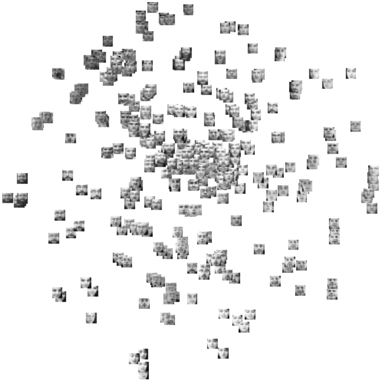
# Basics for Enhanced Visualization: 3D/Data Dimensionality reduction



Rodrigo Cabral

Polytech Nice - Data Science

cabral@unice.fr

Source: *Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization*. Venna J. *et al.* Journal of Machine Learning Research. Volume 11, pages 451-490. 2010.

# Outline

1. Introduction
2. Principal component analysis and its kernel version
3. Multidimensional scaling and ISOMAP
4. Neighborhood structure preservation
5. Conclusions

---

These slides are based on the following lectures:

`https://web.stanford.edu/class/stats202/content/lec24.pdf`
`http://dac.lip6.fr/master/wp-content/uploads/2016/09/fdms_cours4_2016_2017.pdf`

## How to visualize high dimensional data?

- Example: New York Times *corpus*
- 1.8 million articles with tagged subjects: politics, economy, music, visual arts *etc*.
- We can assign to each tagged article a vector with the histogram of its words.

# Introduction

## How to visualize high dimensional data?

- Example: New York Times *corpus*
- 1.8 million articles with tagged subjects: politics, economy, music, visual arts *etc*.
- We can assign to each tagged article a vector with the histogram of its words.
- Objective: can we visualize the data to see if the articles can be clustered by subject?

# Introduction

## How to visualize high dimensional data?

- Example: subset of New York Times *corpus*
- 57 articles about visual arts (A) and 45 articles about music (M).
- For this subset we have 4431 different words.

# Introduction

## How to visualize high dimensional data?

- Example: subset of New York Times *corpus*
- 57 articles about visual arts (A) and 45 articles about music (M).
- For this subset we have 4431 different words.
- An example of data would be

| Article | Tag | Word frequency | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | "abandoned" | ⋯ | "art" | ⋯ | "composers" | ⋯ | "opera" | ⋯ |
| 1 | A | 0 | | 0.001 | | 0 | | 0 | |
| 2 | M | 0 | | 0 | | 0.002 | | 0.001 | |

# Introduction

## How to visualize high dimensional data?

- Example: subset of New York Times *corpus*
- An example of data would be

| Article | Tag | Word frequency | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | "abandoned" | ··· | "art" | ··· | "composers" | ··· | "opera" | ··· |
| 1 | A | 0 | | 0.001 | | 0 | | 0 | |
| 2 | M | 0 | | 0 | | 0.002 | | 0.001 | |

- Each word correspond to a variable $\Longrightarrow$
  each text is a point in the 4431-dimensional space!
- Problem: how can we visualize it?

# How to visualize high dimensional data?

- Problem: how can we visualize it?
- Solution:
  1. Map the points from the high dimensional space with $D$ dimensions (D=4431) to a low dimensional space with $d \ll D$, such that most of the information is retained.
  2. The low dimensional space can then be mapped to visual aesthetics: space, color, size, shape *etc*. Often $d = 2$ and aesthetics are spaces (scatter plot).

# How to visualize high dimensional data?

- Problem: how can we visualize it?
- Solution:
    1. Map the points from the high dimensional space with $D$ dimensions (D=4431) to a low dimensional space with $d \ll D$, such that most of the information is retained.
    2. The low dimensional space can then be mapped to visual aesthetics: space, color, size, shape *etc*. Often $d = 2$ and aesthetics are spaces (scatter plot).

1. is **dimensionality reduction** and it is the subject of this lecture.
2. is visual mapping and it is related to perception.

- The meaning of what is “information" quantitatively is what changes from one method to another.

