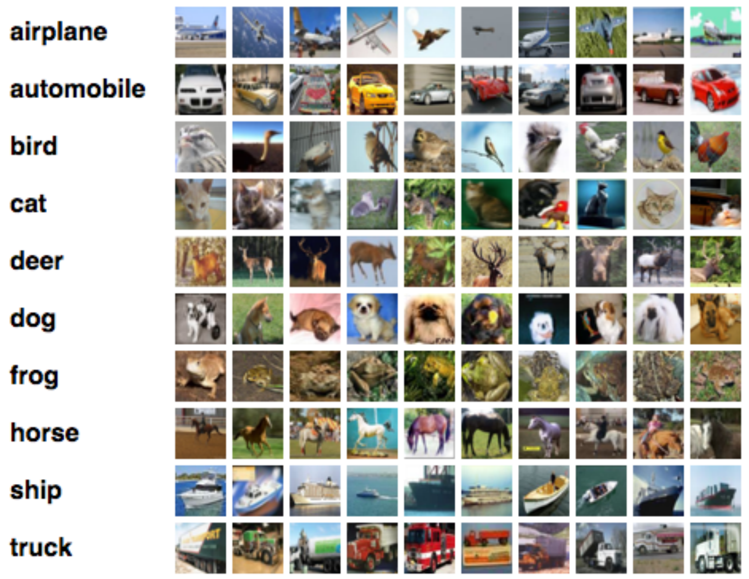
Dimensionality reduction

Milan Vojnovic

ST445 Managing and Visualizing Data

Examples of high-dimensional datasets



Images: pixel intensity vectors
O(1000) dimensions

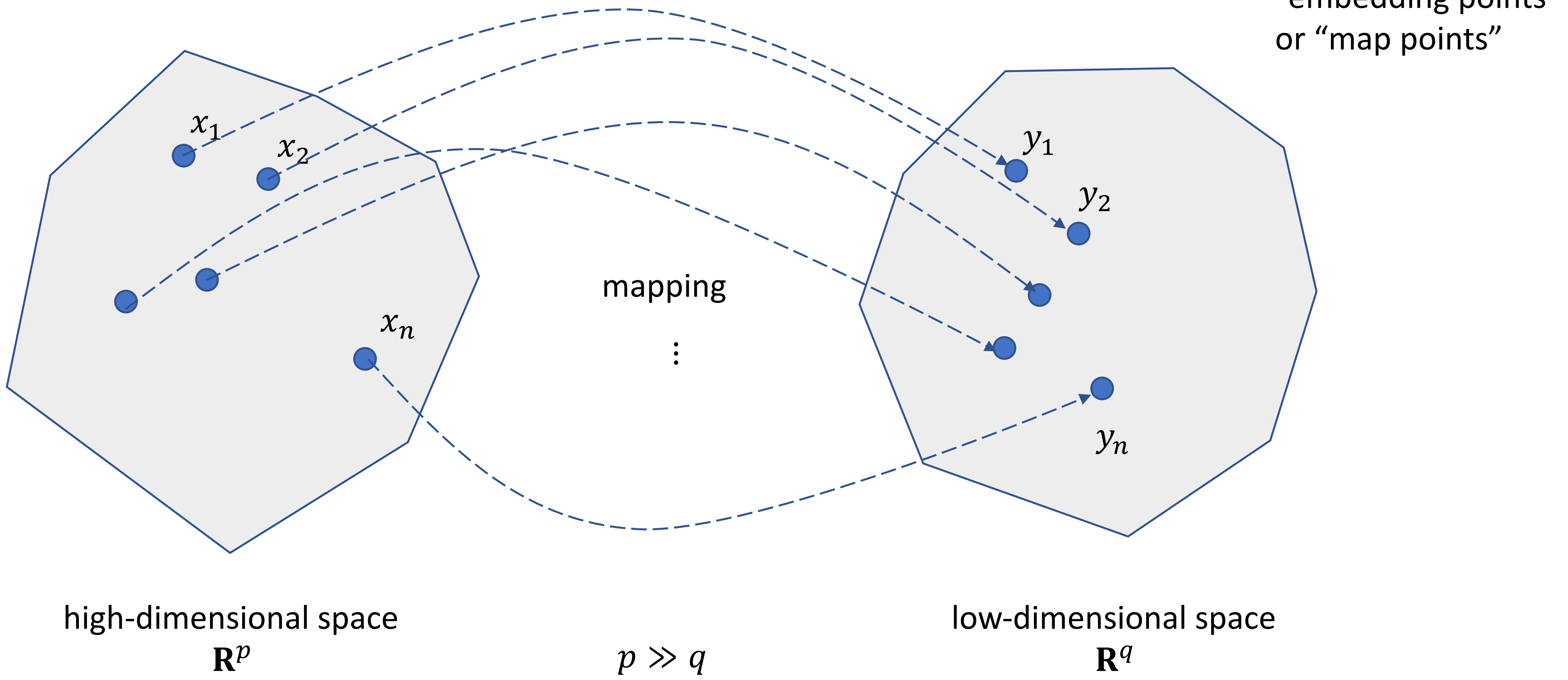
Documents: word count vectors
O(1,000) dimensions

Human brain: everyday perception
O(30,000) dimensions
O(100,000) optical nerve fibers

Dimensionality reduction (DR)

- Dimensionality reduction is about transforming a high-dimensional dataset of points to a low-dimensional representation
- It is defined by a mapping of points from a high-dimensional space $\mathbf{R}^p$ to the corresponding points in a low-dimensional space $\mathbf{R}^q$, where $q \ll p$
- The general goal of DR methods is to preserve as much of the significant structure of the high-dimensional data as possible in a low-dimensionality representation
- For data visualization purposes: $q = 1, 2, \text{or } 3$

Dimensionality reduction



Ex 1: MNIST handwritten digits

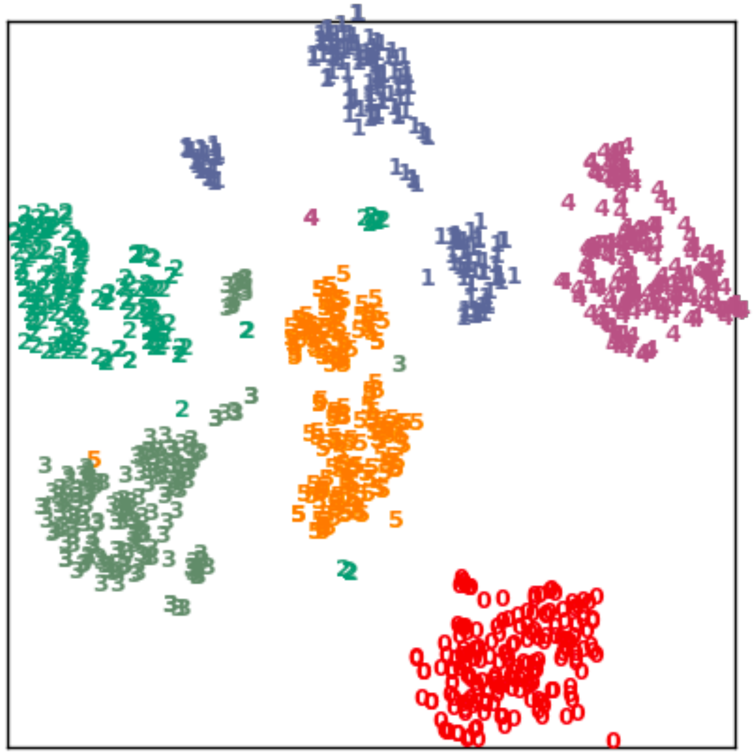
Input data points

Each point is an 8×8 bit image of a decimal digit

$p = 64$

$n = 1797$ points

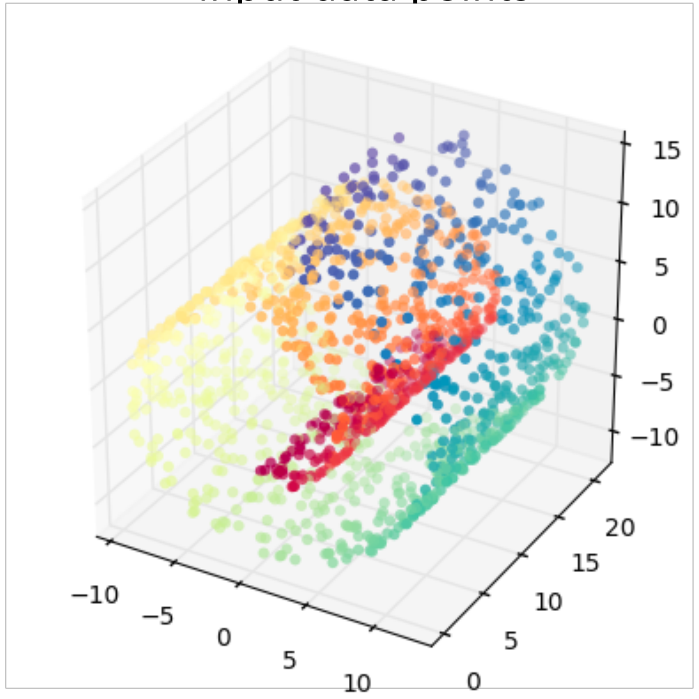
Example embedding data points



$q = 2$

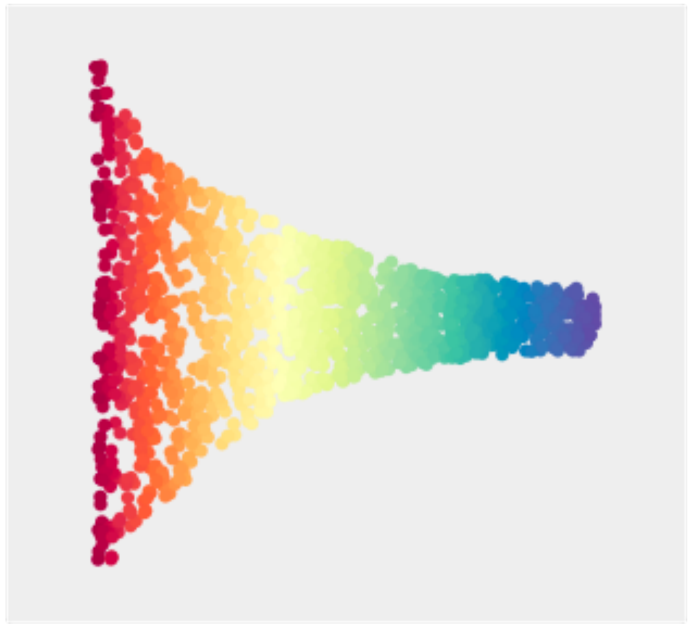
Ex 2: Swiss role (an embedded manifold)

