Introduction into dimensionality reduction

- This lecture : linear methods for dimensionality reduction
    - Principal Component Analysis
    - Independent Component Analysis
    - Non-negative matrix factorization
    - Factor analysis
    - Multi-dimensional scaling
- Next lecture : non-linear methods aka manifold learning techniques

Dimensionality reduction formula

Encoder or ***Projector*** operator

$$R^p \rightarrow R^m,\ m << p$$

Data space

Some space*,**,***

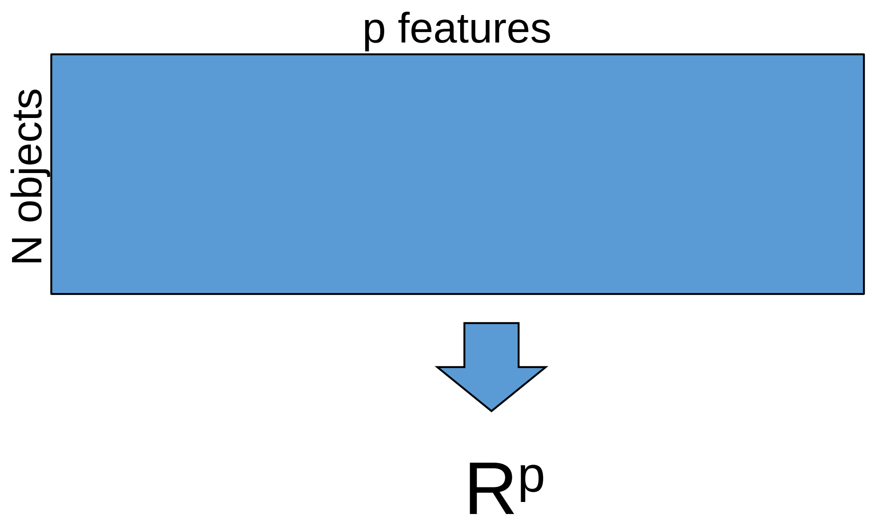
*can have complex geometry and topology

**does not have to be part of R^p

***simplest case: *m*-dimensional linear subspace in R^p

Reminder: modern data are frequently wide, containing more variables than objects

Modern ‘machine learning’



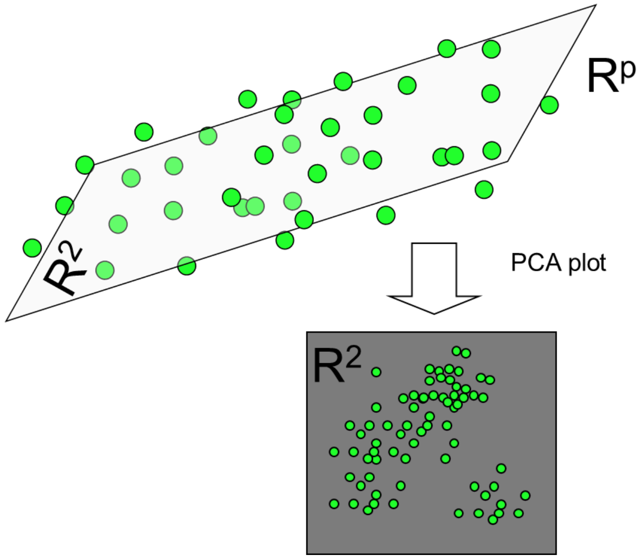
BIG DATA: N >> 1
WIDE DATA: p>>N
REAL-WORLD BIG DATA: p>>N>>1 (most frequently)

