ECE 657A/457B: Representation Learning

Feature Selection, Extraction, Dimensionality Reduction, Manifold Learning

Mark Crowley

February 2, 2025

Outline

Principal Component Analysis

- A way to linearly transform a set of d-dimensional vectors
- $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ into another set of m-dimensional vectors
- Has the property that most of the information content is stored in the first few dimensions, so we can have $m < d$
- The main idea is that high information corresponds to high variance (more discriminating).
- The direction of max variance is parallel to the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the sample matrix A.

Intuitive View

- Find new axes which will be better in terms of variance and errors than original ones.
- PCA is equivalent to minimizing the mean-square error. It also maximizes the scatter.

Visual Explanation:
`http://setosa.io/ev/principal-component-analysis/`

Implementing PCA

- Let R be the $d \times d$ covariance matrix of A
- A is normalized by subtracting mean

$$x'_{ij} = (x_{ij} - \bar{x}_j), i = 1, \ldots, n; j = 1, \ldots, d$$

- R is symmetric positive definite, its eigenvalues are real and positive
- Now we apply an orthogonal transformation to R to diagonalize it.

$$\mathrm{CRC}^T = \Lambda_d \tag{1}$$

- Where Λ_d is a diagonal matrix of the d *eigenvalues* of R
- and C is a matrix with columns corresponding to the *eigenvectors* of R

Implementing PCA

We can sort the eigenvalues of R such that

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \geq \lambda_d \geq 0 \tag{2}$$

and $\hat{c}_1, \hat{c}_2, \hat{c}_3, \ldots, \hat{c}_d$ are the corresponding eigenvectors, called the **Principal Components** (axes)

Selecting m dimensions

- If we want to reduce the dimensions but keep a large percentage of the variance in the data
- Then we can select the 1st m eigenvalues and eigenvectors

Let $H_m = \begin{bmatrix} \hat{c}_1^T \\ \hat{c}_2^T \\ \vdots \\ \hat{c}_m^T \end{bmatrix}$ be an $m \times d$ matrix.

Implementing PCA

Then

$$\begin{aligned} \bar{y}_i &= H_m \bar{x}_i, & i = 1, 2, \ldots, n \\ m \times 1 &= m \times d \cdot d \times 1 \end{aligned}$$

The projected matrix B_m

$$B_m = \begin{bmatrix} \bar{y_1}^T \\ \bar{y_2}^T \\ \vdots \bar{y_n}^T \end{bmatrix} = \begin{bmatrix} \bar{x_1}^T \\ \bar{x_2}^T \\ \vdots \bar{x_n}^T \end{bmatrix} H_m^T = AH_m^T$$

where

$$\begin{aligned} \bar{x}_m^T &= [x_{k1}, x_{k2}, \ldots, x_{kd}] \\ \bar{y}_m^T &= [x_{k1}, x_{k2}, \ldots, x_{kd}] \end{aligned}$$

Implementing PCA

The covariance matrix in the new space can be defined as

$$\begin{aligned}\frac{1}{n}B_m^T B_m &= \frac{1}{n}\sum_{i=1}^{n}\bar{y}_i\bar{y}_i^T = H_m R H_m^T \\ &= H_m(C^T \Lambda C)H_m^T = H_m C^T \Lambda (H_m C^T)^T \\ &= \Lambda_m = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)\end{aligned}$$

$$H_m C^T = \begin{bmatrix}\bar{c}_1^T \\ \bar{c}_2^T \\ \vdots \\ \bar{c}_m^T\end{bmatrix}[\bar{c}_1 \bar{c}_2 \ldots \bar{c}_m] = \begin{bmatrix}100\cdots\cdots 0 \\ 010\cdots\cdots 0 \\ \cdot\cdot 1\cdots\cdots 0 \\ \vdots \\ 10\cdots 1\cdot\cdot 0\end{bmatrix}$$

Which means the *m* new features are uncorrelated.

Sum of Eigenvalues

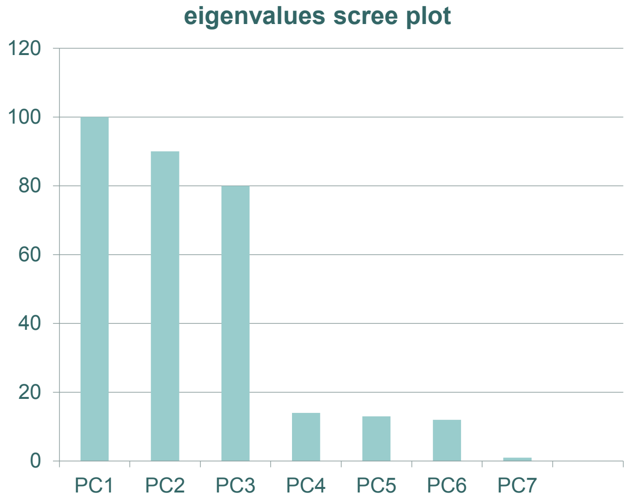
The sum of the Eigenvalues of R are the sample variance in the new space
One would choose m such that

$$r_m = \left(\sum_{i=1}^{m} \lambda_i\right) / \left(\sum_{i=1}^{d} \lambda_i\right) \geq \tau < 1$$

