Machine Intelligence II

Prof. Dr. Klaus Obermayer

March 13, 2019

Disclaimer: The lecture has been restructured in the previous terms (SoSe 2017 as well as SoSe 2018) and several new topics were added (and are being added), too. Many of these changes are so far only reflected in the lecture slides but not yet in these lecture notes which are thus outdated. Although we plan to complete the lecture notes soon, it is unlikely that we reach this goal until the end of the term. Therefore, it is strongly recommended to make hand-written notes within the lectures to complement the information of this script. We apologize for the inconvenience.

Contents

0 Introductory comments

Methods subsumed under the term *unsupervised learning* deal with finding structure or regularities in a set of observations $\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \ldots, \underline{\mathbf{x}}^{(p)}$.

What is the statistical structure of the data?

Many datasets ...

- ... are high-dimensional $\rightarrow$ visualization & dimension reduction
- ... are grouped or clustered $\rightarrow$ definition of groups / categories, construction of taxonomies, preprocessing for prediction (clustering)
- ... may display interesting (or uninteresting) directions $\rightarrow$ definition of "informative" features (projection methods)
- ... may be determined by different causes $\rightarrow$ unmixing a mixture of sources, definition of components, infering causes

Motivation: In contrast to methods of *supervised learning* (see MI1), there is no additional information in terms of "labels" $\underline{\mathbf{y}}^{(\alpha)}$ providing a "correct" classification or target value for data point α.

The statistical structure of a data set is often interesting in itself and can be exploited for knowledge extraction (modeling). Furthermore, it allows to find new representations of the data $\underline{\mathbf{x}}$ that allow more efficient data storage ($\rightsquigarrow$ dimensionality reduction, data compression) or are better suited to solve supervised problems such as prediction, classification, reasoning, decision making.

Unsupervised learning & statistical data analysis: methods for the extraction of the statistical "structure" underlying a set of observations (or a data structure)

- $\Rightarrow$ projection methods: search for "interesting" directions in feature space
- $\Rightarrow$ clustering methods: grouping & categorization (and prototypes)

Note: Both PCA and ICA (discussed in chapters 1.1 and 2.1) are linear projection methods. In many cases (e.g. PCA) kernel methods allow to extend them to nonlinear problems.

1 Projection Methods

Projection methods provide important tools for high-dimensional datasets

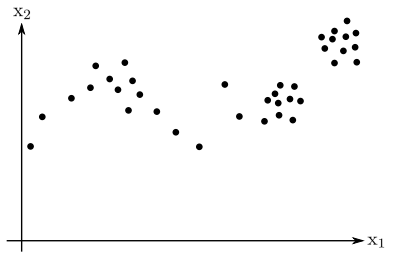


Figure 1: Projection and clustering methods for multidimensional data

1.1 Principal Component Analysis

1.1.1 The Covariance Matrix

observations: $\{\underline{\mathbf{x}}^{(\alpha)}\},\ \alpha = 1, \ldots, p; \quad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

Feature space:

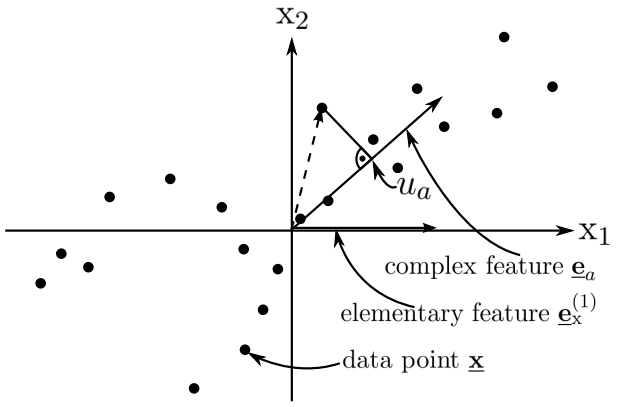


Figure 2: Data, elementary and complex features

elementary features $\underline{\mathbf{e}}_{\mathrm{x}}^{(1)}, \underline{\mathbf{e}}_{\mathrm{x}}^{(2)}, \underline{\mathbf{e}}_{\mathrm{x}}^{(3)}, \ldots \underline{\mathbf{e}}_{\mathrm{x}}^{(N)}$**:** basis vectors of unit length which correspond to the individual measurements

complex feature $\underline{\mathbf{e}}_a$**:** vector of unit ($\|\underline{\mathbf{e}}_a\| = 1$) length which corresponds to a particular direction in feature space

feature value u_a**:** projection of observation $\underline{\mathbf{x}}$ onto the complex feature $\underline{\mathbf{e}}_a$

$$\underbrace{u_a}_{\text{value}} = \underbrace{\underline{\mathbf{e}}_a^T}_{\text{feature}} \cdot \underbrace{\underline{\mathbf{x}}}_{\text{observation}} \tag{1.1}$$

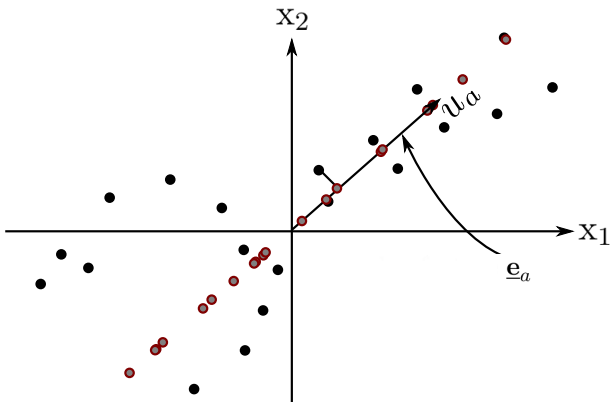


Figure 3: Same data as in Fig. 2, projected onto complex feature $\underline{\mathbf{e}}_a$ with feature values u_a

Example – Leptograpsus data: Dead crabs lose their color and their sexual features – Can we infer species / sex from the shells alone?

$$\text{crabs: L. variegatis} \left\{ \begin{array}{l} \text{orange} \\ \\ \text{blue} \end{array} \right\} \begin{array}{l} \text{two (sub-)species} \\ \\ \rightarrow \text{ male and female crabs} \end{array}$$

Ex: More examples on slide Leptograpsus data, Iris data

- *elementary features:*

$$\left. \begin{array}{ll} \text{width of the frontal lip:} & \mathrm{x}_1 \\ \text{width of the back:} & \mathrm{x}_2 \\ \text{length along midline:} & \mathrm{x}_3 \\ \text{max. width of top shell:} & \mathrm{x}_4 \\ \text{body depth:} & \mathrm{x}_5 \end{array} \right\} \text{5-dim. feature vector}$$

- *complex features:* some direction in feature space (i.e. linear combination of elementary features) which is indicative of color and/or sex.

Moments of the set of observations: Moments of a distribution provide important information about the location and shape of a distribution.

First moment (sample mean/center of mass):

$$\underline{\mathbf{m}} = \frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} \tag{1.2}$$

Second moments (Variances and Covariances):

$$C_{ij} = \frac{1}{p} \sum_{\alpha=1}^{p} \underbrace{\left(\mathrm{x}_i^{(\alpha)} - m_i\right)}_{\substack{\text{deviations from} \\ \text{the mean}}} \underbrace{\left(\mathrm{x}_j^{(\alpha)} - m_j\right)}_{\substack{\text{component} \\ \text{indices } i,j}} \tag{1.3}$$

In vector notation:

$$\underline{\mathbf{C}} = \frac{1}{p} \sum_{\alpha=1}^{p} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{m}}\right) \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{m}}\right)^T \qquad \text{(covariance matrix)}$$

Note: "Centering" the data[1] yields $\underline{\mathbf{m}} = \underline{\mathbf{0}}$ and we obtain

$$C_{ij} = \frac{1}{p} \sum_{\alpha=1}^{p} \mathrm{x}_i^{(\alpha)} \mathrm{x}_j^{(\alpha)} \tag{1.4}$$

and thus

$$\underline{\mathbf{C}} = \frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \tag{1.5}$$

Properties of the covariance matrix

$C_{ij} = C_{ji}$ the covariance matrix is symmetric

$i = j$ $C_{ii} = \frac{1}{p} \sum_{\alpha=1}^{p} \left(\mathrm{x}_i^{(\alpha)} - m_i\right)^2$
$\rightsquigarrow$ variance of the data along the elementary features $\underline{\mathbf{e}}_{\mathrm{x}}^{(i)}$(variance of variable x_i)

$i \neq j$ C_{ij} : covariances
$\rightsquigarrow$ measure of correlations between variables
$\rightsquigarrow C_{ij} = 0 \rightarrow$ variables are uncorrelated

First moment of the data projected onto an arbitrary complex feature $\underline{\mathbf{e}}_a$:

$$\begin{aligned} m_a &= \tfrac{1}{p} \sum_{\alpha=1}^{p} u_a^{(\alpha)} \\ &= \tfrac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{e}}_a^T \cdot \underline{\mathbf{x}}^{(\alpha)} \\ &= \underline{\mathbf{e}}_a^T \cdot \underline{\mathbf{m}} \end{aligned} \qquad \text{(mean)}$$

[1] centering: subtract mean from all data points

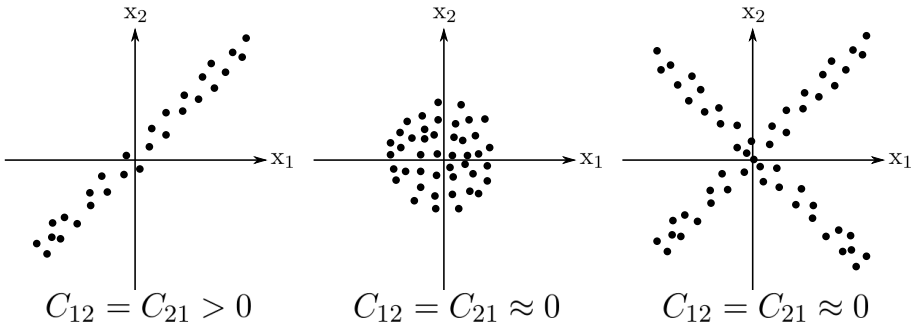


Figure 4: Illustration of data patterns: Note that $C_{ij} = 0$ does not imply that there are no dependencies between the variables. Clearly, $p(\mathrm{x}_1, \mathrm{x}_2) \neq p(\mathrm{x}_1)p(\mathrm{x}_2)$ in the third example.

Second moment along this direction:

$$\begin{aligned}
\sigma_a^2 &= \tfrac{1}{p} \sum_{\alpha=1}^{p} \Big(u_a^{(\alpha)} - m_a \Big)^2 \\
&= \tfrac{1}{p} \sum_{\alpha=1}^{p} \Big(\underline{\mathbf{e}}_a^T \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{e}}_a^T \underline{\mathbf{m}} \Big)^2 \\
&= \tfrac{1}{p} \sum_{\alpha=1}^{p} \left\{ \Big(\underline{\mathbf{e}}_a^T \underline{\mathbf{x}}^{(\alpha)} \Big)^2 - 2 \Big(\underline{\mathbf{e}}_a^T \underline{\mathbf{x}}^{(\alpha)} \Big) \underline{\mathbf{e}}_a^T \underline{\mathbf{m}} + \Big(\underline{\mathbf{e}}_a^T \underline{\mathbf{m}} \Big)^2 \right\}
\end{aligned}$$

with $\underline{\mathbf{e}}_a^T \underline{\mathbf{x}}^{(\alpha)} = \sum_{i=1}^{N} \big(\underline{\mathbf{e}}_a \big)_i \underline{\mathbf{x}}_i^{(\alpha)}$ we obtain

$$= \sum_{i,j=1}^{N} \big(\underline{\mathbf{e}}_a \big)_i \tfrac{1}{p} \sum_{\alpha=1}^{p} \left\{ \mathrm{x}_i^{(\alpha)} \mathrm{x}_j^{(\alpha)} - \mathrm{x}_i^{(\alpha)} m_j - \underbrace{\mathrm{x}_i^{(\alpha)} m_j}_{\substack{= m_i x_j^{(\alpha)} \\ \text{change of} \\ \text{indices}}} + m_i m_j \right\} \big(\underline{\mathbf{e}}_a \big)_j$$

$$= \sum_{i,j=1}^{N} \big(\underline{\mathbf{e}}_a \big)_i \underbrace{\left\{ \frac{1}{p} \sum_{\alpha=1}^{p} \Big(\mathrm{x}_i^{(\alpha)} - m_i \Big) \Big(\mathrm{x}_j^{(\alpha)} - m_j \Big) \right\}}_{= C_{ij}} \big(\underline{\mathbf{e}}_a \big)_j$$

(variance)

$$\boxed{\sigma_a^2 = \underline{\mathbf{e}}_a^T \mathbf{C} \underline{\mathbf{e}}_a} \tag{1.6}$$

Observation: The covariance matrix determines the variance of the data along every possible direction.

