Data Mining Techniques

CS 6220 - Section 3 - Fall 2016

Lecture 13

Jan-Willem van de Meent
(credit: David Lopez-Paz, David Duvenaud, Laurens van der Maaten)

Homework

- Homework 3 is out today (due 4 Nov)
- Homework 1 has been graded
(we will grade Homework 2 a little faster)
- Regrading policy
  - Step 1: E-mail TAs to resolve simple problems (e.g. code not running).
  - Step 2: E-mail instructor to request regrading.
  - We will regrade the entire problem set.
The final grade can be lower than before.

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Change of basis

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

$$\mathbf{z} = (z_1, \ldots, z_n)^\top$$

$$z_j = \mathbf{u}_j^\top \mathbf{x}$$

Inverse Change of basis

$$\mathbf{x} = \mathbf{U}\mathbf{z} = \sum_{j=1}^{n} z_j \mathbf{u}_j$$

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X}\mathbf{X}^\top$$

$$\mathbf{C}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Eigen-decomposition

$$\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top$$

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_n \end{pmatrix}$$

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X}\mathbf{X}^\top$$

$$\mathbf{C}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Eigen-decomposition

$$\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top$$

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_n \end{pmatrix}$$

Claim: Eigenvectors of a symmetric matrix are orthogonal

Review: PCA

For any real matrix A and any vectors $\mathbf{x}$ and $\mathbf{y}$, we have

$$\langle A\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, A^T\mathbf{y}\rangle.$$

Now assume that A is symmetric, and $\mathbf{x}$ and $\mathbf{y}$ are eigenvectors of A corresponding to distinct eigenvalues λ and μ. Then

$$\lambda\langle \mathbf{x}, \mathbf{y}\rangle = \langle \lambda\mathbf{x}, \mathbf{y}\rangle = \langle A\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, A^T\mathbf{y}\rangle = \langle \mathbf{x}, A\mathbf{y}\rangle = \langle \mathbf{x}, \mu\mathbf{y}\rangle = \mu\langle \mathbf{x}, \mathbf{y}\rangle.$$

Therefore, $(\lambda - \mu)\langle \mathbf{x}, \mathbf{y}\rangle = 0$. Since $\lambda - \mu \neq 0$, then $\langle \mathbf{x}, \mathbf{y}\rangle = 0$, i.e., $\mathbf{x} \perp \mathbf{y}$.

Now find an orthonormal basis for each eigenspace; since the eigenspaces are mutually orthogonal, these vectors together give an orthonormal subset of $\mathbb{R}^n$. Finally, since symmetric matrices are diagonalizable, this set will be a basis (just count dimensions). The result you want now follows.

(from stack exchange)

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X} \mathbf{X}^\top$$

$$\mathbf{C} \mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Eigen-decomposition

$$\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top$$

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_n \end{pmatrix}$$

Claim: Eigenvectors of a symmetric matrix are orthogonal

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X}\mathbf{X}^\top$$

$$\mathbf{C}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Truncated decomposition

$$\boldsymbol{C} \simeq \boldsymbol{U} \boldsymbol{\Lambda}^{(k)} \boldsymbol{U}^\top$$

$$\boldsymbol{\Lambda}^{(k)} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_k \end{pmatrix}$$

Review: PCA

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Projection / Encoding

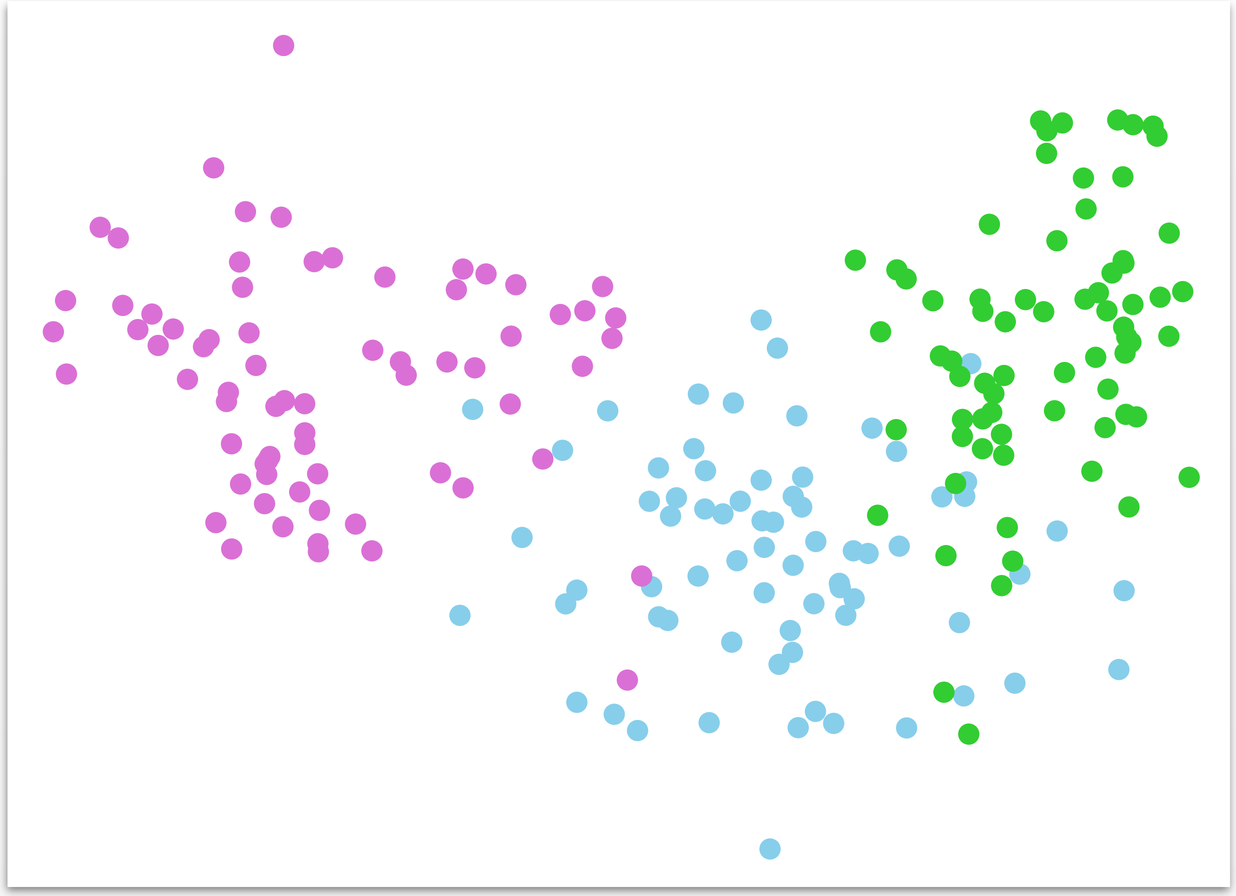
$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

Reconstruction / Decoding

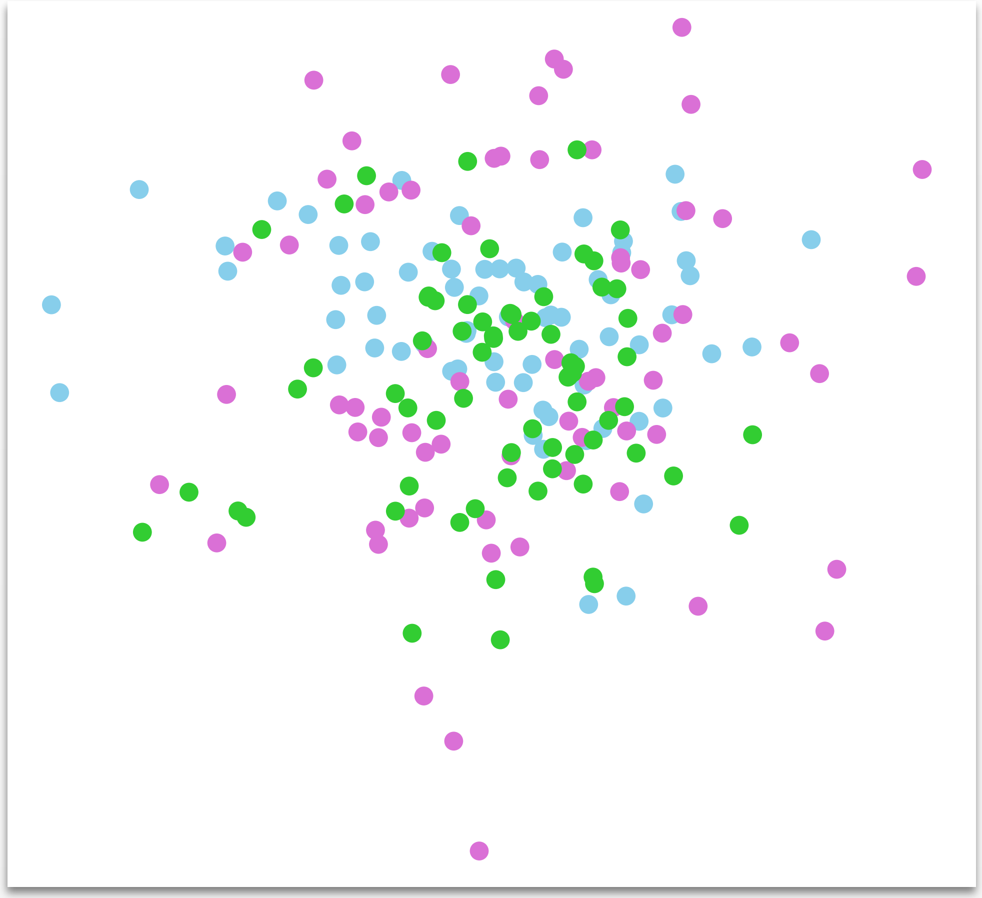
$$\tilde{\mathbf{x}} = \mathbf{U}\mathbf{z}$$

Review: PCA

Top 2 components



Bottom 2 components



Data: three varieties of wheat: Kama, Rosa, Canadian

Attributes: Area, Perimeter, Compactness, Length of Kernel, Width of Kernel, Asymmetry Coefficient, Length of Groove

PCA: Complexity

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Using eigen-value decomposition

- Computation of covariance **C**: $O(n\, d^2)$
- Eigen-value decomposition: $O(d^3)$
- Total complexity: $O(n\, d^2 + d^3)$

PCA: Complexity

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

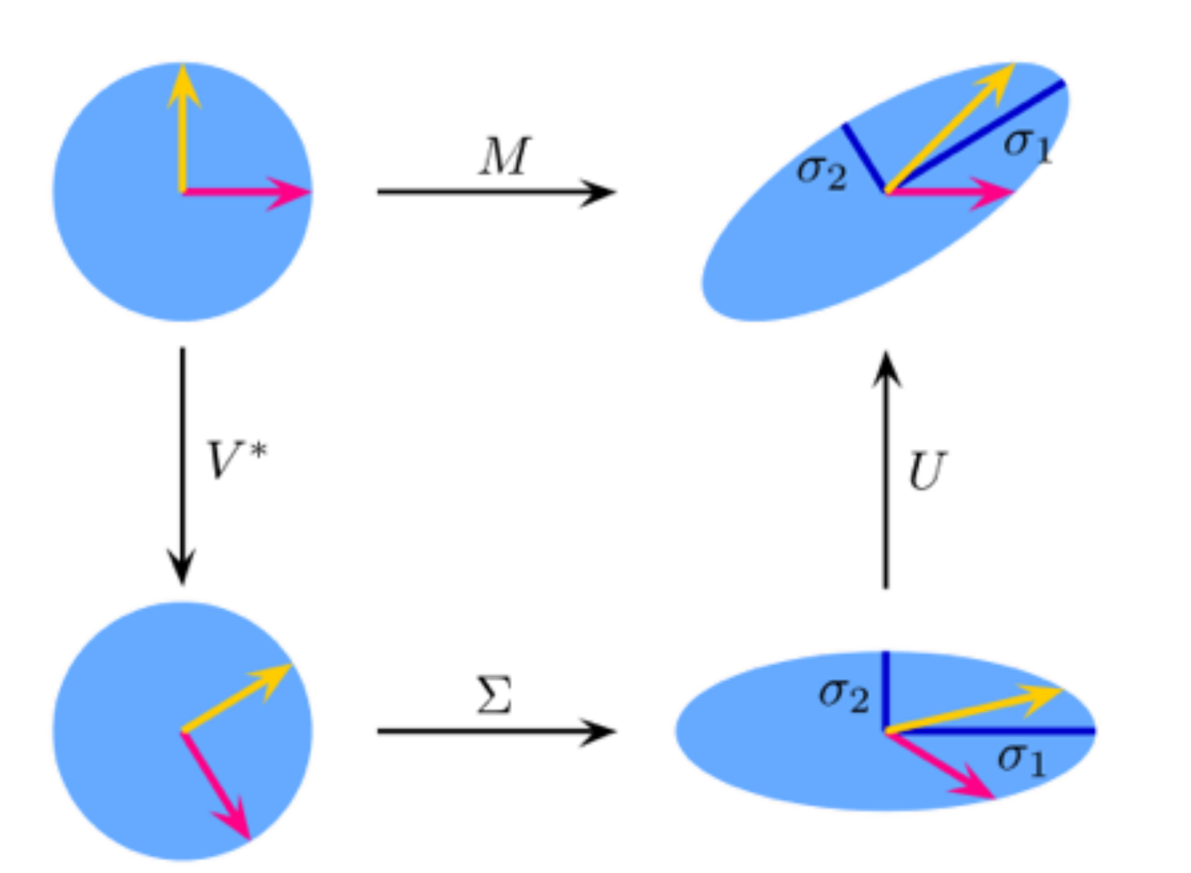
Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Using singular-value decomposition

- Full decomposition: O(min$\{nd^2, n^2d\}$)
- Rank-k decomposition: O($k\ d\ n$ log(n)) (with power method)

Singular Value Decomposition



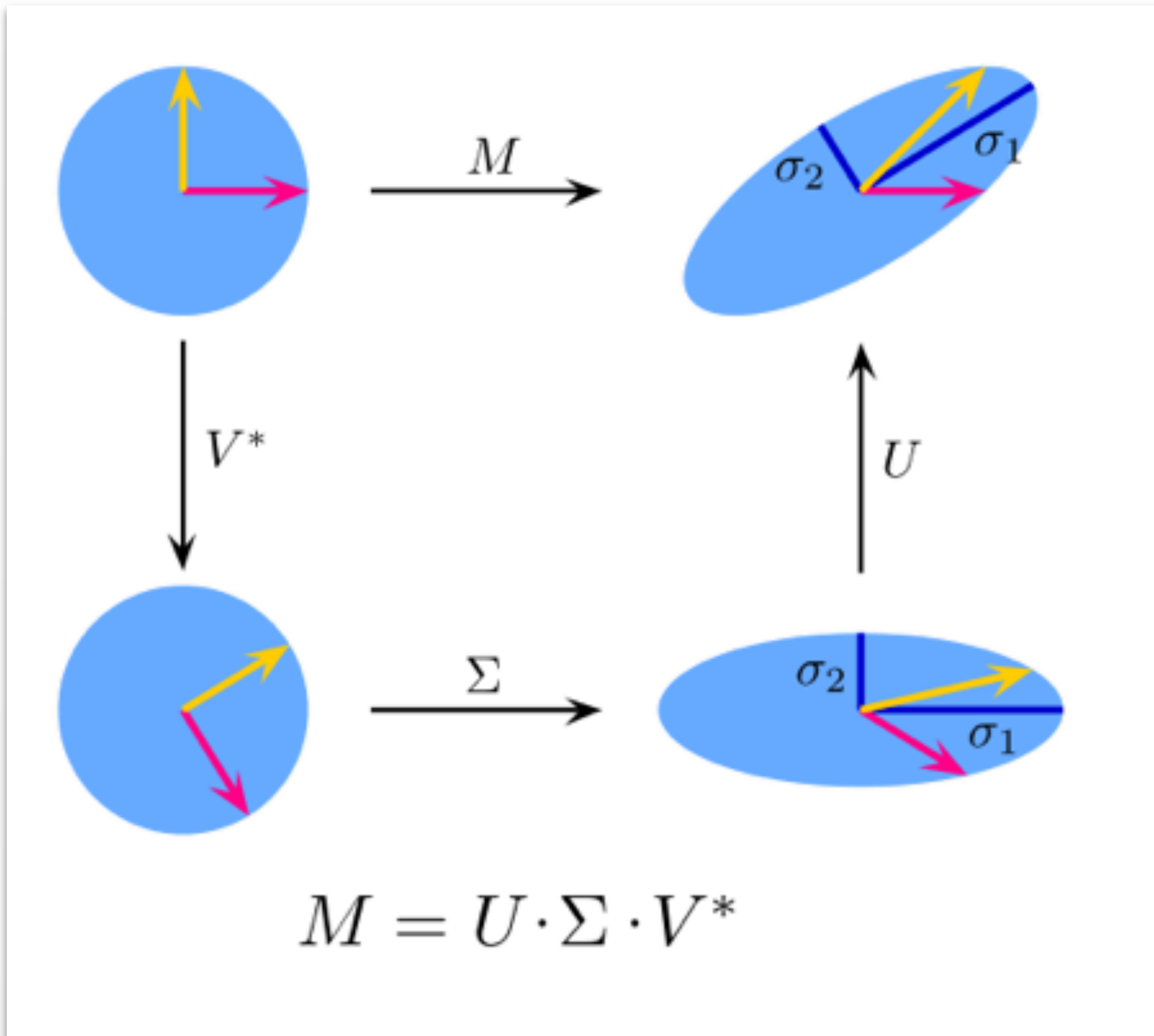
$$M = U \cdot \Sigma \cdot V^*$$

Idea: Decompose a **d x d** matrix *M* into

1. Change of basis *V* (unitary matrix)
2. A scaling *Σ* (diagonal matrix)
3. Change of basis *U* (unitary matrix)

$$\begin{array}{ccccc} M & = & \mathbf{U} & \boldsymbol{\Sigma} & \mathbf{V}^* \\ \begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} & = & \begin{bmatrix} 0 & -i \\ 1 & 0 \end{bmatrix} & \frac{1}{\sqrt{2}}\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} & \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & -1 \\ i & -i \end{bmatrix} \end{array}$$

