Machine Learning

Section 13: ISOMAP, LLE, MVU, MDS

Stefan Harmeling

WS 2021/22

What we have seen so far?

Sections:

1. Introduction
2. Plausible reasoning and Bayes Rule
3. From Logic to Probabilities
4. Bayesian networks
5. Continuous Probabilities
6. The Gaussian distribution
7. More on distributions, models, MAP, ML
8. Linear Regression
9. Matrix Differential Calculus
10. Model selection
11. Support ~~Vector~~ Machines
12. Principal Component Analysis

Machine Learning

Section 13: ISOMAP, LLE, MVU, MDS

Stefan Harmeling

WS 2021/22

Definition 13.1 (Dimensionality reduction)

Given n data points $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$. Find a low dimensional representation $Z = [z_1, \ldots, z_n] \in \mathbb{R}^{d \times n}$ with $d \ll D$ that keeps most of the properties of the higher dimensional data.

What properties should be kept?

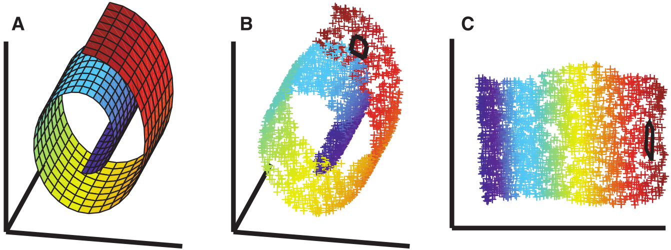
- variance (PCA, linear methods)
- neighborhood relationships (ISOMAP, LLE, MVU, nonlinear methods)

Linear dimensionality reduction

Data points are assumed to lie on an linear subspace, i.e. on a line, plane, hyperplane or some $\mathbb{R}^d$ somewhere inside the $\mathbb{R}^D$ for $d < D$.

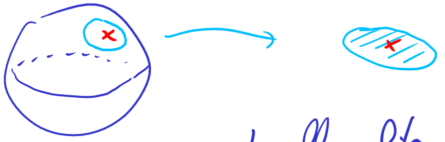
Nonlinear dimensionality reduction

Data points are assumed to lie on an curved subspace, i.e. on a curve or more generally a *manifold* inside the $\mathbb{R}^D$. Simply put, a *manifold* is a subset of $\mathbb{R}^D$ that locally looks like the $\mathbb{R}^d$ with $d < D$.



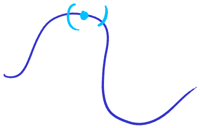
from Roweis/Saul's paper on LLE

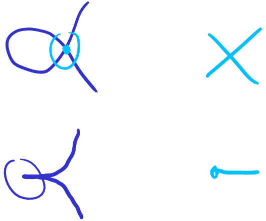
Manifold:



Locally like $\mathbb{R}^2$

„Manifold hypothesis": Our data lies on a manifold





Nonlinear dimensionality reduction

from http://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

- Sammon's mapping, 1969
- Self-organizing map (SOM, aka Kohonen map, based on neural networks), 1982
- Principal curves and manifolds, 1984
- Autoencoders (some neural networks), 19xx
- Generative topographic map (GTM, probabilistic version of SOM), 1996
- Curvilinear component/distance analysis (CCA, CDA), 1997
- Kernel PCA (kPCA), 1998
- ISOMAP, 2000
- Locally-linear embedding (LLE), 2000
- Laplacian Eigenmaps, 2001
- Hessian LLE, 2003
- Gaussian process latent variable models (GPLVM), 2004
- Maximum variance unfolding (MVA, aka semidefinite embedding), 2004
- Relational perspective map, 2004
- Nonlinear PCA (based on neural networks), 2005
- Local tangent space alignment (LTSA), 2005
- Modified LLE, 2006
- Diffusion maps, 2006
- Local multidimensional scaling, 2006
- Manifold alignment, 2008
- Manifold sculpting, 2008
- Diffeomorphic Dimensionality Reduction (Diffeomap), 2009
- Rank visu, 2009
- Topologically Constrained Isometric Embedding (TCIE), 2010
- t-distributed stochastic neighbor embedding (t-SNE), 2008

Two important papers both in a single issue of Science

23; right 36, 13, and 27); superior frontal gyrus (left −9, 31, and 45; right 17, 35, and 37).

17. Although the improvement in WM performance with cholinergic enhancement was a nonsignificant trend in the current study ($P = 0.07$), in a previous study (9) with a larger sample ($n = 13$) the effect was highly significant ($P < 0.001$). In the current study, we analyzed RT data for six of our seven subjects because the behavioral data for one subject were unavailable due to a computer failure. The difference in the significance of the two findings is simply a result of the difference in sample sizes. A power analysis shows that the size of the RT difference and variability in the current sample would yield a significant result ($P = 0.01$) with a sample size of 13. During the memory trials, mean RT was 1180 ms during placebo and 1119 ms during physostigmine. During the control trials, mean RT was 735 ms during placebo and 709 ms during physostigmine, a difference that did not approach significance ($P = 0.24$), suggesting that the effect of cholinergic enhancement on WM performance is not due to a nonspecific increase in arousal.
18. Matched-pair *t* tests (two-tailed) were used to test the significance of drug-related changes in the volume of regions of interest that showed significant response contrasts.
19. H. Sato, Y. Hata, H. Masui, T. Tsumoto, *J. Neurophysiol.* **55**, 765 (1987).
20. M. E. Hasselmo, *Behav. Brain Res.* **67**, 1 (1995).
21. M. G. Baxter, A. A. Chiba, *Curr. Opin. Neurobiol.* **9**, 178 (1999).
22. B. J. Everitt, T. W. Robbins, *Annu. Rev. Psychol.* **48**, 649 (1997).
23. R. Desimone, J. Duncan, *Annu. Rev. Neurosci.* **18**, 193 (1995).
24. P. C. Murphy, A. M. Sillito, *Neuroscience* **40**, 13 (1991).
25. M. Corbetta, F. M. Miezin, S. Dobmeyer, G. L. Shulman, S. E. Peterson, *J. Neurosci.* **11**, 2383 (1991).
26. J. V. Haxby *et al.*, *J. Neurosci.* **14**, 6336 (1994).
27. A. Rosier, L. Cornette, G. A. Orban, *Neuropsychobiology* **37**, 98 (1998).
28. M. E. Hasselmo, B. P. Wyble, G. V. Wallenstein, *Hippocampus* **6**, 693 (1996).
29. S. P. Mewaldt, M. M. Ghoneim, *Pharmacol. Biochem. Behav.* **10**, 1205 (1979).
30. M. Petrides, *Philos. Trans. R. Soc. London Ser. B* **351**, 1455 (1996).
31. M. E. Hasselmo, E. Fransen, C. Dickson, A. A. Alonso, *Ann. N.Y. Acad. Sci.* **911**, 418 (2000).
32. M. M. Mesulam, *Prog. Brain Res.* **109**, 285 (1996).
33. R. T. Bartus, R. L. Dean III, B. Beer, A. S. Lippa, *Science* **217**, 408 (1985).
34. N. Qizilbash *et al.*, *JAMA* **280**, 1777 (1998).
35. J. V. Haxby, J. Ma. Maisog, S. M. Courtney, in *Mapping and Modeling the Human Brain*, P. Fox, J. Lancaster, K. Friston, Eds. (Wiley, New York, in press).
36. We express our appreciation to S. Courtney, R. Desimone, Y. Jiang, S. Kastner, L. Latour, A. Martin, L. Pessoa, and L. Ungerleider for careful and critical review of the manuscript. We also thank M. B. Schapiro and S. I. Rapoport for input during early stages of this project. This research was supported by the National Institute on Mental Health and National Institute on Aging Intramural Research Programs.

