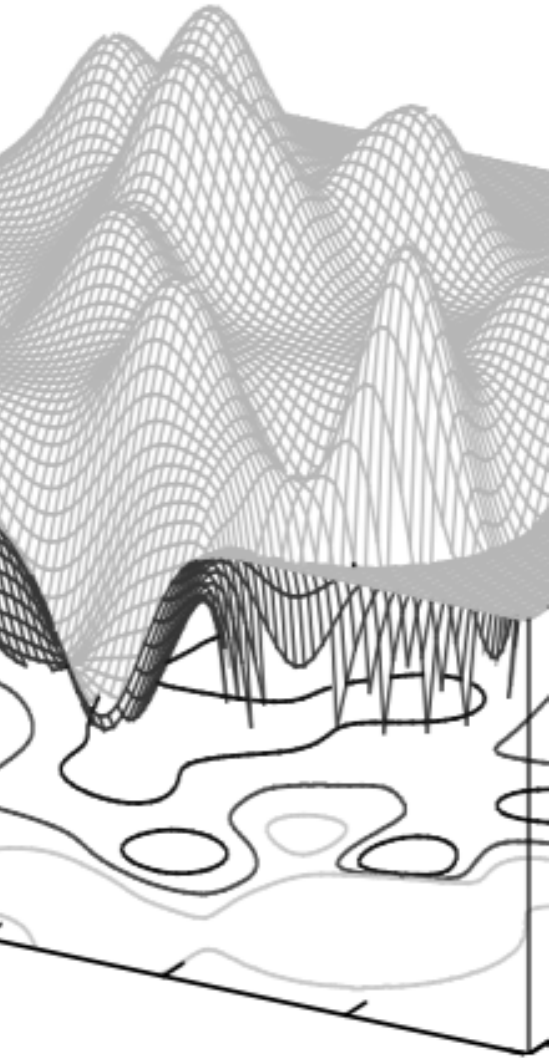


Machine Learning

Unsupervised Learning

PCA, kernel PCA Spectral Clustering, Multidimensional Scaling, ISOMAP Non-negative Matrix Factorization, Factor Analysis*, ICA*, PLS*, Clustering, k-means, Gaussian Mixture model Agglomerative Hierarchical Clustering*

Marc Toussaint
University of Stuttgart
Summer 2019

Unsupervised learning

- What does that mean? Generally: modelling $P(x)$
- Instances:
  - Finding lower-dimensional spaces
  - Clustering
  - Density estimation
  - Fitting a graphical model
- “Supervised Learning as special case”...

Principle Component Analysis (PCA)

- Assume we have data $D = \{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$.

  Intuitively: "We believe that there is an **underlying lower-dimensional space** explaining this data".

- How can we formalize this?

PCA: minimizing projection error

- For each $x_i \in \mathbb{R}^d$ we postulate a lower-dimensional latent variable $z_i \in \mathbb{R}^p$

$$x_i \approx V_p z_i + \mu$$

- Optimality:
  Find V_p, μ and values z_i that minimize $\sum_{i=1}^{n} \|x_i - (V_p z_i + \mu)\|^2$

Optimal V_p

$$\hat{\mu}, \hat{z}_{1:n} = \underset{\mu, z_{1:n}}{\operatorname{argmin}} \sum_{i=1}^{n} \|x_i - V_p z_i - \mu\|^2$$

$$\Rightarrow \hat{\mu} = \langle x_i \rangle = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} x_i \ , \quad \hat{z}_i = V_p^\top (x_i - \mu)$$

Optimal V_p

$$\hat{\mu}, \hat{z}_{1:n} = \underset{\mu, z_{1:n}}{\operatorname{argmin}} \sum_{i=1}^{n} \|x_i - V_p z_i - \mu\|^2$$

$$\Rightarrow \hat{\mu} = \langle x_i \rangle = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} x_i \ , \quad \hat{z}_i = V_p^\top (x_i - \mu)$$

- Center the data $\tilde{x}_i = x_i - \hat{\mu}$. Then

$$\hat{V}_p = \underset{V_p}{\operatorname{argmin}} \sum_{i=1}^{n} \|\tilde{x}_i - V_p V_p^\top \tilde{x}_i\|^2$$

Optimal V_p

$$\hat{\mu}, \hat{z}_{1:n} = \underset{\mu, z_{1:n}}{\operatorname{argmin}} \sum_{i=1}^{n} \|x_i - V_p z_i - \mu\|^2$$

$$\Rightarrow \hat{\mu} = \langle x_i \rangle = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} x_i \ , \quad \hat{z}_i = V_p^\top (x_i - \mu)$$

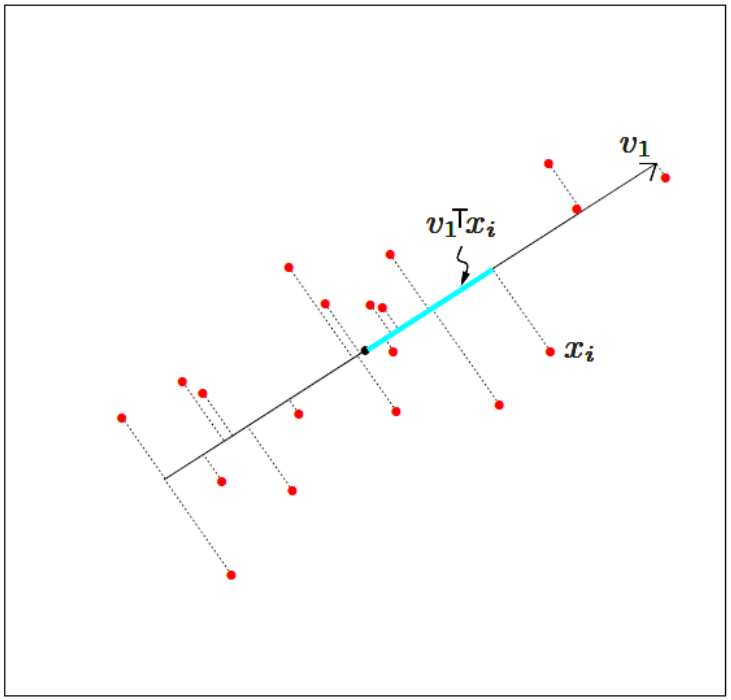
- Center the data $\tilde{x}_i = x_i - \hat{\mu}$. Then

$$\hat{V}_p = \underset{V_p}{\operatorname{argmin}} \sum_{i=1}^{n} \|\tilde{x}_i - V_p V_p^\top \tilde{x}_i\|^2$$

- Solution via Singular Value Decomposition
  - Let $X \in \mathbb{R}^{n \times d}$ be the centered data matrix containing all $\tilde{x}_i$
  - We compute a sorted Singular Value Decomposition $X^\top X = V D V^\top$
    D is diagonal with sorted singular values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
    $V = (v_1 \ v_2 \ \cdots \ v_d)$ contains largest eigenvectors v_i as columns

$$V_p := V_{1:d,1:p} = (v_1 \ v_2 \ \cdots \ v_p)$$

Principle Component Analysis (PCA)