Singular Value Decomposition

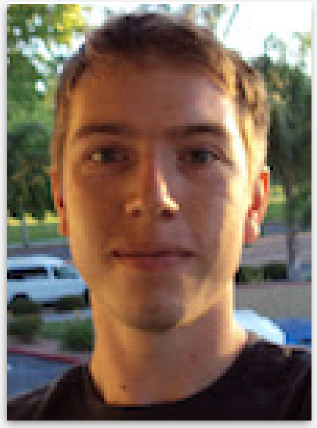


Idea: Decompose the **d x n** matrix *X* into

1. A n x n basis *V* (unitary matrix)
2. A d x n matrix ***Σ*** (diagonal projection)
3. A d x d basis *U* (unitary matrix)

$$\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}_{n\times n}^{\top}$$

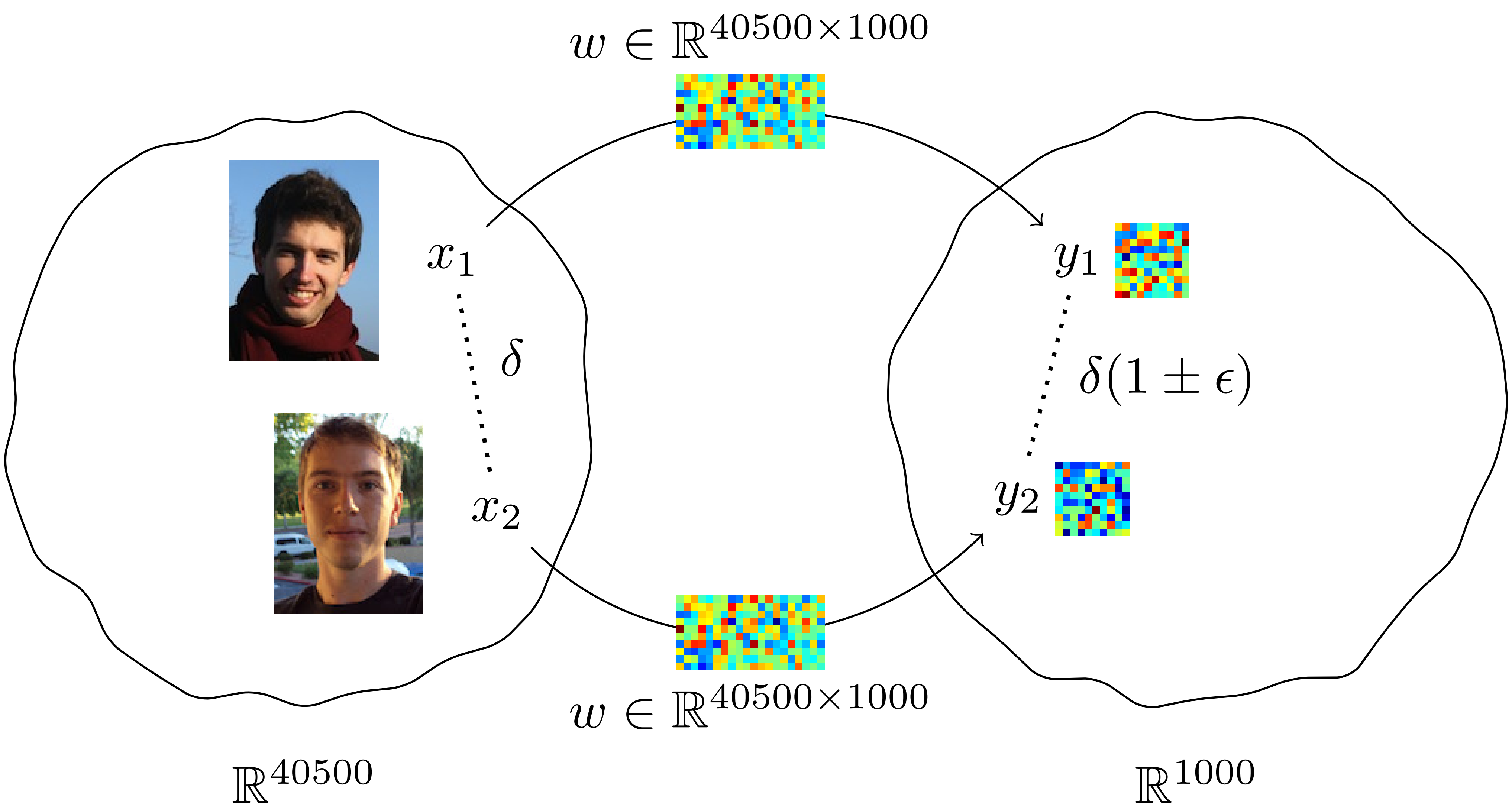
Random Projections



Borrowing from:
David Lopez-Paz
& David Duvenaud

Random Projections

Fast, efficient and & distance-preserving **dimensionality reduction**!



$$(1-\epsilon)\|x_1 - x_2\|^2 \leq \|y_1 - y_2\|^2 \leq (1+\epsilon)\|x_1 - x_2\|^2$$

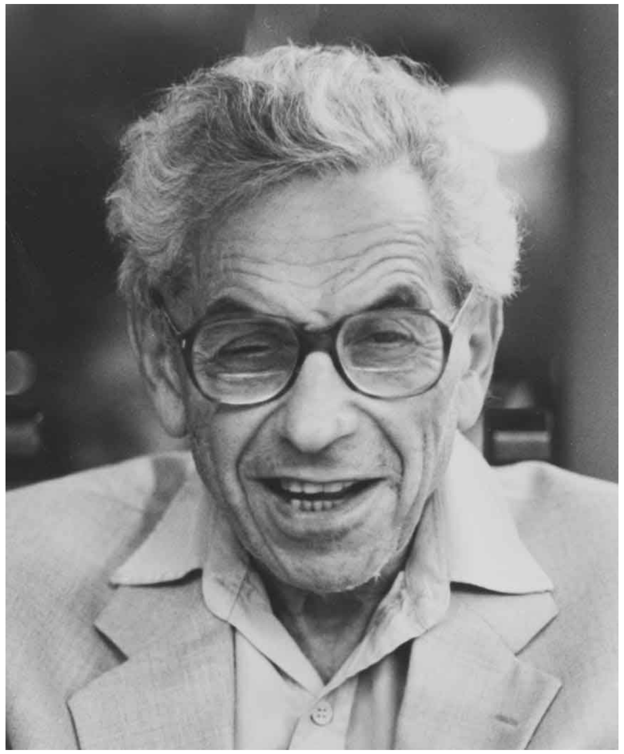
This result is formalized in the *Johnson-Lindenstrauss Lemma*

Johnson-Lindenstrauss Lemma

For any $0 < \epsilon < 1/2$ and any integer $m > 4$, let $k = \frac{20 \log m}{\epsilon^2}$. Then, for any set V of m points in $\mathbb{R}^N$ $\exists$ $f : \mathbb{R}^N \to \mathbb{R}^k$ s.t. $\forall$ $\boldsymbol{u}, \boldsymbol{v} \in V$:

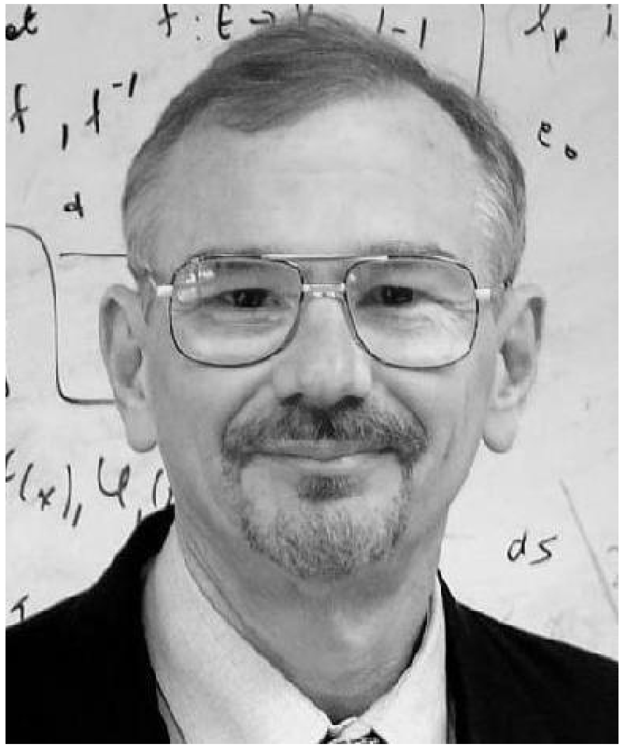
$$(1-\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2 \le \|f(\boldsymbol{u}) - f(\boldsymbol{v})\|^2 \le (1+\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2.$$

The proof is a great example of Erdös' *probabilistic method* (1947).



Paul Erdös
1913-1996

Joram Lindenstrauss
1936-2012



William B. Johnson
1944-

Johnson-Lindenstrauss Lemma

For any $0 < \epsilon < 1/2$ and any integer $m > 4$, let $k = \frac{20 \log m}{\epsilon^2}$. Then, for any set V of m points in $\mathbb{R}^N$ $\exists\ f : \mathbb{R}^N \to \mathbb{R}^k$ s.t. $\forall\ \boldsymbol{u}, \boldsymbol{v} \in V$:

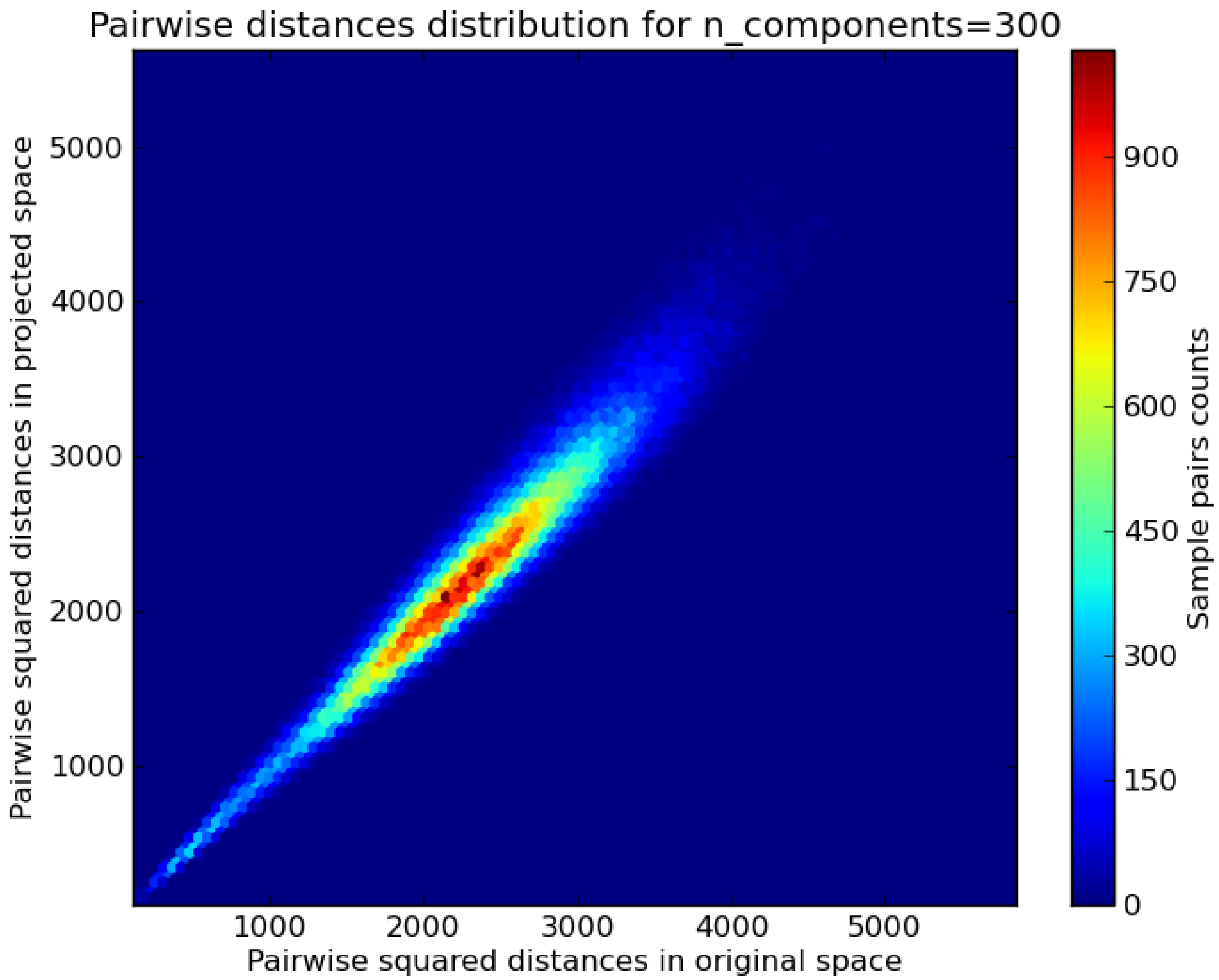
$$(1-\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2 \leq \|f(\boldsymbol{u}) - f(\boldsymbol{v})\|^2 \leq (1+\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2.$$

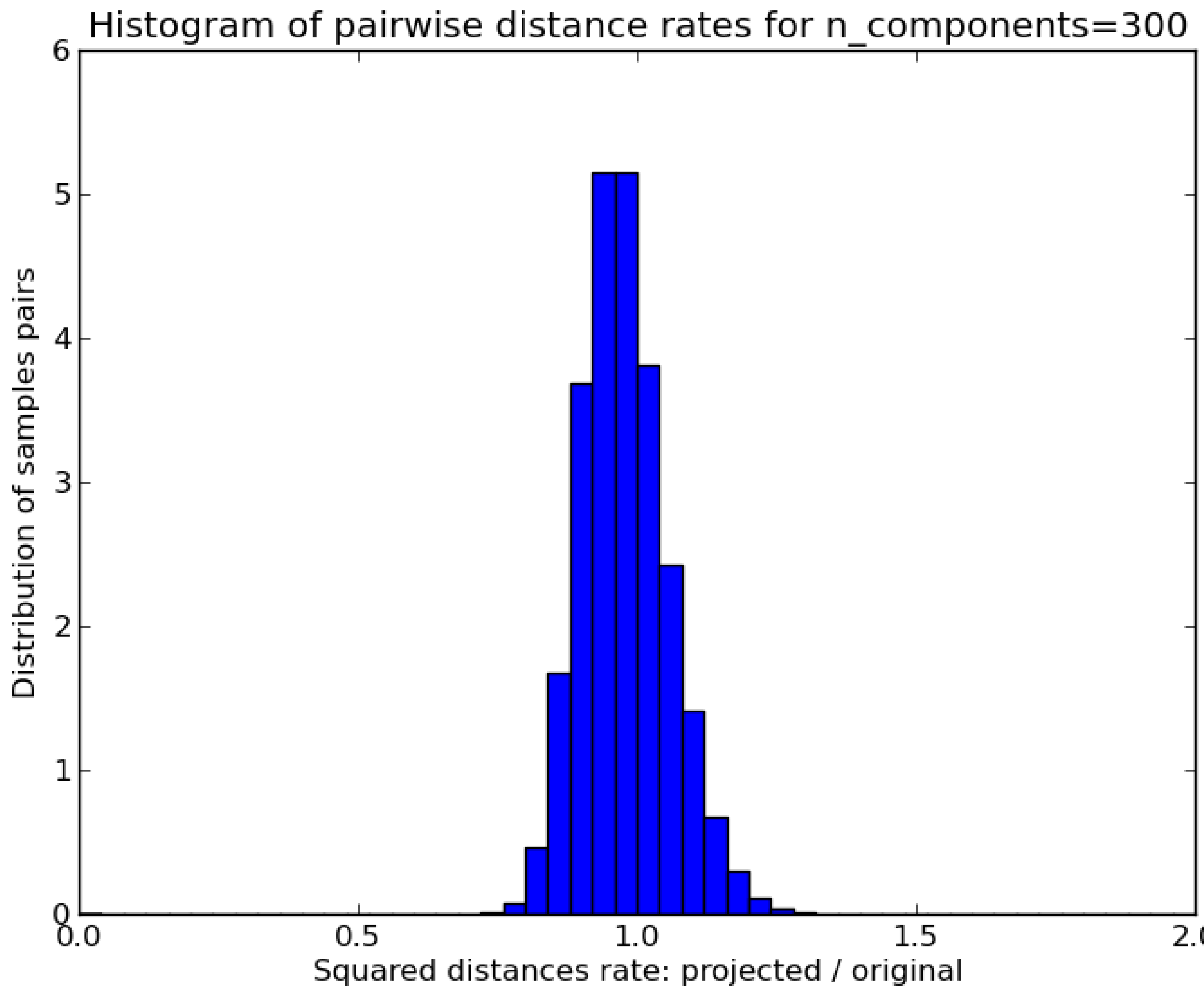
Holds when *f* is linear function with random coefficients

$$f = \frac{1}{\sqrt{k}}\boldsymbol{A},\ \boldsymbol{A} \in \mathbb{R}^{k \times N},\ k < N \text{ and } A_{ij} \sim \mathcal{N}(0,1)$$