# Principal component analysis and its kernel version

# Minimizing reconstruction error with a linear transformation

- $\mathbf{x}_i \in \mathbb{R}^D$: the i-th point in the high-dimensional space.
- $\mathbf{y}_i \in \mathbb{R}^d$: the i-th point in the low dimensional space.

$$\mathbf{y}_i = \mathbf{L}\mathbf{x}_i$$

- $\mathbf{L} \in \mathbb{R}^{d \times D}$: linear mapping (matrix).
- Rows of $\mathbf{L}$ are orthogonal: $\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{I}_d$

# Minimizing reconstruction error with a linear transformation

- $\mathbf{x}_i \in \mathbb{R}^D$: the i-th point in the high-dimensional space.
- $\mathbf{y}_i \in \mathbb{R}^d$: the i-th point in the low dimensional space.

$$\mathbf{y}_i = \mathbf{L}\mathbf{x}_i$$

- $\mathbf{L} \in \mathbb{R}^{d \times D}$: linear mapping (matrix).
- Rows of $\mathbf{L}$ are orthogonal: $\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{I}_d$

- **Reconstruction:** it can be shown that optimal reconstruction in the original space is

$$\hat{\mathbf{x}}_i = \mathbf{L}^\mathsf{T}\mathbf{y}_i$$

## Minimizing reconstruction error with a linear transformation

- Objective: minimize the total reconstruction error in the original space.

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 = \sum_{i=1}^{N} \left\|\mathbf{x}_i - \mathbf{L}^{\mathsf{T}}\mathbf{L}\mathbf{x}_i\right\|_2^2 \\
\text{with respect to} & \mathbf{L} \\
\text{subject to} & \mathbf{L}\mathbf{L}^{\mathsf{T}} = \mathbf{I}_d
\end{array}
$$

# PCA and kernel PCA

## Minimizing reconstruction error with a linear transformation

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 = \sum_{i=1}^{N} \left\|\mathbf{x}_i - \mathbf{L}^\mathsf{T}\mathbf{L}\mathbf{x}_i\right\|_2^2 \\
\text{with respect to} \quad & \mathbf{L} \\
\text{subject to} \quad & \mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{I}_d
\end{aligned}$$

- Solution: $\mathbf{L} = [\mathbf{U}]_{1:d}^\mathsf{T}$

  where $[\mathbf{U}]_{1:d}$ are the $d$ columns of $\mathbf{U}$ from the SVD of $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N] = \mathbf{U\Sigma V}^\mathsf{T}$ corresponding to the the $d$ largest singular values.

## Minimizing reconstruction error with a linear transformation

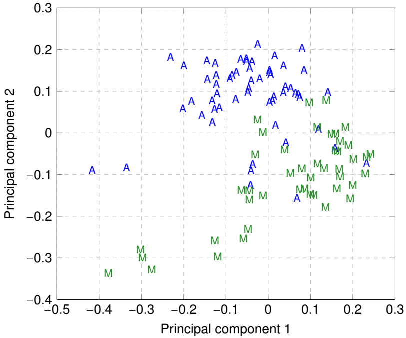
- This is **principal component analysis (PCA)**.
- The rows of **L** correspond to the orthogonal directions with most data variation:
  principal axes or principal components.
- The reduced dimension observations are $\mathbf{y}_i = [\boldsymbol{\Sigma}]_{1:d} \left[\mathbf{V}^{\mathsf{T}}\right]_i$:
  principal components scores.

## Minimizing reconstruction error with a linear transformation

- This is **principal component analysis (PCA)**.
- The rows of $\mathbf{L}$ correspond to the orthogonal directions with most data variation:
  principal axes or principal components.
- The reduced dimension observations are $\mathbf{y}_i = [\boldsymbol{\Sigma}]_{1:d} \left[\mathbf{V}^{\mathsf{T}}\right]_i$:
  principal components scores.
- Relative squared reconstruction error: $\varepsilon = \dfrac{\operatorname{trace}\left([\boldsymbol{\Sigma}]_{d+1:D}\right)}{\operatorname{trace}\left(\boldsymbol{\Sigma}\right)}$
- It works well if data lies in a low dimensional subspace:
  line or a plane!

