EE613
Machine Learning for Engineers

Dimensionality reduction

jean-marc odobez

Overview

- Introduction

- Principle Component Analysis (PCA)
  - Global - compression

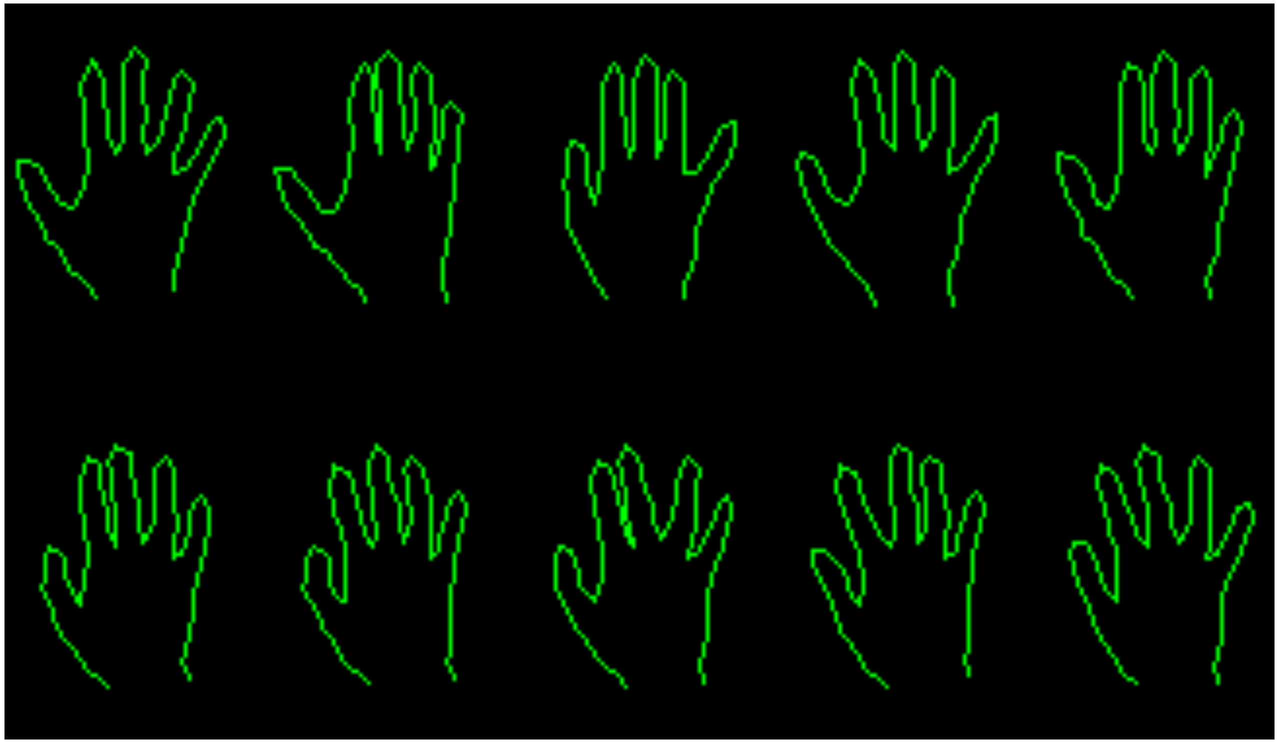
- Probabilistic PCA
  - Global, probabilistic

- t-distributed Stochastic Neighbor Embedding (t-SNE)
  - Local - visualization

Introduction - motivation

Introduction - motivation

Shape analysis



- Flat hand of one person
  - Delineated with 80 points

    => shape representation: position of these points

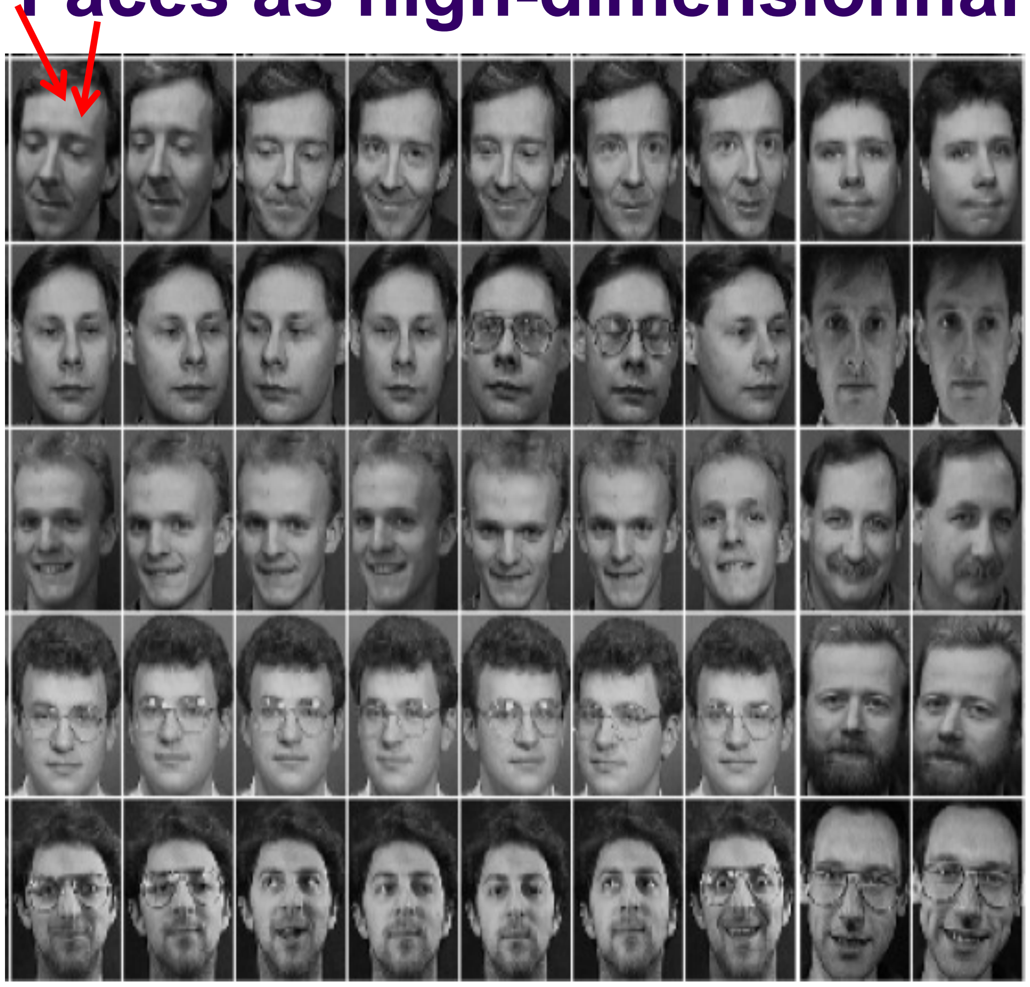
    => original space dimension D=160

  - Assume allowed mobility = independent rotation of each finger

  => manifold of flat hand intrinsic dimension around M=5

$$\mathbf{x} = [x_1, y_1, x_2, y_2, \ldots x_{80}, y_{80}]^T$$

Introduction
Faces as high-dimensionnal data point



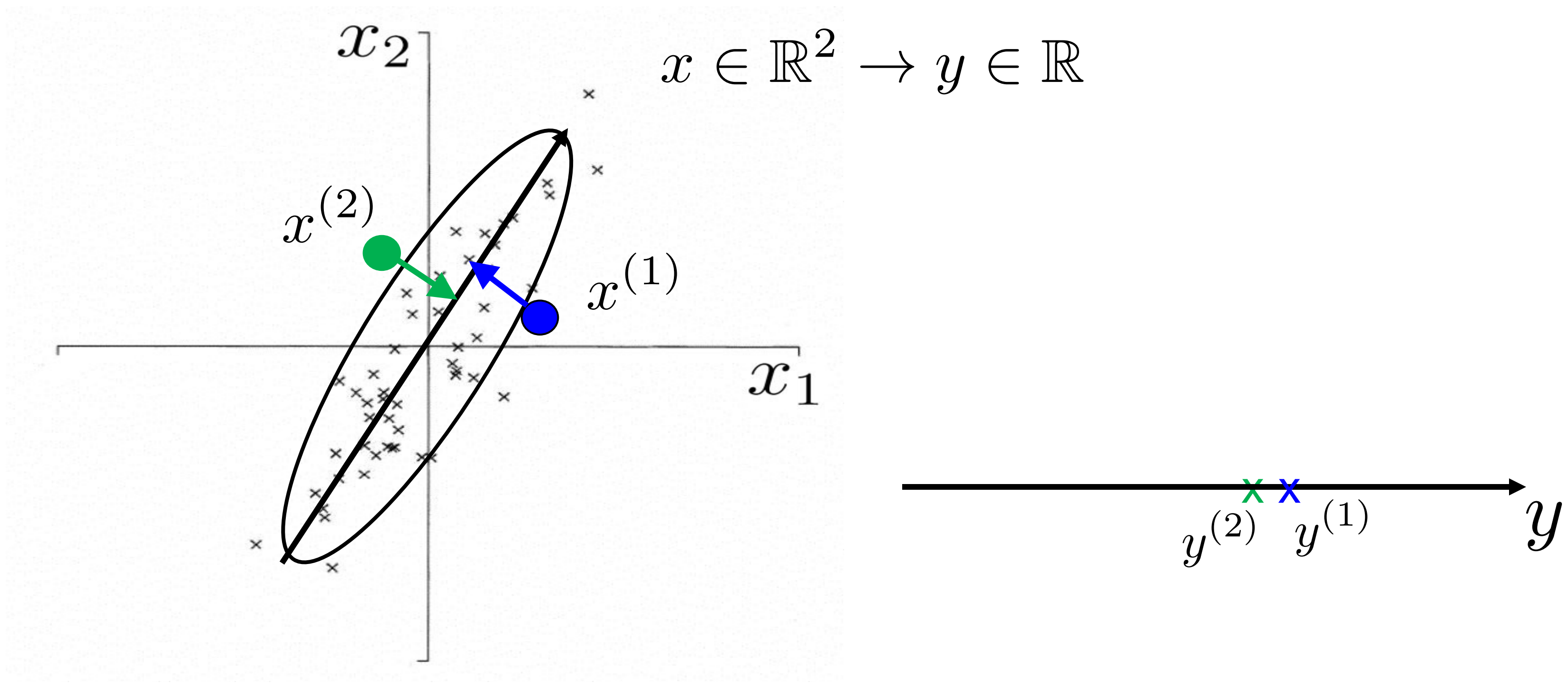
Data components (here pixels) are correlated

- Each of this face
  100x100 patches
  = 10 000 pixels
  D = 10 000 dimensions

- Database of 2000 faces
  => 1 point per 5 dimensions !
  => data is very sparse
  (To represent probability density, need number of training samples > D)

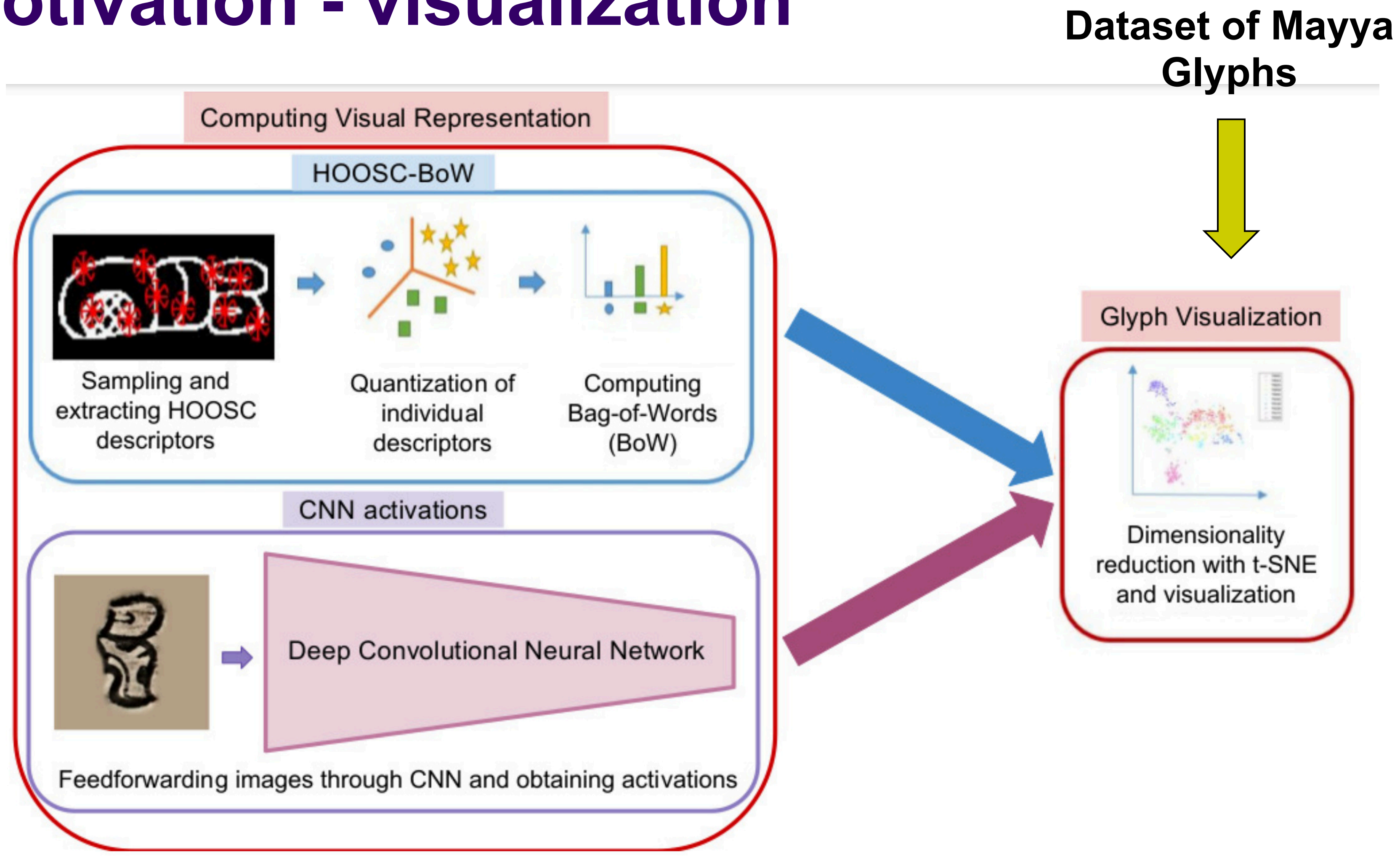
- But the face space has actually a much smaller dimension

Motivation – data compression



- Represent any data point in original space by a vector with less components
- Face case
  - D = 10 000 => 50/100 dimensions
    - Less storage, faster comparisons
    - Easier to compute probability distribution over datapoints, similarities