Example: 20-newsgroups data

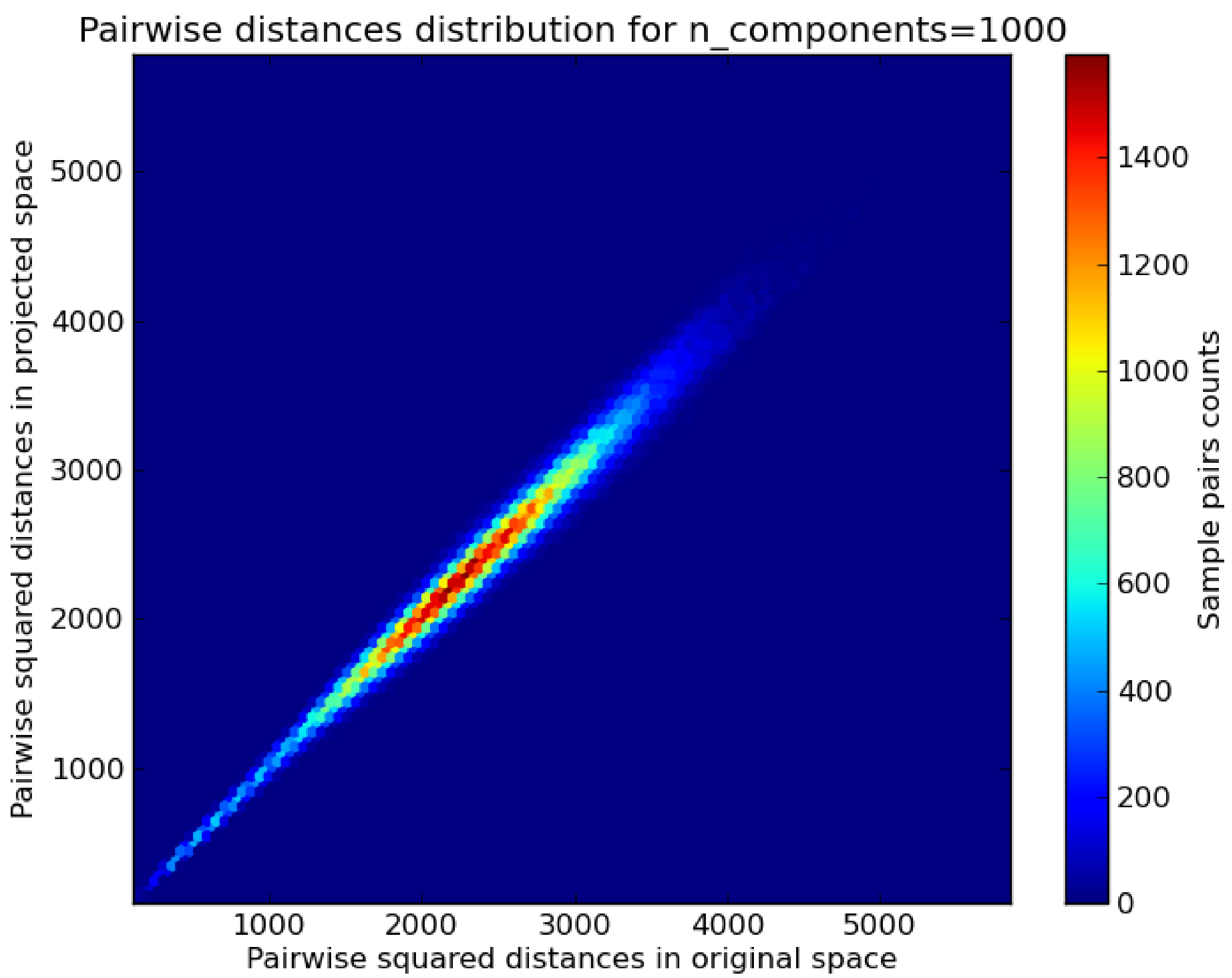
Data: 20-newsgroups, from 100.000 features to 300 (0.3%)

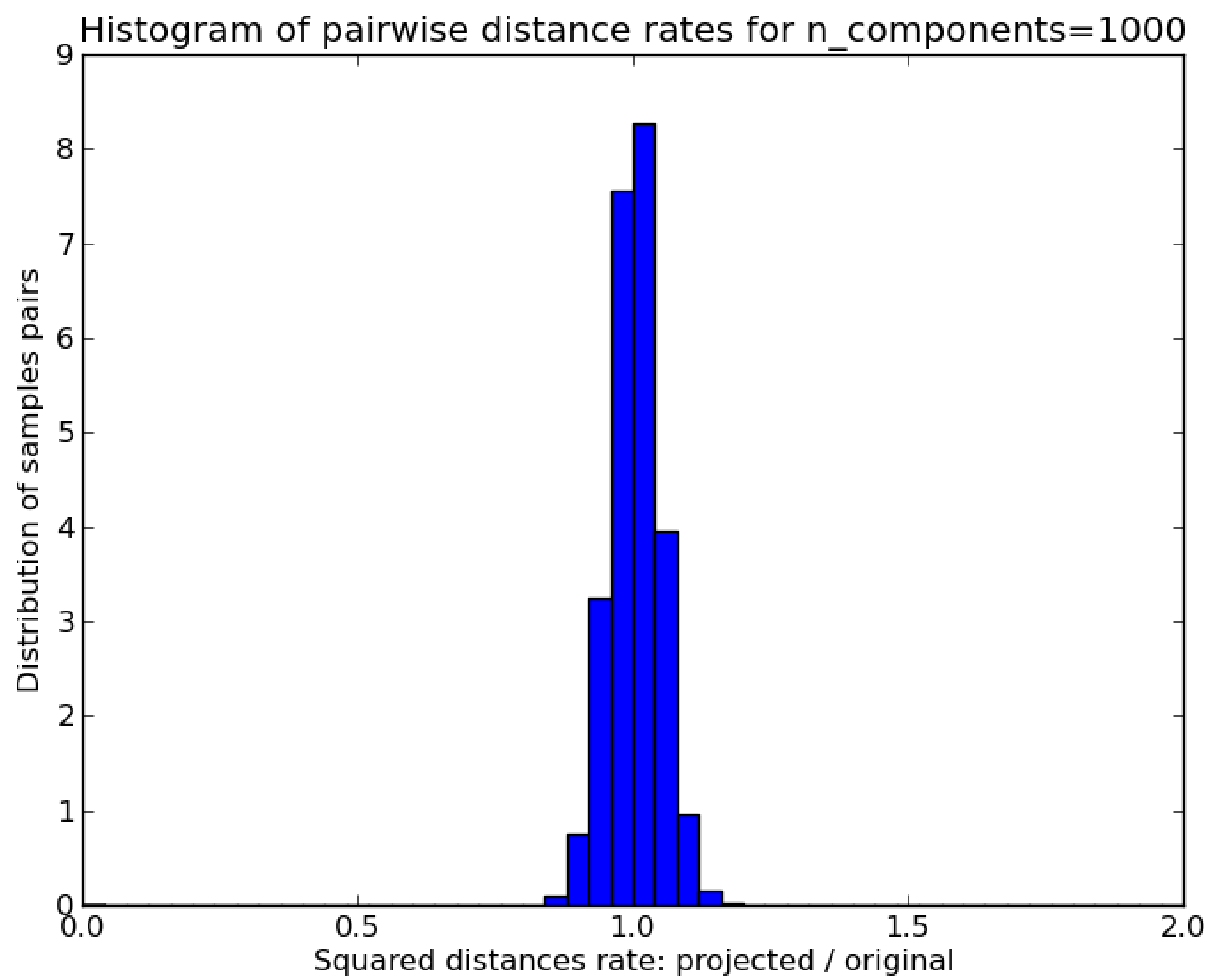




Example: 20-newsgroups data

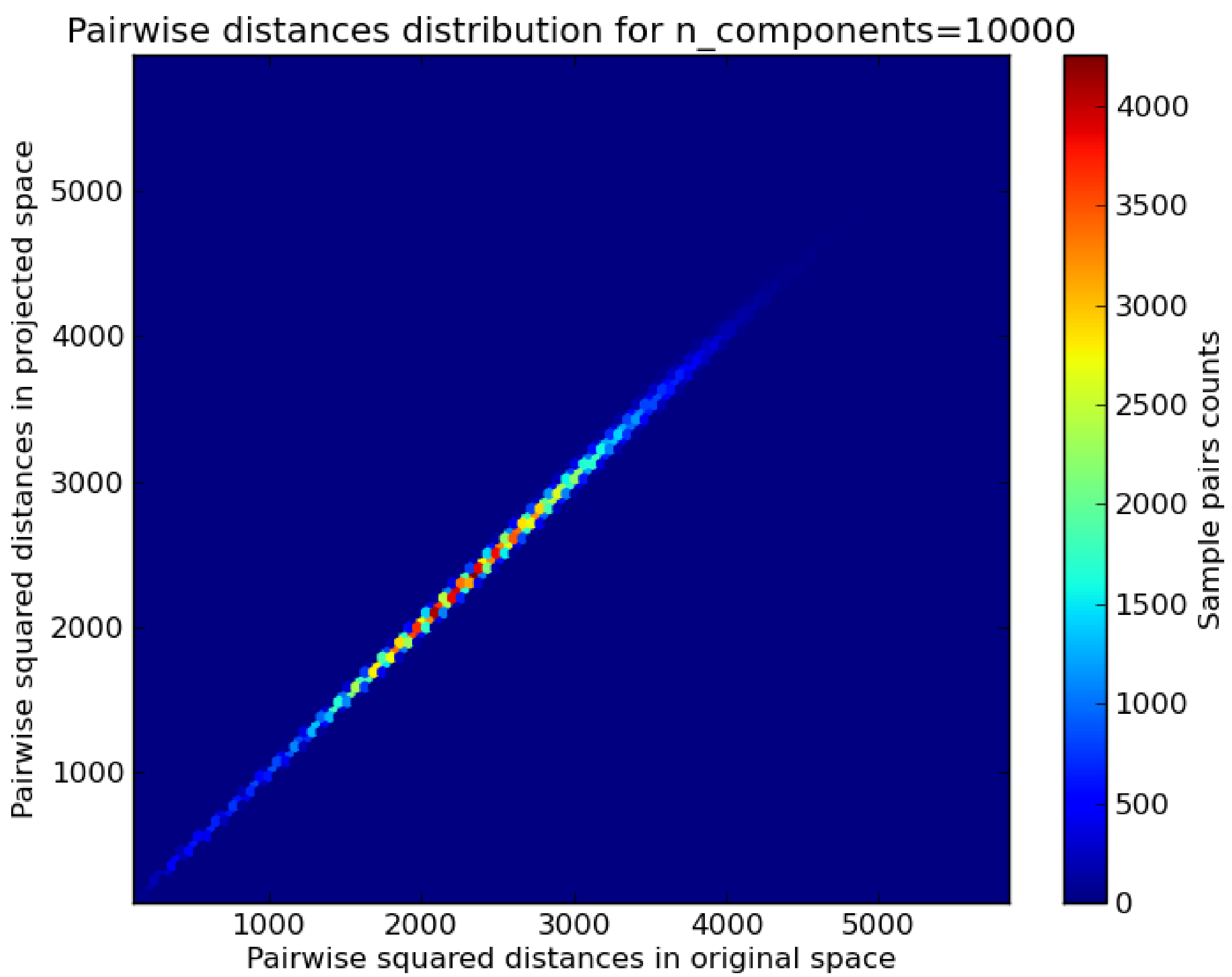
Data: 20-newsgroups, from 100.000 features to 1.000 (1%)

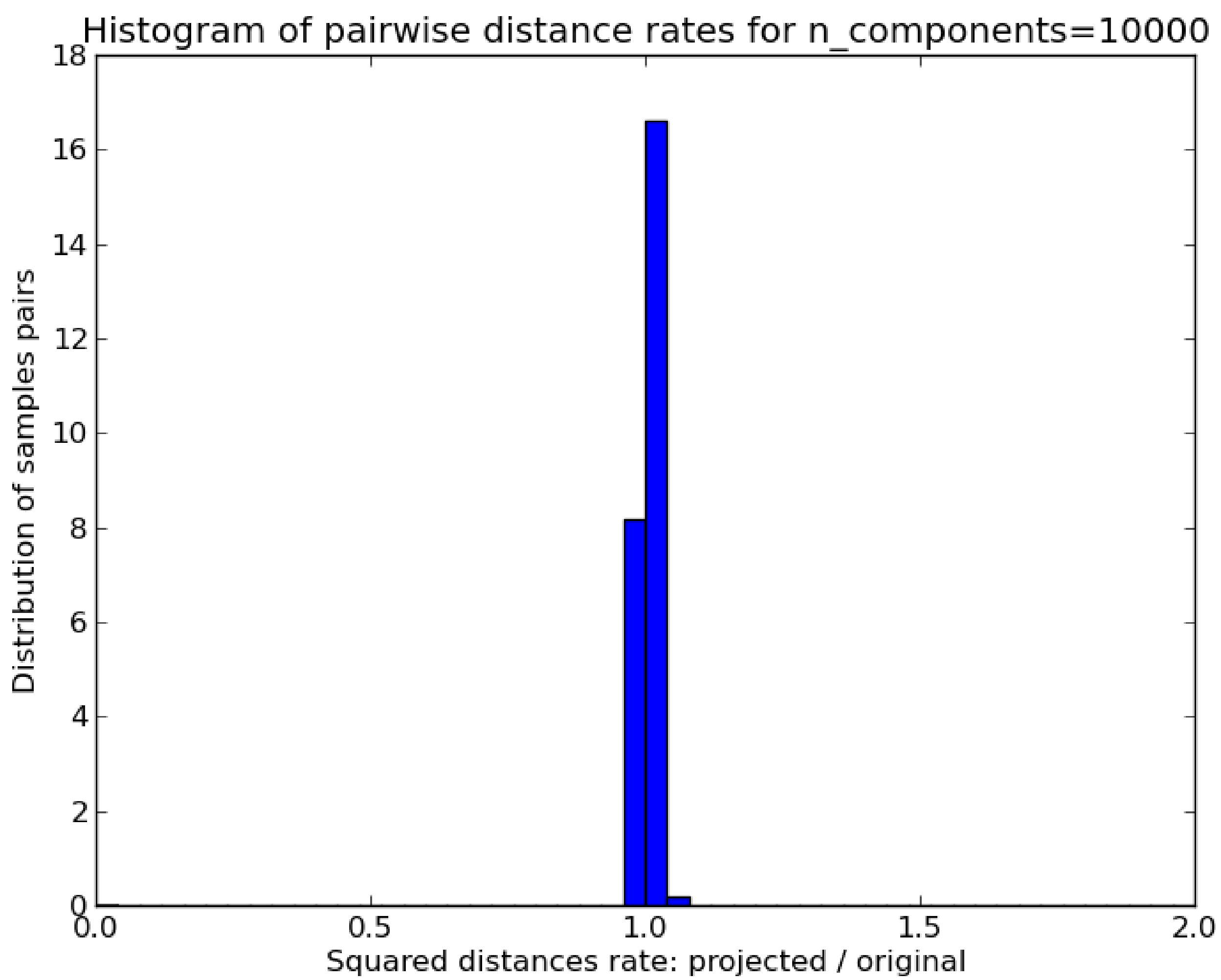




Example: 20-newsgroups data

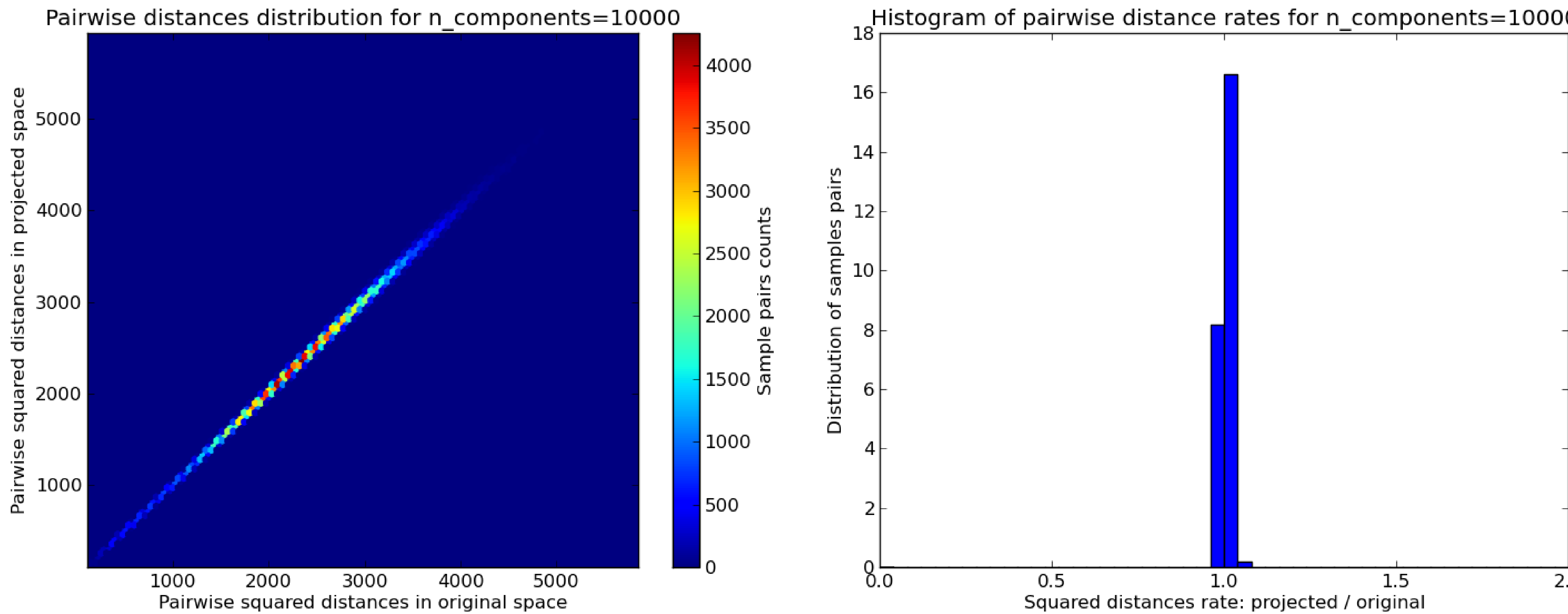
Data: 20-newsgroups, from 100.000 features to 10.000 (10%)





Example: 20-newsgroups data

Data: 20-newsgroups, from 100.000 features to 10.000 (10%)

