MACHINE LEARNING IN BIOINFORMATICS

Part 2: Unsupervised learning

František Mráz
KSVI MFF UK

Sources

- Yang, Zheng Rong. *Machine learning approaches to bioinformatics*. Science, Engineering, and Biology Informatics — Vol. 4. World scientific, 2010.
- Bishop, Christopher M. *Pattern recognition and machine learning*. Vol. 4. No. 4. New York: Springer, 2006.

Learning for Analysis of Biological Data

- **Example:** A mass spectrometry experiment on a set of plants
  - Generates many thousands of metabolites (intermediates and products of metabolism; the term *metabolite* is usually restricted to small molecules <= 900 daltons (1 dalton = weight of a nucleon (neutron or proton)))
  - Each metabolite can be represented by mass and abundance values for replicates
  - Based on masses it is possible to infer a number of chemical formulas of candidate compounds from different pathways
  - One metabolite can be mapped to multiple compounds
- **Supervised learning**
  - Find a mapping from one data space to another data space

    Chemical formulas $\rightarrow$ *pathways*
- **Unsupervised learning**
  - One data space is missing – reorganize the data space to explore the missing data space

Why Unsupervised Learning

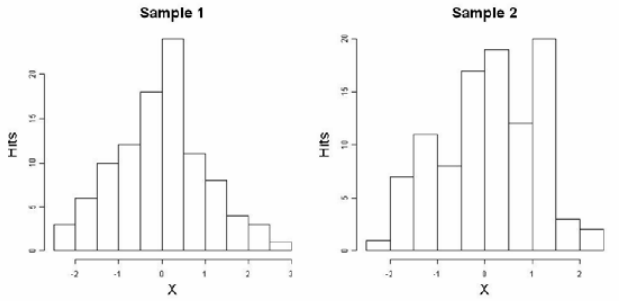
Example: Analysis of gene expression

- It is possible to monitor simultaneously thousands of genes and proteins under different experimental conditions – for studying genome- and proteome-wide functions and regulatory mechanisms
- **Dimensionality reduction** – to reduce the dimensionality of the 'gene space' (e.g. in microarray data) by constructing 'super-genes' – for simplifying structure of the data
- **Visualizations** – reduce the data to 2 or 3 dimensions (e.g. principal components analysis)
- **Clustering** – partitioning the data into groups of objects more 'similar' to each other than objects in different groups – identifying biologically relevant groups of both genes and samples and have also provided insight into gene function and regulation

Subjects of Unsupervised Learning

1. **Density estimation**
   - Find information hidden in the data
   - E.g. the data can be generated from a normal distribution (i.e. x_i is a real value) and we can find the parameters of the Gaussian (mean and standard deviation)

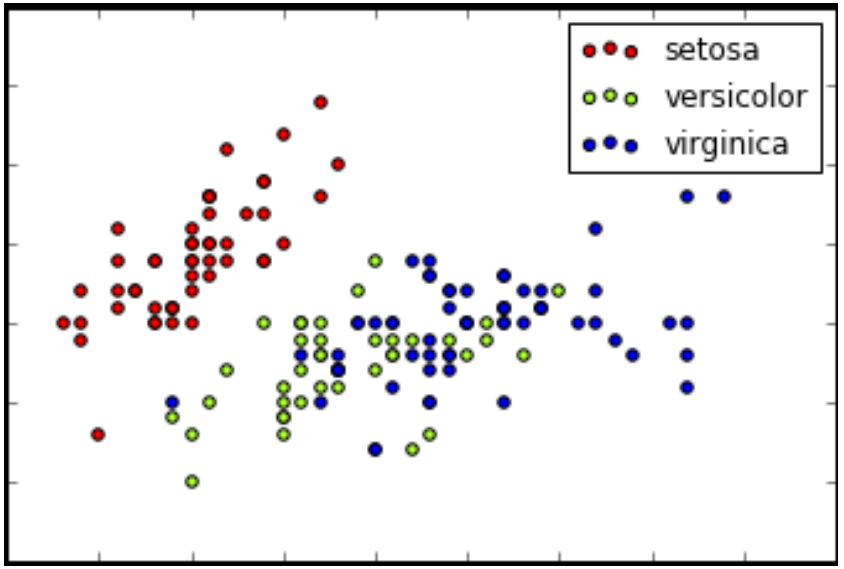
$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}$$



   where $x_1, x_2, \ldots, x_N$ are the data points, $N \geq 1$
   - Gaussian distribution is regarded as data structure and parameters are regarded as inference rules
   - Usually methods from statistics
2. **Data visualization**
3. **Cluster analysis**

Subjects of Unsupervised Learning

1. **Density estimation**
2. **Data visualization**
   - Often many dimensional data is not possible to visualize
   - E.g. gene expression data for a disease may contain only a few samples, but with 1000≈100000 genes as variables
3. **Cluster analysis**



Mapping four dimensional Iris data to a 2-dimensional space. Setosa is well separated from the other two species, which are difficult to separate.

Subjects of Unsupervised Learning

1. **Density estimation**
2. **Data visualization**
3. **Cluster analysis**
   - A data set may be viewed as a composition of disjointed sub-data structures
   - Each sub-structure contains data points with similar properties
   - How to find these sub-structures and quantitatively describe them?

Part 2: Unsupervised learning

PROBABILITY DENSITY ESTIMATION APPROACHES

Probability Density Estimation Approaches

1. The histogram approach – the simplest method
2. A parametric approach – assumes a structure in data (e.g. normal distribution); training data not kept
3. Non-parametric approaches – no explicit data structure
4. Semi-parametric approach

1. Histogram Approach

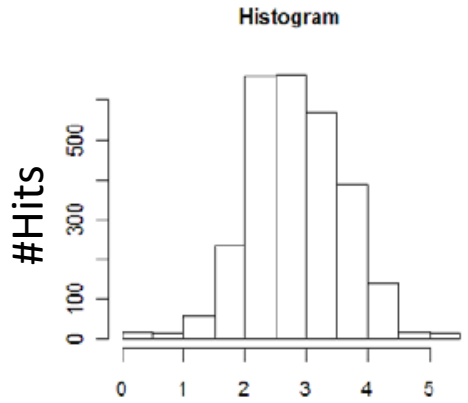
- Each coordinate is divided into segments of a fixed length (called bins)
- If $x \in [a, b]$, the interval is divided into K bins of length

$$\Delta = \frac{b-a}{K}$$

- Each training data point is assigned to the bin it belongs
- Frequency of each bin is

$$\frac{\#\text{training data falling into the bin}}{\#\text{all training data}}$$

- The frequency is used as the probability how likely a point falls into the bin – simple visualization is a histogram



n_i = #data points in bin i

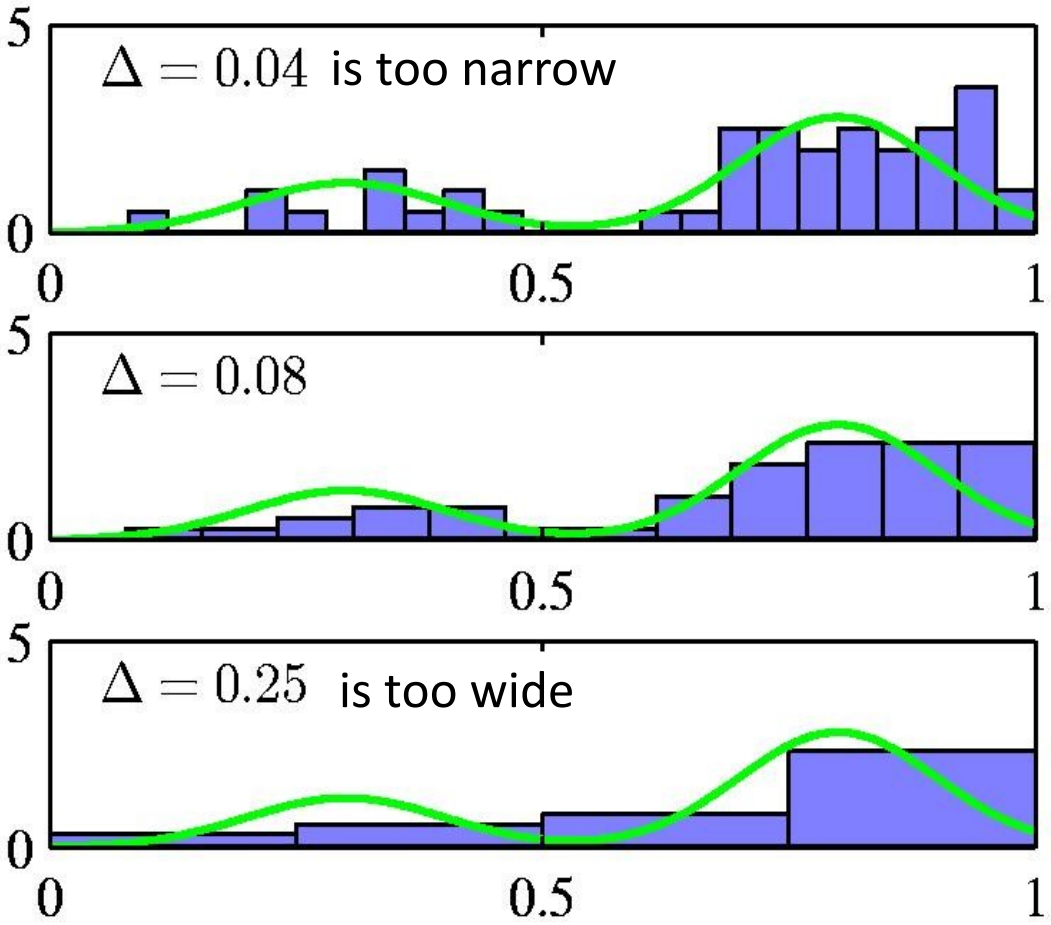
p_i = probability a point falls in bin i

$$p_i = \frac{n_i}{N}$$

How to approximate the probability density function?