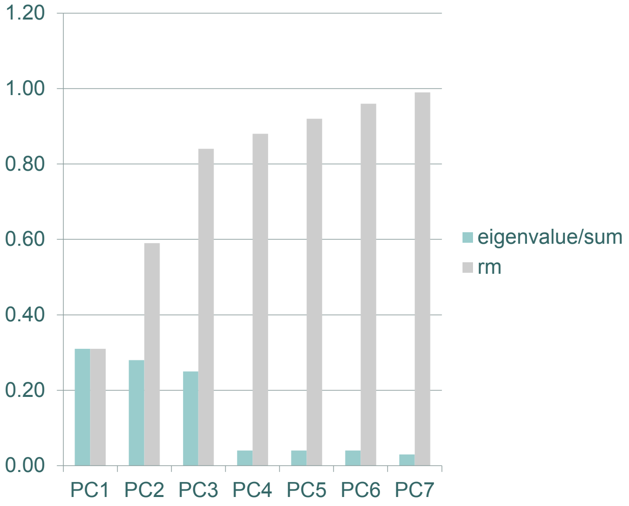
e.g. Choosing $\tau = 0.95$ will ensures that 95% of the variance is retained in the new space.
One way to know the right value is to use a **scree plot**.
The scree plot can be done in different ways:

- simply plotting the eigenvalues of the components (in descending order) and look for a gap or a knee in the plot.
- or plot r_m as a function of m and look for a knee in curve.
- could also plot the cumulative variance.

Scree Plot (Descending Eigenvectors)



Scree Plot (Normalized Eigenvectors)



Cost of Computation of PCA Optimization

- The approach we've shown so far is the most direct way to do PCA, but not the most efficient.
- The covariance matrix $R = A^T A$ is a $d \times d$ matrix and d (features) may be much larger than n (samples).
- Using this approach could be too complex.

Breaking Things Down...

A matrix A can be decomposed using **Singular Value Decomposition (SVD)** into $A_{n\times d} = USV^T$, where:

- U is a $n \times d$ matrix of orthonormal columns $U^TU = I$
  - the left singular vectors
- V is $d \times d$ matrix of orthonormal columns $V^TV = I$
  - the rght singular vectors
- S is $d \times d$ diagonal matrix of singular values.

PCA Using Singular Value Decomposition

SVD can be used to obtain PCA.
Now $AA^T = USV^T(VSU^T) = US^2U^T$
and $A^TA = VSU^T(USV^T) = VS^2V^T$
Which leads to the following facts:

- The singular values are the square root of the eigenvalues of the covariance matrix
- The right singular vectors are the eigenvectors of the covariance matrix.
- So, the SVD gives us the d eigenvalues (ordered) values as well as the principle components.
- Now we can *reduce the dimensions* by selecting the largest m.

Interpretation of PCA

- PCA is Optimal in the sense of min. sum of square of errors.
- It obtains max variance projection by finding orthogonal linear combinations of the original variables.
- It mainly rotates the coordinates (for zero mean data) to find new axes that have the max variance.
- It de-correlates the axes. For uni-modal Gaussian data this will amount to finding independent axes.
- For data that doesn't follow this distribution, the axes may not necessarily be independent.
- The principle components may not be the best for discriminating between classes.

Interpretation of PCA

- PCA is also called Karhunen-Loeve transform (KLT) or the Hotelling transform.
- The transformed points $\bar{y_1}, \bar{y_2}, \ldots, \bar{y_n}$ are sometimes called **scores**
- The eigenvectors or principal components $\bar{c_1}, \bar{c_2}, \ldots, \bar{c_d}$ are called **loadings** represented by **coefficients**
- The eigenvalues of the covariance matrix are sometimes called the **latent representation**
- Hotelling's T^2 value: measures the distance of the projected points from the centre of the entire entire projected space.

Great Explanation

- *PCA is a method of transforming a number of correlated variables into a smaller number of uncorrelated variables.*
- *It's similar to a how Fourier analysis is used to decompose a signal into a set of additive orthogonal sinusoids of varying frequencies, PCA decomposes a signal (or image) into a set of additive orthogonal basis vectors or eigenvectors.*
- *The main difference is that,*
    - *while Fourier analysis uses a fixed set of basis functions,*
    - *the PCA basis vectors are* **learnt** *from the data set via unsupervised training.*

https://blog.cordiner.net/2010/12/02/eigenfaces-face-recognition-matlab/

Manifold of Faces

- PCA can be applied to the task of **face recognition** by converting the pixels of an image into a number of eigenface feature vectors,
- these can then be compared to measure the similarity of two face images.