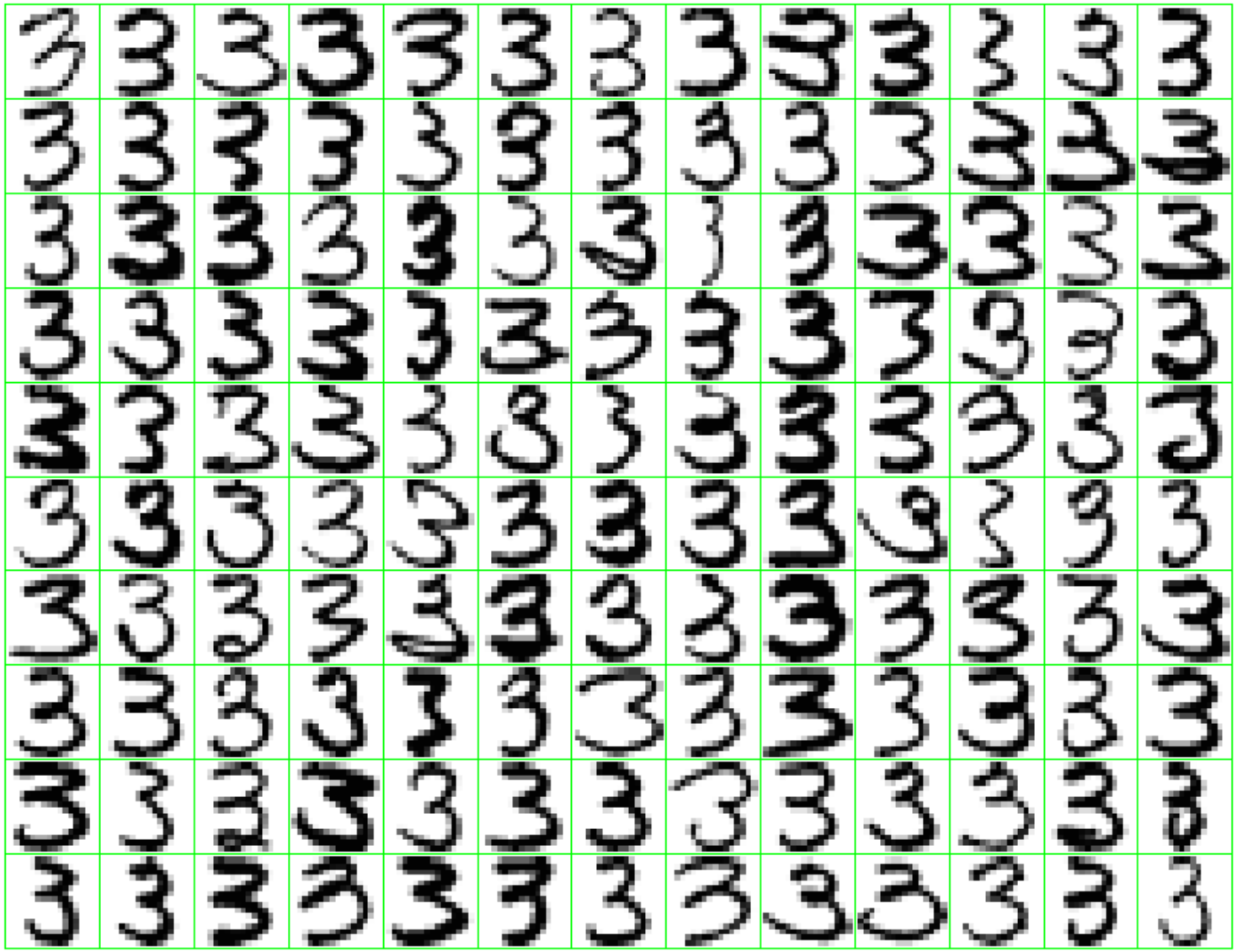
$V_p^\top$ is the matrix that projects to the largest variance directions of $X^\top X$

$$z_i = V_p^\top (x_i - \mu) \ , \quad Z = X V_p$$

- In non-centered case: Compute SVD of variance

$$A = \text{Var}\{x\} = \langle x x^\top \rangle - \mu\mu^\top = \frac{1}{n} X^\top X - \mu\mu^\top$$

Example: Digits



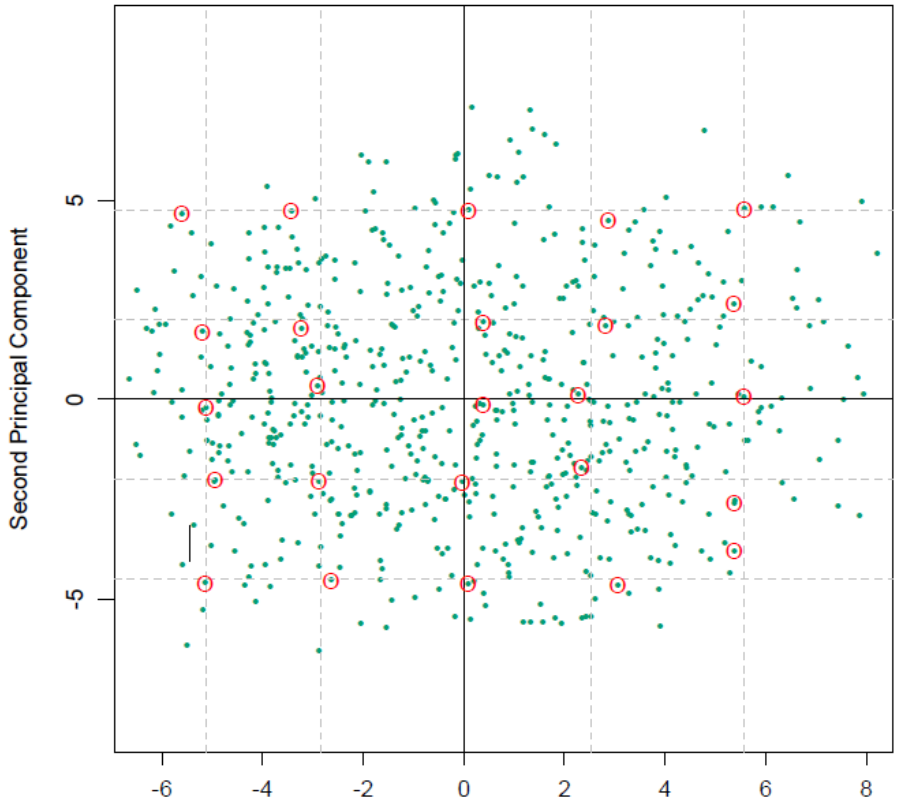
Example: Digits

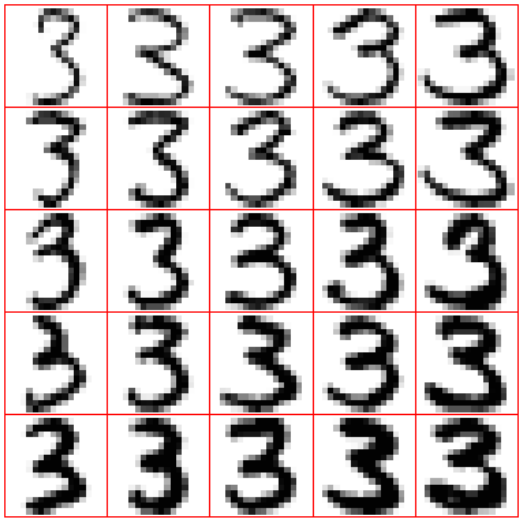
- The "basis vectors" in V_p are also **eigenvectors**
  Every data point can be expressed in these eigenvectors

$$x \approx \mu + V_p z$$

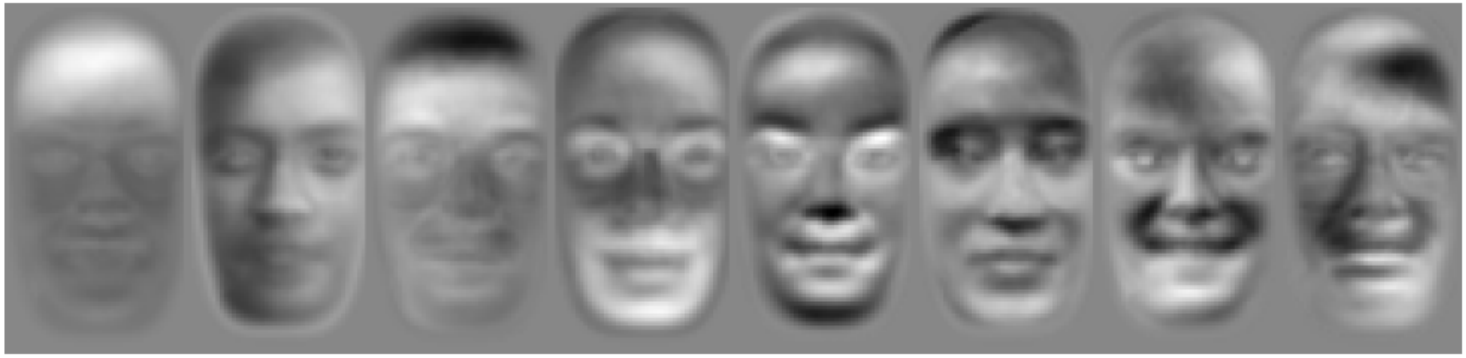
$$= \mu + z_1 v_1 + z_2 v_2 + \dots$$

$$= \boxed{3} + z_1 \cdot \boxed{3} + z_2 \cdot \boxed{3} + \cdots$$





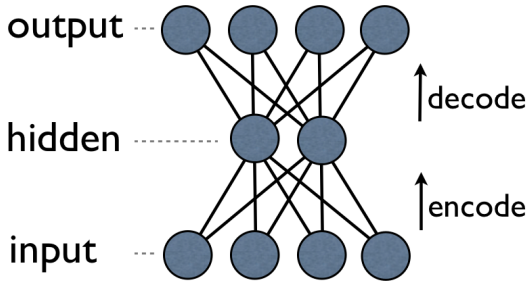
Example: Eigenfaces



(Viola & Jones)

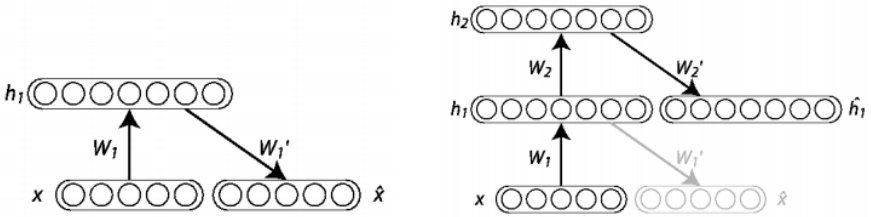
Non-linear Autoencoders

- PCA given the "optimal linear autoencode"
- We can relax the encoding (V_p) and decoding ($V_p^\top$) to be non-linear mappings, e.g., represented as a neural network



