Ingénierie des systèmes décisionnels (NFE212)
Data Mining and Learning

Nicolas Thome
Conservatoire Nationnal des Arts et Métiers (Cnam)
Laboratoire CEDRIC

le cnam

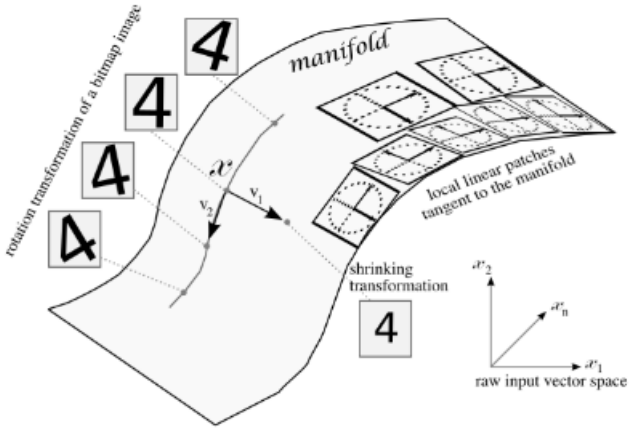
Cédric

Outline

Manifold Learning

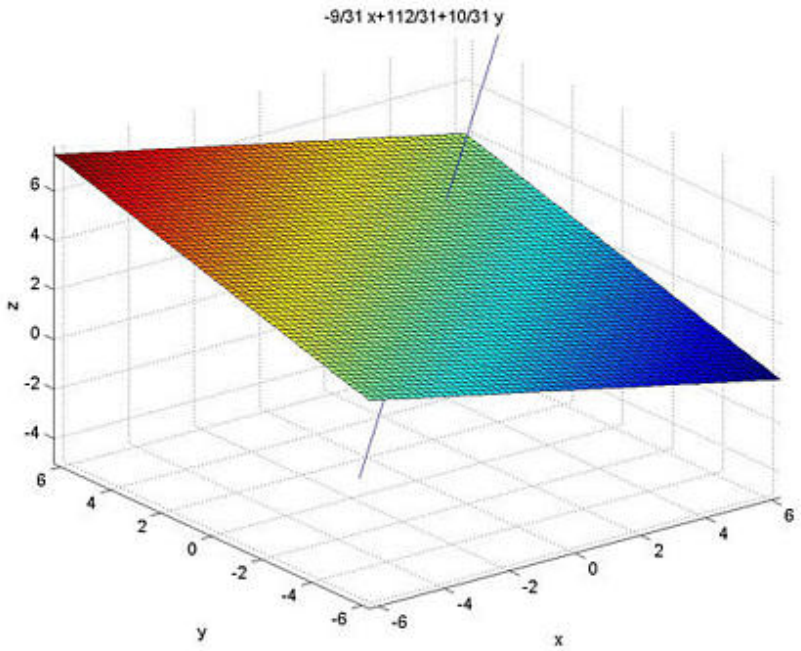
Manifold

- A curve is a manifold of dimension 1
- A surface is a manifold of dimension 2
- A manifold of dimension N is a locally euclidean space
- **Idea in manifold learning:** data lye in a (low-dimensional) manifold