$$f_i = \frac{n_i}{N\Delta}$$

1. Histogram Approach

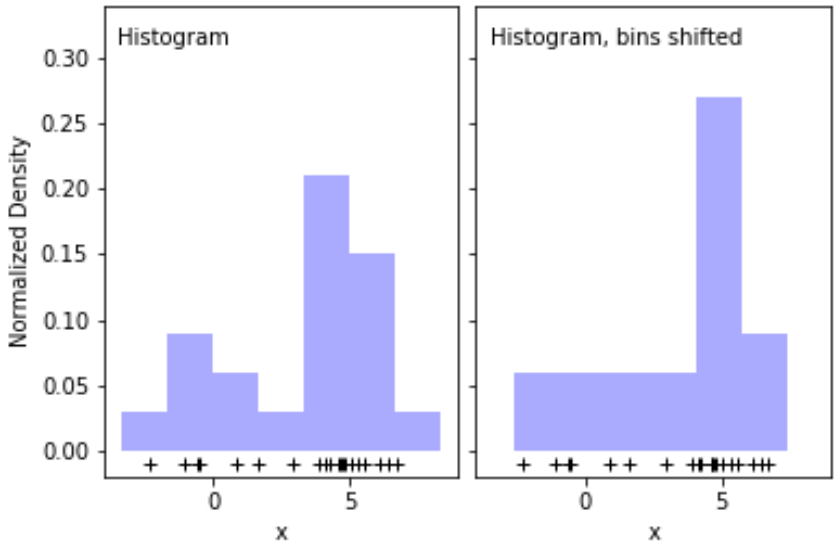


1. Histogram Approach

- Pros:
  - Simple method, no explicit data structure is supposed
  - No need to fit a model to the data
  - We just compute some very simple statistics (the number of data points in each bin) and store them
- Cons:
  - How wide should the bins be? (width=regularizer)
  - Do we want the same bin-width everywhere?
  - Do we believe the density is zero for empty bins?
  - for d variables we need K^d bins; e.g. for $K = 10$ and $d = 10$ require 10^{10} bins
  - The density has discontinuities at the bin boundaries
    - We must be able to do better by some kind of smoothing

1. Histogram Approach

- Cons (cont.):
  - Positions of bin boundaries can change the outcome; see the example from the documentation for scikit-learn
    - On the left: bin boundaries `linspace(-5,10,10)`
    - On the right: bin boundaries `linspace(-4.25,10.75,10)`



2. Parametric Approach

- Assumes a data structure before estimating the probability
- After constructing the density probability function, the training data are discarded
- If we assume a Gaussian data structure
  - A training set of size N is $\mathbf{X} = \{x_i\}_{i=1}^{N}, x_i = (x_{i1}, \dots, x_{id})^T \in \mathbb{R}^d$, (i.e. x_i is a column vector)
  - The training data points are generated by the assumed Gaussian data structure with likelihood $\mathcal{L}$

1D Gaussian probability density function $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$$\mathcal{L} = \prod_{i=1}^{N} p(x_i) \quad \text{where} \quad p(x_i) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}|}} e^{-\frac{(x_i-\mu)^T \mathbf{\Sigma}^{-1} (x_i-\mu)}{2}}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is the covariance matrix (see below), $\mathbf{\Sigma}^{-1}$ is its inverse and $|\mathbf{\Sigma}|$ its determinant, $\mu \in \mathbb{R}^d$ is the mean vector

  - The likelihood function is maximized

2. Parametric Approach

x, μ are a column vectors

- The covariance matrix $\boldsymbol{\Sigma} = \mathrm{E}[(x-\mu)(x-\mu)^T]$ for $x \in \mathbf{X}$, i.e.

$$\boldsymbol{\Sigma}_{ij} = cov(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{E}\left[(\mathbf{x}_i - \mu_i)^T(\mathbf{x}_j - \mu_j)\right]$$

- If the training data are orthogonal, the covariance matrix becomes diagonal

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_d^2 \end{pmatrix}$$

- If the data are homogenous (i.e. $\sigma_1 = \sigma_2 = \cdots = \sigma_d = \sigma$)
  - $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ where $\mathbf{I}$ is the identity matrix
  - $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\sigma^2 = \frac{1}{Nd}\sum_{i=1}^{N}(x_i - \mu)^T(x_i - \mu)$

2. Parametric Approach

- Parametric distribution models are restricted to specific forms, which may not always be suitable; for example, consider modelling a multimodal distribution with a single, unimodal model.

3. Non-Parametric Approach

- Building a model without clearly defined structure
- For a prediction all (training) data are needed
- Local density estimators ≈ estimate the density in a small region

$$p(x) = \frac{K}{NV}$$

where K is the number of points in the region, V is the volume of the region, N is the total number of points

- Two common methods
  A. Kernel approach
  B. K-nearest neighbour approach

3A. Kernel Density Estimators

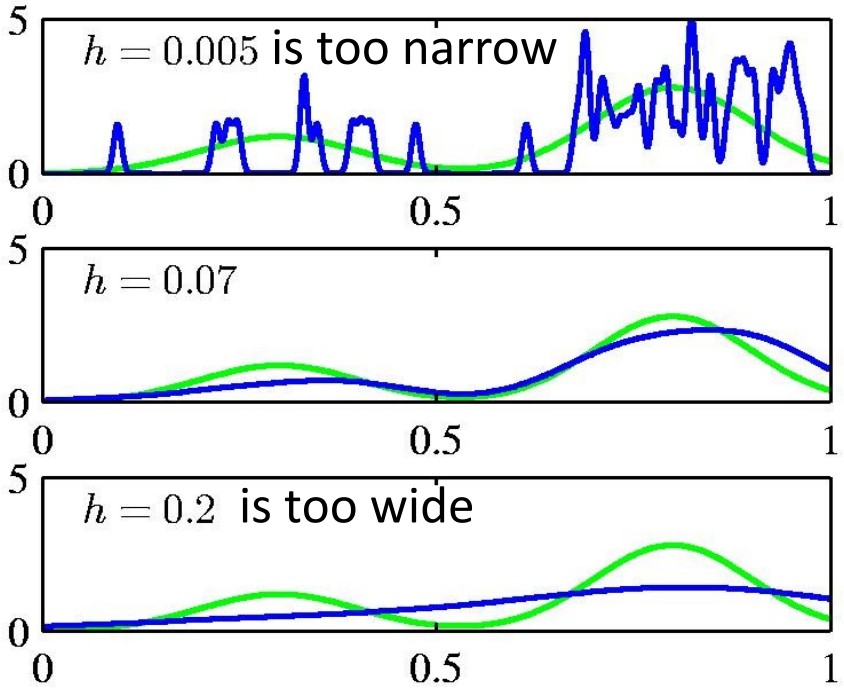
- Use regions centered on the data points
  - Allow regions to overlap
  - Each region contributes by $1/N$ to the total density
  - Avoiding discontinuities by using regions with soft edges

$$p(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{\|x - x_n\|^2}{2h^2}\right)$$

1D Gaussian probability density function $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

d dimensional

$$\prod_{i=1}^{N} p(x_i) \quad \text{where} \quad p(x_i) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{(x_i-\mu)^T \boldsymbol{\Sigma}^{-1} (x_i-\mu)}{2}}$$