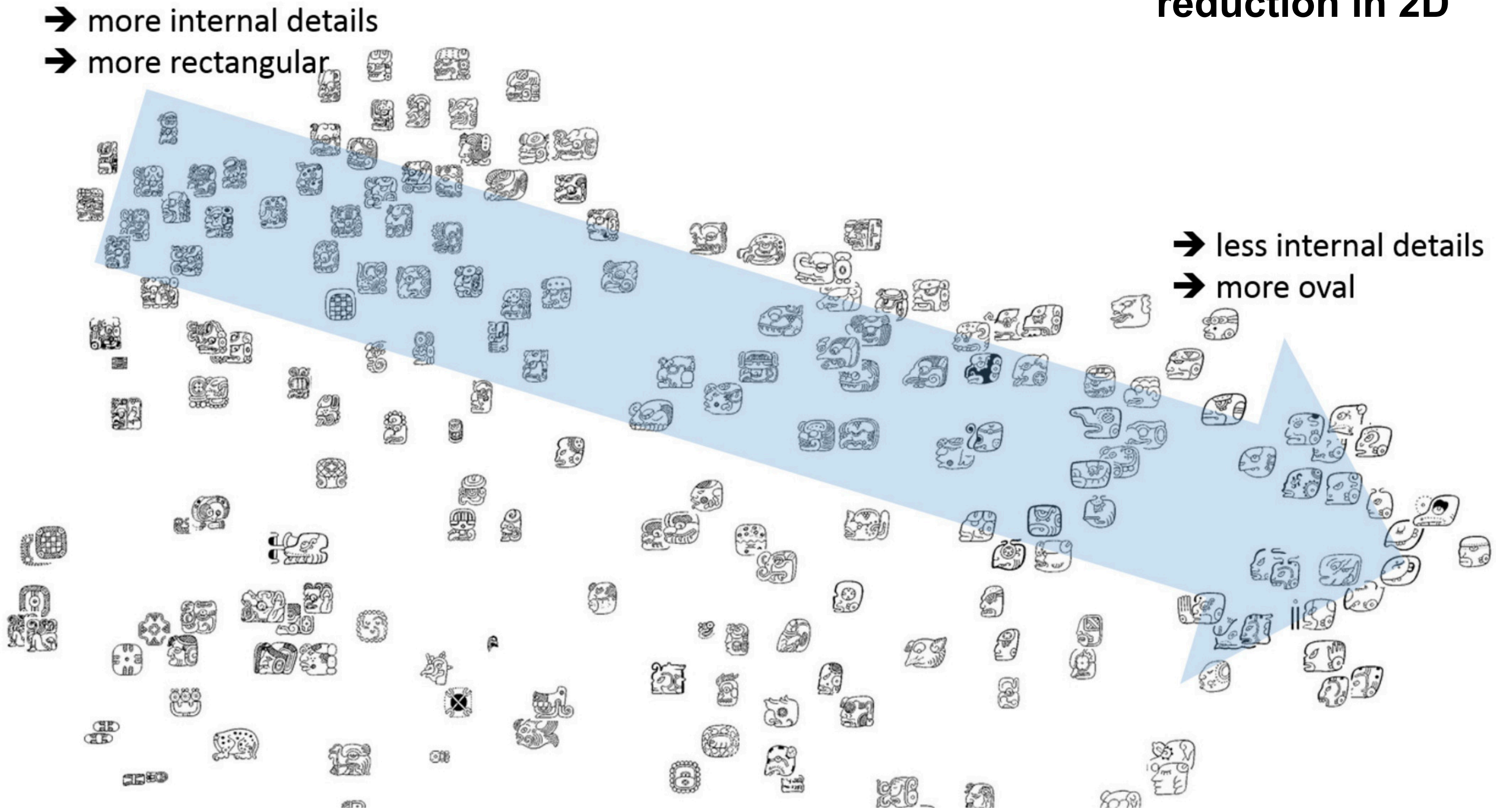
Motivation - visualization


Dataset of Mayya Glyphs
Computing Visual Representation
HOOSC-BoW
Sampling and extracting HOOSC descriptors
Quantization of individual descriptors
Computing Bag-of-Words (BoW)
CNN activations
Deep Convolutional Neural Network
Feedforwarding images through CNN and obtaining activations
Glyph Visualization
Dimensionality reduction with t-SNE and visualization

- Allow data interpretability: what is the ‘structure’ in the high dimensional space

Motivation - visualization

Dimensionality reduction in 2D



- Allow data interpretability: what is the 'structure' in the high dimensional space

Dimension reduction

Main Idea of PCA (and manifold reduction techniques)

- Many real worl datasets
  - Data point lie in large dimensional spaces (manifolds)
  - Data point components are highly correlated (globally, locally)
  - Low linear dimensional subspace captures most data variability

- Dimension reduction technique
  - Learn an explicit or implicit mapping (from data)

$$x \in E = \mathbb{R}^D \rightarrow f(x) = y \in F = \mathbb{R}^d$$

where D > d and the space F keeps some properties of the original space

Principal component analysis (PCA)

Outline PCA

- Introduction

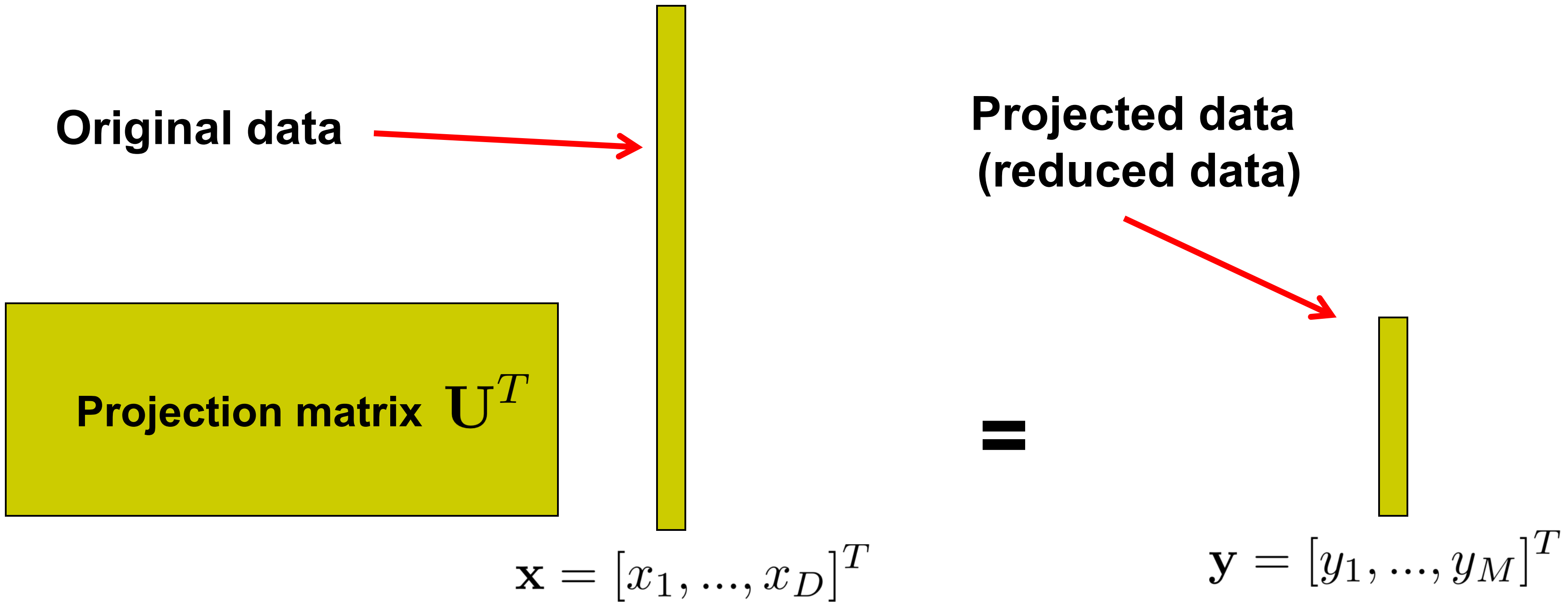
- Principal Component Analysis (PCA) Principles
  - Variance Maximization
  - Reconstruction error minimization

- Examples

- Discussion

PCA principles

- Goal: project data from space of dimension D into lower dimension space (dimension M<D)
- Linear subspace => linear projection


Original data
Projected data (reduced data)
Projection matrix $\mathbf{U}^T$
=
$\mathbf{x} = [x_1, ..., x_D]^T$
$\mathbf{y} = [y_1, ..., y_M]^T$

$$\mathbf{x} \mapsto \mathbf{y} = \mathbf{U}^T \mathbf{x} \text{ with } \mathbf{U} \in R^{D \times M}$$

$$y_i = \mathbf{u}_i^T \mathbf{x}$$

PCA principles

- Data driven approach

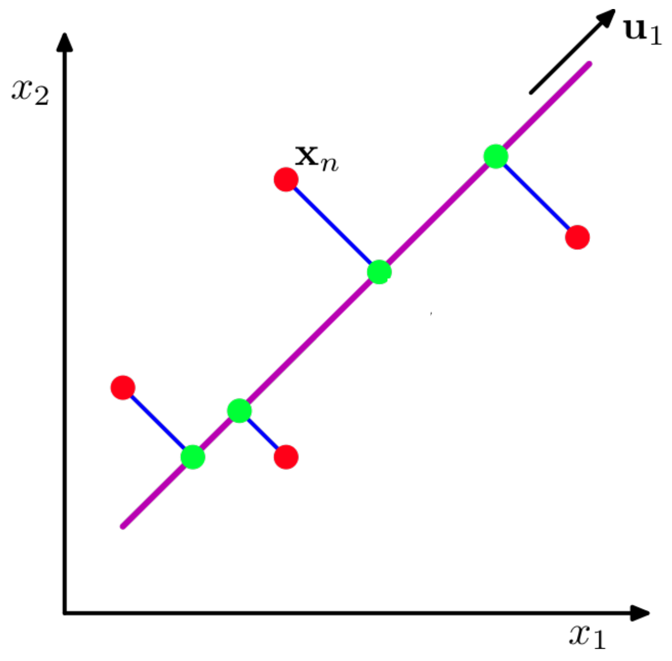
  i.e. learns a transformation from data

  => dataset $\{\mathbf{x}_1, ..., \mathbf{x}_N\}, \mathbf{x}_i \in R^D$

- Projections

$$\mathbf{y}_n = \mathbf{U}^T \mathbf{x}_n \text{ with } y_{in} = \mathbf{u}_i^T \mathbf{x}_n$$

$$y_{1n} = \mathbf{u}_1^T \mathbf{x}_n$$

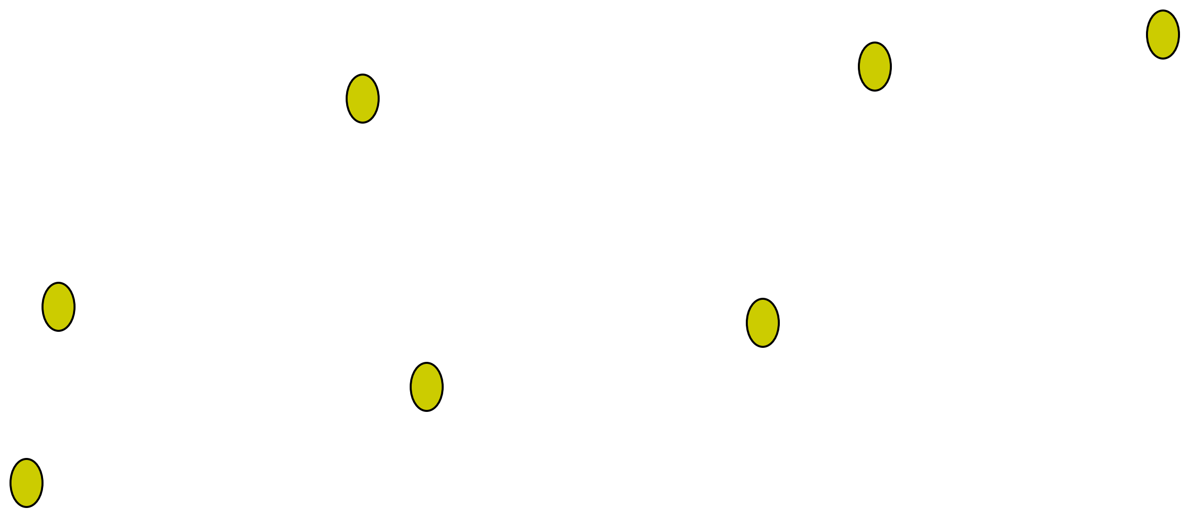


- Two views of PCA. Given the dataset, find the principle components (**u** vectors) retaining most of the information of the original data
  - Capture most data variability of original points

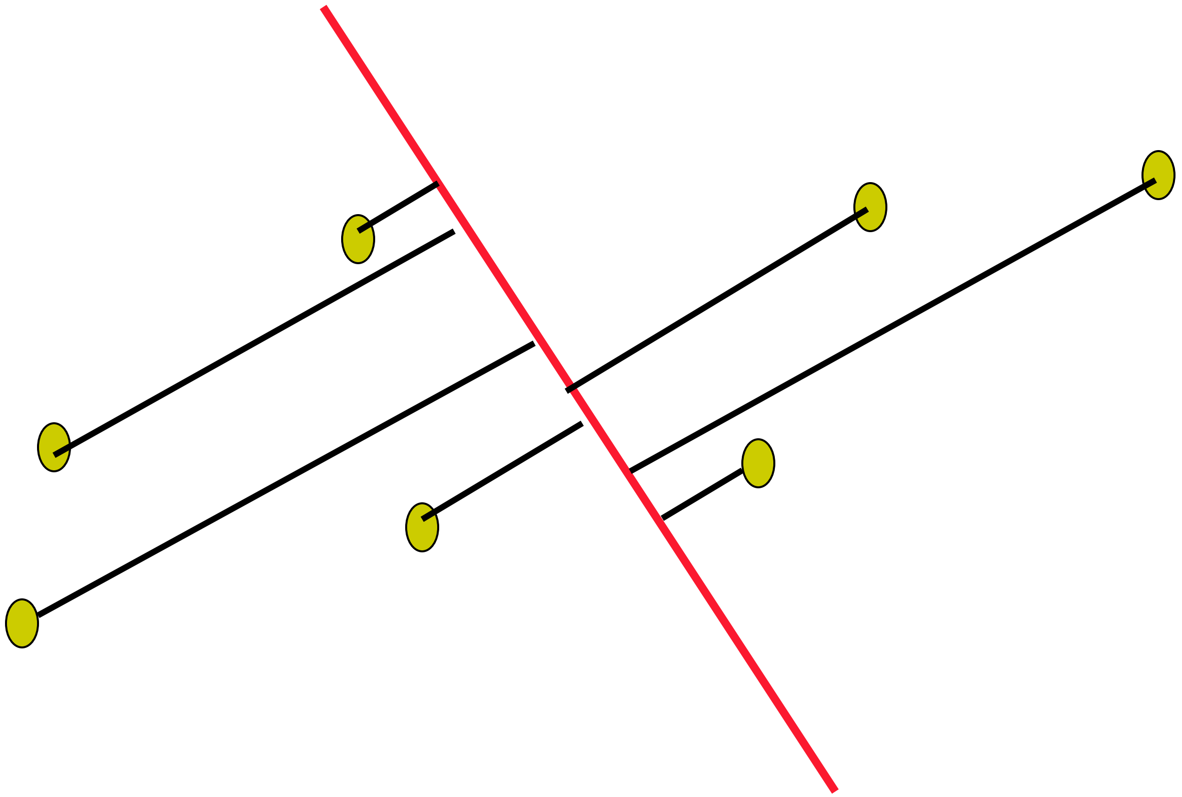
    => maximizes variance of projected points
  - Generate data points as close as possible to initial points