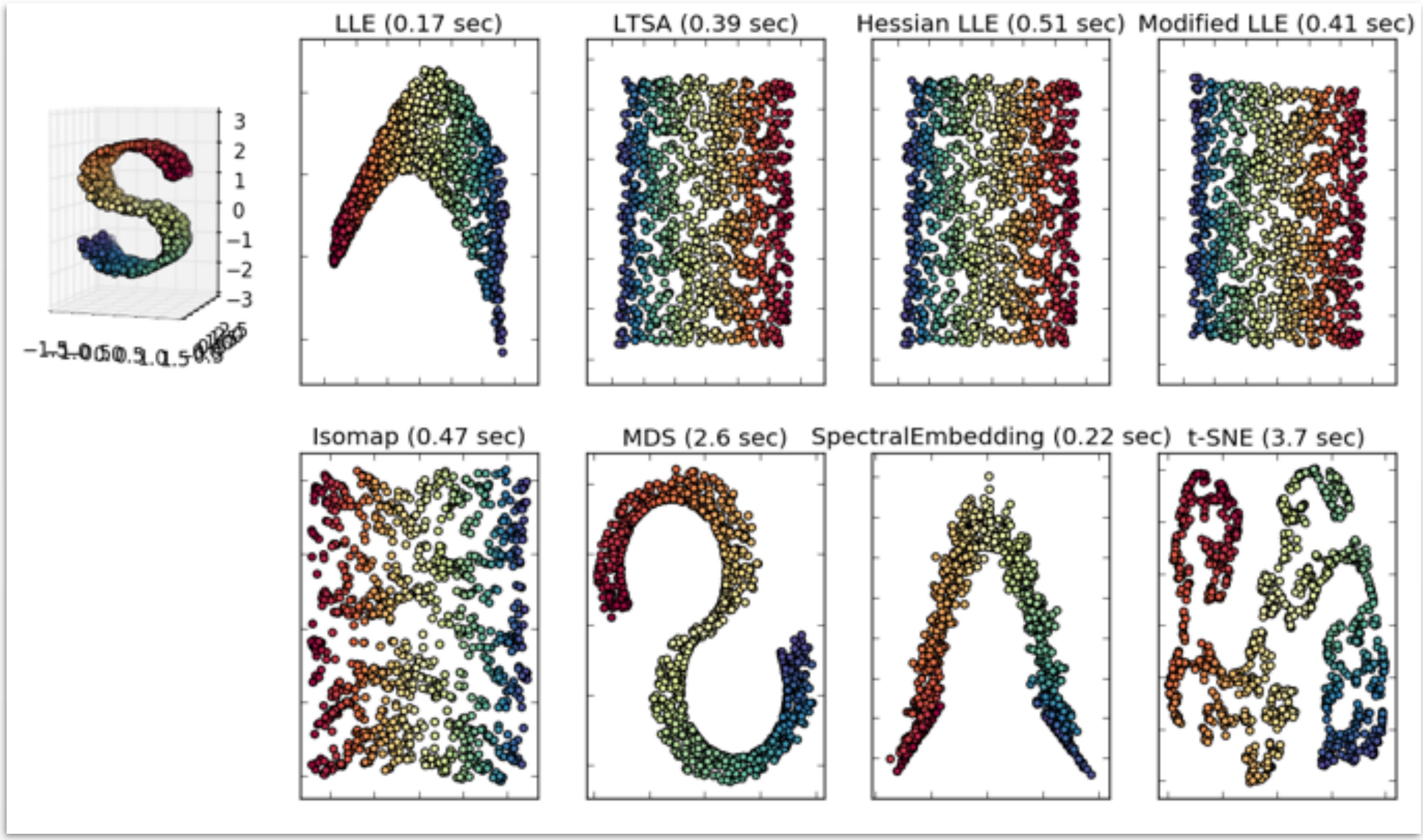


Conclusion: RP preserves distances like PCA,
but faster than PCA number of dimensions is vey large

Stochastic Neighbor Embeddings

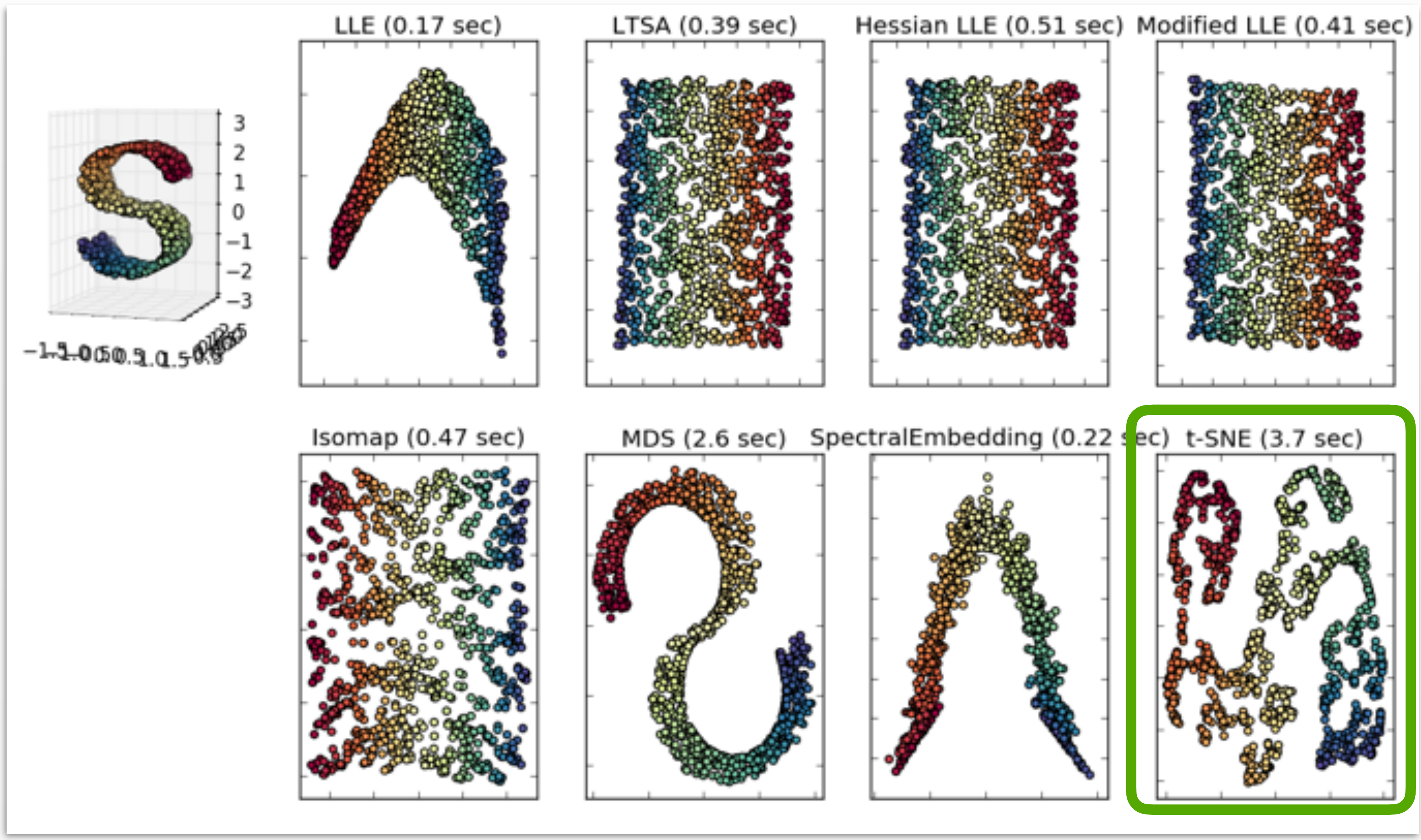
Borrowing from:
Laurens van der Maaten
(Delft -> Facebook AI)

Manifold Learning



Idea: Perform a *non-linear* dimensionality reduction
in a manner that preserves proximity (but not distances)

Manifold Learning



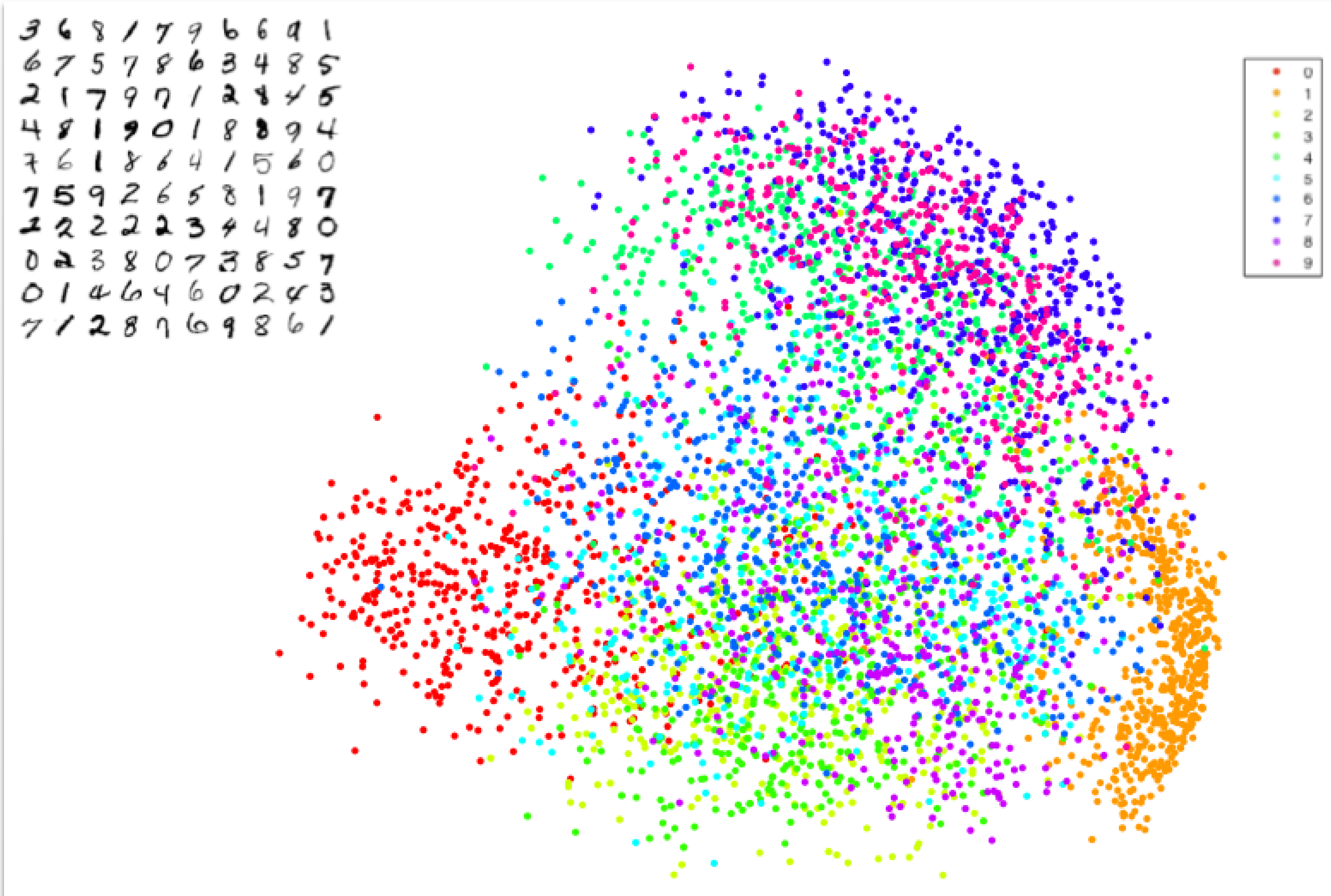
Visualizing data using **t-SNE** [PDF] jmlr.org

L Maaten, G Hinton - Journal of Machine Learning Research, 2008 - jmlr.org

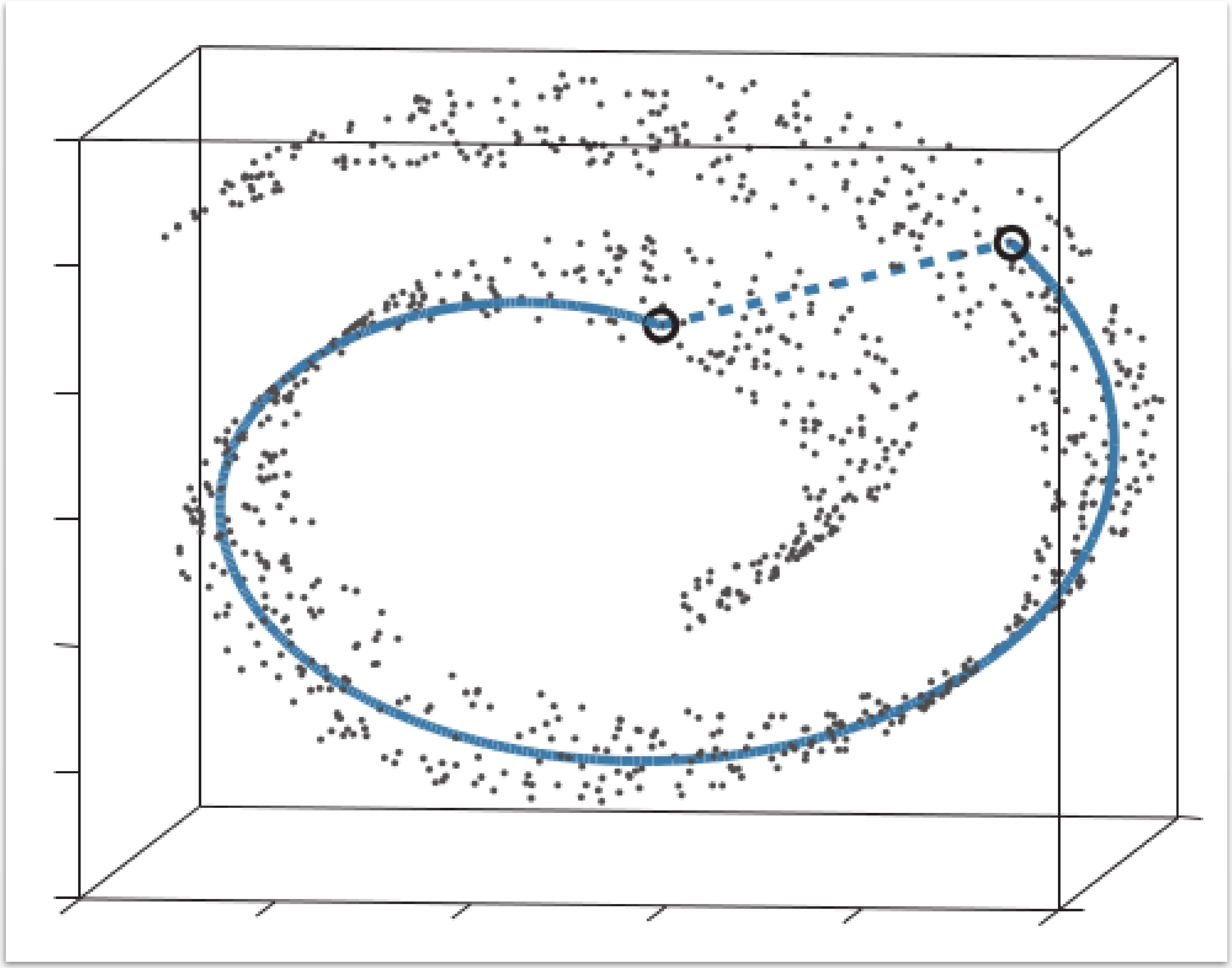
Abstract We present a new technique called" **t-SNE**" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding (Hinton and Roweis, 2002) that is much **...**