7 August 2000; accepted 15 November 2000

A Global Geometric Framework for Nonlinear Dimensionality Reduction

Joshua B. Tenenbaum,[1]* Vin de Silva,[2] John C. Langford[3]

Scientists working with large volumes of high-dimensional data, such as global climate patterns, stellar spectra, or human gene distributions, regularly confront the problem of dimensionality reduction: finding meaningful low-dimensional structures hidden in their high-dimensional observations. The human brain confronts the same problem in everyday perception, extracting from its high-dimensional sensory inputs—30,000 auditory nerve fibers or 10^6 optic nerve fibers—a manageably small number of perceptually relevant features. Here we describe an approach to solving dimensionality reduction problems that uses easily measured local metric information to learn the underlying global geometry of a data set. Unlike classical techniques such as principal component analysis (PCA) and multidimensional scaling (MDS), our approach is capable of discovering the nonlinear degrees of freedom that underlie complex natural observations, such as human handwriting or images of a face under different viewing conditions. In contrast to previous algorithms for nonlinear dimensionality reduction, ours efficiently computes a globally optimal solution, and, for an important class of data manifolds, is guaranteed to converge asymptotically to the true structure.

A canonical problem in dimensionality reduction from the domain of visual perception is illustrated in Fig. 1A. The input consists of many images of a person's face observed under different pose and lighting conditions, in no particular order. These images can be thought of as points in a high-dimensional vector space, with each input dimension corresponding to the brightness of one pixel in the image or the firing rate of one retinal ganglion cell. Although the input dimensionality may be quite high (e.g., 4096 for these 64 pixel by 64 pixel images), the perceptually meaningful structure of these images has many fewer independent degrees of freedom. Within the 4096-dimensional input space, all of the images lie on an intrinsically three-dimensional manifold, or constraint surface, that can be parameterized by two pose variables plus an azimuthal lighting angle. Our goal is to discover, given only the unordered high-dimensional inputs, low-dimensional representations such as Fig. 1A with coordinates that capture the intrinsic degrees of freedom of a data set. This problem is of central importance not only in studies of vision (*1–5*), but also in speech (*6*, *7*), motor control (*8*, *9*), and a range of other physical and biological sciences (*10–12*).

The classical techniques for dimensionality reduction, PCA and MDS, are simple to implement, efficiently computable, and guaranteed to discover the true structure of data lying on or near a linear subspace of the high-dimensional input space (*13*). 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 interpoint distances, equivalent to PCA when those distances are Euclidean. However, many data sets contain essential nonlinear structures that are invisible to PCA and MDS (*4*, *5*, *11*, *14*). For example, both methods fail to detect the true degrees of freedom of the face data set (Fig. 1A), or even its intrinsic three-dimensionality (Fig. 2A).

Here we describe an approach that combines the major algorithmic features of PCA and MDS—computational efficiency, global optimality, and asymptotic convergence guarantees—with the flexibility to learn a broad class of nonlinear manifolds. Figure 3A illustrates the challenge of nonlinearity with data lying on a two-dimensional "Swiss roll": points far apart on the underlying manifold, as measured by their geodesic, or shortest path, distances, may appear deceptively close in the high-dimensional input space, as measured by their straight-line Euclidean distance. Only the geodesic distances reflect the true low-dimensional geometry of the manifold, but PCA and MDS effectively see just the Euclidean structure; thus, they fail to detect the intrinsic two-dimensionality (Fig. 2B).

Our approach 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. The crux is estimating the geodesic distance between faraway points, given only input-space distances. For neighboring points, input-space distance provides a good approxima-

[1]Department of Psychology and [2]Department of Mathematics, Stanford University, Stanford, CA 94305, USA. [3]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15217, USA.