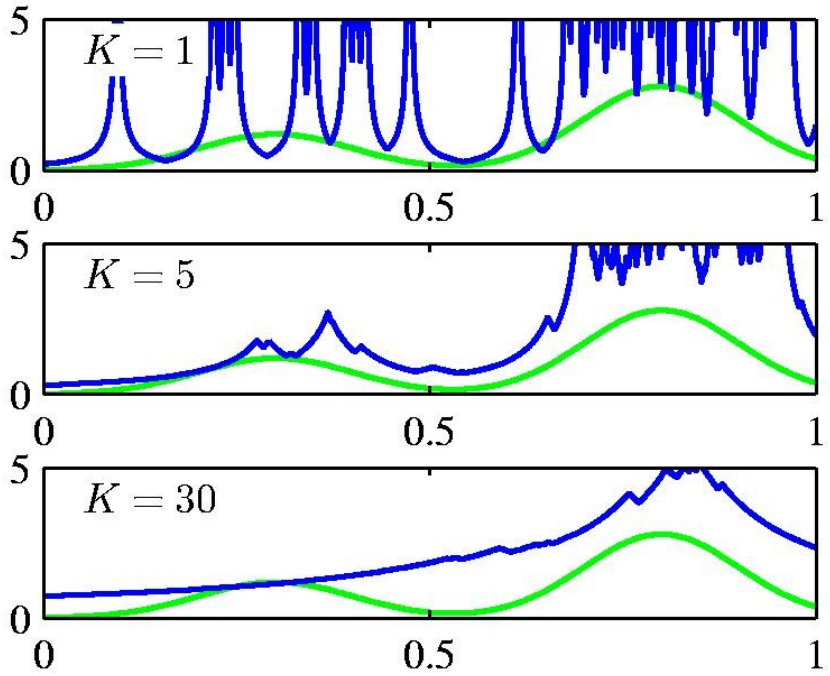
3B. *K*-Nearest Neighbor Approach for Density Estimation

- Similarly to the histogram approach
- Vary the size of a hyper-sphere around each test point so that exactly K training data points fall inside the hyper-sphere.
  - fix K, estimate V from the data. Consider a hyper-sphere centred on x and let it grow to a volume V^*, that includes K of the given N data points

$$p(x) = \frac{K}{NV^*}$$

  - Does this give a fair estimate of the density?
- Nearest neighbors is usually used for classification or regression:
  - For regression, average the predictions of the K-nearest neighbors.
  - For classification, pick the class with the most votes.

3B. *K*-Nearest Neighbour Approach for Density Estimation



K acts as a smoother.

4. Semi-Parametric Approach

- We assume that data are generated from a model with M Gaussians
- Estimated density

$$f(x) = \sum_{m=1}^{M} w_m G_m(x)$$

where w_m , $0 \leq w_m \leq 1$ is the contribution of the m-th component

$$\sum_{m=1}^{M} w_m = 1$$

- $G_m(x)$ is the m-th component Gaussian

$$G_m(x) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_m|}} e^{-\frac{(x_i-\mu)^T \boldsymbol{\Sigma}_m^{-1} (x_i-\mu)}{2}}$$

- For fitting of such mixture model, the Expectation-Maximization (EM) algorithm is used [EM-algorithm is not presented here]

Part 2: Unsupervised learning

DIMENSIONALITY REDUCTION

Why

- Reduces time complexity: Less computation
- Reduces space complexity: Less parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- Better interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc.) if plotted in 2 or 3 dimensions

➢ A (one-to-one) mapping ϕ of the set of data points $\mathcal{D} = \{x_n \in \mathbb{R}^d\}_{n=1}^N$ where N is the number of data points, $d \geq 2$ is their dimension, to $\widetilde{\mathcal{D}} = \left\{y_n \in \mathbb{R}^{\tilde{d}}\right\}_{n=1}^N$ and, where $\tilde{d} \leq 2$ is the new dimension;

$$\phi(x_n) = y_n, \qquad \forall n \in \{1, \ldots, N\}$$

➢ If x_m is the nearest neighbor of $\mathrm{x_n} \in \mathcal{D}$, we expect that

$$\|y_m - y_n\| \leq \|y_i - y_n\|, \quad \forall i \in \{1, \ldots, N\}$$

Feature Selection vs Extraction

- **Feature selection:**
  - Choosing $k < d$ important features, ignoring the remaining $d - k$
  - Subset selection algorithms
- **Feature extraction (feature derivation):**
  - Project the original $\mathbf{x}_i, i = 1, \ldots, d$ dimensions to new $k < d$ dimensions, $\mathbf{z}_j, \quad j = 1, \ldots, k$
  - Algorithms:
    - Principal components analysis (PCA)
    - Linear discriminant analysis (LDA)
    - Factor analysis (FA)
    - Multidimensional scaling (MDS)

Subset Selection

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d) = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nd} \end{pmatrix}$$

- There are 2^d subsets of d features

1. **Forward search:**
   - "Add the best feature at each step"
     - Set of features F initially $\varnothing$
     - At each iteration, find the best new feature
       $$j = \underset{i}{\operatorname{argmin}}\, Error(F \cup \mathbf{x}_i)$$
     - Add $\mathbf{x}_i$ to F if $Error(F \cup \mathbf{x}_i) < Error(F)$
   - Hill-climbing $O(d^2)$ algorithm
2. **Backward search**: Start with all features and remove one (which causes the least error) at a time
3. **Floating search**:
   - Set the set of selected features $F := \varnothing$
   - Set the set of possible features P to the set of all features
   - Alternate adding k features to F and removing l features from P

Principal Components Analysis (PCA)

- Find a low-dimensional space such that when x is projected there, information loss is minimized.
- The projection of x on the direction (a column vector) w_1 is:

$$z = w_1^{\mathrm{T}} x$$

$\mathrm{E}[x] = \mu$

- Find w_1 such that $\mathrm{Var}(z)$ is maximized

$$\mathrm{Var}(z) = \mathrm{Var}(w_1^{\mathrm{T}} x) = \mathrm{E}\left[\left(w_1^{\mathrm{T}} x - \mathrm{E}[w_1^{\mathrm{T}} x]\right)^2\right] = \mathrm{E}\left[\left(w_1^{\mathrm{T}} x - w_1^{\mathrm{T}} \mu\right)^2\right] =$$
$$= \mathrm{E}\left[\left(w_1^{\mathrm{T}} x - w_1^{\mathrm{T}} \mu\right)\left(w_1^{\mathrm{T}} x - w_1^{\mathrm{T}} \mu\right)\right] = E\left[w_1^{\mathrm{T}} (\mathrm{x} - \mu)(x - \mu)^{\mathrm{T}} w_1\right]$$
$$= w_1^{\mathrm{T}} \mathrm{E}[(x - \mu)(x - \mu)^{\mathrm{T}}] w_1 = w_1^{\mathrm{T}} \boldsymbol{\Sigma} w_1$$

where $\mathrm{Var}(x) = \mathrm{E}[(x - \mu)(x - \mu)^{\mathrm{T}}] = \boldsymbol{\Sigma}$ is the correlation matrix

Principal Components Analysis (PCA)

- Find a transformation matrix

$$\mathbf{W} = \left[\begin{pmatrix} \\ w_1 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} \\ w_k \end{pmatrix} \right]$$

we will use for a transformation

$$z = \mathbf{W}^T (x - \mu)$$

Principal Components Analysis (PCA)

- Without constraint we could pick a very big w_1.
- Maximize $\mathrm{Var}(z)$ subject to $\|w_1\| = 1$ using Lagrange multiplier method

$$\max_{w_1} w_1^{\mathrm{T}} \boldsymbol{\Sigma} w_1 - \alpha\left(w_1^{\mathrm{T}} w_1 - 1\right)$$

- The derivative with respect to w_1 (all partial derivatives according to constituents of w_1) should be 0

$$2\boldsymbol{\Sigma} w_1 - 2\alpha w_1 = 0$$

- Hence $\boldsymbol{\Sigma} w_1 = \alpha w_1$ that is, w_1 is an eigenvector of $\boldsymbol{\Sigma}$
- We want to maximize $\mathrm{Var}(z) = w_1^{\mathrm{T}} \boldsymbol{\Sigma} w_1 = \alpha w_1^{\mathrm{T}} w_1 = \alpha \implies$ choose the eigen vector with the largest eigenvalue
- Second principal component: $\max \mathrm{Var}(z_2)$, s.t. $\|w_2\| = 1$ and w_2 is orthogonal to w_1

$$\max_{\mathbf{w_2}} w_2^{\mathrm{T}} \boldsymbol{\Sigma} w_2 - \alpha\left(w_2^{\mathrm{T}} w_2 - 1\right) - \beta\left(w_2^{\mathrm{T}} w_1 - 0\right)$$

Principal Components Analysis (PCA)

- Second principal component: $\max \mathrm{Var}(z_2)$, s.t. $\|w_2\| = 1$ and w_2 is orthogonal to w_1

$$\max_{\mathbf{w_2}} w_2^{\mathrm{T}} \boldsymbol{\Sigma} w_2 - \alpha\left(w_2^{\mathrm{T}} w_2 - 1\right) - \beta\left(w_2^{\mathrm{T}} w_1 - 0\right)$$