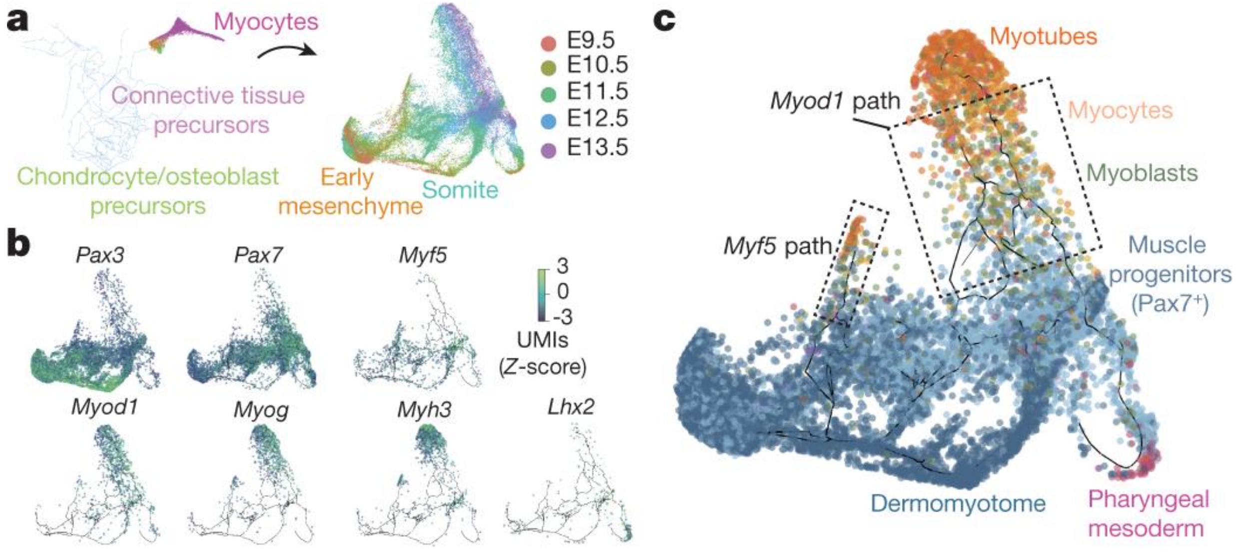
Why do we need to reduce dimension?

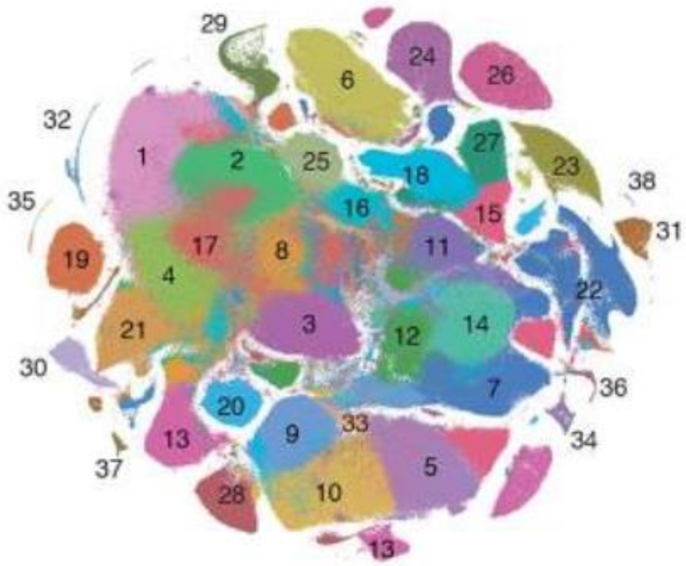
- **Converting wide data to the classical case N>>p**
- Improving signal/noise ratio for many other supervised or unsupervised methods
- Fighting with the curse of dimensionality
- Computational and memory tractability of data mining methods
- Visualizing the data
- Feature construction

Dimensionality reduction and data visualization

$$R^p \rightarrow R^m, m = 1,2,3$$







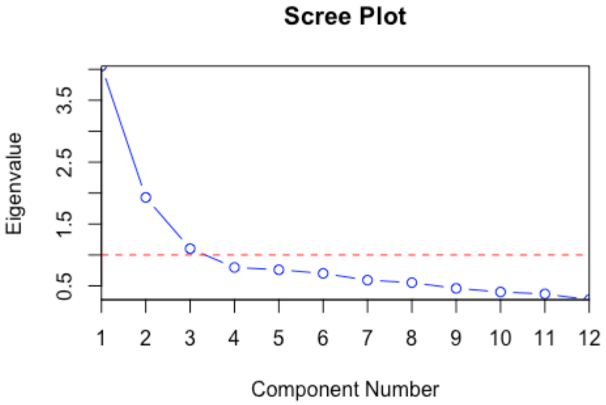
Ambient (total) and Intrinsic dimensionality of data