*To whom correspondence should be addressed. E-mail: jbt@psych.stanford.edu

ISOMAP

REPORTS

35. R. N. Shepard, *Psychon. Bull. Rev.* **1**, 2 (1994).
36. J. B. Tenenbaum, *Adv. Neural Info. Proc. Syst.* **10**, 682 (1998).
37. T. Martinetz, K. Schulten, *Neural Netw.* **7**, 507 (1994).
38. V. Kumar, A. Grama, A. Gupta, G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms* (Benjamin/Cummings, Redwood City, CA, 1994), pp. 257–297.
39. D. Beymer, T. Poggio, *Science* **272**, 1905 (1996).
40. Available at www.research.att.com/~yann/ocr/mnist.
41. P. Y. Simard, Y. LeCun, J. Denker, *Adv. Neural Info. Proc. Syst.* **5**, 50 (1993).
42. In order to evaluate the fits of PCA, MDS, and Isomap on comparable grounds, we use the residual variance $1 - R^2(\hat{D}_M, D_Y)$. D_Y is the matrix of Euclidean distances in the low-dimensional embedding recovered by each algorithm. $\hat{D}_M$ is each algorithm's best estimate of the intrinsic manifold distances: for Isomap, this is the graph distance matrix D_G; for PCA and MDS, it is the Euclidean input-space distance matrix D_X (except with the handwritten "2"s, where MDS uses the tangent distance). R is the standard linear correlation coefficient, taken over all entries of $\hat{D}_M$ and D_Y.
43. In each sequence shown, the three intermediate images are those closest to the points 1/4, 1/2, and 3/4 of the way between the given endpoints. We can also synthesize an explicit mapping from input space X to the low-dimensional embedding Y, or vice versa, using the coordinates of corresponding points $\{\mathbf{x}_i, \mathbf{y}_i\}$ in both spaces provided by Isomap together with standard supervised learning techniques (39).
44. Supported by the Mitsubishi Electric Research Laboratories, the Schlumberger Foundation, the NSF (DBS-9021648), and the DARPA Human ID program. We thank Y. LeCun for making available the MNIST database and S. Roweis and L. Saul for sharing related unpublished work. For many helpful discussions, we thank G. Carlsson, H. Farid, W. Freeman, T. Griffiths, R. Lehrer, S. Mahajan, D. Reich, W. Richards, J. M. Tenenbaum, Y. Weiss, and especially M. Bernstein.

10 August 2000; accepted 21 November 2000

Nonlinear Dimensionality Reduction by Locally Linear Embedding

Sam T. Roweis[1] and Lawrence K. Saul[2]

Many areas of science depend on exploratory data analysis and visualization. The need to analyze large amounts of multivariate data raises the fundamental problem of dimensionality reduction: how to discover compact representations of high-dimensional data. Here, we introduce locally linear embedding (LLE), an unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. Unlike clustering methods for local dimensionality reduction, LLE maps its inputs into a single global coordinate system of lower dimensionality, and its optimizations do not involve local minima. By exploiting the local symmetries of linear reconstructions, LLE is able to learn the global structure of nonlinear manifolds, such as those generated by images of faces or documents of text.

How do we judge similarity? Our mental representations of the world are formed by processing large numbers of sensory inputs—including, for example, the pixel intensities of images, the power spectra of sounds, and the joint angles of articulated bodies. While complex stimuli of this form can be represented by points in a high-dimensional vector space, they typically have a much more compact description. Coherent structure in the world leads to strong correlations between inputs (such as between neighboring pixels in images), generating observations that lie on or close to a smooth low-dimensional manifold. To compare and classify such observations—in effect, to reason about the world—depends crucially on modeling the nonlinear geometry of these low-dimensional manifolds.

Scientists interested in exploratory analysis or visualization of multivariate data (*1*) face a similar problem in dimensionality reduction. The problem, as illustrated in Fig. 1, involves mapping high-dimensional inputs into a low-dimensional "description" space with as many coordinates as observed modes of variability. Previous approaches to this problem, based on multidimensional scaling (MDS) (*2*), have computed embeddings that attempt to preserve pairwise distances [or generalized disparities (*3*)] between data points; these distances are measured along straight lines or, in more sophisticated usages of MDS such as Isomap (*4*), along shortest paths confined to the manifold of observed inputs. Here, we take a different approach, called locally linear embedding (LLE), that eliminates the need to estimate pairwise distances between widely separated data points. Unlike previous methods, LLE recovers global nonlinear structure from locally linear fits.

The LLE algorithm, summarized in Fig. 2, is based on simple geometric intuitions. Suppose the data consist of N real-valued vectors $\vec{X}_i$, each of dimensionality D, sampled from some underlying manifold. Provided there is sufficient data (such that the manifold is well-sampled), we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold. We characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. Reconstruction errors are measured by the cost function

$$\varepsilon(W) = \sum_i \left| \vec{X}_i - \Sigma_j W_{ij} \vec{X}_j \right|^2 \qquad (1)$$

which adds up the squared distances between all the data points and their reconstructions. The weights W_{ij} summarize the contribution of the jth data point to the ith reconstruction. To compute the weights W_{ij}, we minimize the cost

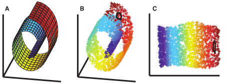


Fig. 1. The problem of nonlinear dimensionality reduction, as illustrated (*10*) for three-dimensional data (B) sampled from a two-dimensional manifold (**A**). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The color coding illustrates the neighborhood-preserving mapping discovered by LLE; black outlines in (**B**) and (**C**) show the neighborhood of a single point. Unlike LLE, projections of the data by principal component analysis (PCA) (*28*) or classical MDS (*2*) map faraway data points to nearby points in the plane, failing to identify the underlying structure of the manifold. Note that mixture models for local dimensionality reduction (*29*), which cluster the data and perform PCA within each cluster, do not address the problem considered here: namely, how to map high-dimensional data into a single global coordinate system of lower dimensionality.

[1]Gatsby Computational Neuroscience Unit, University College London, 17 Queen Square, London WC1N 3AR, UK. [2]AT&T Lab—Research, 180 Park Avenue, Florham Park, NJ 07932, USA.

E-mail: roweis@gatsby.ucl.ac.uk (S.T.R.); lsaul@research.att.com (L.K.S.)