Cited by 1771 Related articles All 35 versions Cite Save

PCA on MNIST Digits

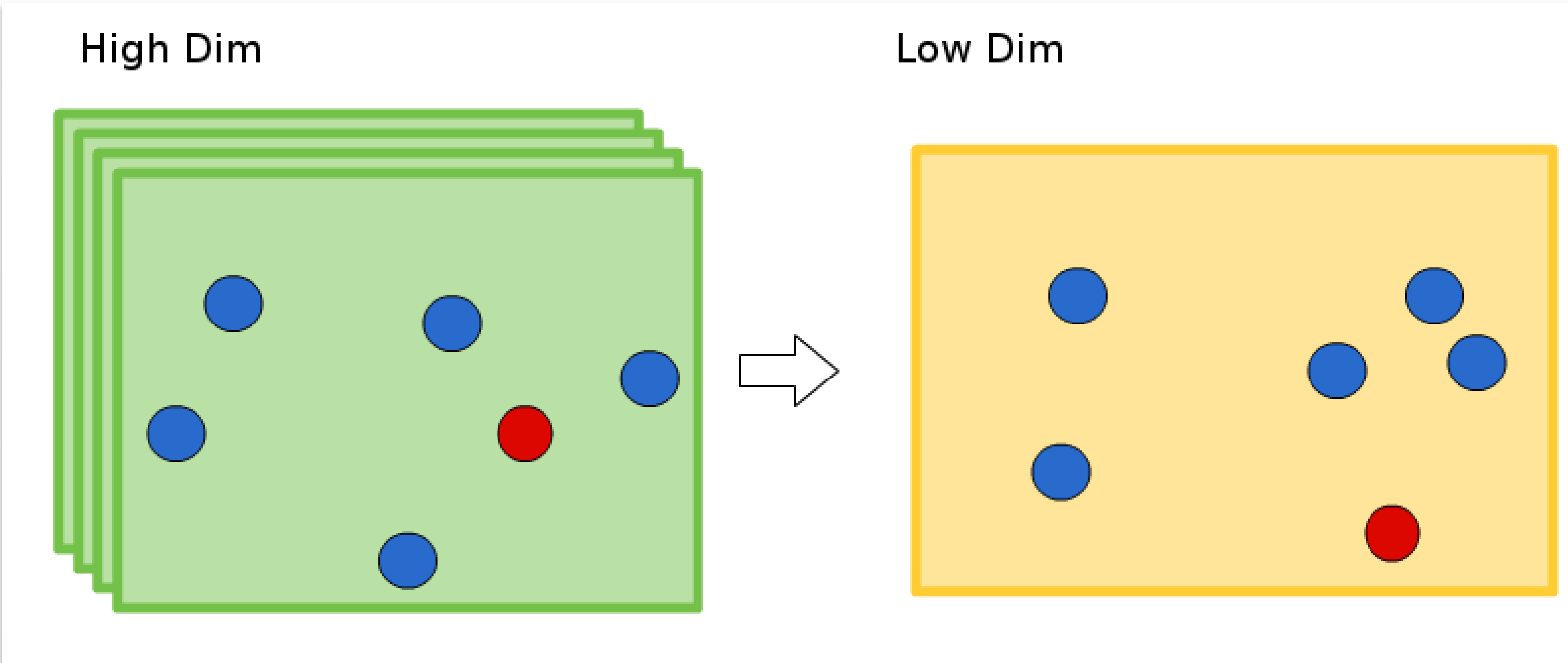


Swiss Roll



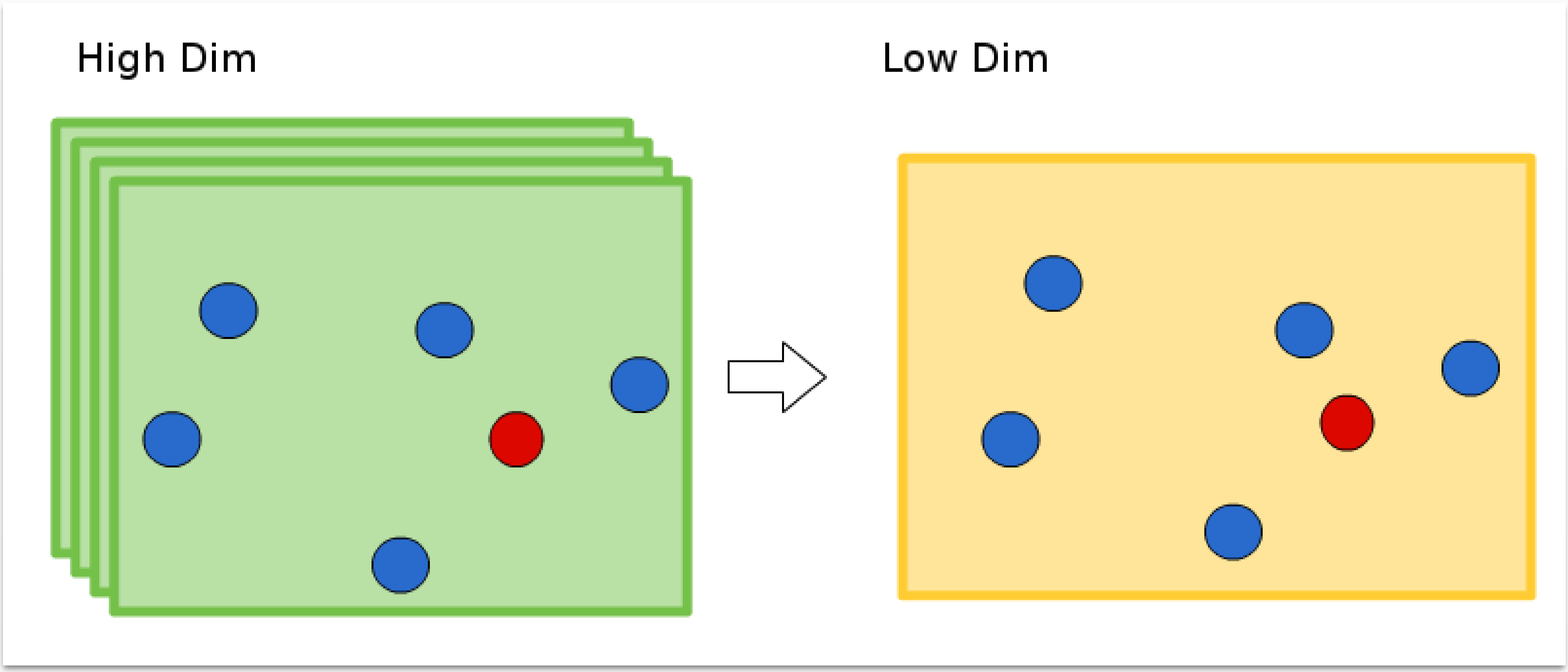
Euclidean distance is not always a good notion of proximity

Non-linear Projection



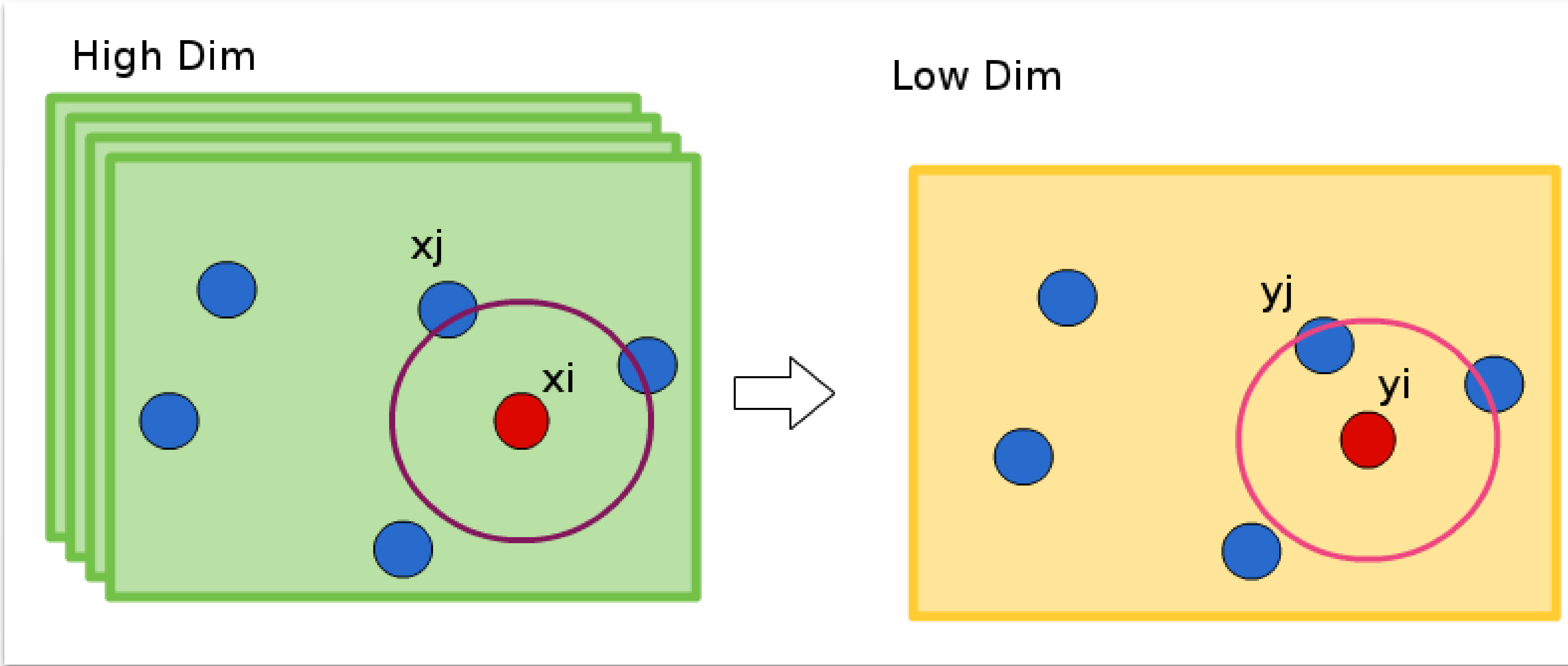
Bad projection: relative position to neighbors changes

Non-linear Projection



Intuition: Want to preserve ***local*** neighborhood

Stochastic Neighbor Embedding



Similarity in high dimension

Similarity in low dimension

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

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

Stochastic Neighbor Embedding

- Similarity of datapoints in High Dimension

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

- Similarity of datapoints in Low Dimension

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

- Cost function

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

Idea: Optimize $\boldsymbol{y_i}$ via gradient descent on C

Stochastic Neighbor Embedding

- Similarity of datapoints in High Dimension

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

- Similarity of datapoints in Low Dimension

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

- Cost function

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

Idea: Optimize $\boldsymbol{y_i}$ via gradient descent on C

Stochastic Neighbor Embedding

Gradient has a surprisingly simple form

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The gradient update with momentum term is given by

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \beta(t)(Y^{(t-1)} - Y^{(t-2)})$$

Stochastic Neighbor Embedding

Gradient has a surprisingly simple form

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The gradient update with momentum term is given by

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \beta(t)(Y^{(t-1)} - Y^{(t-2)})$$

Problem: $p_{j|i}$ is not equal to $p_{i|j}$

Symmetric SNE

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

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

Symmetric SNE

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

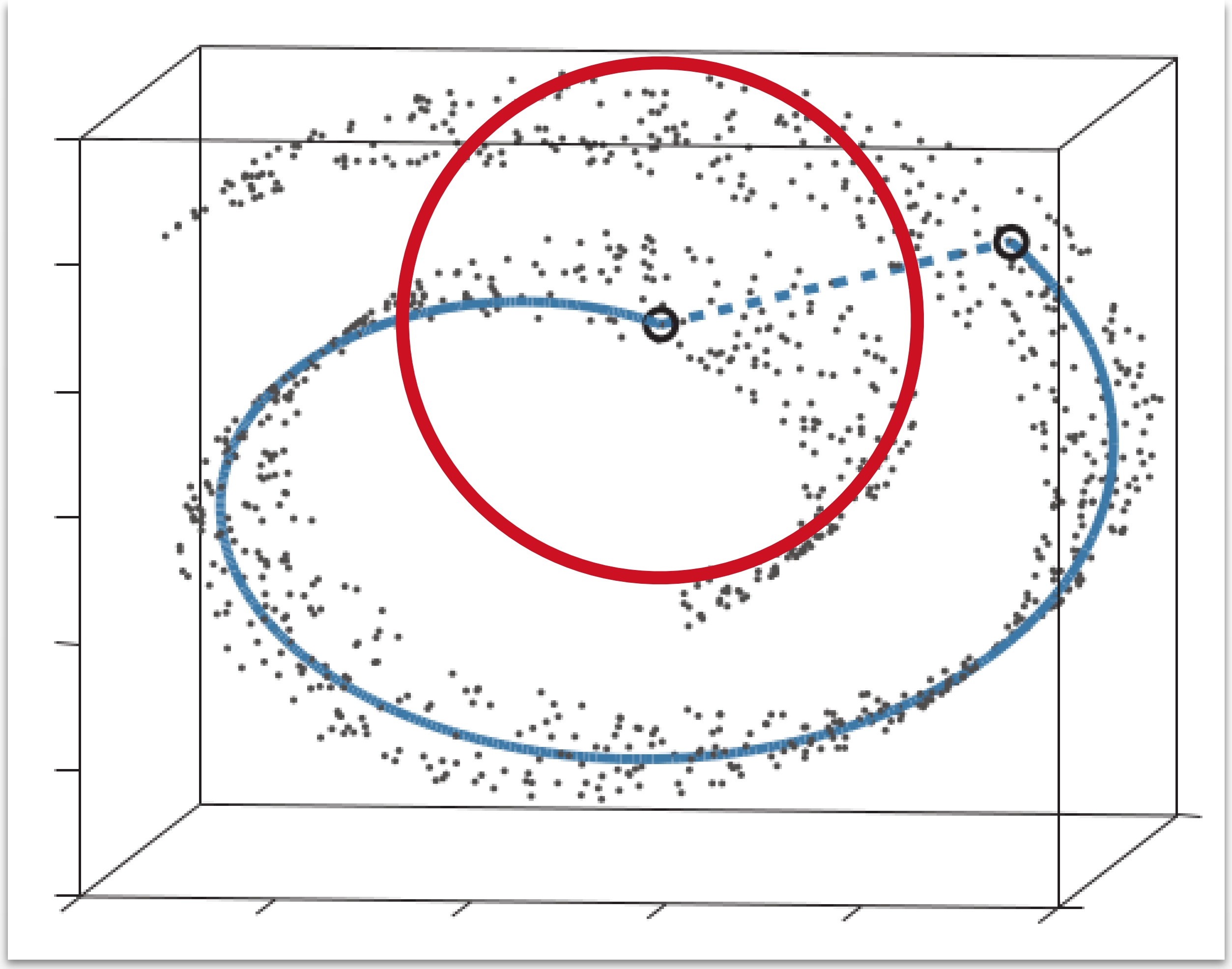
- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

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

Problem: How should we choose σ?

