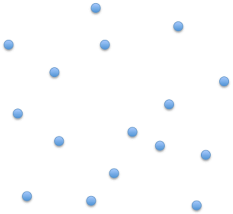
Data Analysis and Manifold Learning Lecture 1: Introduction to spectral and graph-based methods

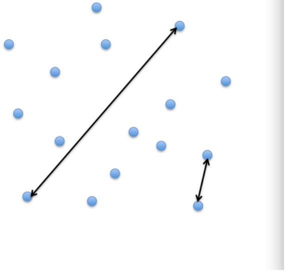
Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inrialpes.fr
http://perception.inrialpes.fr/

Introduction

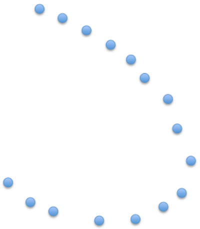
- I do not have a formal definition of **manifold learning**
- The general philosophy of what we want to study:
  - *Input:* An unorganized cloud of points in $\mathbb{R}^D$, where D (the dimension of the observation space) may be arbitrarily large.
  - *Output:* An intrinsic representation (parameterization) of the linear or non-linear subspace that best characterizes the data.
- Linear dimensionality reduction: find a subspace $\mathbb{R}^d \subset \mathbb{R}^D$ with $d < D$, possibly $d \ll D$.
- Non-linear dimensionality reduction: find a manifold $\mathcal{M} \subset \mathbb{R}^d$ and a **global** parameterization of that manifold.

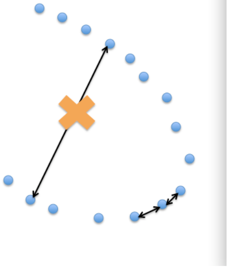
Metric spaces





Manifolds





Some definitions

- **Metric space:** one can compute the *distance* between any two points, e.g., Euclidean distances and Euclidean spaces.
- **Manifold:** every point has a neighborhood that is *homeomorphic* to an open subset of an Euclidean space.
- The dimension of a manifold is equal to the dimension of this Euclidean space
- One may say that a manifold is *locally* Euclidean while *globally* its structure is more complex.
- A *Riemannian manifold* is differentiable; the tangent space at each point on the manifold is an Euclidean space. The dimension of the tangent space is equal to the dimension of the manifold.

Discrete-data analysis and manifolds

- The theoretical properties of continous spaces/manifolds do not easily extend to point clouds.
- Ideally, one would like to deal with dense data that are uniformly sampled from a linear or a non-linear space.
- Of course, this is rarely the case and one is left with the difficult task of analysing sparse and/or non-uniform sampled data.
- The representation of choice is an *undirected graph*:
  - Linear case: it is a complete (fully connected) graph – easy case.
  - Non-linear case: it is a sparse (locally connected) graph – difficult case.

Methods for linear dimensionality reduction

Preamble: The space spanned by the data is linear and not the method itself!

- **Principal component analysis (PCA):** It represents the data using the directions of maximum variance; it boils down to compute the principal eigenvectors of the *covariance* matrix of the data.
- **Multidimensional scaling (MDS):** It is a distance preserving method. It first computes a matrix whose entries are the pairwise dot-products between the data points and then it represents the data using the principal vectors of this *Gram* matrix.
- It can be shown that PCA and MDS are somehow equivalent:
  - PCA needs the point coordinates
  - MDS only needs the pairwise dot-products

Methods for non-linear dimensionality reduction

- **Graph-based methods:** The first step is to build a sparse graph with nodes representing data points and edges representing neighborhood relations. The second step is to build a graph matrix. The third step is to compute the principal eigenvectors of this matrix.
- **Kernel-based methods:** They use a kernel function to evaluate the dot-product and to construct a Gram matrix. They may be seen as a generalization of MDS. They can also be refered to as graph-based kernel methods (more on this later).
- Many other methods can be found in the literature and in textbooks.

Graph-based and kernel methods

- Kernel PCA
- ISOMAP
- Laplacian eigenmaps (LE)
- Locally linear embedding (LLE)
- Hessian eigenmaps (HE)
- Diffusion maps
- Heat-kernel embedding (HKE)
- Maximum variance unfolding
- ...

Other methods

- Principal curves and surfaces
- Curvature component analysis (CCA)
- Manifold charting
- Local tangent-space alignment (LTSA)
- Unsupervised kernel regression
- ...

Where to read about manifold learning?