# Example of PCA

- Application to the subset of the NYT *corpus*:

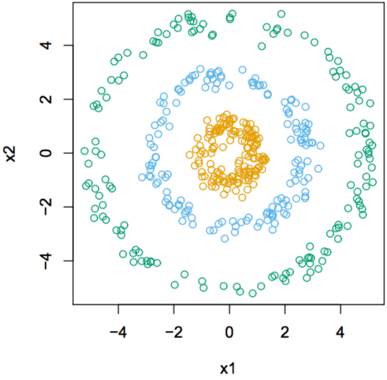



- Subset is almost linearly separable.
- We can use a linear SVM if we want to classify articles.
- Relative reconstruction error is quite high: 0.8 $\Longrightarrow$ which increases the hope of linear separability.

# PCA and kernel PCA

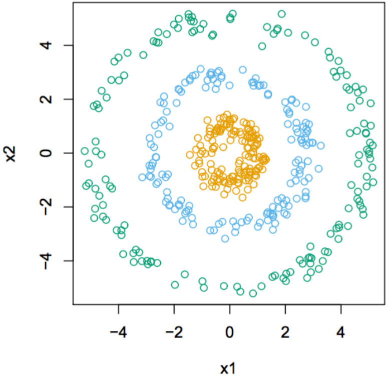
## A more difficult example

- Can we reduce the dimension of these data to have separable classes?



# A more difficult example

- Can we reduce the dimension of these data to have separable classes?




- No, there is no linear subspace structure. All directions have equal variation.

## Reconstruction in a transformed space

- To make PCA non linear, we transform the variables with a non linear function $\Phi(\cdot)$

# Reconstruction in a transformed space

- To make PCA non linear, we transform the variables with a non linear function $\Phi(\cdot)$
- This leads to a different reconstruction problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} \|\Phi(\mathbf{x}_i) - \tilde{\mathbf{x}}_i\|_2^2 = \sum_{i=1}^{N} \left\|\Phi(\mathbf{x}_i) - \mathbf{L}^T\mathbf{L}\Phi(\mathbf{x}_i)\right\|_2^2 \\ \text{with respect to} & \mathbf{L} \\ \text{subject to} & \mathbf{L}\mathbf{L}^T = \mathbf{I}_d \end{array}$$

## Reconstruction in a transformed space

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} \|\Phi(\mathbf{x}_i) - \tilde{\mathbf{x}}_i\|_2^2 = \sum_{i=1}^{N} \left\|\Phi(\mathbf{x}_i) - \mathbf{L}^{\mathsf{T}}\mathbf{L}\Phi(\mathbf{x}_i)\right\|_2^2 \\ \text{with respect to} & \mathbf{L} \\ \text{subject to} & \mathbf{L}\mathbf{L}^{\mathsf{T}} = \mathbf{I}_d \end{array}$$

## Reconstruction in a transformed space

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} \|\Phi(\mathbf{x}_i) - \tilde{\mathbf{x}}_i\|_2^2 = \sum_{i=1}^{N} \left\|\Phi(\mathbf{x}_i) - \mathbf{L}^\mathsf{T}\mathbf{L}\Phi(\mathbf{x}_i)\right\|_2^2 \\ \text{with respect to} & \mathbf{L} \\ \text{subject to} & \mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{I}_d \end{array}$$

- Solution: $\mathbf{L} = [\tilde{\mathbf{U}}]_{1:d}^\mathsf{T}$

  where $[\tilde{\mathbf{U}}]_{1:d}$ are the $d$ columns of $\tilde{\mathbf{U}}$ from the SVD of $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1)\cdots\Phi(\mathbf{x}_N)] = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\mathsf{T}$ corresponding to the the $d$ largest singular values.