(see PCA slides for examples and more details)

Whitening (Sphering)

Goal is to have features that are

- less correlated together, each represents something independently important
- all the features have the same variance (a kind of a normalization)

A Whitening Transformation

- given a vector of random variables with known covariance matrix
- want a linear transformation into a set of new variables such that
  - covariance is the identity matrix (ie. they are uncorrelated)
  - all have variance 1

The transformation changes the input vector into a "white noise" vector.

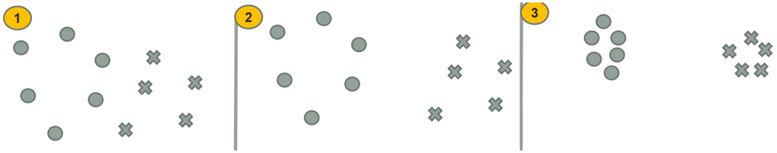
Performing Whitening

1. Shift data to zero mean (subtract mean from each feature)
2. Compute eigenvectors U and eigenvalues S using SVD
3. Project data using U and S to obtain whitened data points

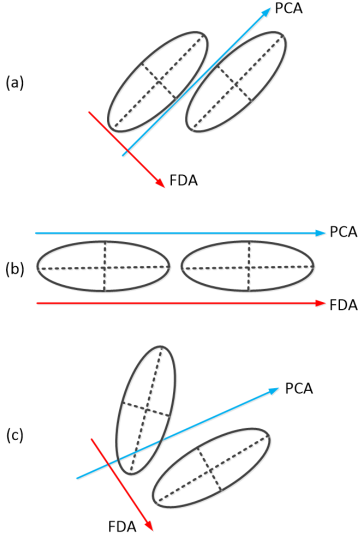
Outline of

Linear Discriminant Analysis (LDA)

- If we know the labels of the data (classes), we can use a supervised learning approach.
- Emphasis is on finding projections that best distinguish between (*discriminate*) the data in lower dimensions.
- Intuition: Maximize the *between-class scatter*(2) while holding minmizing the *within-class scatter*(3).



PCA vs. LDA



Fisher Linear Discriminant

- FLD is a special case of LDA
- Consider the 2 class problem. We have n samples x each of d dimensions divided into two classes
  - C_1 has n_1 samples
  - C_2 has n_2 samples
- We want to project these onto a line w such that the projected n points y (each is a scalar of one dimension) are divided into two classes, D_1 and D_2 such that

$$y = w^T y$$

- Our goal is to have a projection that well separates the projected points into two classes

Separation Measures

Use the difference between the projected points means of each group. The mean of original sample points in each class are:

$$\mathbf{m_i} = \frac{1}{n_i} \sum_{c \in C_i} \mathbf{x}, i = 1, 2$$

And mean of projected points are

$$\begin{aligned} m'_i &= \frac{1}{n_i} \sum_{y \in D_i} y \\ &= \frac{1}{n_i} \sum_{x \in C_i} w^T x \\ &= w^T \mathbf{m_i} \end{aligned}$$

Simple Mean Difference

$$\mathbf{m_i} = \frac{1}{n_i} \sum_{c \in C_i} \mathbf{x}, i = 1, 2$$
$$m_i' = w^T \mathbf{m_i}$$

Note that the original sample means are vectors of means of the features for samples in each class.

$$|m_1' - m_2'| = |w^T(\mathbf{m_1} - \mathbf{m_2})|$$

We can find w that maximizes this difference.
However, this difference can be made large simply by scaling w so we need to normalize it.

Within-Class Scatter

Fisher suggested normalizing the difference by the within-class scatter. The scatter is defined as

$$s_i^2 = \sum_{y \in D_i} (y - m_i')^2$$

$s_1^2 + s_2^2$ is called the within-class scatter of the projected points (n times the variance).
Define the sample within-class scatter of the original samples:

$$S_w = S_1 + S_2$$
$$S_i = \sum_{x \in C_i} (\mathbf{x} - \mathbf{m_i})(\mathbf{x} - \mathbf{m_i})^T$$