LLE

Success of ISOMAP and LLE (checked Dec 2021)

- ISOMAP paper 15000 citations
  (Tenenbaum 75000, De Silva ?, Langford 41000)
- LLE paper 16000 citations
  (Roweis 38000, Saul 41000)
- generated a lot of follow-up papers
- published in Science journal

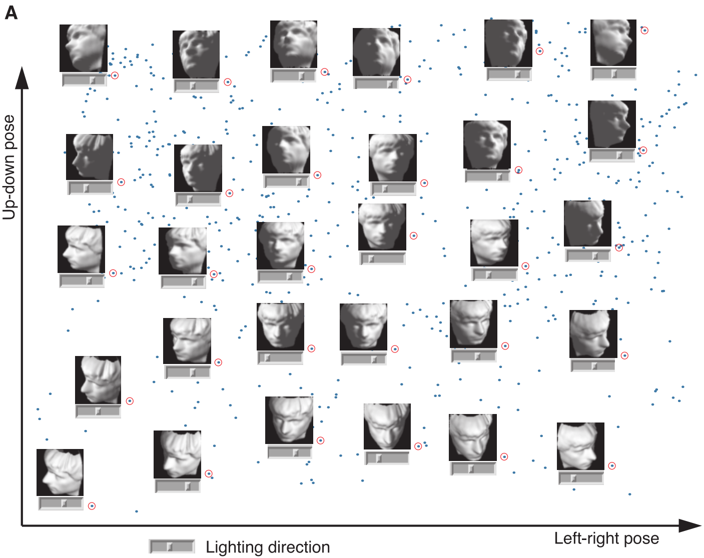
(numbers from `http://scholar.google.de` on 2021-29-11)

Why are ISOMAP and LLE so successful?

- easy to use
- easy to understand
- code freely available
- almost no parameter tweaking
- inspiring examples
- published in Science journal

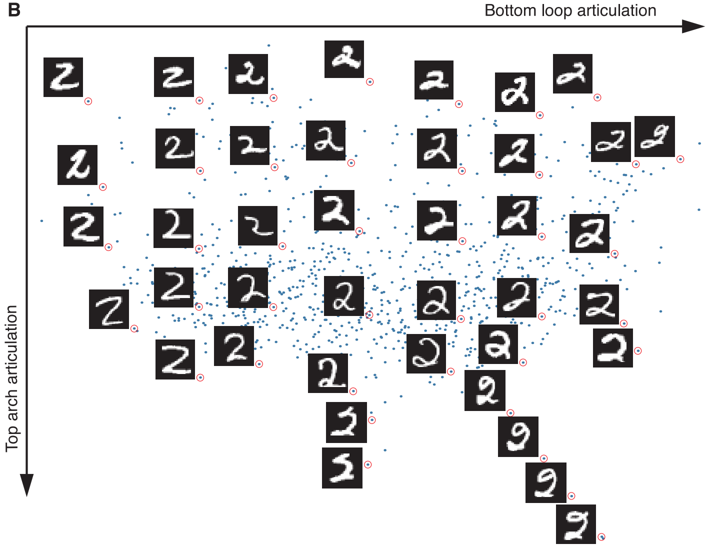
Do this with your work!

ISOMAP: manifold of faces (synthetic)



from Tenenbaum, da Silva, Langford's paper on ISOMAP

ISOMAP: manifold of digits



from Tenenbaum, da Silva, Langford's paper on ISOMAP

LLE: manifold of faces (real)

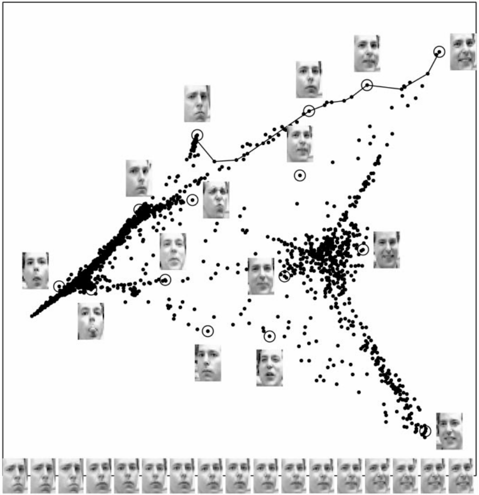


Fig. 3. Images of faces (*11*) mapped into the embedding space described by the first two coordinates of LLE. Representative faces are shown next to circled points in different parts of the space. The bottom images correspond to points along the top-right path (linked by solid line), illustrating one particular mode of variability in pose and expression.

from Roweis/Saul's paper on LLE

Both methods (ISOMAP and LLE) capture the manifold by defining a...

Neighborhood graph, aka proximity graph

Given a $D \times n$ data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$.

- vertices $x_1, \ldots, x_n$.
- edges between neighbors, i.e. close-by points

Examples

choose k

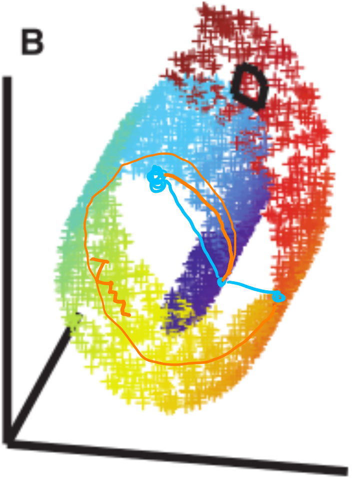
euclidean nearest in ambient space.

- k nearest neighbor graph (knn-graph)
  - the neighbors of x_i are its k-nearest data points
  - k is the parameter (what happens for $k = 0$, $k = 1, \ldots, k = n$)?
- ε graph
  - the neighbors of x_i are data points closer than ε
  - ε is a parameter (what happens for $\varepsilon \in [0, \infty)$?)

Which to choose?

- k nearest neighbor is more flexible (does not depend on the scaling).
- However, both do not guarantee that all points are connected in one graph.

B

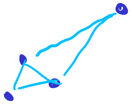


euclidean distance in the ambient space

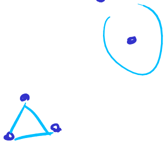
"geodesic distance" walking on the manifold. (jumping from pt. to pt. short dist.)

Idea: (edge weighted) Graph with vertices $x_1, \dots, x_n$ (data pts)
& edges = connection to close neighbors.

2NN-graph



ε-graph



Basic idea of ISOMAP

Calculate (geodesic) distances along the graph and find low dimensional embedding using multi dimensional scaling.
geodesic means “along the manifold”

Basic idea of locally linear embedding (LLE)

Approximate the manifold locally linearly, then reconstruct a low dimensional embedding matching the local linear structure.