Input data points



$p = 3$

Example embedding data points



$q = 3$

$$[-1,1]^2 \to \mathbf{R}^3 \quad \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \sqrt{2+2x_1}\cos(2\pi\sqrt{2+2x_1}) \\ \sqrt{2+2x_1}\sin(2\pi\sqrt{2+2x_1}) \\ 2x_2 \end{pmatrix}$$

Scope of this lecture

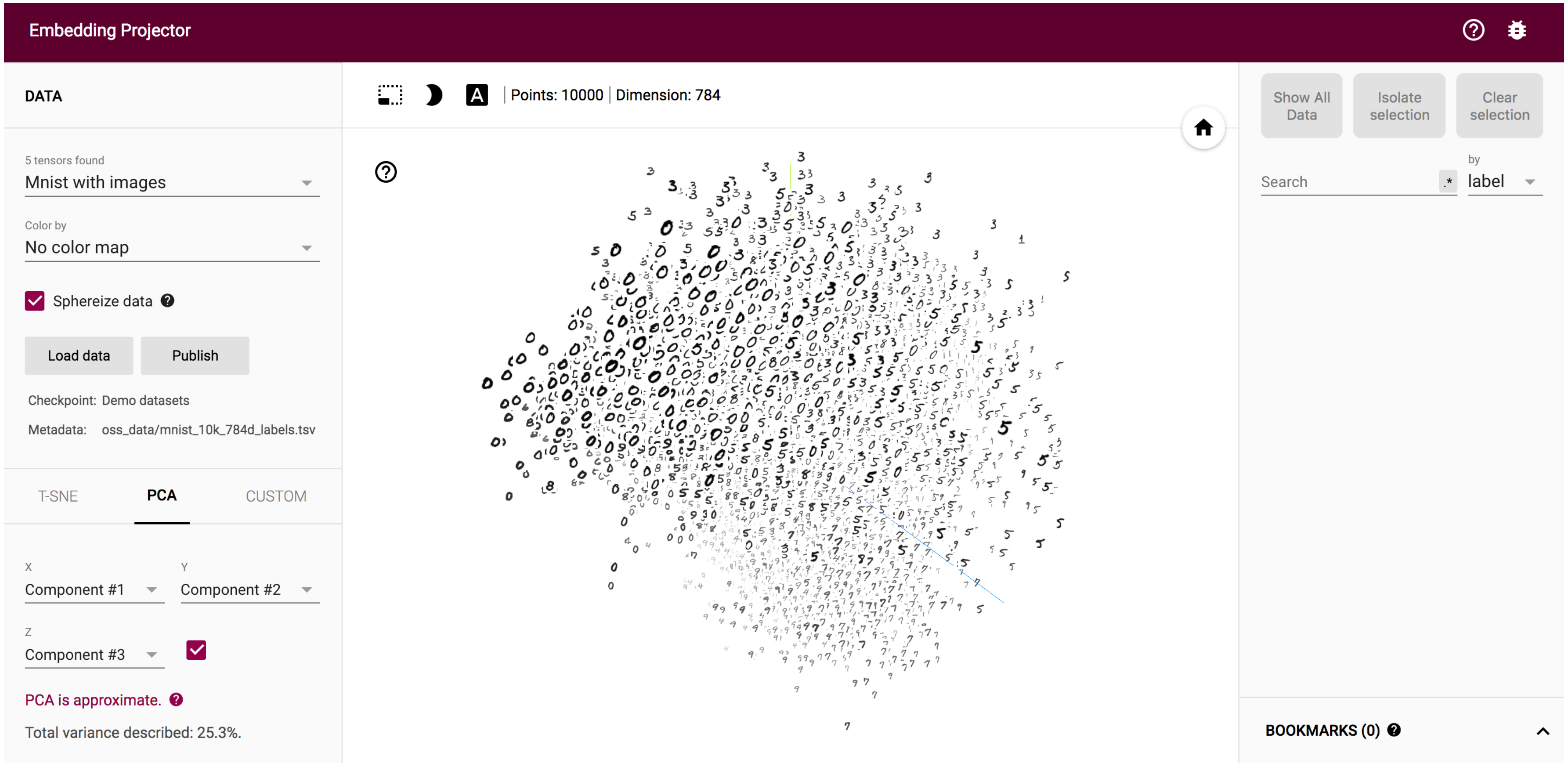
- Learn about main dimensionality reduction methods and their underlying algorithmic design principles

- Focus on commonly used methods, especially those implemented in manifold learning scikit-learn module:

  http://scikit-learn.org/stable/modules/manifold.html

  http://scikit-learn.org/stable/modules/classes.html#module-sklearn.manifold

Google Embedding Projector



Embedding Projector

DATA

5 tensors found
Mnist with images

Color by
No color map

Sphereize data

Load data
Publish

Checkpoint: Demo datasets

Metadata: oss_data/mnist_10k_784d_labels.tsv

T-SNE PCA CUSTOM

X: Component #1
Y: Component #2
Z: Component #3

PCA is approximate.

Total variance described: 25.3%.

Points: 10000 | Dimension: 784

Show All Data
Isolate selection
Clear selection

Search
by label

BOOKMARKS (0)

- http://projector.tensorflow.org

Main developments timeline

• PCA	Hotelling	1933
• MDS	Torgerson	1952
• LLE	Roweis and Saul	2000
• ISOMAP	Tenebaum, de Silva and Langford	2000
• Laplacian eigenmaps	Belkin and Niyogi	2002
• SNE	Hinton and Roweis	2002
• Hessian LLE	Donoho and Grimes	2003
• LTSA	Zhang and Zha	2004
• t-SNE	van der Maaten and Hinton	2008

Linear vs. non-linear methods

- Traditional methods such as PCA and classical MDS are linear techniques aiming to keep the low-dimensionality representation of dissimilar points far apart
- For high-dimensional data that lies on or is near a non-linear manifold, it is usually more important to keep the low-dimensionality representation of very similar data points close together, which is often hard with linear mappings

Principal Component Analysis (PCA)

- Sample covariance matrix: $S_Z = \frac{1}{n-1} Z Z^T$ for $Z \in \mathbf{R}^{m \times n}$

- Input: $X = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^{p \times n}$

- PCA decomposition problem:

  Find a linear mapping $Y = \Phi X$, where Φ is an orthogonal matrix, such that S_Y is a diagonal matrix

  The rows of Φ are called the principal components of X

PCA (cont'd)

- The principal components of X are the eigenvectors of the covariance matrix S_X

- Proof sketch:
  - Let $A = XX^T$ and $A = V\Lambda V^T$ be the eigenvalue decomposition, and $\Phi = V^T$
  - $S_Y = \frac{1}{n-1}\Phi A \Phi^T = \frac{1}{n-1}\Phi(\Phi^T \Lambda \Phi)\Phi^T = \frac{1}{n-1}\Lambda$

Computation complexity

- Computation complexity of PCA = $O(p^2 n + p^3)$

Two components:

- Covariance matrix computation: $O(p^2 n)$
  (p^2 inner products of vectors of dimension n)
- Eigenvalue decomposition: $O(p^3)$
  (p eigenvectors of a $p \times p$ matrix)

Truncated PCA

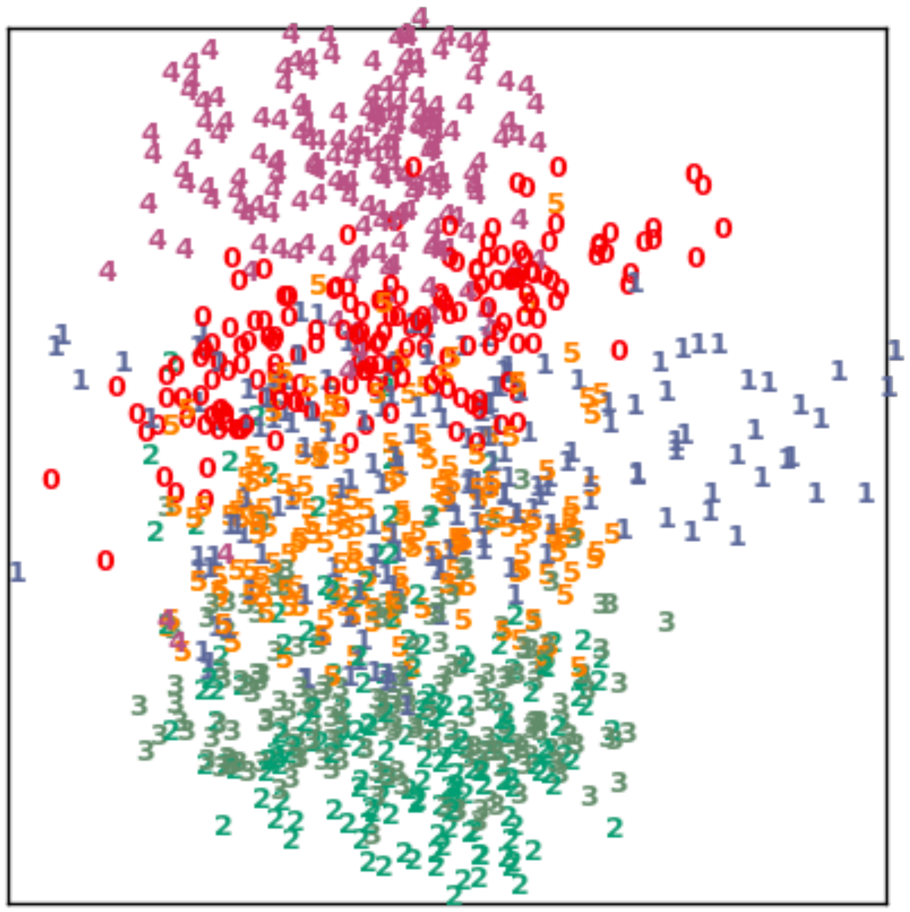
- Truncated PCA is defined by taking q eigenvectors of the sample covariance matrix S_X that correspond to the q largest eigenvalues, for $q \leq p$

- Let the eigenvalues of S_X be ordered $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ and $v_1, v_2, \dots, v_p$ be the corresponding eigenvectors

- Let $V_q = (v_1, v_2, \dots, v_q)$

- The truncated PCA mapping: $Y = V_q^T X$

PCA applied to MNIST digits dataset



Multidimensional Scaling (MDS)

- Two different types: metric and non-metric

- Metric MDS: input is a matrix of inter-point disparities and the goal is to find embedding points with inter-point Euclidean distances that match the input disparities as close as possible

- Non-metric MDS: same input but the goal is weaker: find embedding points such that any two Euclidean distances between them satisfy the same order as the corresponding input dissimilarities

Classical scaling

- Suppose that the input inter-point dissimilarities $(\delta_{i,j})$ are Euclidean distances for some points $X = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^{p \times n}$

- Goal: find embedding points in $\mathbf{R}^p$ with the inter-point Euclidean distances equal to the corresponding input inter-point dissimilarities.