Between-Class Scatter

Substitute for y and m'

$$s_1^2 + s_2^2 = w^T S_w w$$
$$\text{similarly, } (m_1' - m_2')^2 = w^T S_B w$$

Where $S_B = (\mathbf{m_1} - \mathbf{m_2})(\mathbf{m_1} - \mathbf{m_2})^T$ is the **Between-Class Scatter**
So the problem can be now stated as finding w that maximizes

$$\frac{w^T S_B w}{w^T S_w W}$$

Between-Class Scatter

Maximize,

$$\frac{w^T S_B w}{w^T S_w W}$$

- The solution w needs to satisfy $S_W^{-1} S_B w = \lambda w$
- In general this is an eigenvalue problem, but in the case of 2 classes $S_B w$ is always in the direction of $(\mathbf{m_1} - \mathbf{m_2})$

Generalization of FLD to K classes

- Given K classes, we find a projection into $(K-1)$ - dimensional subspace.
- So the problem reduces to finding a $(K-1) \times d$ projection matrix W such that the projected sample points are well separated.

$$\mathbf{y} = W^T \mathbf{x}$$

- Find the projection matrix W by maximizing the between-group scatter while holding the within-group scatter constant.

[From [4]]

Generalization of FLD to K classes

Let there be K classes each of size n_i, for $i \in [1, K]$
Let the points in the ℓ^{th} group be the vectors

$$[\bar{x_1}^{\ell}, \ldots, \bar{x_1}^{\ell}]^T$$

where

$$\bar{x_j}^{\ell} = [x^{\ell}_{j_1}, \ldots, x^{\ell}_{j_d}]^T$$

The mean of the i^{th} feature for the ℓ^{th} group is

$$m_i^{(\ell)} = \frac{1}{n_i} \sum_{j=1}^{n_j} x_{j_i}^{(\ell)}$$

Generalization to K classes

The vector of feature means is then

$$\bar{m}^{(\ell)} = [m_1^{(\ell)}, m_2^{(\ell)}, \ldots, m_d^{(\ell)}]^T$$

and the *pooled mean* m is the mean vector over all samples

$$m = \frac{1}{n}\sum_{\ell=1}^{K} n_\ell m^{(\ell)} \qquad n = \sum_{\ell=1}^{K} n_\ell$$

Scatter Matrix

The scatter matrix is

$$S = \sum_{\ell=1}^{K} \sum_{j=1}^{n_\ell} (\bar{x}_j^{(\ell)} - m)(\bar{x}_j^{(\ell)} - m)^T$$

The scatter matrix of the ℓ^{th} group is

$$S^{(\ell)} = \sum_{j=1}^{n_\ell} (\bar{x}_j^{(\ell)} - \bar{m}^{(\ell)})(\bar{x}_j^{(\ell)} - \bar{m}^{(\ell)})^T$$

Within-Group and Between-Group Scatter

The within-group scatter matrix is then S_W

$$S_W = \sum_{\ell=1}^{K} S^{(\ell)}$$

And the between-group scatter matrix S_B is

$$S_B = \sum_{\ell=1}^{K} \sum_{j=1}^{n_\ell} (\bar{m}^{(\ell)} - m)(\bar{m}^{(\ell)} - m)^T = \sum_{\ell=1}^{K} n_\ell \bar{m}^{(\ell)} (\bar{m}^{(\ell)})^T - n\bar{m}\bar{m}^T$$

Combining Scatter Matricies

$$S = S_B + S_W$$

- The projection is to maximize S_B and keep S_W/S constant.
- The solution gives the rows of W projection matrix which as the $K-1$ eigenvectors of $S_W^{-1}S_B$ whose eigenvalues are non-zero.
- **Reducing dimensions:** to project to $t \leq (k-1)$ then use the eigenvectors of the largest t eigenvalues.
- **LDA vs PCA:** LDA is a supervised method. It's fast. Eigenvector based like PCA but generally better than PCA for classification. Limited to $k-1$ components.

Independent Component Analysis (ICA)

- PCA uses 2nd order stats (ie. mean and variance)
- Linear projection methods can use higher order stats so they can be used for non-Gaussian distributed data : eg. ICA, Projection Pursuit
- ICA was proposed for blind source separation of signals ("the cocktail party problem").
- It finds projections that are independent but may not be orthogonal.
- Each source can be any non-Gaussian distribution.