- The derivative with respect to w_2 should be 0

$$(*) \qquad 2\boldsymbol{\Sigma} w_2 - 2\alpha w_2 - \beta w_1 = 0$$

- Pre-multiply by w_1

$$2w_1^{\mathrm{T}} \boldsymbol{\Sigma} w_2 - 2\alpha w_1^{\mathrm{T}} w_2 - \beta w_1^{\mathrm{T}} w_1 = 0$$

$$2w_1^{\mathrm{T}} \boldsymbol{\Sigma} w_2 - \beta = 0$$

- $w_1^{\mathrm{T}} w_2 = 0$, $w_1^T \boldsymbol{\Sigma} w_2$ is a scalar and equals to the transpose $w_2^T \boldsymbol{\Sigma} w_1$

$$2w_1^T \boldsymbol{\Sigma} w_2 = w_2^T \boldsymbol{\Sigma} w_1 = \lambda_1 w_2^T w_1 = 0$$

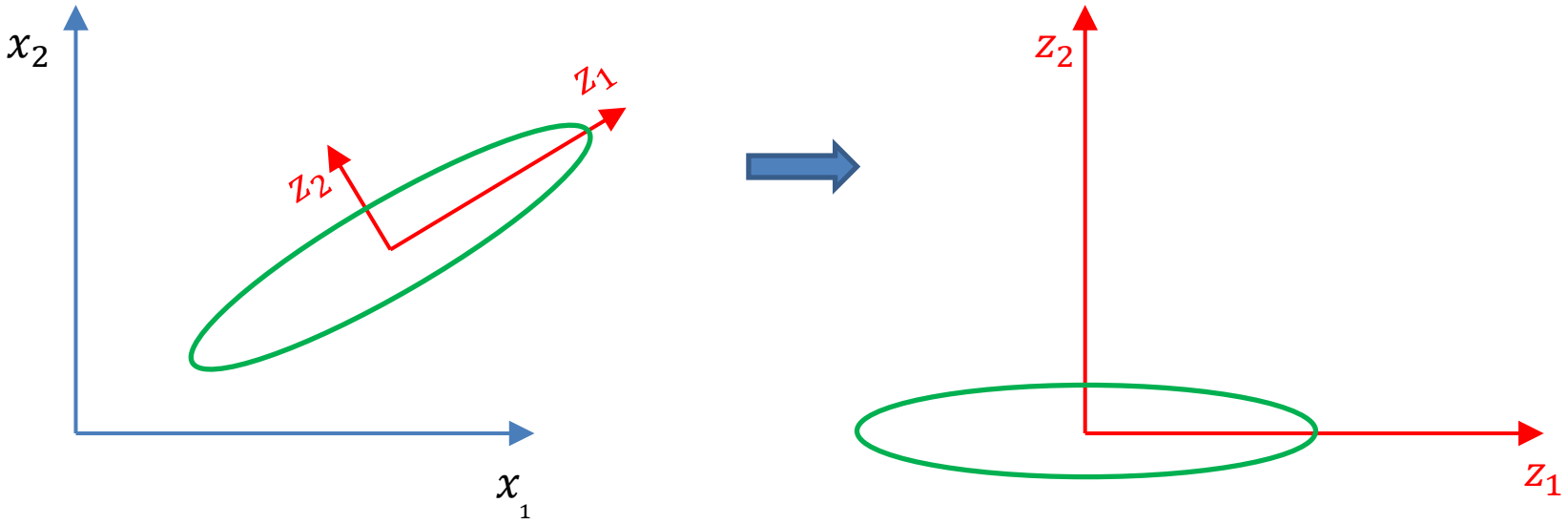
- Then $\beta = 0$ and from (*) we have $\boldsymbol{\Sigma} w_2 = \alpha w_2$ and w_2 should be an eigenvector of $\boldsymbol{\Sigma}$
- Similarly for $w_3, w_4, \ldots$

What PCA Does?

$$z = \mathbf{W}^{\mathrm{T}}(x - \mu)$$

where the columns of $\boldsymbol{W}$ are the eigenvectors of $\boldsymbol{\Sigma}$, and μ is the sample mean

Centers the data at the origin and rotates the axes

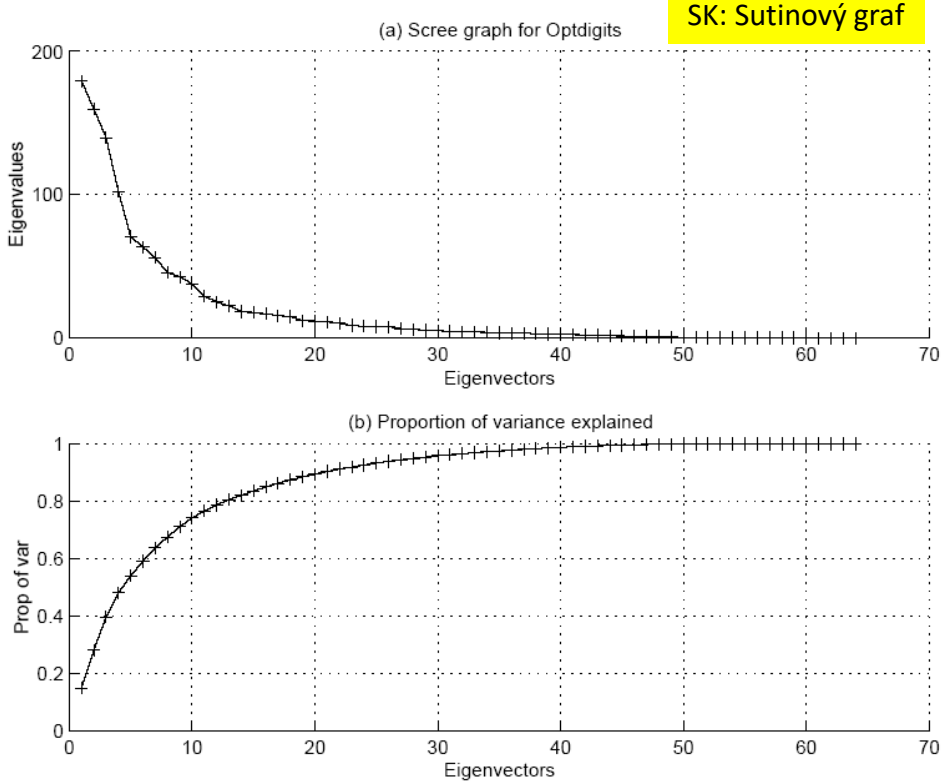


How to Choose the Number of Principal Components?

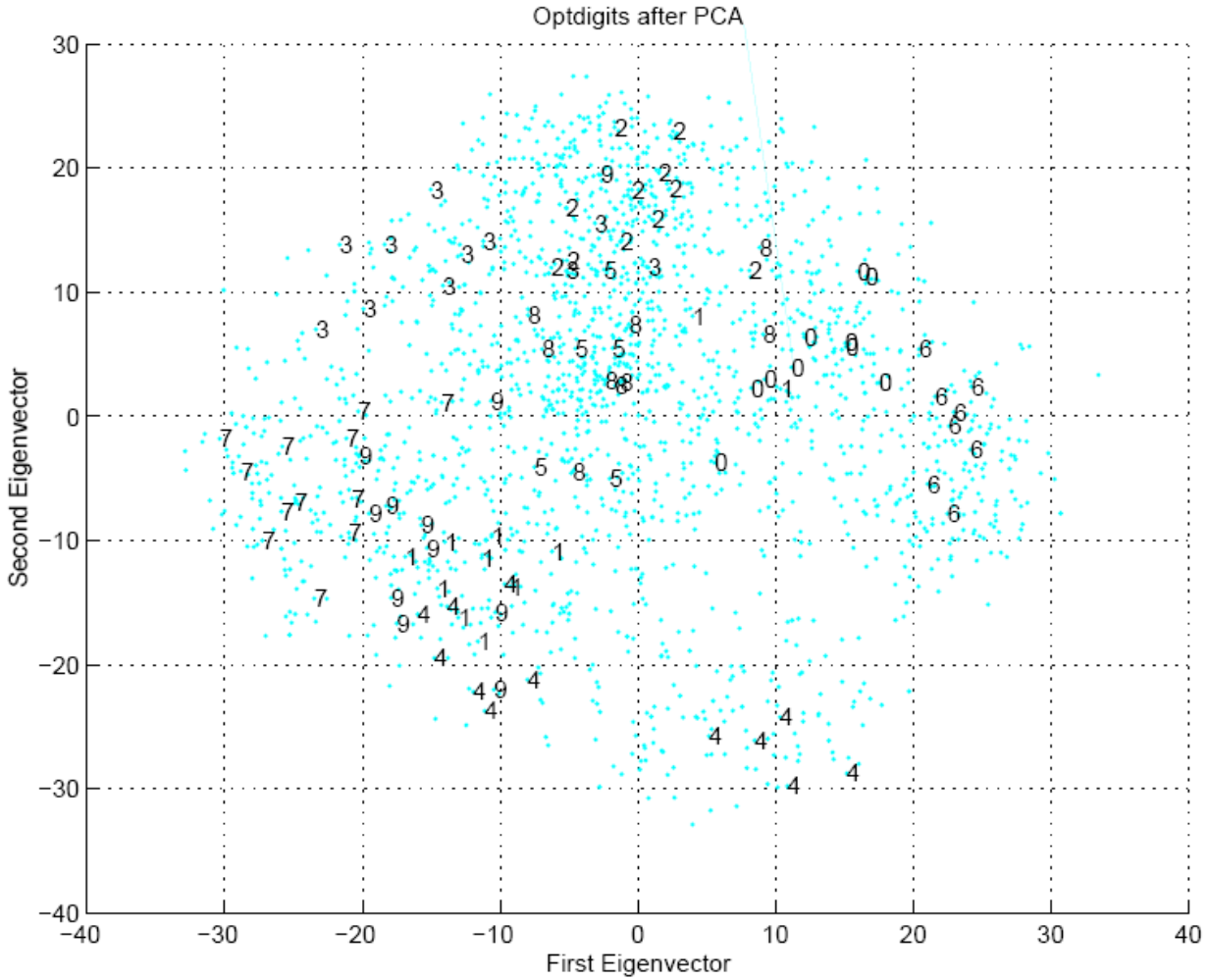
- Proportion of Variance (PoV) explained, when λ_i are sorted in descending order
- Typically, stop at $PoV > 0.9$
- Scree graph plots of PoV vs k, stop at "elbow"