There are numerous classical and recent textbooks that address linear/non-linear dimensionality reduction. Manifold learning is a more recent term. There are several tens of papers in the machine learning and statistics literature: NIPS, JML, JMLR, NECO, PAMI, etc. These books are interesting:

- C. Bishop. Pattern Analysis and Machine Learning (chapter 12).
- J. Shawe-Taylor & N. Cristianini. Kernel Methods in Pattern Analysis (chapters 3, 5 & 6).
- J. A. Lee & M. Verleysen. Nonlinear Dimensionality Reduction.
- A. J. Izenman. Modern Multivariate Statistical Learning Techniques.

Mathematical notations

- Scalars: a, A, α, λ ...
- Vectors: $\boldsymbol{u}$ is a column vector while its transpose $\boldsymbol{u}^\top$ is a row vector:
$$\boldsymbol{u}^\top = (u_1 \dots u_i \dots u_n)$$
- $\mathbf{1}$ denotes a column vectors of 1's.
- Matrices: $\mathbf{U}$ and its transpose $\mathbf{U}^\top$
$$\mathbf{U} = [\boldsymbol{u}_1 \dots \boldsymbol{u}_n] = \left[\begin{array}{ccc} u_{11} & \dots & u_{n1} \\ u_{12} & \dots & u_{n2} \end{array} \right]$$
- $\mathbf{I}_n$ is the identity matrix of size $n \times n$.
- $\mathbf{I}_{m \times n}, m < n$ is a matrix formed with the top m rows of $\mathbf{I}_n$.

Dot-products, norms, distances, etc.

- Dot-product: $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \sum_k x_{ik} x_{jk} = \boldsymbol{x}_i^\top \boldsymbol{x}_j$
- Vector norm: $\|\boldsymbol{x}\|^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$
- Distance: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \langle \boldsymbol{x}_i, \boldsymbol{x}_i \rangle + \langle \boldsymbol{x}_j, \boldsymbol{x}_j \rangle - 2\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$
- Matrix norm: $\|\mathbf{A}\|_F^2 = \sum_i \sum_j A_{ij}^2 = \mathrm{tr}(\mathbf{A}^\top \mathbf{A})$
- This norm is known as the Frobenius norm and it is the most used matrix norm.

An Intuitive Introduction to PCA and MDS

- Let's start with a few more notations:
- The input (observation) space: $\mathbf{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_i \dots \boldsymbol{x}_n]$, $\boldsymbol{x}_i \in \mathbb{R}^D$
- The output (latent) space: $\mathbf{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_i \dots \boldsymbol{y}_n]$, $\boldsymbol{y}_i \in \mathbb{R}^d$
- **Projection:** $\mathbf{Y} = \mathbf{Q}^\top \mathbf{X}$ with $\mathbf{Q}^\top$ a $d \times D$ matrix.
- **Reconstruction:** $\mathbf{X} = \mathbf{Q}\mathbf{Y}$ with $\mathbf{Q}$ a $D \times d$ matrix.
- $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_d$
- *Reconstruction* will be useful for building a generative model – probabilistic PCA.

Computing the spread of the data

- We start with n scalars $x_1 \dots x_n$; the mean and the variance are given by:

$$\overline{x} = \frac{1}{n}\sum_i x_i \;\; \sigma_x = \frac{1}{n}\sum_i (x_i - \overline{x})^2 = \frac{1}{n}\sum_i x_i^2 - \overline{x}^2$$

- More generally, for the data set $\mathbf{X}$:
- The mean: $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_i \boldsymbol{x}_i$
- The covariance matrix is of dimension $D \times D$:

$$\mathbf{\Sigma}_X = \frac{1}{n}\sum_i (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\top = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \overline{\boldsymbol{x}}\,\overline{\boldsymbol{x}}^\top$$

The Gram matrix

- The Gram matrix: Consider n data points $\boldsymbol{x}_1 \dots \boldsymbol{x}_n$ with mean $\overline{\boldsymbol{x}}$. The (i,j) entry of the associated *centred* Gram matrix is the dot-product of two centred data points:

$$G_{ij} = \langle \boldsymbol{x}_i - \overline{\boldsymbol{x}}, \boldsymbol{x}_j - \overline{\boldsymbol{x}} \rangle$$

- The centred Gram matrix writes:

$$\mathbf{G} = \left(\mathbf{X} - \overline{\boldsymbol{x}}\mathbf{1}^\top\right)^\top \left(\mathbf{X} - \overline{\boldsymbol{x}}\mathbf{1}^\top\right) = \mathbf{J}\mathbf{X}^\top\mathbf{X}\mathbf{J}$$

  with: $\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. $\mathbf{G}$ is an $n \times n$ positive semi-definite symmetric matrix.