    => Minimize reconstruction error = distance between o and o

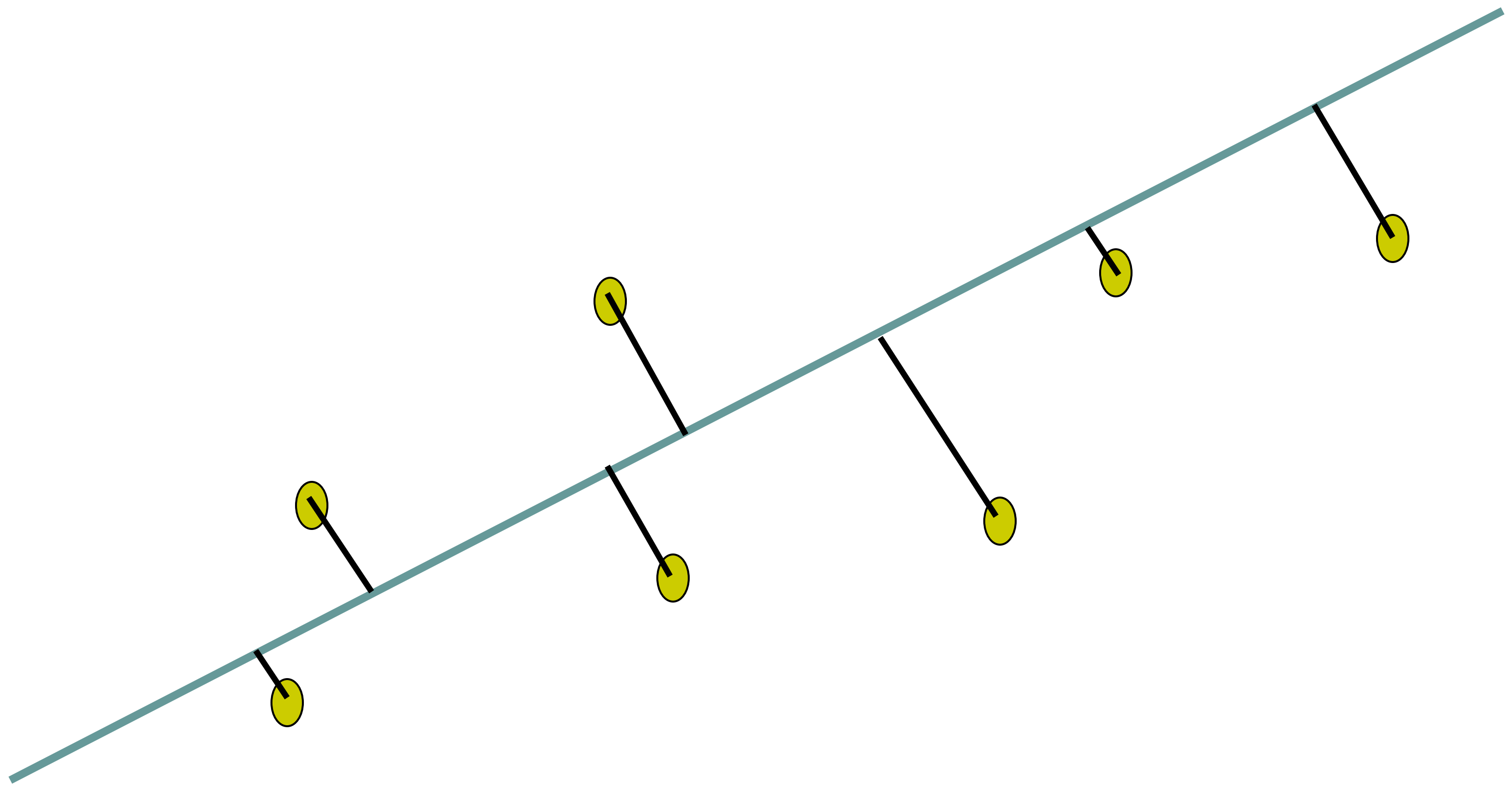
Interpretation of principal components



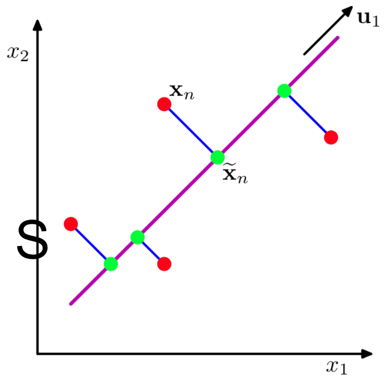
Interpretation of principal components



Interpretation of principal components



Variance maximization



- Mean of data point $\overline{\mathbf{x}}$ - Covariance of original data points S

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$$

$$\begin{aligned}\mathrm{Max}_{\mathbf{u}_1}\ var(y_1) &= \frac{1}{N}\sum_{n=1}^{N}(y_{1n} - \overline{y_1})^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}})^2 \\ &= \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1\end{aligned}$$

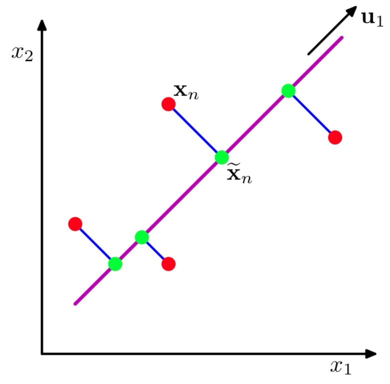
- Maximized if $\|\mathbf{u}_1\| \rightarrow \infty$
- Need constraint $\|\mathbf{u}_1\|^2 = \mathbf{u}_1^t\mathbf{u}_1 = 1$
- This is a constrained optimization problem solved with Lagrangian approach

$$J(\mathbf{u}_1) = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1) \quad \Rightarrow \quad \mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \text{ and } \lambda_1 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

- Conclusion:
  - u1 is the (unitary) eigenvector of S with largest eigenvalue
  - u1 captures the most variation among the training vectors x_n

Note: eigenvectors are sometimes called eigenmodes

Variance maximization



- What about u2 ?
  - It is also an eigenvector of S, orthogonal to u1, whose eigenvalue is the second largest

- Properties.
  - Vectors u_k are orthogonal to each other, define a new coordinate system
  - The principal components coordinates y_k are uncorrelated => covariance matrix of **y** is diagonal with

$$var(y_k) = \lambda_k = \mathbf{u}_k^T \mathbf{S} \mathbf{u}_k$$

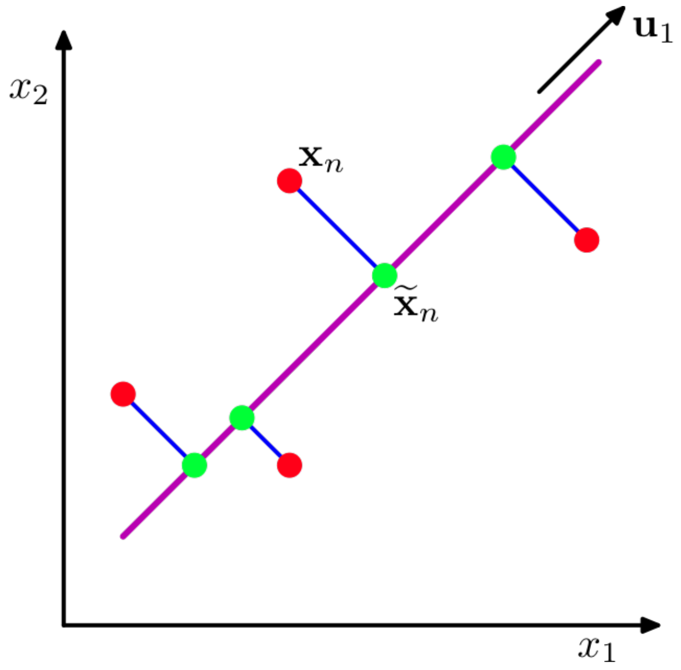
  - The kth largest eigenvalue is the variance of the kth principal component y_k
  - The kth principal component y_k retains the kth fraction of the variability in the sample dataset

Alternative view

- Approximate data points as a linear combination of M new basis vectors

$$\mathbf{x} = \tilde{\mathbf{x}} + noise$$

$$\begin{aligned}\tilde{\mathbf{x}} &= \overline{\mathbf{x}} + y_1\mathbf{u}_1 + \ldots + y_M\mathbf{u}_M \\ &= \overline{\mathbf{x}} + \mathbf{U}\mathbf{y}\end{aligned} \qquad \mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]$$



- we are looking for $\mathbf{u}_i$ that are orthogonal, unitary
- Goal: select the basis vectors minimizing the reconstruction error

$$J = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$$

It can be shown

- $\mathbf{u}_i$ i=1..M are the eigenvectors of S with largest eigenvalues
- Principal component k is given by $y_k = \mathbf{u}_k^T(\mathbf{x} - \bar{\mathbf{x}})$

- Note: if M = D, this corresponds to a change of basis components y_k are the coordinates of x in the new basis

To wrap up

Set of training samples $\{\mathbf{x}_1, ..., \mathbf{x}_N\}, \mathbf{x}_i \in R^D$

- Compute mean and covariance

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

- solve eigenvectors of covariance matrix
  - e.g. SVD decomposition, or more efficient methods if only look for the M first eigenvectors, and D is of large dimension
  - select/keep the M first eigenvectors (and eigenvalues)

=> sort eigenvectors **u**$_i$ by decreasing order of eigennvalues

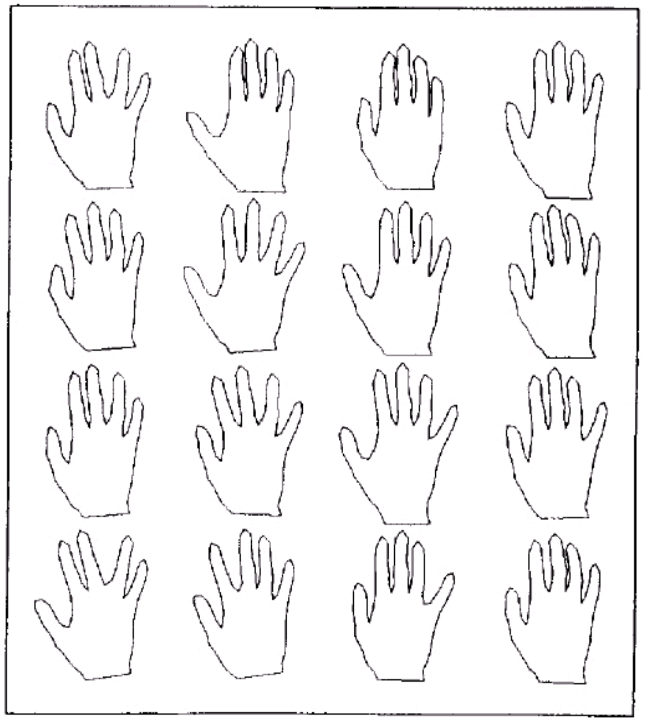
=> form matrix $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_M)$

- lower dimensional representation of datapoints is given by $\mathbf{y}_n = \mathbf{U}^T(\mathbf{x}_n - \bar{\mathbf{x}})$
- approximate reconstruction of data point $\tilde{\mathbf{x}}_n \simeq \bar{\mathbf{x}} + \mathbf{U}\mathbf{y}_n$
- Total square error made by approximation $\sum_{i=1}^{N}(\mathbf{x}_n - \tilde{\mathbf{x}}_n)^2 = (N-1)\sum_{j=M+1}^{D}\lambda_j$

(λ_j discarded eigenvalues in the projection)

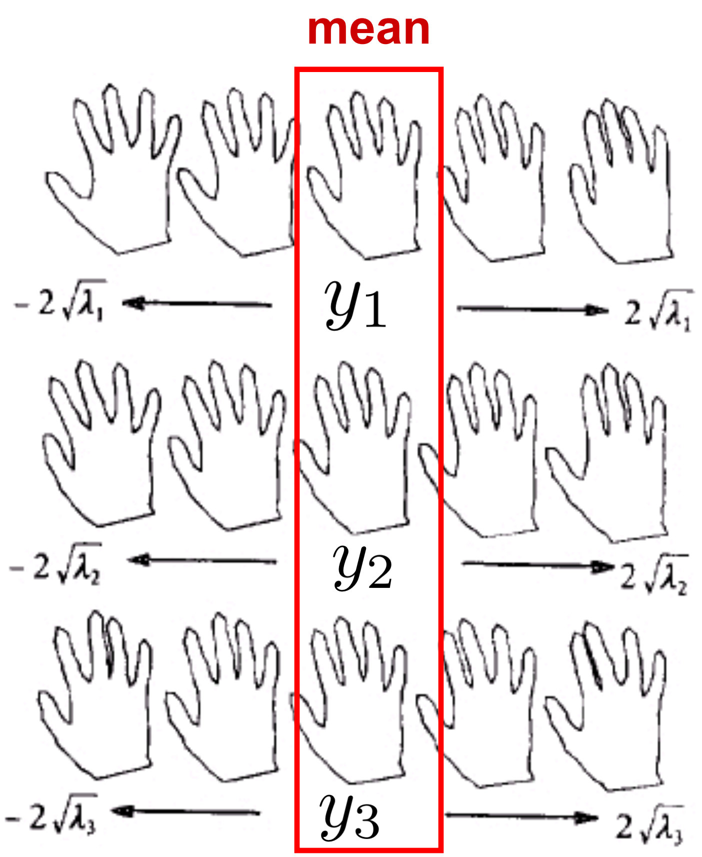
PCA – some examples

Eigenshapes (Cootes)