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k \cdots + \lambda_d}$$

SK: Sutinový graf



PCA on Optidigits



Optidigits plotted into the first two dimensions found by PCA

Factor Analysis

- Find a small number of factors $\mathbf{z} \in \mathbb{R}^N$, which when combined generate $\mathbf{x}$:

$$\mathbf{x}_i - \mu_i = v_{i1}\mathbf{z}_1 + v_{i2}\mathbf{z}_2 + \cdots + v_{ik}\mathbf{z}_k + \boldsymbol{\varepsilon}_i$$

where $\mathbf{z}_j, j = 1, \ldots, k$ $(k < d)$ are the latent factors with

$$\mathrm{E}[\mathbf{z}_j] = 0, \quad \mathrm{Var}(\mathbf{z}_j) = 1, \quad \mathrm{Cov}(\mathbf{z}_i, \mathbf{z}_j) = 0, \quad i \neq j,$$

$\boldsymbol{\varepsilon}_i$ are the noise sources

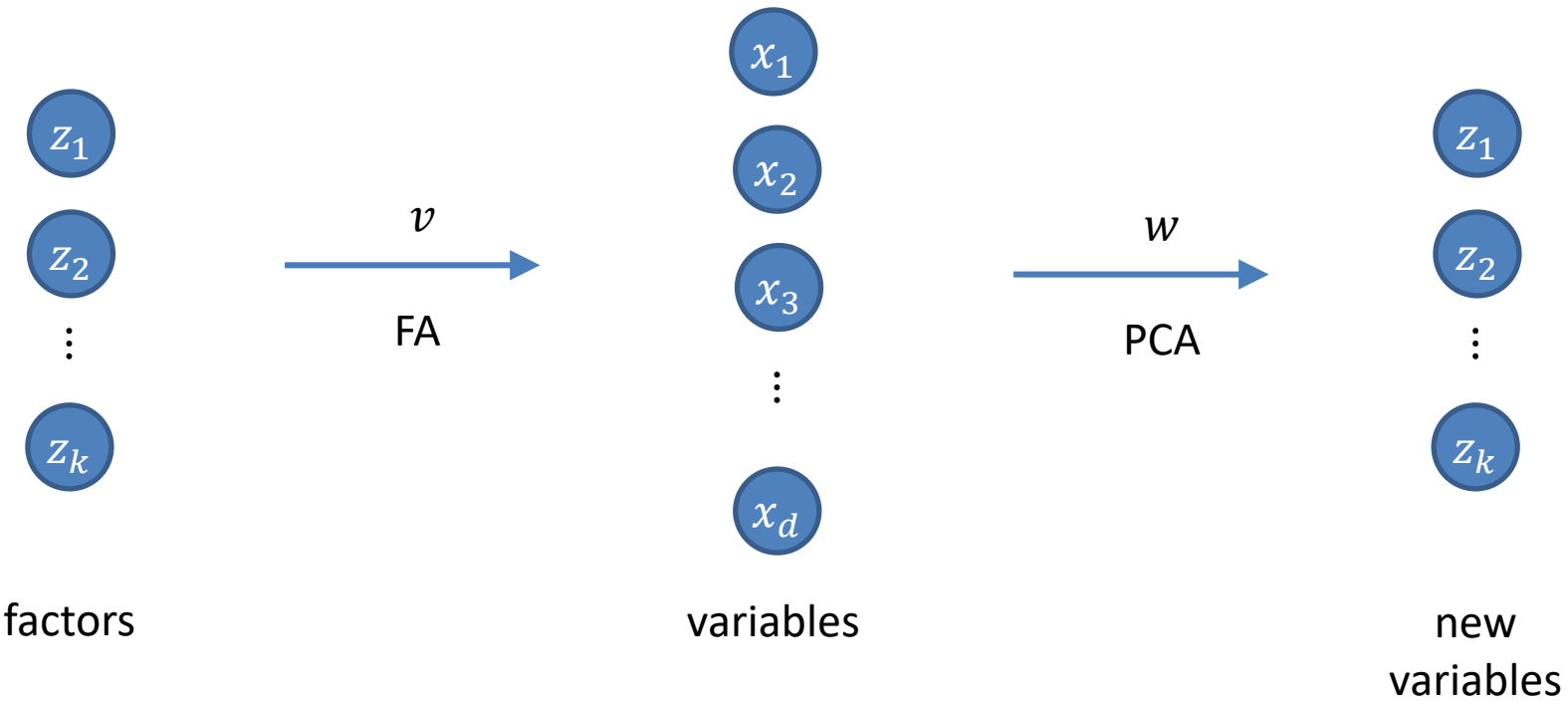
$$\mathrm{E}[\boldsymbol{\varepsilon}_i] = \psi_i; \ \mathrm{Cov}(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_j) = 0, \quad i \neq j; \quad \mathrm{Cov}(\boldsymbol{\varepsilon}_i, \mathbf{z}_j) = 0, \forall i, j$$

and v_{ij} are the factor loadings, $\mathbf{V} = (v_{ij})_{d \times k}$

PCA vs FA

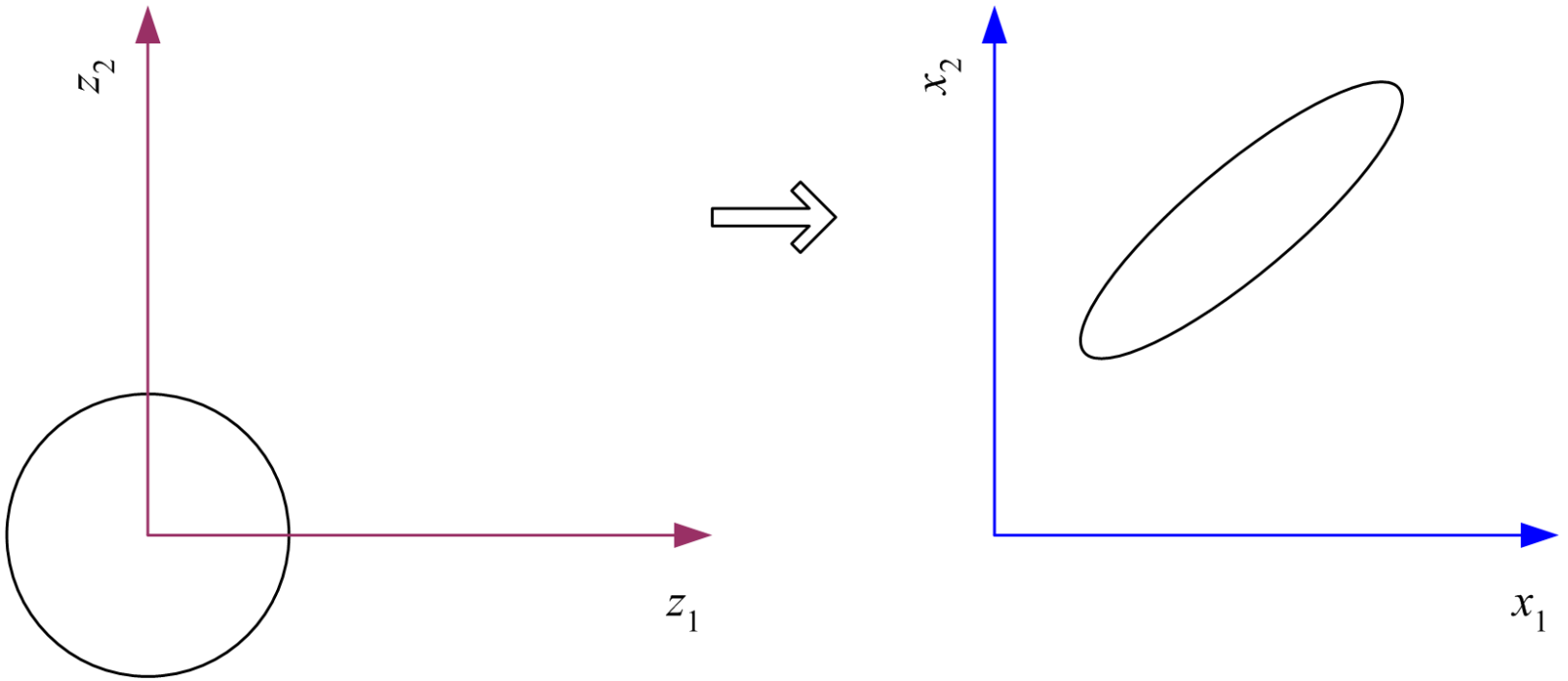
How to transform a vector x into new vector z?