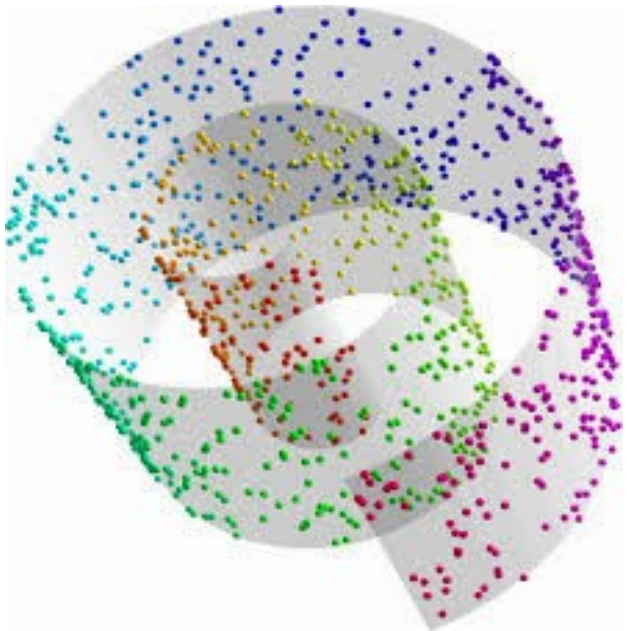


Manifold Learning

Linear and non-linear Manifolds



Linear manifold

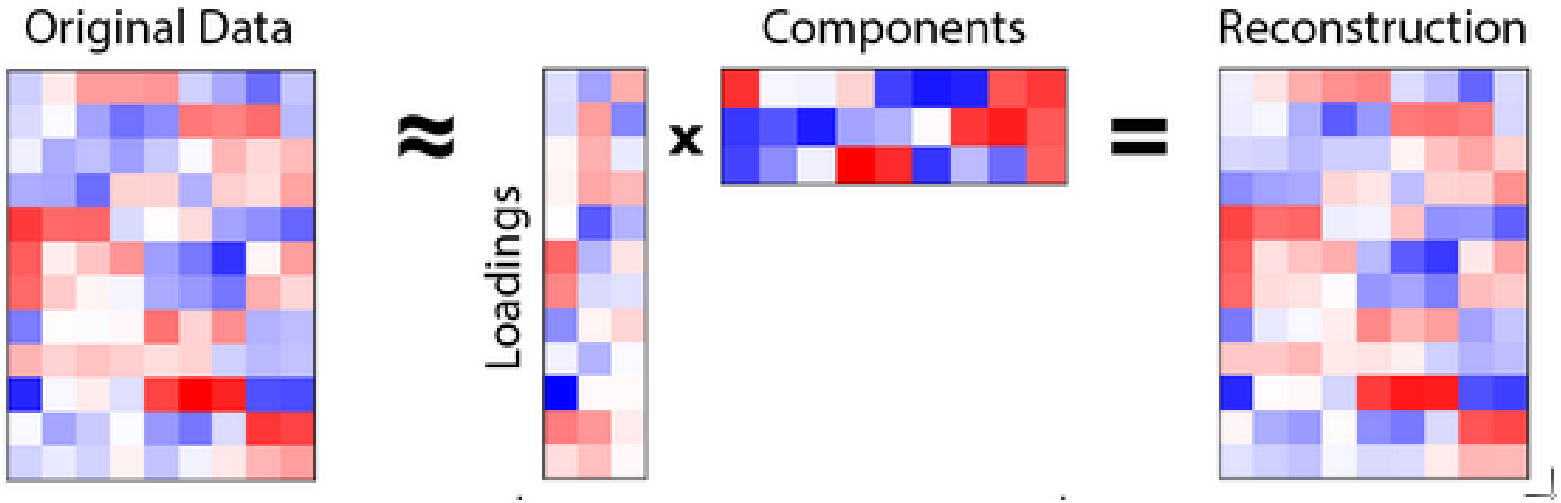


Non Linear Manifold

Linear Manifold Learning

Basis

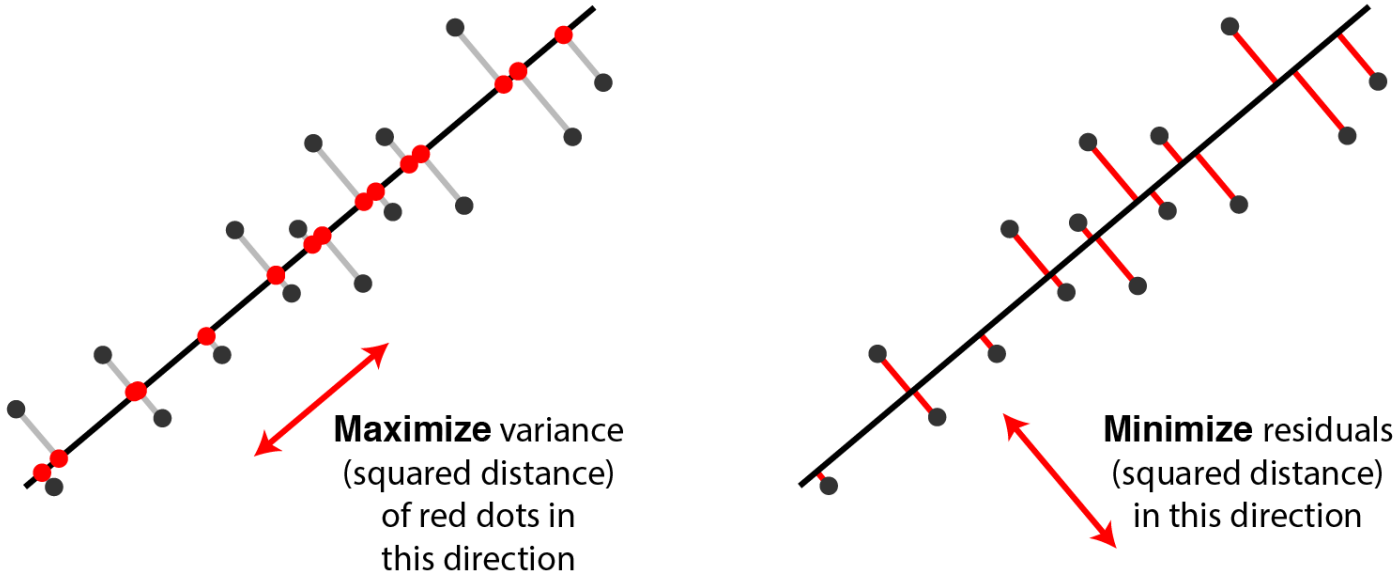
- Points in $\mathbb{R}^p$ vectorial space $\Rightarrow$: n points represented by $X_{p\times n}$ matrix
- Looking for a new basis to represent data: $X_{p\times n} = A_{p\times p}S_{p\times n}$
  - A columns: new basis vectors in $\mathbb{R}^p$
  - S columns: Projection of each example A : p coefficients
- Often related to dimension reduction: only keep a subset of vector basis, $k < p$, $X_{p\times n} \approx A_{p\times k}S_{k\times n}$
- Goal: Learning A and S from data
  - Changing representation space: S better than X to describe data



Linear Manifold Learning

Principal Component Analysis (PCA) [Jol86]

- Two equivalent views of PCA
  1. Maximizing projected variance
  2. Minimizing projected reconstruction error



Linear Manifold Learning

Principal Component Analysis (PCA) Formulation

- Goal: seeking vector leading to maximal projected variance
- Π projection of X on a vectorial line $\mathcal{V}$ defined by unit vector $\overrightarrow{v}$ ($||\overrightarrow{v}|| = 1$): $\Pi = vv^T$
- Projected variance :

$$\sigma_v^2(X) = \frac{1}{n}\sum_{i=1}^{n} ||\Pi(X_i - g)||^2 = \frac{1}{n}\sum_{i=1}^{n} \Pi(X_i - g)^T \ \Pi(X_i - g)$$

- It can be shown that $\sigma_v^2(X) = v^T \ \Sigma \ v$
  - $\Sigma = \frac{1}{n}\sum_{i=1}^{n}(X_i - g)(X_i - g)^T$ variance-covariance X matrix .

Principal Component Analysis (PCA)

Solution

- Maximize :

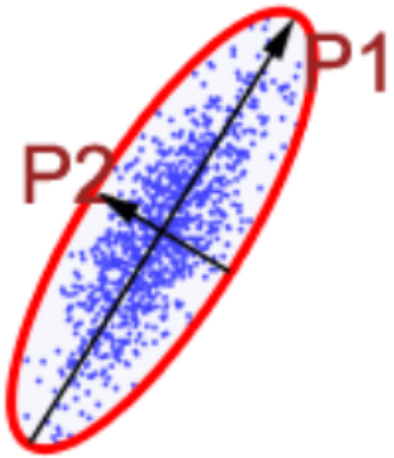
$$\max_{v}(v^T \Sigma v), \text{ st } ||\vec{v}|| = 1 \quad (1)$$

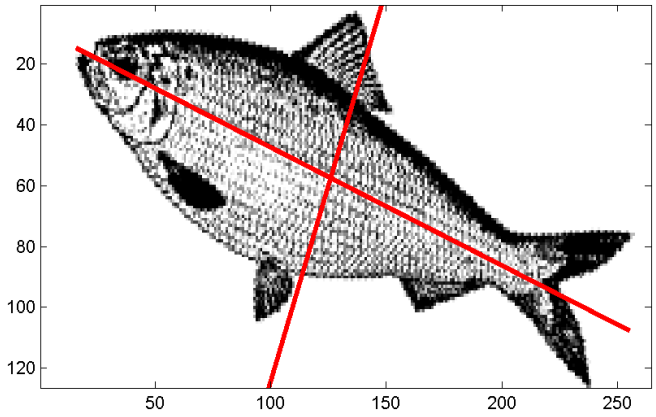
- Necessary condition: $\Sigma v = \lambda v$, $\lambda \in \mathbb{R}$
  - Solution: Σ diagonalization
  - λ_i : eigenvalue of i^{st} eigenvector $\lambda_i = \sigma^2_{v_i}(X)$
  - Sufficient condition: choosing eigenvector with largest eigenvalue
  - Extension to several vectors : look for i^{st} vector maximizing Eq. (1) and orthogonal to previous $i-1$
  - Orthonormal basis of eigenvectors, sorted by decreasing eigenvalues, is the solution to the problem

Principal Component Analysis (PCA)

Visualization

- Points in $\mathbb{R}^2$ plane
- 1^{ex} direction of maximal variation
- PCA solution gives best fitting ellipse





Principal Component Analysis (PCA)

Application: compression

- Various contexts:
  1. Compress a set of n images, each image with p pixels : X
  2. Compress an image: application for a set of n patches with p pixels
  3. Compress an image: X matrix $\Leftrightarrow$ *image*, thus compressing the n columns, each column with p (# rows) dimensions
- Σ Diagonalization and keeping $k << p$ eigenvectors
- Data reconstruction in the sub eigenspace
- Compression : we have $2k + 1$ vectors (A, S et mean) : $tx = \frac{2k+1}{n}$

Principal Component Analysis (PCA) : compression

17.7:1 compression
14 principal components

12.5:1 compression
20 principal components

8.4:1 compression
30 principal components

6.3:1 compression
40 principal components

4.2:1 compression
60 principal components

2.8:1 compression
90 principal components

2.1:1 compression
120 principal components

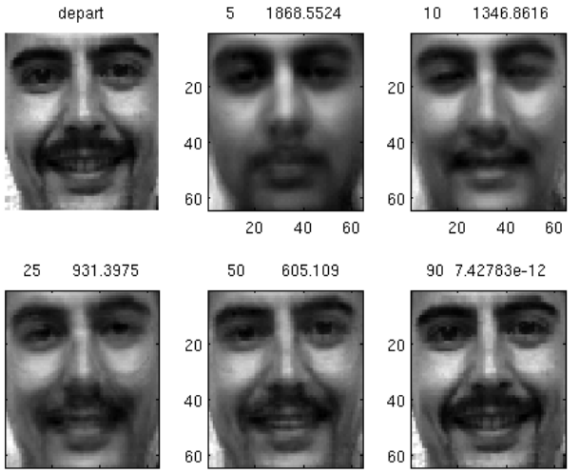
1.7:1 compression
150 principal components

1.4:1 compression
180 principal components

Principal Component Analysis (PCA)

Application : face recognition

- Previous case 1) : we have a set of n images (of faces), each image with p pixels : X
- Σ Diagonalization and keeping $k << p$ eigenvectors, "eigenfaces"
- Data reconstruction in the sub-space corresponding to Eigenfaces



- Identification: project a given image on the sub-space corresponding to Eigenfaces
- Find k-nn in the eigen sub-space $\Rightarrow$ classification