1.1.2 The Principle of Maximal Variance

"interesting" complex features:

⇒ properties of the data which vary most strongly across the data set and might therefore allow to characterize different data points

⇒ direction in feature space along which variance is maximal

$$\boxed{\sigma_a^2 = \max\left(\underline{\mathbf{e}}_a\right)} \qquad \text{optimisation under constraints}$$

$$\boxed{\underline{\mathbf{e}}_a^2 = 1}$$

Solution is found using the method of Lagrange multipliers:

$$\underbrace{\underline{\mathbf{e}}_a^T \underline{\mathbf{C}} \underline{\mathbf{e}}_a}_{\text{objective}} - \underbrace{\lambda}_{\substack{\text{Lagrange} \\ \text{multiplier}}} \underbrace{\left(\underline{\mathbf{e}}_a^2 - 1\right)}_{\text{constraints}} \stackrel{!}{=} \max \tag{1.7}$$

⇝ family of solutions parametrized by λ

⇝ λ must be chosen such that constraints are fulfilled

$$f(\underline{\mathbf{e}}_a) := \sum_{i,j=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i C_{ij} \left(\underline{\mathbf{e}}_a\right)_j - \lambda \left\{ \sum_{i=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i^2 - 1 \right\} \stackrel{!}{=} \max \tag{1.8}$$

$$\frac{\partial f}{\partial \left(\underline{\mathbf{e}}_a\right)_k} \stackrel{!}{=} 0 \text{ for all components } k \tag{1.9}$$

$$\sum_{j=1}^{N} C_{kj} \left(\underline{\mathbf{e}}_a\right)_j + \sum_{i=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i C_{ik} - 2\lambda \left(\underline{\mathbf{e}}_a\right)_k \stackrel{!}{=} 0 \tag{1.10}$$

$$2 \sum_{j=1}^{N} C_{kj} \left(\underline{\mathbf{e}}_a\right)_j - 2\lambda \left(\underline{\mathbf{e}}_a\right)_k \stackrel{!}{=} 0 \text{ because } \underline{\mathbf{C}} \text{ is symmetric} \tag{1.11}$$

$$\boxed{\underline{\mathbf{C}} \underline{\mathbf{e}}_a = \lambda \underline{\mathbf{e}}_a} \qquad \text{(eigenvalue problem)}$$

⇝ $\underline{\mathbf{e}}_a$ is an eigenvector of $\underline{\mathbf{C}}$

⇝ $\underline{\mathbf{e}}_a^2 = 1$ can always be fulfilled (through normalization)

⇝ variance $\sigma_a^2 = \underline{\mathbf{e}}_a^T \underline{\mathbf{C}} \underline{\mathbf{e}}_a = \lambda \underline{\mathbf{e}}_a^2 = \lambda_a$
eigenvalues: variances of the data along the directions given by the eigenvectors

The eigenvectors of $\underline{\mathbf{C}}$ with the largest (smallest) eigenvalue points in the direction of the largest (smallest) variance of the data

1.1.3 Principal Components

Principal component: (normalized) eigenvector $\underline{\mathbf{e}}$ of a covariance matrix $\underline{\mathbf{C}}$

Covariance matrix $\underline{\mathbf{C}}$:

- real valued
- symmetric
- positive semidefinite, all eigenvalues must be positive or zero (they are variances!)

Properties of principal components

(1) covariance matrix $\underline{\mathbf{C}}$ is real and symmetric (all eigenvalues have to be nonnegative – λs are variances!)

→ eigenvectors form an orthonormal basis:

$$\underline{\mathbf{e}}_i^T \cdot \underline{\mathbf{e}}_j = \delta_{ij} \leftarrow \text{ Kronecker-Delta } \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & \text{else} \end{cases} \tag{1.12}$$

→ $\underline{\mathbf{C}}$ is $N \times N$ matrix $\rightsquigarrow$ N eigenvectors

(2) $\underline{\mathbf{C}}$ is diagonal w.r.t. its eigenbasis

let $\underline{\mathbf{M}} = \left(\underline{\mathbf{e}}_1 \underline{\mathbf{e}}_2, \ldots, \underline{\mathbf{e}}_N\right)$ then

$$\underline{\mathbf{M}}^T \underline{\mathbf{C}} \underline{\mathbf{M}} = \widehat{\underline{\mathbf{C}}} = \begin{pmatrix} \lambda_1 & & & & \\ & \lambda_2 & & 0 & \\ & & \ddots & & \\ & 0 & & \ddots & \\ & & & & \lambda_N \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \text{transformation into the eigenbasis} \end{matrix} \tag{1.13}$$

→ principal components are uncorrelated

→ transformation into the eigenbasis leads to uncorrelated "complex" features

(3) interpretation of principal components w.r.t. variance $\square^{\text{interpretation of PCs}}$

$$\begin{array}{ccccccccccc} \lambda_1 & > & \lambda_2 & > & \lambda_3 & > & \ldots\ldots & > & \lambda_{N-1} & > & \lambda_N \\ \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\ \underline{\mathbf{e}}_1 & & \underline{\mathbf{e}}_2 & & \underline{\mathbf{e}}_3 & & & & \underline{\mathbf{e}}_{N-1} & & \underline{\mathbf{e}}_N \end{array} \tag{1.14}$$

direction of largest variance $\longrightarrow ? \longrightarrow$ direction of smallest variance

$\underline{\mathbf{e}}_j$ points to the direction of largest variance within the subspace of $\mathbb{R}^N$ spanned by all $\underline{\mathbf{e}}_i$ with $i > j$.

(4) optimal dimensionality reduction: consider the transformation of data points into eigenvectors of $\underline{\mathbf{C}}$

$$\underline{\mathbf{x}} = \underbrace{a_1}_{\underline{\mathbf{e}}_1^T\underline{\mathbf{x}}}\underline{\mathbf{e}}_1 + \underbrace{a_2}_{\underline{\mathbf{e}}_2^T\underline{\mathbf{x}}}\underline{\mathbf{e}}_2 + \ldots + \underbrace{a_N}_{\underline{\mathbf{e}}_N^T\underline{\mathbf{x}}}\underline{\mathbf{e}}_N \tag{1.15}$$

The projection into the subspace by the M principal components with the largest eigenvalues ($M < N$)

$$\widetilde{\underline{\mathbf{x}}} = a_1\underline{\mathbf{e}}_1 + a_2\underline{\mathbf{e}}_2 + \ldots + a_M\underline{\mathbf{e}}_M \tag{1.16}$$

then yields an approximation error:

$$(\underline{\mathbf{x}} - \widetilde{\underline{\mathbf{x}}})^2 = \sum_{j=M+1}^{N} a_j^2 \tag{1.17}$$

This reconstruction error is minimal w.r.t. all possible projections into M-dimensional subspaces

1.1.4 Latent factors

- the data may appear high dimensional, but there may only be a small number of features underlying variability
- dimensionality reduction: projection of the data into a low dimensional subspace which captures the "essence" of the data
- latent factors: remaining PCs with high variance

☐ Ex: Eigenfaces

☐ Ex: Spiking activity in monkey visual cortex

1.1.5 Summary of Principal Component Analysis (PCA)

given observations: $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

(1) normalization to zero mean

$$\underline{\mathbf{m}} = \frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)} \rightsquigarrow \widehat{\underline{\mathbf{x}}}^{(\alpha)} \leftarrow \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{m}} \tag{1.18}$$

(2) calculation of the covariance matrix $\underline{\mathbf{C}}$

$$C_{ij} = \frac{1}{p}\sum_{\alpha=1}^{p}\widehat{\mathrm{x}}_i^{(\alpha)}\widehat{\mathrm{x}}_j^{(\alpha)} \tag{1.19}$$

(3) solve the eigenvalue problem

$$\underline{\mathbf{C}}\underline{\mathbf{e}} = \lambda \underline{\mathbf{e}} \tag{1.20}$$

Note: The eigenvectors of $\underline{\mathbf{C}}$ are called *Principal Components*

Numerical method: singular value decomposition (see **PressEtAl2007**, chapt. 2.9, Jacobi transformations: chapt. 11.1, reduction and QL: chapt. 11.2-11.3) or simply use a linear algebra package.

projections for Leptograpsus data

Example – PCA of the Leptograpsus data: 5 dimensions → 5 principal components (PCs)

PC1: "size"
PC2: "sex"
PC3: "color/subspecies"

} relative importance of elementary features can be "read out" from the feature vectors

⇒ example for successful visualization

⇒ example for a successful preprocessing for classification

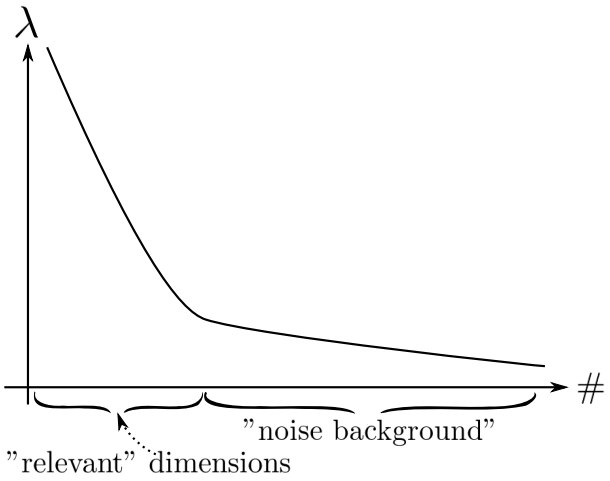


Figure 5: scree plots (ordering of eigenvalues by size):

Comments:

- variance is a scale sensitive measure (e.g. using centimeters instead of millimeters along one dimension will change all PCs)
- max. variance criterion only makes sense if scales are "comparable"

- still: PCA constructs uncorrelated features (independent of scale)
- PCA is often used for "whitening": scale variance along all directions to one

1.2 Hebbian Learning for Linear Neurons

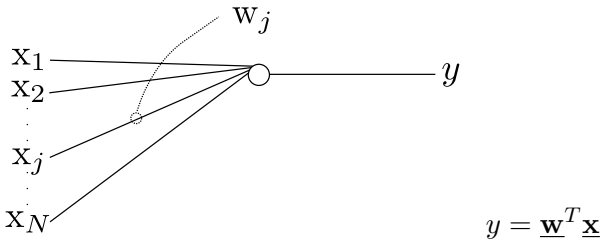


Figure 6: Linear connectionist neuron

observations: $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p, \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

Algorithm 1: Hebbian (correlation-based) learning for linear neurons

initialization of weights (e.g. to small numbers)
choose learning rate ε
begin loop
  Choose an observation $\underline{\mathbf{x}}^{(\alpha)}$
  Change weights according to:

$$\underbrace{\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \underline{\mathbf{x}}^{(\alpha)}}_{\substack{\text{weights increase (decrease) if input} \\ \text{and output are correlated (anticorrelated)}}} \tag{1.21}$$

end

Proposition: Applied to linear neurons, Hebbs rule extracts the PC with the largest Eigenvalue.

Proof: small learning steps $\rightsquigarrow$ average over all patterns

$$\begin{aligned} \Delta \mathrm{w}_j &\approx \tfrac{\varepsilon}{p} \sum_{\alpha=1}^{p} y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{x}_j^{(\alpha)} \\ &= \tfrac{\varepsilon}{p} \sum_{\alpha=1}^{p} \sum_{k=1}^{N} \mathrm{w}_k \mathrm{x}_k^{(\alpha)} \mathrm{x}_j^{(\alpha)} \\ &= \varepsilon \sum_{k=1}^{N} \mathrm{w}_k C_{kj} \end{aligned} \tag{1.22}$$

$$\Delta\underline{\mathbf{w}} = \varepsilon\underline{\mathbf{C}}\underline{\mathbf{w}} \rightsquigarrow \begin{matrix}\text{"analysis" of the} \\ \text{covariance matrix}\end{matrix} \tag{1.23}$$

transformation into the eigenbasis of $\underline{\mathbf{C}}$

let: $\lambda_1 > \lambda_2 > \ldots > \lambda_N \leftarrow$ eigenvalues

$$\underline{\mathbf{w}} = a_1\underline{\mathbf{e}}_1 + a_2\underline{\mathbf{e}}_2 + \ldots + a_N\underline{\mathbf{e}}_N \leftarrow \text{ corresponding eigenvectors} \tag{1.24}$$

then:

$$\Delta a_j = \varepsilon\lambda_j a_j \tag{1.25}$$

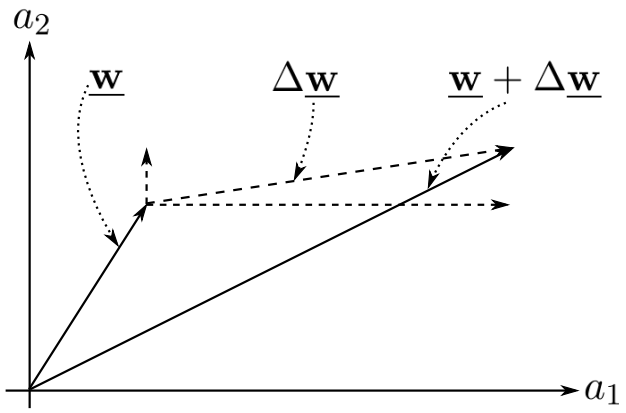


Figure 7: Learning via Ojas rule

$\rightarrow$ $|\underline{\mathbf{w}}| \rightarrow \infty$

$\rightarrow$ $\underline{\mathbf{e}}_{\mathrm{w}} = \frac{\underline{\mathbf{w}}}{|\underline{\mathbf{w}}|}$ converges to $\underline{\mathbf{e}}_1$ (eigenvector with the largest eigenvectors)

Note: in the Haykin book a proof for the more general stochastic algorithm can be found showing the generality of the statement.