- FA From z to x $x - \mu = \mathbf{V}z + \varepsilon$
- PCA From x to z $z = \mathbf{W}^{\mathrm{T}}(x - \mu)$



Factor Analysis

- In FA, factors z_j are stretched, rotated and translated to generate x



Multidimensional Scaling

- Given pairwise distances between N points

$$d^*_{i,j} = \left\| x_i - x_j \right\| \qquad i,j = 1,\dots,N$$

  place the points on a low-dim map s.t. distances are preserved.

- $z = g(x \mid \theta)$
  1. Classical MDS: $z = g(x \mid \theta) = \mathbf{W}^{\mathrm{T}} x$, i.e. $\theta = \mathbf{W}$
  2. In general: find θ that minimizes Sammon stress

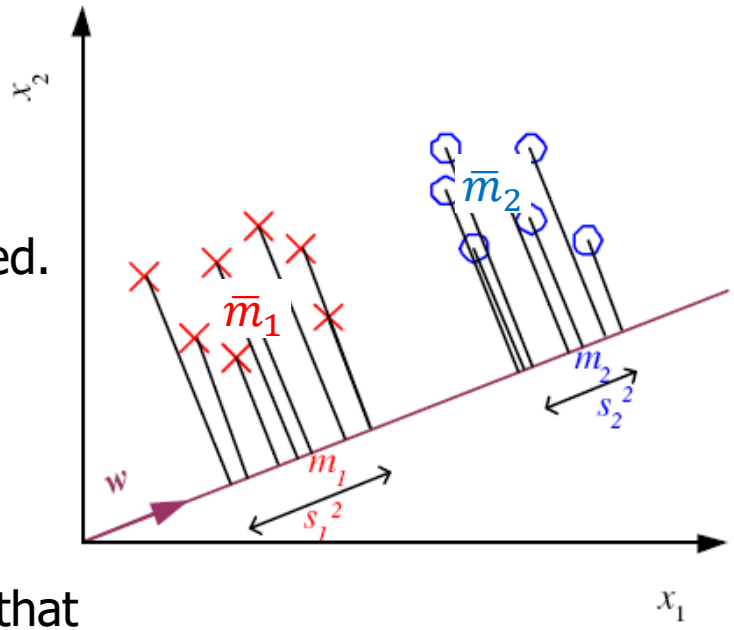
$$\mathrm{E}[\theta|\mathrm{X}] = \frac{1}{\sum_{i<j} d^*_{i,j}} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{\left(d^*_{i,j} - \left\| z_i - z_j \right\|\right)^2}{d^*_{i,j}}$$

  $\mathrm{E}[\theta|\mathrm{X}]$ is minimized using gradient descent methods or some other iterative method

Linear Discriminant Analysis

- a **supervised** method for dimensionality reduction (it is almost classification)
- Find a low-dimensional space such that when x is projected, classes are well-separated.
- Find w that maximizes

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$



- We are given a sample $X = \{(x_t, r_t)\}_{t=1}^{N}$ such that

$$r_t = \begin{cases} 1 & \text{if } x_t \in C_1 \\ 0 & \text{if } x_t \in C_2 \end{cases}$$

$$m_1 = w^{\mathrm{T}}\bar{m}_1 = \frac{\sum_t w^{\mathrm{T}} x_t r_t}{\sum_t r_t} \qquad s_1^2 = \sum_t \left(\mathrm{w}^{\mathrm{T}} x_t - m_1\right)^2 r_t$$

$$m_2 = w^{\mathrm{T}}\bar{m}_2 = \frac{\sum_t w^{\mathrm{T}} x_t (1 - r_t)}{\sum_t (1 - r_t)} \qquad s_2^2 = \sum_t \left(w^{\mathrm{T}} x_t - m_2\right)^2 (1 - r_t)$$

Linear Discriminant Analysis

- Between-class scatter:

$$(m_1 - m_2)^2 = \left(w^{\mathrm{T}}\bar{m}_1 - w^{\mathrm{T}}\bar{m}_2\right)^2$$
$$= w^{\mathrm{T}}(\bar{m}_1 - \bar{m}_2)(\bar{m}_1 - \bar{m}_2)^{\mathrm{T}}w$$
$$= w^{\mathrm{T}}\mathbf{S}_B w \text{ where } \mathbf{S}_B = (\bar{m}_1 - \bar{m}_2)(\bar{m}_1 - \bar{m}_2)^{\mathrm{T}}$$

- Within-class scatter:

$$s_1^2 = \sum_t \left(w^{\mathrm{T}}x_t - m_1\right)^2 r_t$$
$$= \sum_t w^{\mathrm{T}}(x_t - \bar{m}_1)(x_t - \bar{m}_1)^{\mathrm{T}}wr_t = w^{\mathrm{T}}\mathbf{S}_1 w$$
$$\text{where } \mathbf{S}_1 = \sum_t (x_t - \bar{m}_1)(x_t - \bar{m}_1)^{\mathrm{T}}r_t \quad \text{for } \mathbf{S}_2 \text{ analogically}$$
$$s_1^2 + s_2^2 = w^{\mathrm{T}}\mathbf{S}_W w \quad \text{where } \mathbf{S}_W = \mathbf{S_1} + \mathbf{S}_2$$

Fisher's Linear Discriminant

- Find w that max

$$J(w) = \frac{w^T \mathbf{S}_B w}{w^T \mathbf{S}_W w} = \frac{\left| w^T (\bar{m}_1 - \bar{m}_2) \right|^2}{w^T \mathbf{S}_W w}$$

- LDA solution (the derivative of $J(w)$ must be zero):

$$w = c \cdot \mathbf{S}_W^{-1} (\bar{m}_1 - \bar{m}_2) \text{ for some constant } c$$

we can take $c = 1$

- Parametric solution:

$$w = \mathbf{\Sigma}^{-1}(\mu_1 - \mu_2) \quad \text{when } p(x|C_i) \sim \mathcal{N}(\mu_i, \mathbf{\Sigma})$$

Normal distribution with mean μ_i and covariance matrix $\mathbf{\Sigma}$

- Moreover, this solution can be used also when the classes are not normal

K>2 Classes

- Classes $C_1, \ldots, C_K$ $$r_t^{(i)} = \begin{cases} 1 & \text{if } x_t \in C_i \\ 0 & \text{if } x_t \notin C_i \end{cases}$$
- We want to map d-dimensrional space into k-dimensional space
$$z = \mathbf{W}^{\mathrm{T}} x \quad \mathbf{W} \in \mathbb{R}^{d \times k}, z \in \mathbb{R}^k$$
- Total within-class scatter matrix:
$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i \quad \mathbf{S}_i = \sum_t r_t^{(i)} (x_t - m_i)(x_t - m_i)^{\mathrm{T}}$$

Within-class scatter matrix for C_i

- Between-class scatter matrix:
$$\mathbf{S}_B = \sum_{i=1}^{K} N_i (m_i - m)(m_i - m)^{\mathrm{T}} \quad m = \frac{1}{K} \sum_{i=1}^{K} m_i \quad N_i = \sum_{i=1}^{N} r_t^{(i)}$$