Linear Manifold Learning

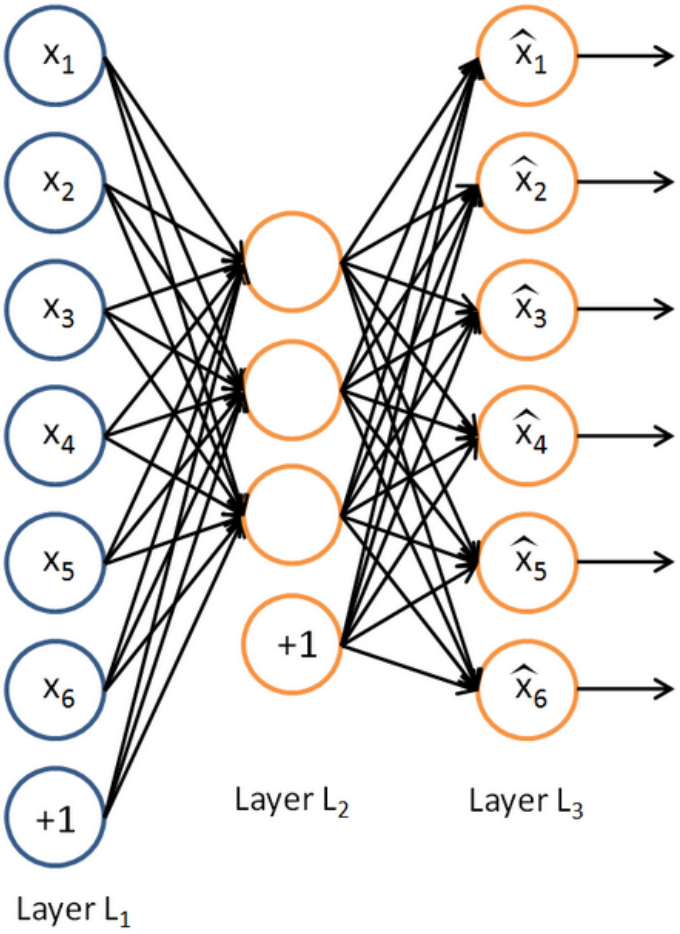
Auto-encoders [GBC16]

- $z = f(Wx)$
- $\tilde{x} = g(W^t z)$
- Auto-encoder objective function:

$$C = \sum_{i=1}^{N} ||x_i - \tilde{x}||^2$$

- If $f = g = Id$ (linear auto-encoder): $\sim$ PCA (reconstruction formulation):

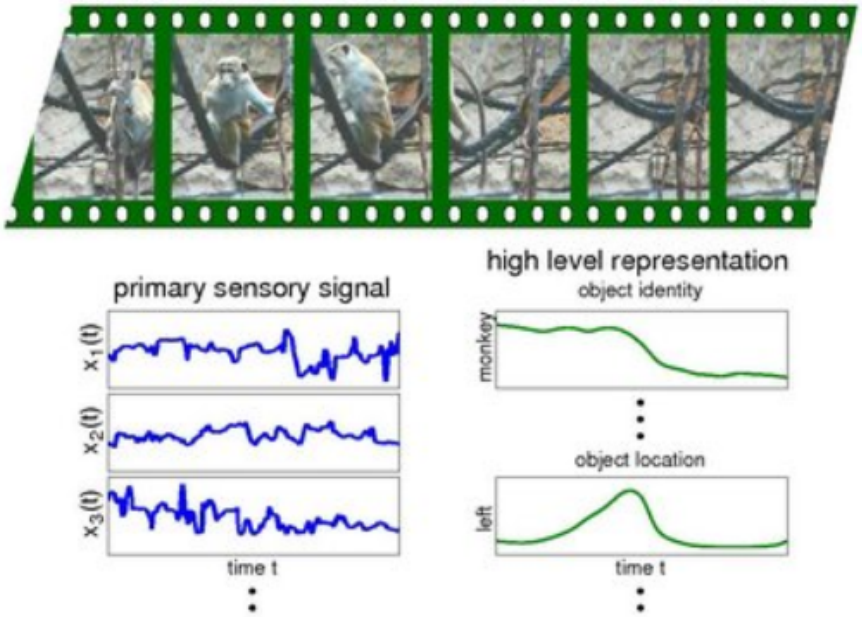
$$C = \sum_{i=1}^{N} ||x_i - W^t W x||^2$$



Linear Manifold Learning

Slow Feature Analysis (SFA) [WS02]

- Intuition: noisy measurements, stable perceptions

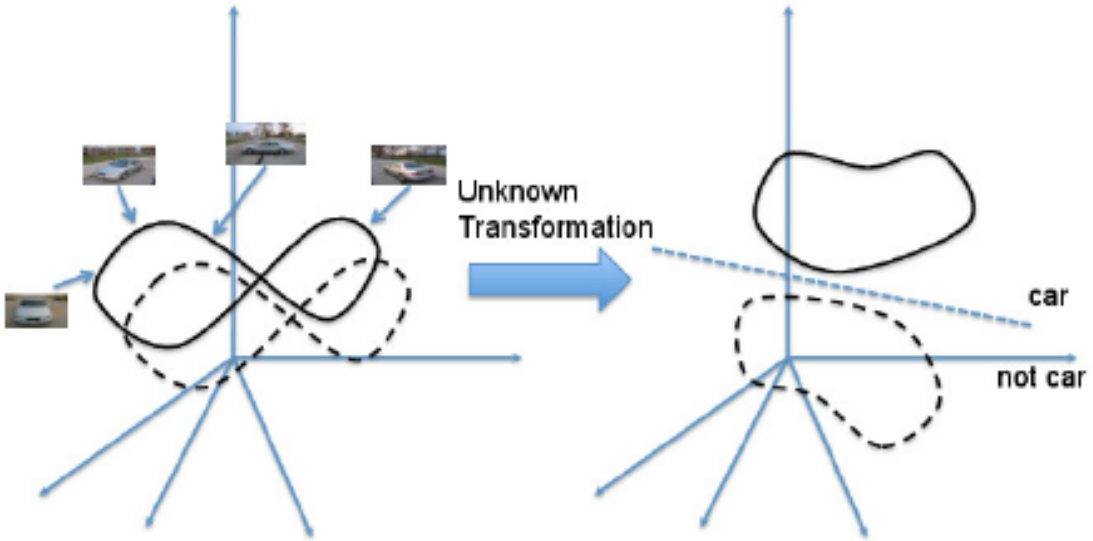


Source : http://www.scholarpedia.org/article/Slow_feature_analysis

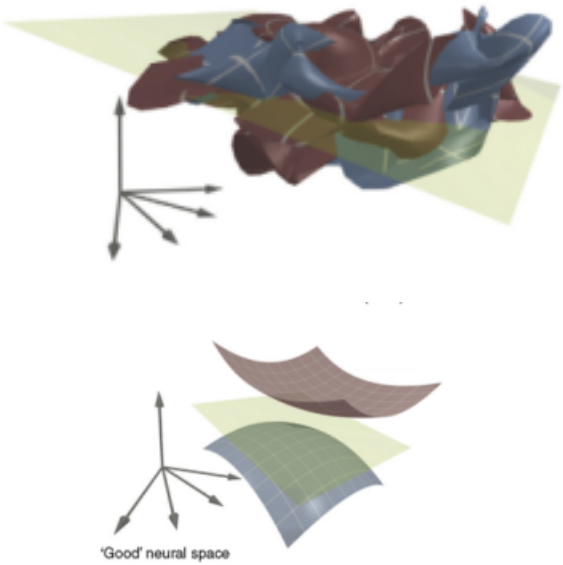
- Goal: learning "stable" representations from input signal
  - Related to the manifold untangling problem
- Mean: Extract low signal components, *i.e* "slow down" signal
- Ex application for images / videos

Slow Feature Analysis [WS02]

