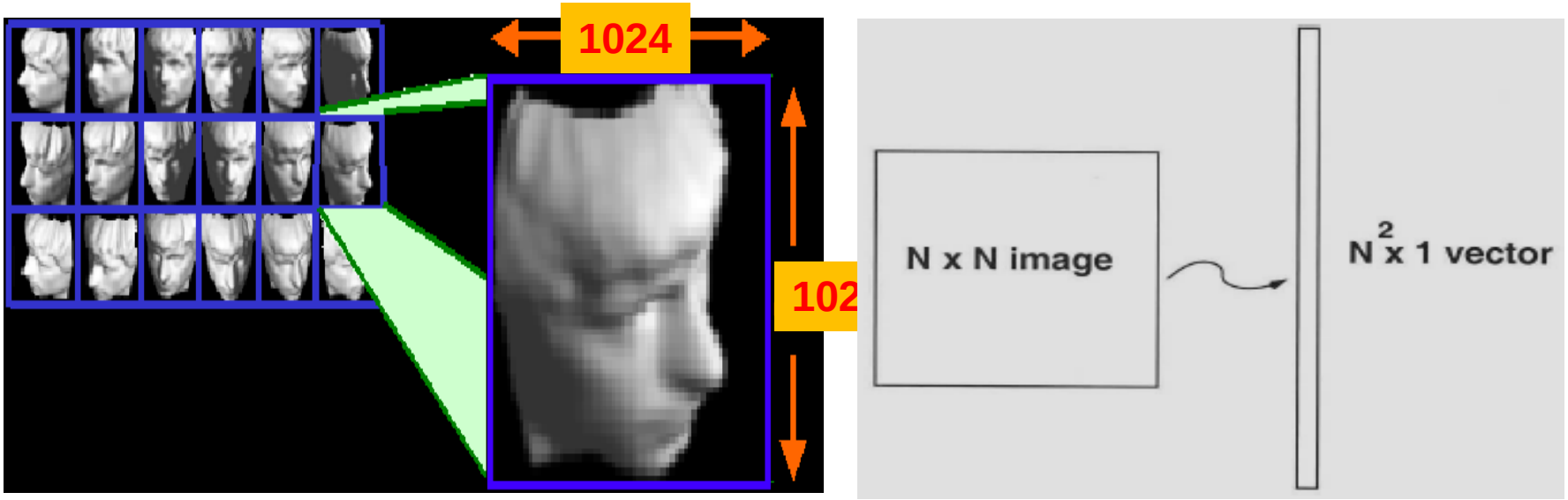
17 January

Some slides by Evgeny Burnaev are used
Also Ermek Kapushev helped during the preparation of these slides

Dimensionality Reduction Problem

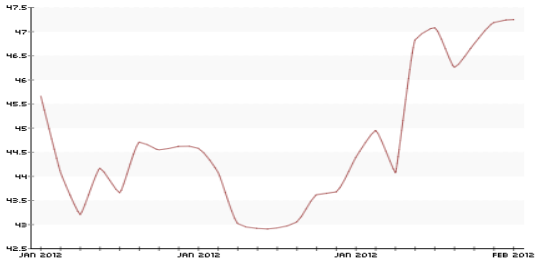
Object O ois described by p-мdimensional vector $X(O) \in \mathbb{R}^p$.
Components $X(O)$ are features O

Example 1 a face: pixel representation at a greyscale



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

Multidimensional time series



Electricity price

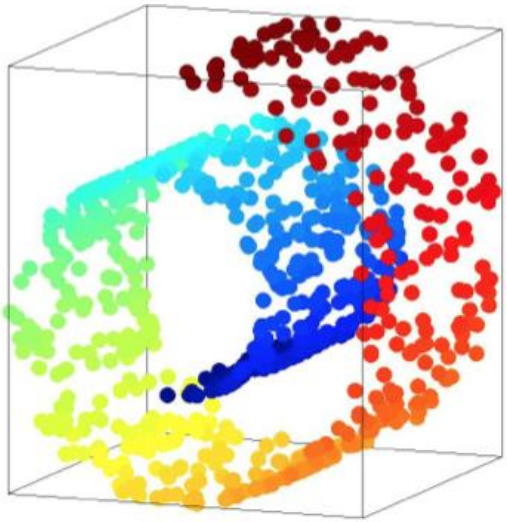
Speech recognition

- A curve f(x) is a vector

$$f = (f_1, \dots, f_p)^\top \in \mathbb{R}^p, f_j = f(t_j)$$

Example 2 (MNIST)

Example 3 Toy “Swiss Roll” Problem



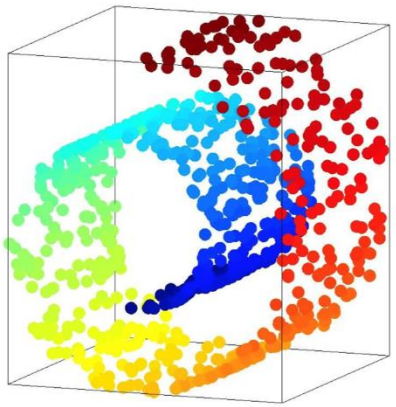
We need dimension reduction

- High dimensionality $\boldsymbol{p}$ of $X(\boldsymbol{O})$ is critical for efficient learning
- We can visualize only in 2D/3D

- Dimension Reduction = construct reduced dimension representation $y(\boldsymbol{O}) \in \mathbb{R}^q, q \ll p$, of $\boldsymbol{O}$ without "significant loss of information"

X → X' S.T.
dim(X') << dim(X)

uncovers the intrinsic dimensionality (invertible)



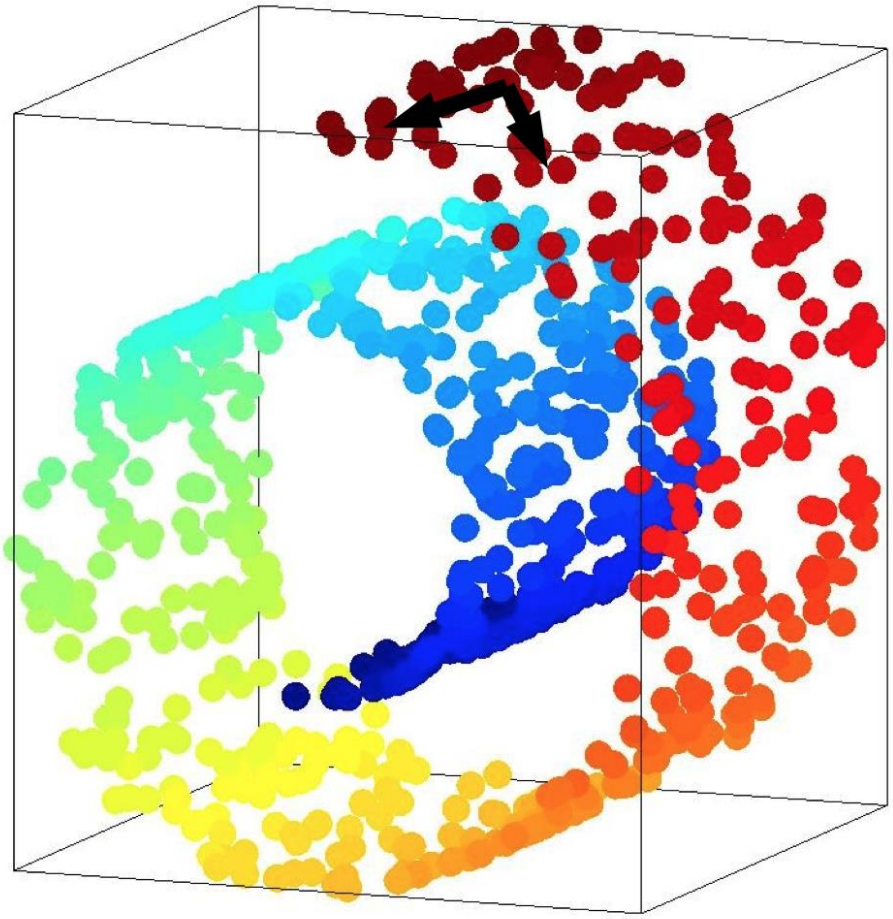
There are various purposes for usage of dimension reduction

- **Visualization**
- Data compression
- “curse of dimensionality”
- De-noising
- Reasonable distance metrics

What is “a smart distance”?

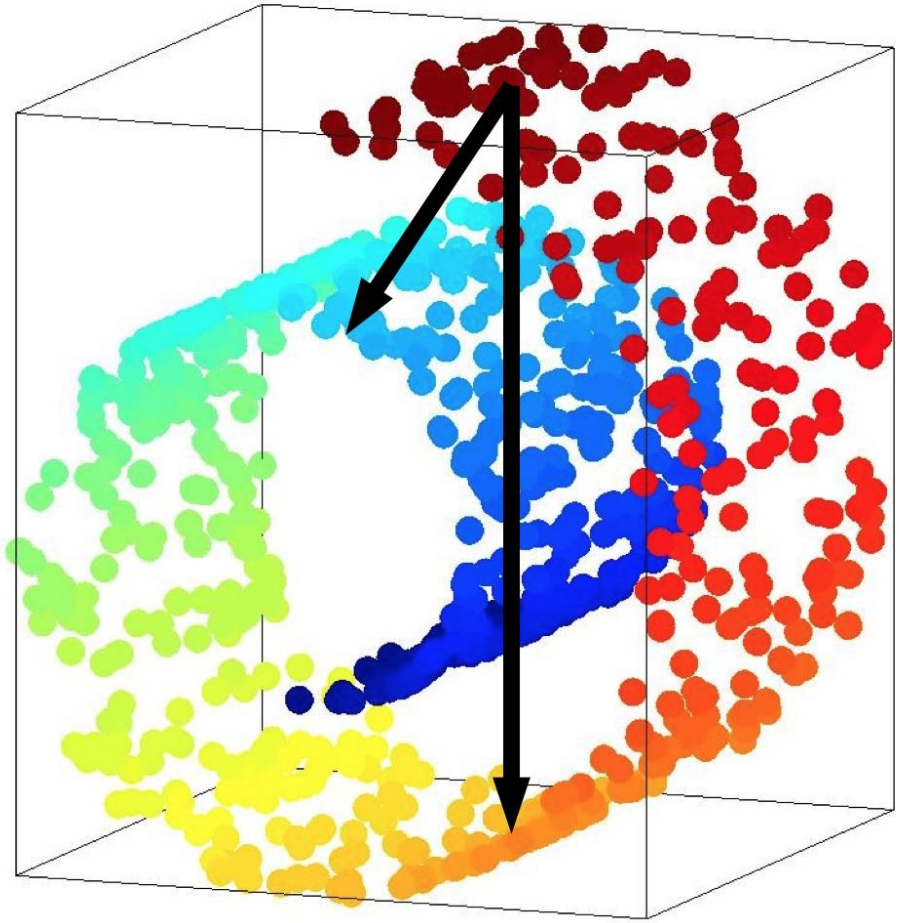
Деформация

What is “a smart distance”?