Neurobiological implications

- receptive fields and coding
- adaptive tracking of the direction of largest variance: "on-line" PCA

Problem: $\|w\| \rightarrow \infty \rightarrow$ requires some form of normalization

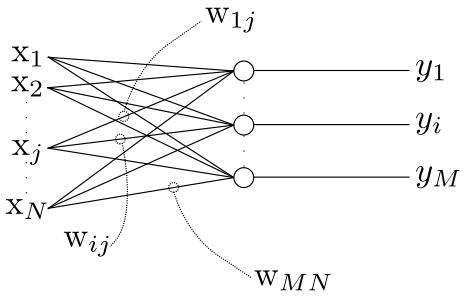
Solution: Normalization via Oja's rule

$$\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \left\{ \underbrace{\mathrm{x}_j^{(\alpha)}}_{\substack{\text{Hebbian} \\ \text{learning}}} - \underbrace{y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{w}_j}_{\substack{\text{decay} \\ \text{term}}} \right\} \tag{1.26}$$

Oja's rule converges to the unit vector which points into the direction of the largest variance

proof: supplementary material

Hebbian PCA (generalized Hebbian algorithm):



M linear neurons: $y_i = \underline{\mathbf{w}}_i^T \underline{\mathbf{x}} = \sum_{j=1}^{N} w_{ij} x_j$, $i = 1, \ldots, M$
observations: $\underline{\mathbf{x}}^{(\alpha)}, \quad \alpha = 1, \ldots, p, \quad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$
The (feedforward) neural network extracts the M PCs with the largest eigenvalues

$\rightsquigarrow$ online-PCA for data with time-varying statistics

Extended learning (Sanger's rule)

$$\Delta w_{ij} = \varepsilon y_i \left\{ \underbrace{x_j}_{\text{Hebbian rule}} - \underbrace{\sum_{k=1}^{i} w_{kj} y_k}_{\sum_{k=1}^{i-1} w_{kj} y_k \text{ is added to Oja's rule}} \right\}$$

$\rightsquigarrow$ weights converge to the M eigenvectors with the largest eigenvalues

$$\begin{aligned} \underline{\mathbf{w}}_1 &\to \underline{\mathbf{e}}_1 \\ \underline{\mathbf{w}}_2 &\to \underline{\mathbf{e}}_2 \\ &\vdots \\ \underline{\mathbf{w}}_M &\to \underline{\mathbf{e}}_M \end{aligned}$$

$\rightsquigarrow$ $y_i = \underline{\mathbf{e}}_i^T \underline{\mathbf{x}} =: a_i$ after learning

Learning: Oja's rule & Gram-Schmidt orthonormalization
Sanger's rule: $\Delta w_{ij} = \varepsilon y_i \left\{ x_j - \sum_{k=1}^{i} w_{kj} y_k \right\}$

- Define $\hat{\mathrm{x}}_j^{(i)} := \mathrm{x}_j - \sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k$
- Then $\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \hat{\mathrm{x}}_j^{(i)} - y_j \mathrm{w}_{ij} \right\} \longrightarrow$ Oja's rule with modified input

Case $i = 1$::
$\hat{\mathrm{x}}_j^{(1)} = \mathrm{x}_j \quad \rightsquigarrow$ original form of Oja's rule
$\rightsquigarrow$ $\underline{\mathbf{w}}_1$ converges to eigenvector $\pm \underline{\mathbf{e}}_1$

Case $i = 2$:
$\hat{\mathrm{x}}_j^{(2)} = \mathrm{x}_j - \mathrm{w}_{1j} y_1$
$\underline{\mathbf{w}}_1 = \underline{\mathbf{e}}_1 \rightarrow y_1 = \underline{\mathbf{x}}^T \underline{\mathbf{e}}_1 =: a_1$
& $\hat{\mathrm{x}}_j^{(2)} = \mathrm{x}_j - (\underline{\mathbf{e}}_1)_j \, a_1$

$\rightarrow$ $\hat{\underline{\mathbf{x}}}^{(2)}$ is the projection of $\underline{\mathbf{x}}$ onto subspace orthogonal to $\underline{\mathbf{e}}_1$:

$\rightarrow$ $\underline{\mathbf{w}}_2$ converges to $\pm \underline{\mathbf{e}}_2$ by Oja's rule since $\underline{\mathbf{e}}_2$ is the direction of largest variance in that subspace

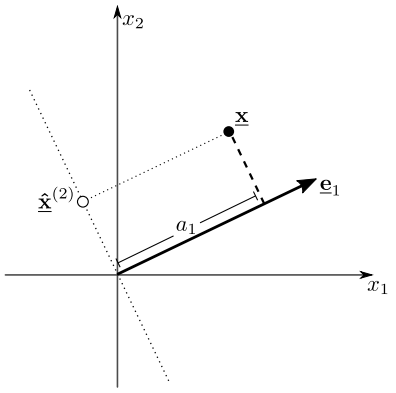


Figure 8: novelty orthonormalization

Case $i = 3$:
$\hat{\mathrm{x}}_j^{(3)} = \mathrm{x}_j - \mathrm{w}_{1j} y_1 - \mathrm{w}_{2j} y_2$

$\underline{\mathbf{w}}_1 = \underline{\mathbf{e}}_1$ & $\underline{\mathbf{w}}_2 = \underline{\mathbf{e}}_2 \rightarrow y_1 = a_1, \;\; y_2 = \underline{\mathbf{x}}^T \;\; \underline{\mathbf{e}}_2 =: a_2$ & $\hat{\mathrm{x}}_j^{(3)} = \mathrm{x}_j - a_1 \, (\underline{\mathbf{e}}_1)_j - a_2 \, (\underline{\mathbf{e}}_2)_j$

$\rightarrow$ $\hat{\underline{\mathbf{x}}}^{(3)}$ is the projection of $\underline{\mathbf{x}}$ onto subspace orthogonal to span $\{\underline{\mathbf{e}}_1, \underline{\mathbf{e}}_2\}$

$\rightarrow$ $\underline{\mathbf{w}}_3$ converges to $\pm \underline{\mathbf{e}}_3$ by Oja's rule

Case $i = M$:
$\hat{\mathrm{x}}_j^{(M)} = \mathrm{x}_j - \sum_{k=1}^{M-1} \mathrm{w}_{kj} y_k$
$\underline{\mathbf{w}}_k = \underline{\mathbf{e}}_k$ for $k = 1, \ldots, M-1 \rightarrow y_k = \underline{\mathbf{x}}^T \underline{\mathbf{e}}_k =: a_k$
& $\hat{\mathrm{x}}_j^{(M)} = \mathrm{x}_j - \sum_{k=1}^{M-1} a_k \, (\underline{\mathbf{e}}_k)_j$

$\rightarrow$ $\hat{\underline{\mathbf{x}}}^{(M)}$ is the proj. of $\underline{\mathbf{x}}$ onto subspace orthogonal to span $\left\{\underline{\mathbf{e}}_1, \ldots, \underline{\mathbf{e}}_{M-1}\right\}$

$\rightarrow$ $\underline{\mathbf{w}}_M$ converges to $\pm \underline{\mathbf{e}}_M$ by Oja's rule

Summary of Hebbian learning:
I. Hebbian learning without constraint

$$\underbrace{\Delta\underline{\mathbf{w}} = \varepsilon y \underline{\mathbf{x}}}_{\text{Hebb's rule}} \rightsquigarrow \lim_{t\to\infty} \underline{\mathbf{w}} || \underline{\mathbf{e}}_1 \text{ (orthogonal)}$$

$\rightsquigarrow$ weights converge to direction of largest variance in the data
$\rightsquigarrow$ but: $|\underline{\mathbf{w}}| \to \infty$ for $t \to \infty$

II. Hebbian learning with normalization

$$\underbrace{\Delta\underline{\mathbf{w}} = \varepsilon y \left(\underline{\mathbf{x}} - y\underline{\mathbf{w}}\right)}_{\text{Oja's rule}} \rightsquigarrow \lim_{t\to\infty} \underline{\mathbf{w}} \in \{+\underline{\mathbf{e}}_1, -\underline{\mathbf{e}}_1\}$$

$\rightsquigarrow$ weights remain finite: $|\underline{\mathbf{w}}| = 1$

III. Hebbian PCA with M neurons and normalization

$$\underbrace{\Delta \mathrm{w}_{ij} = \varepsilon y_i \left\{ \mathrm{x}_j - \sum_{k=1}^{i} \mathrm{w}_{kj} y_k \right\}}_{\text{Sanger's rule}} \rightsquigarrow \lim_{t\to\infty} \underline{\mathbf{w}}_i \in \{+\underline{\mathbf{e}}_i, -\underline{\mathbf{e}}_i\}, i = 1, \ldots, M$$

$\rightsquigarrow$ combination of Oja's rule & Gram-Schmidt-orthonormalization

1.3 Kernel Principal Component Analysis

Kernel PCA extends the linear dimensionality reduction approach of PCA to extract nonlinear structure.

1.3.1 Non-linear Manifolds

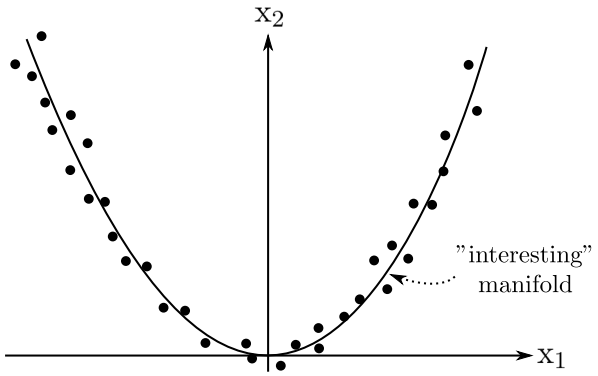


Figure 9: Example of a nonlinear dependency

Consider the data represented in the original space:

- standard PCA: two directions with high variance
- "interesting" feature is a non-linear combination of elementary features

$\rightsquigarrow$ this dependency is not properly extracted by standard PCA

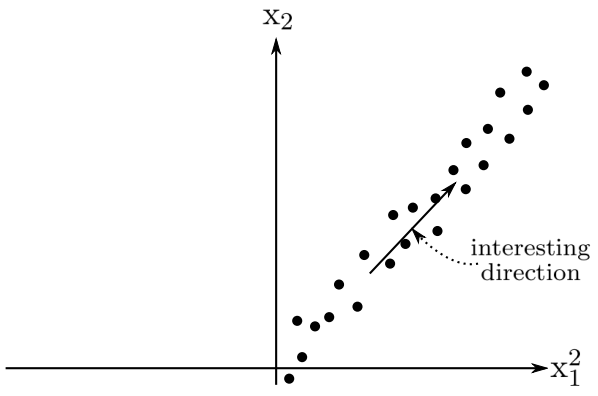


Figure 10: Nonlinear dependency becomes linear in transformed space

Idea: analyse data in *transformed* (feature) space:

$\rightsquigarrow$ standard PCA: one direction of high variance

$\rightsquigarrow$ "interesting" feature will be discovered

Agenda: non-linear preprocessing, then application of a standard linear method $\Rightarrow$ non-linear analysis method, i.e. given a set of observations: $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

(1) transformation into an "appropriate" feature space

$$\underline{\phi} : \underline{\mathbf{x}} \rightarrow \underline{\phi}_{(\underline{\mathbf{x}})} \tag{1.27}$$

(2) apply *linear* analysis method on $\underline{\phi}_{(\underline{\mathbf{x}})}$

Problem: Some feature spaces may be extremely high-dimensional

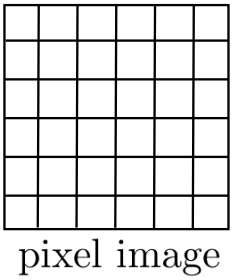


Figure 11: Pixel image as a high dimensional patterns

$\rightsquigarrow$ interesting structure is hidden in correlations between pixel values

$\rightsquigarrow$ 'interesting" feature space: space spanned by all d^{th}-order monomials

$$\text{e.g. } d = 2 : \mathrm{x}_1^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_3, \mathrm{x}_2\mathrm{x}_3, \mathrm{x}_3^2, \ldots \tag{1.28}$$

$\rightsquigarrow$ number of dimensions

$$\frac{(N+d-1)!}{d!(N-1)!} = O\left(N^d\right) \tag{1.29}$$

Are there methods which work in the original space $\mathbb{R}^N$?
Yes! Use the kernel trick (*see also MI I, chapter 2.2.4*).