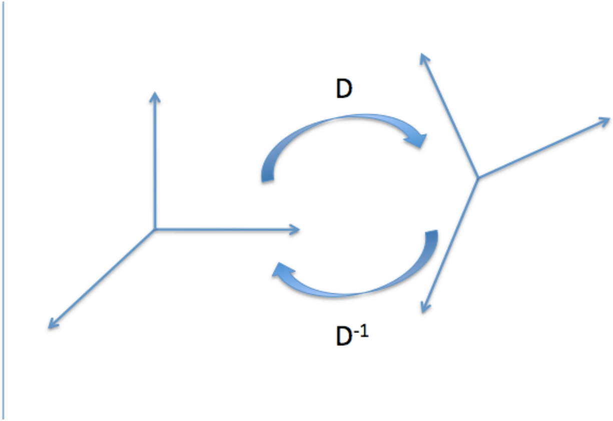
- Note that its dimension corresponds to the number of data points and not to the dimension of the underlying space.

The covariance and Gram matrices, side by side

- For the same **centred data set** we have:
- A $D \times D$ covariance matrix: $\mathbf{\Sigma}_X = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$
- A $n \times n$ Gram matrix: $\mathbf{G}_X = \mathbf{X}^\top\mathbf{X}$
- Let $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ be the singular value decomposition (SVD) of the data set.
- We obtain for our matrices

$$n\mathbf{\Sigma}_X = \mathbf{U}\mathbf{S}^2\mathbf{U}^\top \text{ and } \mathbf{G}_X = \mathbf{V}\mathbf{S}^2\mathbf{V}^\top$$

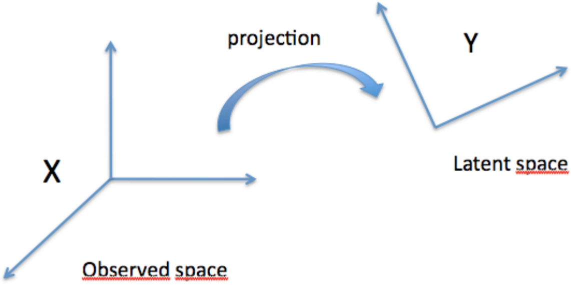
Changing the coordinate frame



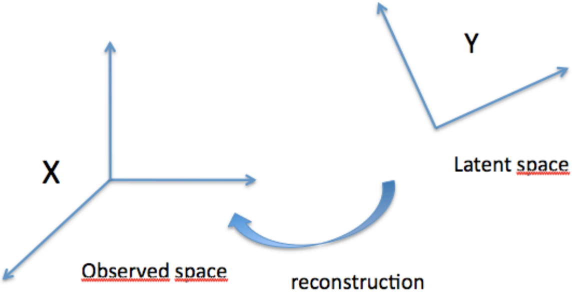
Orthogonal transformations

- If we ignore the translation between two frames, $\mathbf{D}$ is reduced to a $D \times D$ orthonormal matrix $\mathbf{R}$:
- $\mathbf{R}\mathbf{R}^\top = \mathbf{I}_D$
- The rows are mutually orthogonal, the columns are mutually orthogonal, the norm of each row- and column-vector is equal to 1.
- $\det(\mathbf{R}) = \pm 1$
- These matrices belong to O_D which is a notation for the *orthogonal* group of dimension D
- The *special orthogonal group* SO_D is characterized by $\det(\mathbf{R}) = +1$

Projecting the data



"Reconstructing" the data



Projection Versus Reconstruction

- Projection of $\mathbb{R}^D$ onto $\mathbb{R}^d$: Remove $D - d$ rows of $\mathbf{R}^\top$ to obtain a $d \times D$ row-orthogonal matrix $\mathbf{Q}^\top$.
- Reconstructin of $\mathbb{R}^D$ from $\mathbb{R}^d$: Remove $D - d$ columns of $\mathbf{R}$ to obtain a $D \times d$ column-orthogonal matrix $\mathbf{Q}$
- $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_d$ but $\mathbf{Q}\mathbf{Q}^\top \neq \mathbf{I}_D$!!
- Questions:
  - How to choose the low-dimensional reference frame?
  - How to choose d?
  - How to select d **principal** directions?
- Both PCA and MDS attempt to answer these questions.