- This is possible under conditions explained in the following slides

Classical scaling (cont'd)

- Let $A = \left(-\frac{1}{2}\delta_{i,j}^2\right)$ and $B = HAH$

  where $H = I - \frac{1}{n}ee^T$ and e is a vector whose all elements are equal to 1

- Note:

$$B = (XH)^T(XH)$$

  and

  B is a real symmetric positive semi-definite matrix of rank p

Classical scaling (cont'd)

- Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be the eigenvalues of B
- Let $B = V_p \Lambda_p V_p^T$ where $V_p = (v_1, v_2, \dots, v_p)$ and $\Lambda_p = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_p)$
- Let embedding points be given by: $Y = \Lambda_p^{1/2} V_p^T$
- Note $B = Y^T Y$
- The arbitrary sign of the eigenvectors of B: the invariance of the solution with respect to reflection in the origin

Classical scaling in q-dim embedding space

- A mapping to q-dimensional space can be obtained by using the q largest eigenvalues so that

$$Y = \Lambda_q^{1/2} V_q^T$$

- Referred as the principal coordinates of X in q dimensions

Cost minimization formulation

- Input: inter-point dissimilarities $(\delta_{i,j})$

  Assumed to be Euclidean distances in $\mathbf{R}^p$, i.e.

$$\delta_{i,j} = \|x_i - x_j\| \text{ for some } X \in \mathbf{R}^{p\times n}$$

- Let us denote $d_{i,j}(Y) = \|y_i - y_j\|$ for $Y \in \mathbf{R}^{q\times n}$, $1 \leq q \leq p$

- Goal: find a linear projection $Y = RX$ onto $\mathbf{R}^{q\times n}$ that minimizes the cost function

$$f(Y) = \sum_{i,j} \left(\delta_{i,j}^2 - d_{i,j}(Y)^2\right)$$

The optimality theorem

Theorem: Let

- $(\delta_{i,j})$ be a Euclidean distance matrix for points $X \in \mathbf{R}^{p\times n}$
- $1 \le q < p$
- R be a $p\times p$ orthogonal matrix
- $R = (R_1, R_2)$ where R_1 is a $p\times q$ matrix

Then, among all linear projections $Y = R_1^T X$, $f(Y)$ is minimized when X is projected onto its principal coordinates in q dimensions

Stated as Theorem 14.4.1 in Mardia, Kent and Bibby (1979)

Optimality of classical scaling (cont'd)

- Furthermore for the projection in the principal coordinates in q dimensions, the following properties hold:

  $d_{i,j}(Y) \leq \delta_{i,j}$, for all i, j

  The optimum cost value is: $2n(\lambda_{q+1} + \cdots + \lambda_p)$

- The last theorem holds for any distance matrix (not necessarily Euclidean)*

* See, e.g., Theorem 14.4.2 in Mardia, Kent and Bibby (1979)

Another metric MDS

- Input:
  - matrix of pairwise dissimilarities $(\delta_{i,j})$ for a set of n points in $\mathbf{R}^p$
  - matrix of weights $W = (w_{i,j})$

- Goal: find embedding points in $\mathbf{R}^q$ that are a solution to:

$$\begin{array}{ll} \text{minimize} & f(Y) = \sum_{i<j} w_{i,j} \left(\delta_{i,j} - d_{i,j}(Y)\right)^2 \\ \text{subject to} & Y \in \mathbf{R}^{q \times n} \end{array}$$

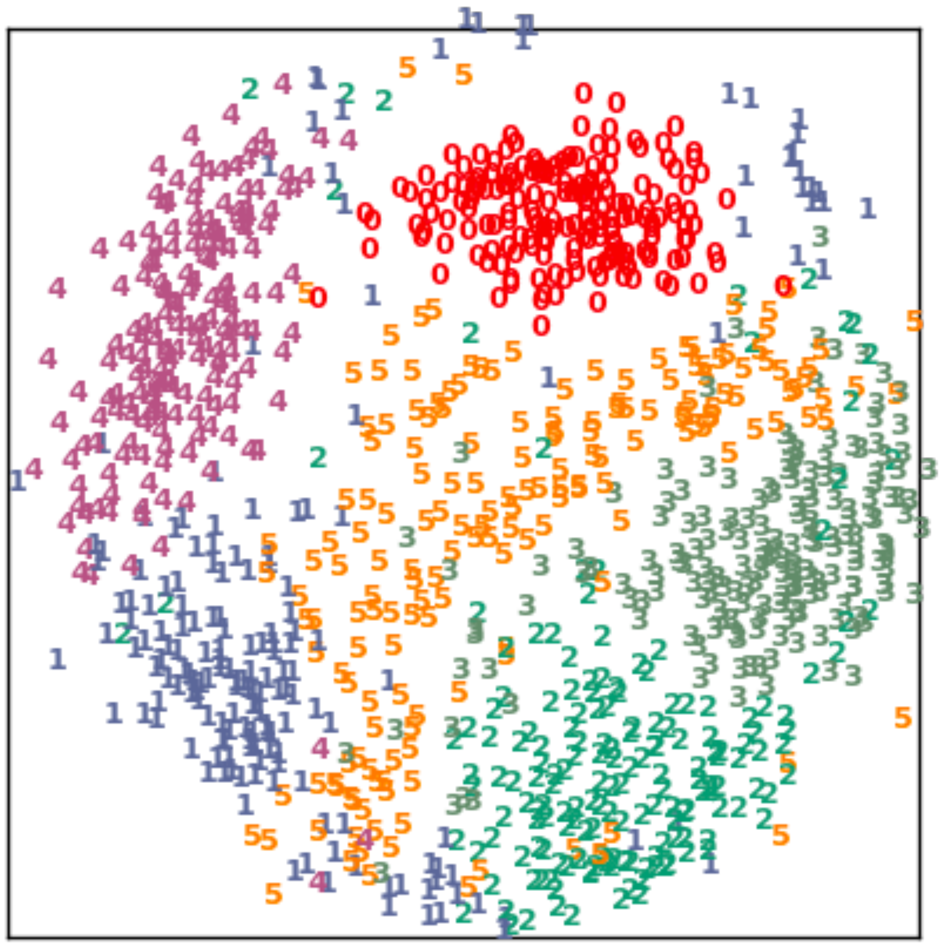
where $d_{i,j}(Y)$ denotes the Euclidean distance between points y_i and y_j in $\mathbf{R}^q$

http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html

Another metric MDS

- Admits an iterative solver referred to as SMACOF (scaling by majorizing a complicated function)
- This solver is based on general MM iterative optimization method (MM stands for minimize majorant or maximize minorant)
- More in appendix and one may check the source code in slearn.manifold.MDS

MDS applied to MNIST digits dataset



Summary for PCA and MDS

- PCA and MDS are able to discover true structure of data lying on or near a linear subspace of the high-dimensional input space

- PCA finds a low-dimensional embedding of the data points that best preserves their variance as measured in the high-dimensional input space

- Classical MDS finds an embedding that preserves the inter-point distances, equivalent to PCA when those distances are Euclidean

- Many datasets contain nonlinear structures that are invisible to PCA and MDS

ISOMAP

- A method originally proposed by Tenebaum, de Silva and Langford (2000)

- It aims to find an embedding that preserves as much as possible the geodesic distance (distance measured on the manifold) for each pair of points on the manifold

- It uses measured local metric information to learn the underlying global geometry of a dataset

- Unlike to PCA or MDS capable to discover the nonlinear degrees of freedom that underlie complex natural observations

http://scikit-learn.org/ … sklearn.manifold.Isomap.html#sklearn.manifold.Isomap

ISOMAP (cont'd)

- Points that are far apart on the underlying manifold, as measured by geodesic distance may appear deceptively close in the high-dimensional space, as measured by the Euclidean distance
  - PCA and MDS effectively see just the Euclidean structure

- ISOMAP builds on classical MDS but seeks to preserve the intrinsic geometry of the data, as captured in the geodesic manifold distances between all pairs of data points
  - For neighboring points, input-space distance provides a good approximation to geodesic distance
  - For faraway points, geodesic distance can be approximated by adding up a sequence of short hops between neighboring points

- Geodesic distance approximated by shortest path distance in a nearest neighbor graph

ISOMAP: three steps

- Step 1: Compute a nearest neighborhood graph G based on Euclidean distances between pairs of points in the input space
- Step 2: Estimate geodesic distances between all pairs of points on the manifold by shortest path distances in G
- Step 3: Compute embedding points by applying classical MDS to the matrix of graph distances