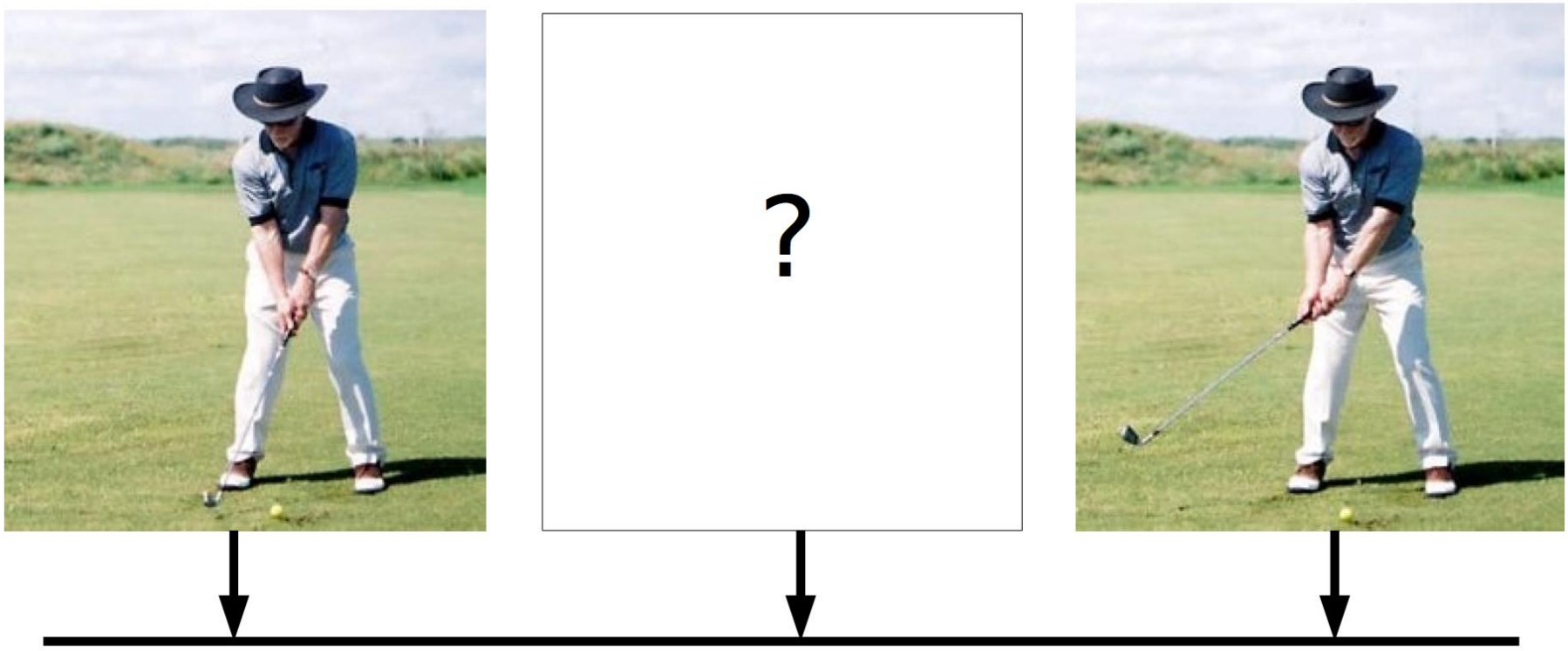


Skoltech
Skolkovo Institute of Science and Technology

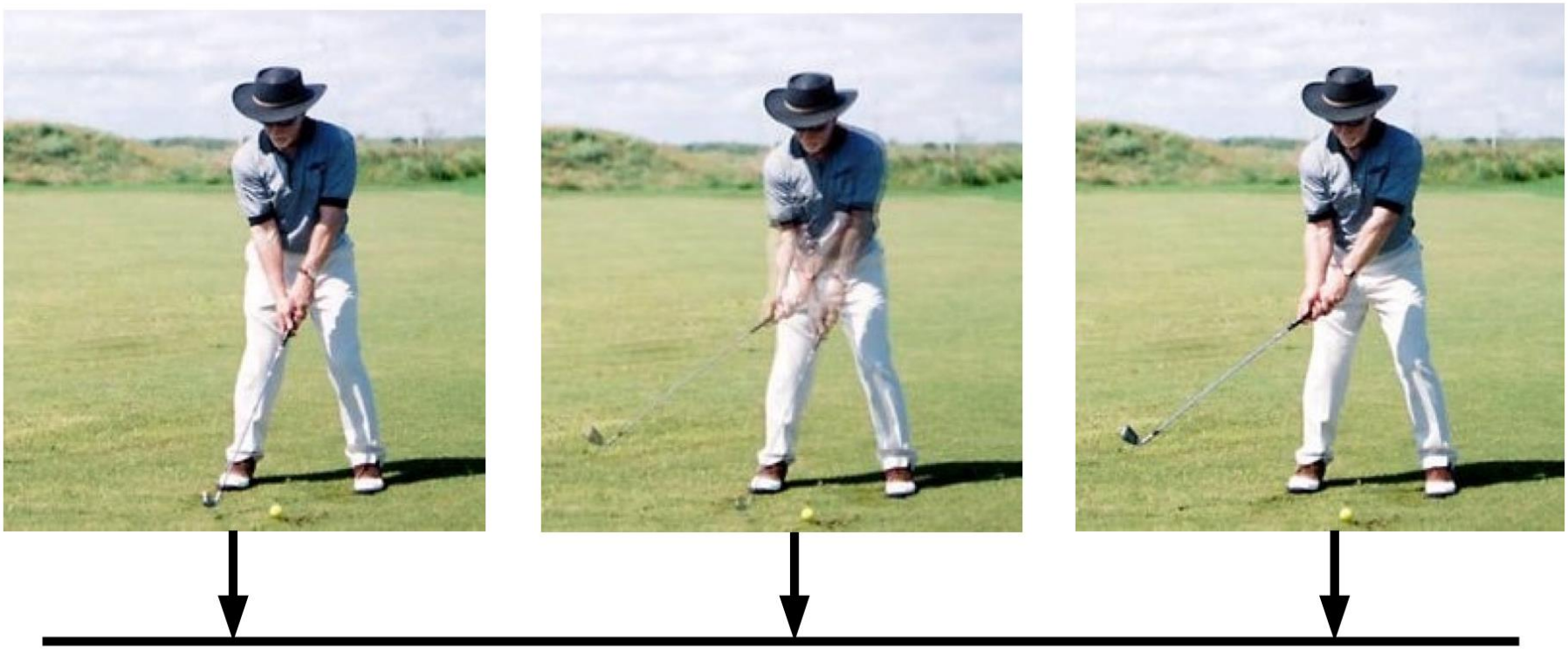
What is “a smart distance”?