K>2 Classes

- The between-class and within-class scatter matrices after projection are

$$\mathbf{W}^{\mathrm{T}}\mathbf{S}_B\mathbf{W} \quad \text{and} \quad \mathbf{W}^{\mathrm{T}}\mathbf{S}_W\mathbf{W}$$

- Find $\mathbf{W}$ that maximalizes

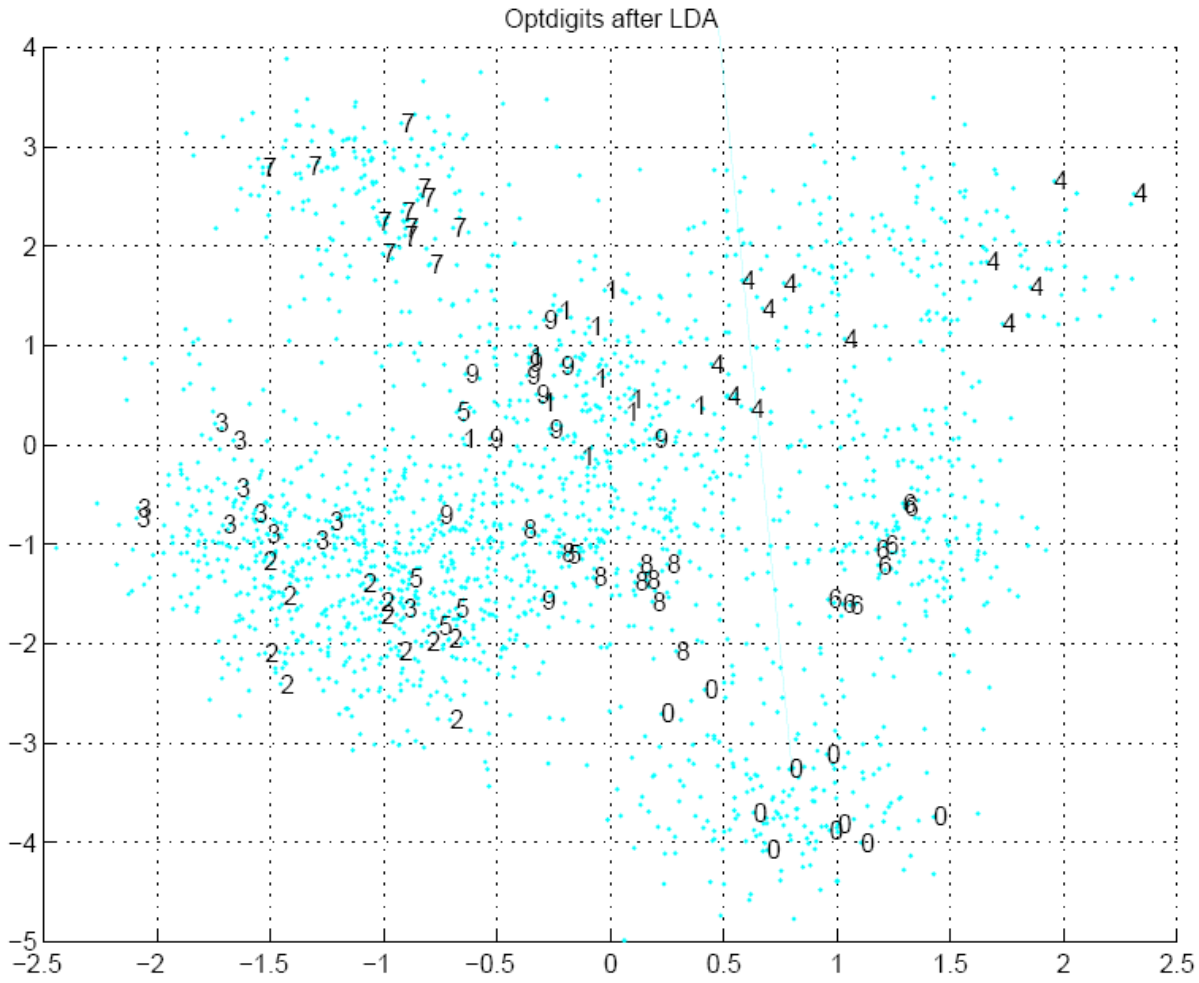
$$J(\mathbf{W}) = \frac{\left|\mathbf{W}^{\mathrm{T}}\mathbf{S}_B\mathbf{W}\right|}{\left|\mathbf{W}^{\mathrm{T}}\mathbf{S}_W\mathbf{W}\right|}$$

- The solution are the largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i(m_i - m)(m_i - m)^{\mathrm{T}}$$

is a sum of K matrices of rank 1, only $K-1$ of them are independent, therefore we take $k = K - 1$

Linear Discriminant Analysis

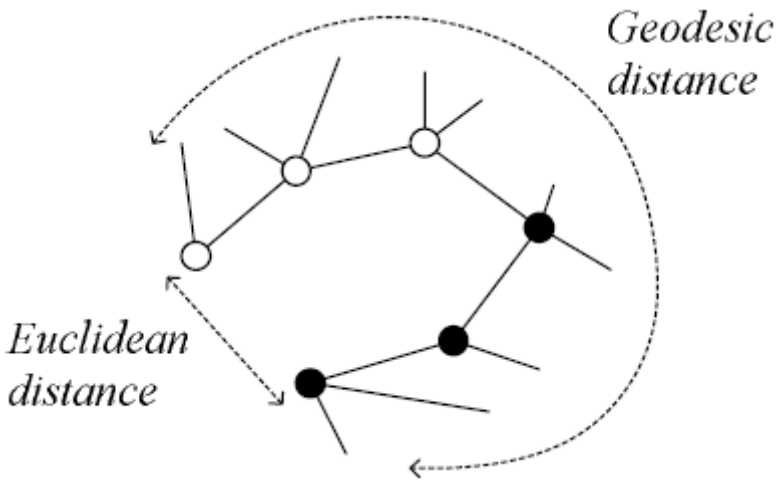


Optidigits plotted into the first two dimensions found by linear discriminant analysis

Note that the classes are better separated than in the case of PCA

Isomap

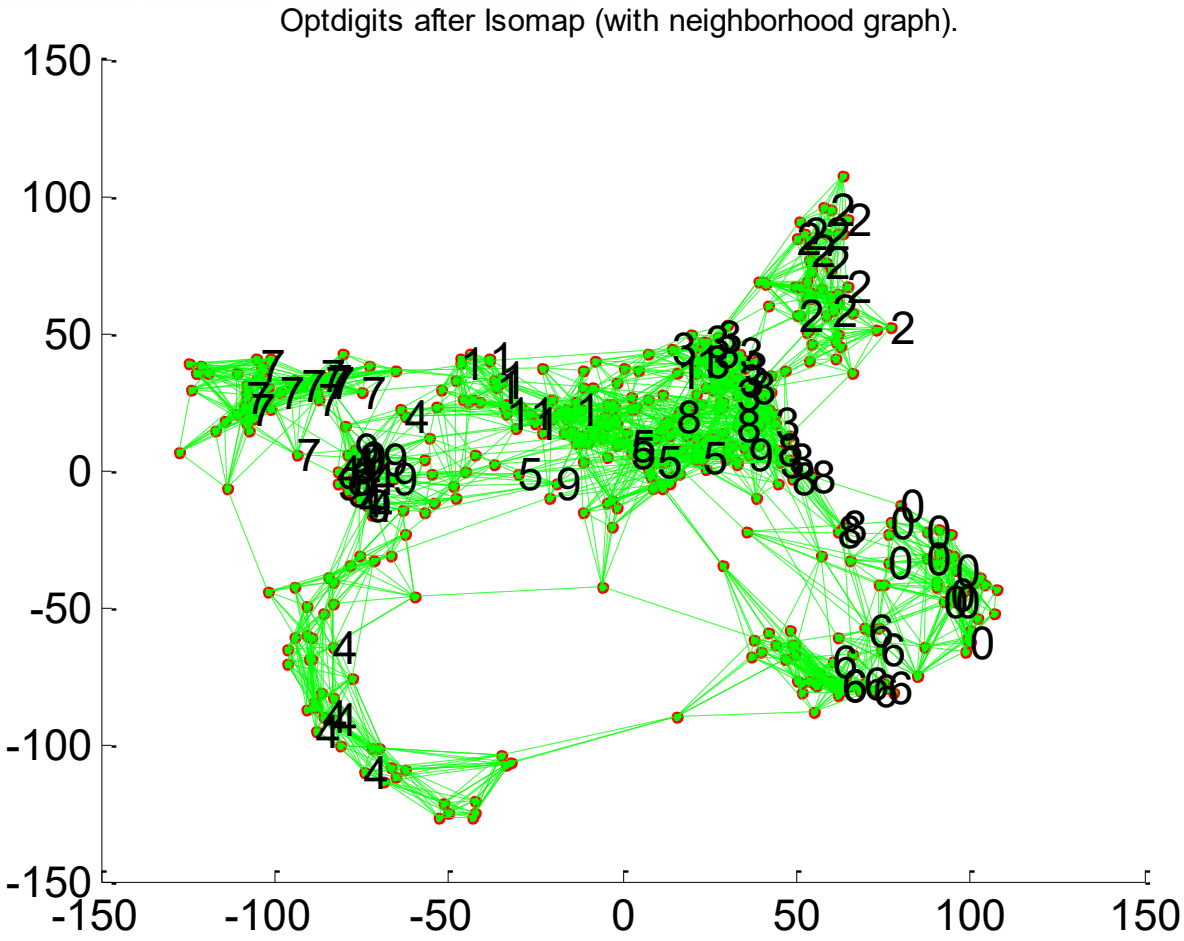
- Geodesic distance is the distance along the manifold that the data lies in, as opposed to the Euclidean distance in the input space