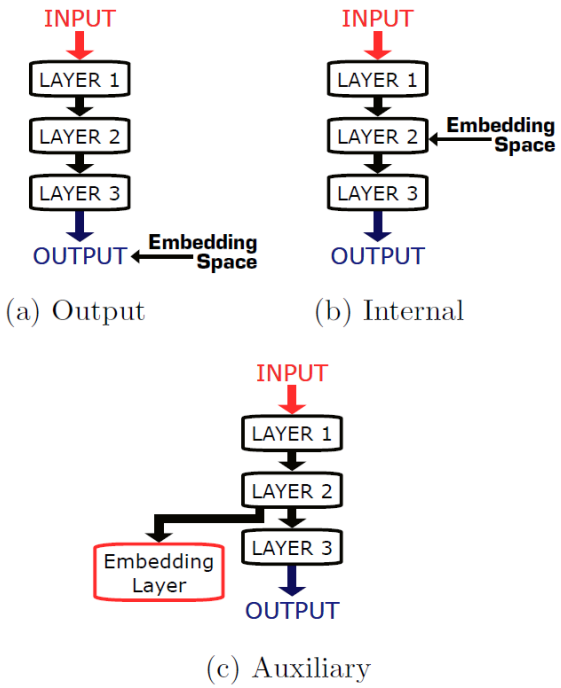
A NN which is trained to reproduce the input: $\min_i \|y(x_i) - x_i\|^2$
The hidden layer ("bottleneck") needs to find a good representation/compression.

- Stacking autoencoders:



Augmenting NN training with semi-supervised embedding objectives

- Weston et al. (ICML, 2008)



(a) Output (b) Internal

(c) Auxiliary

Mnist1h dataset, deep NNs of 2, 6, 8, 10 and 15 layers; each hidden layer 50 hidden units

	2	4	6	8	10	15
NN	26.0	26.1	27.2	28.3	34.2	47.7
$Embed^{O}$NN	19.7	15.1	15.1	15.0	13.7	11.8
$Embed^{ALL}$NN	18.2	12.6	7.9	8.5	6.3	9.3

What are good representations?

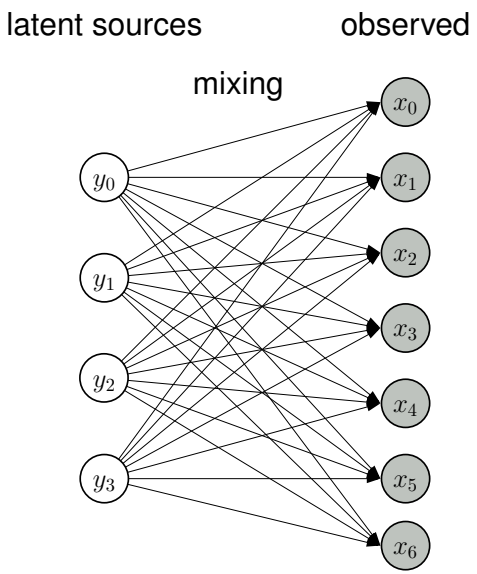
- Reproducing/autoencoding data, maintaining maximal information
- Disentangling correlations (e.g., ICA)
- those that are most correlated with desired *outputs* (PLS, NNs)
- those that maintain the clustering
- those that maintain relative distances (MDS)

...

- those that enable efficient reasoning, decision making & learning in the real world
- How do we represent our 3D environment, enabeling physical & geometric reasoning?
- How do we represent things to enable us inventing novel things, machines, technology, science?

Independent Component Analysis*

- Assume we have data $D = \{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$.
  PCA: $P(x_i \,|\, z_i) = \mathcal{N}(x_i \,|\, W z_i + \mu, \mathbf{I})$, $\quad P(z_i) = \mathcal{N}(z_i \,|\, 0, \mathbf{I})$
  Factor Analysis: $P(x_i \,|\, z_i) = \mathcal{N}(x_i \,|\, W z_i + \mu, \Sigma)$, $\quad P(z_i) = \mathcal{N}(z_i \,|\, 0, \mathbf{I})$
  ICA: $P(x_i \,|\, z_i) = \mathcal{N}(x_i \,|\, W z_i + \mu, \epsilon\mathbf{I})$, $\quad P(z_i) = \prod_{j=1}^d P(z_{ij})$



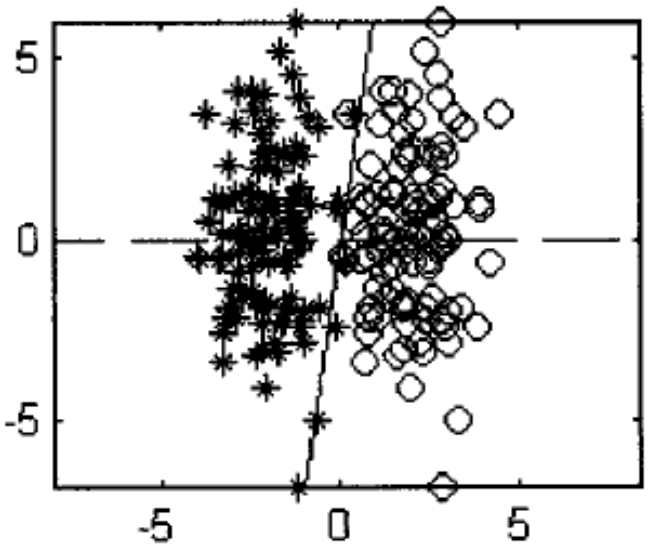
- In ICA
  1) We have (usually) as many latent variables as observed $\dim(x_i) = \dim(z_i)$
  2) We require all latent variables to be **independent**
  3) We allow for latent variables to be **non-Gaussian**

  Note: without point (3) ICA would be without sense!

Partial least squares (PLS)*

- Is it really a good idea to just pick the p-higest variance components??

  Why should that be a good idea?



PLS*

- Idea: The first dimension to pick should be the one **most correlated with the OUTPUT**, not with itself!

Input: data $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$
Output: predictions $\hat{y} \in \mathbb{R}^n$

1: initialize the *predicted output*: $\hat{y} = \langle y \rangle \, 1_n$
2: initialize the *remaining input dimensions*: $\hat{X} = X$
3: **for** $i = 1, .., p$ **do**
4: $\quad$ i-th input 'basis vector': $z_i = \hat{X}\hat{X}^\top y$
5: $\quad$ update prediction: $\hat{y} \leftarrow \hat{y} + Z_i y$ where $Z_i = \frac{z_i z_i^\top}{z_i^\top z_i}$
6: $\quad$ remove "used" input dimensions: $\hat{X} \leftarrow \hat{X}(\mathbf{I} - Z_i)$
7: **end for**

(Hastie, page 81)

Line 4 identifies a new input "coordinate" via maximal correlation between the remaning input dimensions and y.
Line 5 updates the prediction to include the project of y onto z_i
Line 6 removes the projection of input data $\hat{X}$ along z_i. All z_i will be orthogonal.

PLS for classification*

- Not obvious.
- We'll try to invent one in the exercises :-)

- back to linear autoencoding, i.e., PCA - but now linear in RKHS

“Feature PCA” & Kernel PCA

- The *feature* trick: $X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$

- The *kernel* trick: rewrite all necessary equations such that they only involve scalar products $\phi(x)^\top \phi(x') = k(x, x')$:

We want to compute eigenvectors of $X^\top X = \sum_i \phi(x_i)\phi(x_i)^\top$. We can rewrite this as

$$X^\top X v_j = \lambda v_j$$

$$\underbrace{XX^\top}_{K} \underbrace{Xv_j}_{K\alpha_j} = \lambda \underbrace{Xv_j}_{K\alpha_j}, \quad v_j = \sum_i \alpha_{ji} \phi(x_i)$$