1.3.2 Reformulation of PCA using only Scalar Products

Observations: $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

Simplifying assumptions:

$$\frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} = \underline{\mathbf{0}} \rightsquigarrow \text{ zero mean data} \tag{1.30}$$

Correlation matrix $\underline{\mathbf{C}}$:

$$C_{ij} = \frac{1}{p}\sum_{\alpha=1}^{p} \mathrm{x}_i^{(\alpha)}\mathrm{x}_j^{(\alpha)} \Rightarrow \underline{\mathbf{C}} = \frac{1}{p}\sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \tag{1.31}$$

PCA requires a solution of the eigenvalue problem:

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \tag{1.32}$$

Expansion of the eigenvectors into data points (linear combination, not necessarily unique):

$$\underbrace{\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)}\underline{\mathbf{x}}^{(\beta)}}_{\substack{\text{always possible, because principal components} \\ \text{lie in the subspace of the data}}} \tag{1.33}$$

Insertion into the eigenvalue equation leads to:

$$\frac{1}{p}\sum_{\alpha,\beta=1}^{p} a_k^{(\beta)}\left[\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{x}}^{(\beta)}\right]\underline{\mathbf{x}}^{(\alpha)} = \lambda_k \sum_{\beta=1}^{p} a_k^{(\beta)}\underline{\mathbf{x}}^{(\beta)} \tag{1.34}$$

Multiplying with data points $\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T$, $\gamma = 1, \ldots, p$:

$$\frac{1}{p}\sum_{\alpha,\beta=1}^{p} a_k^{(\beta)}\left[\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{x}}^{(\beta)}\right]\left[\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T\underline{\mathbf{x}}^{(\alpha)}\right] = \lambda_k \sum_{\beta=1}^{p} a_k^{(\beta)}\left[\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T\underline{\mathbf{x}}^{(\beta)}\right] \tag{1.35}$$

Formulation of the eigenvalue problem in terms of scalar products:

$$K_{\alpha\beta} := \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{x}}^{(\beta)} \tag{1.36}$$

in matrix notation (generalized eigenvalue problem):

$$\underline{\mathbf{K}}^2\underline{\mathbf{a}}_k = p\lambda_k\underline{\mathbf{K}}\underline{\mathbf{a}}_k \tag{1.37}$$

Remark: $\underline{\mathbf{K}}$ is a positive semidefinite matrix! Why? Because for an arbitrary vector $\underline{\mathbf{y}}$:

$$\begin{aligned} \underline{\mathbf{y}}^T\underline{\mathbf{K}}\underline{\mathbf{y}} &= \sum_{\alpha,\beta=1}^{p} y^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\underline{\mathbf{x}}^{(\beta)}y^{(\beta)} \\ &= \left(\sum_{\alpha=1}^{p} y^{(\alpha)}\underline{\mathbf{x}}^{(\alpha)}\right)^2 \\ &\geq 0 \end{aligned} \tag{1.38}$$

Positive semidefiniteness of $\underline{\mathbf{K}}$ means, it has only non-negative eigenvalues (but potentially 0) → solutions of the following transformed (standard eigenvalue) problem differ only by components corresponding to uninteresting directions with zero eigenvalues (see e.g. **Bishop2006**).

Transformed eigenvalue problem:

$$\boxed{\underline{\mathbf{K}}\mathbf{a}_k = p\lambda_k \underline{\mathbf{a}}_k} \tag{1.39}$$

$\underline{\mathbf{K}}$: matrix of scalar products between data points

λ_k : variance along principal component $\underline{\mathbf{a}}_k$

$\underline{\mathbf{a}}_k$: principal component, represented in the - may be autocomplete - basis $\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \alpha = 1, \ldots, p$

Normalization of principal components

$$\begin{aligned} \underline{\mathbf{e}}_k^2 &= \sum_{\alpha,\beta=1}^{p} a_k^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)} a_k^{(\beta)} \\ &= \underline{\mathbf{a}}_k^T \underline{\mathbf{K}}\underline{\mathbf{a}}_k \\ &= \lambda_k \underline{\mathbf{a}}_k^2 p \\ &\stackrel{!}{=} 1 \end{aligned} \tag{1.40}$$

$$\boxed{\underline{\mathbf{a}}_k^{\text{norm.}} = \frac{1}{\sqrt{p}\sqrt{\lambda_k}|\underline{\mathbf{a_k}}|}\underline{\mathbf{a}}_k} \tag{1.41}$$

Projection u_k of a new data vector $\underline{\mathbf{x}}$ onto the principal components

$$\begin{aligned} u_k &= \underline{\mathbf{e}}_k^T \cdot \underline{\mathbf{x}} \\ &= \sum_{\beta=1}^{p} a_k^{(\beta)} \underbrace{\left[\left(\underline{\mathbf{x}}^{(\beta)}\right)^T \cdot \underline{\mathbf{x}}\right]}_{\text{scalar product}} \end{aligned} \tag{1.42}$$

Note: This formulation of PCA is exclusively in terms of scalar products

1.3.3 The Kernel Trick

Idea: perform PCA in a suitable feature space

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}_{(\underline{\mathbf{x}})} \tag{1.43}$$

Kernel trick:

$\Rightarrow$ avoid direct transformation $\underline{\phi}$

$\Rightarrow$ replace all scalar products by "kernel functions"

$$\underline{\phi}^T_{(\underline{\mathbf{x}})}\underline{\phi}_{(\underline{\mathbf{x}}')} \longleftrightarrow k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} \tag{1.44}$$

Mercer's theorem: every positive definite kernel k corresponds to a scalar product in some metric feature space[2]

Typical kernel functions

$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \left(\underline{\mathbf{x}}^T\underline{\mathbf{x}}' + 1\right)^d$ polynomial kernel of degree d *image processing (pixel correlation)*

$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \exp\left\{-\frac{\left(\underline{\mathbf{x}}-\underline{\mathbf{x}}'\right)^2}{2\sigma^2}\right\}$ RBF-kernel with range σ *infinite dimensional feature space*

$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \tanh\left\{K\underline{\mathbf{x}}^T\underline{\mathbf{x}}' + \theta\right\}$ neural network kernel with parameters K and θ *not necessarily positive definite*

$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \frac{1}{|\underline{\mathbf{x}}-\underline{\mathbf{x}}'+\varepsilon|^N}$ Plummer kernel with parameter ε *scale invariant kernel*

Note: no need to explicitly project into feature space – if an algorithm can be formulated solely in terms of scalar products, a non-linear version of it can be derived via this approach.

$\rightsquigarrow$ *cf. support vector machines (MI I)*

$\rightsquigarrow$ Fisher discriminant analysis, Canonical Correlation Analysis

$\rightsquigarrow$ K-means clustering & self-organizing maps (MI II)

Problem: derivation assumes zero mean – but normalization to zero mean in original space does not guarantee normalization in feature space

$$\frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)} \stackrel{!}{=} 0 \nrightarrow \frac{1}{p}\sum_{\alpha=1}^{p}\underline{\phi}\left(\underline{\mathbf{x}}^{(\alpha)}\right) = \underline{\mathbf{0}} \tag{1.45}$$

Solution: calculation of "centered" kernel matrix elements

data points $\underline{\mathbf{x}}^{(\alpha)}$: "training" data: $\alpha \in \{1,\ldots,p\}$; new (test) data: $\alpha > p$

$$\underbrace{\underline{\phi}\left(\underline{\mathbf{x}}^{(\alpha)}\right)}_{\substack{\text{"centered"}\\ \text{feature vectors}}} = \widetilde{\underline{\phi}}\left(\underline{\mathbf{x}}^{(\alpha)}\right) - \frac{1}{p}\sum_{\gamma=1}^{p}\underbrace{\widetilde{\underline{\phi}}\left(\underline{\mathbf{x}}^{(\gamma)}\right)}_{\substack{\text{uncentered}\\ \text{feature vectors}}} \tag{1.46}$$

[2] for details see MI I, chapt. 2.2.4

$$
\begin{aligned}
K_{\alpha\beta} &= \underline{\phi}^T\big(\underline{\mathbf{x}}^{(\alpha)}\big) \cdot \underline{\phi}\big(\underline{\mathbf{x}}^{(\beta)}\big) \\
&= \left(\underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\alpha)}\big) - \tfrac{1}{p}\sum_{\gamma=1}^{p}\underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\gamma)}\big)\right)\left(\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\beta)}\big) - \tfrac{1}{p}\sum_{\delta=1}^{p}\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\delta)}\big)\right) \\
&= \underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\alpha)}\big)\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\beta)}\big) - \tfrac{1}{p}\sum_{\gamma=1}^{p}\underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\gamma)}\big)\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\beta)}\big) \qquad (1.47) \\
&\quad -\tfrac{1}{p}\sum_{\delta=1}^{p}\underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\alpha)}\big)\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\delta)}\big) + \tfrac{1}{p^2}\sum_{\delta=1}^{p}\underline{\widetilde{\phi}}^T\big(\underline{\mathbf{x}}^{(\gamma)}\big)\underline{\widetilde{\phi}}\big(\underline{\mathbf{x}}^{(\delta)}\big) \\
&= \widetilde{K}_{\alpha\beta} - \tfrac{1}{p}\sum_{\gamma=1}^{p}\widetilde{K}_{\gamma\beta} - \tfrac{1}{p}\sum_{\gamma=1}^{p}\widetilde{K}_{\alpha\gamma} + \tfrac{1}{p^2}\sum_{\gamma,\delta}\widetilde{K}_{\gamma\delta}
\end{aligned}
$$

1.3.4 Summary of the Kernel-PCA Method

(1) calculate the un-normalized kernel matrix $\underline{\widetilde{\mathbf{K}}}$

$$\widetilde{K}_{\alpha\beta} = \underbrace{k\big(\underline{\mathbf{x}}^{(\alpha)},\underline{\mathbf{x}}^{(\beta)}\big)}_{\substack{\text{kernel}\\ \text{function}}} \qquad (1.48)$$

(2) normalize to zero mean

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p}\sum_{\gamma=1}^{p}\widetilde{K}_{\gamma\beta} - \frac{1}{p}\sum_{\gamma=1}^{p}\widetilde{K}_{\alpha\gamma} + \frac{1}{p^2}\sum_{\gamma,\delta}\widetilde{K}_{\gamma\delta} \qquad (1.49)$$

(3) solve the eigenvalue problem

$$\underline{\mathbf{K}}\underline{\widetilde{\mathbf{a}}}_k = p\lambda_k\underline{\widetilde{\mathbf{a}}}_k \qquad (1.50)$$

(4) normalize eigenvectors to unit length (in feature space)

$$\underline{\mathbf{a}}_k = \frac{1}{\sqrt{p\lambda_k}\,||\underline{\widetilde{\mathbf{a}}}_k||}\underline{\widetilde{\mathbf{a}}}_k \qquad (1.51)$$

(5) calculate projections of data points $\underline{\mathbf{x}}^{(\alpha)}$ onto eigenvectors

$$u_k(\underline{\mathbf{x}}^{(\alpha)}) = \sum_{\beta=1}^{p} a_k^{(\beta)} K_{\beta\alpha} \leftarrow \begin{matrix}\text{use normalized}\\ \text{matrix element!}\end{matrix} \qquad (1.52)$$

More generally, for arbitrary $\underline{\mathbf{x}}$ the projection is computed as:

$$
\begin{aligned}
u_k(\underline{\mathbf{x}}) &= \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\phi}^T_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} \underline{\phi}_{(\underline{\mathbf{x}})} \qquad \leftarrow \text{centered feature vectors} \\
&= \sum_{\beta=1}^{p} a_k^{(\beta)} \Big(\Big[\underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} - \frac{1}{p} \sum_{\gamma=1}^{p} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\gamma)}\right)} \Big]^T \Big[\underline{\widetilde{\phi}}_{(\underline{\mathbf{x}})} - \frac{1}{p} \sum_{\delta=1}^{p} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\delta)}\right)} \Big] \Big) \\
&= \sum_{\beta=1}^{p} a_k^{(\beta)} \Big(k(\underline{\mathbf{x}}^{(\beta)}, \underline{\mathbf{x}}) - \frac{1}{p} \sum_{\delta=1}^{p} \widetilde{K}_{\beta\delta} - \frac{1}{p} \sum_{\gamma=1}^{p} k(\underline{\mathbf{x}}^{(\gamma)}, \underline{\mathbf{x}}) + \frac{1}{p^2} \sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta} \Big)
\end{aligned}
$$

Comments:

- kernel-PCA $\widehat{=}$ PCA in feature space
  - $\rightarrow$ "principal components" are uncorrelated
  - $\rightarrow$ λ_k: variance of the data along principal component k (in feature space)
  - $\rightarrow$ mean-squared approximation error (in feature space) in representing the observations by components with largest eigenvalues is minimal
- for KPCA, number of principal components can exceed number of dimensions in the original space
- reconstruction is harder as in linear PCA: no explicit data space representation of the principal component subspace
  - typically only projection coefficients $u_k(\underline{\mathbf{x}})$ are used
  - data space is mapped to low-dim. manifold in feature space
  - for general feature vectors: no pre-image in data space
  - but: approximative pre-images can be computed (application: denoising)
- Kernel PCA can be used for feature extraction, e.g., to solve classification problems in feature space
- Optimal kernel parameters (σ, d, etc.) depend on data & task (selection via cross-validation possible for classification tasks but in general no measure-of-goodness of the PC projections available)
- Custom kernels can be used (any positive definite kernel matrix)
- expansion of principal components into data points is <u>not</u> sparse $\rightsquigarrow$ high computational burden when calculating projections

possible solution: approximate principal components using expansions with less data vectors $q < p$:

$$\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\phi}\left(\underline{\mathbf{x}}^{(\beta)}\right) \quad \text{principal (normalized) components } k$$

$$\widehat{\underline{\mathbf{e}}}_k = \sum_{\gamma=1}^{q} \widehat{a}_k^{(\gamma)} \underbrace{\underline{\phi}\left(\underline{\mathbf{z}}^{(\gamma)}\right)}_{} \quad \text{approximation by } q \text{ data points } \left(\text{in new order, } \underline{\mathbf{z}}^{(\gamma)} = \underline{\mathbf{x}}^{(i_\gamma)}\right)$$

quadratic error φ:

$$\begin{aligned}
\varphi &= \left(\underline{\mathbf{e}}_k - \widehat{\underline{\mathbf{e}}}_k\right)^2 \\
&= \underbrace{|\underline{\mathbf{e}}_k|^2}_{=1} + |\widehat{\underline{\mathbf{e}}}_k|^2 - 2\underline{\mathbf{e}}_k^T \widehat{\underline{\mathbf{e}}}_k \\
&= 1 + \sum_{\gamma,\delta=1}^{q} \widehat{a}_k^{(\gamma)} \widehat{a}_k^{(\delta)} k\left(\underline{\mathbf{z}}^{(\gamma)}, \underline{\mathbf{z}}^{(\delta)}\right) - 2 \sum_{\beta=1}^{p} \sum_{\gamma=1}^{q} a_k^{(\beta)} \widehat{a}_k^{(\gamma)} k\left(\underline{\mathbf{x}}^{(\beta)}, \underline{\mathbf{z}}^{(\gamma)}\right)
\end{aligned}$$

choose $\widehat{a}_k^{(\gamma)}$ and $\underline{\mathbf{z}}^{(\gamma)}$ such that

$$\varphi \stackrel{!}{=} \min$$

e.g. using gradient-based methods

- kernel matrices may be very large

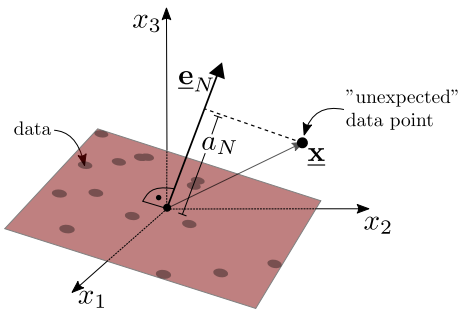
  → only eigenvectors to the largest eigenvalues are of interest