Isomap

- Instances r and s are connected in the graph if $\|x_r - x_s\| < \varepsilon$ or if x_s is one of the k neighbors of x_r. The edge length is $\|x_r - x_s\|$
- For two nodes r and s not connected, the distance is equal to the shortest path between them
- Once the $N \times N$ distance matrix is thus formed, use MDS to find a lower-dimensional mapping

Isomap – Example



Matlab source from http://web.mit.edu/cocosci/isomap/isomap.html

Locally Linear Embedding

1. Given x_r find its neighbors $x_s^{(r)}, s = 1, \ldots, S$
2. Find $\mathbf{W}_{r \times s}$ that minimize error in the original space

$$\mathrm{E}(\mathbf{W} \mid X) = \sum_r \left\| x_r - \sum_s \mathbf{W}_{rs} x_s^{(r)} \right\|^2$$

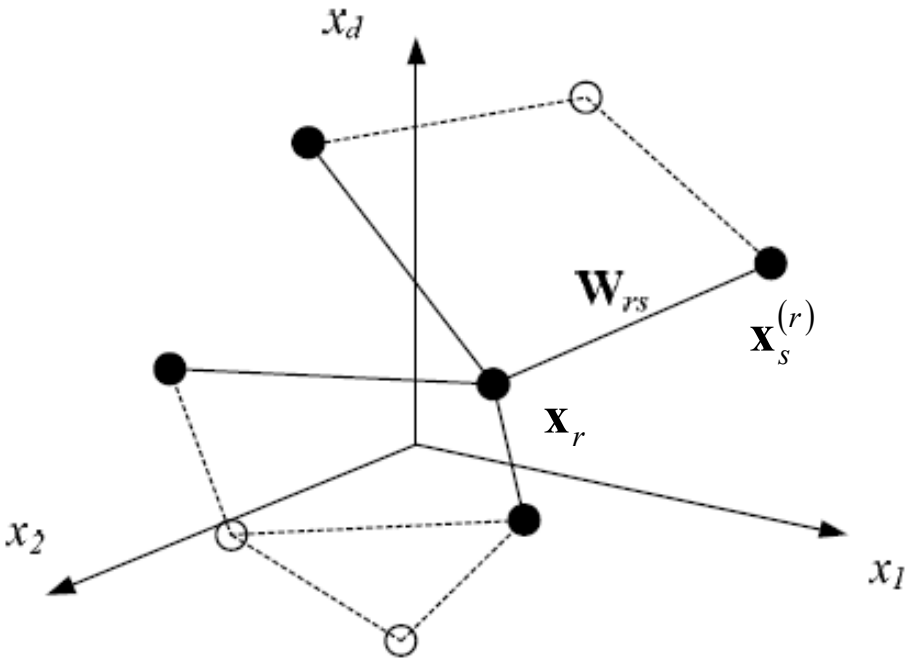
using least squares, subject to

$$\mathbf{W}_{rr} = 0, \qquad \sum_s \mathbf{W}_{rs} = 1, \qquad \forall r$$

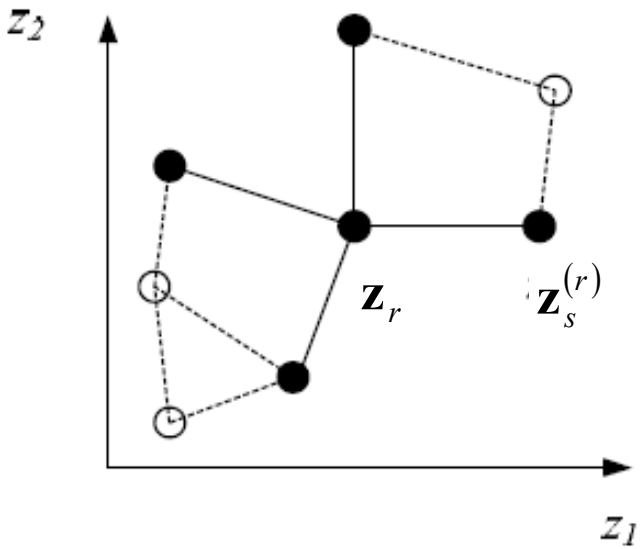
3. Find the new coordinates z_r that minimize

$$E(z \mid \mathbf{W}) = \sum_r \left\| z_r - \sum_s \mathbf{W}_{rs} z_s^{(r)} \right\|^2$$

Locally Linear Embedding

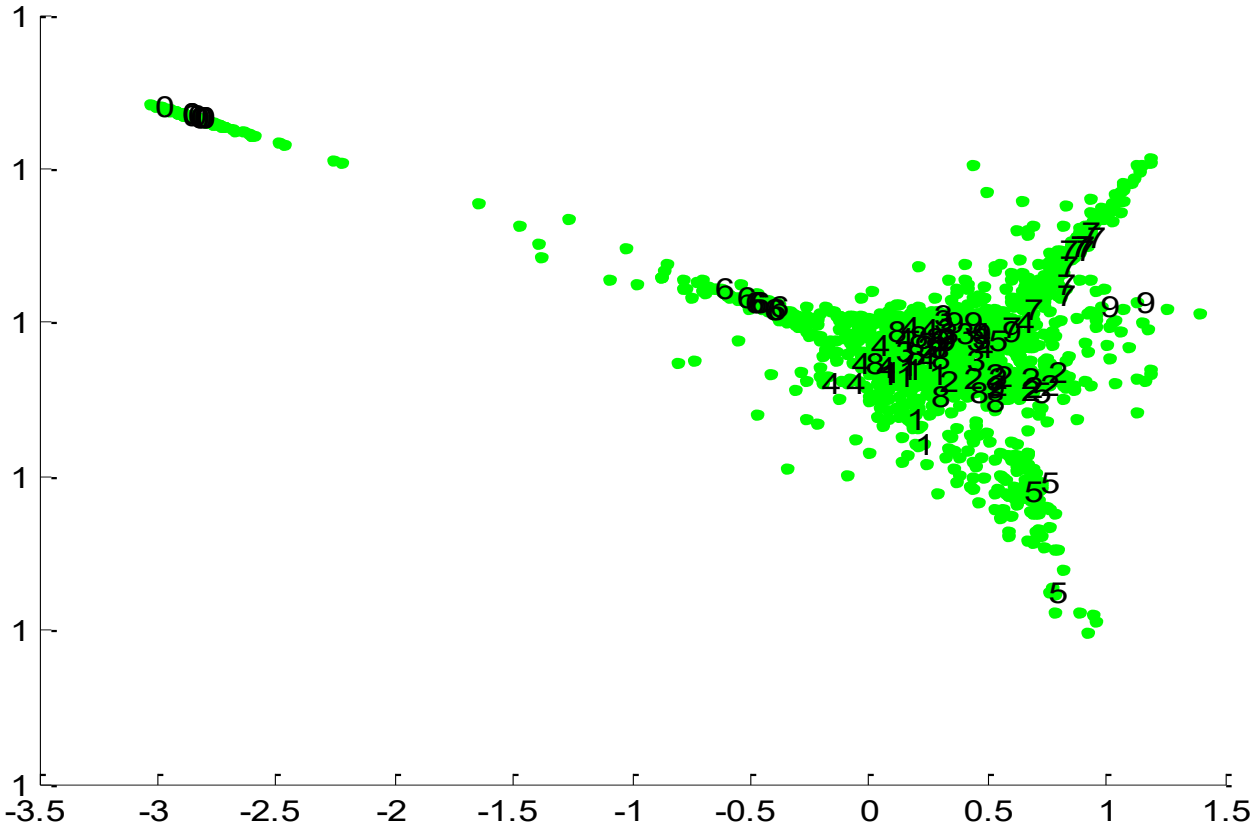


x space



z space

Locally Linear Embedding on Optdigits



Matlab source from http://www.cs.toronto.edu/~roweis/lle/code.html