Manifold Untangling



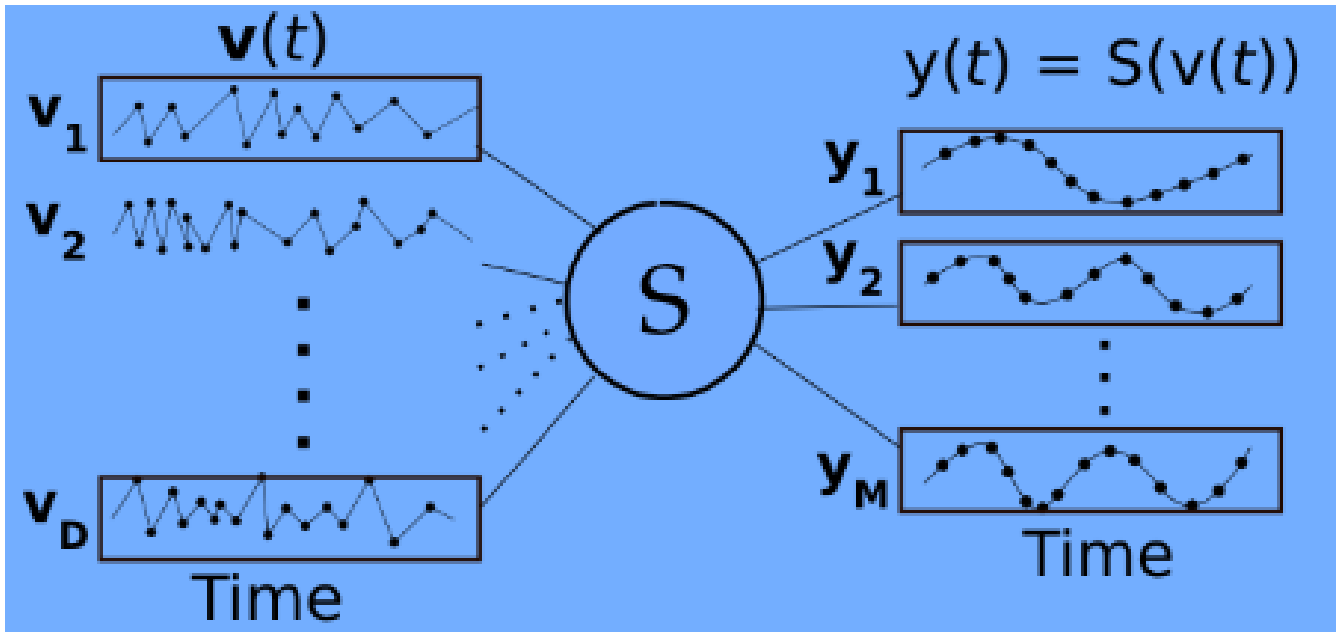
Credit: DiCarlo



Slow Feature Analysis

Formulation

- Still in the basis context (*e.g.* PCA), but different criterion
- Input : D-dimensional temporal signal $\mathrm{v}(t) = [v_1(t)v_2(t)...v_D(t)]^T$
- Output : M-dimensional temporal signal $\mathrm{y}(t) = [y_1(t)y_2(t)..y_M(t)]^T$
- $y_j(t) = S_j v(t)$, $\forall t$ et $\mathrm{S} \in \mathbb{R}^{D \times M}$



Slow Feature Analysis

Formulation

- $y_j(t) = S_j v(t)$, $\forall t$ and $S \in \mathbb{R}^{D \times M}$. Let us define:
  - $\langle y \rangle_t$ temporal average of y
  - $\dot{y}$ temporal derivative of y
- Objective function :

$$\min_{S_j} \langle \dot{y_j}^2 \rangle_t \tag{2}$$

Under constraints:

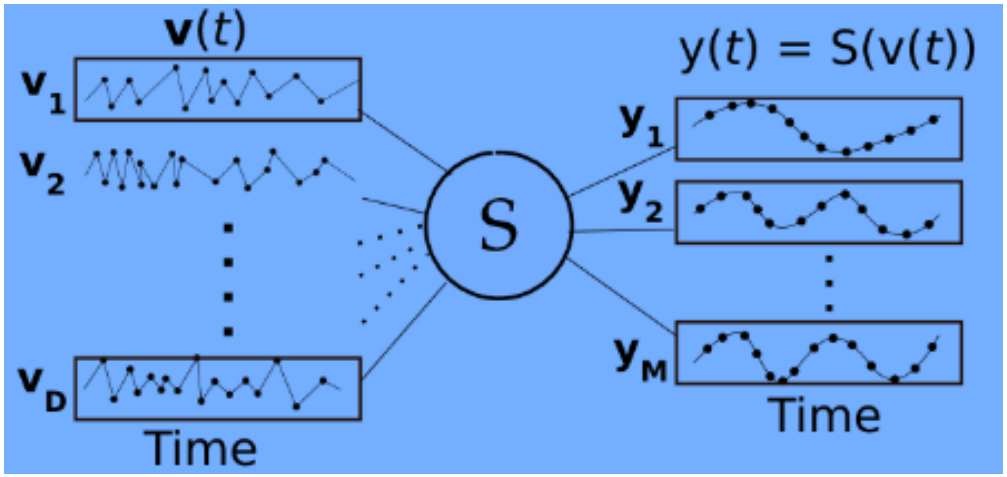
1. $\langle y_j \rangle_t = 0$ (zero mean)
2. $\langle y_j^2 \rangle_t = 1$ (unit variance)
3. $\forall j < j' : \langle y_j, y_{j'} \rangle_t = 0$ (decorrelation)

- Problem leading to solve:

$$\langle \dot{v}\dot{v}^T \rangle_t S_j = \lambda_j S_j \tag{3}$$

Slow Feature Analysis

Formulation

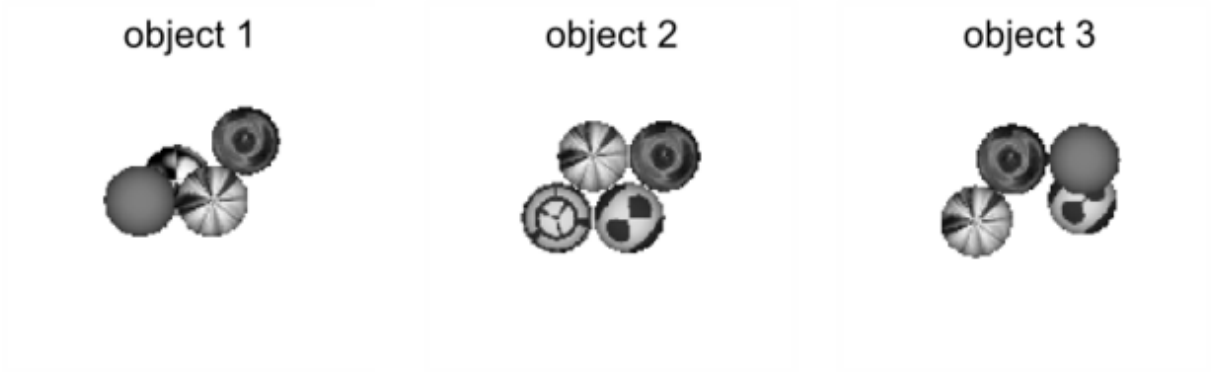


- Problem leading to solve: $\langle \dot{v}\dot{v}^T \rangle_t S_j = \lambda_j S_j$
- Diagonalizing $\dot{v}\dot{v}^T$
- Keep the M eigenvectors associated to the **smallest eigenvalues**