## Reconstruction in a transformed space

- In practice, we are interested in the eigenvalue decomposition of a kernel matrix $\mathbf{K}$ with elements $\mathbf{K}(\mathbf{i},\mathbf{j}) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$.
- Thus, if we want we can define a kernel function $\mathbf{K}(\mathbf{x},\mathbf{x}')$ instead of $\Phi$. This is often the case.
- Example: radial basis kernel function

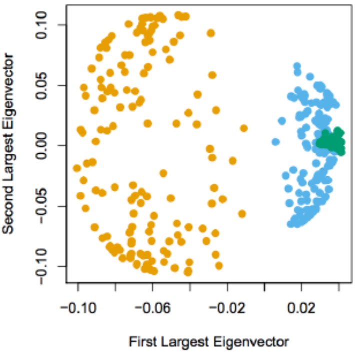
$$\mathbf{K}(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}\right)$$

$\sigma$ is a parameter to be defined.

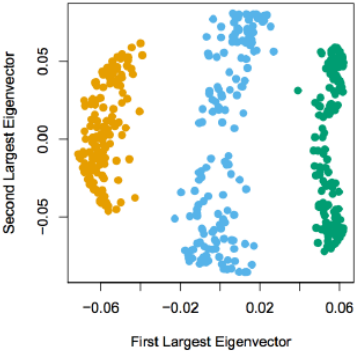
## Reconstruction in a transformed space

- Application to the previous dataset:

$\sigma = \frac{1}{2}$




$\sigma = \frac{1}{10}$



## Reconstruction in a transformed space

- This is **kernel PCA (kPCA)**.
- Great difficulty: how to choose the kernel?
- There are many methods that learn the kernel from the data.

## Reconstruction in a transformed space

- This is **kernel PCA (kPCA)**.
- Great difficulty: how to choose the kernel?
- There are many methods that learn the kernel from the data.
- Many nonlinear methods for dimensionality reduction can be seen as kPCA with a specific way of learning the kernel from data.
- Since a kernel is a measure of similarity and similarity may be meaningful only between neighbors, some methods use nearest neighbors.

# Multidimensional scaling and ISOMAP

## Preserving the distances

- We can consider that the information that should be retained after mapping should be the pairwise distances between data points.
- For points $\mathbf{x}_i$ and $\mathbf{x}_j$, we have a distance $d_{i,j}$.
- We can then find the low dimensional mapped points with the following minimization problem:

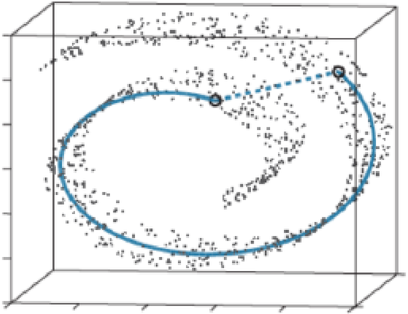
$$\begin{array}{ll} \text{minimize} & \sum_{i,j} \left( d_{i,j} - \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \right)^2 \\ \text{with respect to} & \mathbf{y}_1, \cdots, \mathbf{y}_N \\ \text{subject to} & \sum_{i=1}^{N} [\mathbf{y}_i]_k = 0, \text{ for all } k \end{array}$$

## Preserving the distances

- This is called **multidimensional scaling (MDS)**.
- If $d_{i,j} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2$, then it can be shown that $\mathbf{y}_i$ are mapped to PCA scores.
- This method can also be applied to non Euclidean distances.

## Geodesics

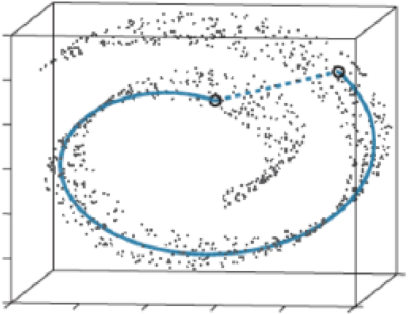
- Suppose that data are clustered on a low dimensional manifold.



- What is the relevant distance in this case?

## Geodesics

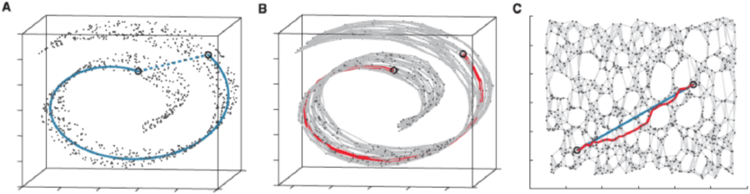
- The relevant distance may be different from the Euclidean distance. It can be the shortest distance on the manifold, a **geodesic**.