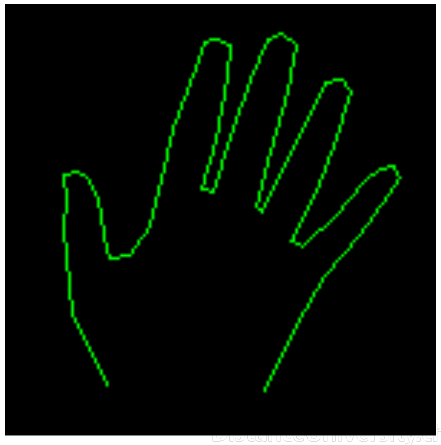


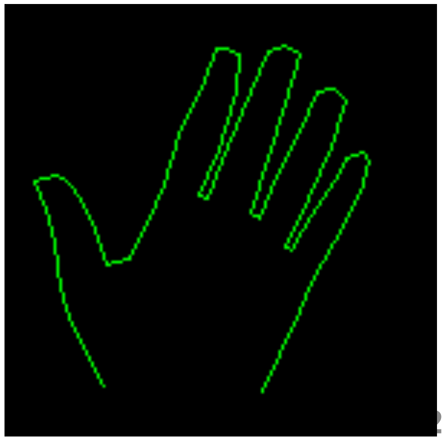
Training data

$$\tilde{\mathbf{x}} = \overline{\mathbf{x}} + \sum_k y_k \mathbf{u}_k$$

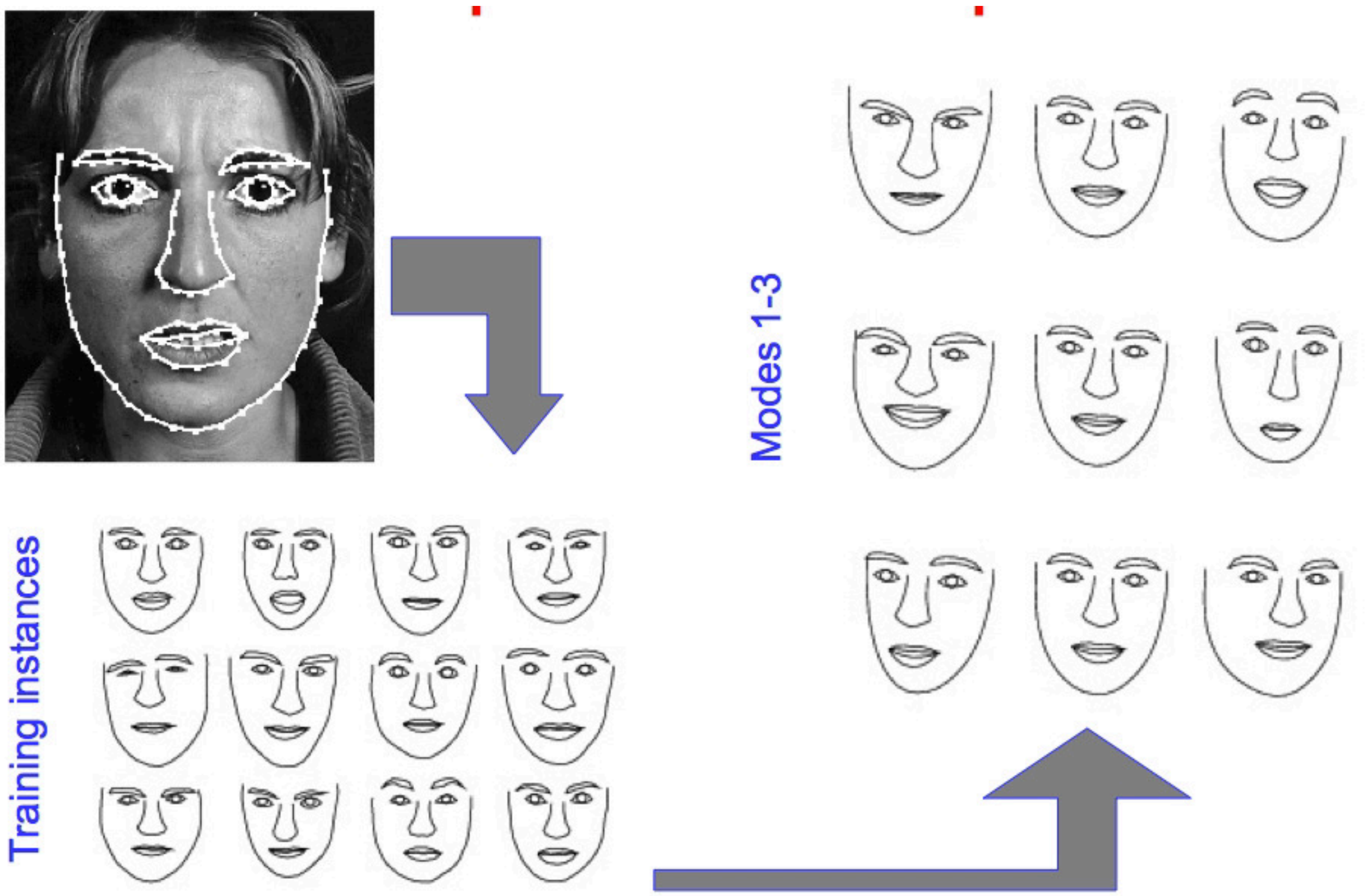


- Synthetize new shapes by varying the principal component coefficient y_k





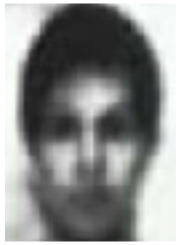
Faces shapes

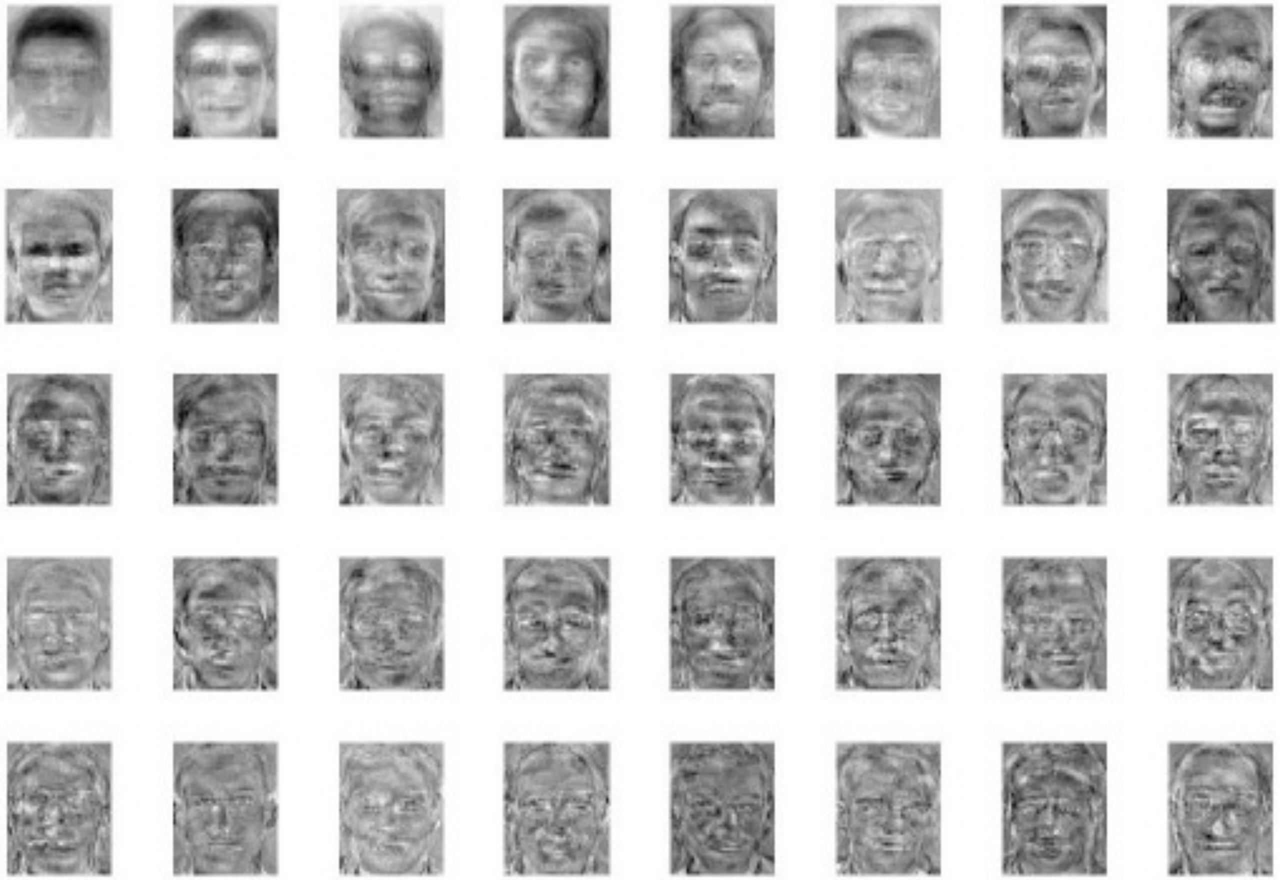


Appearance - eigenfaces

Training images   x1….xn

Eigenfaces

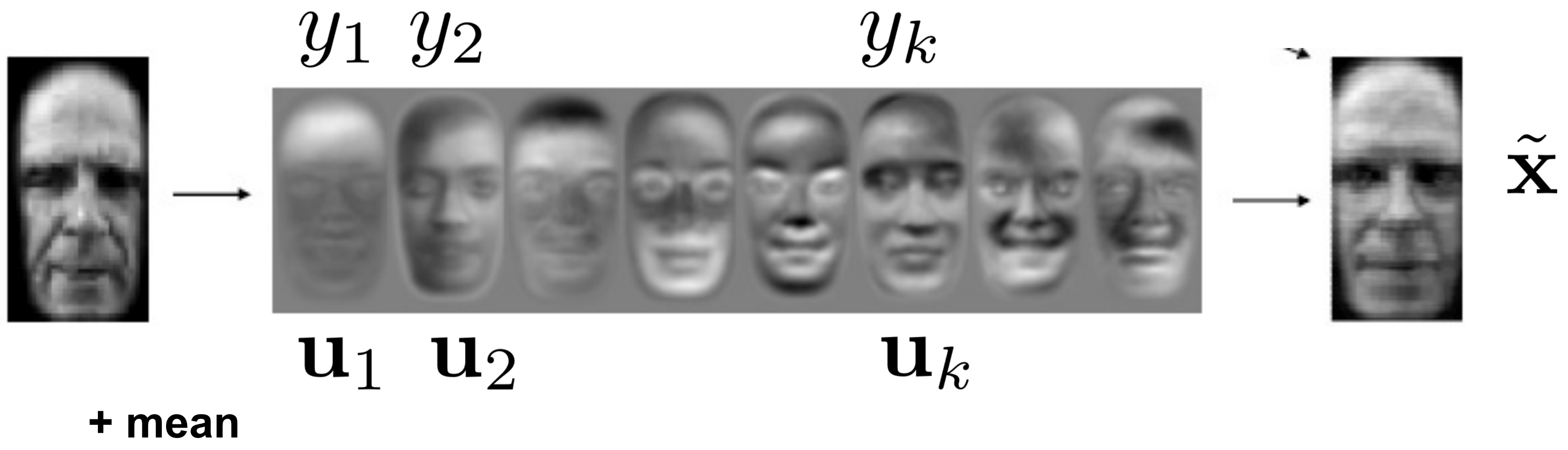

Mean face



- each eigenface has the dimension of an image
- model variations around mean intensity
- contain positive (bright) and negative values (dark)

Eigenfaces

- Face x in ‘face space’ coordinates



$$\tilde{\mathbf{x}} = \overline{\mathbf{x}} + \sum_k y_k \mathbf{u}_k$$

Discussion

Considerations (1)

- How to choose the number M of eigenmodes to keep ?
  - Eigenmodes account for data variability =>
    select smallest M such that the Total Variance in projection space is above a fraction of the original variance

Total variance of training data in subspace of size M

$$\frac{\sum_{k=1}^{M} \lambda_k}{\sum_{k=1}^{D} \lambda_k} > Thresh$$

Total variance of data in full space

(we assume eigenvalues are ranked in decreasing order)

  - Other: if PCA is used as preprocessing for a classification
    Select M through cross-validation

Considerations (2)

- Note: if M=D (if we keep all dimensions), is PCA useful ?

  Yes. Data point components y in new basis are decorrelated

  => covariance is diagonal

  Interesting when using GMM with diagonal covariance

  in further modeling steps

- PCA application - Data whitening
  - heterogeneous data: different dimensions correspond to different measures

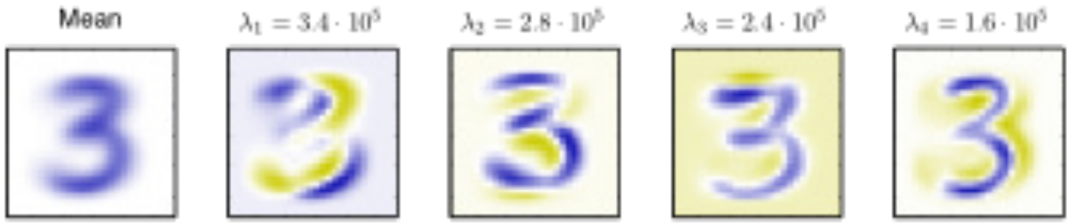
    e.g. for a person: x = (age,height, weight) etc => pb: not comparable units
  - before further step (which may require distance computation), normalize data
    - standardization: rescale each component so that it has zero meand and unit standard deviation
    - with PCA: $$\mathbf{y}_n = \mathbf{L}^{-1/2}\mathbf{U}^T(\mathbf{x}_n - \bar{\mathbf{x}})$$
    - where **L** contains the DxD diagonal matrix with elements λ_j

    => the y elements are uncorrelated with **unit variance** (the covariance is the identity matrix)

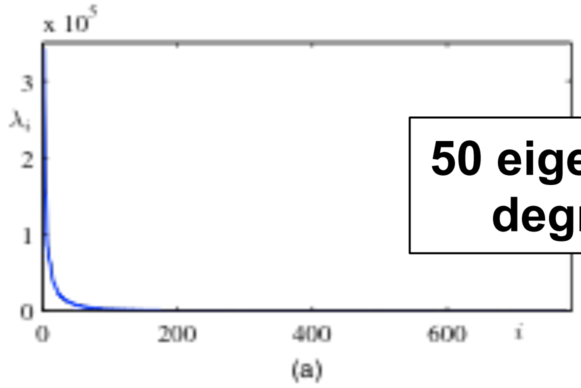
Limitations

Training data:
784 letter 3 images translated, rotated

Mean and eigenvectors

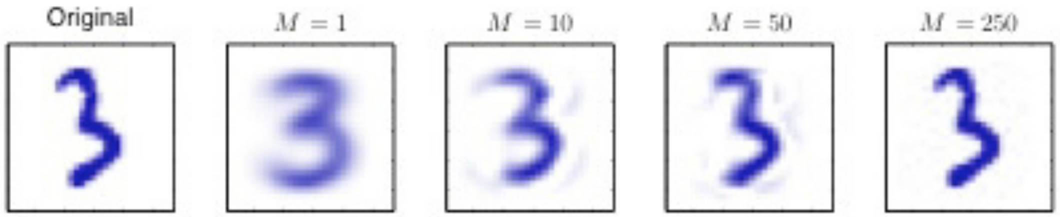

Mean
$\lambda_1 = 3.4 \cdot 10^5$
$\lambda_2 = 2.8 \cdot 10^5$
$\lambda_3 = 2.4 \cdot 10^5$
$\lambda_4 = 1.6 \cdot 10^5$

Ranked eigenvalues



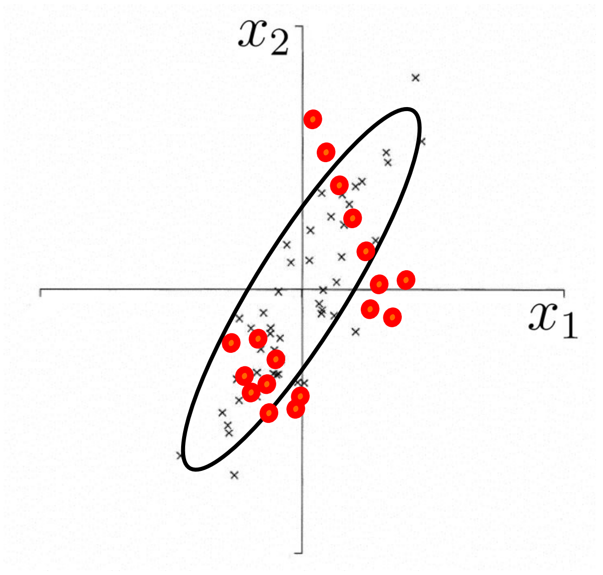
50 eigenvectors to model 3 degrees of freedom…