Basic idea of ISOMAP

Calculate (geodesic) distances along the graph and find low dimensional embedding using multi dimensional scaling.
geodesic means "along the manifold"

Algorithm 13.2 (ISOMAP, sketch)

1. *construct neighborhood graph*
2. *compute all-pairs-shortest paths along the graph*
3. *apply MDS to get low dimensional embedding*

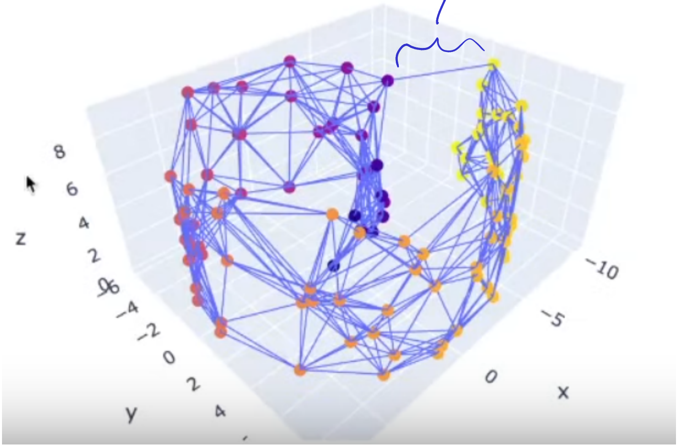
Basic idea of locally linear embedding (LLE)

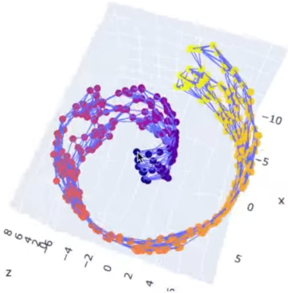
Approximate the manifold locally linearly, then reconstruct a low dimensional embedding matching the local linear structure.

Algorithm 13.3 (LLE, sketch)

1. *construct neighborhood graph*
2. *express data points as local linear combinations*
3. *solve eigenvalue problem to get low dimensional embedding*

k too big in relation to sample density




−10
−5
0
5
x
z

ISOMAP

ISOMAP: construct the neighborhood graph

Algorithm 13.4 (construct neighborhood graph)

Given a $D \times n$ *data matrix* $X = [x_1, \dots, x_n] \in \mathbb{R}^{D \times n}$.

1. *calculate all distances* $D_{ij} = \|x_i - x_j\|$
2. *for each point* x_i *make a list of its neighbors (either via* k *or* ε*)*
3. *define a weights matrix* W *with entries*

for those: geodesic = euclidean distance

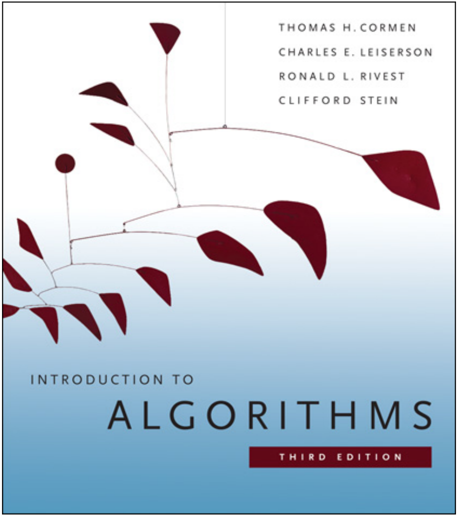
$$W_{ij} = \begin{cases} 0 & \text{if } i = j \\ D_{ij} & \text{if } x_i \text{ and } x_j \text{ are neighbors} \\ \infty & \text{otherwise} \end{cases}$$

For $0 < W_{ij} < \infty$ *there is an edge from* x_i *to* x_j.

- Question: is W symmetric?
- Answer: yes, for ε-graphs, no for knn-graph
- also for knn-graph we might symmetrise...

ISOMAP: calculate all-pairs-shortest paths (1)

Don’t reinvent the wheel, instead look it up, e.g. in CLRS:



ISOMAP: calculate all-pairs shortest paths (2)

Algorithm 13.5 (Floyd-Warshall algorithm, all-pairs shortest paths)

Given a graph with edge weights W_{ij} (their lengths) calculate a matrix D of all lengths of all paths along the graph. Initialize $D = W$.

1. *update all lengths in D wrt paths via x_1*
2. *update all lengths in D wrt paths via x_2*
3. *...*

n. *update all lengths in D wrt paths via x_n*

$$\begin{pmatrix} 0 & a & b & \infty & \cdots & \infty \\ & 0 & & & & \\ & & \ddots & & & \\ & \infty & & \ddots & & \\ \infty & \infty & & & & 0 \end{pmatrix}$$

```
def floyd_warshall(W):    # Python/numpy: 3 loops
    # assumes non-squared distances in W
    D = W.copy()
    n = D.shape[0]
    for k in range(n):
        for i in range(n):
            for j in range(n):
                D[i,j] = min(D[i,j], D[i,k]+D[k,j])
    return D
```

$\mathcal{O}(n^3)$

ISOMAP: calculate all-pairs shortest paths (2)

Algorithm 13.6 (Floyd-Warshall algorithm, all-pairs shortest paths)

Given a graph with edge weights W_{ij} (their lengths) calculate a matrix D of all lengths of all paths along the graph. Initialize $D = W$.

1. *update all lengths in D wrt paths via x_1*
2. *update all lengths in D wrt paths via x_2*
3. *...*

n. *update all lengths in D wrt paths via x_n*

```
# check with %prun or %lprun which is faster!
def floyd_warshall(W):    # Python/numpy: 2 loops
    # assumes non-squared distances in W
    D = W.copy()
    n = D.shape[0]
    for k in range(n):
        for i in range(n):
            D[i,:] = np.minimum(D[i,:], D[i,k]+D[k,:])
    return D
```

$\mathcal{O}(n^3)$

Result: Matrix with geodesic distances

ISOMAP: calculate all-pairs shortest paths (2)

Algorithm 13.7 (Floyd-Warshall algorithm, all-pairs shortest paths)

Given a graph with edge weights W_{ij} (their lengths) calculate a matrix D of all lengths of all paths along the graph. Initialize $D = W$.