PCA vs. ICA

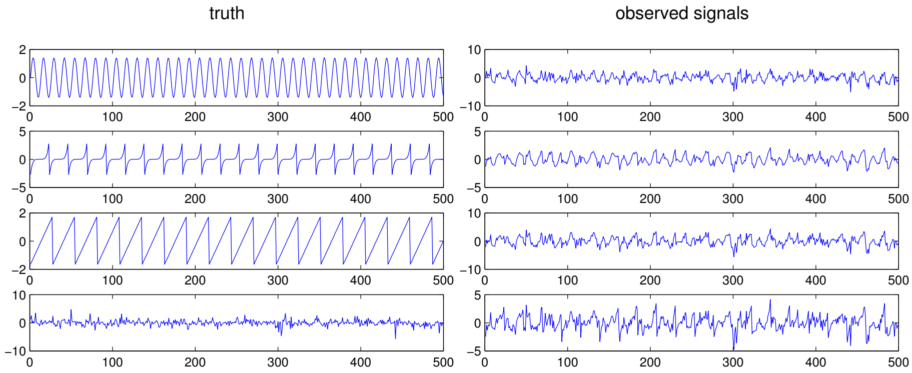


Figure: Truth vs. Observed

PCA vs. ICA

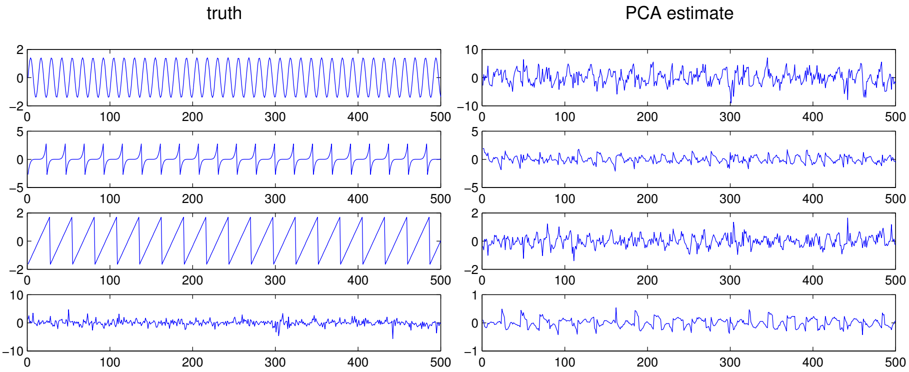


Figure: Truth vs. PCA

PCA vs. ICA

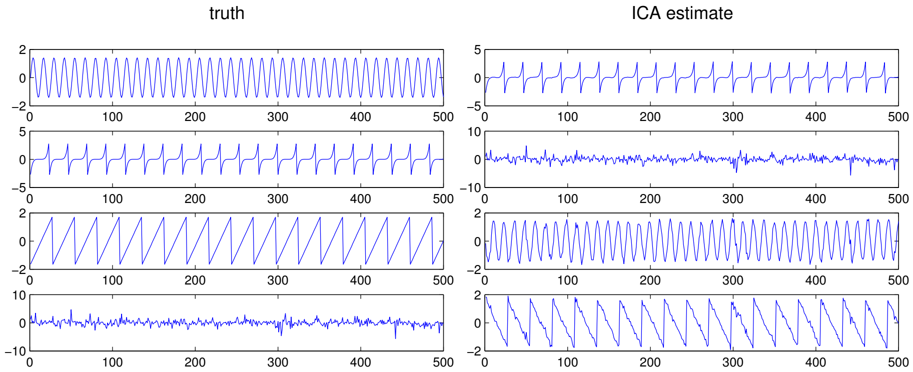


Figure: Truth vs. ICA

Projection Pursuit

- How do we choose each projection dimension?
- Projection pursuit uses a measure of interestingness of a projection.
  - Interestingness is a measure of some aspects of not being Gaussian (such as entropy).
  - It tries to find a projection that maximize the measure
  - Data is reduced by removing components along this projection.
  - Projection performs projections one at a time such that the extracted signal is as non-Gaussian as possible
    - The Gaussian distribution is the maximum entropy distribution.
    - So, we can maximize the **negative entropy (negentropy)**
    - This is equivalent to MLE up to a sign change and addition of a constant

Implementing ICA

Maximum Likelihood Estimation formulation :

$$\mathbf{x}_t = \mathbf{W}\mathbf{z}_t + \epsilon_t$$

where $\mathbf{x}_t = \Re^D$ is the observed signal and $\mathbf{z}_t = \Re^L$ is the vector of source signals.

$$\begin{aligned}
\mathrm{E}[\mathbf{x}\mathbf{x}^T] &= \mathbf{I} \text{ if data centred and whitened} \\
cov[\mathbf{x}] &= \mathrm{E}[\mathbf{x}\mathbf{x}^T] = \mathbf{W}\mathrm{E}[\mathbf{z}\mathbf{z}^T] \\
\mathbf{V} &= \mathbf{W}^{-1} \\
NLL(\mathbf{V}) &= \sum_{j=1}^{L} \mathrm{E}\left[G_j(z_j)\right]
\end{aligned}$$

Implementing ICA

Maximum Likelihood Estimation formulation :

$$\mathbf{x}_t = \mathbf{W}\mathbf{z}_t + \epsilon_t$$

where $\mathbf{x}_t = \Re^D$ is the observed signal and $\mathbf{z}_t = \Re^L$ is the vector of source signals.
But we don't know the form of G. So we fit it using:

- gradient descent (slow)
- approximate Newton method
- fit using Expectation-Maximization (EM) algorithm
- minimize mutual information

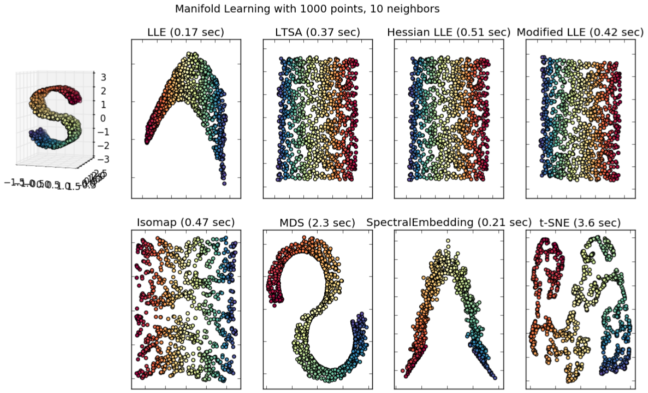
Outline of

Nonlinear DR or Manifold Learning

General Idea:

- Linear projections are not able to preserve complex data structures
- Linear projections attempt to find linear or near linear embedded structure in the data.
- They will fail to capture the intrinsic low dimension **manifold** that has a nonlinear structure.
- Example: data that lies on a simple nonlinear curve such as a circle or spiral in 2D (or 3D).

Manifold Learning



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

Multidimensional Scaling (MDS)

- Family of Nonlinear methods that map or find projection of the high dimensional data to low dimensions
- Focus on preserving the pairwise distance between data points.
- The discrepancy or error between the distances in the original and projected data is measured using a **stress function** based on distances.

[From [9]]

Common Stress Functions

(1) Based on square of the differences in distances

$$J = \sum_{i,j} (||x_i - x_j|| - ||y_i - y_j||)^2 = \sum_{i,j} (d(x_i, x_j) - d(y_i, y_j))^2$$

where

- $||x_i - x_j||$ is the Euclidean distance between the high dimensional points x_i, x_j. Each is a vector of d feature values.
- $||y_i - y_j||$ is the Euclidean distance between the corresponding low dimensional points y_i, y_j. Each is a vector of m features.

Common Stress Functions

(2) Based on fractional error. [Sammon, 1969]

$$J_s = \frac{1}{\sum_{ij} ||x_i - x_j||} \sum_{i \neq j} \frac{(||x_i - x_j|| - ||y_i - y_j||)^2}{||x_i - x_j||}$$

Comparison:

- (1) focuses on the errors whether the distance are large or small.
- (2) puts more emphasis on very small relative distances
- Variation of these functions have been also proposed.

MDS Solution

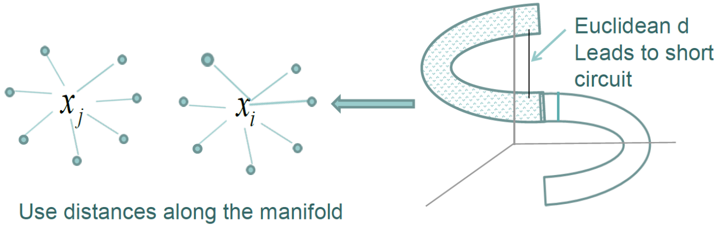
- **Objective:** find the y configuration that minimizes the stress function.
- Can be solved using iterative methods such as gradient-descent or Newton method
  - Start from a randomly chosen initial configuration
  - Iteratively improving solution
- One way to select this initial configuration in the m-space is to choose coordinates having largest variance (similar to PCA).

Comments on MDS

- Slow due to the computational cost
- The use of Euclidean distances may lead to considering point close while they are far on the manifold
- Doesn't model the geometry (the manifold) *[Silva and Tenenbaum, 2002]*
- To reduce the computational cost:
  1. Select a number of points less than n as landmark points.
  2. Perform MDS on the landmark points.
  3. Map the *rest of the points* using their distances to the landmark points.

ISOMAP

Isomap tries to preserve the distances along the manifold by using geodesic (or curvilinear) distance or approximation of it. (tenenbaum et.al, science 2000)



ISOMAP

Isomap tries to preserve the distances along the manifold by using geodesic (or curvilinear) distance or approximation of it. (tenenbaum et.al, science 2000)

three steps:

1- construct neighbourhood graph

2- approximate the geodesic distance between for all pairs in graph

3- apply MDS on pairwise distances

Step 1 - Construct the Neighbourhood Graph

- Every data point x_i is connected with its K nearest neighbors
- Neighbour defined as points within distance ϵ
- Construct graph $G = (X, E)$ where each edge $e = (x_i, x_j) \in E$ is the Euclidean distance between the points.