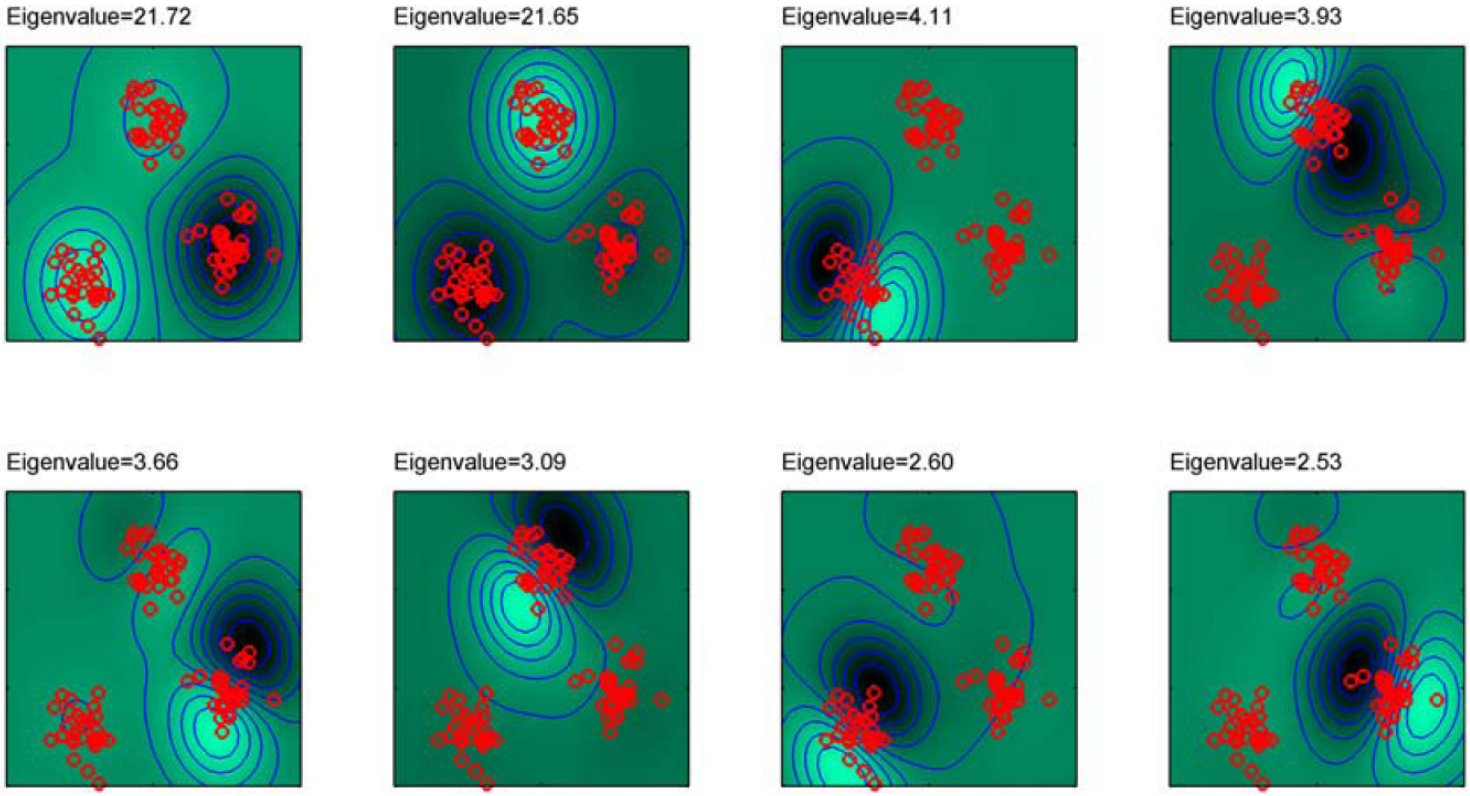
$$K\alpha_j = \lambda \alpha_j$$

Where $K = XX^\top$ with entries $K_{ij} = \phi(x_i)^\top \phi(x_j)$.
$\rightarrow$ We compute SVD of the kernel matrix $K \rightarrow$ gives eigenvectors $\alpha_j \in \mathbb{R}^n$.
Projection: $x \mapsto z = V_p^\top \phi(x) = \sum_i \alpha_{1:p,i} \phi(x_i)^\top \phi(x) = A\kappa(x)$
(with matrix $A \in \mathbb{R}^{p \times n}$, $A_{ji} = \alpha_{ji}$ and vector $\kappa(x) \in \mathbb{R}^n$, $\kappa_i(x) = k(x_i, x)$)
Since we cannot *center the features* $\phi(x)$ we actually need “the double centered kernel matrix” $\widetilde{K} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top) K (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$, where $K_{ij} = \phi(x_i)^\top \phi(x_j)$ is uncentered.

Kernel PCA

red points: data

green shading: eigenvector $\boldsymbol{\alpha}_j$ represented as functions $\sum_i \alpha_{ji} k(x_j, x)$



Kernel PCA “coordinates” allow us to discriminate clusters!

Kernel PCA

- Kernel PCA uncovers quite surprising structure:

  While PCA "merely" picks high-variance dimensions
  Kernel PCA picks high variance *features*—where features correspond to basis functions (RKHS elements) over x

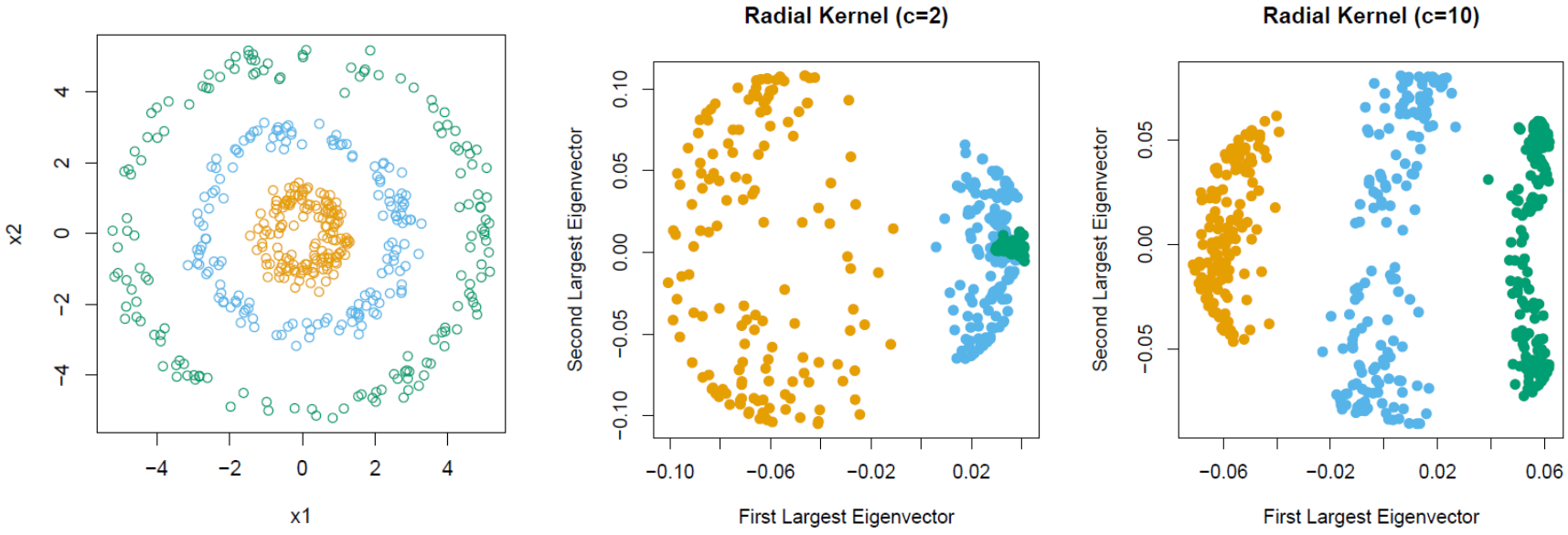
- Kernel PCA may map data x_i to latent coordinates z_i where *clustering* is much easier

- All of the following can be represented as kernel PCA:
  – Local Linear Embedding
  – Metric Multidimensional Scaling
  – Laplacian Eigenmaps (Spectral Clustering)

  see "Dimensionality Reduction: A Short Tutorial" by Ali Ghodsi

Kernel PCA clustering

- Using a kernel function $k(x, x') = e^{-\|x-x'\|^2/c}$:



- Gaussian mixtures or k-means will easily cluster this

Spectral Clustering*

Spectral Clustering is very similar to kernel PCA:

- Instead of the kernel matrix K with entries $k_{ij} = k(x_i, x_j)$ we construct a weighted *adjacency matrix*, e.g.,

$$w_{ij} = \begin{cases} 0 & \text{if } x_i \text{ are not a } k\text{NN of } x_j \\ e^{-\|x_i - x_j\|^2/c} & \text{otherwise} \end{cases}$$

  w_{ij} is the weight of the *edge* between data point x_i and x_j.