- But how to evaluate the geodesic?

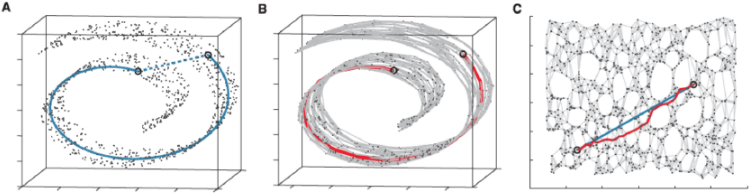
# Approximation with nearest neighbors

- Nearest neighbors may give us an approximation:
  1. Use the length of the shortest path on a neighborhood graph as distance.
  2. Apply multidimensional scaling to visualize the mapped points.



# Approximation with nearest neighbors

- Nearest neighbors may give us an approximation:
    1. Use the length of the shortest path on a neighborhood graph as distance.
    2. Apply multidimensional scaling to visualize the mapped points.

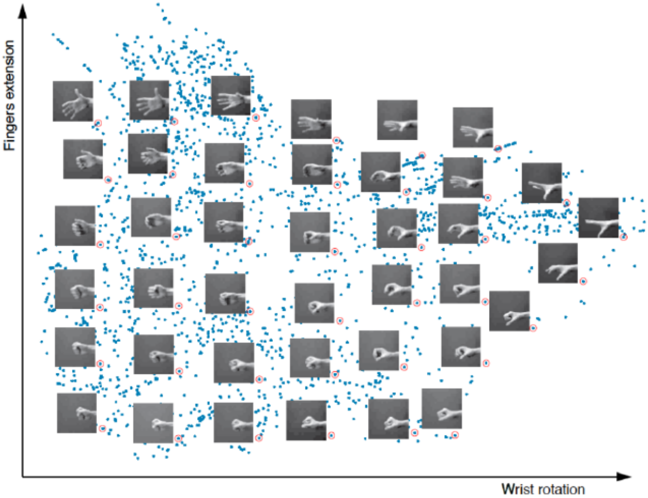



- This method is called **ISOMAP**.

# MDS and ISOMAP

## ISOMAP example: hands dataset

- Data are hands images:
  each image is a vector (size of vector = number of pixels).



# Neighborhood structure preservation

# Neighborhood structure preservation

## Precision and recall

- Visualization: visual neighborhood structure reflects actual data neighborhood structure.
- Implication on dimensionality reduction: neighborhood structure should be preserved after projection.

# Neighborhood structure preservation

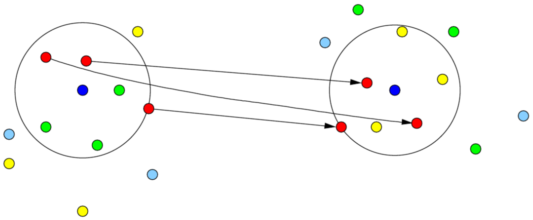
## Precision and recall

- Visualization: visual neighborhood structure reflects actual data neighborhood structure.
- Implication on dimensionality reduction: neighborhood structure should be preserved after projection.
- We can use two quality measures from information retrieval:
  1. **Precision**: neighbors on the visualization are real neighbors.
  2. **Recall**: real neighbors are neighbors on the visualization.