- *p* = ambient dimensionality (number of variables after data preprocessing)
- Intrinsic dimensionality (ID): ‘how many variables are needed to generate a good approximation of the data’
- *m* should be close to intrinsic dimensionality

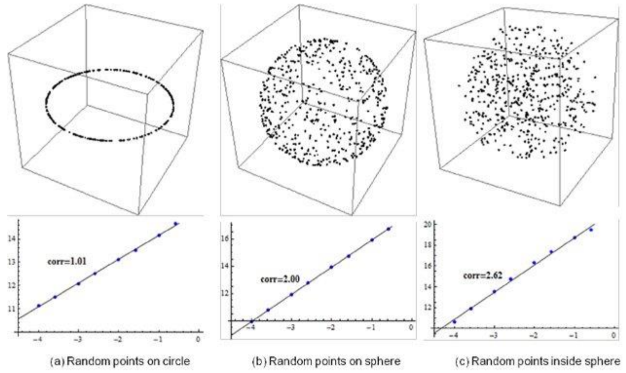
$$R^p \rightarrow R^m, m << p$$

Methods for intrinsic dimension estimation*

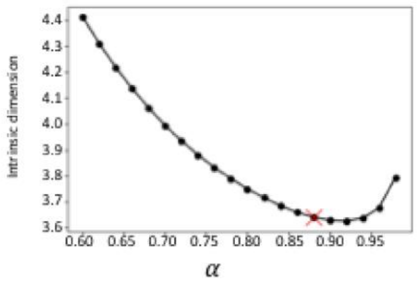
- Based on explained variance

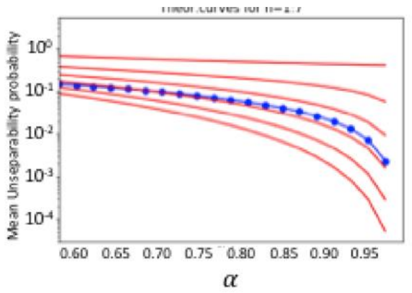


- Correlation dimension



- Based on quantifying concentration of measure





*Just an idea, more details later

Feature selection vs Feature construction (extraction)

- Feature selection : focus on the most informative variables, where ‘informative’ is with respect to the problem to be solved (e.g., supervised classification)

- Feature construction : create a set of fewer variables, each of which would be a function (linear or non-linear) of the initial variables

Projective vs Injective methods

Projective

ENCODE or PROJECT

R^p → R^m, $m \ll p$

Variant 1: The projector is known for any $\boldsymbol{y} \in \boldsymbol{R}^p$
Variant 2: The projector is know only for $\boldsymbol{y} \in \boldsymbol{X}$
(in this case one can first project a new data point into the nearest point of $\boldsymbol{X}$)