1. *update all lengths in D wrt paths via x_1*
2. *update all lengths in D wrt paths via x_2*
3. *...*

n. *update all lengths in D wrt paths via x_n*

```
function fw(W)   % matlab version
    D = W # W[i,i]==0, W[i,j]=dij for neighbors, otherwise W[i,j]=inf
    n = size(W,1)
    for k in 1:n
       for i in 1:n
          for j in 1:n
             D[i,j] = min(D[i,j], D[i,k]+D[k,j])
          end
       end
    end
    return D
end
```

ISOMAP: summary

Algorithm 13.8 (ISOMAP)

Given a $D \times n$ data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$.

1. *construct neighborhood graph represented by $n \times n$ matrix W*
2. *compute all-pairs-shortest paths along the graph, ie. calc D*
3. *apply MDS transform distances D into embedding $Z = [z_1, \ldots, z_n]$*

What is MDS?

MDS = Multi-dimensional scaling

MDS in a nutshell

Given a distance matrix, recover a data matrix with those distances.

Data matrix

$$X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n} \text{ (with entries } X_{ki}) \quad \text{where we assume } \sum_{i=1}^{n} x_i = 0.$$

Gram matrix, aka inner product matrix, aka kernel matrix

$$G = X^\top X \in \mathbb{R}^{n \times n} \qquad \text{with entries } G_{ij} = x_i^\top x_j$$

Covariance matrix, aka outer product matrix (omitting $1/n$)

$$C = XX^\top = \sum_{i=1}^{n} x_i x_i^\top \in \mathbb{R}^{d \times d} \qquad \text{with entries } C_{kl} = \sum_{i=1}^{n} X_{ki} X_{li}$$

Squared distance matrix

$\langle x_i, x_i \rangle + \langle x_j, x_j \rangle - 2 \langle x_i, x_j \rangle$

$||$

$$D \in \mathbb{R}^{n \times n} \qquad \text{with entries } D_{ij} = \|x_i - x_j\|^2 = (x_i - x_j)^\top (x_i - x_j)$$

$G = X^T X$

Algorithm 13.9 (Multi-dimensional scaling, MDS, sketch)

Given a matrix D of the squared distances.

1. *Calculate Gram matrix $G = \ldots$ (the difficult step)*
2. *Calculate EVD of $G = V \Lambda V^{\mathsf{T}}$*
3. *Calculate data matrix $X = \Lambda^{1/2} V^{\mathsf{T}}$*

G symmetric & pos semidef.

$$\Lambda^{\frac{1}{2}} = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \end{pmatrix}$$

Note:

- $X^{\mathsf{T}} X = V \Lambda^{1/2} \Lambda^{1/2} V^{\mathsf{T}} = G$
- the mean of X is arbitrary, since it doesn't change the distances
- X can be arbitrarily rotated, since it doesn't change the distances

Question:

- How can we calculate the Gram matrix from the distance matrix?

Lemma 13.10

1. *Squared distances can be calculated from the Gram matrix*

$$D_{ij} = \boxed{x_i^\top x_i} + \boxed{x_j^\top x_j} - 2x_i^\top x_j = (x_i - x_j)^\top (x_i - x_j)$$
$$D = (G \odot I_n)1_n 1_n^\top + 1_n 1_n^\top (G \odot I_n) - 2G$$

with $G \odot I_n$ = `diagm(diag(G))` *being the matrix with the squared norms* $x_1^\top x_1, \ldots, x_n^\top x_n$ *along the diagonal.*

2. *Removing the mean from the ones matrix results in the zero matrix*

$$1_n 1_n^\top H = 1_n 1_n^\top - 1_n (1_n^\top 1_n) 1_n^\top / n = 0_{n \times n} = H 1_n 1_n^\top$$

where $H = I_n - 1_n 1_n^\top / n$ *is the centering matrix, note* $1_n^\top 1_n = n$

3. *Assuming the mean of dataset* X *is zero, i.e.* $X 1_n = 0_n$*, we have*

$$XH = X - X 1_n 1_n^\top / n = X - 0_n 1_n^\top / n = X$$

Furthermore we have for the Gram matrix $G = X^\top X$

$$HGH = HX^T XH = X^\top X = G$$

$$A \odot B = \begin{pmatrix} a_{11} & & a_{1n} \\ & \ddots & \vdots \\ & & a_{nn} \end{pmatrix} \odot \begin{pmatrix} b_{11} & & \\ & \ddots & \\ & & b_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} \cdot b_{11} & & \\ & \ddots & \\ & & a_{nn} \cdot b_{nn} \end{pmatrix}$$

Theorem 13.11

Assuming the mean of dataset X is zero, we have

$$G = -\frac{1}{2} HDH$$

Thus we can calculate the Gram matrix from the squared distance matrix under the assumption of zero mean.

Proof:

$$\begin{aligned} HDH &= H(G \odot I_n) 1_n 1_n^\top H + H 1_n 1_n^\top (G \odot I_n) H - 2HGH \\ &= H(G \odot I_n) 0_{n\times n} + 0_{n\times n} (G \odot I_n) H - 2G \\ &= -2G \end{aligned}$$

Algorithm 13.12 (Multi-dimensional scaling, MDS)

Given a matrix D of the squared distances.

1. *Calculate Gram matrix* $G = -\frac{1}{2} HDH$
2. *Calculate EVD of* $G = V \Lambda V^{\mathrm{T}}$
3. *Calculate data matrix* $X = \Lambda^{1/2} V^{\mathrm{T}}$

take „square root" of G

the columns of this will be our dimension reduced data pts.

`MDSDemo.ipynb`

ISOMAP: summary

Algorithm 13.13 (ISOMAP)

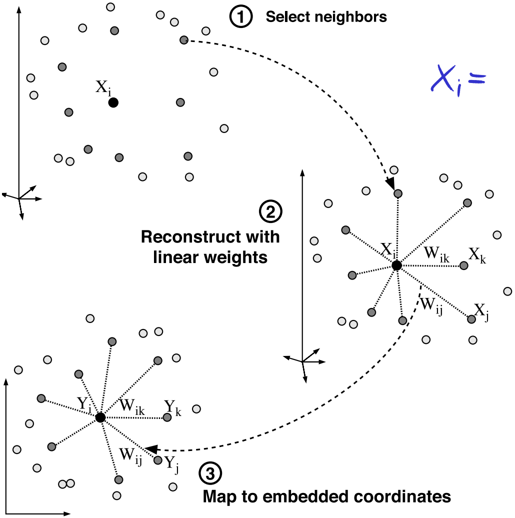
Given a $D \times n$ data matrix $X = [x_1, \dots, x_n] \in \mathbb{R}^{D \times n}$.