# Neighborhood structure preservation

## Precision and recall

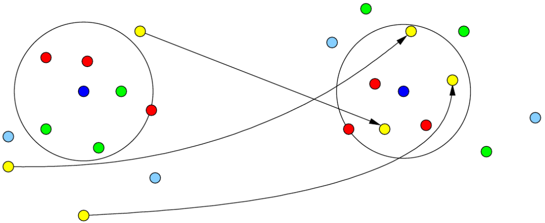
- Original data set and projection: correct projection



Source: `http://dac.lip6.fr/master/wp-content/uploads/2016/09/fdms_cours4_2016_2017.pdf`

## Precision and recall

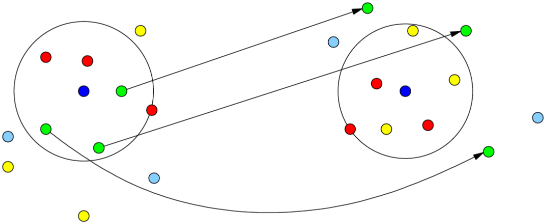
- Original data set and projection: precision violation



Source: `http://dac.lip6.fr/master/wp-content/uploads/2016/09/fdms_cours4_2016_2017.pdf`

## Precision and recall

- Original data set and projection: recall violation



Source: http://dac.lip6.fr/master/wp-content/uploads/2016/09/fdms_cours4_2016_2017.pdf

# Neighborhood structure preservation

Are your neighbors in the visualization true neighbors in the data?

- $D_k(\mathbf{x}_i)$: k-NN (k-Nearest neighbors) of $\mathbf{x}_i$ in the data.
- $P_k(\mathbf{x}_i)$: k-NN of $\mathbf{x}_i$ in the projection.
- $F_k(\mathbf{x}_i) = P_k(\mathbf{x}_i) \setminus D_k(\mathbf{x}_i)$.

- **Precision**
  - Maximal precision: $P_k(\mathbf{x}_i) \subset D_k(\mathbf{x}_i)$
  - Mean on $i$ of $1 - \dfrac{\#F_k(\mathbf{x}_i)}{\#P_k(\mathbf{x}_i)}$

# Neighborhood structure preservation

## Do you miss any neighbors in the visualization?

- $D_k(\mathbf{x}_i)$: k-NN of $\mathbf{x}_i$ in the data.
- $P_k(\mathbf{x}_i)$: k-NN of $\mathbf{x}_i$ in the projection.
- $M_k(\mathbf{x}_i) = D_k(\mathbf{x}_i) \setminus P_k(\mathbf{x}_i)$.
- **Recall**
  - Maximal recall: $D_k(\mathbf{x}_i) \subset P_k(\mathbf{x}_i)$
  - Mean on $i$ of $1 - \dfrac{\#M_k(\mathbf{x}_i)}{\#P_k(\mathbf{x}_i)}$

# Neighborhood structure preservation

## Probabilistic neighborhood

- Trying to optimize precision and recall is difficult due to the highly nonlinear behavior of k-NN.
- We define a probabilistic measure of neighborhood as follows:

  With respect to point $\mathbf{x}_i$, when we want to pick a point in its neighborhood, we will pick point $\mathbf{x}_j$ with probability $p_{j|i}$, where

$$p_{j|i} = \frac{\exp\left(-\frac{d_D(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma^2}\right)}{\sum_{j \neq i} \exp\left(-\frac{d_D(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma^2}\right)}$$

- $d_D(\mathbf{x}, \mathbf{x}')$ is a distance function and $\sigma$ is a constant parameter.

# Neighborhood structure preservation

## Probabilistic neighborhood

- The same can be defined for the mapped points:

$$q_{j|i} = \frac{\exp\left(-\frac{d_P(\mathbf{y}_i, \mathbf{y}_j)^2}{\sigma'^2}\right)}{\sum_{j \neq i} \exp\left(-\frac{d_P(\mathbf{y}_i, \mathbf{y}_j)^2}{\sigma'^2}\right)}$$

- The probability distributions $p_{j|i}$ and $q_{j|i}$ contain the neighbor structure information.
- Preserve most of the neighborhood structure
  $\Longrightarrow$ pick $\mathbf{y}_i$ such that $q_{j|i}$ is close to $p_{j|i}$

## Probabilistic neighborhood

- Preserve most of the neighborhood structure
  $\Longrightarrow$ pick $\mathbf{y}_i$ such that $q_{j|i}$ is close to $p_{j|i}$.
- A dissimilarity measure for probability distributions is the Kullback-Leibler divergence:

$$KL(p_i \| q_i) = \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

# Neighborhood structure preservation

## Probabilistic neighborhood

- Preserve most of the neighborhood structure
  $\Longrightarrow$ pick $\mathbf{y}_i$ such that $q_{j|i}$ is close to $p_{j|i}$.
- A dissimilarity measure for probability distributions is the Kullback-Leibler divergence:

$$KL(p_i \| q_i) = \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- If we want to preserve the neighborhood structure, we have to consider the sum of the KL divergences for all points.