  → use specialized routines (**PressEtAl2007** chapt. 11.5 - 11.7)

□ toy example (Schölkopf et al., MPI Biol. Cybern. Technical Report 44, Fig. 2)

1.4 Novelty Filter

Principal Components with smallest eigenvalues (e.g., $\underline{\mathbf{e}}_N$):

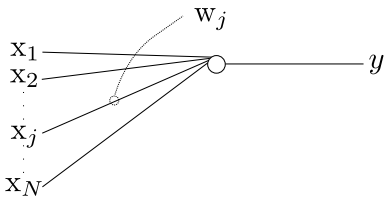


⇝ outliers / data with novel features can be identified by projecting to last PCs

$\rightsquigarrow$ $y = \underline{\mathbf{e}}_N^T \underline{\mathbf{x}} =: a_N$ after learning $\rightsquigarrow$ projection onto smallest PC

$\rightsquigarrow$ large output for unexpected data $\rightarrow$ Novelty Filter

Novelty Filter: On-line Learning



$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$$

Anti-Hebbian rule:

$$\Delta \mathrm{w}_j = \overbrace{-}^{\text{"Anti"-Hebbian}} \varepsilon y^{(\alpha)} \mathrm{x}_j^{(\alpha)} \tag{1.53}$$

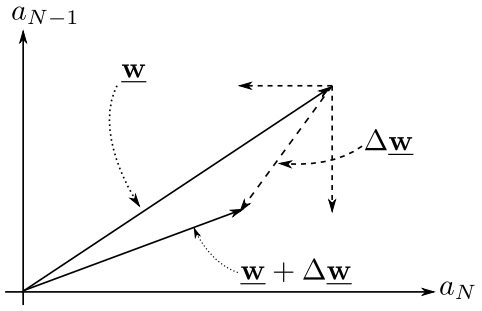
Conjecture:
$\underline{\mathbf{w}}$ converges to the direction of smallest eigenvector.
Proof:
Learning rule:

$$\Delta \mathrm{w}_j = -\varepsilon y^{(\alpha)} \mathrm{x}_j^{(\alpha)} \tag{1.54}$$

Assume small learning steps $\rightarrow$ average over all patterns

$$\Delta \mathrm{w}_j = -\frac{\varepsilon}{p} \sum_{\alpha=1}^{p} y^{(\alpha)} \mathrm{x}_j^{(\alpha)} = -\frac{\varepsilon}{p} \sum_{\alpha=1}^{p} \mathrm{x}_j^{(\alpha)} \sum_{k=1}^{N} \mathrm{x}_k^{(\alpha)} \mathrm{w}_k = -\varepsilon \sum_{k=1}^{N} C_{jk} \mathrm{w}_k \tag{1.55}$$

$$\Delta \underline{\mathbf{w}} = -\varepsilon \underline{\mathbf{C}\mathbf{w}} \tag{1.56}$$



Transformation into eigenbasis of covariance matrix:

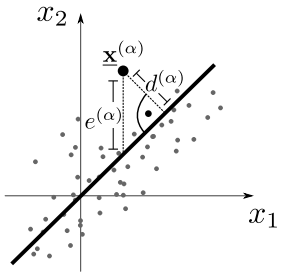
$$\underline{\mathbf{w}} = a_1 \underline{\mathbf{e}}_1 + a_2 \underline{\mathbf{e}}_2 + \cdots + a_N \underline{\mathbf{e}}_N$$
$$\Delta a_j = -\varepsilon \lambda_j a_j s$$

$\rightsquigarrow$ for $\lambda_j > 0 : a_j \to 0$, constraints required

$\rightsquigarrow$ for $\lambda_j = 0 : a_j$ remains unchanged

$\rightsquigarrow$ weights converge to the eigenvector with the smallest eigenvalue

Novelty Filter and linear regression



Ordinary least squares:

$$\frac{1}{p}\sum_{\alpha=1}^{p}\left(e^{(\alpha)}\right)^2 \stackrel{!}{=} \min.$$

$\rightsquigarrow$ correct if data points are noisy along x_2-component only

$\rightsquigarrow$ wrong if data points are also noisy along x_1-component

total least squares:

$$\frac{1}{p}\sum_{\alpha=1}^{p}\left(d^{(\alpha)}\right)^2 \stackrel{!}{=} \min.$$

tacit assumption: same variance noise
centered data $\rightarrow \underline{\mathbf{w}}^T\underline{\mathbf{x}} = 0$

Cost function:

$$\mathbb{E}(\underline{\mathbf{w}}) = \frac{1}{p}\sum_{\alpha=1}^{p}\left(d^{(\alpha)}\right)^2 \stackrel{!}{=} \min_{\underline{\mathbf{w}}} \quad \text{s.t.}|\underline{\mathbf{w}}| = 1$$

$$\underline{\mathbf{w}}^T\mathbf{C}\underline{\mathbf{w}} \stackrel{!}{=} \min_{\underline{\mathbf{w}}} \quad \text{s.t.}|\underline{\mathbf{w}}| = 1$$

solution:
$\underline{\mathbf{w}}$ is the normalized eigenvector to the smallest eigenvalue of the covariance matrix.

Novelty Filter with normalization

$$\Delta \underline{\mathrm{w}} = -\varepsilon \frac{y^{(\alpha)} \left\{ \underline{\mathbf{x}}^{(\alpha)} - y^{(\alpha)} \underline{\mathbf{w}} \right\}}{\left| \underline{\mathbf{w}} - \varepsilon y^{(\alpha)} \left\{ \underline{\mathbf{x}}^{(\alpha)} - y^{(\alpha)} \underline{\mathbf{w}} \right\} \right|}$$

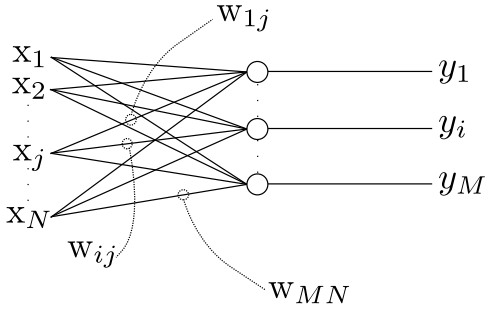
Anti-Hebbian version of Oja's rule:

$$\Delta \mathrm{w}_j = -\varepsilon y^{(\alpha)} \Big\{ \overbrace{\mathrm{x}_j^{(\alpha)}}^{\substack{\text{Anti-Hebbian} \\ \text{learning}}} \underbrace{-y^{(\alpha)} \mathrm{w}_j \Big\} + \varepsilon \Big\{ 1 - \overbrace{\sum_{k=1}^{N} \mathrm{w}_k^2}^{=|\underline{\mathbf{w}}|^2} \Big\} \mathrm{w}_j}_{\text{normalization}}$$

$\rightsquigarrow$ $\underline{\mathbf{w}}$ converges to $\pm \underline{\mathbf{e}}_N$

Feedforward network as a Novelty Filter
Extension of the learning rule to N neurons:

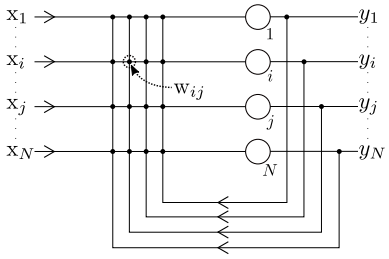
$$\Delta \mathrm{w}_{ij} = -\varepsilon y_i^{(\alpha)} \Big\{ \mathrm{x}_j^{(\alpha)} - \overbrace{\sum_{k=1}^{i} \mathrm{w}_{kj} y_k^{(\alpha)}}^{-\sum_{k=1}^{i-1} \mathrm{w}_{kj} y_k^{(\alpha)} \text{ is added}} \Big\} + \varepsilon \Big\{ 1 - \sum_{k=1}^{N} \mathrm{w}_{ik}^2 \Big\} \mathrm{w}_{ij}$$



$\rightsquigarrow$ result:

$\underline{\mathbf{w}}_1 \to \underline{\mathbf{e}}_N$	(PC with smallest eigenvalue)
$\underline{\mathbf{w}}_2 \to \underline{\mathbf{e}}_{N-1}$	$\vdots$
$\vdots$	$\vdots$
$\underline{\mathbf{w}}_M \to \underline{\mathbf{e}}_{N-M+1}$	(PC with largest eigenvalue, if $N = M$)

Sequential calculation of eigenvectors (cf. Sanger's rule).



Recurrent network as a Novelty Filter

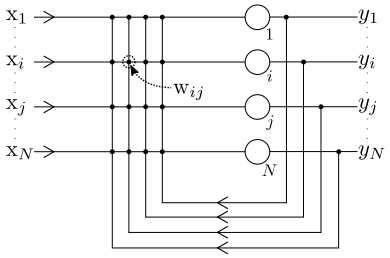
$$y_i^{(\alpha)}(t+1) = \sum_{j=1}^{N} \mathrm{w}_{ij} y_j^{(\alpha)}(t) + \mathrm{x}_i^{(\alpha)}$$
$$\underline{\mathbf{y}}^{(\alpha)}(t+1) = \underline{\mathbf{W}}\underline{\mathbf{y}}^{(\alpha)}(t) + \underline{\mathbf{x}}^{(\alpha)}$$

Stationary state:

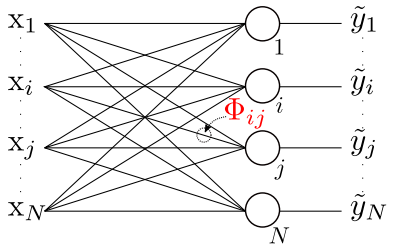
$$\underline{\tilde{\mathbf{y}}}^{(\alpha)}(t+1) = \underline{\tilde{\mathbf{y}}}^{(\alpha)}(t) =: \underline{\tilde{\mathbf{y}}}^{(\alpha)}$$
$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \left(\underline{\mathbf{I}} - \underline{\mathbf{W}}\right)^{-1} \underline{\mathbf{x}}^{(\alpha)}$$

Convergence is guaranteed, if weight matrix is symmetric.
Learning rule:

$$\Delta \mathrm{w}_{ij} = -\varepsilon \tilde{y}_i \tilde{y}_j$$



$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \left(\underline{\mathbf{I}} - \underline{\mathbf{W}}\right)^{-1} \underline{\mathbf{x}}^{(\alpha)}$$
$$\Delta \underline{\mathbf{w}} = -\varepsilon \underline{\tilde{\mathbf{y}}}^{(\alpha)} \left(\underline{\tilde{\mathbf{y}}}^{(\alpha)}\right)^T$$



$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \underline{\mathbf{\Phi}}\underline{\mathbf{x}}^{(\alpha)}$$
$$\Delta \underline{\mathbf{\Phi}} = -\varepsilon \underline{\mathbf{\Phi}}^2 \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{\Phi}}^2$$

Recurrent network as a Novelty Filter - Algorithm

initialization of $\underline{\mathbf{\Phi}}$ with identity matrix, $\underline{\mathbf{\Phi}} = \underline{\mathbf{I}}$
repeat
  choose an observation $\underline{\mathbf{x}}^{(\alpha)}$
  change weight matrix according to:

$$\Delta \underline{\mathbf{\Phi}} = -\varepsilon \underline{\mathbf{\Phi}}^2 \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{\Phi}}^2$$

until *convergence*

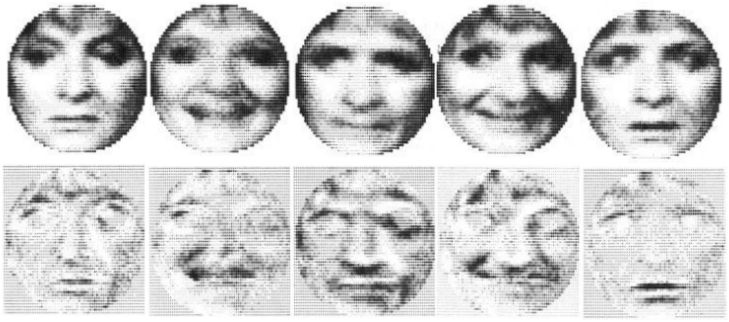
$\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ converges to a matrix, which projects onto the subspace orthogonal to the training data

example:

training data $\left\{\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)}\right\} \subseteq \text{span}\left\{\underline{\mathbf{e}}_1, \ldots, \underline{\mathbf{e}}_{N-1}\right\} \qquad \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

$$\underline{\mathbf{\Phi}} \xrightarrow[t\to\infty]{} \underline{\mathbf{I}} - \sum_{k=1}^{N-1} \underline{\mathbf{e}}_k \underline{\mathbf{e}}_k^T \rightsquigarrow \underline{\mathbf{\Phi}}\underline{\mathbf{e}}_j = \underline{\mathbf{e}}_N \delta_{jN} \rightsquigarrow \underline{\mathbf{\Phi}}\underline{\mathbf{x}} = \underbrace{\left(\underline{\mathbf{e}}_N^T \underline{\mathbf{x}}\right)}_{=:a_N} \underline{\mathbf{e}}_N$$