Smart distance

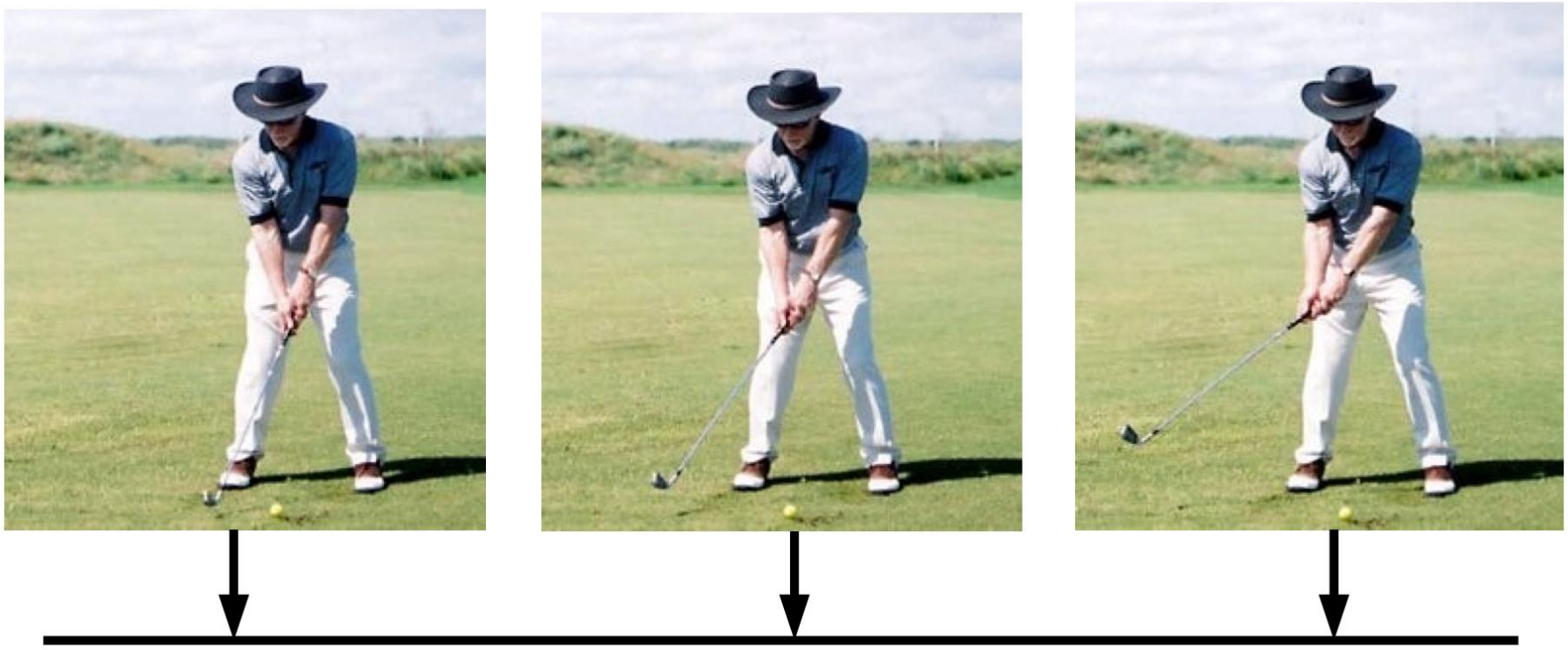


Smart distance



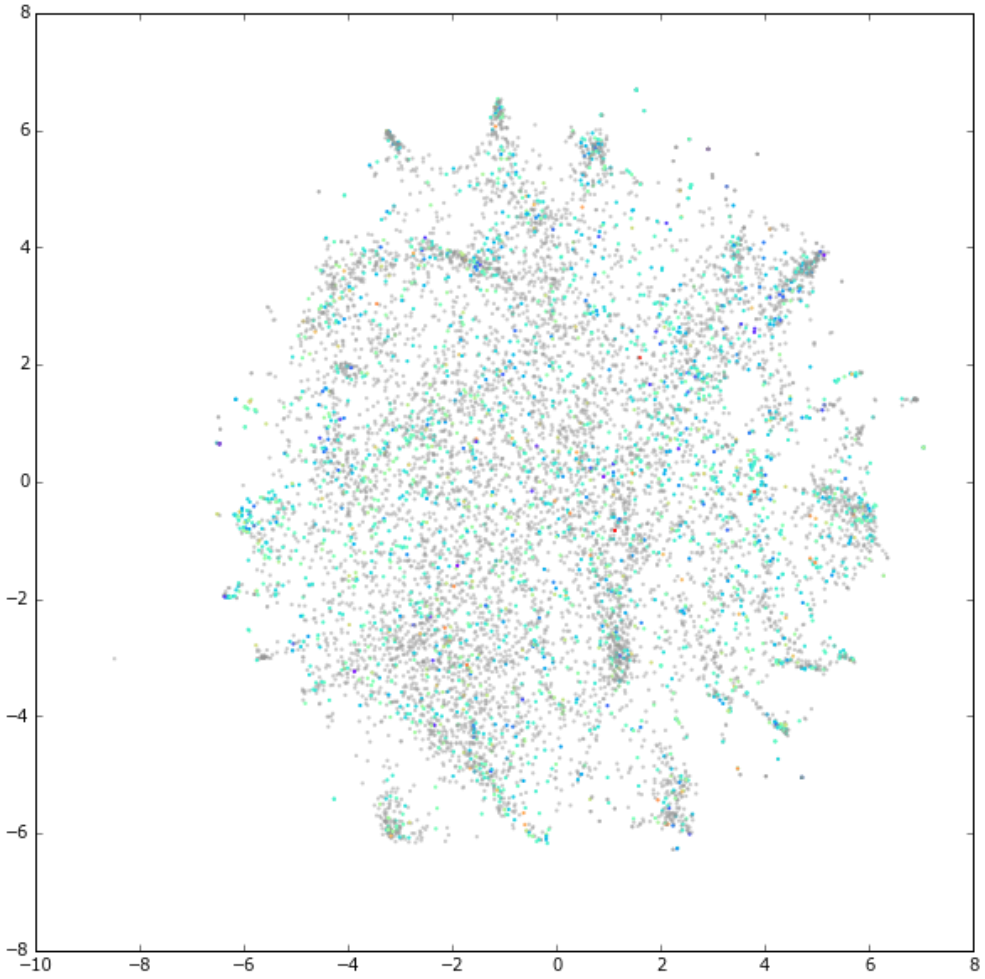
Linear interpolation

Smart distance

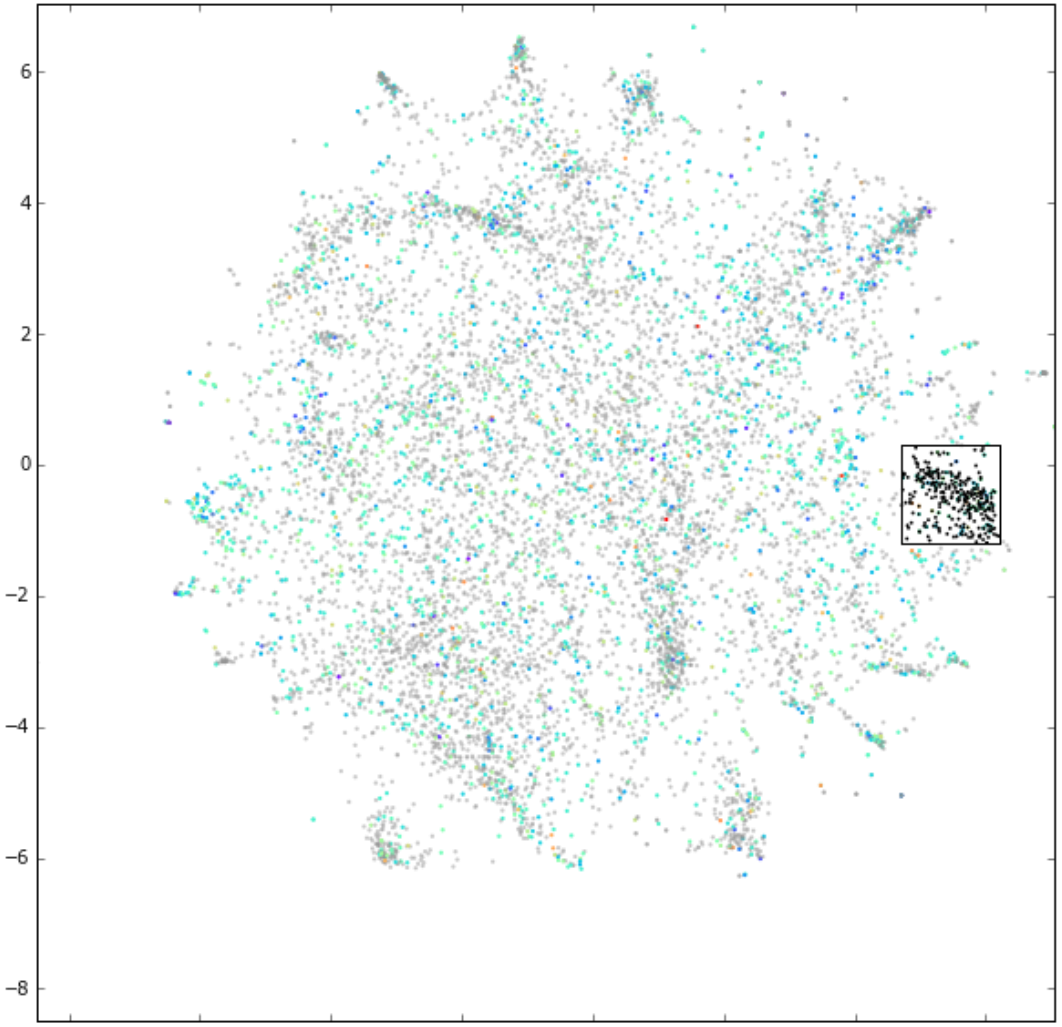


Manifold distance

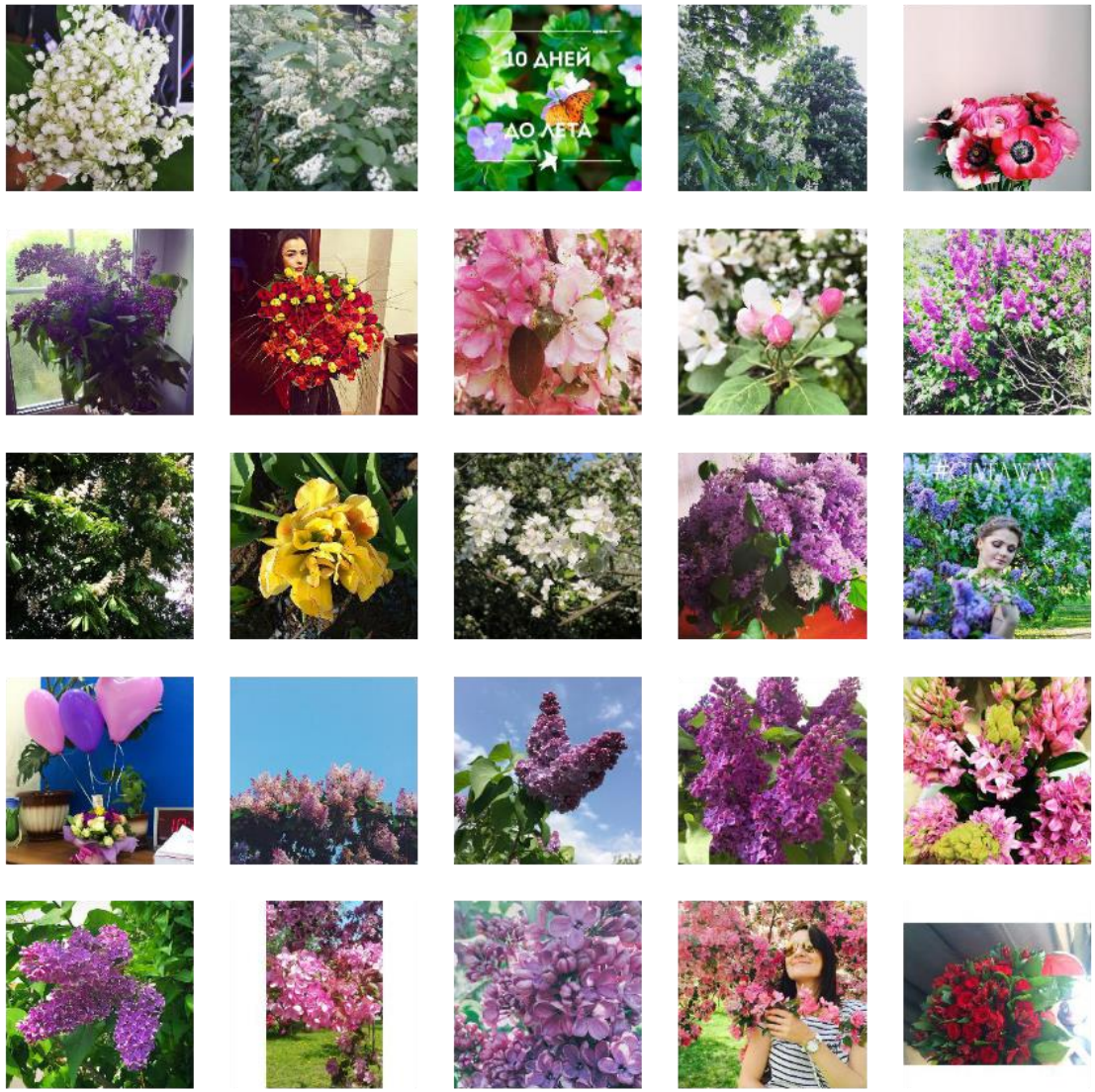
Embedding of images



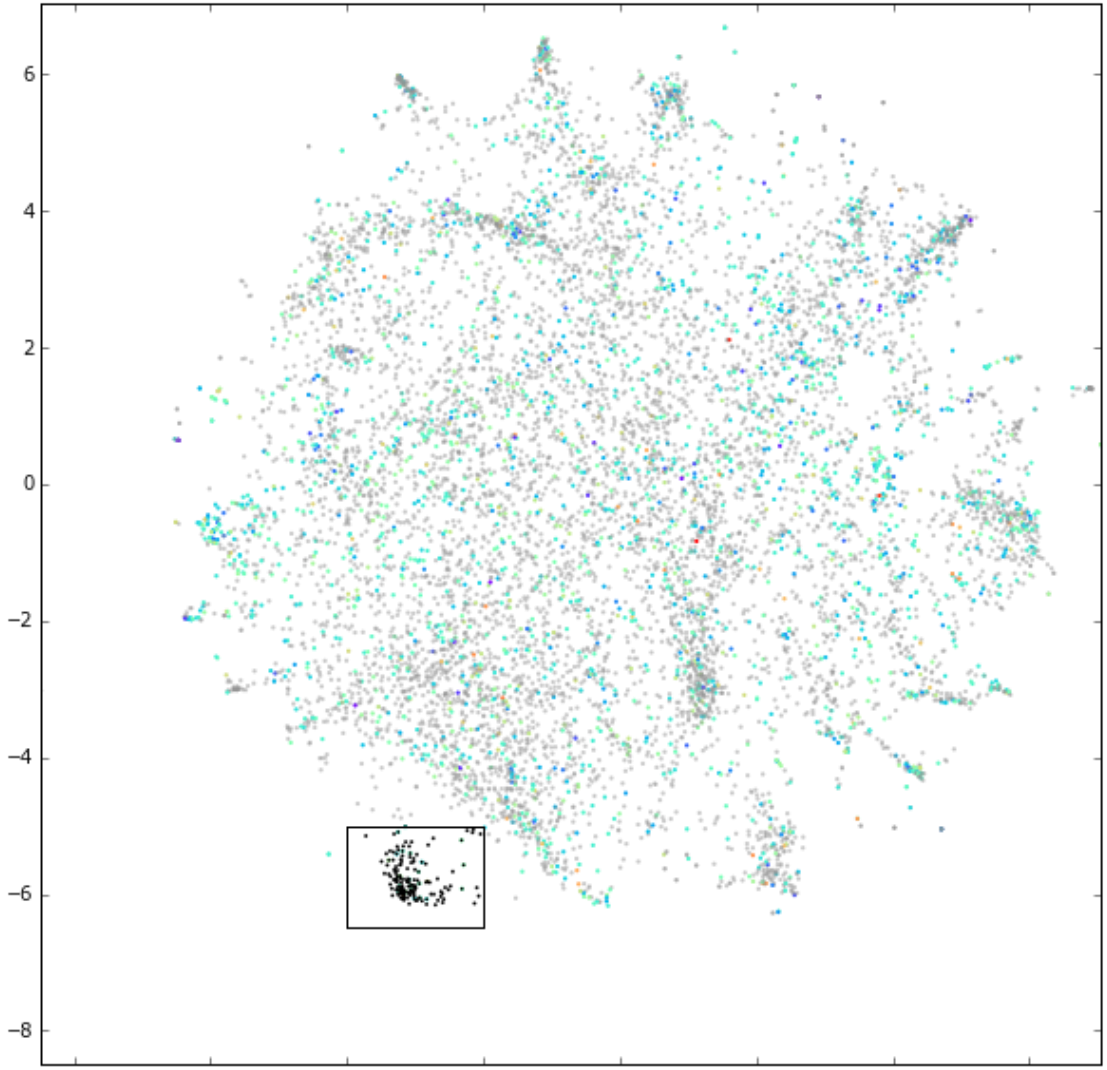
Embedding of images



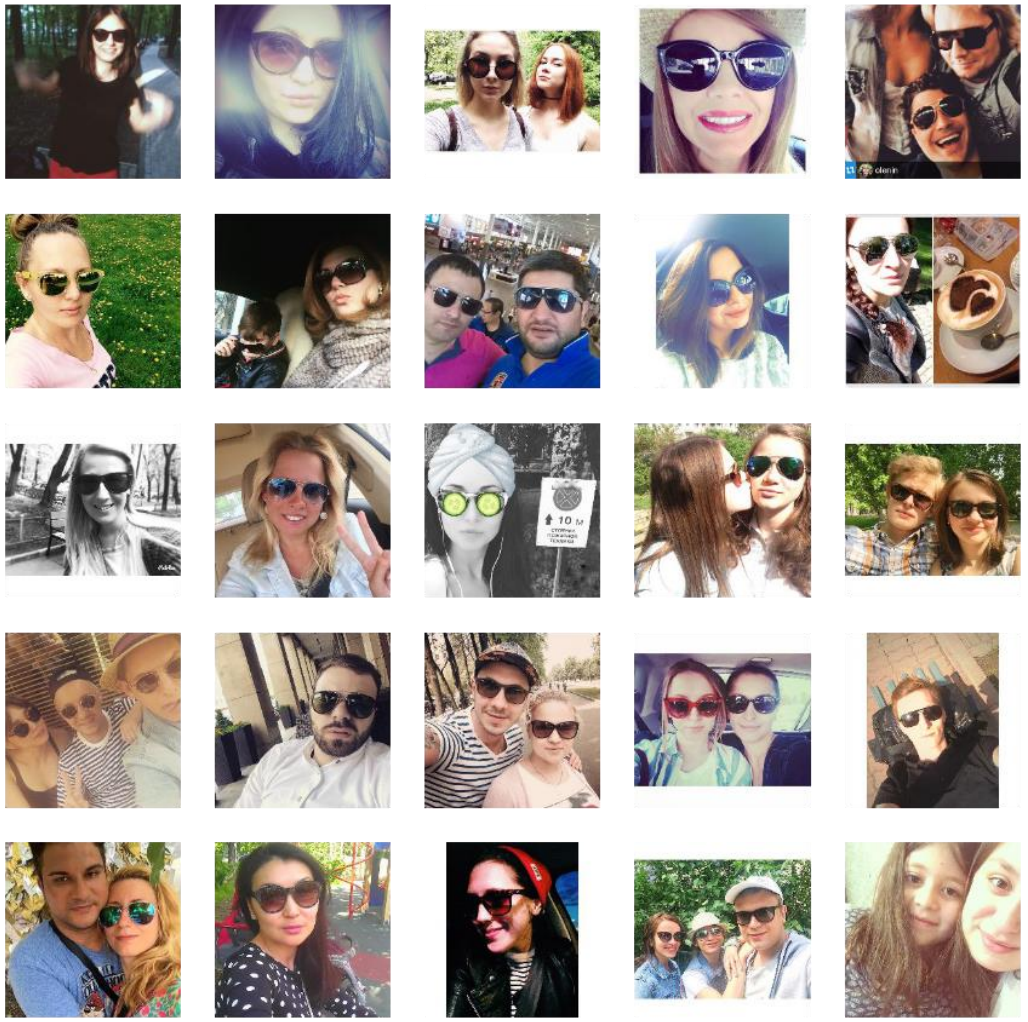
Embedding of images



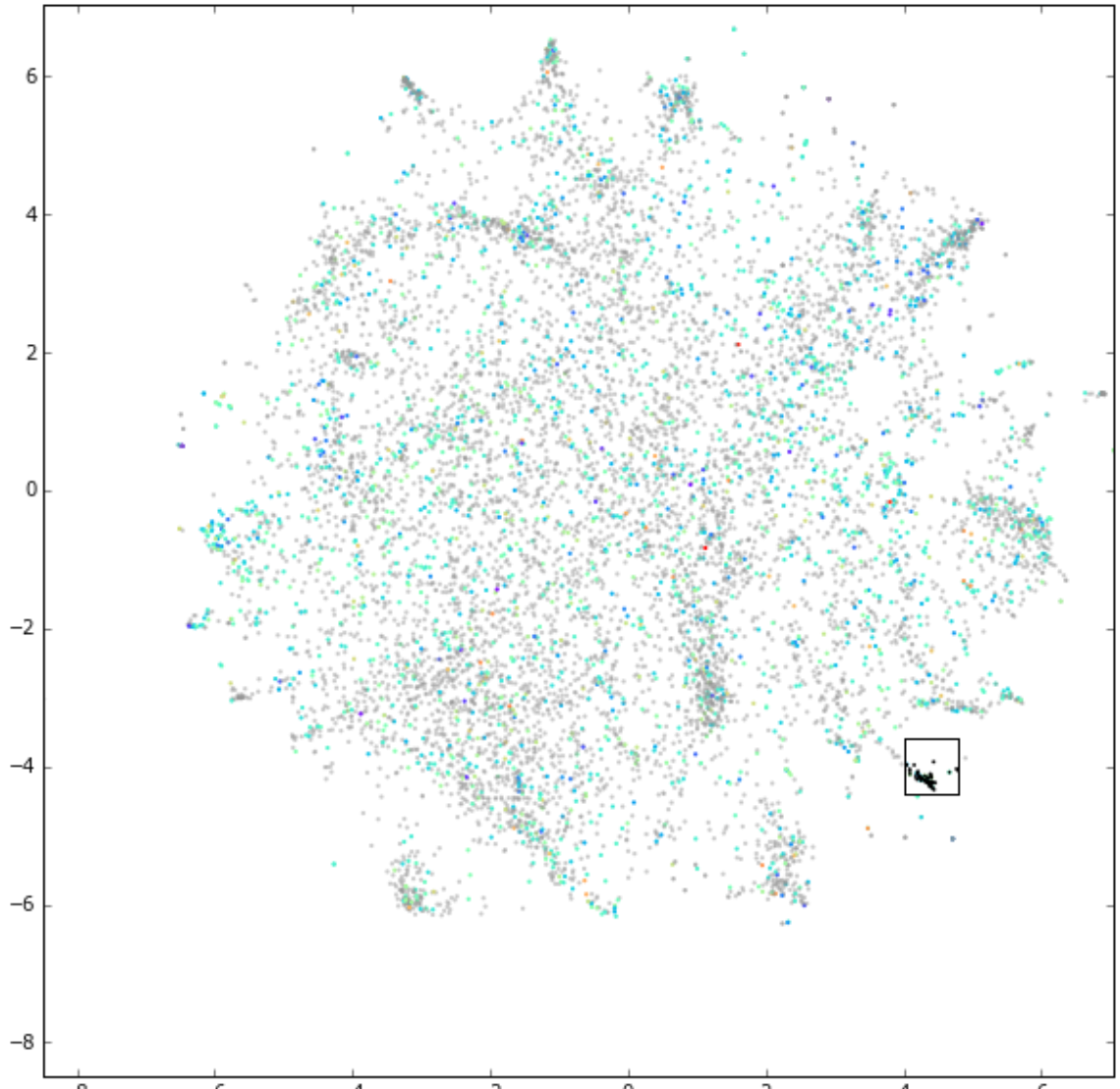
Embedding of images



Embedding of images



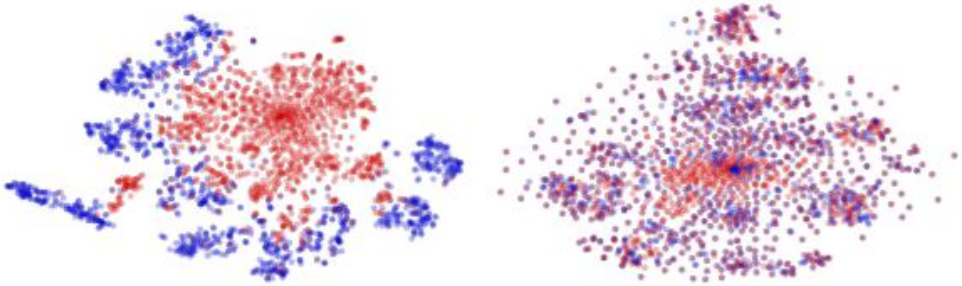
Embedding of images



Embedding of images

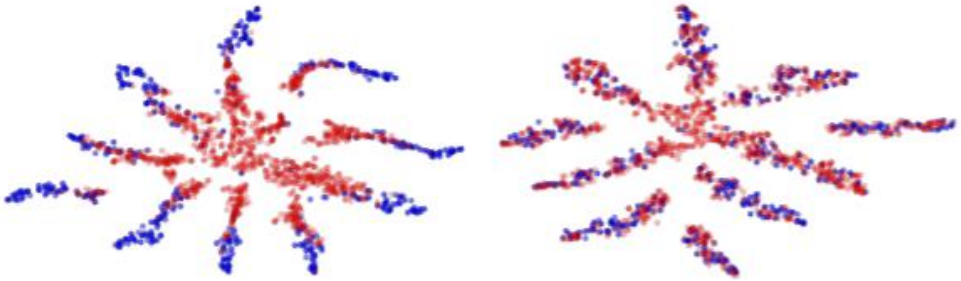
How and **how well** does our domain adaptation work?

MNIST → MNIST-M: top feature extractor layer



(a) Non-adapted (b) Adapted

SYN NUMBERS → SVHN: last hidden layer of the label predictor



(a) Non-adapted (b) Adapted

Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." *arXiv preprint arXiv:1409.7495* (2014).

MSCVS2011: V. Lempitsky, Skoltech - Writing a computer vision paper https://youtu.be/-Oq8AHIC92A

How to Write a Great Research Paper, S.P. Jones, Microsoft Research. https://youtu.be/WP-FkUaOcOM

Typical scheme for dimension reduction

1. Construct a loss function $L(y_1, \dots, y_n)$. E.g. for Multi Dimensional Scaling, MDS