- Instead of computing *maximal* eigenvectors of $\widetilde{K}$, compute *minimal* eigenvectors of

$$L = \mathbf{I} - \widetilde{W}\ , \quad \widetilde{W} = \mathrm{diag}(\textstyle\sum_j w_{ij})^{-1} W$$

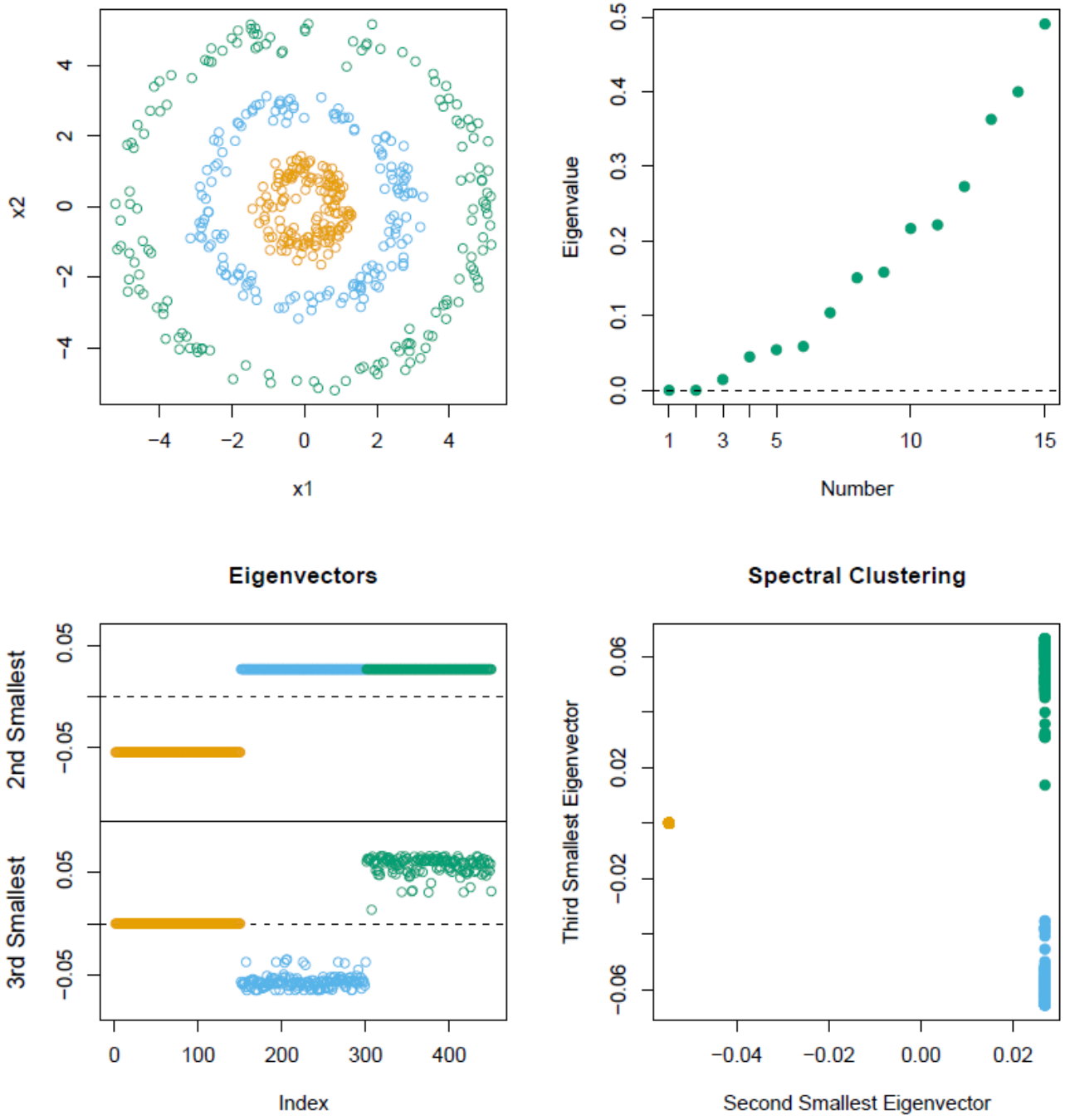
  ($\sum_j w_{ij}$ is called *degree of node* i, $\widetilde{W}$ is the normalized weighted adjacency matrix)

- Given $L = UDV^\top$, we pick the p smallest eigenvectors $V_p = V_{1:n,1:p}$ (perhaps exclude the trivial smallest eigenvector)

- The latent coordinates for x_i are $z_i = V_{i,1:p}$

- Spectral Clustering provides a method to compute latent low-dimensional coordinates $z_i = V_{i,1:p}$ for each high-dimensional $x_i \in \mathbb{R}^d$ input.

- This is then followed by a standard clustering, e.g., Gaussian Mixture or k-means


x2
x1
Eigenvalue
Number
Eigenvectors
2nd Smallest
3rd Smallest
Index
Spectral Clustering
Third Smallest Eigenvector
Second Smallest Eigenvector

- Spectral Clustering is similar to kernel PCA:
  - The kernel matrix K usually represents similarity
    The weighted adjacency matrix W represents proximity & similarity
  - High Eigenvectors of K are similar to low EV of $L = \mathbf{I} - W$

- Original interpretation of Spectral Clustering:
  - $L = \mathbf{I} - W$ (weighted graph Laplacian) describes a diffusion process:
    The diffusion rate W_{ij} is high if i and j are close and similar
  - Eigenvectors of L correspond to stationary solutions

- The Graph Laplacian L: For some vector $f \in \mathbb{R}^n$, note the following identities:

$$
\begin{aligned}
(Lf)_i &= (\sum_j w_{ij}) f_i - \sum_j w_{ij} f_j = \sum_j w_{ij}(f_i - f_j) \\
f^\top L f &= \sum_i f_i \sum_j w_{ij}(f_i - f_j) = \sum_{ij} w_{ij}(f_i^2 - f_i f_j) \\
&= \sum_{ij} w_{ij}(\frac{1}{2} f_i^2 + \frac{1}{2} f_j^2 - f_i f_j) = \frac{1}{2} \sum_{ij} w_{ij}(f_i - f_j)^2
\end{aligned}
$$

where the second-to-last = holds if $w_{ij} = w_{ji}$ is symmetric.

Metric Multidimensional Scaling

- Assume we have data $D = \{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$.
  As before we want to indentify latent lower-dimensional representations $z_i \in \mathbb{R}^p$ for this data.

- A simple idea: Minimize the stress

  $$S_C(z_{1:n}) = \sum_{i \neq j} (d_{ij}^2 - \|z_i - z_j\|^2)^2$$

  We want distances in high-dimensional space to be equal to distances in low-dimensional space.

Metric Multidimensional Scaling = (kernel) PCA

- Note the relation:

$$d_{ij}^2 = \|x_i - x_j\|^2 = \|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - 2(x_i - \bar{x})^\top (x_j - \bar{x})$$

This translates a distance into a (centered) scalar product

Metric Multidimensional Scaling = (kernel) PCA

- Note the relation:

$$d_{ij}^2 = \|x_i - x_j\|^2 = \|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - 2(x_i - \bar{x})^\top (x_j - \bar{x})$$

This translates a distance into a (centered) scalar product

- If may we define

$$\widetilde{K} = (\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^\top) D (\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^\top) \ , \quad D_{ij} = -d_{ij}^2/2$$

then $\widetilde{K_{ij}} = (x_i - \bar{x})^\top (x_j - \bar{x})$ is the normal covariance matrix and MDS is equivalent to kernel PCA

Non-metric Multidimensional Scaling

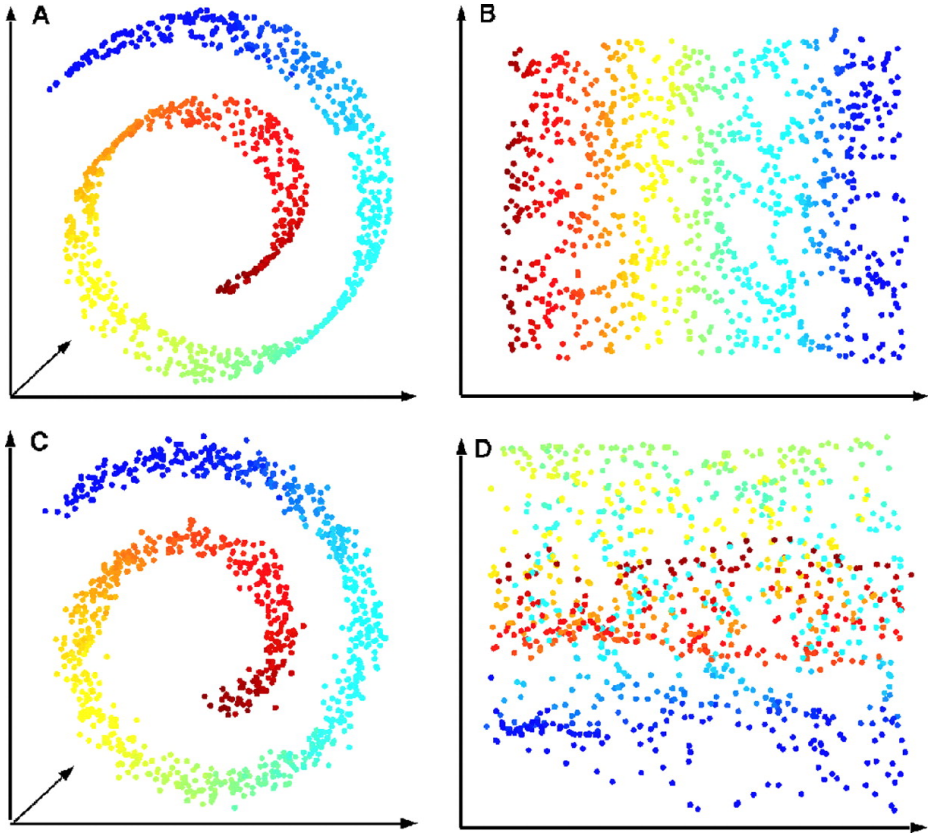
- We can do this for any data (also non-vectorial or not $\in \mathbb{R}^d$) as long as we have a data set of comparative dissimilarities d_{ij}

$$S(z_{1:n}) = \sum_{i \neq j} (d_{ij}^2 - |z_i - z_j|^2)^2$$

- Minimize $S(z_{1:n})$ w.r.t. $z_{1:n}$ *without any further constraints*!