Maximum Variance Formulation of PCA

- Let's project the data $\mathbf{X}$ onto a line along a unit vector $\boldsymbol{u}$. The variance along this line writes:

$$\begin{aligned}\sigma_u &= \frac{1}{n}\sum_i (\boldsymbol{u}^\top(\boldsymbol{x}_i - \overline{\boldsymbol{x}}))^2 \\ &= \boldsymbol{u}^\top \left(\frac{1}{n}\sum_i (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\top\right)\boldsymbol{u} \\ &= \boldsymbol{u}^\top \boldsymbol{\Sigma}_X \boldsymbol{u}\end{aligned}$$

- Maximizing the variance under the constraint that $\boldsymbol{u}$ is a unit vector:

$$\boldsymbol{u}^\star = \arg\max\left\{\boldsymbol{u}^\top\boldsymbol{\Sigma}_X\boldsymbol{u} + \lambda(1 - \boldsymbol{u}^\top\boldsymbol{u})\right\}$$

Maximum variance solution

- First note that the $D \times D$ covariance matrix is a symmetric semi-definite positive matrix. Therefore the quadratic form above is non-negative.
- Taking the derivative with respect to $\boldsymbol{u}$ and setting the derivatives equal to 0, yields: $\boldsymbol{\Sigma}_X \boldsymbol{u} = \lambda \boldsymbol{u}$
- Making use of the fact that $\boldsymbol{u}$ is a unit vector we obtain: $\sigma_u = \lambda$
- **Solution:** The *principal* or largest eigenvector–eigenvalue pair ($\boldsymbol{u}_{\max}$,$\lambda_{\max}$) of the covariance matrix.

Eigendecomposition of the Covariance Matrix

- Assume that the data are centred:

$$n\mathbf{\Sigma}_X = \mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

  Where $\mathbf{U}$ is a $D \times D$ orthogonal matrix and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues.

- If the data point lie on a lower dimensional space:

$$\text{rank}(\mathbf{X}) = d < D$$

  and

$$\mathbf{\Lambda} = [\lambda_1 \dots \lambda_d \;\; 0 \dots 0]$$
$$n\mathbf{\Sigma}_X = \widetilde{\mathbf{U}}\mathbf{\Lambda}_d\widetilde{\mathbf{U}}^\top$$

- $\widetilde{\mathbf{U}} = \mathbf{U}\mathbf{I}_{D\times d}$ is a $D \times d$ column-orthgonal matrix (reconstruction).
- $\widetilde{\mathbf{U}}^\top = \mathbf{I}_{D\times d}^\top\mathbf{U}^\top$ is a $d \times D$ row-orthgonal matrix (projection).

Data Projection on a Linear Subspace

- From $\mathbf{Y} = \mathbf{Q}^\top \mathbf{X}$ we have

$$\mathbf{Y}\mathbf{Y}^\top = \mathbf{Q}^\top \mathbf{X}\mathbf{X}^\top \mathbf{Q} = \mathbf{Q}^\top \widetilde{\mathbf{U}} \mathbf{\Lambda}_d \widetilde{\mathbf{U}}^\top \mathbf{Q}$$

1. The projected data has a diagonal covariance matrix: $\mathbf{Y}\mathbf{Y}^\top = \mathbf{\Lambda}_d$, by identification we obtain

$$\mathbf{Q}^\top = \widetilde{\mathbf{U}}^\top$$

2. The projected data has an identity covariance matrix, this is called *whitening the data*: $\mathbf{Y}\mathbf{Y}^\top = \mathbf{I}_d$

$$\mathbf{Q}^\top = \mathbf{\Lambda}_d^{-\frac{1}{2}} \widetilde{\mathbf{U}}^\top$$

- Projection of the data points onto principal direction $\boldsymbol{u}_i$:

$$(y_1 \dots y_n) = \underbrace{\lambda_i^{-1/2}}_{\text{whitening}} \boldsymbol{u}_i^\top (\boldsymbol{x}_1 \dots \boldsymbol{x}_n)$$