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

2. Optimize loss function
3. Visually analyze results

Popular embedding methods:

- Principal Component Analysis (PCA)
- Kernel PCA, KPCA
- Locally Linear Embedding, LLE; Conformal Eigenmaps
- Laplacian Eigenmaps, Hessian Eigenmaps
- ISOmetric MAPing, ISOMAP
- Local Tangent Space Alignment, LTSA

Principal Component Analysis (PCA)

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

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

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

$$\boldsymbol{X}_n = \{X_i, i = 1, 2, \dots, n\} \subset \mathbb{R}^p.$$

В PCA: "best" = minimizes $\boldsymbol{x_0}, \{\boldsymbol{e_1}, \boldsymbol{e_2}, \dots, \boldsymbol{e_q}\} \subset \mathbb{R}^{\mathrm{p}}$

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

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

Principal Component Analysis (PCA)

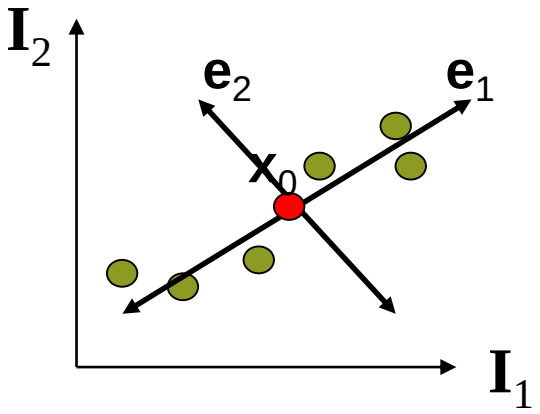
- x_{mean} – empirical mean $\{X_i, i = 1, 2, \dots, n\}$,
- $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_p\}$ – eigenvectors of $(p \times p)$-covariance matrix

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

They form an orthonormal basis in $\mathbb{R}^p$

Example: $p = 2, q = 1$