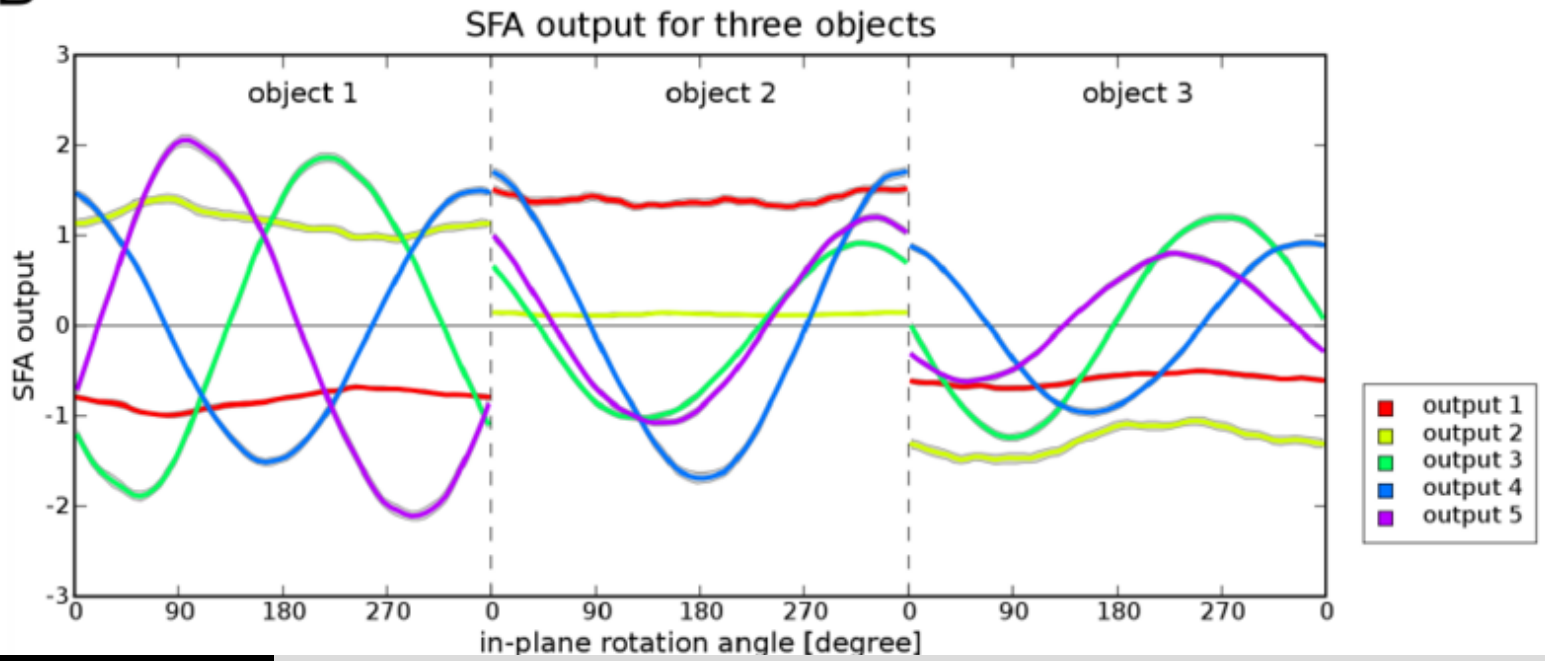
Slow Feature Analysis

Application: learning invariances

A

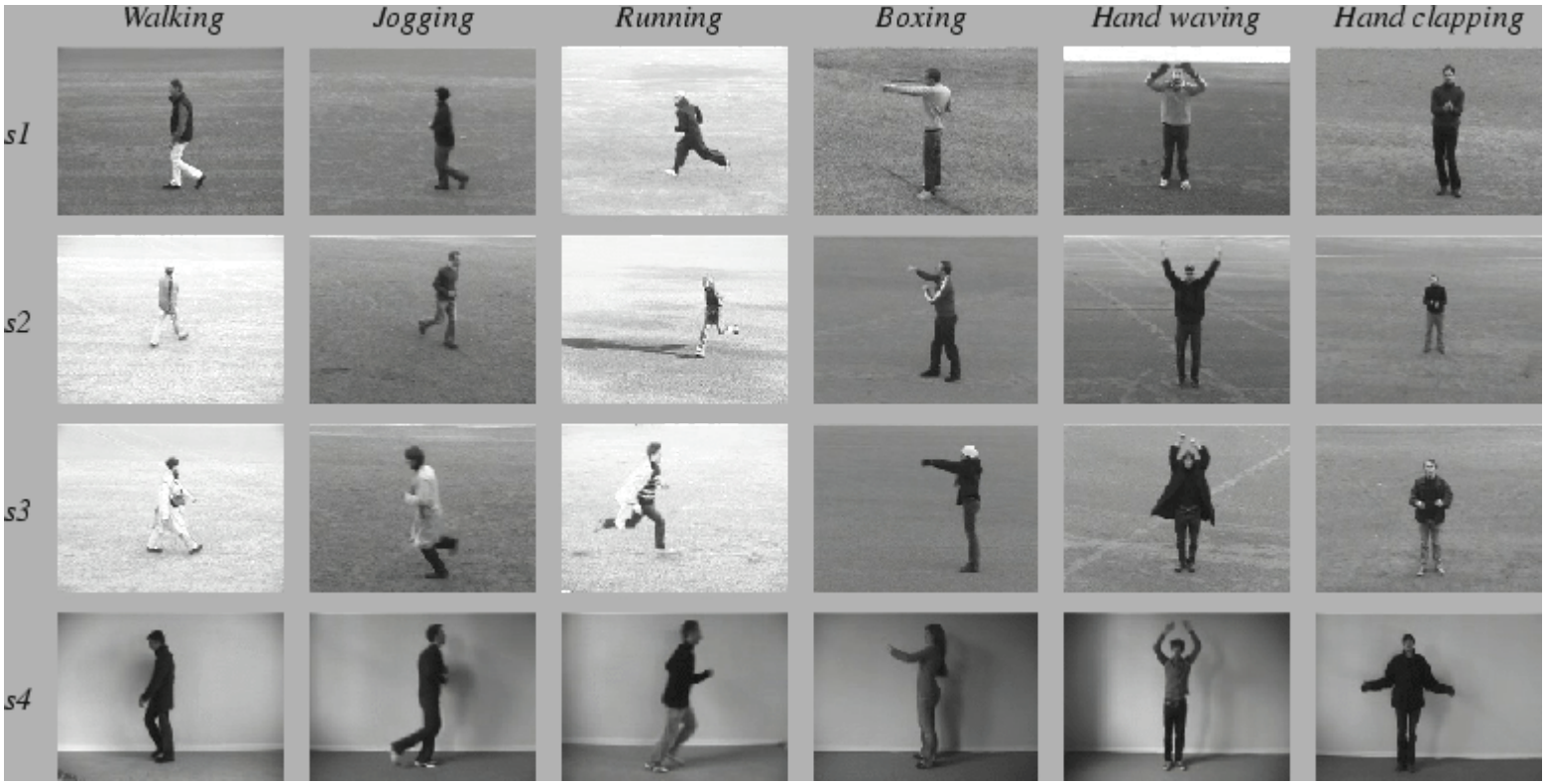


B



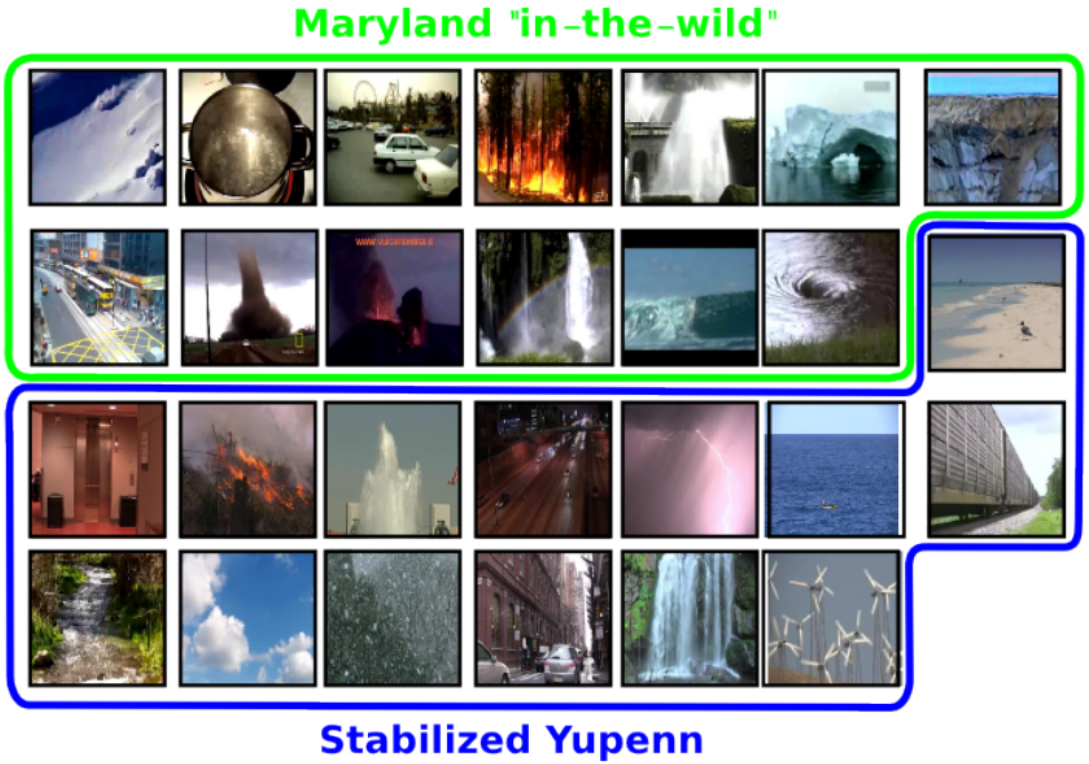
Slow Feature Analysis

Application: human action recognition



Slow Feature Analysis

Application: dynamic scene classification



- Input $v(t)$: image patch which
- SFA learns motion descriptors

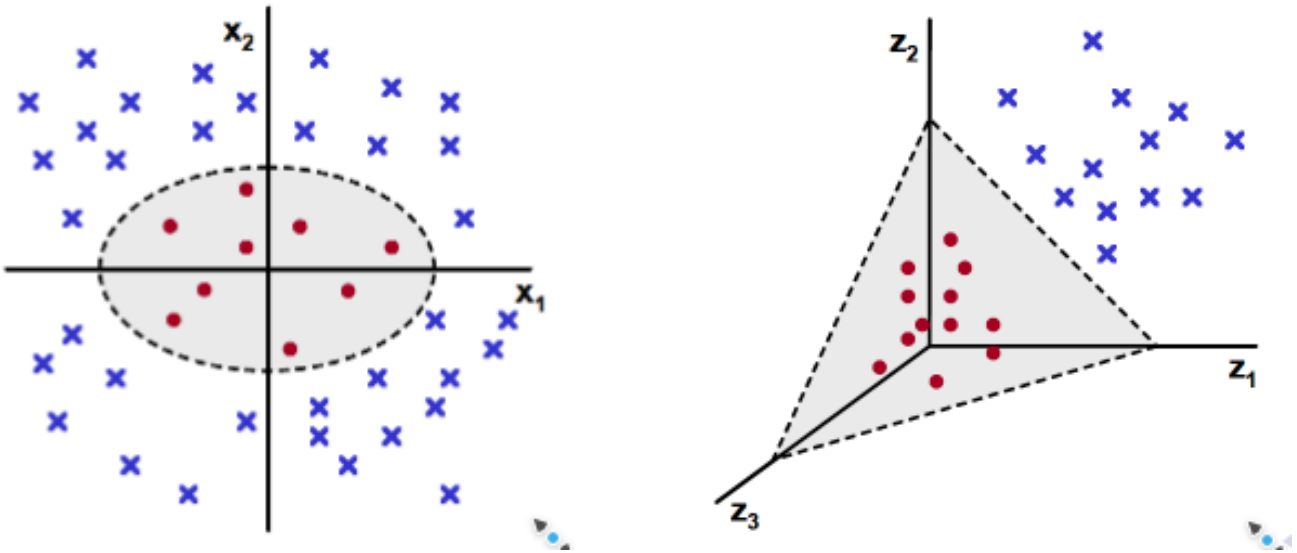
Non-Linear Manifold Learning

- Limit of linear methods: "good" data representations often require non-linear mapping
  1. Kernelization
  2. Visualization methods

Non-Linear Manifold Learning

Kernel methods [STC04]

- One option: define a non-linear injection function ϕ. Ex : $\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$



- Apply linear methods in the induced space

Kernelization

Kernel Trick

- Injection function $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^m$, often $m > p$
- Apply linear methods in the induced space $\mathbb{R}^m$
- Projection in the induced space: dot product $\langle \phi(x); \phi(y) \rangle$
- Mercer Kernel: if $k(x, y) = \langle \phi(x); \phi(y) \rangle$, then:
  - $\forall (x_1, ..., x_n)\ k(x_i, x_j)$ is a PSD matrix
  - Reverse: if $k(x, y)$ is PSD, $\exists \phi$ s.t. $k(x, y) = \langle \phi(x); \phi(y) \rangle$
- Mercer Kernel examples:
  - Linear : $k(x, y) = \langle x; y \rangle$
  - Polynomial : $k(x, y) = (1 + \langle x; y \rangle)^d$
  - RBF : Gaussian $k(x, y) = e^{\frac{-||x-y||^2}{\sigma^2}}$, Laplace $k(x, y) = 1/2e^{-\gamma|x-y|}$