Injective*

ENCODE or PROJECT

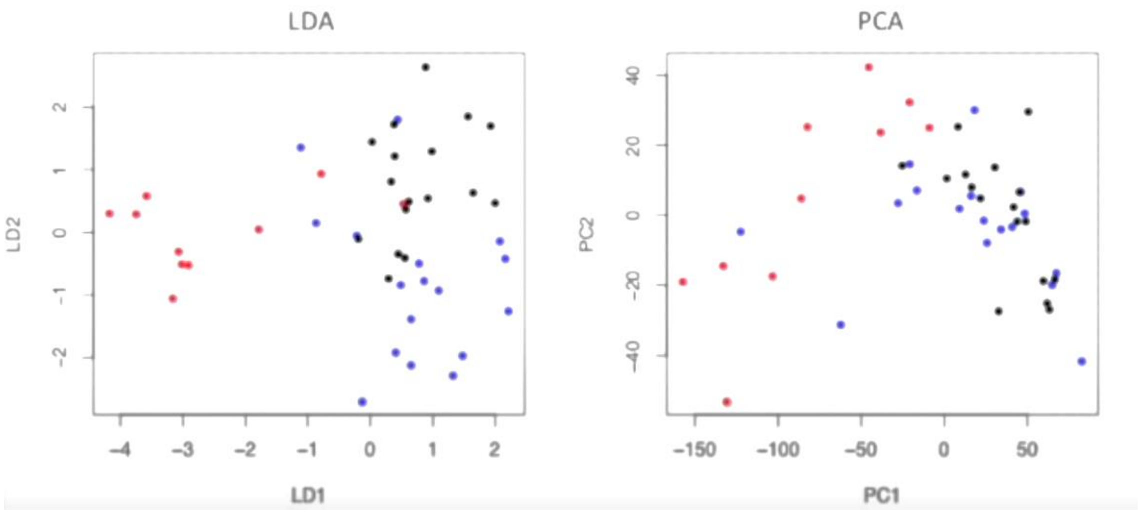
R^p ⇄ R^m, $m \ll p$

DECODE or INJECT

*we know where to find ANY point from R^m in R^p

Supervised approaches to dimensionality reduction

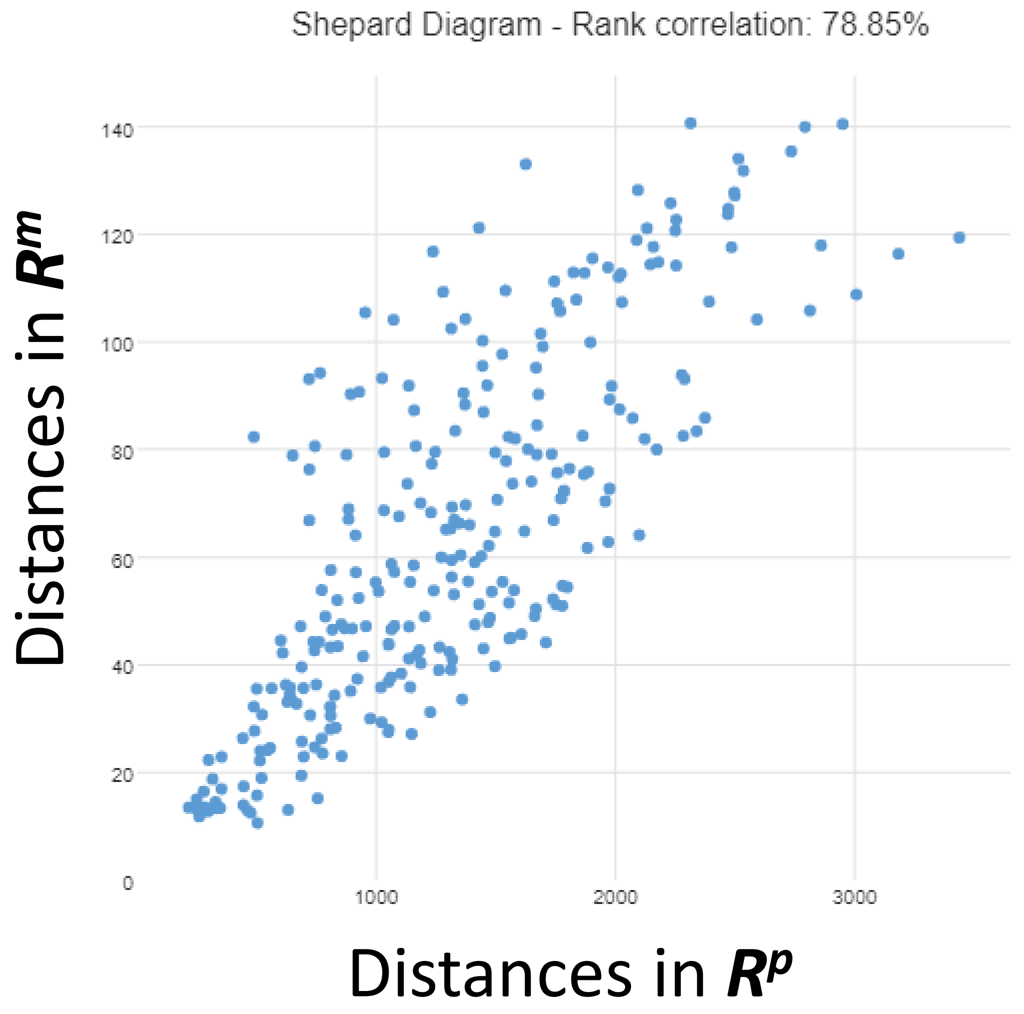
- Classical example: Linear Discriminant Analysis (LDA)