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



Solution:

- $x_0 = x_{mean}$,
- $L(q) = x_0 \oplus Span(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_q)$, the eigenvectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_q\}$ of the matrix Σ correspond to q largest eigenvalues

PCA solves the full dimension reduction problem

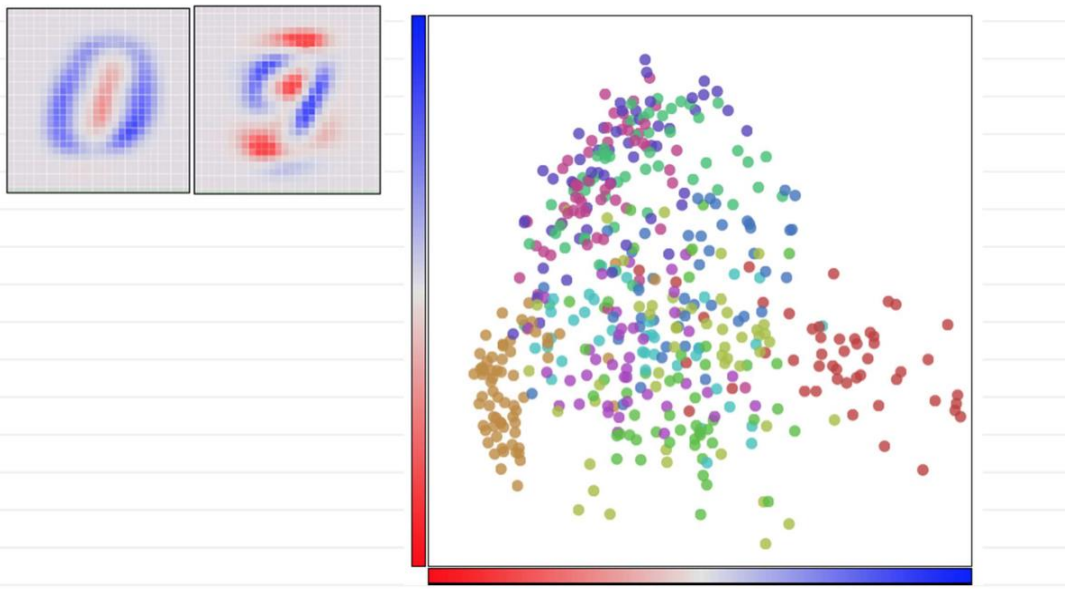
Dimension reduction:

$$X \in \mathbb{R}^p \to h(X) = \left(y_1(X_1), y_2(X_2), \ldots, y_q(X_q)\right)^T \in \mathbb{R}^q$$

Reconstruction:

$$\boldsymbol{y} = \left(y_1, y_2, \ldots, y_q\right)^T \in \mathbb{R}^q \to g(y) = x_0 + y_1 \times \boldsymbol{e}_1 + y_2 \times \boldsymbol{e}_2 + \ldots + y_q \times e_q$$

PCA for MNIST



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

Skoltech
Skolkovo Institute of Science and Technology

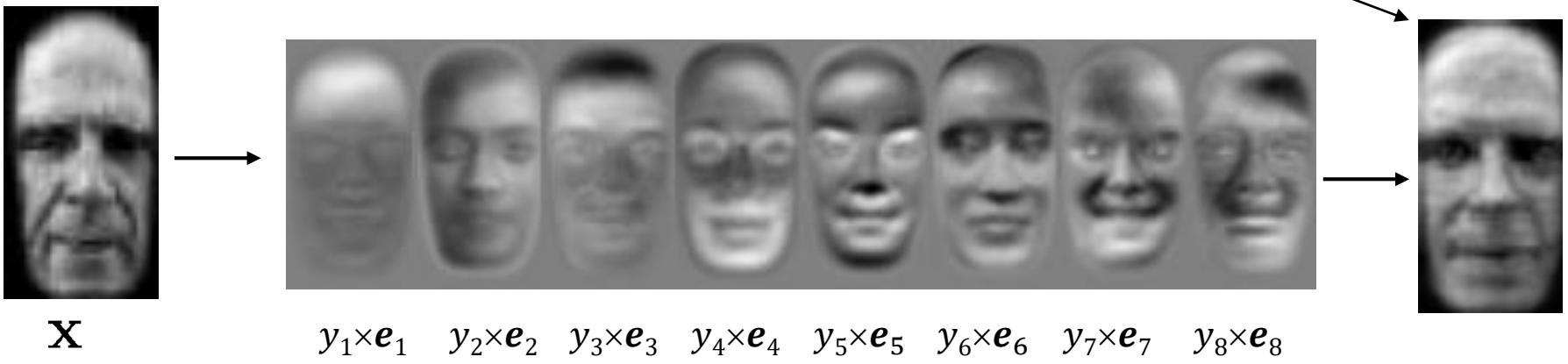
PCA for face recognition

- Eigenvectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_p \in \mathbb{R}^p$ for faces space: $p \sim 106$

$$X = \Big(\underbrace{(X - x_0, \boldsymbol{e}_1)}_{y_1}, \ldots, \underbrace{\big(X - x_0, \boldsymbol{e}_p\big)}_{y_p}\Big)^T \in \mathbb{R}^p$$

- Select $q \sim 102$ eigenvectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_q \in \mathbb{R}^p$