Step 2 - Compute Geodesic Distances

Geodesic Distance: number of edges between two nodes in some *shortest path* in the graph.
Approximation: if there is a path from x_i to x_j:

$$d_G(x_i, x_j) = \textsc{ShortestPath}_G(x_i, x_j)$$

otherwise:

$$d_G(x_i, x_j) = \infty$$

Step 2 - Compute Approximate Shortest Paths

The shortest path which is either a direct connection (i.e one the K neighbors) or through other points (sum of distances)

$$d_G(x_i, x_j) = \textsc{ShortestPath}_G(x_i, x_j)$$

This could be implemented by:

- Dijkstra's Algorithm
- Floyd-Warshall Algorithm

Note:

- K should be small to get good approximation of local neighborhood.
- $K = n$ reduces to direct Euclidean distances as in the pure MDS approach.

Step 3 - Apply MDS on Distances

- Apply Multi-dimensional Scaling
- Use the approximate pairwise geodesic distances between all the points in x as lower dimensional space y.
- Solve iteratively using gradient-descent or Newton method as usual.

Properties of ISOMAP

- ISOMAP captures the geometric properties of the space in which the data resides.
- Has proof for its convergence.
- Cannot handle new/online data directly (”out-of-sample points”).
- Computationally demanding (slow).
- May construct erroneous connections in the neighborhood graph (short circuiting).
- May suffer if there is a gap in the manifold or it may miss some ofthe samples if the manifold has disconnected components (or points don’t get connected through the neighborhood graph)

Locally Linear Embedding (LLE)

- Introduced same time as ISOMAP. (Roweis Saul, Science 2000)
- It models the manifold as a union of linear patches (local properties of the data).
- Local properties constructed by writing the data points x_i as a **linear combination of their $K(i)$ nearest neighbors** in the high dimension.
- Then the lower dimensional points y_i are expressed as the same linear combination of their corresponding neighbors $K(i)$ in the low dimension space.

Locally Linear Embedding (LLE)

Algorithm Contains 3 steps:

1 - Identify the neighbors $K(i)$ of each point x_i

2 - Compute weights vectors w_i that best linearly reconstruct x_i from its neighbors minimizing reconstruction errors

$$\min_w \sum_{i=1}^{n} ||x_i - \sum_{j \in k(i)} w_{ij} x_j||^2$$

s.t. $\sum_j w_{ij} = 2, w_{ij} = 0$ if $x_j \notin K(i)$
This can be solved using least squares method

Locally Linear Embedding (LLE)

3 - Given w find the low-dimensional set y_i which is best reconstructed by these weights

$$\min_y \sum_i ||y_i - \sum_{j \in k(i)} w_{ij} y_j||^2$$

Solving this with the assumption that $\frac{1}{n} YY^T = I$ amounts to Eigen decomposition

$$MY^T = \Lambda Y^T$$

Where M is sparse matrix given by:

$$M = (I - W)(I - W)^T$$

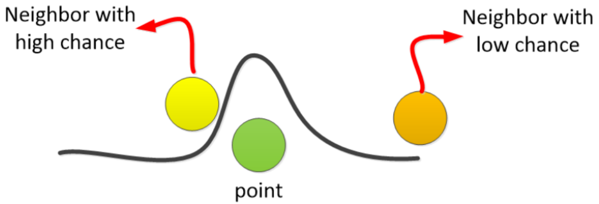
Result is Y^T are the Eigenvectors of M with Eigenvalues on diagonal of Λ.

Properties of LLE

- More computationally efficient than global methods like ISOMAP.
- May be able to represent wider range of manifolds whose local properties are captured by Euclidean geometry.
- Doesn't provide a direct form that can be applied to new data (no out of sample extension)
- No estimate of dimensionality (selecting m) unlike PCA where we have a ratio of variance
- Handles non-uniform sample densities poorly.
- May miss some sample points if
    - they don't get connected through the K neighbors
    - if the manifold has disconnected components.

t-Stochastic Neighbor Embedding (t-SNE)

- Rather than saying that this point is neighbor of that point but the other point is not a neighbor, we can have a *probabilistic* approach.
- We say, all points are *neighbours* of a point with some probability.
- A very similar/dissimilar point to some other point is its neighbour with high/low probability.