Illustration of PCA - the input data

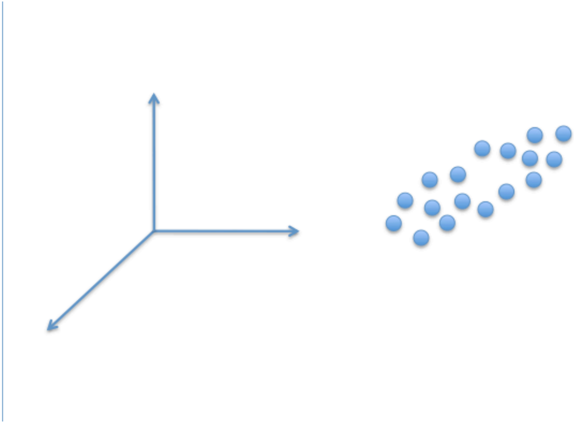


Illustration of PCA - centering the data

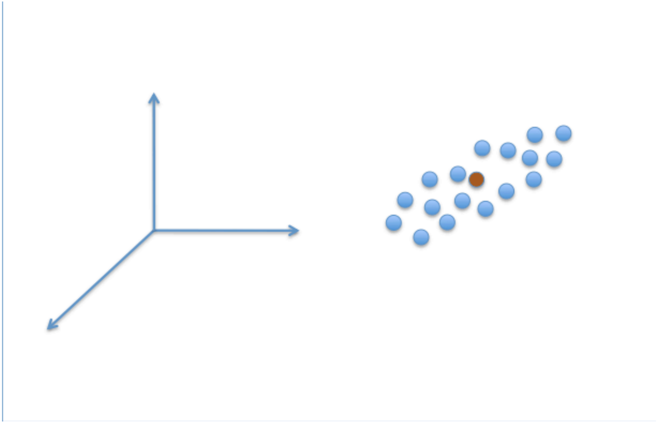
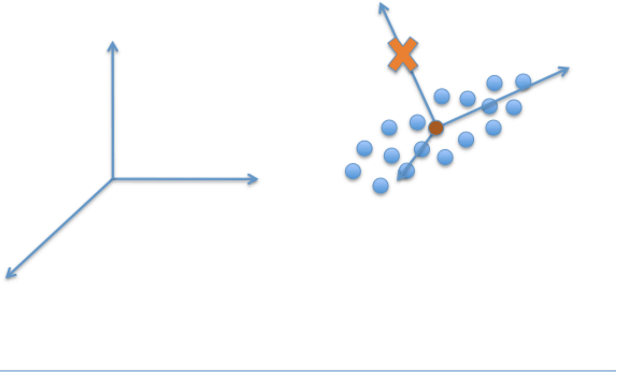


Illustration of PCA - principal eigenvectors of the data



Metric MDS

- MDS uses the Gram matrix (dot-products). The data points $\mathbf{X}$ are not explicitly required.
- Minimization criterion:

$$\min_{\mathbf{Y}} \|\mathbf{G}_X - \mathbf{Y}^\top \mathbf{Y}\|_F^2 \text{ with } \mathbf{G}_X = \widetilde{\mathbf{V}} \mathbf{\Lambda}_d \widetilde{\mathbf{V}}^\top$$

- Note that:

$$\|\mathbf{G}_X - \mathbf{Y}^\top \mathbf{Y}\|_F^2 = \text{tr}(\mathbf{G}_X^\top \mathbf{G}_X) + \text{tr}((\mathbf{Y}^\top \mathbf{Y})^2) - 2\text{tr}(\mathbf{G}_X \mathbf{Y}^\top \mathbf{Y})$$

- The criterion becomes: $\min_{\mathbf{Y}} \left\{ \text{tr}((\mathbf{Y}^\top \mathbf{Y})^2) - 2\text{tr}(\mathbf{G}_X \mathbf{Y}^\top \mathbf{Y}) \right\}$ and the solution and its covariance are:

$$\mathbf{Y} = \mathbf{\Lambda}_d^{\frac{1}{2}} \widetilde{\mathbf{V}}^\top, \; n\mathbf{\Sigma}_Y = \mathbf{Y}\mathbf{Y}^\top = \mathbf{\Lambda}_d$$

ISOMAP (non-metric MDS)

- This is the first example of a method that can deal with a data set that does not span a linear space.
- ISOMAP (Tenenbaum et al. 2000) is a method that does exactly this:
  1. Use the K nearest neighbor algorithm (KNN) to build a *sparse* graph over the data
  2. Compute the *geodesic distances* between **all** the vertex pairs
  3. Apply the MDS algorithm