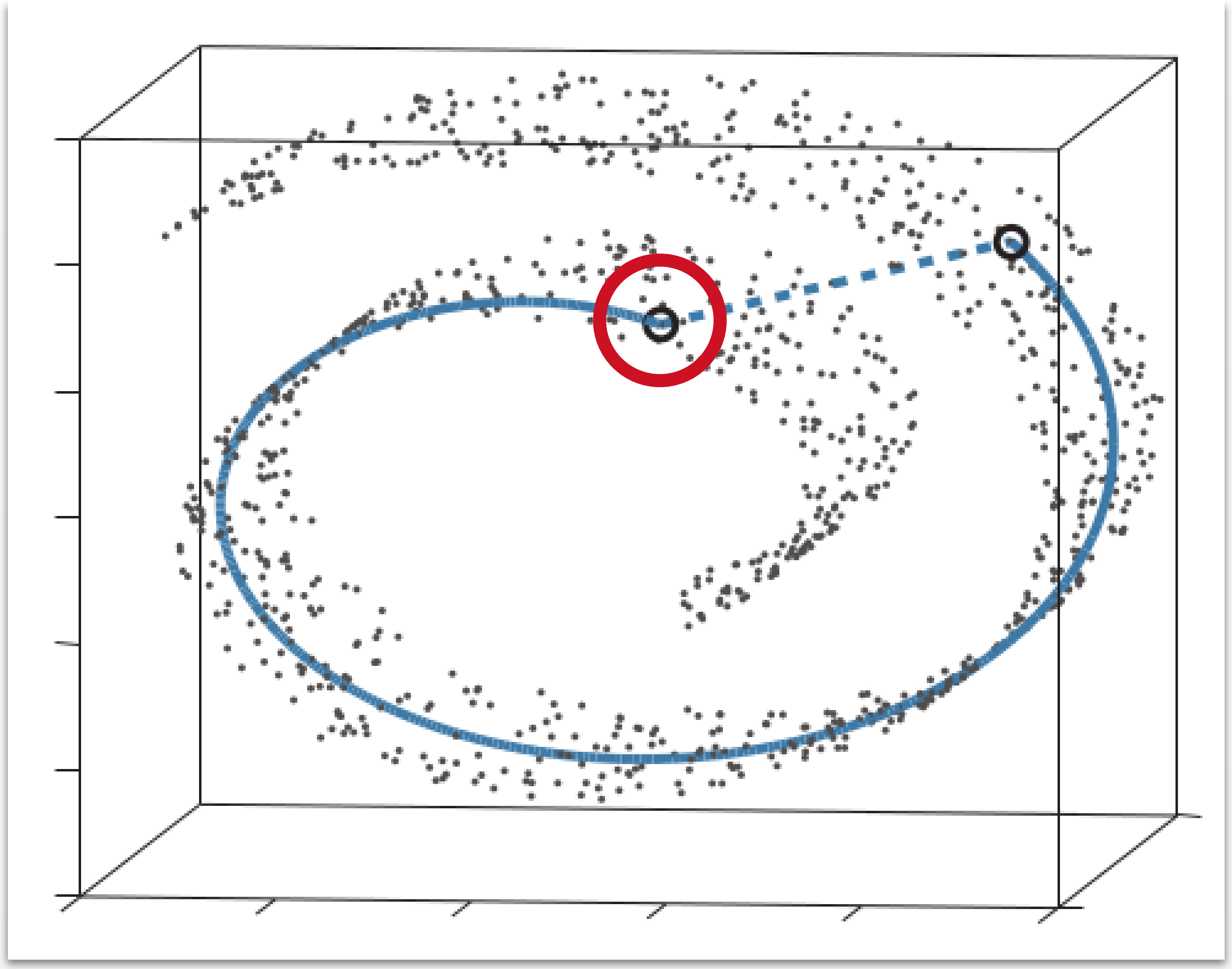
Choosing the bandwidth



$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

Bad σ: Neighborhood is not local in manifold

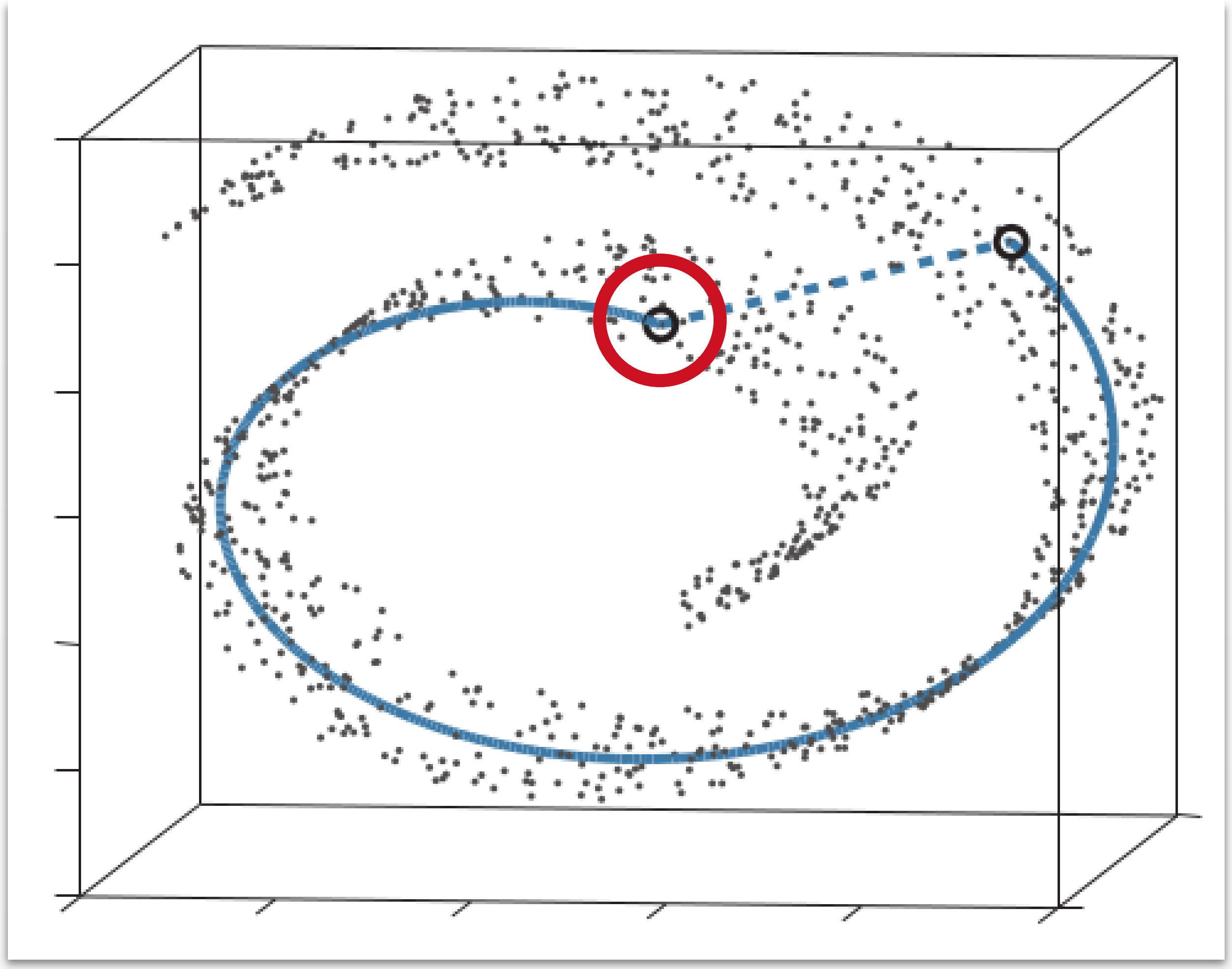
Choosing the bandwidth



$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

Good σ: Neighborhood contains 5-50 points

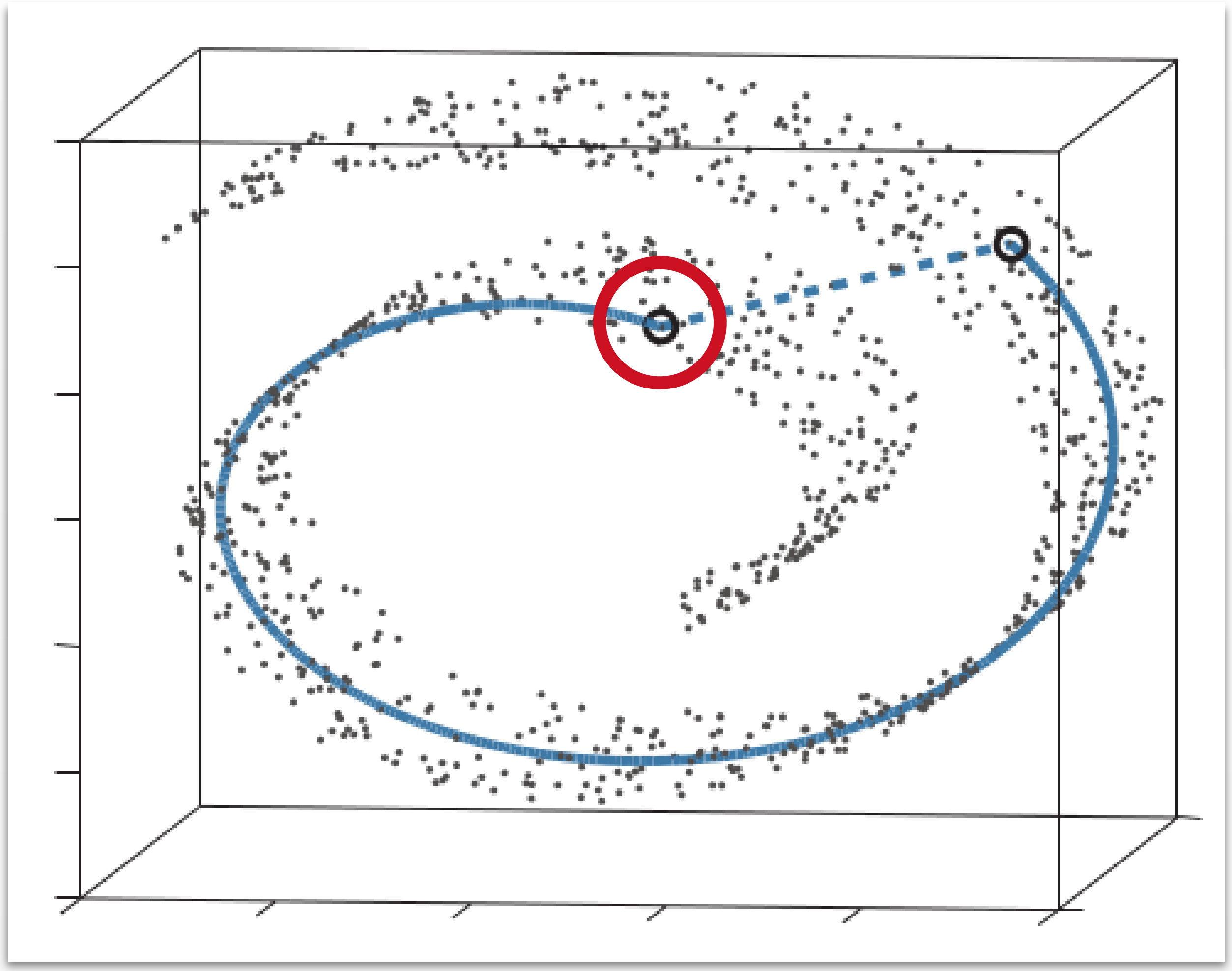
Choosing the bandwidth



$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

Problem: optimal σ may vary if density not uniform

Choosing the bandwidth

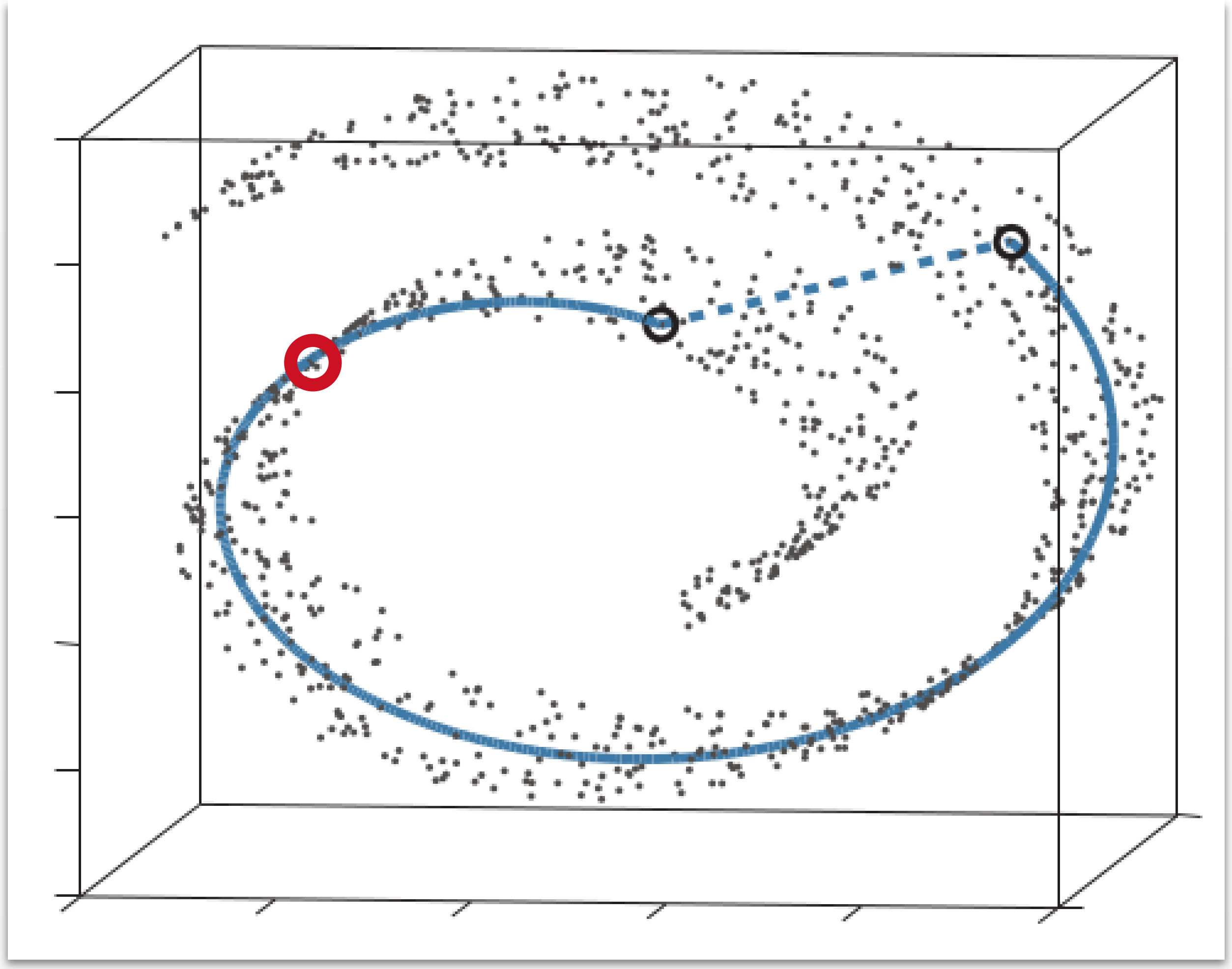


$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

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

Solution: Define σ_i per point.

Choosing the bandwidth

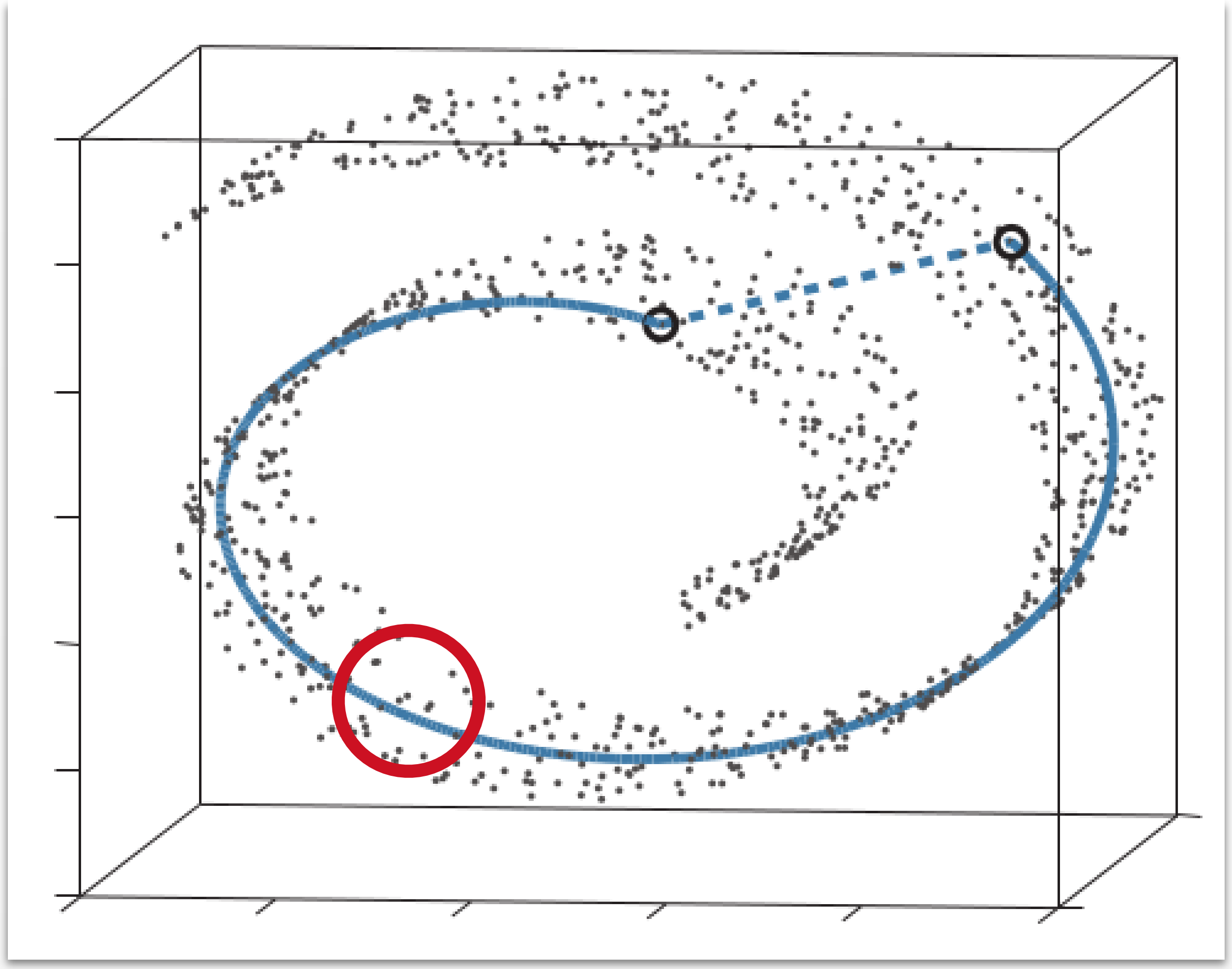


$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

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

Solution: Define σ_i per point.

Choosing the bandwidth

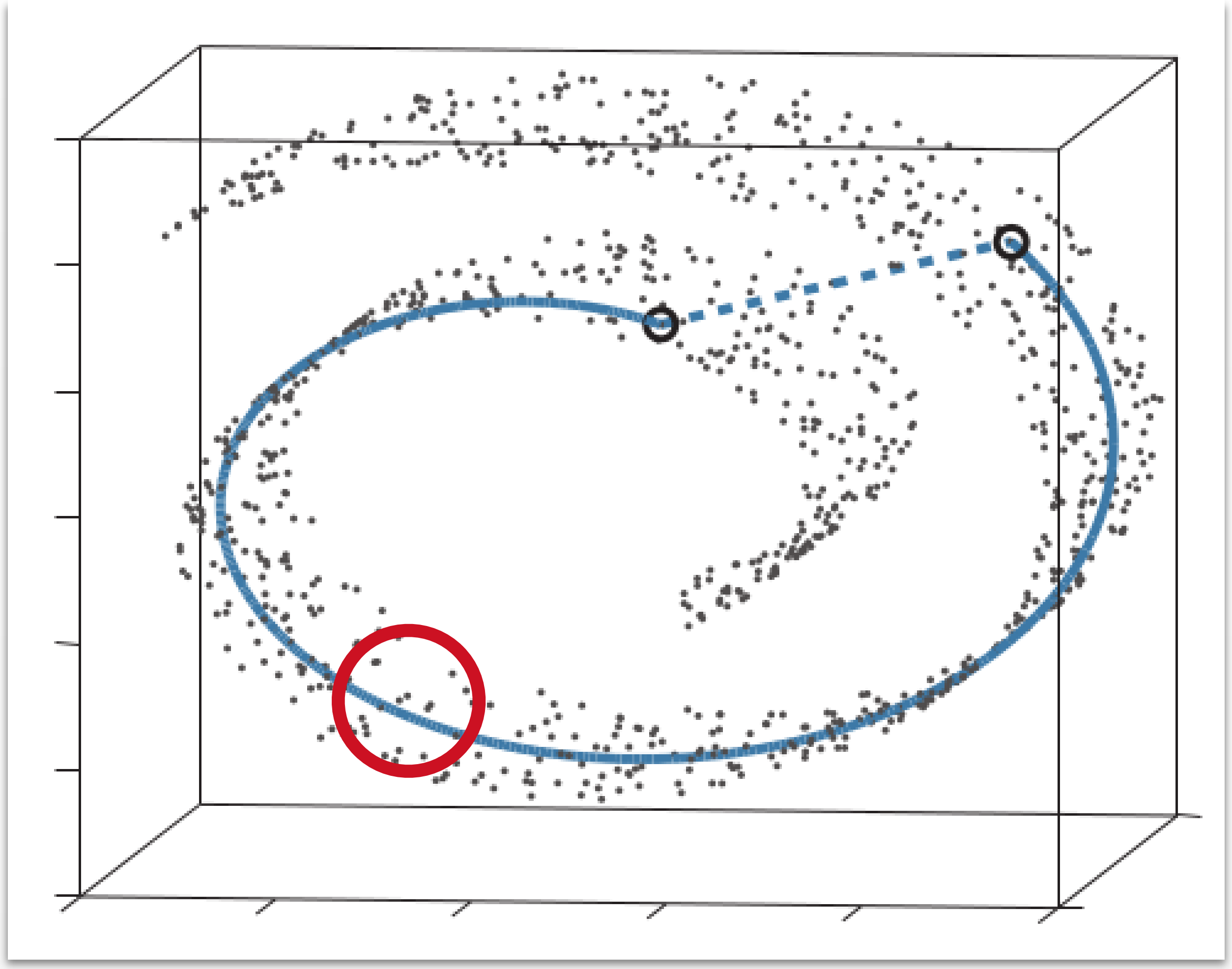


$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

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

Solution: Define σ_i per point.