Recurrent network as a Novelty Filter - Neural facial expressions



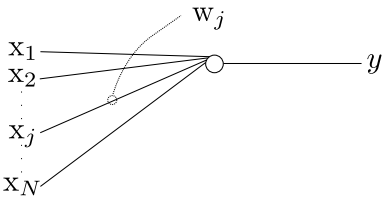
(taken from Kohonen 1989)

- training data: "neutral" facial expressions (not shown here)

  $\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ projects into space orthogonal to this data

- top row: faces $\underline{\mathbf{x}}^{(\beta)}$ with different expressions
- bottom row: projection $\underline{\mathbf{\Phi}}\underline{\mathbf{x}}^{(\beta)}$

Novelty Filter: Summary
One neuron:

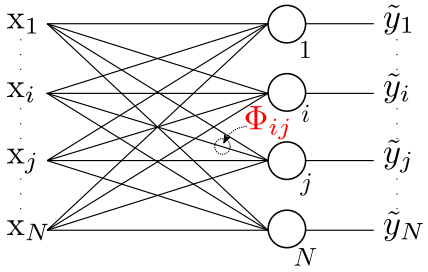


$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}}$

Anti-Hebbian learning rule

$$\Delta \mathrm{w}_j = -\varepsilon y^{(\alpha)} \Big\{ \overbrace{\mathrm{x}_j^{(\alpha)}}^{\text{Anti-Hebbian learning}} \underbrace{-y^{(\alpha)} \mathrm{w}_j \Big\} + \varepsilon \Big\{ 1 - \overbrace{\sum_{k=1}^{N} \mathrm{w}_k^2}^{=|\underline{\mathbf{w}}|^2} \Big\} \mathrm{w}_j}_{\text{normalization}}$$

N **neurons:**



$$\underline{\tilde{\mathbf{y}}}^{(\alpha)} = \underline{\mathbf{\Phi}} \underline{\mathbf{x}}^{(\alpha)}$$

Learning rule:

$$\Delta \underline{\mathbf{\Phi}} = -\varepsilon \underline{\mathbf{\Phi}}^2 \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} \right)^T \underline{\mathbf{\Phi}}^2$$

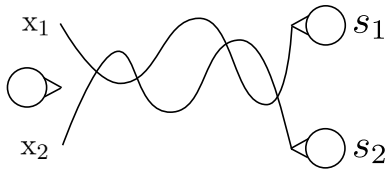
$\rightsquigarrow$ $\underline{\mathbf{\Phi}}$ converges to projection matrix onto subspace orthogonal to training data

2 Independent Component Analysis

2.1 Independent Component Analysis

The methods described in this section use a linear generative model to extend decorrelation methods like PCA to find statistically independent sources.

The cocktail party problem: linear superposition of sources



$$\underbrace{\underline{\mathbf{x}}}_{\text{observations}} = \underbrace{\underline{\mathbf{A}}}_{\substack{\text{mixing}\\ \text{matrix}}} \underbrace{\underline{\mathbf{s}}}_{\text{sources}} \tag{2.57}$$

Figure 13: The cocktail party problem

Question: Can we recover the sources from the observations - with no prior knowledge about the mixing process (i.e. the matrix $\underline{\mathbf{A}}$)?

Yes - but we need to make assumptions about the statistical properties of the sources.

$\rightsquigarrow$ source separation methods differ in what prior knowledge they exploit

Approach 1 (direct)	$\underline{\mathbf{x}}^{(\alpha)} \overset{\text{iid}}{\sim} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})$ Data	*Approach* 2 (Infomax)
	$\Downarrow$	
$\hat{\underline{\mathbf{s}}} := \underline{\mathbf{W}}\,\underline{\mathbf{x}}$	Models	$\hat{u}_i := \hat{f}_i(\overbrace{e_i^T \underline{\mathbf{W}}\,\underline{\mathbf{x}}}^{=:\hat{s}_i})$
	$\Downarrow$	
$D_{\mathrm{KL}}(P_{\underline{\mathbf{s}}}(\hat{\underline{\mathbf{s}}}) \,\|\|\, \prod_i P_{s_i}(\hat{s}_i))$	Performance measure	$H(\hat{\underline{\mathbf{u}}})$
	$\Downarrow$	
$\min_{\underline{\mathbf{W}}} D_{\mathrm{KL}}$	Optimization	$\max_{\underline{\mathbf{W}}} H(\hat{\underline{\mathbf{u}}})$

Table 1: Overview of the 2 approaches: Note that the two approaches are equivalent if the "transition functions" $\hat{f}_i$ match the marginal distribution functions of the true independent sources i.e. $f_i = cdf(s_i)$.

Goal: recovery of independent sources $\hat{S}$ from observations

Procedure: Given observations $\underline{\mathbf{x}}$, find $\underline{\mathbf{W}}$ with

$$\underbrace{\underline{\widehat{\mathbf{s}}}}_{\substack{\text{"estimated}\\ \text{sources"}}} = \underbrace{\underline{\mathbf{W}}}_{\substack{\text{"unmixing}\\ \text{matrix"}}} \underbrace{\underline{\mathbf{x}}}_{\text{observations}} \tag{2.58}$$

such that the joint distribution of sources factorizes:

$$P_{\underline{\mathbf{s}}}(\underline{\widehat{\mathbf{s}}}) = \prod_{i=1}^{N} P_{\underline{\mathbf{s}}}(\widehat{s}_i) \tag{2.59}$$

In the special case of a noise-free mixing process, this means:

$$\underline{\mathbf{W}} = \underline{\mathbf{A}}^{-1} \tag{2.60}$$

Why not PCA? Decorrelation is not independence!

$$P_{\underline{\mathbf{s}}}(\underline{\mathbf{s}}) = P_{s_1}(s_1) \cdot P_{s_2}(s_2)$$

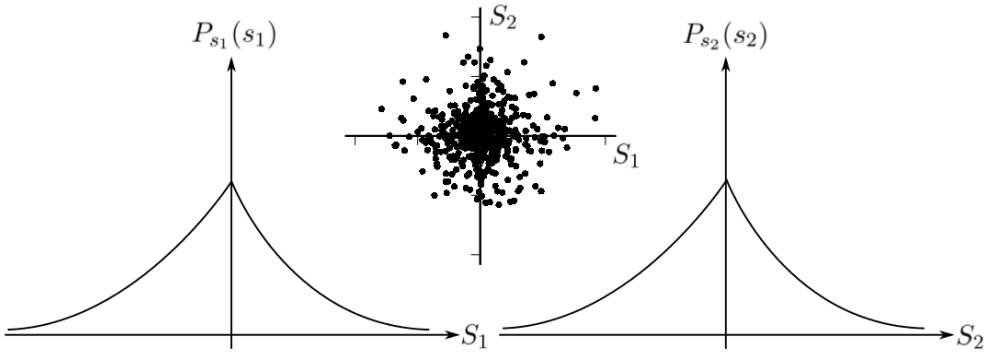


Figure 14: Marginal and joint distribution of independent variables

$$\underline{\mathbf{x}} = \underline{\mathbf{A}}\underline{\mathbf{s}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} a_{11}s_1 + a_{12}s_2 \\ a_{21}s_1 + a_{22}s_2 \end{pmatrix} \tag{2.61}$$

let $\underline{\mathbf{A}} = \left(\underline{\mathbf{a}}_1, \underline{\mathbf{a}}_2\right)$:

$$\begin{aligned} \underline{\mathbf{s}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightsquigarrow \underline{\mathbf{x}} = \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} = \underline{\mathbf{a}}_1 \quad \text{1st source} \\ \underline{\mathbf{s}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightsquigarrow \underline{\mathbf{x}} = \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} = \underline{\mathbf{a}}_2 \quad \text{2nd source} \end{aligned} \tag{2.62}$$

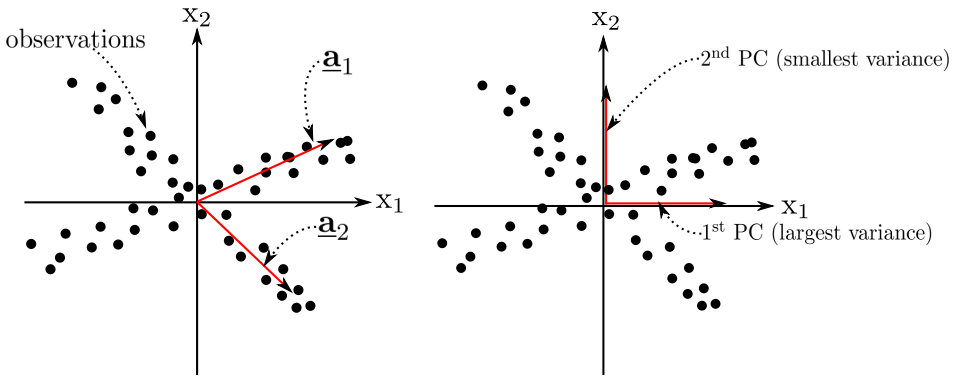


Figure 15: Motivation for ICA: In a set of observations, sources are "interesting" directions in feature space (left). Decorrelation (e.g. through PCA, right), might not find the relevant directions ⇒ new methods needed!

Limits to recovery

(1) permutations of sources

$$\begin{pmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \widehat{=} \quad \begin{pmatrix} \widehat{s}_2 \\ \widehat{s}_1 \end{pmatrix} = \begin{pmatrix} w_{21} & w_{22} \\ w_{11} & w_{12} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$P_{s_1}(\widehat{s}_1) \cdot P_{s_2}(\widehat{s}_2) \qquad\qquad P_{s_2}(\widehat{s}_2) \cdot P_{s_1}(\widehat{s}_1)$$

both alternatives are solutions to the unmixing problem

(2) amplitude of the sources

$$\begin{pmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \widehat{=} \quad \begin{pmatrix} a\widehat{s}_1 \\ b\widehat{s}_2 \end{pmatrix} = \begin{pmatrix} aw_{11} & aw_{12} \\ bw_{21} & bw_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$P_{s_1}(\widehat{s}_1) \cdot P_{s_2}(\widehat{s}_2) \qquad\qquad P_{s_1}(a\widehat{s}_1) \cdot P_{s_2}(b\widehat{s}_2)$$

both alternatives are solutions to the unmixing problem

(3) Gaussian distributions for sources and observations

$$\underline{\widehat{\mathbf{s}}} = \underline{\mathbf{W}\mathbf{x}}$$

Assuming whitened variables, the probability depends only on the length

of the vector, i.e. distance of the point from the origin.

$$\begin{aligned} P_{\underline{\mathbf{s}}}(\underline{\widehat{\mathbf{s}}}) &= \tfrac{1}{2\pi}\exp\left\{-\tfrac{\widehat{\mathbf{s}}^2}{2}\right\} \\ &= \underbrace{\left[\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\widehat{s_1}^2}{2}\right\}\right]\left[\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\widehat{s_2}^2}{2}\right\}\right]}_{\text{solution to the unmixing problem}} \end{aligned} \tag{2.63}$$

let $\underline{\mathbf{B}}$ be an orthogonal matrix: $\underline{\mathbf{B}}^T\underline{\mathbf{B}} = \underline{\mathbf{1}}$

$$\begin{aligned} \underline{\widetilde{\mathbf{s}}} &= \underline{\mathbf{B}}\underline{\widehat{\mathbf{s}}} \\ &= \underline{\mathbf{BWx}} \\ &= \underline{\mathbf{W}'\mathbf{x}} \end{aligned} \tag{2.64}$$

$$\begin{aligned} \underline{\widetilde{\mathbf{s}}}^2 := \|\underline{\widetilde{\mathbf{s}}}\|_2^2 &= \sum_{i=1}^{2}\widetilde{s}_i^2 \\ &= \sum_{i,k,l=1}^{2} B_{ik}\widehat{s}_k B_{il}\widehat{s}_l \\ &= \sum_{k,l=1}^{2}\widehat{s}_k\left(\sum_{i=1}^{2}B_{ki}^T B_{il}\right)\widehat{s}_l \\ &= \underline{\widehat{\mathbf{s}}}^T\underbrace{\left(\underline{\mathbf{B}}^T\underline{\mathbf{B}}\right)}_{\stackrel{!}{=}\underline{\mathbf{1}}}\underline{\widehat{\mathbf{s}}} \\ &= \underline{\widehat{\mathbf{s}}}^2 \end{aligned} \tag{2.65}$$

$$\underline{\widetilde{\mathbf{s}}} = \underline{\mathbf{W}'}\underline{\mathbf{x}} \tag{2.66}$$

$$\begin{aligned} P_{\underline{\mathbf{s}}}(\underline{\widetilde{\mathbf{s}}}) &= \tfrac{1}{2\pi}\exp\left\{-\tfrac{\widetilde{\mathbf{s}}^2}{2}\right\} \\ &= \underbrace{\left[\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\widetilde{s_1}^2}{2}\right\}\right]\left[\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\widetilde{s_2}^2}{2}\right\}\right]}_{\text{also solution to the unmixing problem}} \end{aligned} \tag{2.67}$$