Kernelization

Dual Formulation

- For many linear methods: learned vector $v_k = \sum_{i=1}^{N} \alpha_i \phi(x_i)$
- For $x \in \mathbb{R}^m$: $\langle \phi(x); v_k \rangle = \sum_{i=1}^{N} \alpha_i \langle \phi(x); \phi(x_i) \rangle = \sum_{i=1}^{N} \alpha_i k(x, x_i)$
- **Convenience:** Computing projection in induced space only using vectors in the input space $\mathbb{R}^n$
  - Especially useful for infinite induced spaces (*e.g.* Gaussian kernels)
- Optimization problem: rewritten wrt α parameters (see Kernel PCA)

Kernel PCA (KPCA)

Formulation

- PCA, recall: $\Sigma v = \lambda v$, $\lambda \in \mathbb{R}$, avec
  $\Sigma = \frac{1}{n} \sum_{i=1}^{n} (X_i - g)(X_i - g)^T$
- Non-linear mapping with ϕ: X_i and $v \in$ induced space
- Assuming centered data, $v = \sum_i \alpha_i \phi(X_i)$ since
  $v = \frac{1}{n\lambda} \sum_i \phi(X_i)\phi(X_i)^T v = \frac{1}{n\lambda} \sum_i \langle \phi(X_i); v \rangle \phi(X_i)$
- Substituting, KPCA formulation becomes:

$$K\alpha_k = n\lambda_k \alpha_k \tag{4}$$

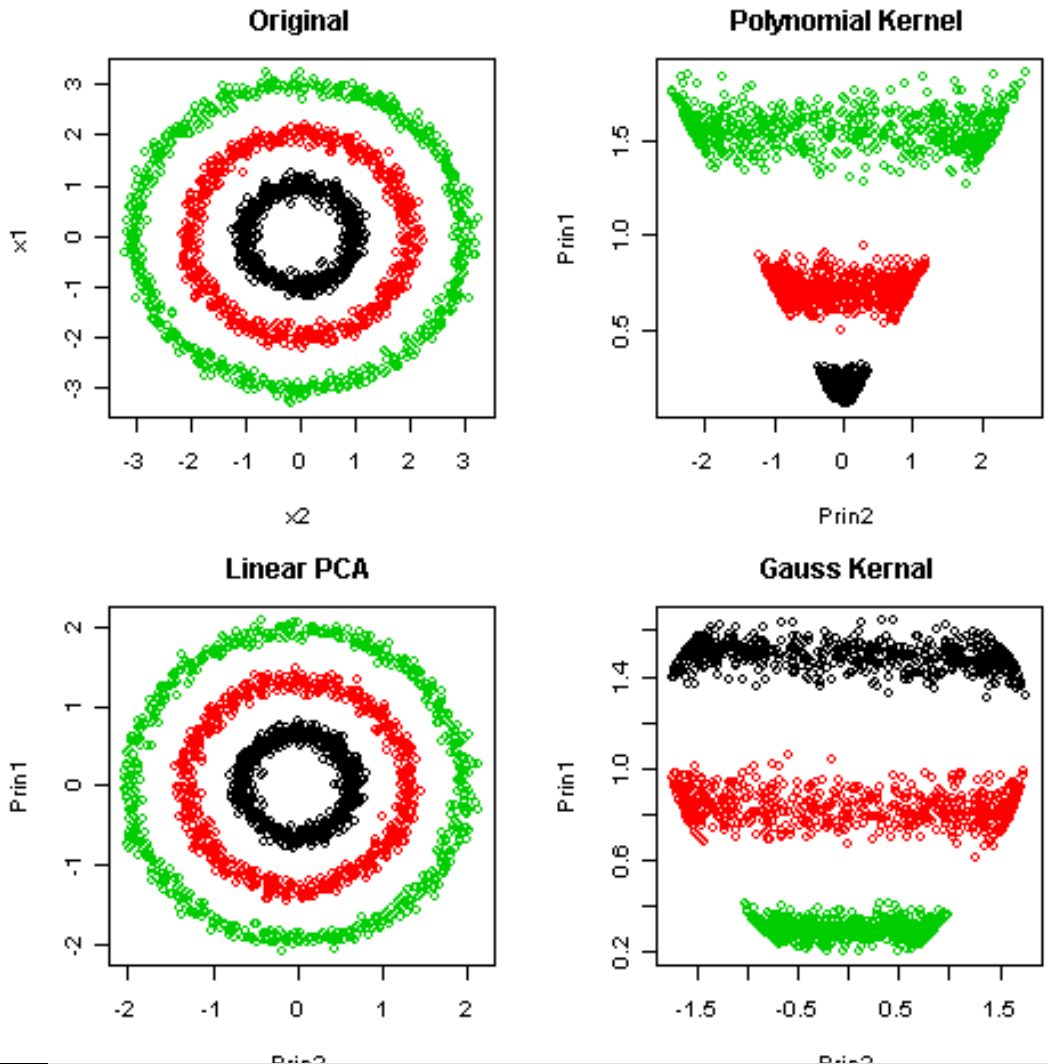
KPCA

Algorithm

- Diagonalizing Gram matrix $K = \{k(x_i, x_j)\}$, $(i, j) \in n \times n$
  - Each α_k vector defines v_k
- Normalization condition:
  $\langle v_k; v_k \rangle = 1 \Rightarrow \alpha_k^T K \alpha_k = 1 \Rightarrow \lambda_k n \alpha_k^T \alpha_k = 1$
  - Each α_k vector must be of norms $\frac{1}{\sqrt{\lambda_k n}}$
- Projecting new point $\phi(x) \in \mathbb{R}^m$ on v_k : $\sum_{i=1}^{n} \alpha_{k,i} k(x, x_i)$