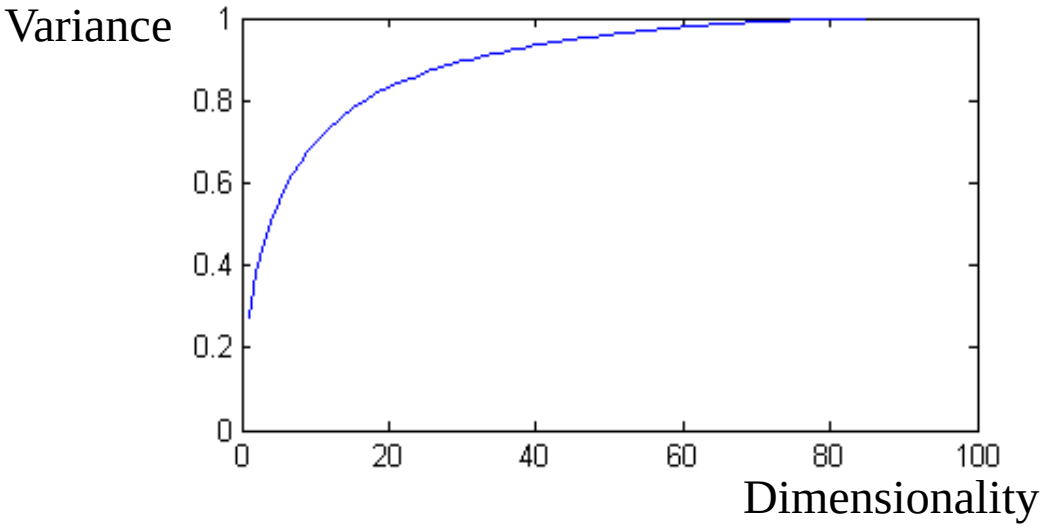
$$X^* \approx x_0 + y_1 \times \boldsymbol{e}_1 + y_2 \times \boldsymbol{e}_2 + \ldots + y_q \times \boldsymbol{e}_q$$



$\mathbf{X}$ $\quad y_1{\times}\boldsymbol{e}_1 \quad y_2{\times}\boldsymbol{e}_2 \quad y_3{\times}\boldsymbol{e}_3 \quad y_4{\times}\boldsymbol{e}_4 \quad y_5{\times}\boldsymbol{e}_5 \quad y_6{\times}\boldsymbol{e}_6 \quad y_7{\times}\boldsymbol{e}_7 \quad y_8{\times}\boldsymbol{e}_8$

PCA for face recognition

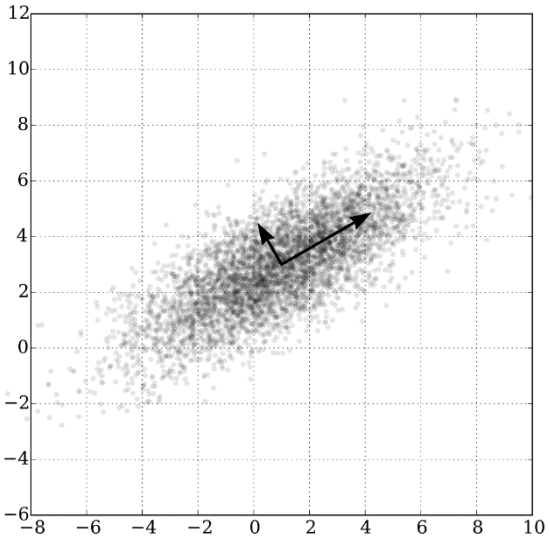
From the left to the right: reconstruction using 84, 40, 20, 3, 2, and 1 dimensions.



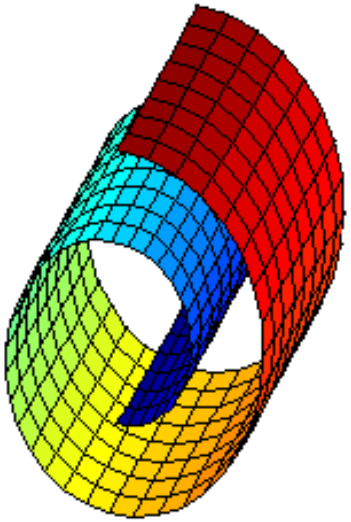
Skoltech
Skolkovo Institute of Science and Technology

PCA for face recognition

- PCA is a linear approach to dimension reduction
- It was quite popular before 2000s
- The idea is to find the best projections

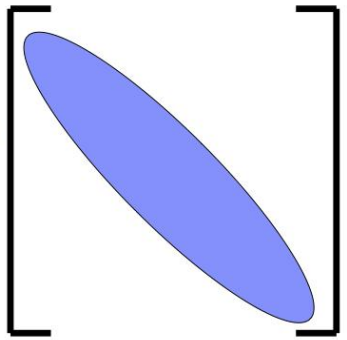


PCA fails in a nonlinear setting

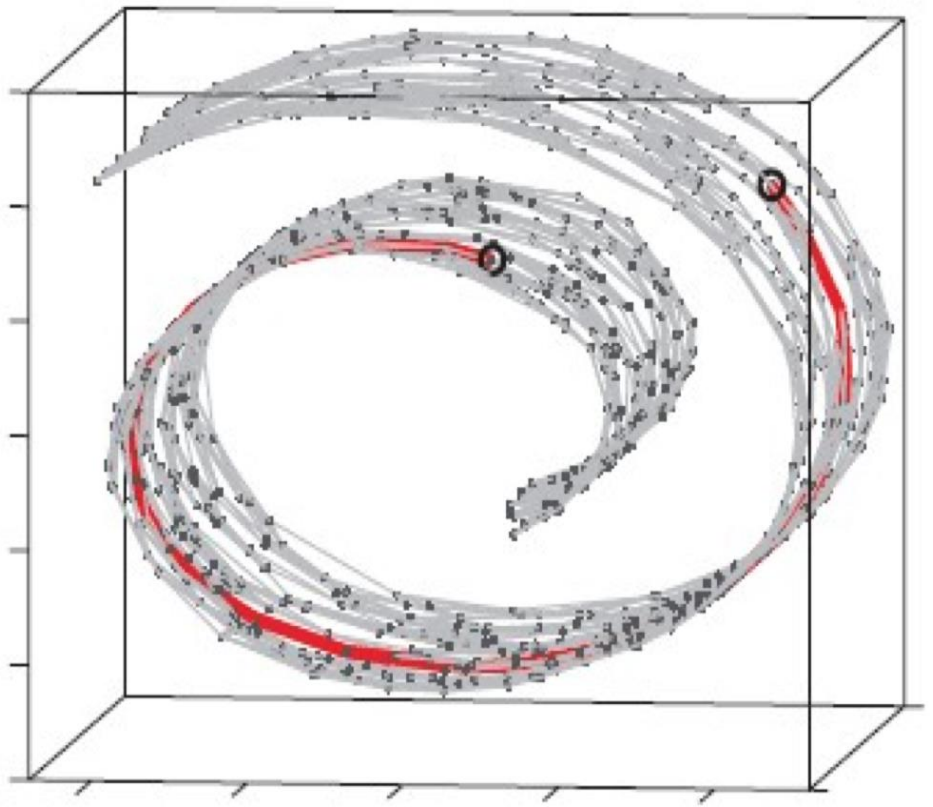


ISOMAP

1. Construct sparse k nearest neighbors graph

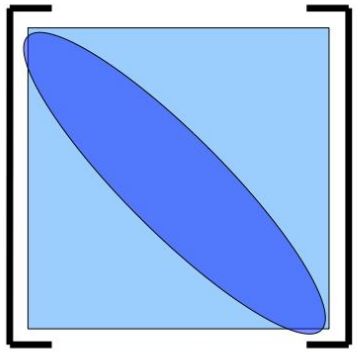
$D_g =$ 

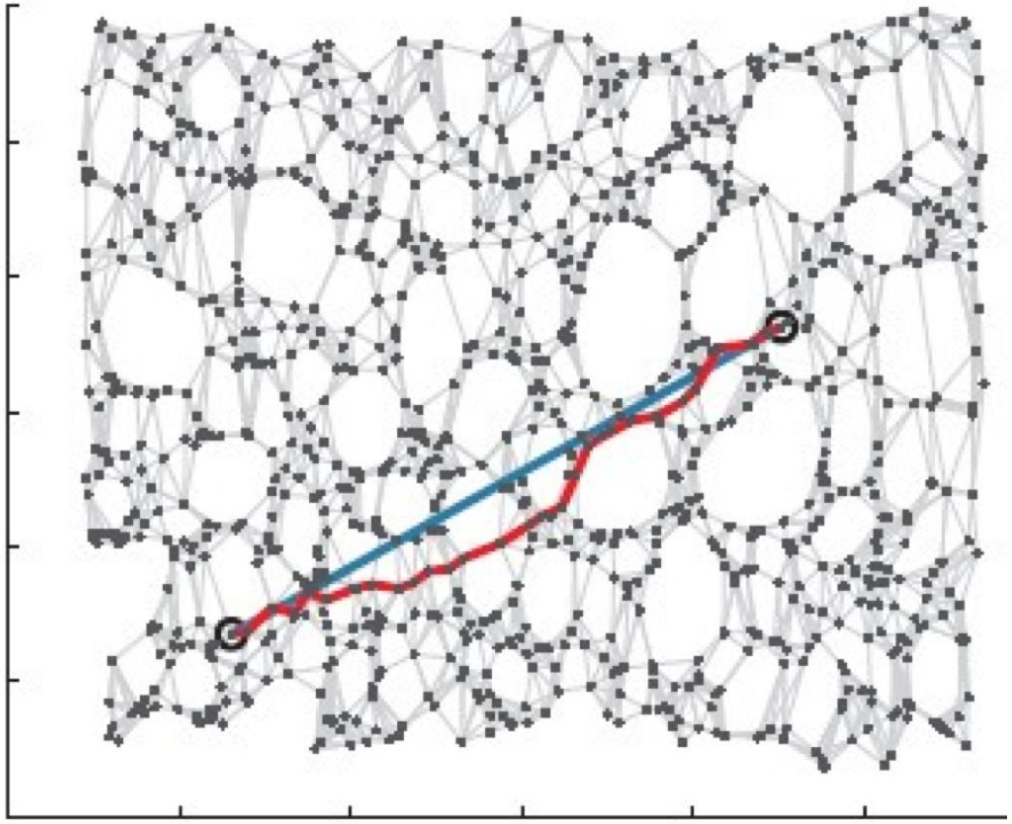
(distance matrix is sparse)