Example for Non-Metric MDS: ISOMAP

- Construct kNN graph and label edges with Euclidean distance
  - Between any two x_i and x_j, compute "geodescic" distance d_{ij} (shortest path along the graph)
  - Then apply MDS



by Tenenbaum et al.

The zoo of dimensionality reduction methods

- PCA family:
  – kernel PCA, non-neg. Matrix Factorization, Factor Analysis

- All of the following can be represented as kernel PCA:
  – Local Linear Embedding
  – Metric Multidimensional Scaling
  – Laplacian Eigenmaps (Spectral Clustering)

They all use different notions of distance/correlation as input to kernel PCA

see “Dimensionality Reduction: A Short Tutorial” by Ali Ghodsi

PCA variants*

PCA variant: Non-negative Matrix Factorization*

- Assume we have data $D = \{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$.
  As for PCA (where we had $x_i \approx V_p z_i + \mu$) we search for a lower-dimensional space with linear relation to x_i

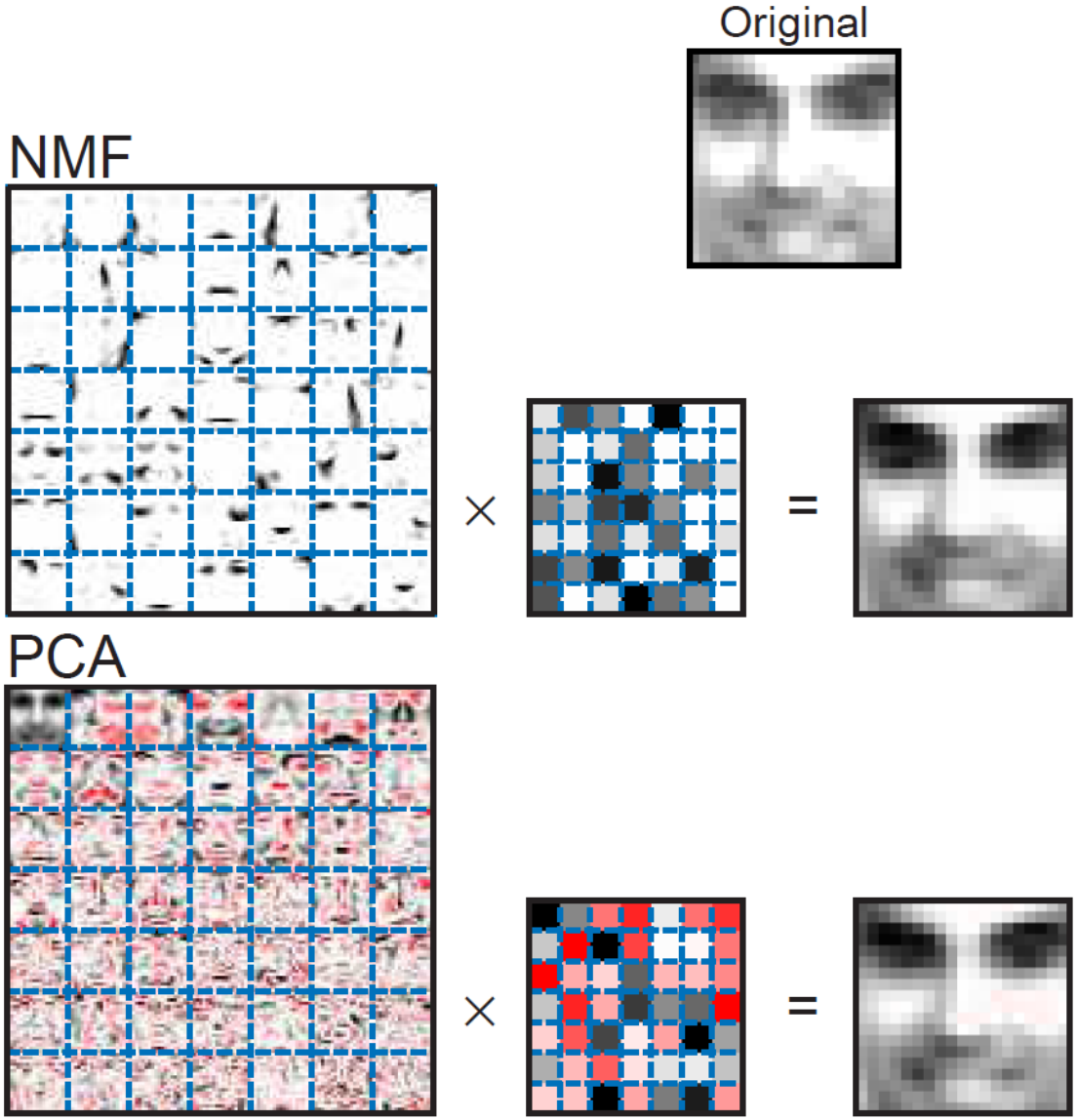
- In NMF we require everything is **non-negative**: the data x_i, the projection W, and latent variables z_i
  Find $W \in \mathbb{R}^{p \times d}$ (the tansposed projection) and $Z \in \mathbb{R}^{n \times p}$ (the latent variables z_i) such that
  $$X \approx ZW$$

- Iterative solution: (E-step and M-step like...)

$$z_{ik} \leftarrow z_{ik} \frac{\sum_{j=1}^d w_{kj} x_{ij} / (ZW)_{ij}}{\sum_{j=1}^d w_{kj}}$$

$$w_{kj} \leftarrow w_{kj} \frac{\sum_{i=1}^N z_{ik} x_{ij} / (ZW)_{ij}}{\sum_{i=1}^N z_{ik}}$$

PCA variant: Non-negative Matrix Factorization*



(from Hastie 14.6)

PCA variant: Factor Analysis*

Another variant of PCA: (Bishop 12.64)

Allows for different noise in each dimension

$P(x_i \mid z_i) = \mathcal{N}(x_i \mid V_p z_i + \mu, \Sigma)$ (with Σ diagonal)

Clustering

- Clustering often involves two steps:
- First map the data to some embedding that emphasizes clusters
  - (Feature) PCA
  - Spectral Clustering
  - Kernel PCA
  - ISOMAP
- Then explicitly analyze clusters
  - k-means clustering
  - Gaussian Mixture Model
  - Agglomerative Clustering

k-means Clustering

- Given data $D = \{x_i\}_{i=1}^n$, find K centers μ_k, and a data assignment $c : i \mapsto k$ to minimize