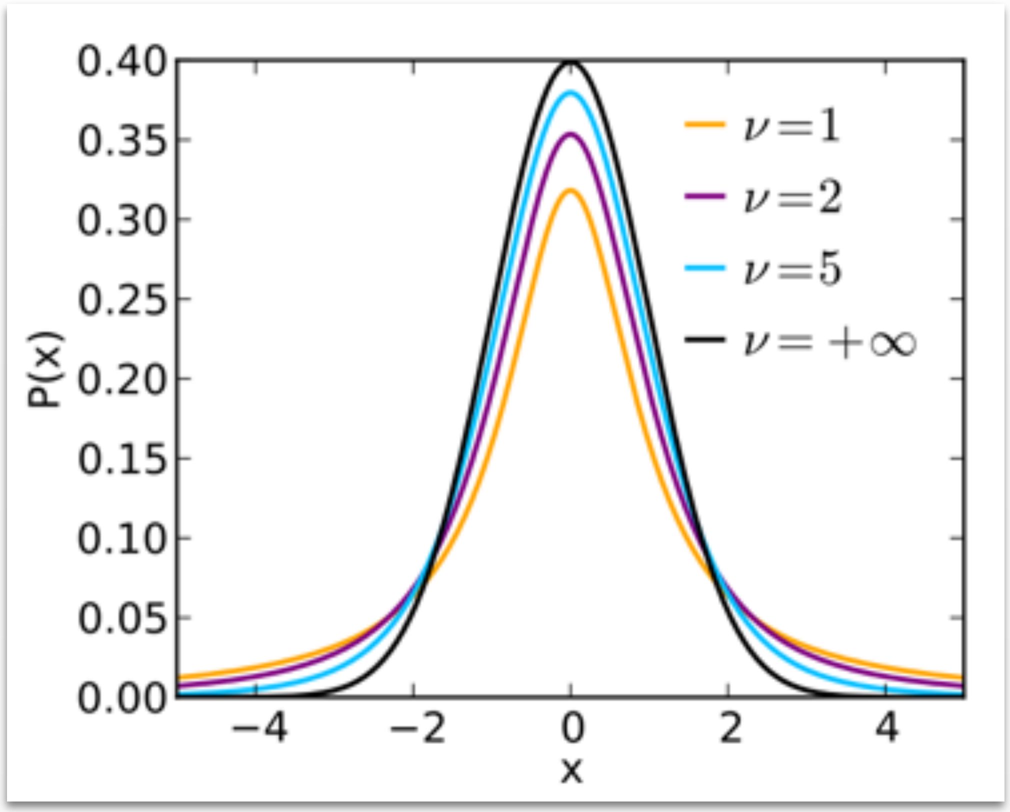
Choosing the bandwidth



$$\mathrm{Perp}(\mathbf{p}_{j|i}) = \exp H(\mathbf{p}_{j|i}) = \exp^{-\sum_j \mathbf{p}_{j|i} \log \mathbf{p}_{j|i}}$$

Set σ_i to ensure constant perplexity

t-SNE: SNE with a t-Distribution



$$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Similarity in High Dimension

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k\neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

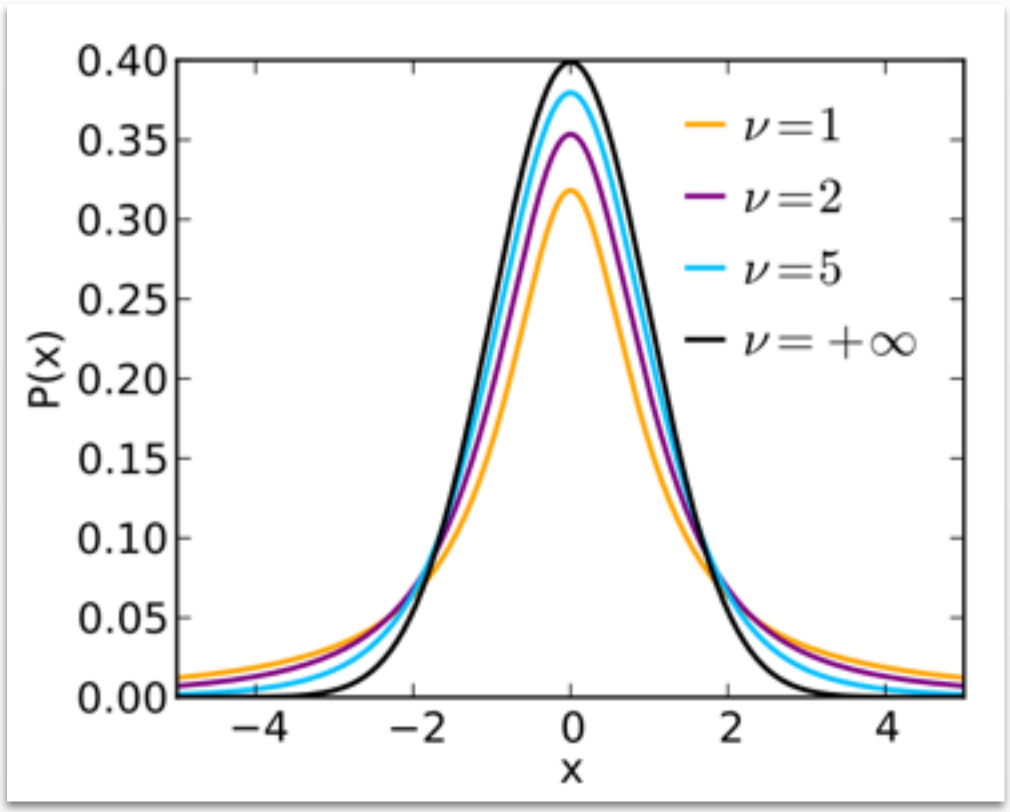
Similarity in Low Dimension

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

Gradient

$$\frac{\partial C}{\partial y_i} = 4\sum_{j\neq i}(p_{ij} - q_{ij})(1+||y_i - y_j||^2)^{-1}(y_i - y_j)$$

t-SNE: SNE with a t-Distribution



$$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Similarity in High Dimension

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

Similarity in Low Dimension

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

Gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

Symmetric SNE

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

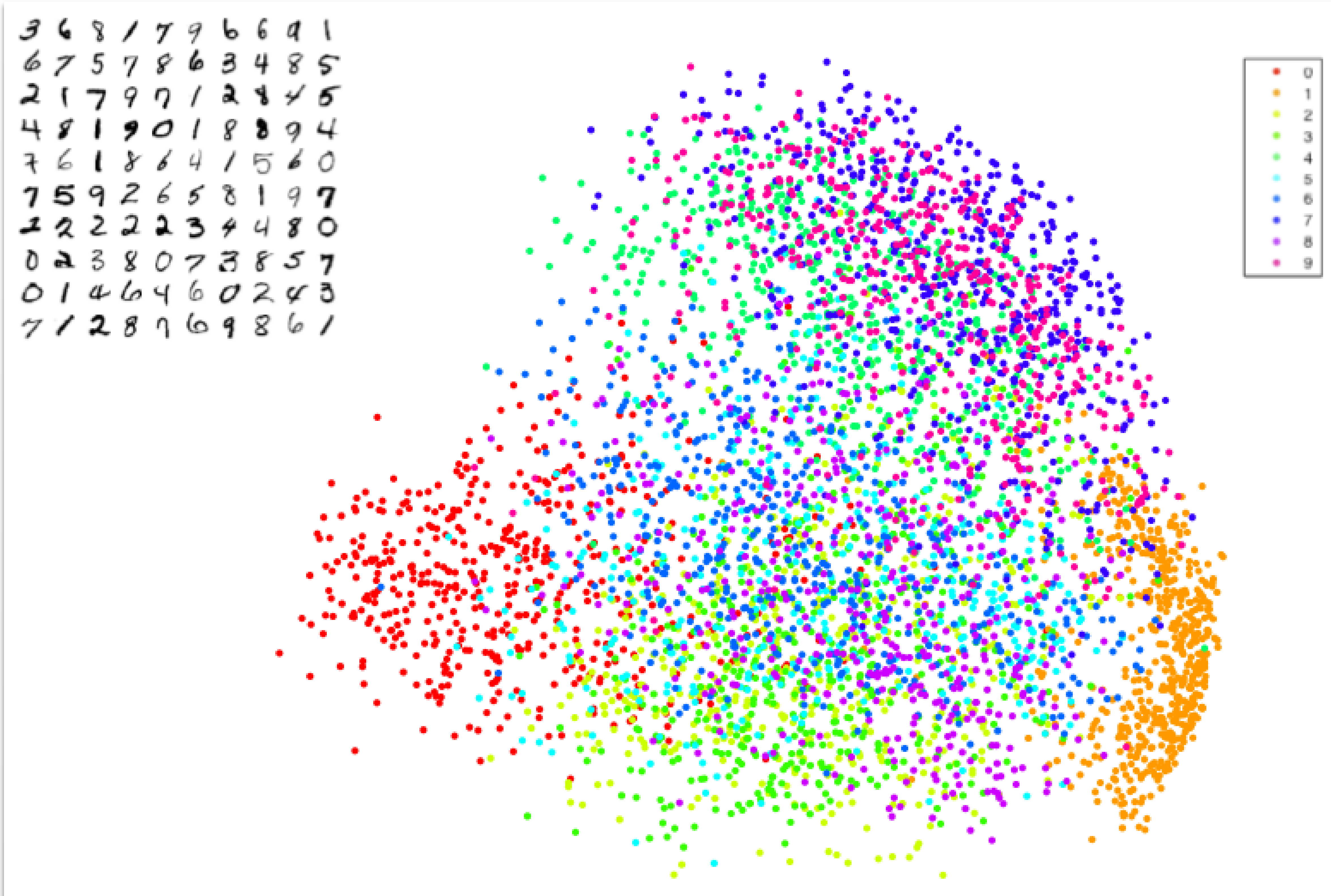
- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

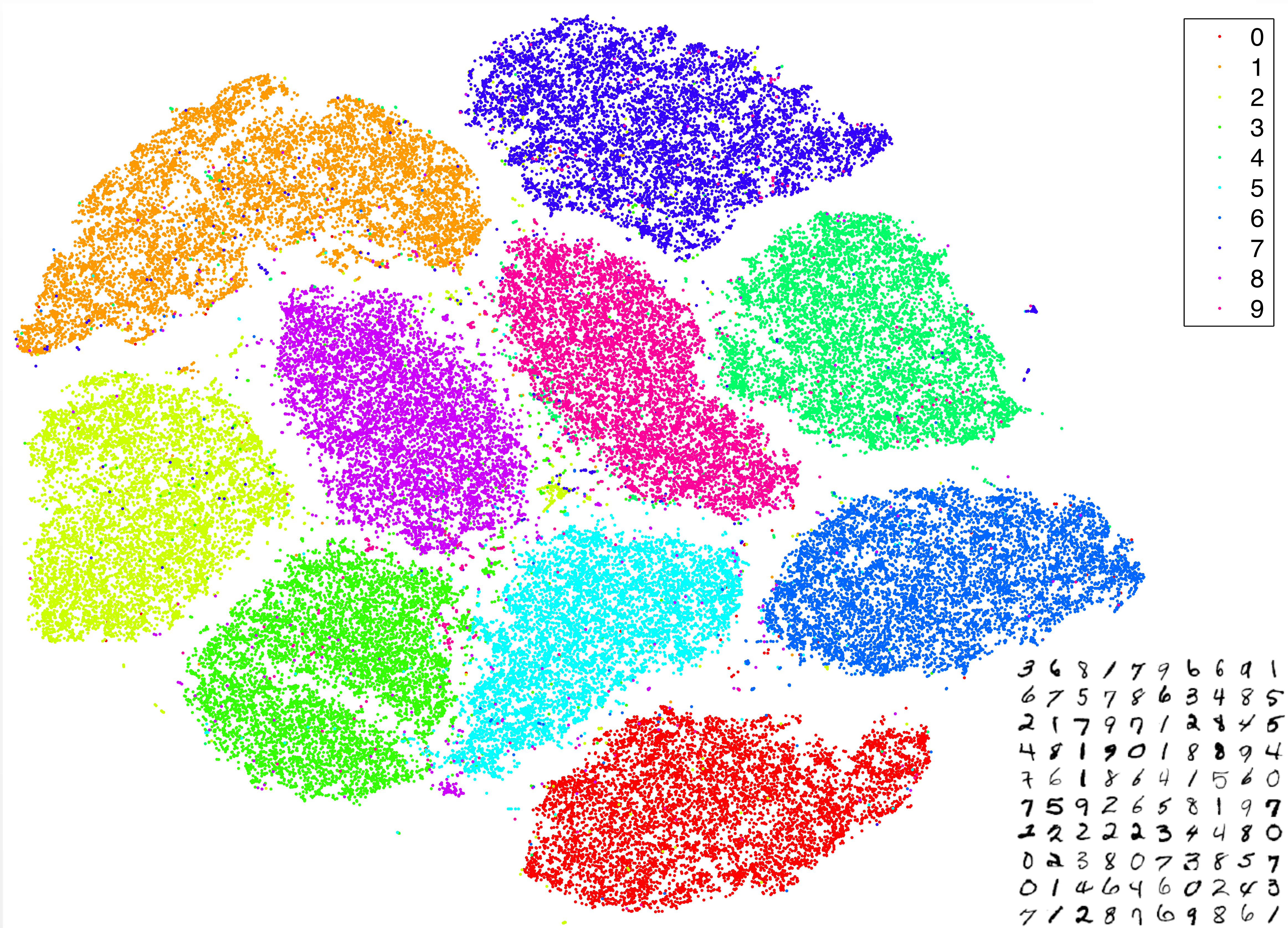
$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_l - y_k||^2)}$$

Problem: How should we choose σ?