KPCA

Illustration



Outline

Non-Linear Manifold Learning & Visualization

MDS (Multi-Dimensional Scaling) [CC00]

- Input: matrix D of euclidean distances between N points
- Distance D $\sim$ similarity B $= X^t X$
  - X size dxN, D and B size NxN
  - $d_{ij}^2 = d_{ii}^2 + d_{jj}^2 - 2bij$
- Goal: find points y in low dimensional space, which reflects these pairwise distances:

$$\phi(Y) = \sum_{i \neq j = 1,..,N} (d_{ij}^2 - ||y_i - y_j||)^2 \quad (5)$$

- Result: Minimizing Eq. 5 obtained by finding eigen decomposition of gram matrix B $= X^t X$

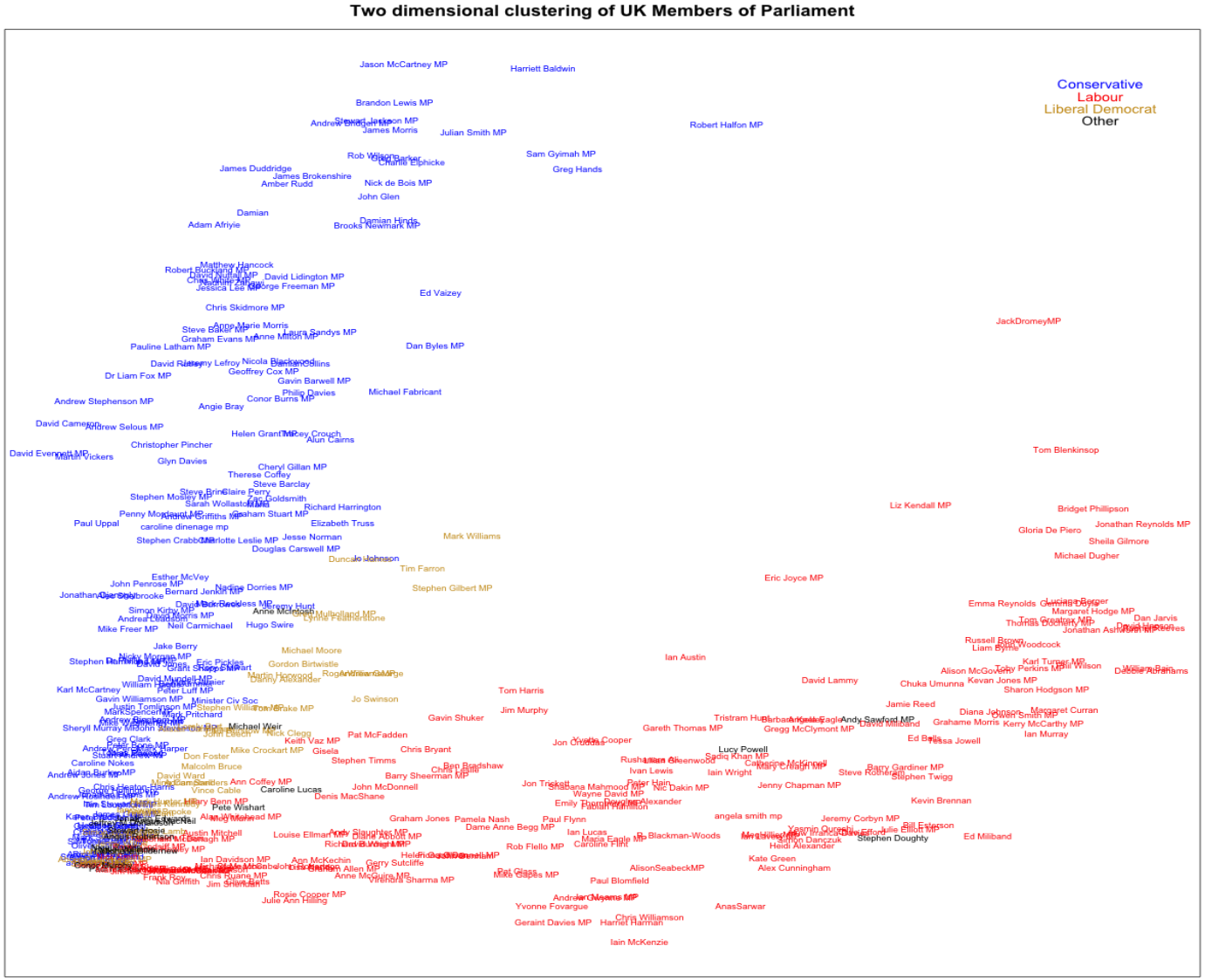
MDS (Multi-Dimensional Scaling)

classical Multidimensional Scaling
input: $D \in \mathbb{R}^{n \times n}(D_{ii} = 0, \ D_{ij} \geq 0), \ d \in \{1, \ldots, n\}$

1. Set $B := -\frac{1}{2}HDH$, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix.
2. Compute the spectral decomposition of B: $B = U\Lambda U^\top$.
3. Form Λ_+ by setting $[\Lambda_+]_{ij} := \max\{\Lambda_{ij}, 0\}$.
4. Set $X := U\Lambda_+^{1/2}$.
5. Return $[X]_{n \times d}$.

- MDS is actually closely connected to PCA (KPCA and Graam matrix)
- Actually linear dimension reduction from input space

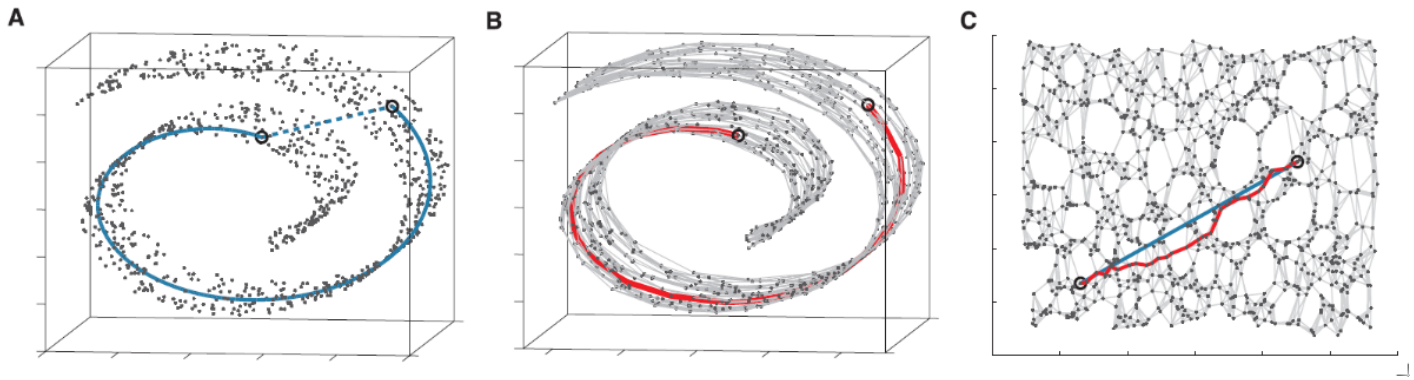
MDS (Multi-Dimensional Scaling)



Non-Linear Manifold Learning

ISOMAP [TdSL00]

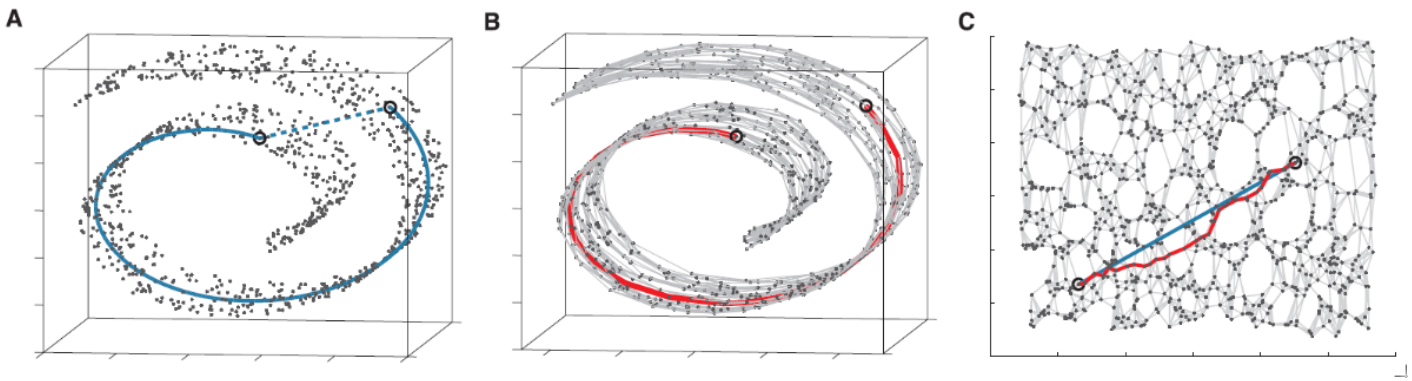
- Problem with MDS: euclidean distance when data lie in non-linear manifold
- ISOMAP solution: apply MDS using geodesic distance in the manifold !