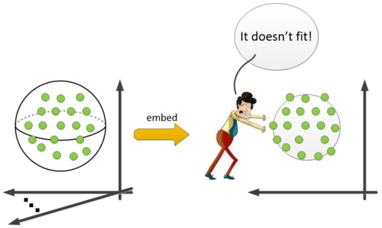


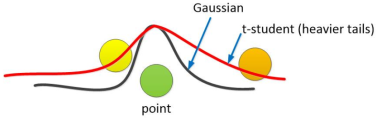
Stochastic Neighbour Embedding

- The original SNE approach uses *Gaussian distribution* for probability of neighbourhood.
- *See tSNE slides for more details*

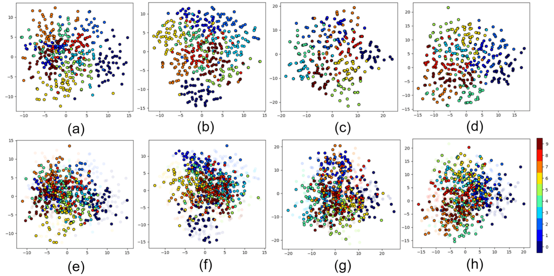
The Crowding Problem

- If we want to fit one million people in a room, they don't fit. So, let's enlarge the room!
- Crowding problem: If we want to fit the large information of high dimensional data into low dimensional subspace, we should enlarge the distribution!
- Student-t distribution has heavier tails than the Gaussian distribution.



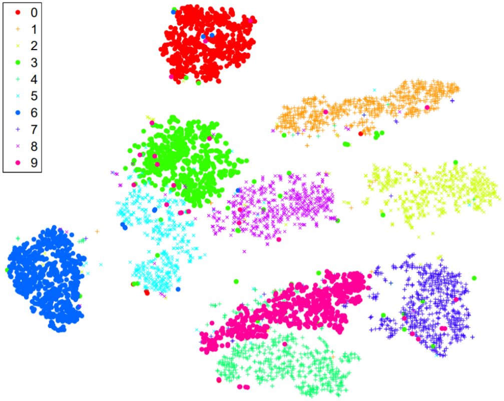


t-SNE Examples



The embeddings of training data are shown in (a) SNE, (b) symmetric SNE, (c) t-SNE (Cauchy-SNE), and (d) t-SNE with general degrees of freedom. The out-of-sample embeddings are shown in (e) SNE, (f) symmetric SNE, (g) t-SNE (Cauchy-SNE), and (h) t-SNE with general degrees of freedom.

t-SNE Examples



[Dunham, Data Mining Intro and Advanced Topics, 2003]
Margaret Dunham, Data Mining Introductory and Advanced Topics, ISBN:0130888923, Prentice Hall, 2003.

[Han,Kamber and Pei. Data Mining, 2011]
Jiawei Han, Micheline Kamber and Jian Pei, *Data Mining: Concepts and Techniques*, 3rd ed, Morgan Kaufmann Publishers, May 2011.

[Duda, Pattern Classification, 2001]
R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification (2nd ed.)*, John Wiley and Sons, 2001.

[Jain and Dubes. Algs for Clustering Data, 1988]
A. K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, ISBN: 0-13-022278-x, Prentice Hall, 1988.

[Cohen,Empirical Methods for Artificial Intelligence, 1995]
P. Cohen, Empirical Methods for Artificial Intelligence, ISBN:0-262-03225-2, MIT Press, 1995.

[Ackoff, From Data to Wisdom, 1989]

Ackoff, *From Data to Wisdom*, Journal of Applied Systems Analysis, 1989.

[Sima and Dougherty, 2008]
Sima, C. and Dougherty, E. R. *The Peaking Phenomenon in the Presence of Feature Selection.* Pattern Recognition Letters, 29, 1667-1674, 2008.

[Zhu and Ghodsi, 2006]
Mu Zhu, Ali Ghodsi, *Automatic dimensionality selection from the scree plot via the use of profile likelihood"*, Computational Statistics & Data Analysis 51 918 930, 2006.

[Cox, 2000]
Trevor Cox and M.A.A Cox, *Multidimensional Scaling*, Chapman and Hall/CRC, Second Edition, 2000.

Nonlinear Methods For Dimensionality Reduction

Additional material for the MDS section is based on the following references:

- Cox, T.F., Cox, M.A.A. Multidimensional Scaling. Chapman and Hall, 2001
- Tenenbaum, J. B., de Silva, V, Langford, J.C., "A global geometric framework for nonlinear dimensionality reduction " Science 290(5500): 2319-2323, 2000
- de Silva, V., Tenenbaum, J.B., Global versus local methods in nonlinear dimensionality reduction. In Neural Information Processing Systems. 15, 721-728, 2003.
- Roweis, S.T. and Saul, L.K., Nonlinear dimensionality reduction by Locally Linear Embedding. Science, 290(5500):2323-2326, 2000.
- van der Maaten, L.J.P., Postma, E.O., van den Herik, H.J., Dimensionality reduction: a comparative review. Tilburg University Technical Report, TiCC-TR 2009-005, 2009.