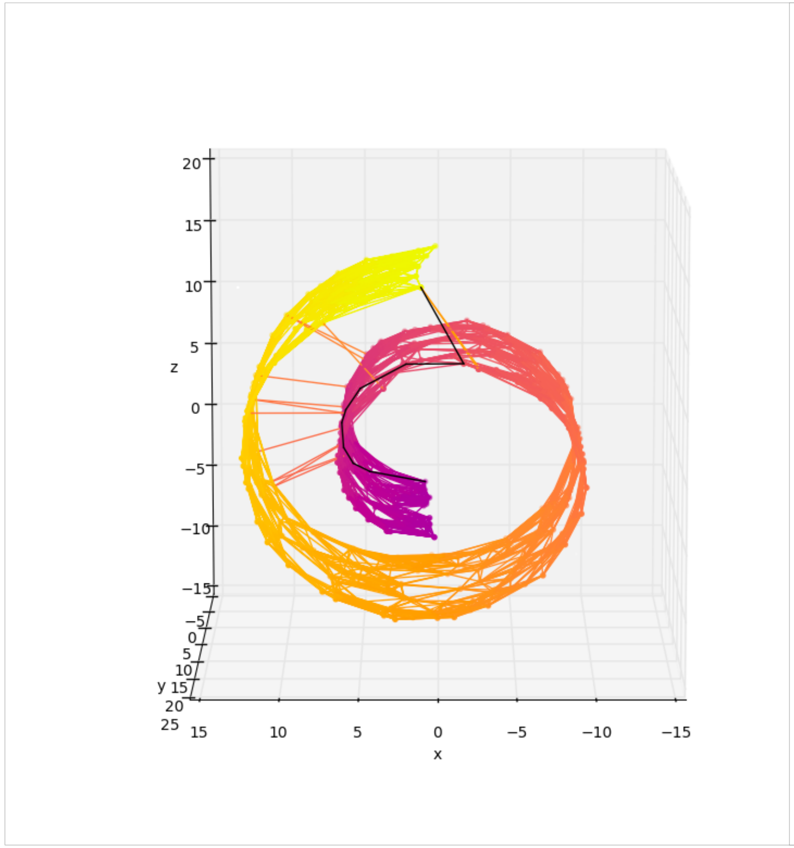
Nearest neighborhood graph

- Let $G = (V, E)$ be a graph with vertices corresponding to input data points
- Two ways to define edges:
  - $(i, j) \in E$ if, and only if, $\|x_i - x_j\| \leq \epsilon$
  - $(i, j) \in E$ if, and only if i is one of the k nearest neighbors of j
- Edge weights set to Euclidean distances of the corresponding input data points:

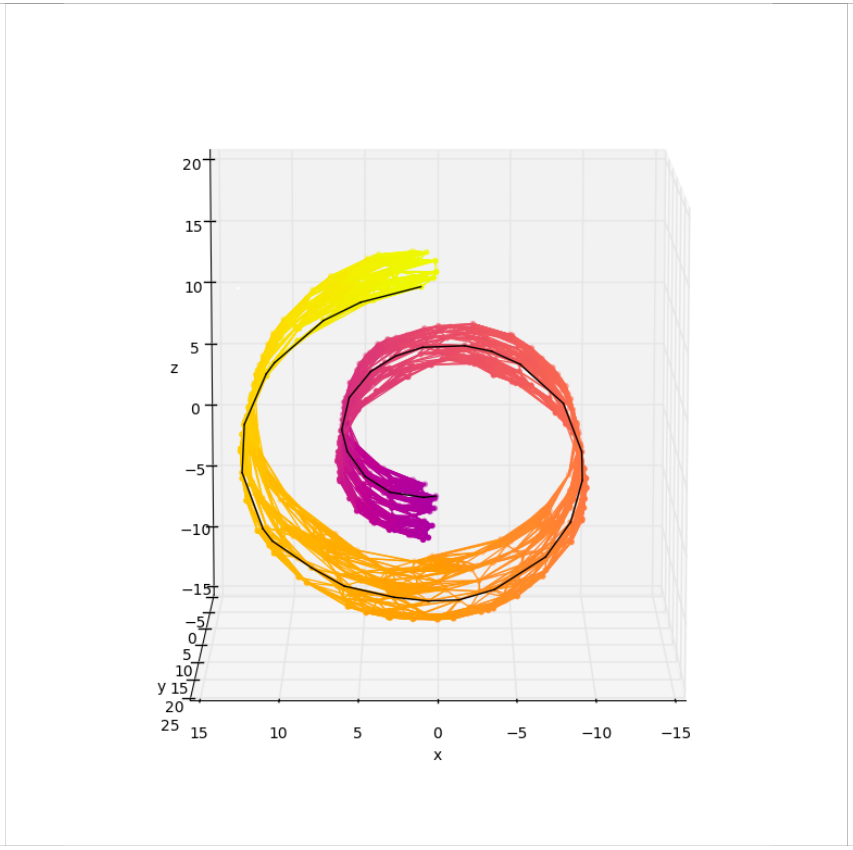
$$w_{i,j} = \|x_i - x_j\|, \text{ for } (i, j) \in E$$

Note: ϵ and k are parameters

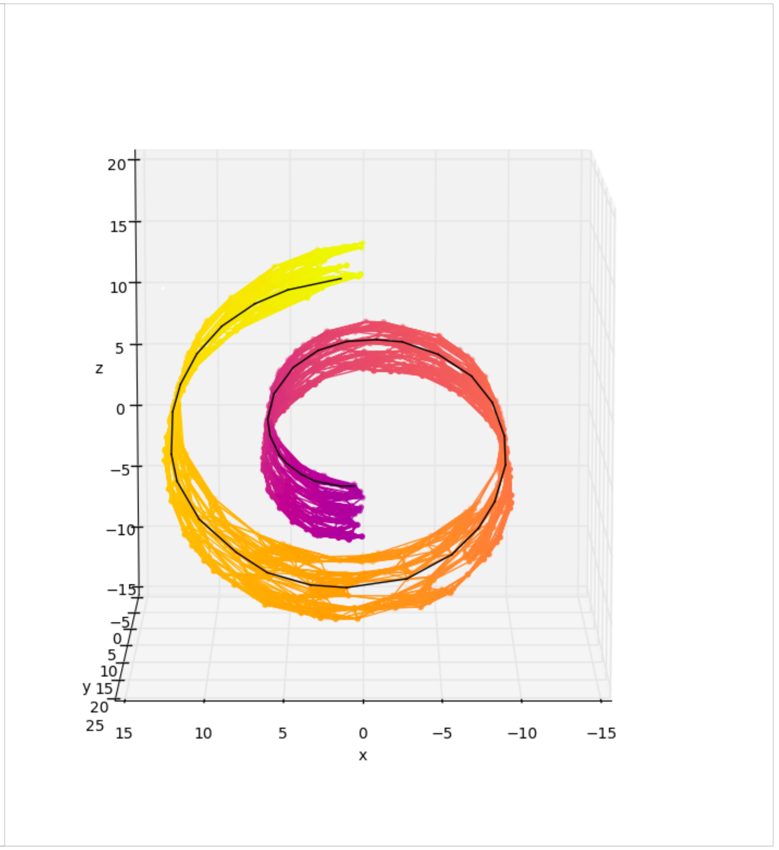
Approximate geodesic distances



n = 500



750



1000

ISOMAP guarantees

- As with PCA and MDS the true dimensionality of the data can be estimated from the decrease in the error as the dimensionality of the embedding space increases

- PCA and MDS are guaranteed to recover the true structure of a linear manifold given sufficient data

- ISOMAP is guaranteed to recover the true dimensionality and geometric structure of a larger class nonlinear manifolds: those whose intrinsic geometry is that of a convex region in a Euclidean space

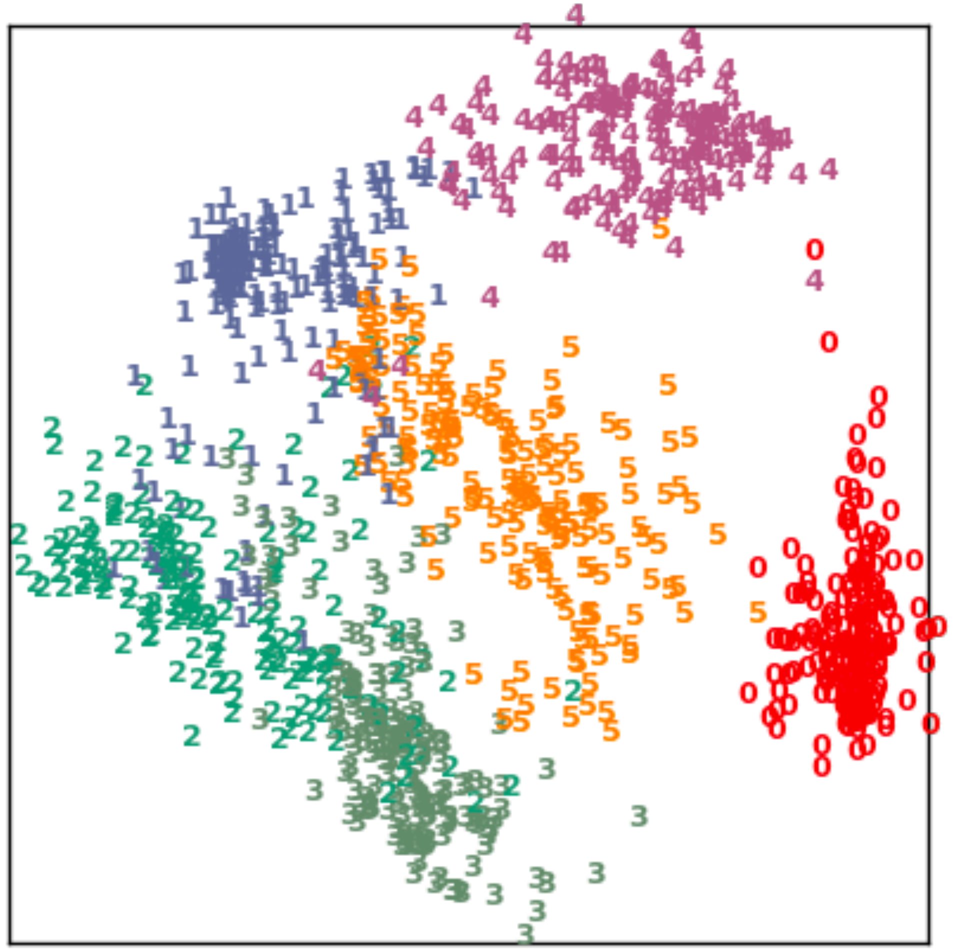
Computation complexity

- Computation complexity of ISOMAP = $O(qn^2 + (k + \log n)n^2 \ + p \log k \ n \log n)$

Two components:

- Nearest neighbor search: $O(p \log k \ n \log n)$
- Shortest-path search: $O\big((k + \log n)n^2\big)$
  (Dijkstra's algorithm)
- Partial eigenvalue decomposition: $O(qn^2)$
  (q eigenvectors of a $n \times n$ matrix)