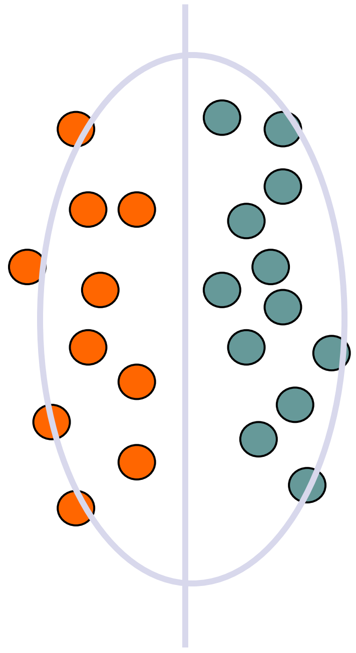
Reconstruction



Limitations

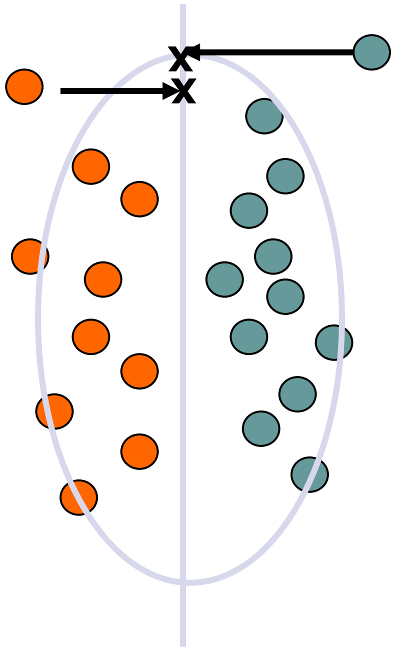
- Assumes that data are distributed as a Gaussian => may not be true
- If there is a model for data generation => fit the model (e.g. using regression models)
- Are the principle component vector good for classifications ? maybe not. use of the classe label (cf Linear Discriminant Analysis, LDA)



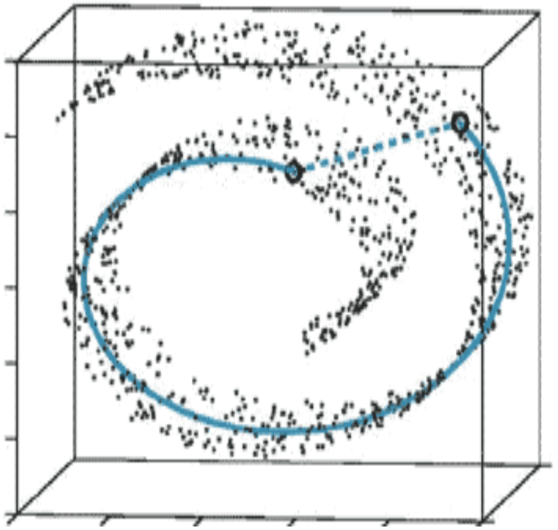


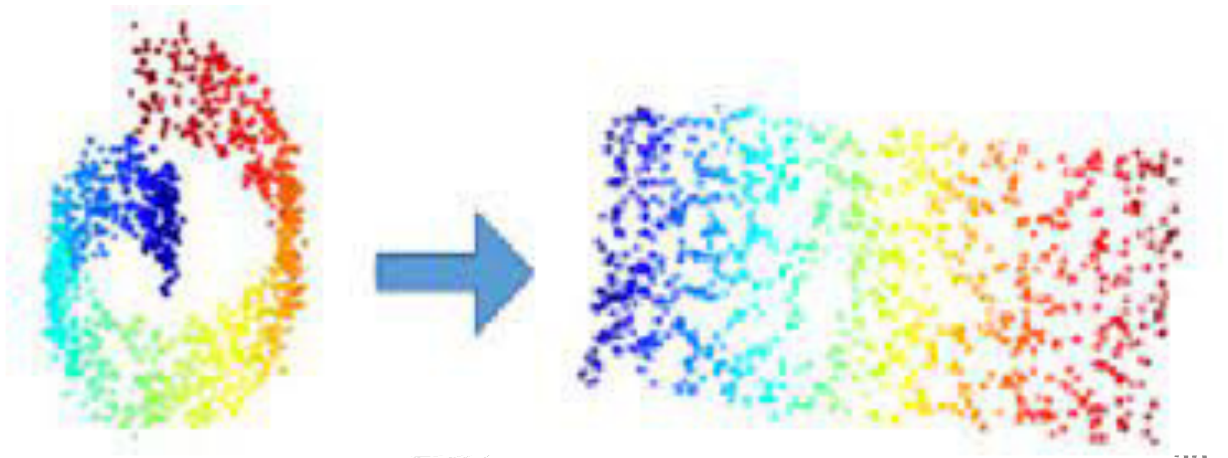
Limitation

- Capture global information
  - Fit data with an hyperplane (linearity)
  - Far away points might be projected close



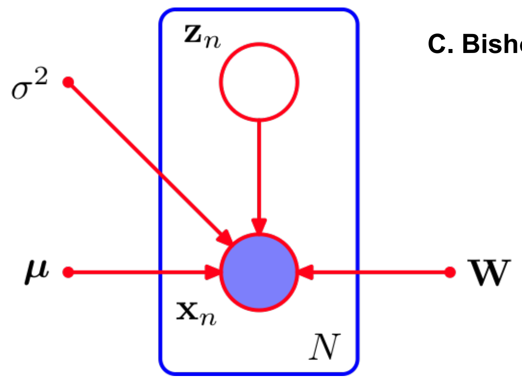
- Some data – located on manifolds
  - ⇒ dimensionality reduction :
  - Non-linear embeddings
  - Preserve Local Structures





Extensions

C. Bishop



- Probabilistic PCA
  - Fully generative model
  - Allows training using EM algorithm for training, mixtures of PCA
- Kernal PCA
  - Use non-linear mapping functions (cf later course on SVM)
- Other linear/non-linear feature reduction techniques
  - Isomap, Kohonen maps
  - Locally Linear Embedding, t-SNE

Summary about PCA

- Interest
  - Curse of dimensionality: high-dimensionality data difficult to manipulate
  - Intrinsic data dimension is usually small

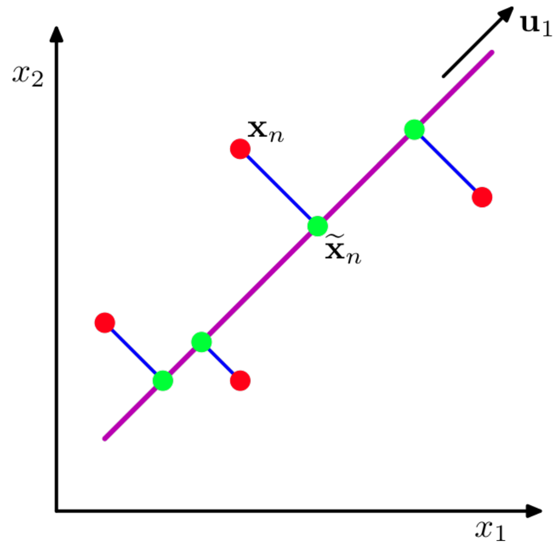
- PCA
  - Feature reduction technique, unsupervised (no data label)
  - Project initial data points with a linear projection
  - Projection directions given by eigenvectors of covariance matrix
  - Projected points keep maximum variance of initial training points

- Application
  - Data compression: less coordinates needed – efficient storage
  - Visualization: project high-dim points into 2D or 3D
  - Synthesis of new data point feasible
  - Noise removal: keep only the essential information

    => positive effect on subsequent steps

Probabilistic PCA (PPCA)

PCA - Summary



- Way to remove correlation between points
  => reduce dimensions through linear projection

- Data driven: training samples $\{\mathbf{x}_1, ..., \mathbf{x}_N\}, \mathbf{x}_i \in R^D$
  - compute mean and covariance $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$ $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$
  - find largest eigenvalues of covariance matrix
    => sort eigenvectors ui by decreasing order of eigenvalues
    => form matrix $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_M)$
  - lower dimensional representation of datapoints is given by $\mathbf{y}_n = \mathbf{U}^T(\mathbf{x}_n - \bar{\mathbf{x}})$
  - approximate reconstruction $\tilde{\mathbf{x}}_n \simeq \bar{\mathbf{x}} + \mathbf{U}\mathbf{y}_n$

Probabilistic view of PCA => probabilistic PCA

- PCA: algebraic view of data

- Alternative

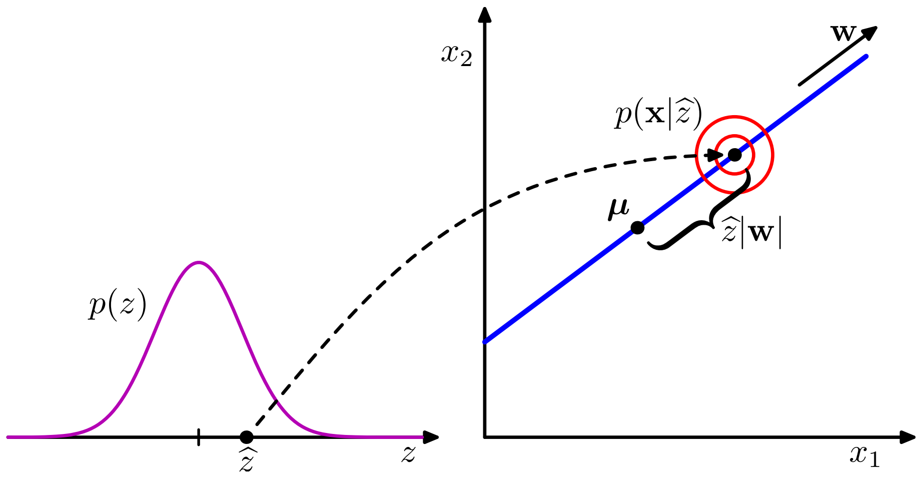
  => find a lower-dimensional **probabilistic** description of the data

  => PCA as maximum likelihood solution of a probabilistic latent variable model

- What are the advantages ? what do we gain ?

  => we first present the model and then show its added properties

Probabilistic PCA – Generative process



$$\mathbf{x} \in \mathbb{R}^D$$

$$\mathbf{z} \in \mathbb{R}^M$$

- Generative process : from latent (low dimensional space) to data space
  - draw $\hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ $\qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$
  - draw $\hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}|\mathbf{W}\hat{\mathbf{z}} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$ $\qquad p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$$

- dimension D x M
- column vectors define subspace in data space

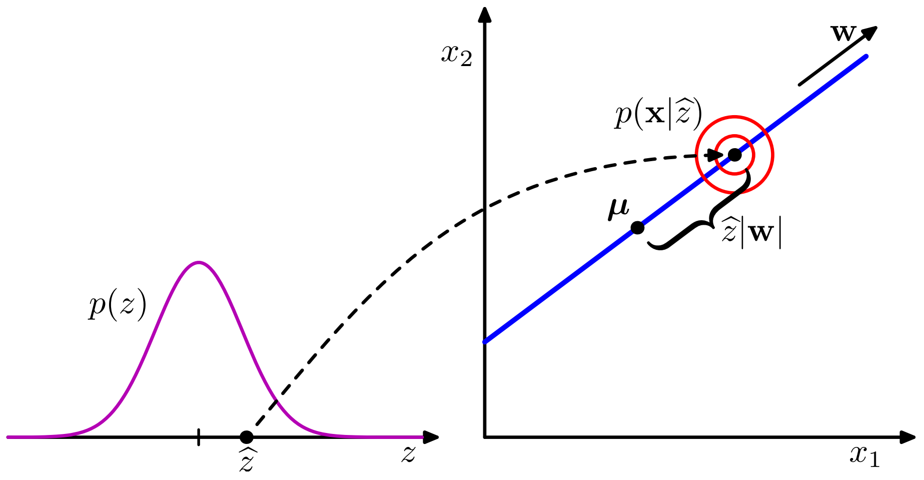
places origin in data space

- random white noise
- uncorrelated with **z**
- isotropic in data space

  - model parameters : $\boldsymbol{\mu}, \mathbf{W}, \sigma^2$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \sigma^2\mathbf{I})$$

Probabilistic PCA – Generative process



chris bishop

- Generative process : from latent (low dimensional space) to data space

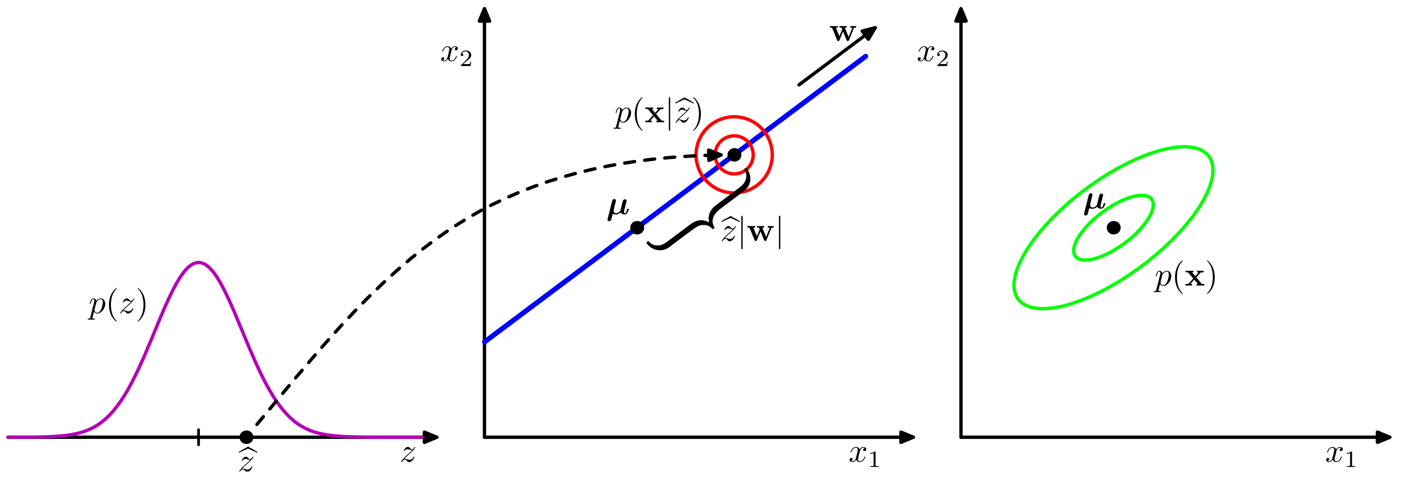
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \qquad p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

- Model defines a probability density in data space $p(\mathbf{x})$
  - Note : distributions are Gaussians => all involved distributions are Gaussian
  - mean and covariance of $p(\mathbf{x})$