- Supervised Principal Component Analysis (Supervised PCA)
- Partial Least Squares (PLS)
- Many others…

Shepard Diagram

the simplest measure of quality of dimension reduction



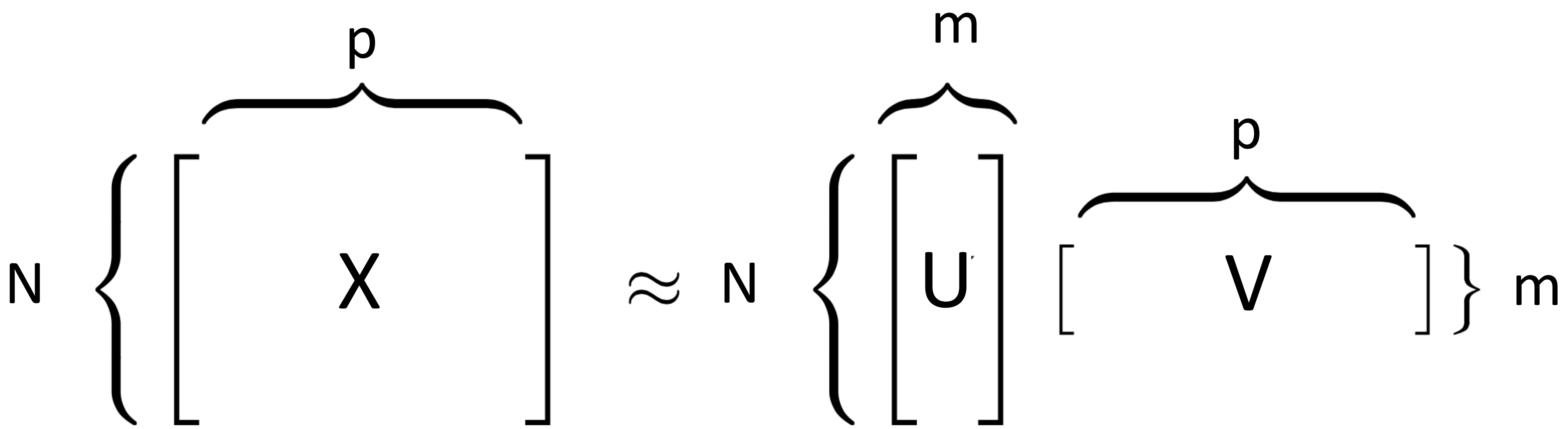
Remark 1. Not all dimension reduction methods aims at reproducing ALL distances