2.2 Model-based Source Separation Techniques

2.2.1 The Infomax Principle

observations: $\underline{\mathbf{x}} \in \mathbb{R}^N, P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \leftarrow$ true (unknown) density

estimated sources: $\widehat{\underline{\mathbf{s}}} = \underline{\mathbf{W}}\underline{\mathbf{x}}, \widehat{\underline{\mathbf{s}}} \in \mathbb{R}^N, \underline{\mathbf{W}}$ regular $N \times N$ matrix

$\rightsquigarrow$ $P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}})$: family of true (unknown) densities, parametrized by $\underline{\mathbf{W}}$

Approach 1: direct model selection (cf. section 5.2)

Idea: select the model, which yields a distribution most similar to a factorizing density

$$\widehat{P}_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}}) = \prod_{i=1}^{N} \widehat{P}_{s_i}(\widehat{s}_i) \leftarrow \text{ ICA assumption} \tag{2.68}$$

This naturally leads to formulation of the following cost function

$$D_{KL} = \int d\widehat{\underline{\mathbf{s}}} P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}}) \ln \frac{P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}})}{\prod_{i=1}^{N} \widehat{P}_{s_i}(\widehat{s}_i)} \stackrel{!}{=} \min \tag{2.69}$$

which could be directly optimized. But: estimation of $\underline{\mathbf{W}} \mathrel{\widehat{=}}$ estimation of the probability density $P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}})$

$\rightsquigarrow$ parametrized density estimate required (which can be expensive)

Approach 2: Information Maximization (Bell & Sejnowski, 1995)

Idea: Under certain conditions, maximizing the *mutual information* between inputs (mixed signals) and outputs (recovered sources) of a system yields *independent* outputs.[3] Note that if sources are independent, then transformations of these sources are independent, too.

$\rightsquigarrow$ Choosing the transformation in a clever way (such that their marginal distributions become uniform, see below) simplifies the computations to find independent sources.

Excursion on Density Transformations: We will see that computations simplify if we choose a transformation $\widehat{f}_i$ leading to uniformly distributed sources, i.e. $\widehat{u}_i = \widehat{f}_i(\widehat{s}_i)$, such that $\widehat{P}_{u_i}(\widehat{u}_i) = \text{const}$.

The transformation can be found using *conservation of probability*

$$\widehat{P}_{u_i}(\widehat{u}_i) d\widehat{u}_i \;=\; \widehat{P}_{s_i}(\widehat{s}_i) d\widehat{s}_i. \tag{2.70}$$

Using the general rule for density transformations and applying it here yields

$$\widehat{P}_{u_i}(\widehat{u}_i) \quad = \quad \left| \frac{d\widehat{s}_i}{d\widehat{u}_i} \right| \widehat{P}_{s_i}(\widehat{s}_i) \quad = \quad \frac{1}{\left| \widehat{f}_i'(\widehat{s}_i) \right|} \widehat{P}_{s_i}(\widehat{s}_i) \tag{2.71}$$

where $\left| \frac{d\widehat{s}_i}{d\widehat{u}_i} \right|$ is called *functional determinant* of the transformation $\widehat{f}_i$. For a general transformation, this is illustrated in figure 16. Applying the principle

[3] For a more detailed description of this argument, see **BellSejnowski1995**.

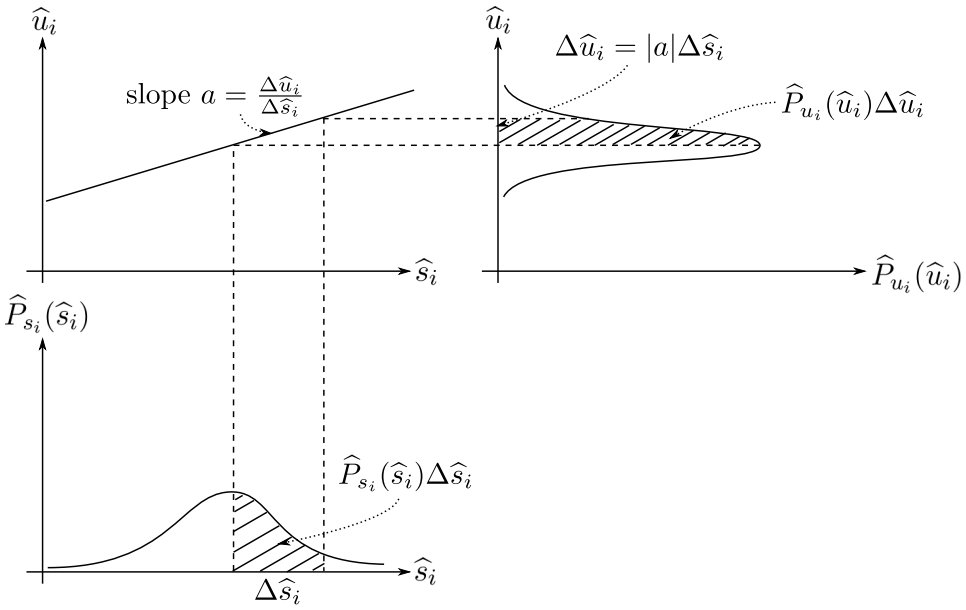


Figure 16: Illustration of a density transformation

of conservation of probability (equal size areas) to find a transformation resulting in a uniformly distributed variable with a constant density yields:

$$\widehat{P}_{u_i}(\widehat{u}_i) \;=\; \frac{1}{\left|\widehat{f}_i'(\widehat{s}_i)\right|}\widehat{P}_{s_i}(\widehat{s}_i) \;\stackrel{!}{=}\; const \qquad \Rightarrow \qquad \left|\widehat{f}_i'(\widehat{s}_i)\right| = a\widehat{P}_{s_i}(\widehat{s}_i) \quad (2.72)$$

and therefore

$$\Rightarrow \widehat{f}_i(\widehat{s}_i) = \int_{-\infty}^{\widehat{s}_i} dy\; a\widehat{P}_{s_i}(y) \quad (2.73)$$

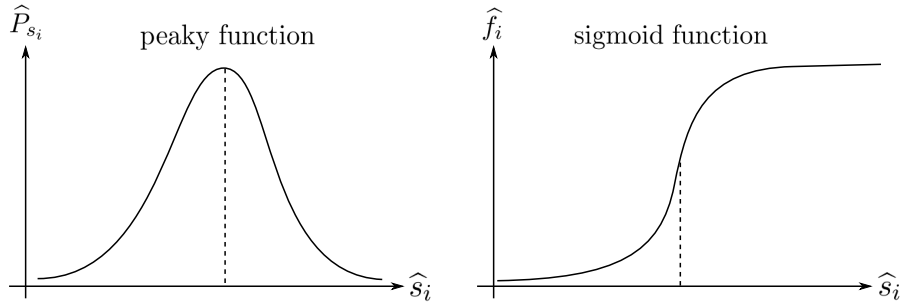


Figure 17: density (pdf) and corresponding distribution function (cdf)

Application of density transformations to solve the ICA problem:
KL-divergence for the transformed densities:

$$D_{KL} = \int d\underline{\mathbf{s}} P_{\underline{\mathbf{s}}} \ln \frac{P_{\underline{\widehat{\mathbf{s}}}}}{\prod_i P_{\widehat{s}_i}} \tag{2.74}$$

$$= \int d\underline{\mathbf{s}} P_{\underline{\widehat{\mathbf{s}}}} \ln \frac{P_{\underline{\widehat{\mathbf{s}}}} \quad \prod_i \frac{1}{f_i'}}{\prod_i P_{\widehat{s}_i} \quad \frac{1}{f_i'}} \quad = \int d\underline{\mathbf{u}} P_{\underline{\widehat{\mathbf{u}}}} \ln \frac{P_{\underline{\widehat{\mathbf{u}}}}}{\prod_i P_{\widehat{u}_i}} \tag{2.75}$$

$$= \underbrace{\int d\underline{\widehat{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}})}_{\substack{=-H(u) \\ \text{negative entropy} \\ \text{of the transformed} \\ \text{(true) density}}} - \int d\underline{\widehat{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \Bigg(\underbrace{\ln \prod_{i=1}^{N} \widehat{P}_{u_i}(\widehat{u}_i)}_{\text{const. for 'flat' } u_i} \Bigg) \tag{2.76}$$

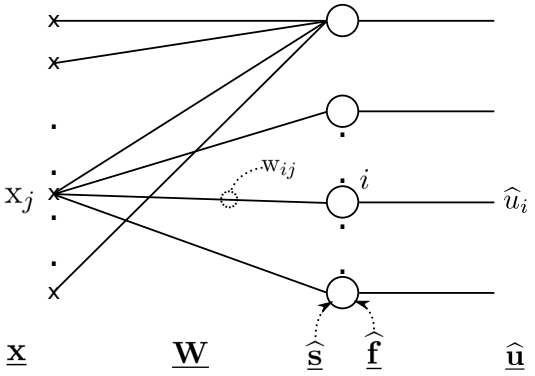
This motivates the socalled *Infomax principle* (**BellSejnowski1995**)

$$H = -\int d\underline{\widehat{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \stackrel{!}{=} \max \tag{2.77}$$

using the transformed estimated sources

$$\widehat{u}_i = \widehat{f}_i\big(\underbrace{\underline{\mathbf{e}}_i^T \underline{\mathbf{W}}\, \underline{\mathbf{x}}}_{=\widehat{s}_i}\big) \tag{2.78}$$

2.2.2 ICA via Neural Networks and Empirical Risk Minimization



perceptron network

$$u_i = \underbrace{\widehat{f}_i\Big(\sum_j \mathrm{w}_{ij}\mathrm{x}_j\Big)}_{\substack{\text{often a} \\ \text{sigmoid function}}}$$

observations: $\underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N, \alpha = 1, \ldots, p$

$\rightarrow$ determine the weights through inductive learning

derivation of the cost function

$$P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) d\underline{\widehat{\mathbf{u}}} = P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \tag{2.79}$$

$$\begin{aligned} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) &= \left|\frac{d\underline{\mathbf{x}}}{d\underline{\widehat{\mathbf{u}}}}\right| P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \\ &= \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{\left|\frac{d\underline{\widehat{\mathbf{u}}}}{d\underline{\mathbf{x}}}\right|} = \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{|\mathbf{M}|} \end{aligned} \tag{2.80}$$

with elements of the functional determinant $\mathbf{M}$ being given as

$$\begin{aligned} \mathbf{M}_{ij} = \frac{\partial \widehat{u}_i}{\partial \mathrm{x}_j} &= \frac{\partial}{\partial \mathrm{x}_j} \widehat{f}_i \left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k \right) \\ &= \mathrm{w}_{ij} \widehat{f}_i' \left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k \right). \end{aligned} \tag{2.81}$$

We therefore obtain for the value of the functional determinant

$$|\mathbf{M}| = \left|\frac{\partial \underline{\widehat{\mathbf{u}}}}{\partial \underline{\mathbf{x}}}\right| = |\underline{\mathbf{w}}| \prod_{l=1}^{N} \widehat{f}_l' \left(\sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k \right). \tag{2.82}$$

Inserting this into the Infomax cost function (2.77) gives

$$H = -\int d\underline{\widehat{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \tag{2.83}$$

$$= -\int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{|\mathbf{M}|} \tag{2.84}$$

$$= \underbrace{-\int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}_{\text{constant w.r.t. } \underline{\mathbf{w}}} + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln |\mathbf{M}| \tag{2.85}$$

and with (2.82) can be written in terms explicitly depending on w:

$$H = const + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln |\underline{\mathbf{w}}| + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \sum_{l=1}^{N} \ln \widehat{f}_l' \left(\sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k \right). \tag{2.86}$$

As mentioned above, the entropy can then be used for model selection:

$$E^G = \ln |\underline{\mathbf{w}}| + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \left\{ \sum_{l=1}^{N} \ln \widehat{f}_l' \left(\sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k \right) \right\} \quad \text{(generalization cost)}$$

Via the *principle of empirical risk minimization*

$$\text{mathematical expectation } E^G \longrightarrow \text{empirical average } E^T$$

the *training cost*

$$E^T = \ln |\underline{\mathbf{w}}| + \frac{1}{p} \sum_{\alpha=1}^{p} \sum_{l=1}^{N} \ln \widehat{f}_l' \left(\sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k^{(\alpha)} \right) \tag{2.87}$$

can be used for model selection using empirical data

$$\boxed{E^T \stackrel{!}{=} \max} \tag{2.88}$$

2.2.3 Learning by Gradient Ascent

Model parameters can be optimized by stepwise adustment along the direction of the gradient of the cost function.