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \epsilon] = \boldsymbol{\mu}$$
$$\mathrm{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{W}\mathbf{z} + \epsilon)(\mathbf{W}\mathbf{z} + \epsilon)^T] = \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + \mathbb{E}[\epsilon\epsilon^T] = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}) \qquad \mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

Probabilistic PCA – Generative process



© chris bishop

- Generative process : from latent (low dimensional space) to data space

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \qquad p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

- Note: unidentifiability of matrix **W** in data space
  - select matrix $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$

    R: M x M rotation matrix in latent space

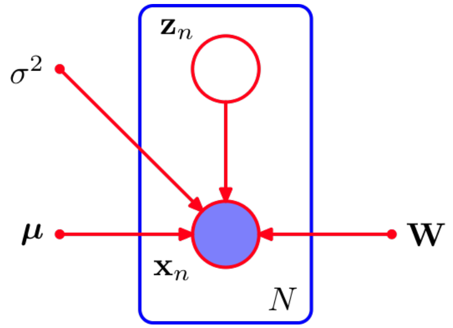
$$\tilde{\mathbf{C}} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^T + \sigma^2\mathbf{I} = \mathbf{W}\mathbf{R}\mathbf{R}^T\mathbf{W}^T + \sigma^2\mathbf{I} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} = \mathbf{C}$$

=> defines the same distribution in data space

=> due to isotropy in latent space

=> redundancy in parameterization of **W**

Probabilistic PCA – Learning



Training data set $\mathbf{X} = \{\mathbf{x}_n\}$

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\bar{\mathbf{x}})(\mathbf{x}_n-\bar{\mathbf{x}})^T$$

- Link with PCA ? maximum likelihood PCA parameter estimation
  - Likelihood $\ln p(\mathbf{X}|\boldsymbol{\mu},\mathbf{W},\sigma^2) = \sum_{n=1}^{N}\ln p(\mathbf{x}_n|\boldsymbol{\mu},\mathbf{W},\sigma^2)$

$$= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{x}_n-\boldsymbol{\mu})$$

  - Optimization w.r.t. parameters: closed form solution

$$\boldsymbol{\mu}_{\mathrm{ML}} = \bar{\mathbf{x}} \qquad \mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \qquad \sigma^2_{\mathrm{ML}} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i$$

$\mathbf{U}_M$ Columns given by the M eigenvectors corresponding to the M largest eigenvalue of **S**

$\mathbf{L}_M$ MxM diagonal matrix has elements given by the corresponding eigenvalues

$\mathbf{R}$ arbitrary rotation matrix => non-uniqueness of **W** (cf previous slide)

λ_i eigenvalues sorted in descending order of magnitude

Probabilistic PCA vs PCA

- Comparison with PCA

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{W}_{\mathrm{ML}}\mathbf{z} + noise$$

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2} \qquad \mathrm{cov}(\mathbf{z}) = \mathbf{I}_M$$

PCA : reconstruction

$$\tilde{\mathbf{x}} \simeq \bar{\mathbf{x}} + \mathbf{U}_M\mathbf{y}$$

$$\mathrm{cov}(\mathbf{y}) = \mathbf{L}_M$$

=> (taking R=I) similar expression (except -σ^2 scaling), in a probabilistic framework

=> recovers PCA when σ^2 tends to 0

- variance $\sigma^2_{\mathrm{ML}} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i$

variance associated with discarded dimensions cf variance of reconstruction error in PCA

- projection in latent space – it can be shown that for the ML case

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|E[\mathbf{z}|\mathbf{x}], \sigma^{-2}_{\mathrm{ML}}\mathbf{L}_M)$$

$$E[\mathbf{z}|\mathbf{x}] = \mathbf{L}_M^{-1}\mathbf{W}_{\mathrm{ML}}^T(\mathbf{x} - \bar{\mathbf{x}})$$

PCA

$$\mathbf{y}_n = \mathbf{U}^T(\mathbf{x}_n - \bar{\mathbf{x}})$$

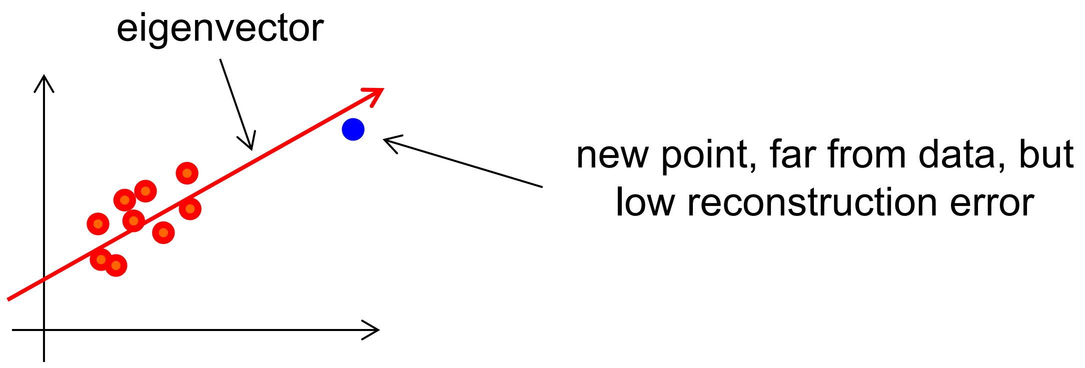
  - Recover PCA case when σ^2 tends to 0 (but probability distribution get ill-defined)
  - Note: this does not correspond to an orthogonal projection

Probabilistic view of PCA => probabilistic PCA

- What are the advantages ? what do we gain ?
  - Probabilistic PCA: constrained form of Gaussian distribution on $p(\boldsymbol{x})$
    - number of free parameters is restricted (compared to full covariance model)
    - still captures dominant data correlation (compared to diagonal covariance model)
  - Generative process (we can sample random vectors)
  - Likelihood function of data points

    more informative than the PCA reconstruction error

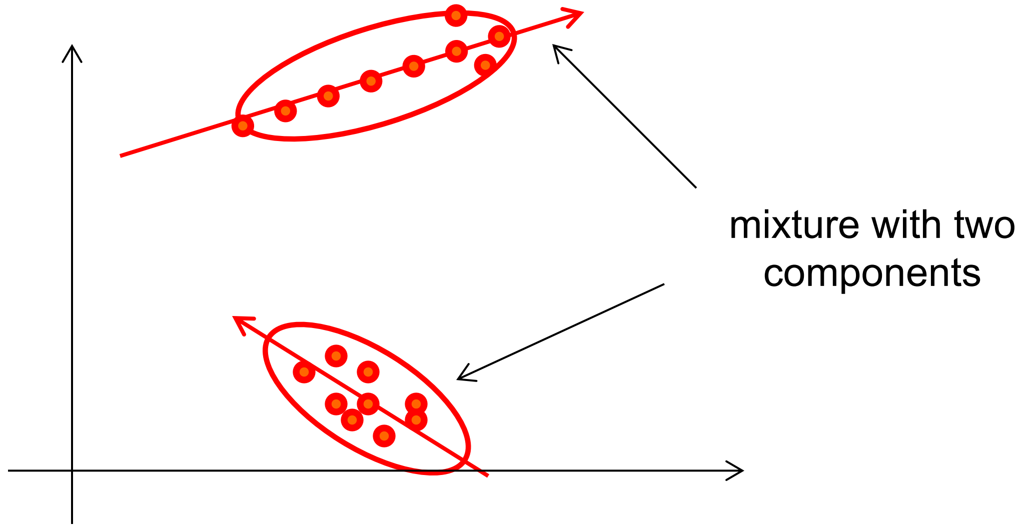
    allow comparison with other probability density models



Probabilistic view of PCA => probabilistic PCA

- What are the advantages ? what do we gain ?
  - latent space model => derivation of computationally efficient EM algorithm for PCA, that does not require computation of covariance matrix
  - Probabilistic model + EM => handling of missing values in the dataset

    (allows PCA projections even if there are missing values in the input values)
  - Principled extensions to other models and particularly Mixtures of probabilistic PCA models



Probabilistic PCA : a specific case of Factor Analysis

- Factor analysis : same principle, but noise in data space is not isotropic

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \qquad p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \underset{\text{general diagonal covariance matrix}}{\underbrace{\boldsymbol{\Psi}}})$$

vs isotropic noise in PPCA $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$

- Notes
  - in FA, the covariance is still taken diagonal, so that the components of x are all independent conditioned on **z** (this is true of course for PPCA as well)
  - in this case, no closed-form solution for the estimation of parameters (W and **ψ**)

Conclusion : Probabilistic PCA

- Probabilistic PCA: classical probabilistic method for finding low-dimensional representation of the data
- Continuous latent model with closed form solution
- Extends to more models (Factor analysis, mixture models)
- Presents advantages compared to traditional PCA (eg handles missing components in the data)

t-Stochastique-Neighbor-Embedding (t-SNE)

What is t-SNE

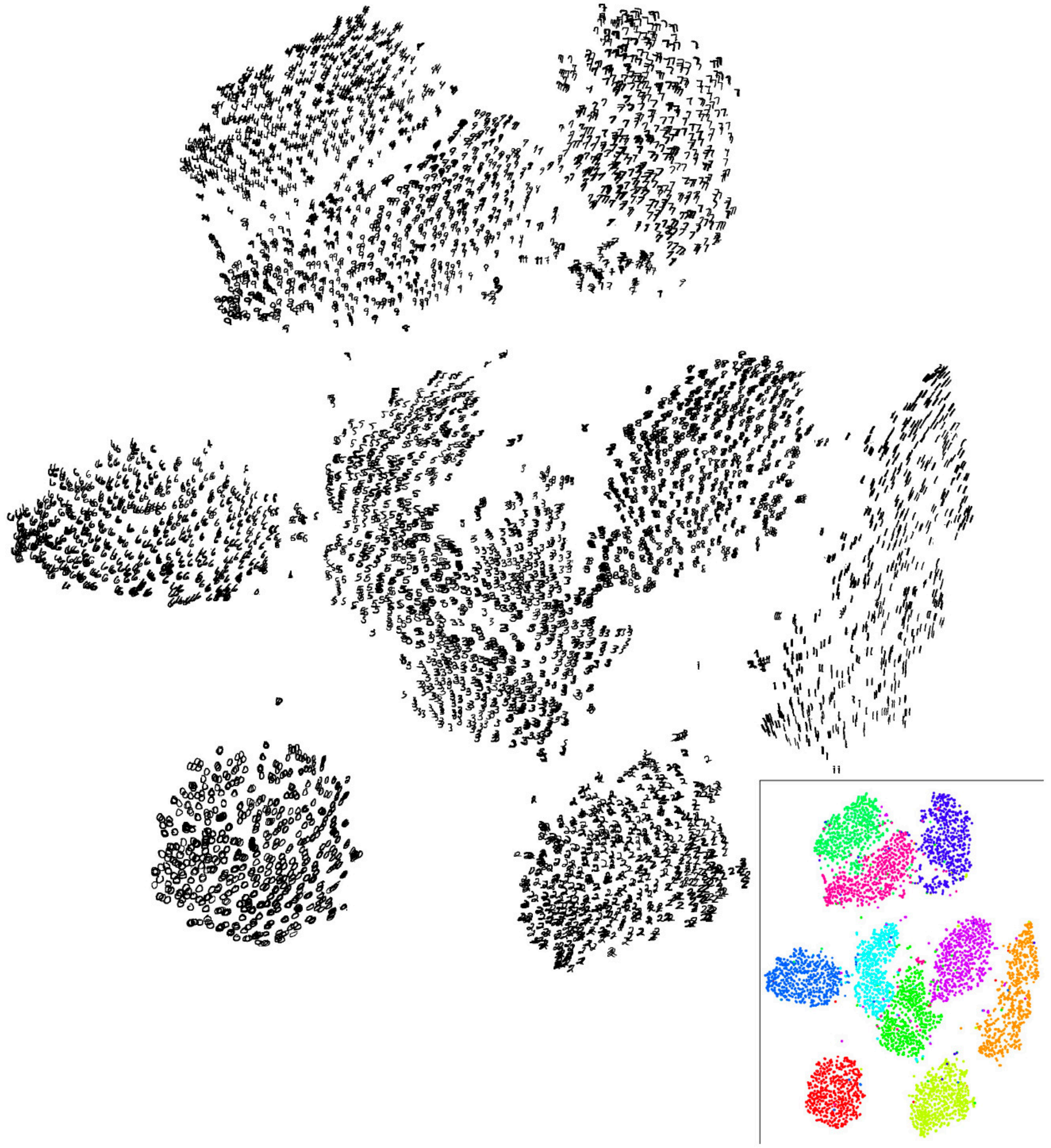
- Dimensionality reduction algorithm
- PCA : global structure and mapping to low dimensional space
  - may lead to some inconsistencies (far away points can become nearest neighbors)
- t-SNE : preserve local structure
  - **low dimensional neighborhood should be the same as original neighborhood**
  - one example of local **embedding** methods (others: Isomap, Linear Local Embedding (LLE), Spectral clustering)

$$x \in E = \mathbb{R}^D \to f(x) = y \in F = \mathbb{R}^d$$

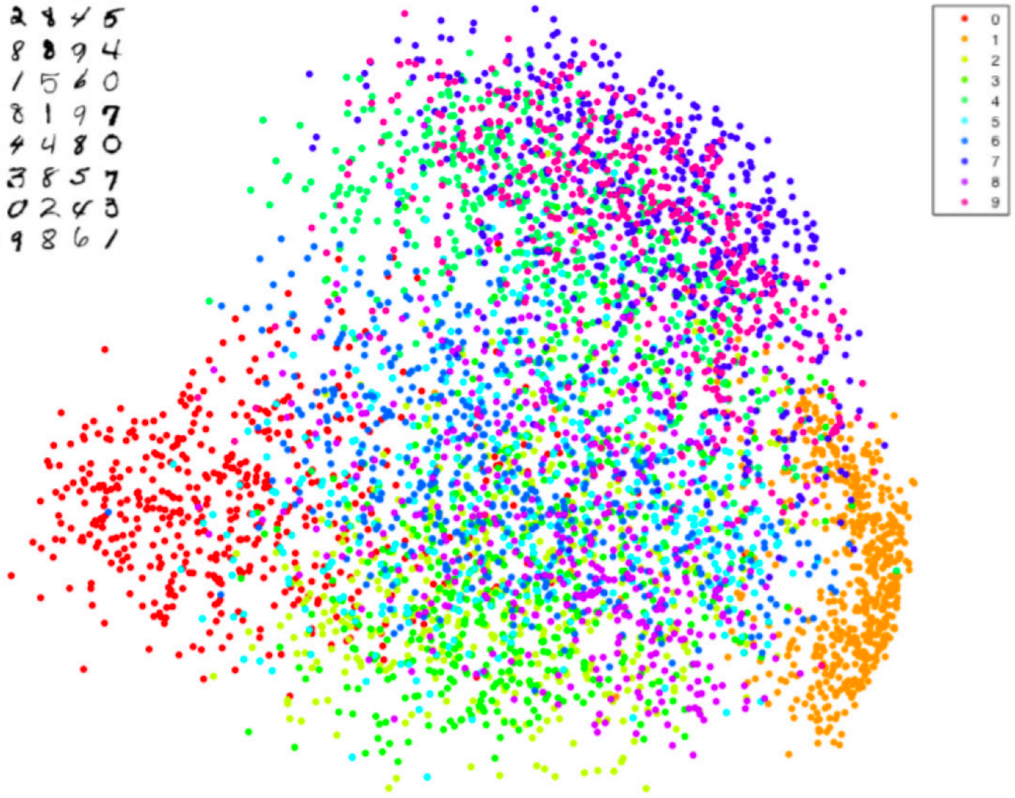
Embedding: projection with the notion of keeping some form of similarity