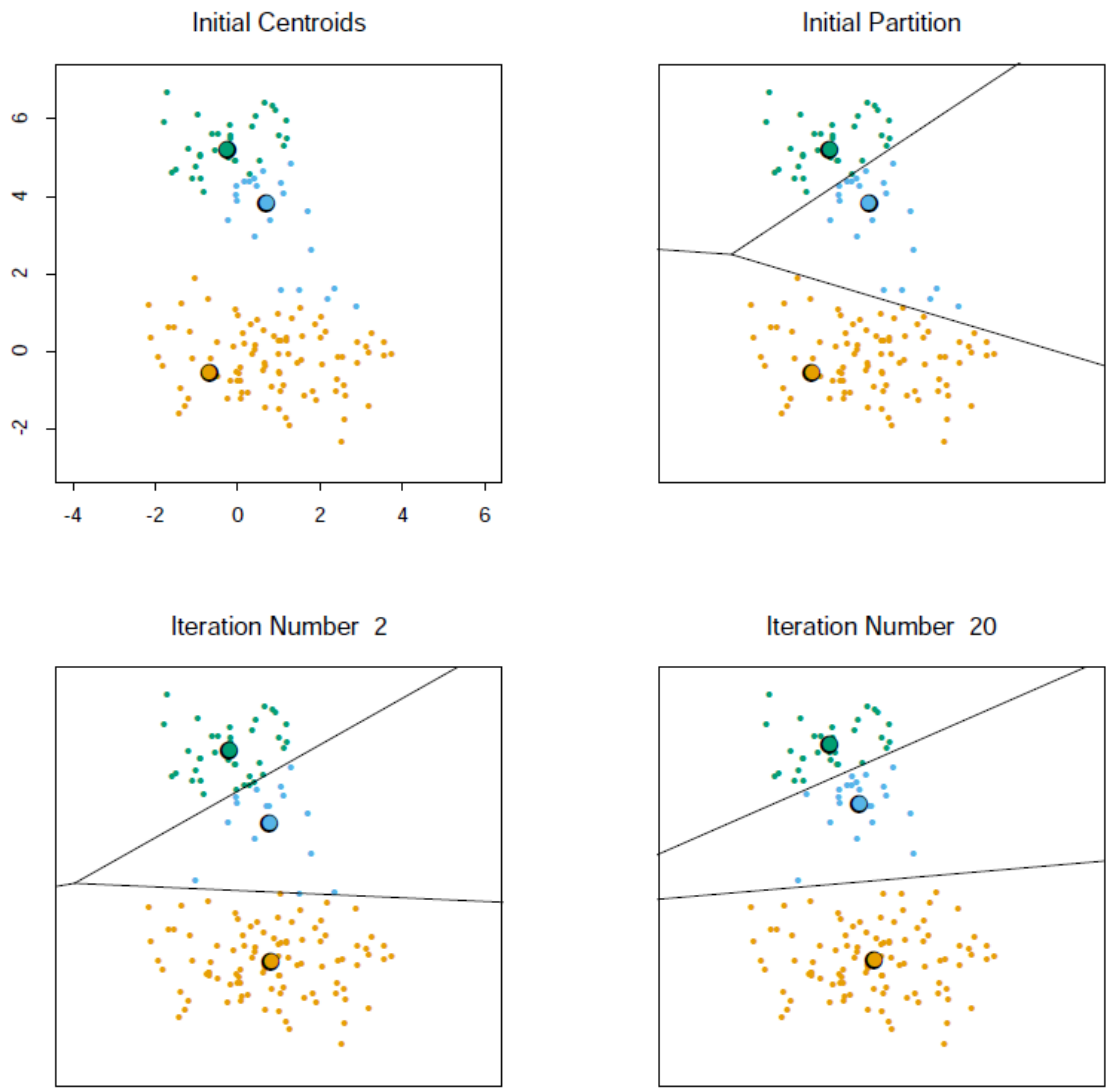
$$\min_{c,\mu} \sum_i (x_i - \mu_{c(i)})^2$$

- k-means clustering:
  - Pick K data points randomly to initialize the centers μ_k
  - Iterate adapting the assignments $c(i)$ and the centers μ_k:

$$\forall_i : \; c(i) \leftarrow \operatorname*{argmin}_{c(i)} \sum_j (x_j - \mu_{c(j)})^2 = \operatorname*{argmin}_k (x_i - \mu_k)^2$$

$$\forall_k : \; \mu_k \leftarrow \operatorname*{argmin}_{\mu_k} \sum_i (x_i - \mu_{c(i)})^2 = \frac{1}{|c^{-1}(k)|} \sum_{i \in c^{-1}(k)} x_i$$

k-means Clustering

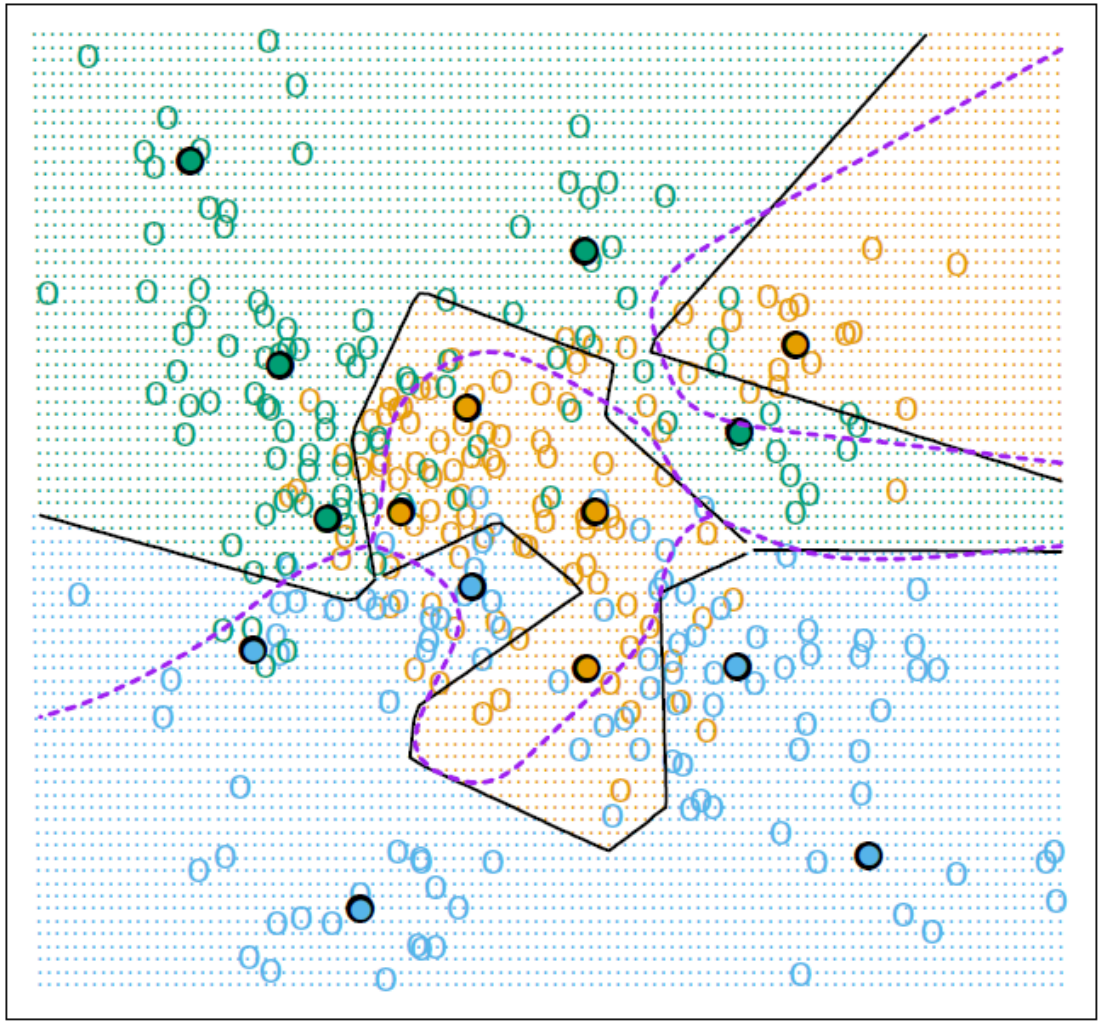


from Hastie

k-means Clustering

- Converges to local minimum $\rightarrow$ **many restarts**
- Choosing k? Plot $L(k) = \min_{c,\mu} \sum_i (x_i - \mu_{c(i)})^2$ for different k – choose a tradeoff between model complexity (large k) and data fit (small loss $L(k)$)

k-means Clustering for Classification



from Hastie

Gaussian Mixture Model for Clustering

- GMMs can/should be introduced as *generative* probabilistic model of the data:
  - K different Gaussians with parmameters μ_k, Σ_k
  - Assignment RANDOM VARIABLE $c_i \in \{1, .., K\}$ with $P(c_i = k) = \pi_k$
  - The observed data point x_i with $P(x_i \,|\, c_i = k; \mu_k, \Sigma_k) = \mathcal{N}(x_i \,|\, \mu_k, \Sigma_k)$
- EM-Algorithm described as a kind of soft-assignment version of k-means
  - Initialize the centers $\mu_{1:K}$ randomly from the data; all covariances $\Sigma_{1:K}$ to unit and all π_k uniformly.
  - **E-step:** (probabilistic/soft assignment) Compute

$$q(c_i = k) = P(c_i = k \,|\, x_i, \mu_{1:K}, \Sigma_{1:K}) \propto \mathcal{N}(x_i \,|\, \mu_k, \Sigma_k)\ \pi_k$$

  - **M-step:** Update parameters (centers AND covariances)