ISOMAP applied to MNIST digits dataset



Spectral embedding

- A method proposed by Belkin and Niyogi (2003) referred also as Laplacian eigenmaps
- Goal is to find an embedding that preserves pairwise distances between points
- Three steps:
  - Step 1: Find a nearest neighborhood graph
  - Step 2: Associate edges with weights
  - Step 3: Find embedding points that minimize a cost function (defined such that similar data points are mapped near to each other)

http://scikit-learn.org/ … .manifold.SpectralEmbedding.html#sklearn.manifold.SpectralEmbedding

Special case: 1-dimensional embedding

- Let $f(y)$ be the cost function defined for $y \in \mathbf{R}^n$:

$$f(y) = \sum_{i<j} w_{i,j}\left(y_i - y_j\right)^2 = y^T L_W y$$

- Minimizing $f(y)$ subject to $y^T D_W y = 1$ is the solution of the generalized eigenvalue problem:

$$L_W y = \lambda D_W y$$

- Recall L_W and D_W denote the Laplacian and degree matrices, respectively
- $y^* = (\frac{1}{\|W\|_F}, \ldots, \frac{1}{\|W\|_F})$ is a trivial solution; eigenvector corresponding to 0 eigenvalue
- Any other eigenvector is orthogonal to y^*: add the constraint $y^T D_W e = 0$

Special case: 1-dimensional embedding (cont'd)

- Let $\widetilde{W} = D_W^{-1/2} W D_W^{-1/2}$ and $\tilde{y} = D_W^{1/2} y$

- The generalized eigenvalue problem corresponds to the standard eigenvalue problem:

$$L_{\widetilde{W}} \tilde{y} = \lambda \tilde{y}$$

- The cost minimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i<j} \widetilde{w}_{i,j} \left(y_i - y_j\right)^2 \\ \text{subject to} & y^T y = 1 \\ & y^T e = 0 \\ & y \in \mathbf{R}^n \end{array}$$

- The solution is the eigenvector that corresponds to the second smallest eigenvalue of Laplacian matrix $L_{\widetilde{W}}$

General case: q-dimensional embedding

- Let $Y = (y_1, y_2, \dots, y_n) \in \mathbf{R}^{q \times n}$

- The cost minimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i<j} \widetilde{w}_{i,j} \left\| y_i - y_j \right\|^2 = \mathbf{tr}(Y^T L_{\widetilde{W}} Y) \\ \text{subject to} & Y^T Y = I \\ & Ye = 0 \\ & Y \in \mathbf{R}^{q \times n} \end{array}$$

- The solution are the eigenvectors of the Laplacian matrix $L_{\widetilde{W}}$ corresponding to the eigenvalues $\lambda_2, \lambda_3, \dots, \lambda_{q+1}$

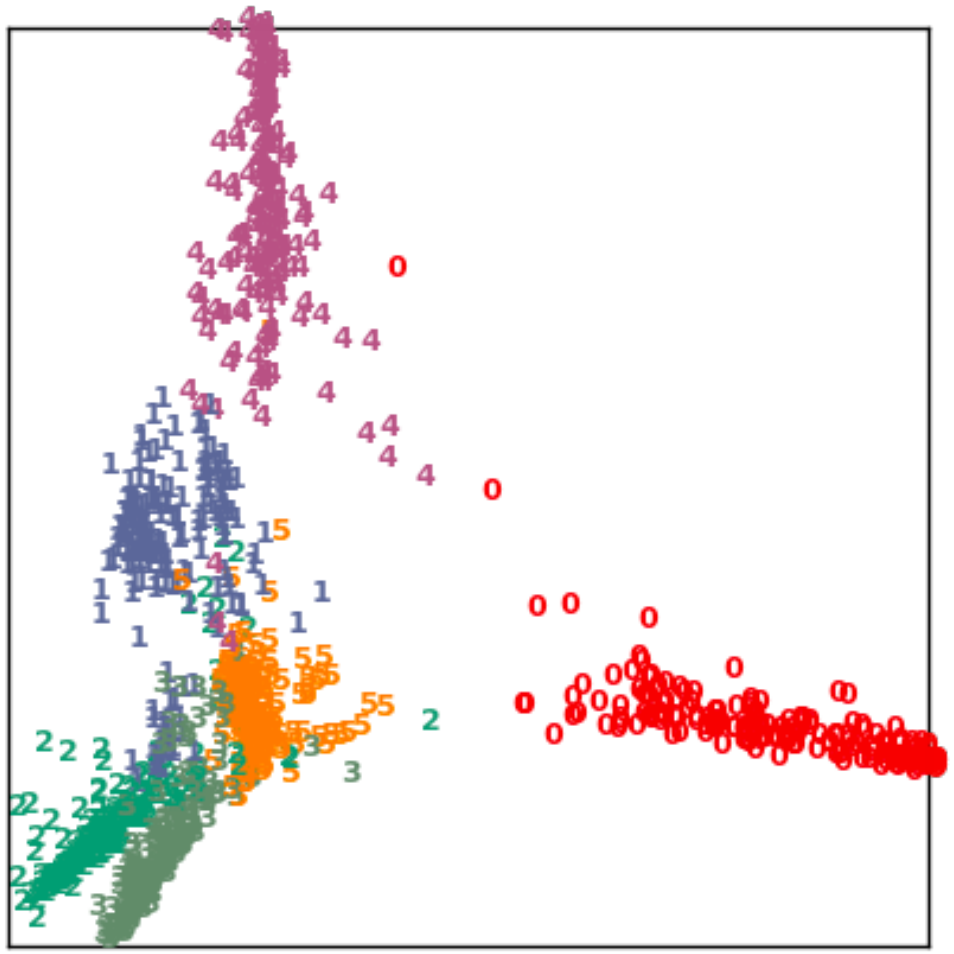
Computation complexity

- Computation complexity = $O(qn^2 + pnk^3 + p \log k \, n \log n)$

Components:

- Weighted graph construction: $O(p \log k \, n \log n)$
- Laplacian computation: $O(pnk^3)$
- Eigenvalue decomposition: $O(qn^2)$ (q eigenvectors of a $n \times n$ matrix)

Spectral embedding applied to MNIST digits



Locally Linear Embedding (LLE)

- A method originally proposed by Roweis and Saul (2000)
- The method consists of three steps:
  - Step 1: Compute a nearest neighborhood graph
  - Step 2: Compute local weights
  - Step 3: Compute embedding points

http://scikit-learn.org/ … .locally_linear_embedding.html#sklearn.manifold.locally_linear_embedding

Computing local weights

- Find a matrix $W \in \mathbf{R}^{n\times n}$ that is a solution to the following optimization problem:

minimize $\sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{i,j} x_j \right\|^2$

subject to $\sum_{j=1}^{n} w_{i,j} = 1$ for $i = 1,2,\ldots,n$
$w_{i,j} = 0$ for $(i,j) \notin E$
$W \in \mathbf{R}^{n\times n}$

- Intuition: each data point and its neighbors approximately lie on a locally linear patch of a manifold

Computing local weights (cont'd)

- The problem is separable: for each $i = 1,2,\ldots,n$ solve

$$\begin{array}{ll} \text{minimize} & \left\| x_i - \sum_{j=1}^{n} w_{i,j} x_j \right\|^2 \\ \text{subject to} & \sum_{j=1}^{n} w_{i,j} = 1 \text{ for } i = 1,2,\ldots,n \\ & w_{i,j} = 0 \text{ for } (i,j) \notin E \\ & \left(w_{i,1}, w_{i,2}, \ldots, w_{i,n}\right)^T \in \mathbf{R}^n \end{array}$$

- Finding optimal weights amounts to solving least square problems
- The optimal weights are invariant to rotation, rescaling and translation of each data point and its neighbors

Computing embedding points

- The cost minimization problem:

$$\begin{aligned} \text{minimize} \quad & f(Y) = \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{i,j} y_j \right\|^2 \\ \text{subject to} \quad & Y^T e = 0 \\ & \tfrac{1}{n} Y^T Y = I \\ & Y \in \mathbf{R}^{n \times q} \end{aligned}$$

- Note: $f(Y) = \mathbf{tr}(Y^T L_W^T L_W Y)$
- Solution corresponds to eigenvectors of the matrix $L_W^T L_W$ corresponding to q eigenvalues $\mu_2, \mu_3, \dots, \mu_{q+1}$
- Same as eigenvectors of L_W corresponding to eigenvalues $\lambda_2, \lambda_3, \dots, \lambda_{q+1}$

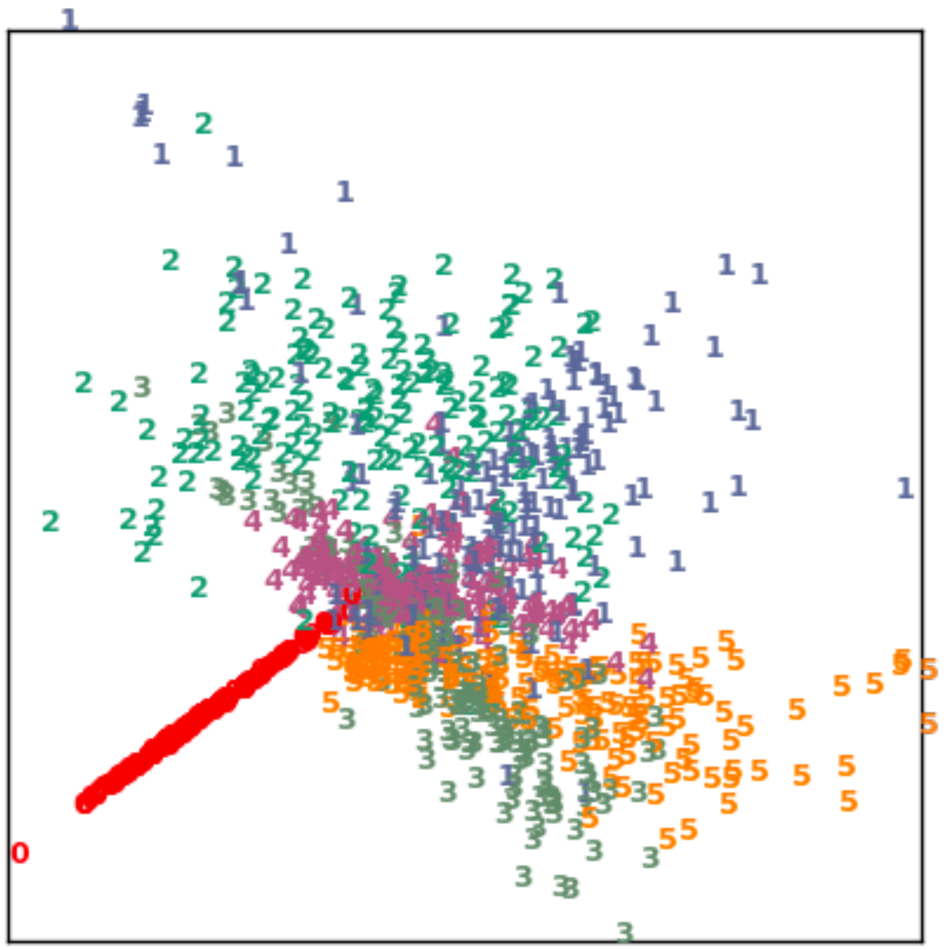
Computation complexity

- Computation complexity = $O(qn^2 + pnk^3 + p \log k \, n \log n)$

Components:

- Nearest neighbour search: $O(p \log k \, n \log n)$
- Weighted matrix construction: $O(pnk^3)$
  (n linear systems with k equations and k unknowns)
- Partial eigenvalue decomposition: $O(qn^2)$

LLE applied to MNIST digits dataset



Hessian LLE

- A method proposed by Donoho and Grimes (2003) referred to as Hessian eigenmaps
- Based on estimating a quadratic form (Hessian)
- Allows for a wider class of manifolds than ISOMAP (not necessarily convex)

http://scikit-learn.org/ … .LocallyLinearEmbedding.html#sklearn.manifold.LocallyLinearEmbedding

Historical remarks

- David Donoho
- Professor of statistics at Stanford University
- Worked on the development of effective methods for the construction of low-dimensional representations for high-dimensional data problems, development of wavelets for denoising and compressed sensing

Local Tangent Space Alignment (LTSA)

- A method proposed by Zhang and Zha (2004)

- Three steps:
  - Step 1: Compute a nearest neighbor graph
  - Step 2: Estimate local linear manifolds for each input data point
  - Step 3: Compute global embedding points

http://scikit-learn.org/ … .LocallyLinearEmbedding.html#sklearn.manifold.LocallyLinearEmbedding

Estimating local linear manifolds

- Consider an arbitrary input data point x_i
- Let $X_i = (x_{i_1}, x_{i_2}, \dots, x_{i_k})$ be the neighbor points of x_i
- We seek to find a local linear mapping $m(\tilde{Y}_i) = \tilde{A}_i \tilde{Y}_i + \tilde{b}_i e^T$ where $\tilde{A}_i \in \mathbf{R}^{p \times q}$ and $\tilde{b}_i \in \mathbf{R}^p$ are a solution to

$$\begin{array}{ll} \text{minimize} & \left\| X_i - \left(\tilde{A}_i \tilde{Y}_i + \tilde{b}_i e^T \right) \right\|^2 \\ \text{subject to} & \tilde{A}_i \in \mathbf{R}^{p \times q}, \tilde{b}_i \in \mathbf{R}^p, \tilde{Y}_i \in \mathbf{R}^{q \times k} \\ & \tilde{A}_i^T \tilde{A}_i = I \end{array}$$

Estimating local linear manifolds (cont'd)

- The optimal solution satisfies $b_i = \frac{1}{k} X_i e$

- Hence, we can write the objective function as

$$\left\| \tilde{X}_i - \tilde{A}_i \tilde{Y}_i \right\|^2$$

  where $\tilde{X}_i = X_i (I - \frac{1}{k} e e^T)$

- Let the columns of $\tilde{A}_i$ be the q left singular vectors of $\tilde{X}_i$
- Since these singular vectors are orthonormal the constraint $\tilde{A}_i^T \tilde{A}_i = I$ holds
- The objective of zero value is obtained by: $\tilde{Y}_i = \tilde{A}_i^T \tilde{X}_i$

Computing global embedding points

- We seek to find global coordinates $Y_i \in \mathbf{R}^{q\times k}$ corresponding to the local coordinates $\tilde{Y}_i$ that minimize the following objective function:

$$\sum_{i=1}^{n} \left\| Y_i - (A_i \tilde{Y}_i + b_i e^T) \right\|^2$$

- It is optimal that

$$b_i = \frac{1}{k} Y_i e \quad \text{and} \quad A_i = Y_i \left(I - \tfrac{1}{k} e e^T \right) \tilde{Y}_i^{+}$$

Computing global embedding points (cont'd)

- The objective function can be written as

$$\sum_{i=1}^{n} \left\| Y_i \left(I - \frac{1}{k} e e^T\right)\left(I - \tilde{Y}_i^{+} \tilde{Y}_i\right) \right\|^2$$

- This objective function can be written as $\|YSW\|_F^2$ where

$Y = (y_1, y_2, \ldots, y_n)$

$S_i \in \{0,1\}^{n \times k}$ is such that $Y_i = Y S_i$

$S = (S_1, S_2, \ldots, S_n)$ and $W = \mathrm{diag}(W_1, W_2, \ldots, W_n)$

$W_i = \left(I - \frac{1}{k} e e^T\right)\left(I - \tilde{Y}_i^{+} \tilde{Y}_i\right)$

Computing global embedding points (cont'd)

- Solve

$$\begin{array}{ll} \text{minimize} & \|YSW\|_F^2 \\ \text{subject to} & YY^T = I \\ & Y \in \mathbf{R}^{q \times n} \end{array}$$

- An optimal solution is $Y = (v_2, v_3, \ldots, v_{q+1})$ where $v_2, v_3, \ldots, v_{q+1}$ are the eigenvectors of $SW(SW)^T$ corresponding to eigenvalues $\lambda_2, \lambda_3, \ldots, \lambda_{q+1}$

Side remark

- The following identity holds

$$W_i = I - G_i G_i^T$$

where $G_i = \left(\frac{1}{k}e, H_i\right)$ and H_i is a $k \times q$ matrix whose columns are the right singular vectors corresponding to the first q singular values of $\tilde{X}_i$

- This identity is used in the source code implementation

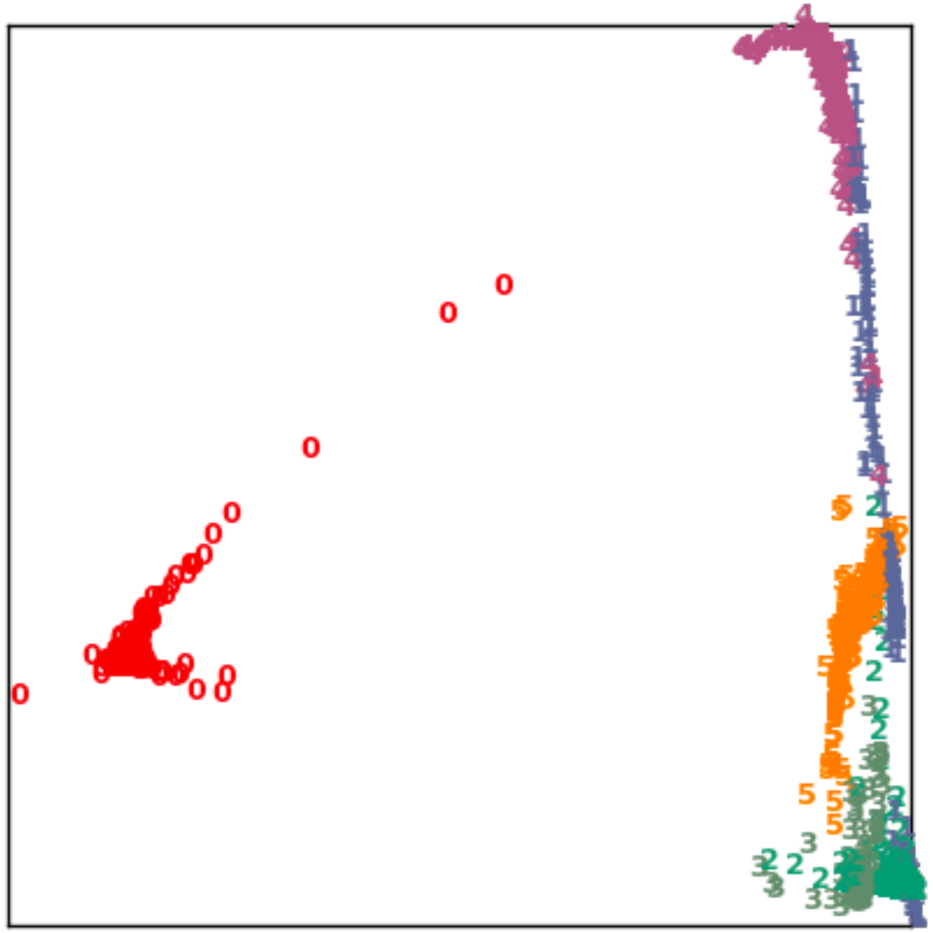
Computation complexity

- Computation complexity = $O(qn^2 + pnk^3 + qk^2 + p \log k \, n \log n)$

Components:

- Nearest neighbour search: $O(p \log k \, n \log n)$
- Weighted matrix construction: $O(pnk^3 + qk^2)$
- Partial eigenvalue decomposition: $O(qn^2)$

LTSA applied to MNIST digits dataset



Stochastic Neighbor Embedding (SNE)

- A method proposed by Hinton and Roweis (2002)
- A modified version (t-SNE) proposed by van der Maaten and Hinton (2008)
- Both are based on a model under which points in the input space and the embedding space are sampled from specified parametric distributions and the embedding points are defined as points that minimize a divergence function between these two distributions

http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn.manifold.TSNE

Historical remarks

- Geoffrey Hinton
- British-born Canadian cognitive psychologist and computer scientist, most noted for his work on artificial neural networks
- One of the first researchers who demonstrated the use of backpropagation algorithm for training multilayer neural networks, playing an important role in deep learning