1. *construct neighborhood graph represented by $n \times n$ matrix W*
2. *compute all-pairs-shortest paths along the graph, ie. calc D*
3. *apply MDS transform distances D into embedding $Z = \underbrace{[z_1, \dots, z_n]}$*

n-dim points, but flat in $\mathbb{R}^n$, then do PCA for example

LLE

LLE: sketch (1)



$X_i = \sum_k W_{ik} Y_k$

from Roweis/Saul's paper on LLE

LLE: sketch (2)

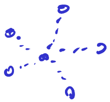
1. for each data point x_i select its neighbors, e.g. its k nearest neighbors
2. find weights W that locally reconstruct the data linearly (recall definition of manifold), i.e. minimize

$$\epsilon(W) = \sum_i \left\| x_i - \sum_j W_{ij} x_j \right\|^2$$

   with $W_{ij} = 0$ if x_j is not a neighbor of x_i and $\sum_j W_{ij} = 1$

3. find low dimensional embedding $Z = [z_1, \ldots, z_n]$ that minimizes

$$\phi(Z) = \sum_i \left\| z_i - \sum_j W_{ij} z_j \right\|^2$$



LLE: finding the weights (1)

Constraint optimization problem:

$$\begin{aligned} \min_W \quad & \sum_i \Big\| x_i - \sum_j W_{ij} x_j \Big\|^2 \\ \text{s.t.} \quad & \sum_j W_{ij} = 1 \text{ for } i = 1, \ldots, n \end{aligned}$$

Can be solved for each data point x with neighbors η_j separately

$$\begin{aligned} \min_w \quad & \Big\| x - \sum_j w_j \eta_j \Big\|^2 \\ \text{s.t.} \quad & \sum_j w_j = 1 \end{aligned}$$

LLE: finding the weights (2)

Rewrite objective

$$
\begin{aligned}
\frac{1}{2}\Big\| x - \sum_j w_j \eta_j \Big\|^2 &= \frac{1}{2}\Big\| \sum_j w_j (x - \eta_j) \Big\|^2 \qquad \text{because } \sum_j w_j = 1 \\
&= \frac{1}{2} \sum_{jk} w_j \underbrace{(x - \eta_j)^\mathsf{T}(x - \eta_k)}_{=:C_{jk}} w_k \\
&= \frac{1}{2} w^\mathsf{T} C w \qquad \text{where matrix } C \text{ has entries } C_{jk}
\end{aligned}
$$

Lagrangian

$$
\begin{aligned}
L(w, \lambda) &= \frac{1}{2} w^\mathsf{T} C w - \lambda (1^\mathsf{T} w - 1) \\
dL &= (w^\mathsf{T} C - \lambda 1^\mathsf{T}) dw \\
w &= \lambda C^{-1} 1 \\
w_j &= \lambda \sum_k (C^{-1})_{jk}
\end{aligned}
$$

LLE: finding the weights (3)

How to choose λ?

λ is the normalizing constant, i.e. choose it s.t. $1^\mathsf{T} w = 1$:

$$\text{solve } Cw = 1 \qquad\qquad \text{then enforce } 1^\mathsf{T} w = 1$$

Summary (how to find a weight vector for a single data point)

1. find nearest neighbors $\eta_1, \ldots, \eta_k$
2. calculate local covariance matrix C with entries $(x - \eta_j)^\mathsf{T}(x - \eta_k)$
3. define weights w according to formula above

Summary (how to combine the weight vectors into a $n \times n$ matrix W)

1. calculate weight vectors for all data points
2. for a data point x_i store the weights for each neighbor in the appropriate entries in the ith column of W, i.e. fill lots of zeros for non-neighbors

LLE: finding the embedding

Constraint optimization problem

$$\begin{aligned} \min_Y \quad & \sum_i \Big\| y_i - \sum_j W_{ij} y_j \Big\|^2 \\ \text{s.t.} \quad & Y\mathbf{1} = 0 \quad \text{zero mean} \\ & YY^\mathsf{T} = I \quad \text{covariance identity} \end{aligned}$$

Can be rewritten as:

$$\begin{aligned} \min_Y \quad & \operatorname{tr} YMY^\mathsf{T} \\ \text{s.t.} \quad & Y\mathbf{1} = 0 \\ & YY^\mathsf{T} = I \\ & M = I - W^\mathsf{T} - W + W^\mathsf{T} W = (I - W)^\mathsf{T} (I - W) \end{aligned}$$

This can be solved: one can show that the rows Y are the eigenvectors to the smallest eigenvalues. The math is almost identical to the simultaneous PCA problem (see appendix of PCA section).

LLE: summary

almost exactly copied from `https://www.cs.nyu.edu/~roweis/lle/algorithm.html`

Algorithm 13.14 (Locally linear embedding (LLE))

Given a $D \times n$ data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$.

1. *find nearest neighbors*

```
for i=1:N
  compute the distance from Xi to every other point Xj
  find the K smallest distances
  assign the corresponding points to be neighbours of Xi
end
```

2. *solve for reconstruction weights*

```
for i=1:N
  create matrix Z consisting of all neighbours of Xi
  subtract Xi from every column of Z
  compute the local covariance C=Z'*Z
  solve linear system C*w = 1 for w
  set Wij=0 if j is not a neighbor of i
  set the remaining elements in the ith row of W equal to w/sum(w);
end
```

3. *compute embedding coordinates*

```
create sparse matrix M = (I-W)'*(I-W)
find bottom d+1 eigenvectors of M
  (corresponding to the d+1 smallest eigenvalues)
set the qth ROW of Y to be the q+1 smallest eigenvector
  (discard the bottom eigenvector [1,1,1,1...] with eigenvalue zero)
```

MVU

Basic idea of maximum variance unfolding (MVU, aka semidefinite embedding)

Unfold the manifold by keeping local distances constant and maximizing all other distances.