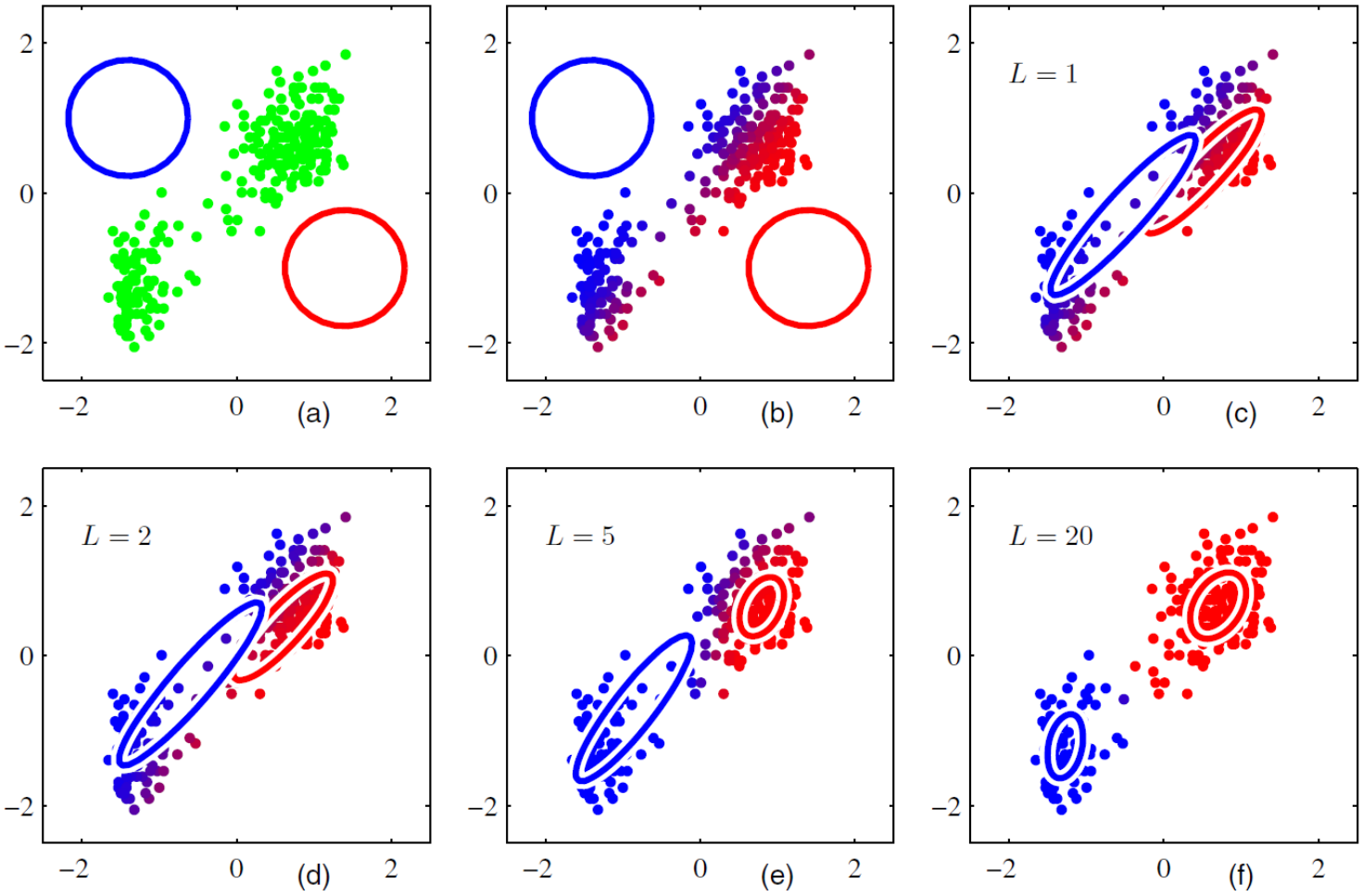
$$\pi_k = \frac{1}{n} \textstyle\sum_i q(c_i = k)$$

$$\mu_k = \frac{1}{n\pi_k}\ \textstyle\sum_i q(c_i = k)\ x_i$$

$$\Sigma_k = \frac{1}{n\pi_k}\ \textstyle\sum_i q(c_i = k)\ x_i x_i^\top - \mu_k \mu_k^\top$$

Gaussian Mixture Model

EM iterations for Gaussian Mixture model:

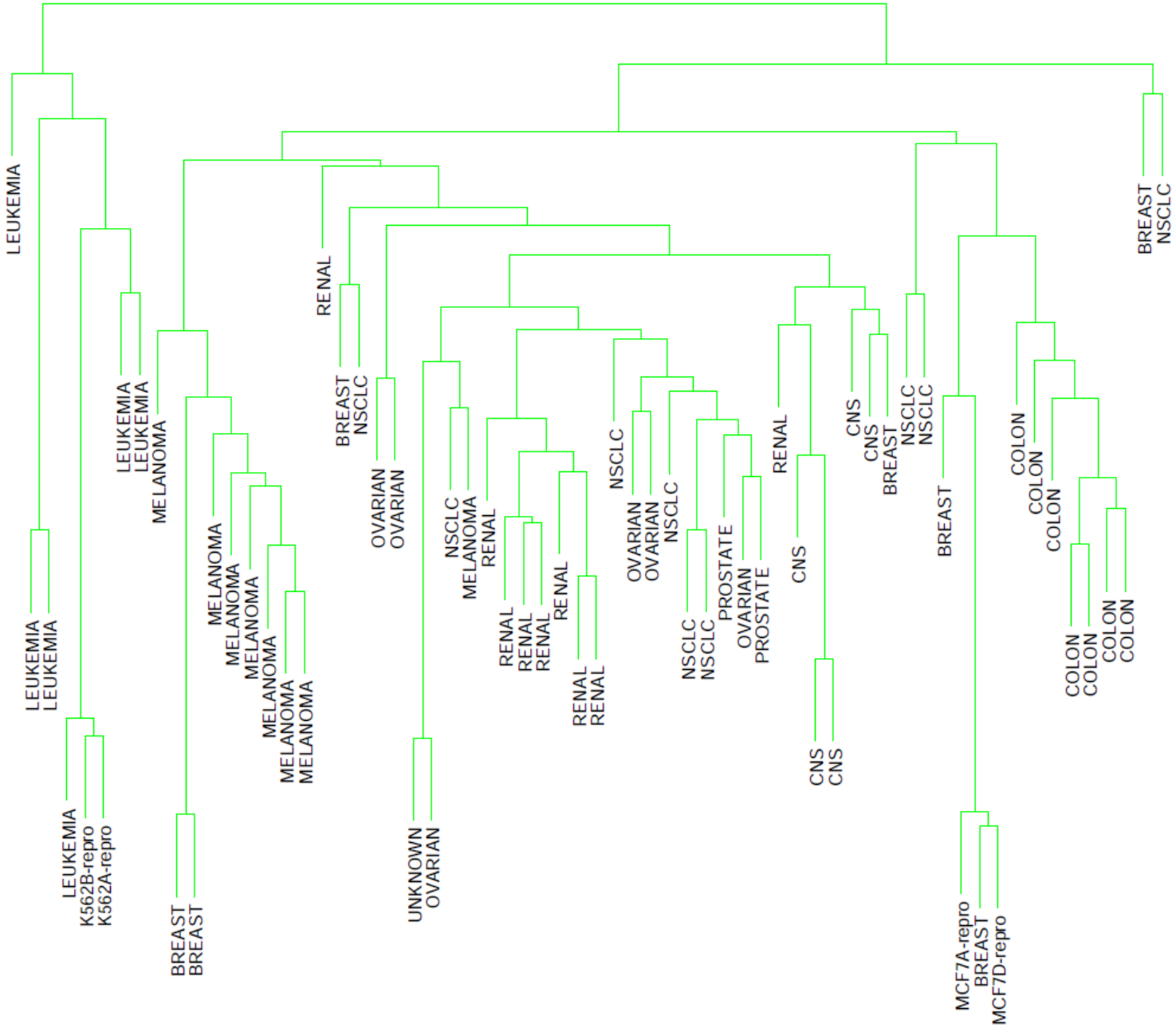


from Bishop

Agglomerative Hierarchical Clustering

- *agglomerative* = bottom-up, *divisive* = top-down
- Merge the two groups with the smallest intergroup dissimilarity
- Dissimilarity of two groups G, H can be measures as
  - Nearest Neighbor (or "single linkage"): $d(G, H) = \min_{i \in G, j \in H} d(x_i, x_j)$
  - Furthest Neighbor (or "complete linkage"): $d(G, H) = \max_{i \in G, j \in H} d(x_i, x_j)$
  - Group Average: $d(G, H) = \frac{1}{|G||H|} \sum_{i \in G} \sum_{j \in H} d(x_i, x_j)$

Agglomerative Hierarchical Clustering



Appendix: Centering & Whitening

- Some prefer to *center* (shift to zero mean) the data before applying methods:

$$x \leftarrow x - \langle x \rangle \ , \quad y \leftarrow y - \langle y \rangle$$

  this spares augmenting the bias feature 1 to the data.

- More interesting: The loss and the best choice of λ depends on the *scaling* of the data. If we always scale the data in the same range, we may have better priors about choice of λ and interpretation of the loss

$$x \leftarrow \frac{1}{\sqrt{\text{Var}\{x\}}}\, x \ , \quad y \leftarrow \frac{1}{\sqrt{\text{Var}\{y\}}}\, y$$

- **Whitening:** Transform the data to remove all correlations and variances.

  Let $A = \text{Var}\{x\} = \frac{1}{n} X^\top X - \mu\mu^\top$ with Cholesky decomposition $A = MM^\top$.

$$x \leftarrow M^{-1} x \ , \quad \text{with Var}\{M^{-1} x\} = \mathbf{I}_d$$