SNE model

- The Euclidean distances between input data points are transformed to

$$p_{i,j}(x;\sigma) = \frac{1}{Z_i}\exp\left(-\frac{\left\|x_i - x_j\right\|^2}{\sigma_i^2}\right)$$

  where Z_i is the normalization constant such that $\sum_j p_{i,j}(x;\sigma) = 1$
  Interpretation: the probability that x_i picks x_j as a neighbor, if this picking is in proportion to the Gaussian density function with mean x_i and variance σ_i^2

- Similarly, for the embedding points:

$$q_{i,j}(y) = \frac{1}{Z_i(y)}\exp\left(-\left\|y_i - y_j\right\|^2\right)$$

  where $Z_i(y)$ is such that $\sum_j q_{i,j}(y) = 1$

SNE model fitting

- Find embedding points y that minimize the Kullback-Leibler (KL) divergence between distributions P and Q:

$$f(y;\sigma) = \mathrm{KL}(P||Q) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}\,(x;\sigma)\log\left(\frac{p_{i,j}(x;\sigma)}{q_{i,j}(y)}\right)$$

- Note that different types of error in distances between embedding points are not weighted equally:
    - Large cost for widely separated map points to represent nearby input data points
    - Small cost for nearby map points to represent widely separated input data points
- SNE cost function focuses on preserving local structure of the data

Setting the model parameters

- Parameters $\sigma_1, \sigma_2, \ldots, \sigma_n$ are set such that perplexities are equal to a specified constant, where perplexities are defined by

$$\mathrm{per}(P_i) := 2^{H(P_i)}$$

where $H(P)$ is the entropy of distribution P

$$H(P) = -\sum_i p_i \log_2(p_i)$$

- A “guess” about the number of close neighbours of an input data point

- The original proposal suggested choosing a value between 5 and 50

Cost function minimization

- Cost function f minimization by using the gradient descent method:

$$y^{(t+1)} = y^{(t)} - \eta_t \nabla f\left(y^{(t)}\right)$$

where $\nabla f(y) = (\nabla_{y_1} f(y), \ldots, \nabla_{y_n} f(y))$ and

$$\nabla_{y_i} f(y) = 2 \sum_{j=1}^{n} (p_{j,i} - q_{j,i}(y) + p_{i,j} - q_{i,j}(y))(y_i - y_j)$$

- Initial point set to a sample from a product-form Gaussian distribution with zero mean and small variance

Cost function minimization (con't)

- Gradient descent augmented with a momentum term:

$$y^{(t+1)} = y^{(t)} - \eta_t \nabla f\left(y^{(t)}\right) + \alpha_t\left(y^{(t)} - y^{(t-1)}\right)$$

  where α_t is the momentum at iteration step t

- Momentum term used to speed up the optimization and avoid poor local minima

- To avoid poor local minima, also a Gaussian noise is added to each embedding point at each iteration with a gradually decreasing variance with the number of iterations

Cost function minimization (cont’d)

- The cost function is difficult to optimize
- Non-convex optimization problem
- Poor local minima

Crowding problem

- There is a volume difference between the input space and the map space
  - Modelling moderate distances accurately in the map space requires most of them to be placed far away in the map space
  - The resulting attractive force between two dissimilar points is small, however, these small attractive forces add up resulting in bringing map points together

- A way to mitigate this is to use a heavy-tail distribution in the map space
  - Allows to represent moderate distances in the input space with large distances in the map space
  - Reduces the unwanted attractive forces between map points that represent moderately dissimilar input data points

t-Distributed Stochastic Neighbor Embedding

- A variant of SNE with the following two main differences:
  - Use of a symmetric cost function
  - Use of a Student-t distribution instead of a Gaussian distribution for pairwise distances between map points
- Student-t distribution has a heavy tail (power-law)
  - Alleviates the crowding problem and simplifies the optimization

Symmetrizing the cost function

- Symmetric inter-point probabilities defined as: $\tilde{p}_{i,j} = \frac{1}{n}\frac{1}{2}(p_{i,j} + p_{j,i})$

- Cost function is redefined to: $f(y;\sigma) = KL(\tilde{P}(x;\sigma)||Q(y))$

- Symmetric cost function alleviates the problem of outliers: suppose that input data point is far apart from all other input data points, which results in this point having little effect on the cost function and thus the position of this point is not well defined in the map space

- Symmetric cost function ensures that $\sum_j \tilde{p}_{i,j} \geq \frac{1}{2n}$ for all i,thus each point has a contribution to the cost function

Using a t-Student distribution

- Define $q_{i,j}(y) = \frac{1}{Z_i(y)} \frac{1}{1+\|y_i - y_j\|^2}$

- Heavy-tailed distribution
  - Allows moderately distant points to be modeled as far apart in the map space
  - As a result, eliminates the unwanted attractive forces (crowding problem)

- Scales as $1/d^2$ for two points at large distance d

Gradient descent

- Gradient descent update:

$$\nabla_{y_i} f(y) = 4 \sum_{j \neq i} (\tilde{p}_{i,j} - q_{i,j}(y))(y_i - y_j) \frac{1}{1+\|y_i - y_j\|^2}$$

- Strongly repels dissimilar input data points that have a small distance in the map space

- Emphasis on
  - Modelling of dissimilar data points by means of large pair-wise distances
  - Modelling of similar data points by means of small pair-wise distances

Computation complexity

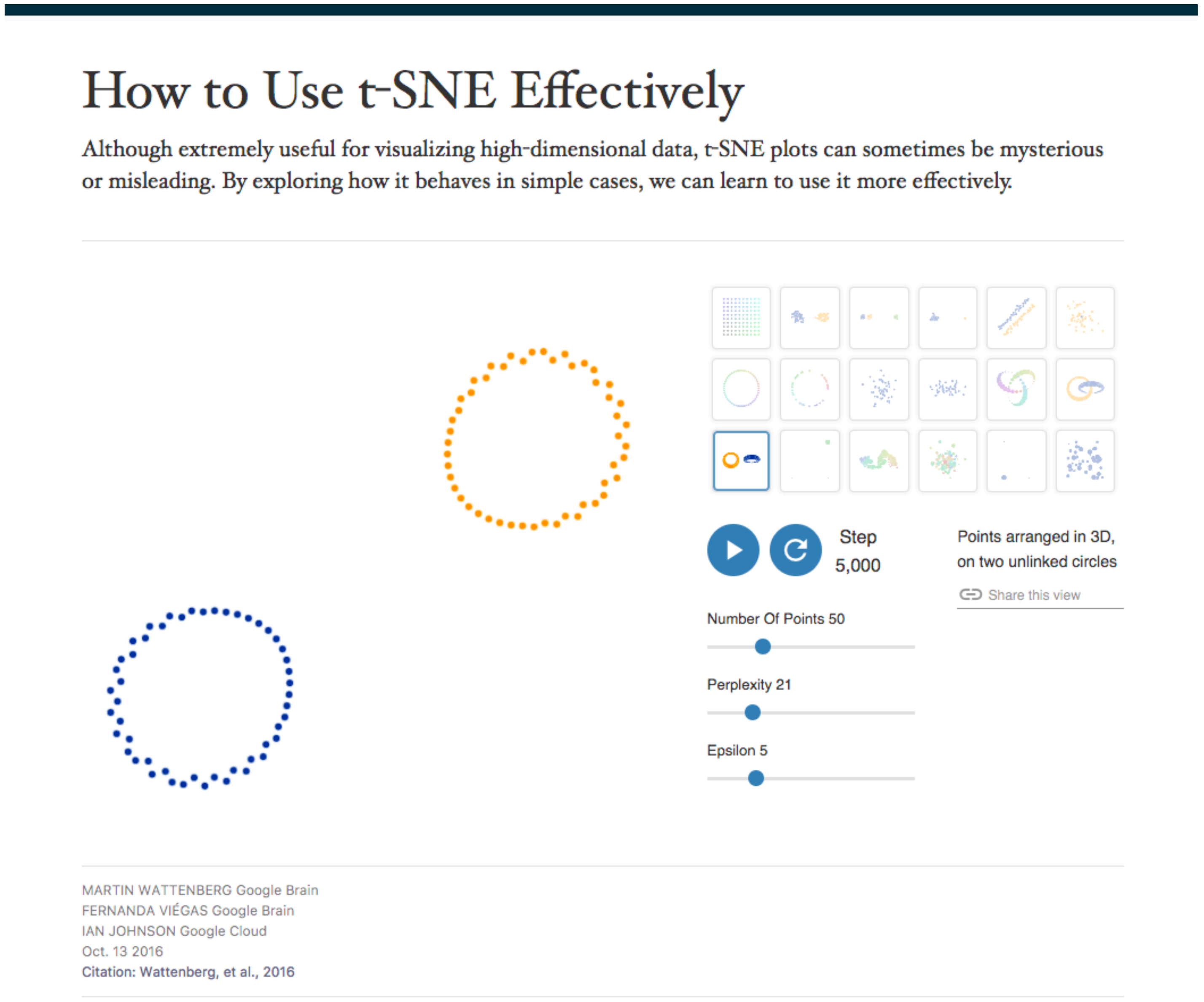
- Computation complexity = $\Omega(n^2)$

- Gradient update requires computing $\Theta(n^2)$ elements

Summary of issues of t-SNE

- Sensitivity to the choice of the perplexity parameter
- Sensitivity to the number of iteration steps
- Different outputs on successive runs (randomization)
- Additional hyper-parameters related to the optimization problem
- Cluster sizes in the map space are not guaranteed to reflect original cluster sizes
- Random noise may appear as non random

t-SNE



How to Use t-SNE Effectively

Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.