Non-Linear Manifold Learning

ISOMAP

- ISOMAP: apply MDS using geodesic distance on the manifold
- Problem: manifold is unknown to us !
- Assumption: manifold locally smooth, euclidean distance good assumption of the geodesic distance for close points
  1. Generate an adjacency graph between points in input space
  2. Compute shortest path in graph between all pairs of points (Dijkstra's or Floyd's shortest-path) $\Rightarrow$ Approximation of the geodesic distance
  3. Apply MDS on the estimated geodesic distance matrix



Non-Linear Manifold Learning

Stochastic Neighbor Embedding (SNE) [HR03]

- Observation: large distance in high-dimensional space unreliable (curse of dimensionality)
- Trying to reflect large distance as in MDS / ISOMAP not a good thing
- Idea: Focus on reflecting small distances for close points
  - Motivation shared by Local Linear Embedding (LLE) [RS00]
- In input space (*e.g.* $\mathbb{R}^d$): $p_{i|j} = \frac{e^{-||x_i - x_j||^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-||x_i - x_k||^2/2\sigma_i^2}}$
- In output space (*e.g.* $\mathbb{R}^2$): $q_{i|j} = \frac{e^{-||y_i - y_j||^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-||y_i - y_k||^2/2\sigma_i^2}}$

Non-Linear Manifold Learning

Stochastic Neighbor Embedding (SNE)

- $P_i = \sum_j p_{j|i}$ conditional probability distribution over all other datapoints given x_i
- Objective function: Kullback-Leiber divergence between P_i and Q_i
- $C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \ log \frac{p_{j|i}}{q_{j|i}}$
- Solved by gradient descent:
  $\frac{\partial C}{\partial y_i} = 2 \sum_i (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$

Non-Linear Manifold Learning

t-SNE [vdMH08]

Two improvements from SNE:

1. Symmetric SNE
   - Defining P as the joint probabiliy over all pairs of points
   - Objective function becomes Kullback-Leiber divergence between P and Q:

$$C = \sum_i KL(P||Q) = \sum_i \sum_j p_{ij} \; log \frac{p_{ij}}{q_{ij}}$$

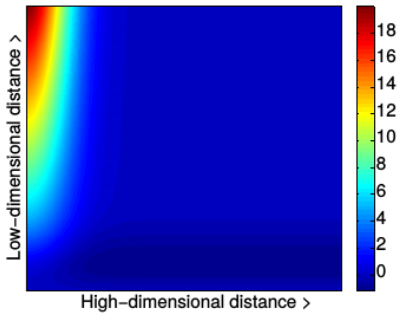
- $p_{ij} = \frac{e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2}}{\sum_{k \neq l} e^{-||\mathbf{x}_k - \mathbf{x}_l||^2/2\sigma^2}}$
- $q_{ij} = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2/2\sigma_i^2}}{\sum_{k \neq l} e^{-||\mathbf{y}_k - \mathbf{y}_l||^2/2\sigma^2}}$
- Gradient: $\frac{\partial C}{\partial y_i} = 4 \sum_i (p_{ij} - q_{ij})(\mathrm{y}_i - \mathrm{y}_j)$

Non-Linear Manifold Learning

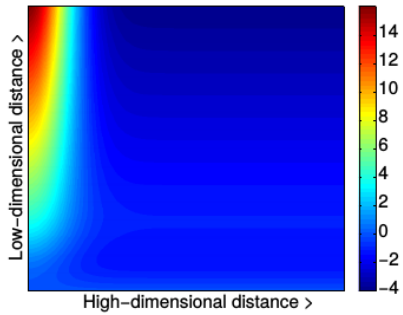
t-SNE

Two improvements from SNE:

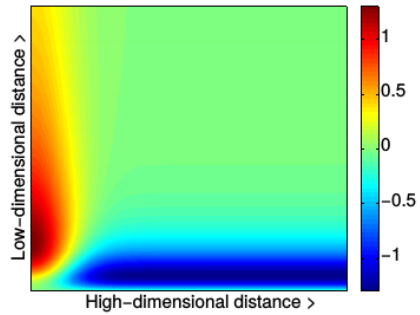
2. Using a Student t-distribution (with 1 dof, *i.e. Cauchy distribution*) for the output space:
   - $q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}}$
   - Motivation:
   - Gradient: $\frac{\partial C}{\partial y_i} = 4 \sum_i (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$



(a) Gradient of SNE.



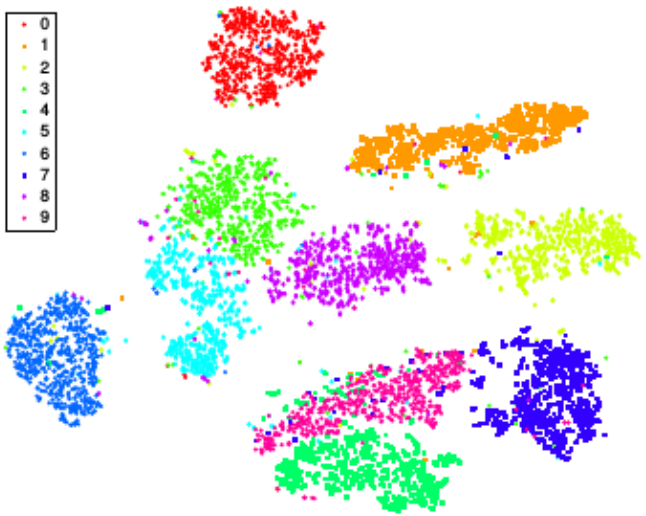
(b) Gradient of UNI-SNE.



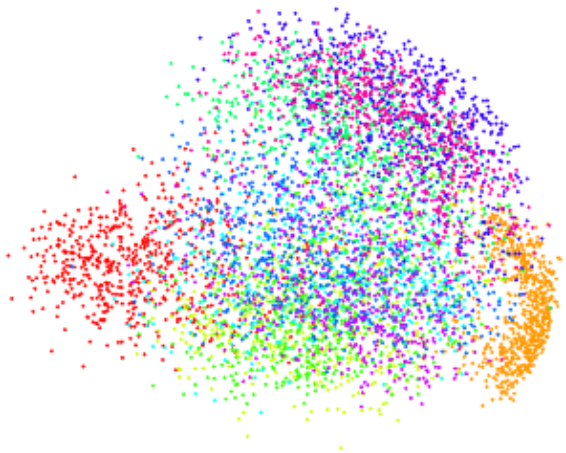
(c) Gradient of t-SNE.

t-SNE Results

Results from t-SNE paper in the MNIST dataset



(a) Visualization by t-SNE.



(b) Visualization by PCA.

Experiment it in practical session !

References I


Trevor F. Cox and M.A.A. Cox, *Multidimensional scaling, second edition*, 2 ed., Chapman and Hall/CRC, 2000.


Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, Cambridge, MA, USA, 2016, `http://www.deeplearningbook.org`.


Geoffrey E Hinton and Sam Roweis, *Stochastic neighbor embedding*, Advances in Neural Information Processing Systems (S. Becker, S. Thrun, and K. Obermayer, eds.), vol. 15, MIT Press, 2003.


I.T. Jolliffe, *Principal component analysis*, Springer Verlag, 1986.


Sam T. Roweis and Lawrence K. Saul, *Nonlinear Dimensionality Reduction by Locally Linear Embedding*, Science **290** (2000), no. 5500, 2323–2326.


John Shawe-Taylor and Nello Cristianini, *Kernel methods for pattern analysis*, Cambridge University Press, USA, 2004.


Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science **290** (2000), no. 5500, 2319.


Laurens van der Maaten and Geoffrey Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research **9** (2008), no. 86, 2579–2605.


Laurenz Wiskott and Terrence J. Sejnowski, *Slow feature analysis: Unsupervised learning of invariances*, Neural Comput. **14** (2002), no. 4, 715–770.