Algorithm 13.15 (MVU, sketch)

1. *construct neighborhood graph*
2. *solve semidefinite programming problem*

Maximum variance unfolding (1)

see e.g. http://en.wikipedia.org/wiki/Semidefinite_embedding

Maximize the variance while keeping local distances constant:

$$\max_Y \quad \frac{1}{2n}\sum_{ij}\|y_i - y_j\|^2$$

$$\text{s.t.} \quad \|y_i - y_j\|^2 = \|x_i - x_j\|^2 \text{ if } x_i \text{ and } x_j \text{ are neighbors}$$

$$\sum_i y_i = 0 \qquad \text{i.e. embedding has mean zero}$$

Unfortunately, this problem is tough to solve. Is it convex? No.

Let's rewrite the problem with the Gram matrix $G = Y^\top Y$.

Maximum variance unfolding (2)

see e.g. http://en.wikipedia.org/wiki/Semidefinite_embedding

Assuming mean zero for the embedding implies:

$$\sum_{ij} G_{ij} = 1_n^\top G 1_n = (Y1_n)^\top Y1_n = 0 \quad \text{since } Y1_n = 0$$

Rewrite distances:

$$\|y_i - y_j\|^2 = G_{ii} + G_{jj} - G_{ij} - G_{ji}$$

Rewrite objective function:

$$\begin{aligned}\sum_{ij} \|y_i - y_j\|^2 &= \sum_{ij} (y_i^\top y_i + y_j^\top y_j - y_i^\top y_j - y_j^\top y_i) \\ &= \sum_{ij} y_i^\top y_i + \sum_{ij} y_j^\top y_j - \sum_{ij} y_i^\top y_j - \sum_{ij} y_j^\top y_i \\ &= 2(\sum_{ij} G_{ii} - \sum_{ij} G_{ij}) = 2n \operatorname{tr} G\end{aligned}$$

Maximum variance unfolding (2a) — matrix version

see e.g. http://en.wikipedia.org/wiki/Semidefinite_embedding

Assuming mean zero for the embedding implies:

$$1_n^\top G 1_n = 1_n^\top Y^\top Y 1_n = 0 \;\text{ since }\; Y 1_n = 0$$

Rewrite matrix of squared distances:

$$\begin{aligned} D_{ij} &= \|y_i - y_j\|^2 \\ D &= (G \odot I) 1_n 1_n^\top + 1_n 1_n^\top (G \odot I) - 2G \end{aligned}$$

Rewrite objective function, sum of squared distances:

$$\begin{aligned} 1_n^\top D 1_n &= 1_n^\top (G \odot I) 1_n 1_n^\top 1_n + 1_n^\top 1_n 1_n^\top (G \odot I) 1_n - 2(1_n^\top G 1_n) \\ &= 1_n^\top \operatorname{diag}(G) n + n \operatorname{diag}(G)^\top 1_n - 0 \\ &= 2n \operatorname{tr} G \end{aligned}$$

Maximum variance unfolding (3)

see e.g. http://en.wikipedia.org/wiki/Semidefinite_embedding

Maximum variance unfolding as semi-definite programming

$$\begin{aligned} \max_G \quad & \operatorname{tr} G \\ \text{s.t.} \quad & G_{ii} + G_{jj} - G_{ij} - G_{ji} = \|x_i - x_j\|^2 \text{ if } x_i \text{ and } x_j \text{ are neighbors} \\ & \sum_{ij} G_{ij} = 0 \\ & G \geq 0 \end{aligned}$$

The constraint $G \geq 0$ means that G must be positive (semi-)definite. This ensures that we can recover Y s.t. $G = Y^\mathsf{T} Y$.

Notes:

- now it is a convex optimization problem (see Boyd's book)
- called *semi-definite programming*, fast solvers available
- however, still expensive since G has many variables

How to map back and forth?

Summary of ISOMAP, LLE, MVU

Words of the day

- nonlinear dimensionality reduction
- manifold, swissroll
- proximity graph, knn graph, ε graph
- Euclidean vs geodesic distances
- ISOMAP, LLE, MVU, (and many more)
- semidefinite programming

How to estimate the dimension of the manifold?

X data matrix $(D < n)$ $\left(x_1 \middle| \dots \middle| x_n\right) = \begin{pmatrix} a_1 \\ \vdots \\ a_D \end{pmatrix}$

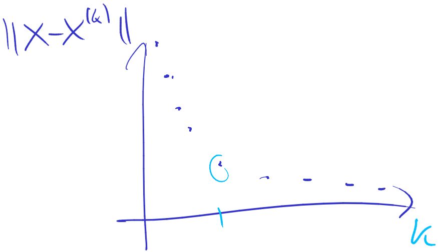
has rank D if we really need all of the D dim.s

Can try to approx. X by a rank d-matrix $X^{(d)}$

$\left(X = U \Sigma V^T \text{ SVD of } X \right.$, $\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ & 0 & & 0 \end{pmatrix}$

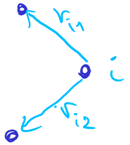
$\left. X^{(k)} := U \Sigma^{(k)} V^T \right)$ $\Sigma^{(k)} = \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & 0 \\ & & \sigma_k & & \\ & 0 & & 0 & \\ & & & & \ddots & 0 \end{pmatrix}$

Check $\|X - X^{(k)}\|$ for many k, then take last k where the distance is "small" $=: d$.



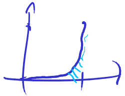
$\|X - X^{(k)}\|$
k

Another option: 2NN estimator (Facco et. al. 2017)



$$\mu_i = \frac{r_{i,2}}{r_{i,1}}$$

Curse of dimensionality:



Prob distr. of the random var μ:

$$P(\mu) = d \cdot \frac{1}{\mu^{d+1}}$$

Mean: $\displaystyle\int_1^\infty \mu \cdot d \cdot \frac{1}{\mu^{d+1}} \, d\mu = \int_1^\infty d \cdot \mu^{-d} \, d\mu$

$$= \left[-\frac{d}{d-1} \mu^{-(d-1)} \right]_1^\infty = \frac{d}{d-1}$$

Estimate $E\mu = \frac{d}{d-1}$, then solve for d.