Points arranged in 3D, on two unlinked circles

Number Of Points 50

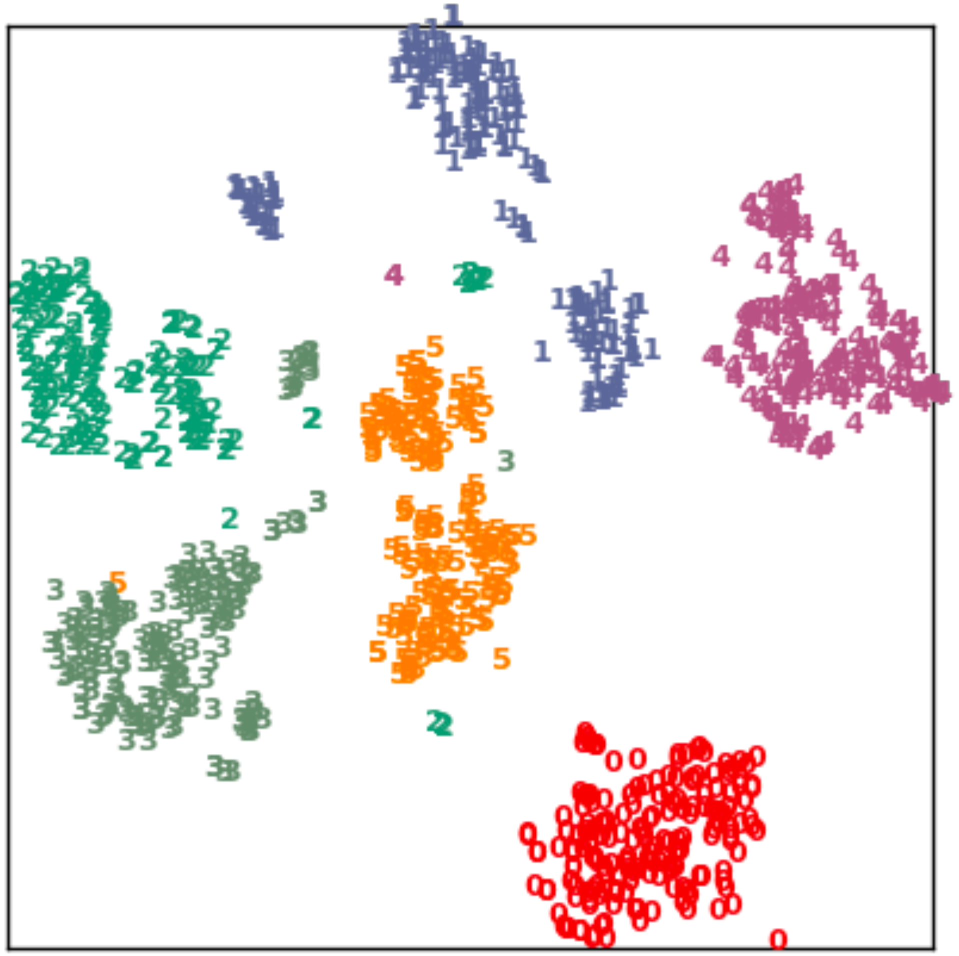
Perplexity 21

Epsilon 5

MARTIN WATTENBERG Google Brain
FERNANDA VIÉGAS Google Brain
IAN JOHNSON Google Cloud
Oct. 13 2016
Citation: Wattenberg, et al., 2016

- https://distill.pub/2016/misread-tsne

t-SNE applied to MNIST digits dataset



Summary

- Several different dimensionality reduction methods are commonly used
- There is no universally best dimensionality-reduction method
- Some common principles:
  - Using a nearest neighbor graph
  - Modelling local non linearity
  - Finding embedding points by minimizing a cost function

References

- M. Belkin and P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, NIPS, 2002
- D. L. Donoho and C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensionality data, PNAS, Vol 100, No 10, pp. 5591-5596, 2003
- G. E. Hinton and S. T. Roweis, Stochastic neighbor embedding, NIPS 2002
- H. Hotelling, Anaysis of a complex of statistical variables into principal components, Journal of educational psychology, Vol 24, pp. 417-41, 1933
- S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science, Vol 290, pp. 2323-26, 2000
- J. B. Tenenbaum, V. de Silva and J. C. Langford, A global framework for nonlinear dimensionality reduction, Science, Vol 290, pp. 2319-22, 2000
- W. S. Torgerson, Multidimensional scaling I: Theory and method, Psychometrika, Vol 17, pp. 401-19, 1952
- MDS http://forrest.psych.unc.edu/teaching/p208a/mds/mds.html

References (cont'd)

- L. van der Maaten and G. Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research, Vol 9, pp 2579-2605, 2008
- L. van der Maaten, E. Postma and J. van den Herik, Dimensionality Reduction: A Comparative Review, Technical Report, Tilburg University 2009
- C. K. I. Williams, On a Connection between Kernel PCA and Metric Multidimensional Scaling, NIPS 2001
- Z. Zhang and H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, Journal of Shanghai University, 8:406, 2004

Books

- I. Borg and P. J. F. Groenen, Modern Multidimensional Scaling, 2nd edition, 2005
- T. F. Cox and M. A. A. Cox, Multidimensional Scaling, 2nd edition, Chapman & Hall/CRC, 2001
- J. A. Lee and M. Verleysen, Nonlinear Dimensionality Reduction, Springer, 2007
- K. V. Mardia, J. T. Kent and J. M. Bibby, Multivariate Analysis, Academic Press, 1979

Appendix

The optimality of classical scaling: proof sketch

- Since R is an orthogonal matrix, the distance between the columns of X are the same as the distances between the columns of $R^T X$

- The distances can be written as

$$\delta_{i,j}^2 = \sum_{k=1}^{p} \left(x_{i,k} - x_{j,k}\right)^2 = \sum_{k=1}^{p} \left(x_i^T e_k - x_j^T e_k\right)^2$$

$$d_{i,j}^2 = \sum_{k=1}^{q} \left(x_{i,k} - x_{j,k}\right)^2 = \sum_{k=1}^{q} \left(x_i^T e_k - x_j^T e_k\right)^2$$

$\Rightarrow \; d_{i,j} \leq \delta_{i,j}$

Proof sketch (cont'd)

- The cost function can be written as

$$f(Y) = \sum_{i,j} \sum_{k=q+1}^{p} \left(x_i^T e_k - x_j^T e_k\right)^2$$

$$= \mathbf{tr}\left(R_2 \sum_{i,j} (x_i - x_j)(x_i - x_j)^T R_2^T\right)$$

$$= 2n^2 \mathbf{tr}(R_2 S R_2^T)$$

- Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ be eigenvalues of nS and $v_1, v_2, \ldots, v_p$ be the corresponding eigenvectors

- Let $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ and $V = (v_1, v_2, \ldots, v_p)$

Proof sketch (cont’d)

- The cost function can be written as

$$f(Y) = 2n\mathbf{tr}(F_2 \Lambda F_2^T)$$

where $F_2 = R_2 V$

- F_2 is a row orthonormal matrix: $F_2 F_2^T = I_{p-q}$

- We observe that $f(Y)$ is minimized when $F_2 = (\mathbf{0}, I_{p-q})$

- That is when $R_2 = \left(v_{q+1}, \dots, v_p\right)^T$

Proof sketch (cont'd)

- Rows of R_1 span the space of the first q eigenvectors of nS
- It follows that $R_1^T X$ represents the principal coordinates of X in q dimensions
- For the principal coordinates projection, we have

$$f(Y) = 2n(\lambda_{q+1} + \cdots + \lambda_p)$$

Classical scaling and PCA

- Let $S_X = \frac{1}{n-1}(XH)(XH)^T$ be the sample covariance of X

- Theorem: Projecting centered X onto the eigenvectors of $(n-1)S_X$ returns the classical scaling solution

Classical scaling and PCA: proof sketch

- The eigenvalues of $(n-1)S_X$ are the p non-zero eigenvalues of B

- Let v_i be the unit length eigenvector of B corresponding to eigenvalue λ_i

- $(XH)Bv_i = (n-1)S_X(XH)v_i = \lambda_i(XH)v_i$

  $\Rightarrow$ λ_i is an eigenvalue of $(n-1)S_X$ with the corresponding eigenvector

$$\xi_i = (XH)v_i$$

Classical scaling and PCA: proof sketch (cont'd)

- Note: $\xi_i^T \xi_i = \lambda_i$
- Let $\tilde{\xi}_i = \lambda_i^{-1/2} \xi_i$ be the scaled vector so that it is of unit length
- Projecting the centered X on the the unit vector $\tilde{\xi}_i$ we obtain

$$(XH)^T \tilde{\xi}_i = \lambda_i^{1/2} v_i$$

$\Rightarrow$ projecting centered X onto the eigenvectors of $(n-1)S_X$ is the classical scaling solution

SMACOF

- SMACOF: scaling by majorizing a complicated function
- An iterative optimization method for metric MDS
- We can write:

$$f(Y) = \mathbf{tr}(Y^T L_W Y) - 2\mathbf{tr}(Y^T B(Y) Y) + C_{\delta,w}$$

where

$(B(Y))_{i,j} = w_{i,j}\delta_{i,j}/d_{i,j}(Y)$ if $d_{i,j}(Y) > 0$ and $(B(Y))_{i,j} = 0$ otherwise

and $C_{\delta,w} = \sum_{i<j} w_{i,j}\delta_{i,j}$

MM iterative method

- Function $g(y, z)$ is a majorization function of $f(y)$ if it satisfies the following two conditions:
  - Dominance condition: $f(y) \leq g(y, z)$ for all y, for some z
  - Tangency condition: $f(y) = g(y, y)$
- MM iterative method: given initial point $Y^{(0)}$,

$$Y^{(t+1)} = \mathrm{argmin}_Y \, g(Y, Y^{(t)}), \text{ for } t \geq 0$$

- Convergence: $f\left(Y^{(t+1)}\right) \leq g\left(Y^{(t+1)}, Y^{(t)}\right) \leq g\left(Y^{(t)}, Y^{(t)}\right) = f(Y^{(t)})$

MM iterative method for MDS

- $f(Y)$ is majorized by

$$g(Y,Z) = \mathbf{tr}(Y^T L_W Y) - 2\mathbf{tr}(Y^T B(Z) Z) + C_{\delta,w}$$

- Proof hint: show that

$$\tfrac{1}{2}\big(f(Y) - g(Y,Z)\big) = \mathbf{tr}(Y^T L_W Y) - \mathbf{tr}(Y^T B(Z) Z) \geq 0$$

- Note that

$$\nabla_Y g(Y,Z) = 2L_W Y - 2B(Z)Z$$

- Hence, $\nabla_Y g(Y,Z) = 0$ is equivalent to $Y = L_W^{+} B(Z) Z$ (Guttman's transform)

MM iterative method for MDS (cont'd)

- The MM iterative method for MDS is given by

$$Y^{(t+1)} = L_W^{+} B\left(Y^{(t)}\right) Y^{(t)} \text{ for } t \geq 0$$

- Special case for unit weights:

$$Y^{(t+1)} = \frac{1}{n} B\left(Y^{(t)}\right) Y^{(t)} \text{ for } t \geq 0$$