Remark 2. Simple Shepard Diagram contains many redundant comparisons

Choice of languages: matrix vs geometrical vs probabilistic

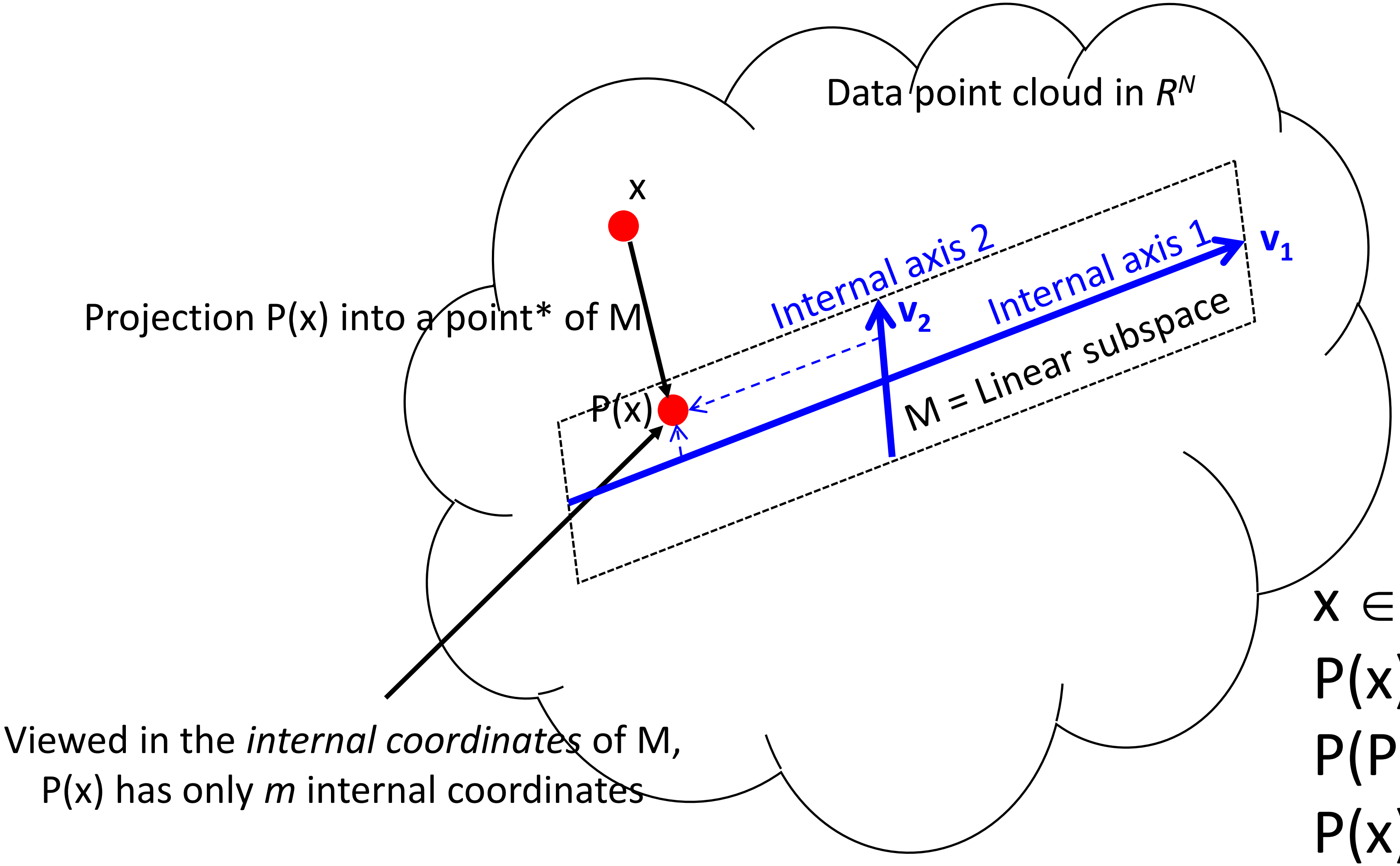
- Singular value decomposition = Principal Component Analysis
- Low-rank matrix factorization
- Geometrical : axes, basis, vectors, projection
- Probabilistic: log-likelihood, distribution, factor
- *These languages (matrix vs geometry vs probabilistic) can be easily mutually translated in linear case*

Low rank matrix factorization $X = UV$



Each column in *U* and row in *V* (together) are called a *component*
Elements of *U* can be used for further analysis as a new data matrix
Elements of *V* can be used for *explaining components*

Simplest geometrical image



$x \in R^N$

$P(x) \in R^N$

$P(P(x)) = P(x)$

$P(x) = u^1\mathbf{v}_1 + \ldots + u^m\mathbf{v}_m$

*for example, into the closest point, P(x) = arg min || y - x ||

What you should ask about a *dimensionality reduction* method

Base level

- Input information (data table, distance table, KNN-graph, …)
- Computational complexity (time and memory requirements), scalability for big data ($O(p^l m^s N^k)$, p – number of dimensions, N – number of data points, *m* – number of intrinsic dimensions)
- *What are the general assumption on the data distribution?*
- *What distances are more faithfully represented: short or long?*
- *How many intrinsic dimensions is possible to compute?*
- *What does it optimize?*
- Key parameters and requirements for domain knowledge to determine

Technicality

- Possibility to work with various distance metrics
- *Projective or injective?*
- *Can we map (reduce) the data not participating in the training?*
- Sensitivity to noise and outliers
- Ability to work in high-dimensional spaces

Flexibility

- Ability for online learning
- Incorporation of user-specified constraints
- Interpretability and usability