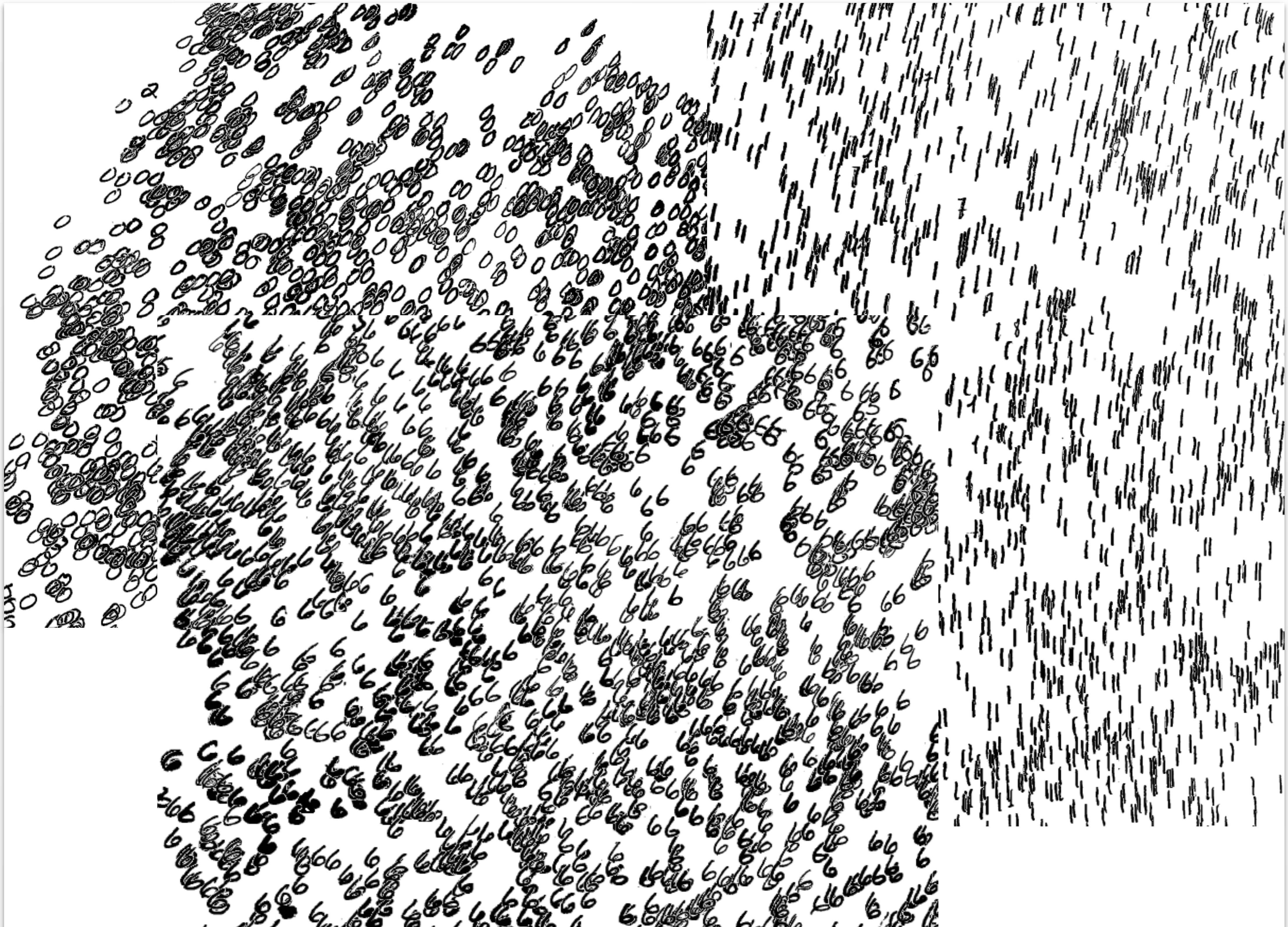
PCA on MNIST Digits



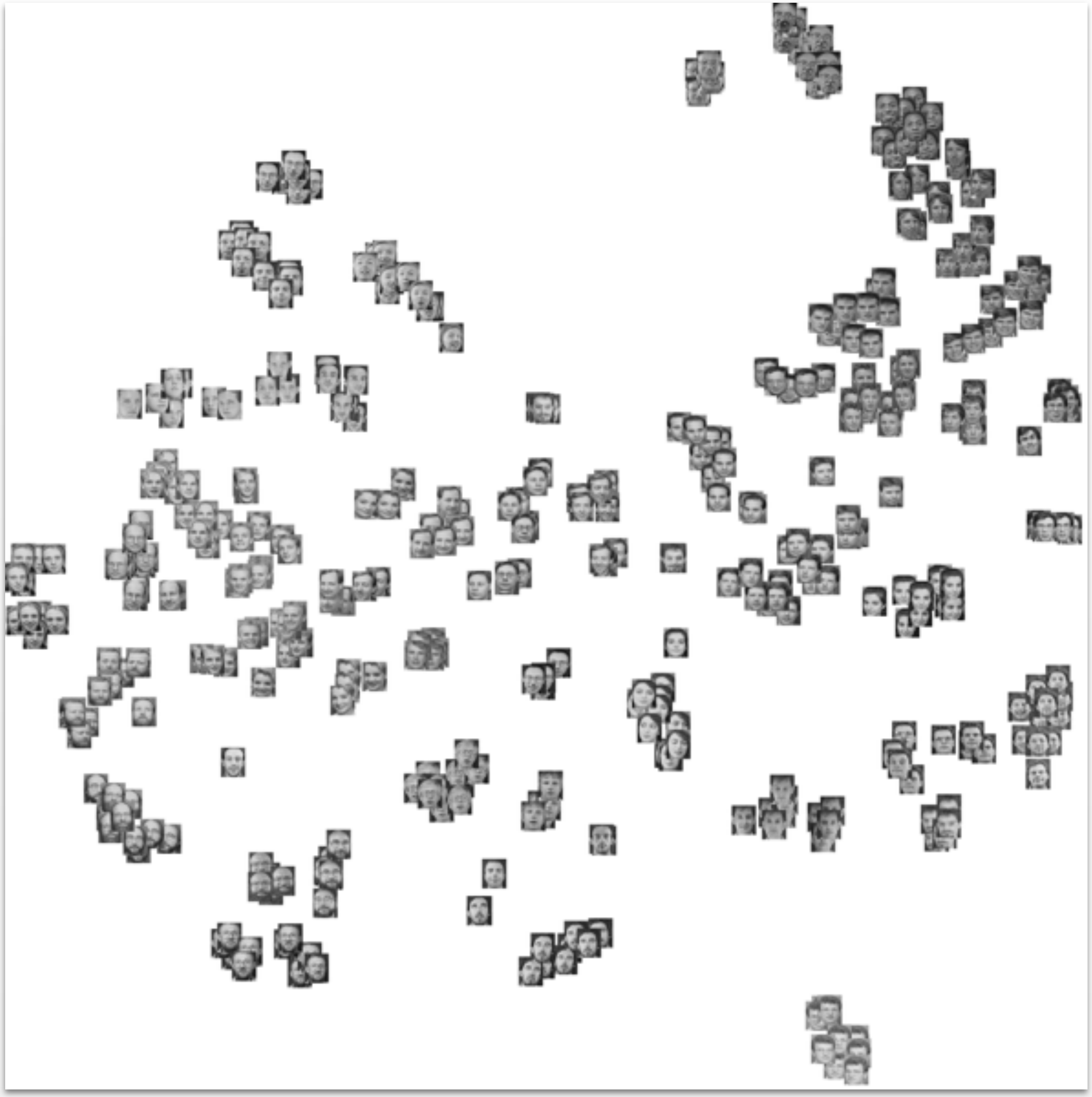
t-SNE on MNIST Digits



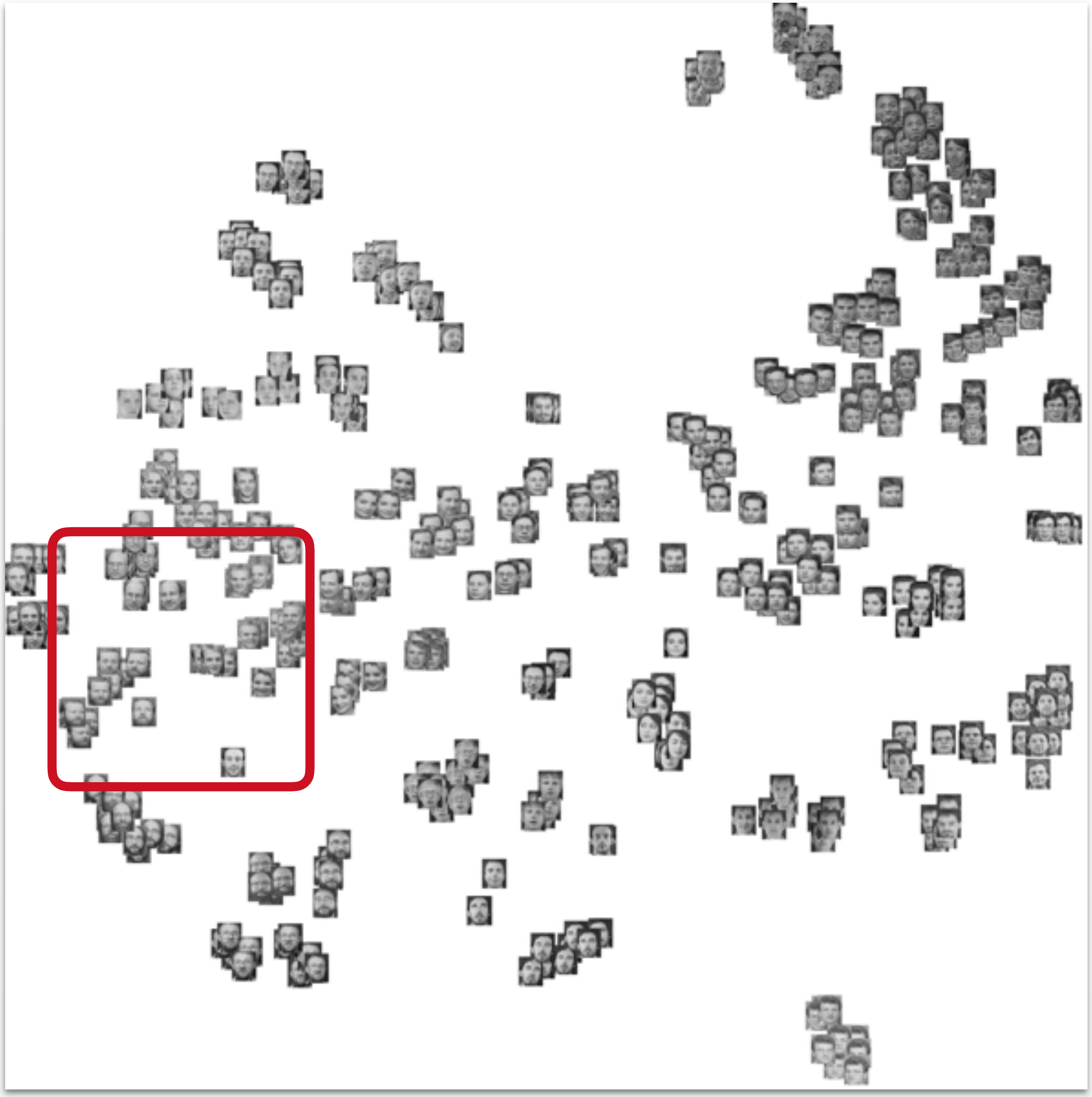
t-SNE on MNIST Digits



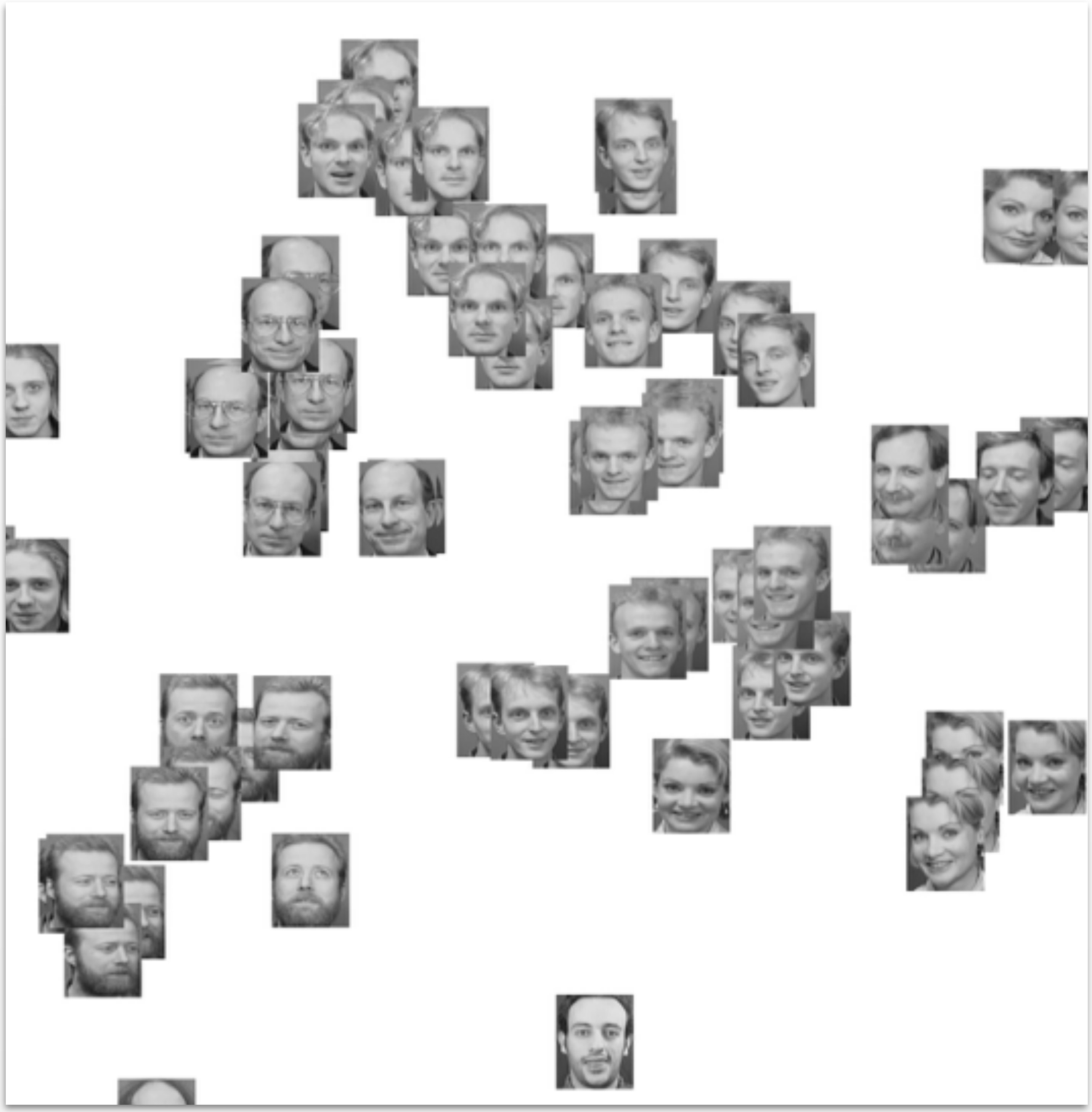
t-SNE on Olivetti Faces



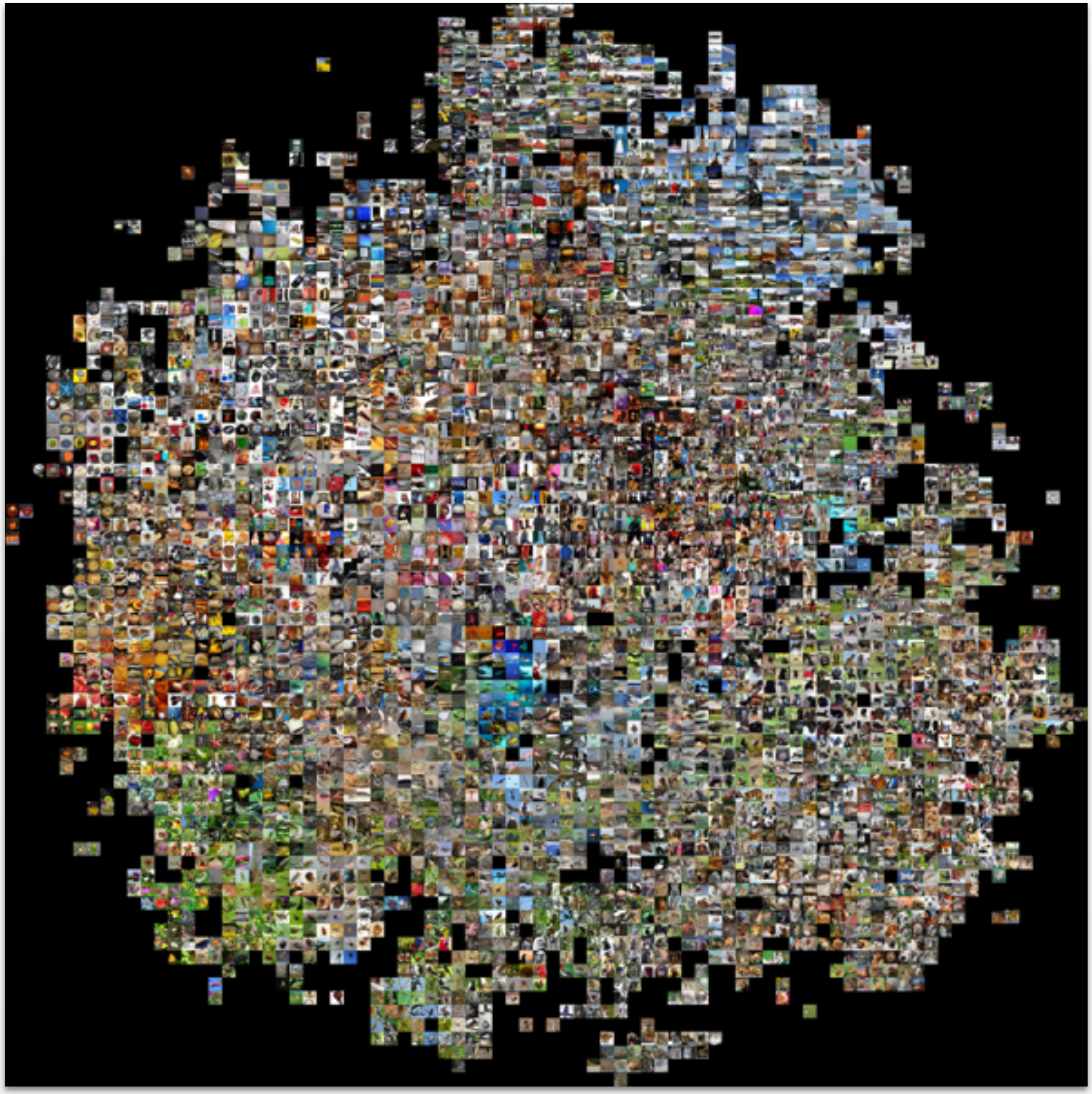
t-SNE on Olivetti Faces



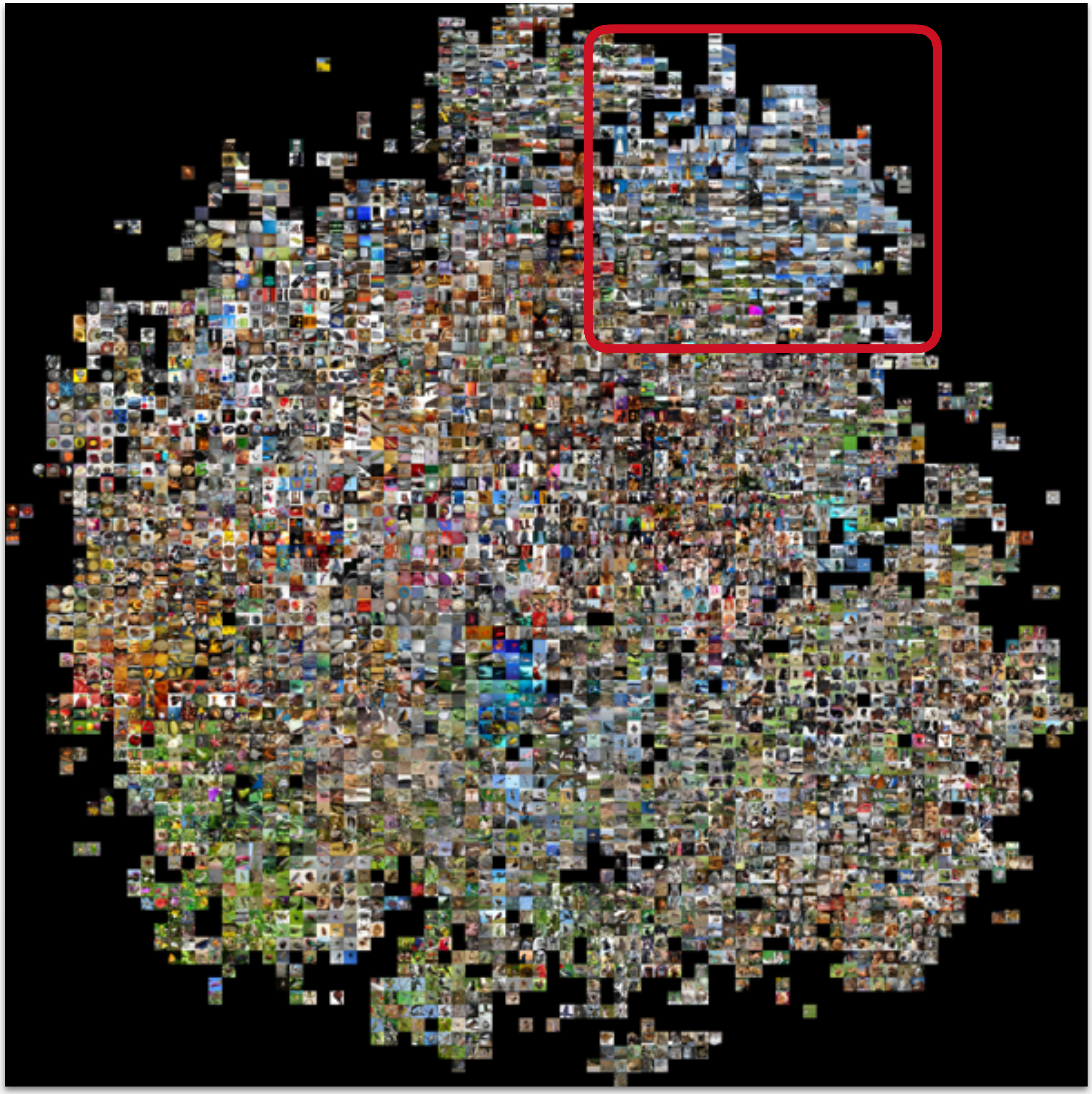
t-SNE on Olivetti Faces



t-SNE on ImageNet



t-SNE on ImageNet



t-SNE on ImageNet

Next lecture: Recommender Systems

	2			4	5	2.94*
	5		4			1
			5		2	2.48*
		1		5		4
			4			2
	4	5		1		1.12*
sim(i,j)	-1	-1	0.86	1	NA	