- Use mainly for data-visualization
  - No easy way to embed new points

Example – t-SNE on MNIST



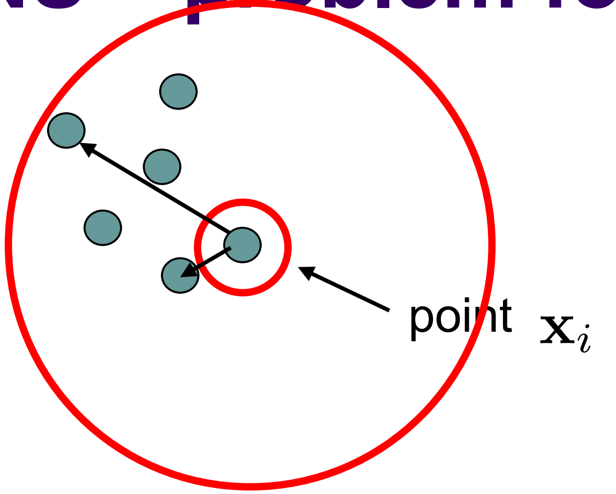
Example – PCA on MNIST

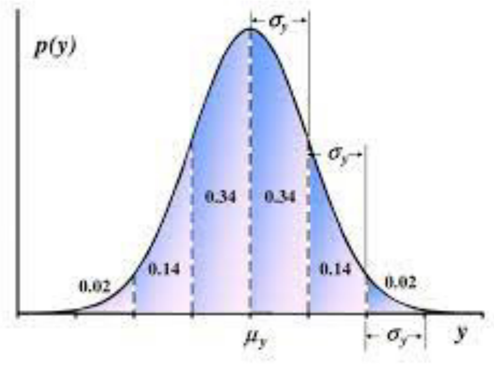


SNE – Stochastique Neighbor embedding

- Basic idea
  - Set of points in high-dimensional space: $\{\mathbf{x}_i \in \mathbb{R}^D, i = 1, \ldots, N\}$
  - Encode high dimensional neighborhood information as a distribution
  - Distribution intuition
    - Introduce probability $p_{j|I}$ that point j is a neighbor of point i
    - Higher probability for close points
  - Find low dimensional points $\{\mathbf{y}_i \in \mathbb{R}^d, i = 1, \ldots, N\}$
    - such that their neighborhood distribution is similar
      - If $\mathbf{x}_5$ and $\mathbf{x}_7$ are close, then $\mathbf{y}_5$ and $\mathbf{y}_7$ should be close as well
    - Similarity measure ? Standard Kullback-Leibler (KL) divergence

SNS – problem formulation (1)





- Define probability that one point j is the neighbor of point i

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma_i^2\right)}{\sum_{k\neq i} \exp\left(-\|\mathbf{x}_k - \mathbf{x}_i\|^2 / 2\sigma_i^2\right)} \qquad p_{i|i} = 0$$

- Parameter σ_i set the size of the neighborhood
  - Very low value – most probability in nearest neighbor
  - Very high value – uniform weights, all points are neighbors
  - Note: we use a different σ_i for each datapoint
- Results depends heavily on this parameter => defines neighborhood we try to preserve

SNE – problem formulation (2)

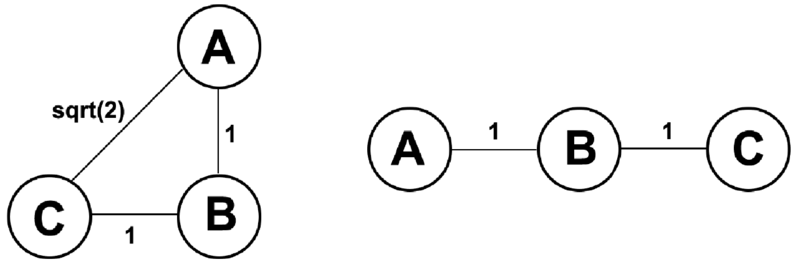
- Given the data $\{\mathbf{x}_i \in \mathbb{R}^D, i = 1, ..., N\}$, define the distribution

$$P_i = [p_{1|i}, \dots, p_{N|i}]$$

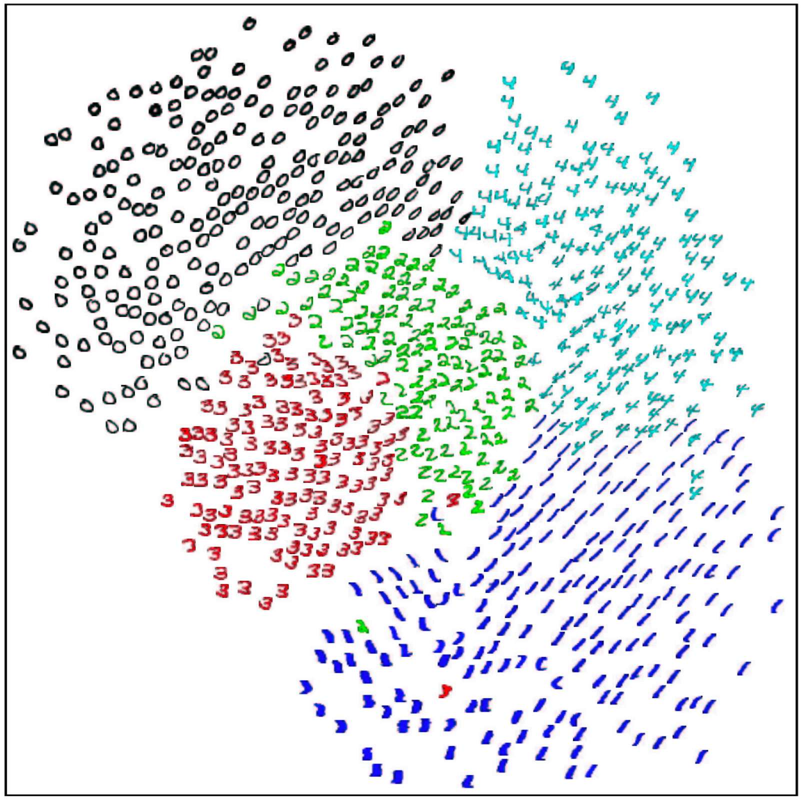
- Goal
  - Find embeddings $Y = \{\mathbf{y}_i \in \mathbb{R}^d, i = 1, ..., N\}$ (with d=2 or 3)
  - Embedding quality ?
    - define neighbor distribution in embedding space $q_{j|i} = \frac{\exp\left(-\|\mathbf{y}_j - \mathbf{y}_i\|^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{y}_k - \mathbf{y}_i\|^2\right)}$
      (note: no sigma parameter)
    - minimize cost function $C(Y) = \sum_i KL(P_i|Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
- Note
  - Cost focuses on local structures (high p's)
  - The embeddings Y are the parameters we are optimizing
  - How to embed a new point ? No embedding function! $\mathbf{x} \rightarrow \mathbf{y} = f(\mathbf{x})$
  - Optimization via Gradient Descent – not convex! Use multiple restart!

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_i - \mathbf{y}_j)\left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)$$

SNE example/issue



In 1D (right):
AB and BC preserved, but AC too large



- Local relationship preserved
- "Crowding problem"
  - High dimension space: more room, easy to have multiple neighbors
  - Low dimensional space: area available to accomodate moderately distant point not large enough compared to area for nearby data points
  - Distinct cluster in high dimensional space pushed closer in lower space

    => might not be distinguishable

t-SNE (t-student SNE)

Two main differences with SNE

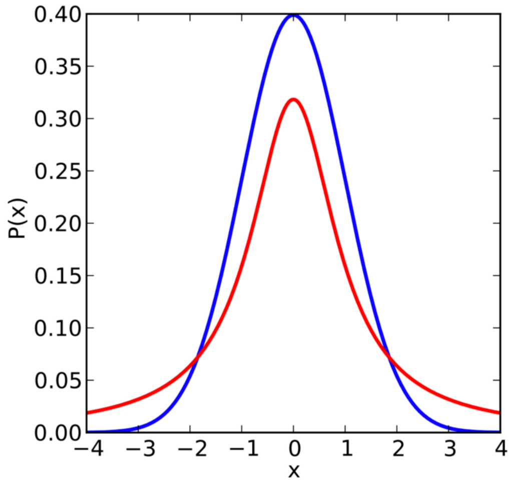
- Symmetrized distribution
  - less sensitivity to outliers
  - leads to simpler gradient

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

- Use of an heavy-tail distribution (Student's t-distribution, in red) to define Q_i

  (P_i unchanged)
  - goes slower to 0 than Gaussian
  - slower change => more space to move points around in medium distance

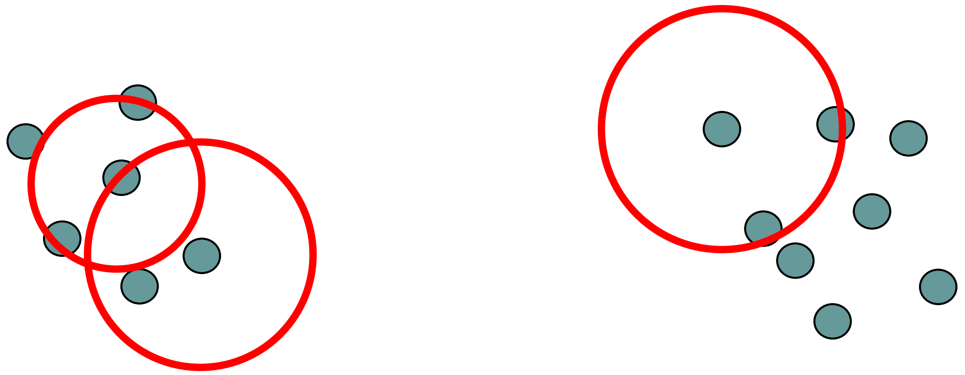
$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}}$$



  - New cost (note distribution is over all ij)

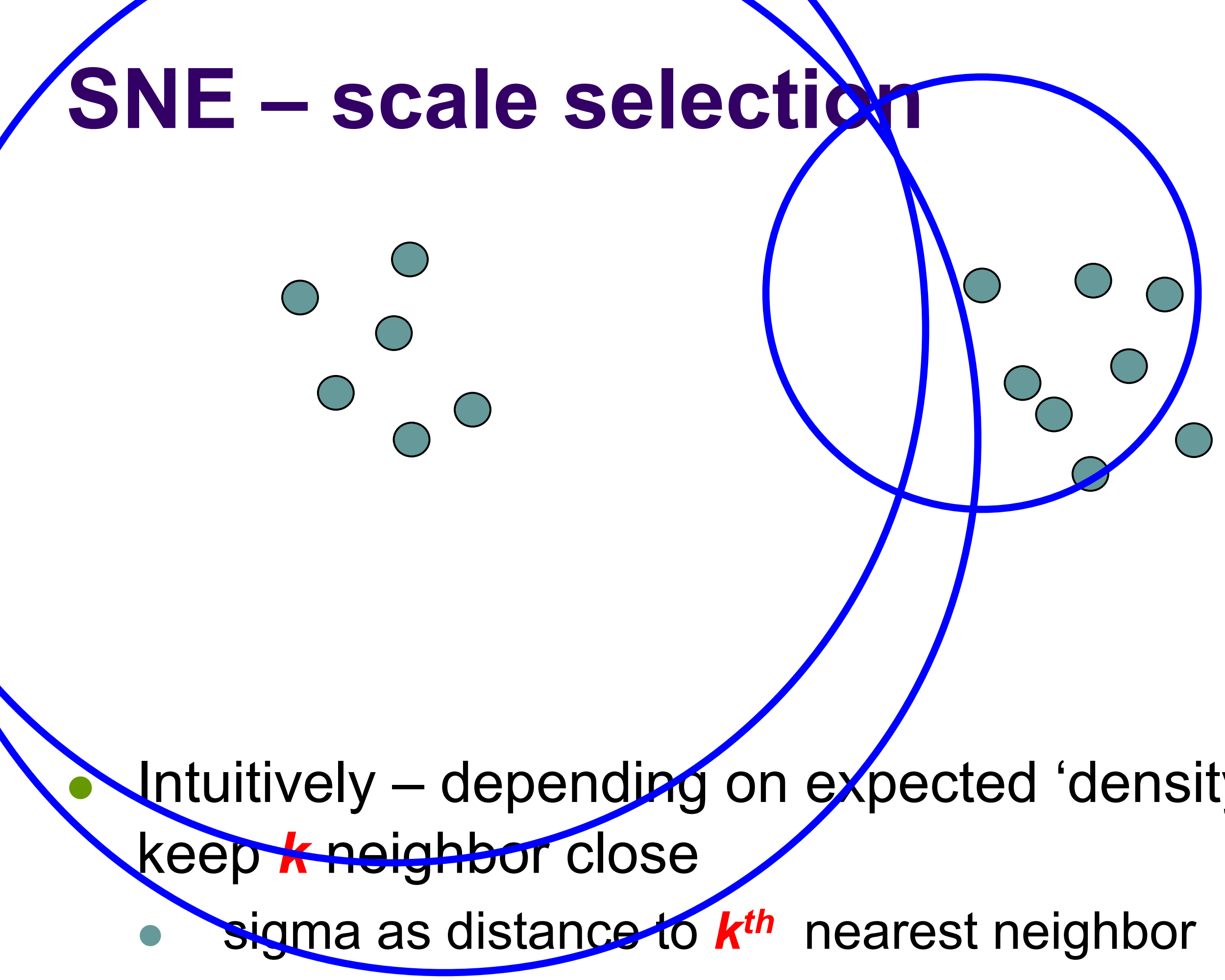
$$C(Y) = KL(P|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

SNE – scale selection



- Intuitively – depending on expected ‘density’ select a scale to keep ***k*** neighbor close
  - sigma as distance to k^{th} nearest neighbor
  - ***k* = 2**

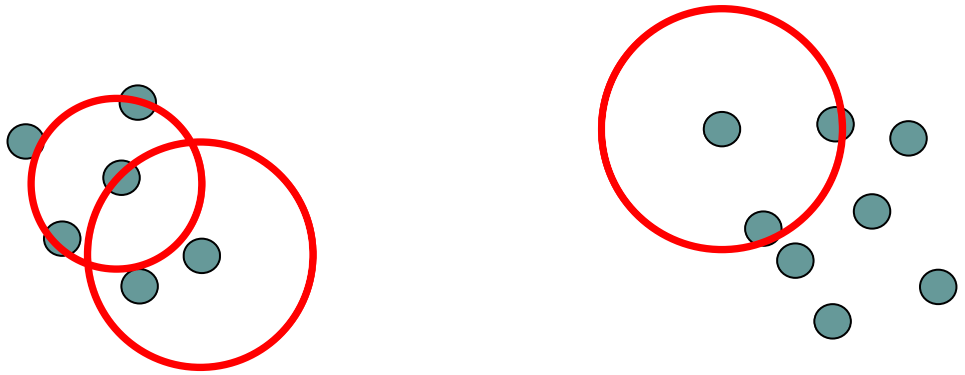
SNE – scale selection



- Intuitively – depending on expected 'density' select a scale to keep ***k*** neighbor close
  - sigma as distance to k^{th} nearest neighbor
  - ***k* = 2**
  - ***k* = 6**