ISOMAP

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

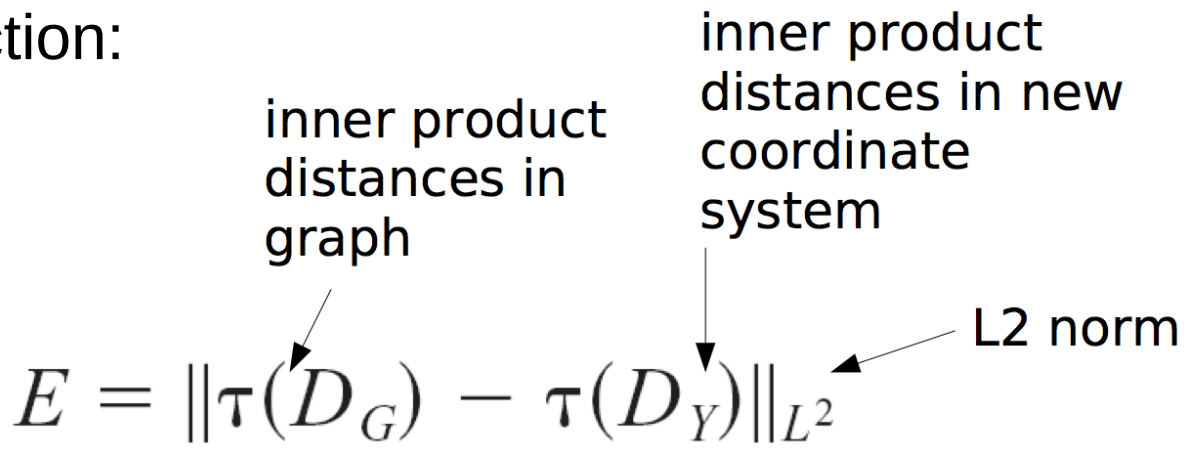
$D_g =$ 



ISOMAP

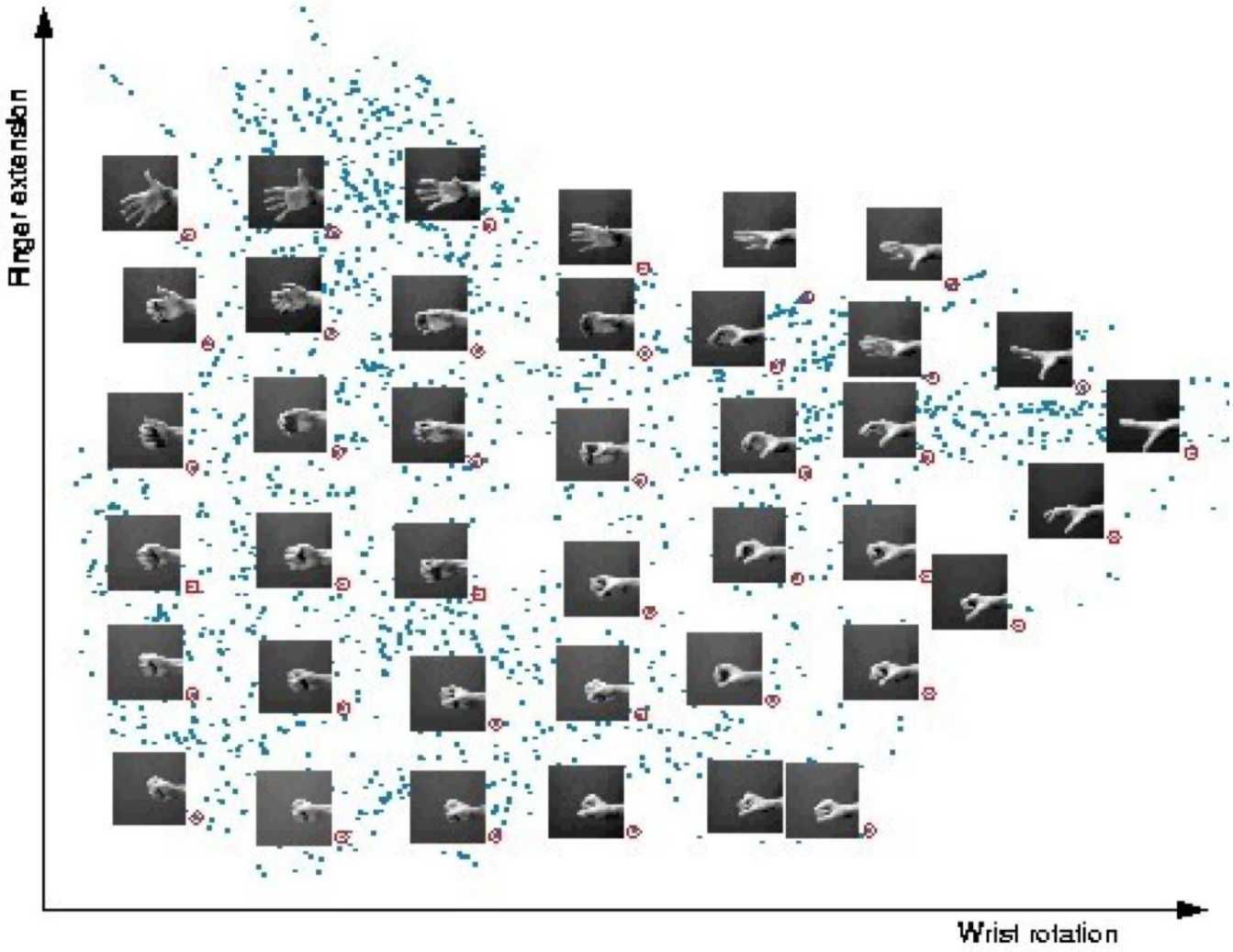
3. MDS: find low-dimensional embedding, that preserves distance matrix

Error function:

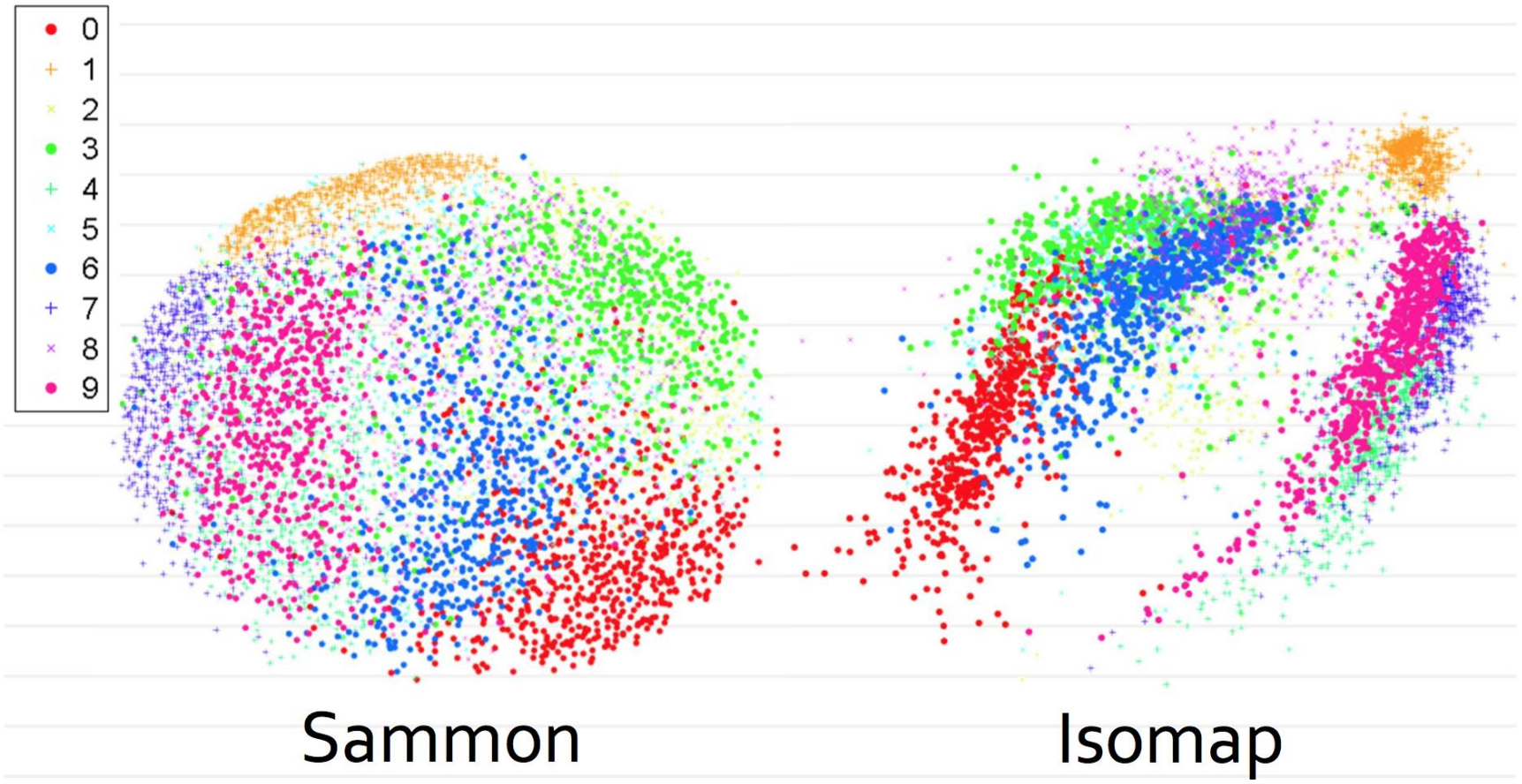


Solution – set points Y to top eigenvectors of D_g

ISOMAP results: hands embedding



MNIST: Linear approach vs. ISOMAP



Pros and cons of ISOMAP

- Preserves global structure
- Small number of parameters
- Sensitive to noise, noisy edges
- Computationally expensive: requires eigenvectors and eigenvalues of a dense matrix

Stochastic neighbours embedding (SNE)

"Stochastic" closeness in the initial space

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

Closeness in a new space

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

t-SNE

"Stochastic" closeness in the initial space

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

To avoid local optimum:

- Initialize with Gaussian noise
- Add noise of decreasing variance at each step

t-SNE: define closeness on the base of $t(1)$ distribution

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

37

t-SNE

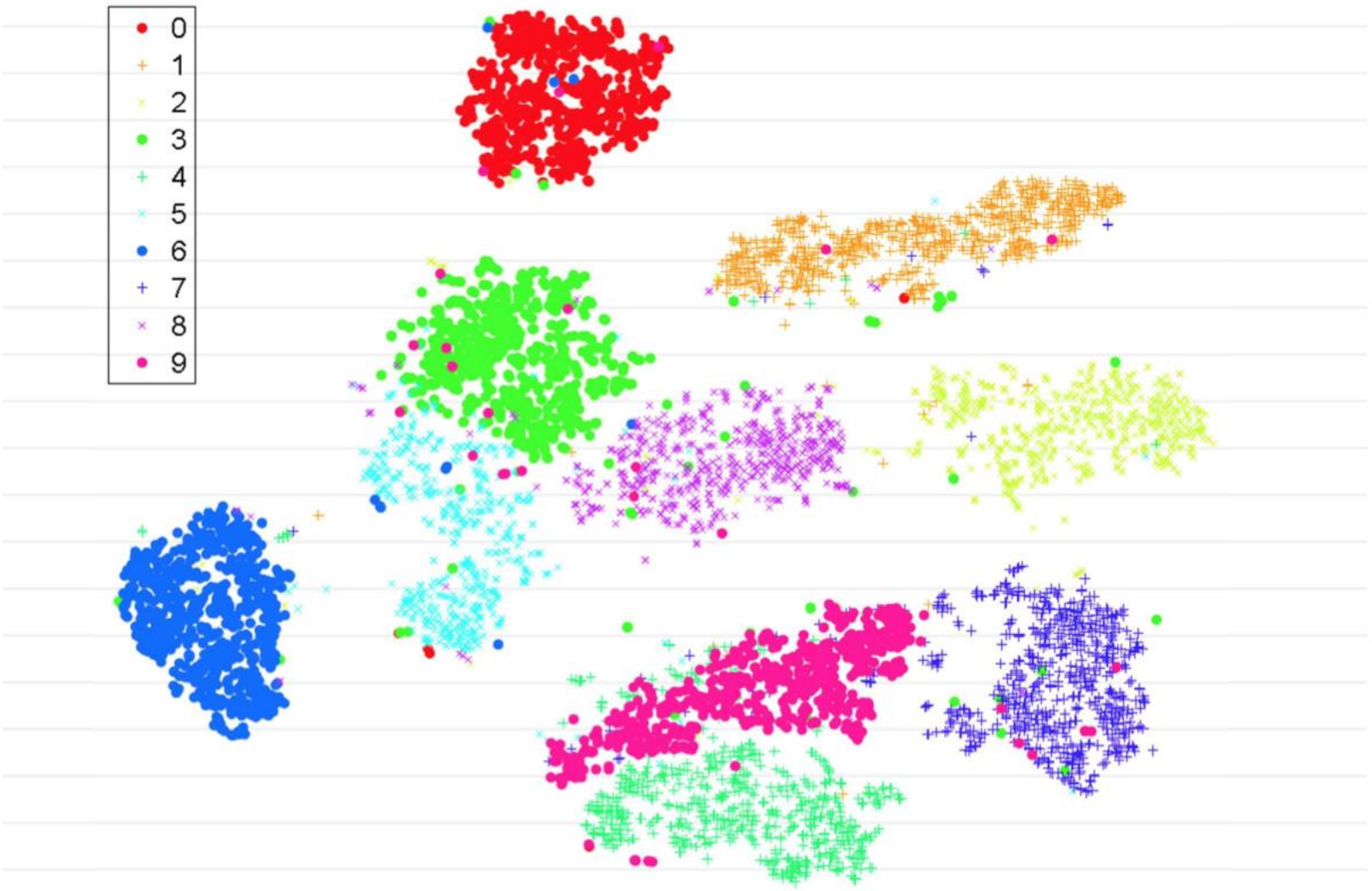
t-sne: minimize KL between P and Q

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

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

38

t-SNE for MNIST is good!



Pros and cons of tSNE

Pros:

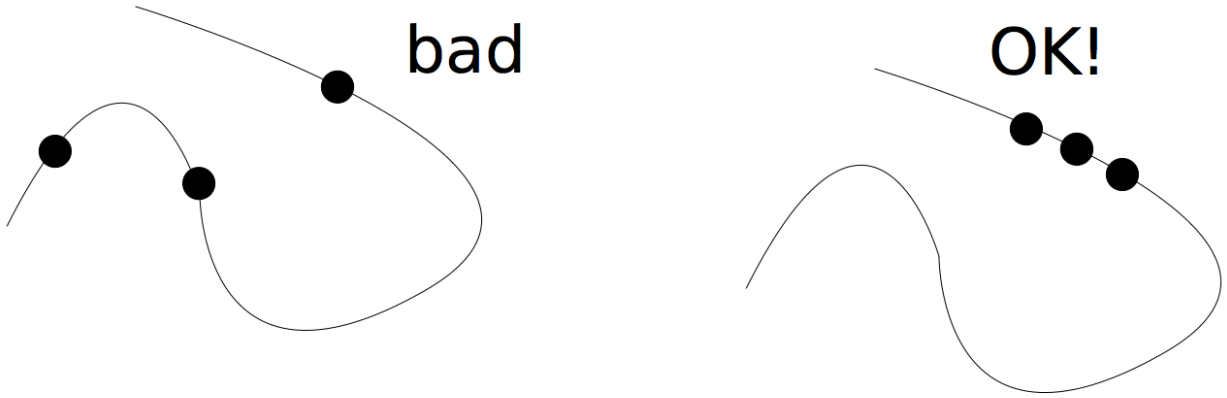
- works surprisingly well in 2D/3D visualization
- Provide nonlinear embedding

Cons:

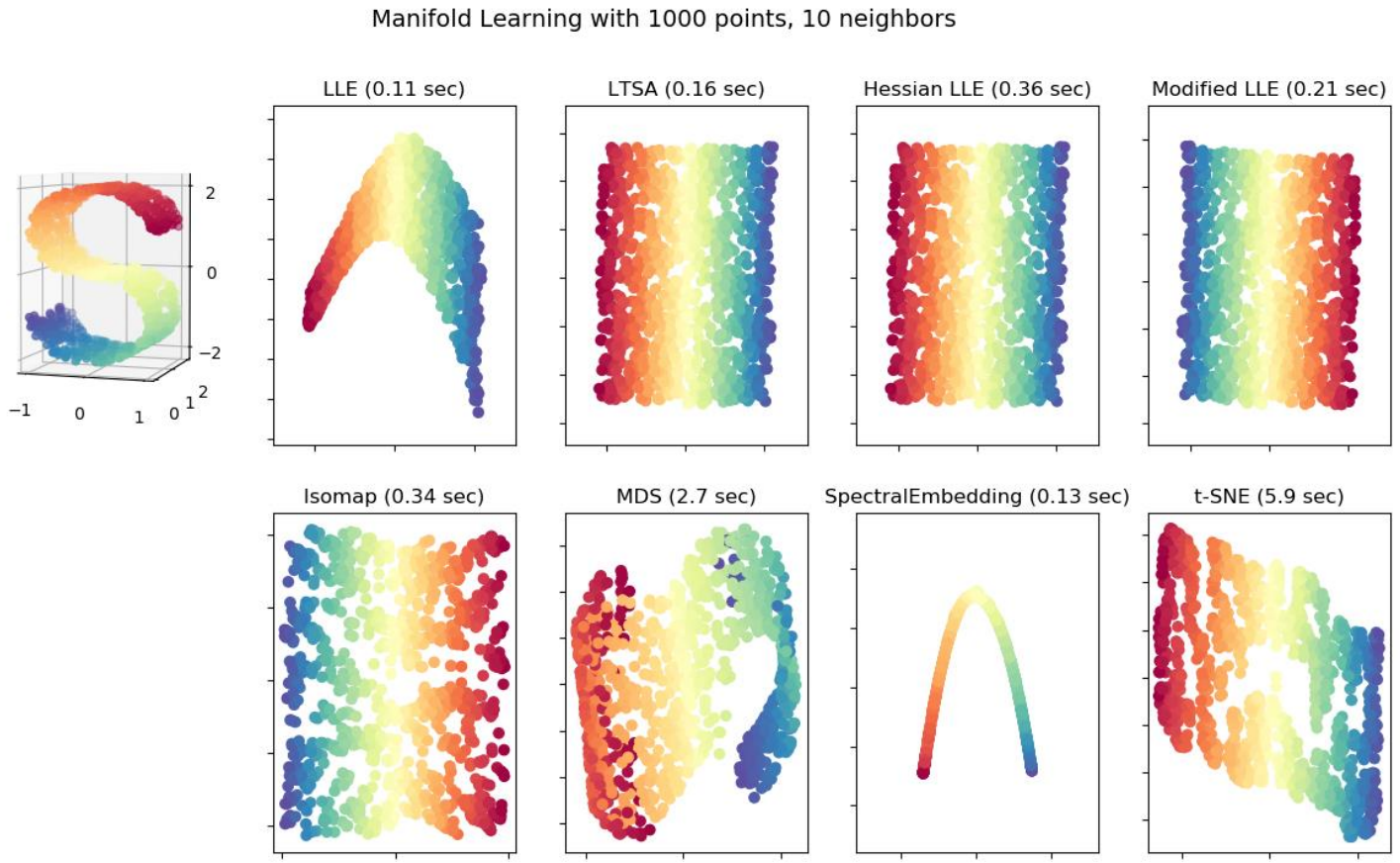
- quite slow
- Requires retraining if new points are added

No Free Lunch

For more complex manifolds you need more complex data



There are many other approaches

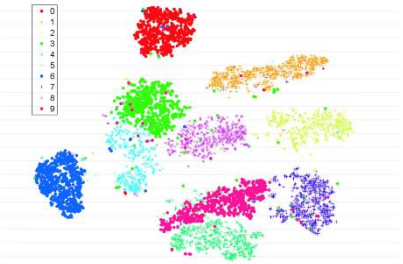


https://scikit-learn.org/stable/auto_examples/manifold/plot_compare_methods.html

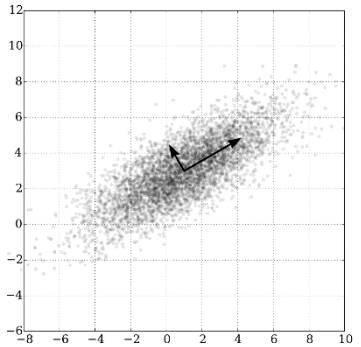
Skoltech
Skolkovo Institute of Science and Technology

Dimension reduction: take home messages

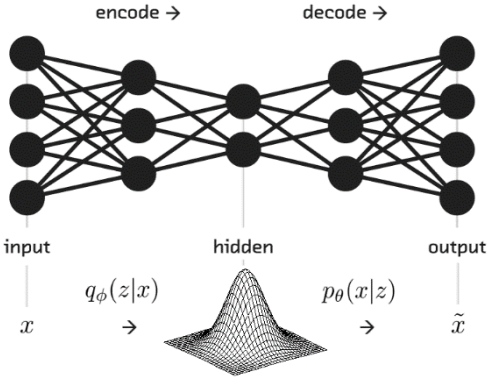
- Start with tSNE – it works surprisingly good



- PCA helps if data is linear



- There are also neural networks based embeddings



https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a

Skoltech
Skolkovo Institute of Science and Technology