## Stochastic neighbor embedding (SNE)

- This leads us to the following optimization problem:

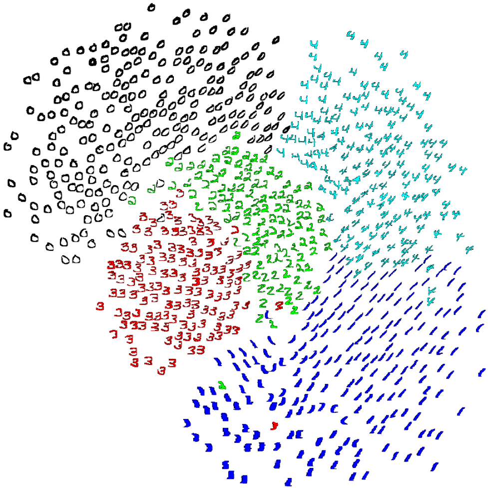
$$\begin{array}{ll} \text{minimize} & \sum_i KL(p_i \| q_i) = \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \\ \text{with respect to} & \mathbf{y}_1, \cdots, \mathbf{y}_N \end{array}$$

- This method is called **stochastic neighbor embedding (SNE)**.
- The minimization can be carried out using a gradient algorithm, since the cost function is smooth.

# Neighborhood structure preservation

## SNE example: digits dataset

- Data are digits images: each digit is a $16 \times 16$ pixels image.



Source: Hinton, G. and Roweis, S. (2002, January). Stochastic neighbor embedding. In NIPS (Vol. 15, pp. 833-840).

## Precision and recall trade-off

- It can be shown that the previous minimization problem is an approximation of the maximization of the **recall**.
- If we want to approximately maximize the **precision** we should replace $KL(p_i \| q_i)$ with $KL(q_i \| p_i)$.
- We can also maximize a trade-off of both by minimizing their linear combination.

# Neighborhood structure preservation

## Neighbor retrieval visualizer (NeRV)

- We get the following optimization problem:

$$\begin{aligned} &\text{minimize} && \sum_i \left[\lambda KL(p_i \| q_i) + (1-\lambda) KL(q_i \| p_i)\right] \\ &\text{with respect to} && \mathbf{y}_1, \cdots, \mathbf{y}_N \end{aligned}$$

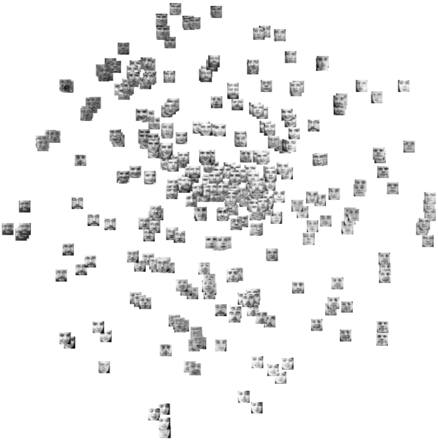
  where $\lambda \in [0, 1]$

- This method is called **neighbor retrieval visualizer (NeRV)**.
- The parameter $\lambda$ can be used to control the **recall**/**precision** trade-off.

# Neighborhood structure preservation

## NeRV example: faces dataset

- Data are faces images.



Source: *Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization*. Venna J. *et al*. Journal of Machine Learning Research. Volume 11, pages 451-490. 2010.

# Conclusions

# Conclusions

- Dimensionality reduction is a growing field within data science.
- It is a mandatory step if we want to visualize high dimensional data.
- When reducing dimensionality we should take into account what information should be retained.
- In the case of visualization this can be distances or neighborhood structure. But no clear definition of visual information exist.
- Remember: when you reduce dimensions some information is lost.