  => left point : points on the right become neighbors

SNE – scale selection



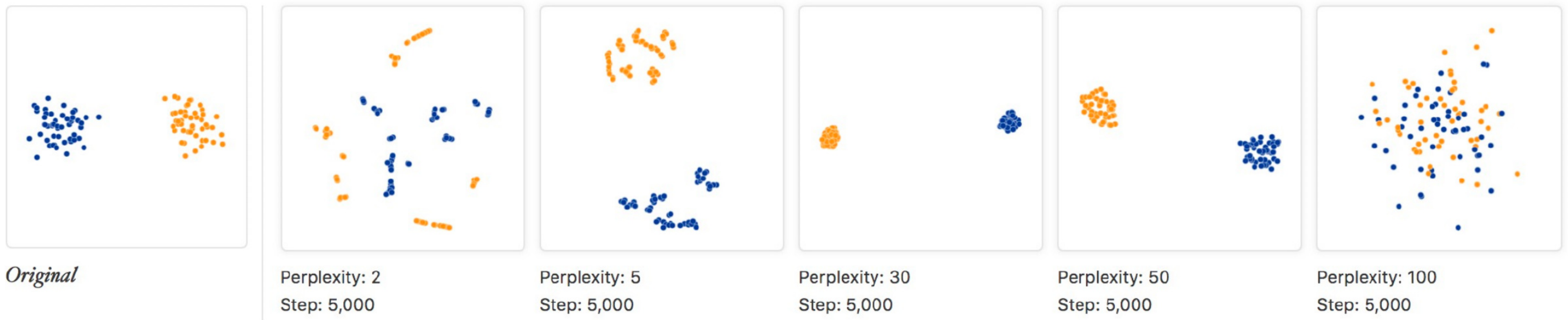
- In SNE: sigma selected to obtain a given perplexity

$$perp(P_i(\sigma_i)) = 2^{H(P_i)} \quad H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

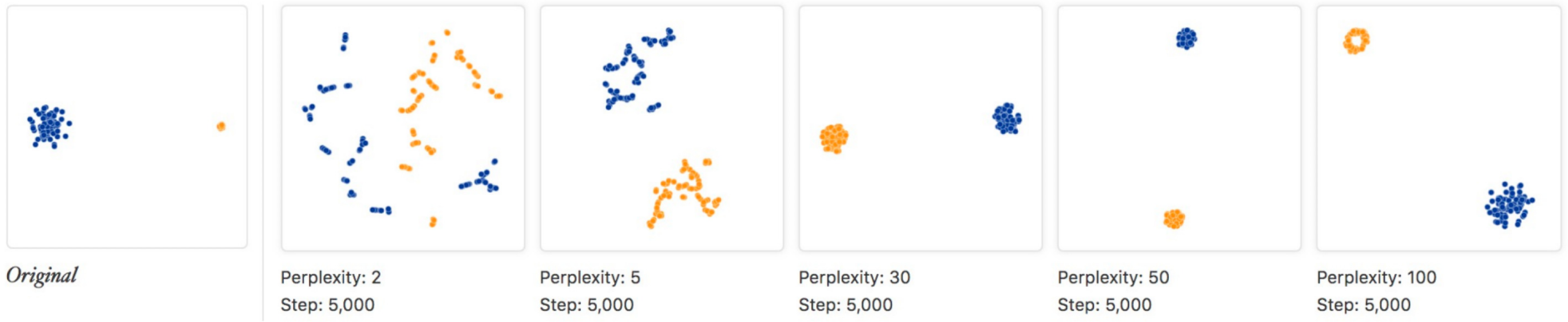
Shannon entropy, in bits

  - P uniform over k elements => perplexity is k
    - Low perplexity – low entropy – small sigma
    - High perplexity – high entropy – large sigma
  - Define the perplexity => search the sigma_i closest to it (via binary search)
  - Important parameter : different perplexity capture different scales in data

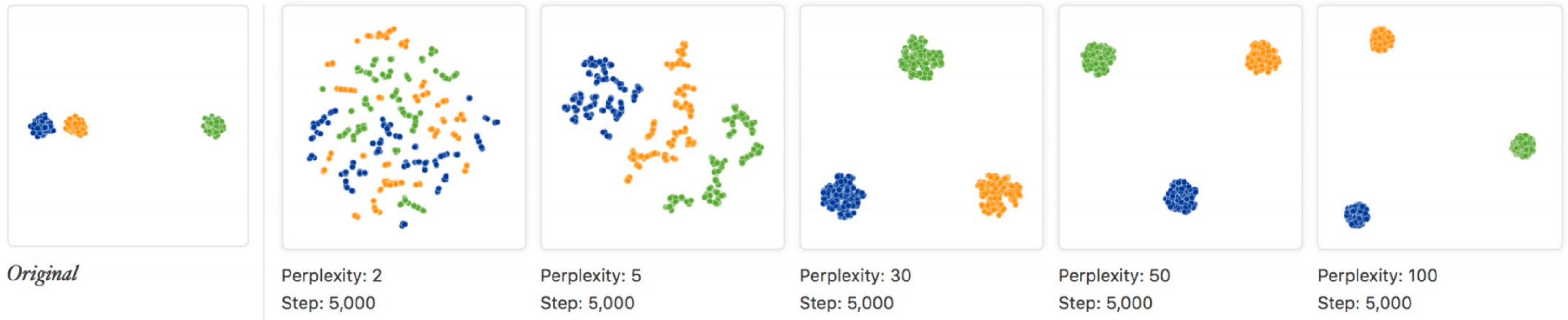
Impact of perplexity



Change of perplexity => impact on 'scale of analysis'



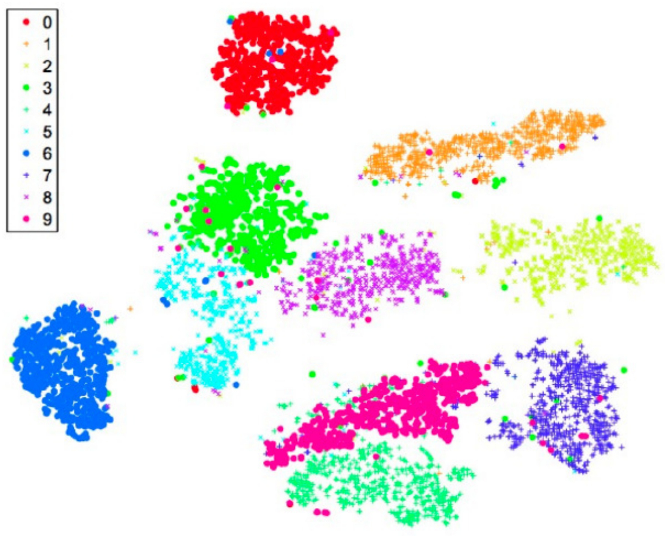
Output cluster size do not reflect cluster size in original space



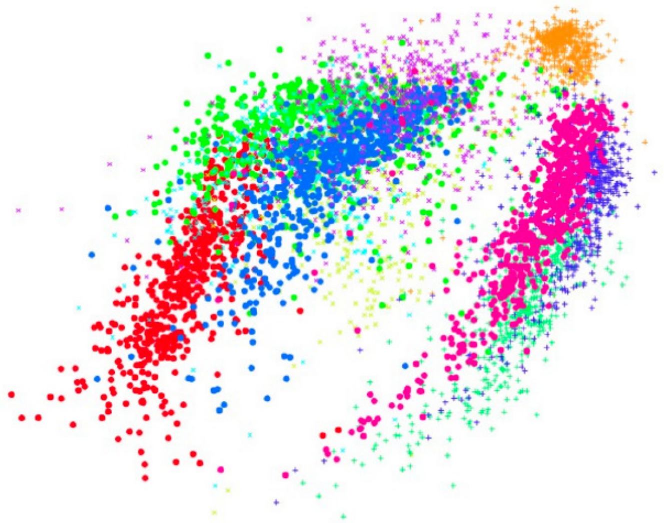
Relative position and distance in output does not reflect inout relative position and distance (orange cluster in-between, closer to blue cluster)

Comparison with other Local methods

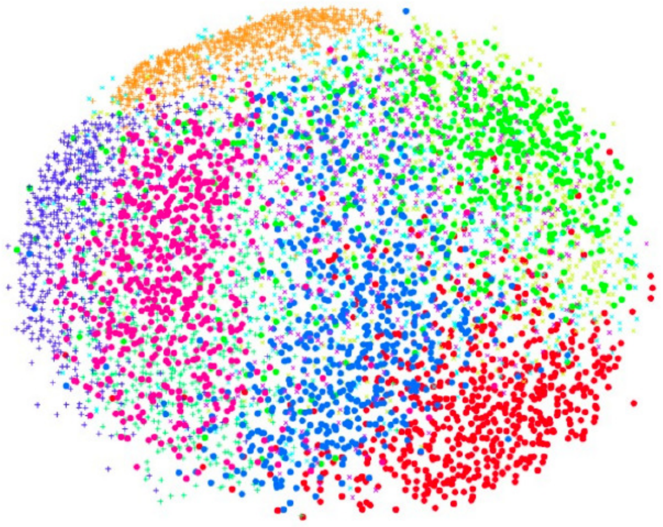
- MNIST



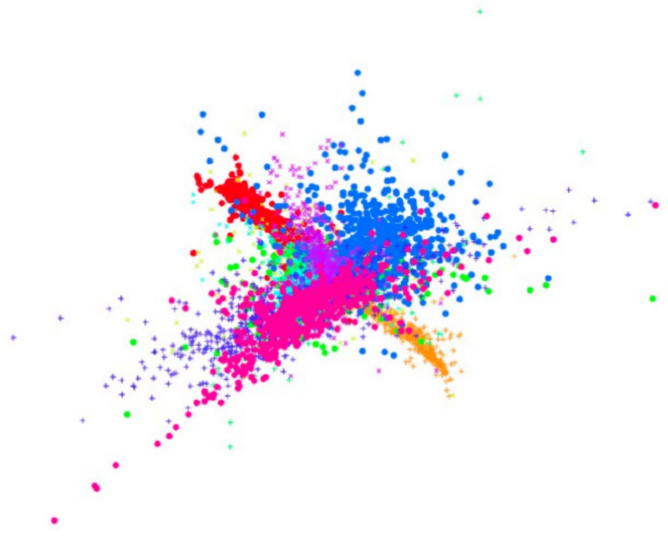
(a) Visualization by t-SNE.



(a) Visualization by Isomap.



(b) Visualization by Sammon mapping

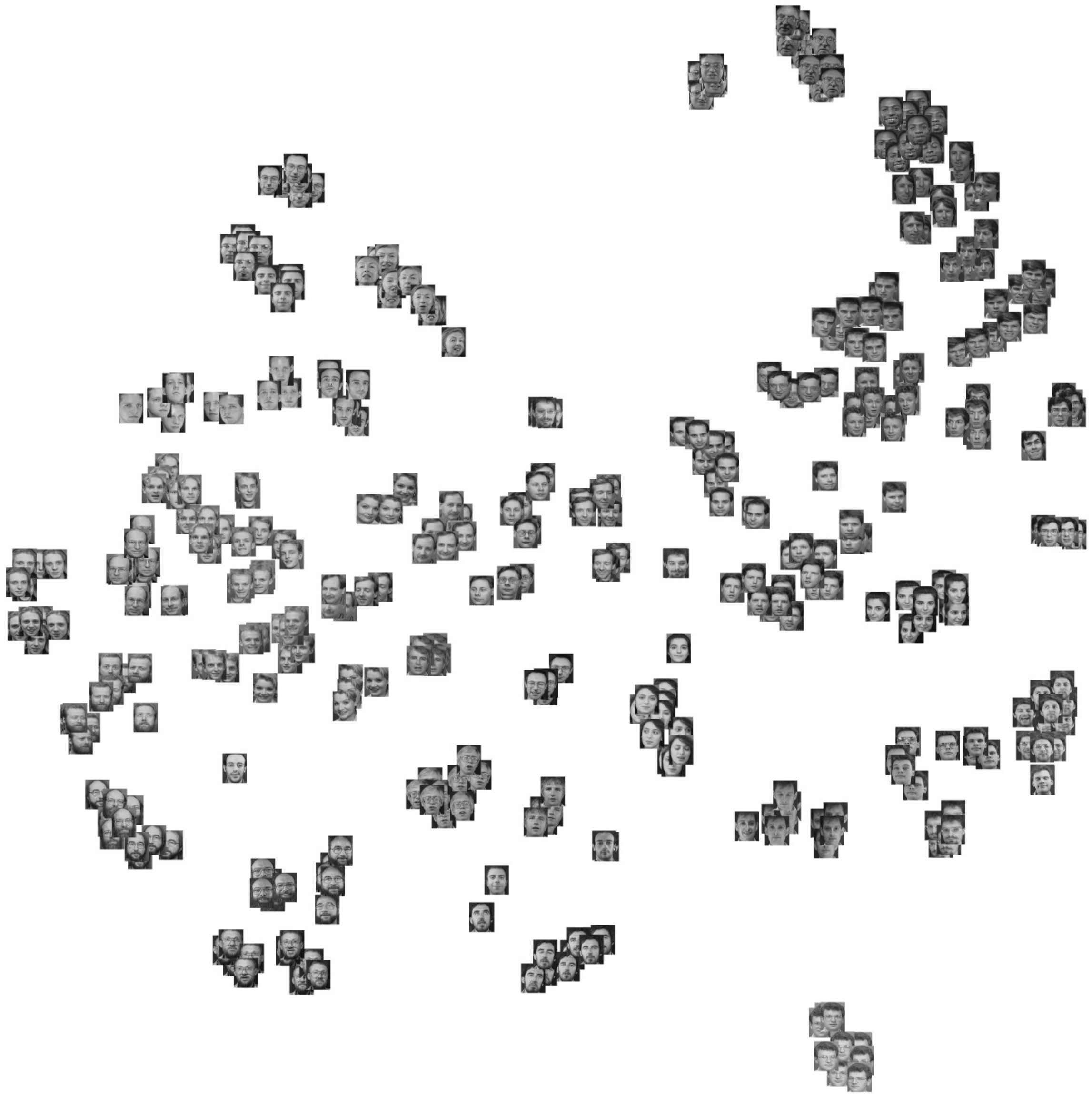


(b) Visualization by LLE.

CNN embeddings – 4096 dim features

- Karpathy - CNN embeddings

CNN embeddings – from face CNN



t-SNE : conclusions

- Great way to visualize high-dimensional space data
  - Example: Deep Networks embeddings (CNN features)

- Local approach
  - preserve neighborhood information
  - heavy-tail distribution to avoid crowding problem
  - non-convex optimization

  - curse of dimensionality (euclidian distance in original space, so depends on local linearity in the data manifold)
  - dependency on perplexity factor
  - no embedding function