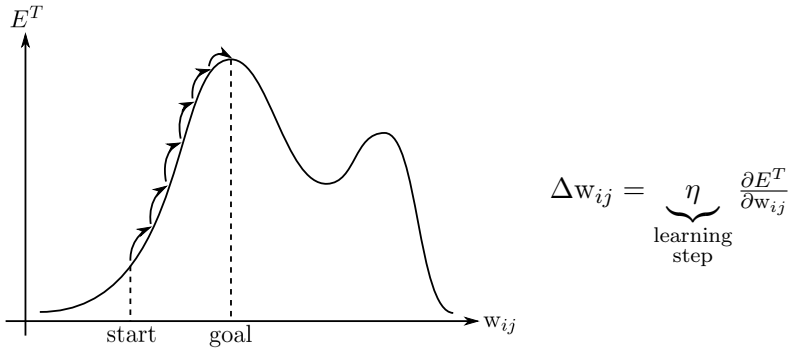


Figure 18: Gradient descent using the training cost

Taking partial derivatives of the training cost (2.87) wrt. the model parameters w_{ij} yields

$$\frac{\partial E^T}{\partial \mathrm{w}_{ij}} = \underbrace{\frac{1}{p}\sum_{\alpha=1}^{p}\sum_{l=1}^{N}\frac{\partial}{\partial \mathrm{w}_{ij}}\left\{\ln \widehat{f}_l\left(\sum_{k=1}^{N}\mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right)\right\}}_{\frac{1}{p}\sum_{\alpha=1}^{p}\frac{\widehat{f}_i''\left(\sum_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}\cdot\mathrm{x}_j^{(\alpha)}} + \underbrace{\frac{\partial}{\partial \mathrm{w}_{ij}}\left(\ln|\underline{\mathbf{w}}|\right)}_{\left(\underline{\mathbf{w}}^{-1}\right)_{ji}} \tag{2.89}$$

with individual costs:

$$e^{(\alpha)} = \ln|\underline{\mathbf{w}}| + \sum_{l=1}^{N}\ln \widehat{f}_l\left(\sum_{k=1}^{N}\mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right) \tag{2.90}$$

$$\frac{\partial e^{(\alpha)}}{\partial \mathrm{w}_{ij}} = \underbrace{\left(\underline{\mathbf{w}}^{-1}\right)_{ji}}_{\substack{\text{costly}\\ \text{computation}}} + \underbrace{\frac{\widehat{f}_i''\left(\sum_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}}_{:=\varphi_i^{(\alpha)}}\cdot\mathrm{x}_j^{(\alpha)} \tag{2.91}$$

this can be used for *batch-learning*:

$$\Delta \mathrm{w}_{ij} = \frac{\eta}{p}\sum_{\alpha=1}^{p}\frac{\partial e^{(\alpha)}}{\partial \mathrm{w}_{ij}} \tag{2.92}$$

or using *on-line-learning* (see algorithm 2).[4]

[4] see also MI I, chapters 1.3.4 and 1.4.1-1.4.3

Algorithm 2: On-line learning for ICA

$t \leftarrow 1$
random initialization of weights w_{ij}
begin
  $\eta_t = \frac{\eta_0}{t}$
  select next data point $\underline{\mathbf{x}}^{(\alpha)}$
  change all w_{ij} according to: $\Delta \mathrm{w}_{ij}^{(t)} = \eta_t \frac{\partial e_t^{(\alpha)}}{\partial \mathrm{w}_{ij}}$
  $t \leftarrow t + 1$
end

2.2.4 Natural Gradient Learning

Amari1998 describes a particularly efficient gradient descent algorithm to optimize the ICA-cost function. For details regarding the underlying theoretical framework, see **AmariEtAl2007**.

linear transformations: $d\underline{\mathbf{w}}, \underline{\mathbf{w}}$

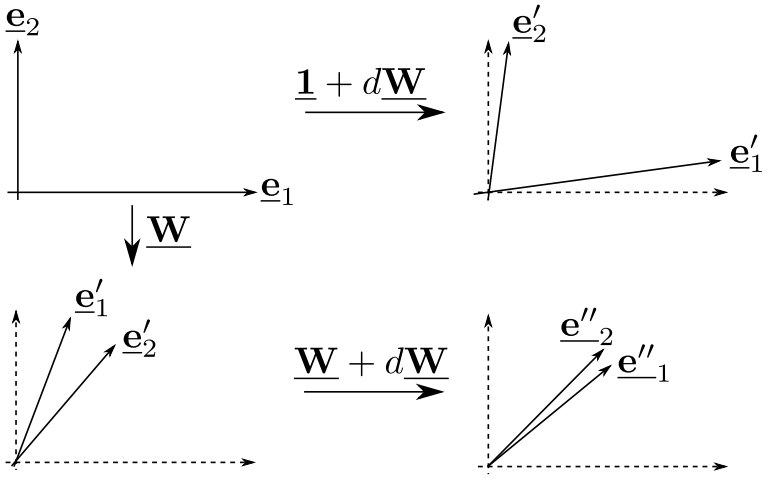


Figure 19: Illustration of gradient descent in transformed coordinate system

$$d\underline{\mathbf{Z}} = \underbrace{d\underline{\mathbf{w}}}_{\text{then do } d\underline{\mathbf{w}}} \cdot \underbrace{\underline{\mathbf{w}}^{-1}}_{\substack{\text{transfer back to } \underline{\mathbf{1}} \\ \Rightarrow \text{ makes learning} \\ \text{steps "comparable"}}} \tag{2.93}$$

This allows to do steepest ascent under normalized step size:

Taylor-expansion of e ($e^{(\alpha)}$ but α supressed in the following):

$$e_{(\underline{\mathbf{w}}+d\underline{\mathbf{w}})} = e_{(\underline{\mathbf{w}})} + \nabla e^T_{(\underline{\mathbf{w}})} dw \tag{2.94}$$

$$= e_{(\underline{\mathbf{w}})} + \underbrace{\eta}_{d\underline{\mathbf{w}}=\eta\underline{\mathbf{z}}_{\mathrm{w}}} \left[\nabla e_{(\underline{\mathbf{w}})}\right]^T \cdot \underbrace{\underline{\mathbf{z}}_{\mathrm{w}}}_{d\underline{\mathbf{w}}=\eta\underline{\mathbf{z}}_{\mathrm{w}}} \tag{2.95}$$

learning step:

$$\begin{aligned} d\underline{\mathbf{Z}} &= d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1} \\ &= \eta \underline{\mathbf{z}}_{\mathrm{w}} \cdot \underline{\mathbf{w}}^{-1} \end{aligned} \tag{2.96}$$

direction of steepest ascent under normalized step-size:

$$\left[\nabla e_{(\underline{\mathbf{w}})}\right]^T \underline{\mathbf{z}}_{\mathrm{w}} \stackrel{!}{=} \max \tag{2.97}$$

$$\left(\underline{\mathbf{z}}_{\mathrm{w}} \cdot \underline{\mathbf{w}}^{-1}\right)^2 \stackrel{!}{=} 1 \tag{2.98}$$

The solution for z_{ij} can be found using Lagrange multipliers:

$$\sum_{i,j=1}^{N} \frac{\partial e}{\partial \mathrm{w}_{ij}} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ij} - \lambda \sum_{i,j,k,l=1}^{N} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ij} \left(\underline{\mathbf{w}}^{-1}\right)_{jl} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ik} \left(\underline{\mathbf{w}}^{-1}\right)_{kl} \stackrel{!}{=} \max \tag{2.99}$$

taking the derivative wrt. the $z_{\gamma,s}$ and setting to zero yields

$$\frac{\partial e}{\partial w_{\gamma s}} - 2\lambda \sum_{k=1}^{N} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{\gamma k} \sum_{l=1}^{N} \left(\underline{\mathbf{w}}^{-1}\right)_{kl} \left(\underline{\mathbf{w}}^{-1}\right)_{ls} \stackrel{!}{=} 0 \tag{2.100}$$

$$\frac{\partial e}{\partial \underline{\mathbf{w}}} = 2\lambda \underline{\mathbf{z}}_{\mathrm{w}} \underline{\mathbf{w}}^{-1} \left(\underline{\mathbf{w}}^{-1}\right)^T \tag{2.101}$$

$$\underline{\mathbf{z}}_{\mathrm{w}} = \frac{1}{2\lambda} \frac{\partial e}{\partial \underline{\mathbf{w}}} \underline{\mathbf{w}}^T \underline{\mathbf{w}} \tag{2.102}$$

yielding the direction for "natural" gradient ascent

$$\Delta \underline{\mathbf{w}} = \eta \frac{\overbrace{\partial e}^{\substack{\text{"original"} \\ \text{gradient}}}}{\partial \underline{\mathbf{w}}} \underbrace{\underline{\mathbf{w}}^T \underline{\mathbf{w}}}_{\substack{\text{normalization} \\ \text{of step size}}} \tag{2.103}$$

$$\boxed{\Delta \mathrm{w}_{ij} = \eta \sum_{l=1}^{N} \left\{ \delta_{il} + \frac{\widehat{f}_i''\left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k^{(\alpha)}\right)} \sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k^{(\alpha)} \right\} \mathrm{w}_{lj}} \tag{2.104}$$

- efficient & fast learning rule (no matrix inversions necessary!)
- this normalization of stepsize is equivalent to imposing a Riemannian metric (with metric tensor $\underline{\mathbf{G}} = \underline{\mathbf{w}}^{-1} \cdot \underline{\mathbf{w}}^T$) on space of $\underline{\mathbf{w}}$'s

2.2.5 Some Practical Aspects

(1) undetermined source amplitudes $\rightsquigarrow$ convergence problems

Bell-Sejnowski solution:

$$\Delta \mathrm{w}_{ii} = 0 \text{ and } \mathrm{w}_{ii} = 1 \text{ for all } i \tag{2.105}$$

Amari solution: Learning steps always orthogonal to subspace of equivalent unmixing matrices.

$$\Delta \mathrm{w}_{ij} = \eta \frac{\widehat{f}_i'\left(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i\left(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)} \sum_{l\neq i}^{N} \left(\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right) \mathrm{w}_{lj} \tag{2.106}$$

(2) choice of $\widehat{f}_i$: true distribution is typically unknown

Idea: probability density with one maximum is very likely $\rightsquigarrow$ cdf will be roughly sigmoidal

typical choice:

$$\widehat{f}_{(y)} = \frac{1}{1+\exp(-y)} \qquad \text{(logistic function)}$$

$$\frac{\widehat{f}_{(y)}''}{\widehat{f}_{(y)}'} = 1 - 2\widehat{f}_{(y)} \tag{2.107}$$

Observation: ICA is fairly robust against false choice of $\widehat{f}$.

Ansatz: Ignoring scaling of $\hat{s}_i$, i.e. using (2.104): for stationary state of natural gradient ascent (batch) the following has to hold:

$$\Delta \underline{\mathbf{w}}_{ij} \stackrel{!}{=} 0 \tag{2.108}$$

$$\delta_{il} \stackrel{!}{=} -\frac{1}{p}\sum_{\alpha=1}^{p} \underbrace{\frac{\widehat{f}_i''\left(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i\left(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}}_{\varphi_i\left(\widehat{s}_i^{(\alpha)}\right)} \cdot \underbrace{\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}}_{\widehat{s}_l^{(\alpha)}} \tag{2.109}$$

Ansatz: $\widehat{s}_i = \lambda_i s_i$ estimated $\sim$ true source signal

$i = l$:

$$-\frac{1}{p}\sum_{\alpha=1}^{p} \varphi_i\left(\widehat{s}_i^{(\alpha)}\right)\lambda_i s_i^{(\alpha)} \stackrel{!}{=} 1 \tag{2.110}$$

As λ is a free parameter, this condition can always be fulfilled through proper choice of λ_i.

For $i \neq l$ and in the limit of large number of observations:

$$\delta_{il} \frac{1}{p} \sum_{\alpha=1}^{p} \varphi_i\Big(\widehat{s}_i^{(\alpha)}\Big) \lambda_l s_l^{(\alpha)} \to \Big\langle \varphi_i\Big(\widehat{s}_i^{(\alpha)}\Big) \lambda_l s_l^{(\alpha)} \Big\rangle_{P_{\underline{\mathbf{s}}}} \tag{2.111}$$

$$\Big\langle \varphi_i\big(\lambda_i s_i\big) \lambda_l s_l \Big\rangle \underbrace{=}_{\substack{\text{statistical} \\ \text{independence}}} \Big\langle \varphi_i\big(\lambda_i s_i\big) \Big\rangle \Big\langle \lambda_l s_l \Big\rangle \stackrel{!}{=} 0 \tag{2.112}$$

Can always be fulfilled if data is "centered": $< s_l >= 0$

Therefore true (independent) source signals are always a fixed point of the natural gradient ascent $\rightarrow$ independent of choice of $\widehat{f}_i$

$\Rightarrow$ however: if $\widehat{f}_i$ deviates too strongly from its true shape, the fixed point may become unstable

$\Rightarrow$ if in doubt (and enough data available)

- $\rightsquigarrow$ make a parametrized ansatz for $\widehat{f}_i$
- $\rightsquigarrow$ estimate parameters in addition to $\underline{\mathbf{w}}$

2.3 Second Order Blind Source Separation

Compared to the estimation of correlations (second order moments), the estimation of higher order moments from (small) samples of noisy observations is often unreliable. In such cases, using additional knowledge about the source signals can be exploited to extend decorrelation methods and separate sources more robustly.

The following two methods (FFDIAG, QDIAG) exploit the assumption of finite length *autocorrelations* to implement noise robust unmixing. Typical examples of such non-iid data displaying temporal correlation structure include or time series such as videos, EEG, fMRI, MEG data.

2nd order "separation" idea: find an unmixing matrix $\underline{\mathbf{w}}$, such that all cross-correlation functions vanish[5]

$$\text{statistical independence} \overset{?}{\underset{\Rightarrow}{\leftarrow}} \text{all cross-correlations vanish}$$

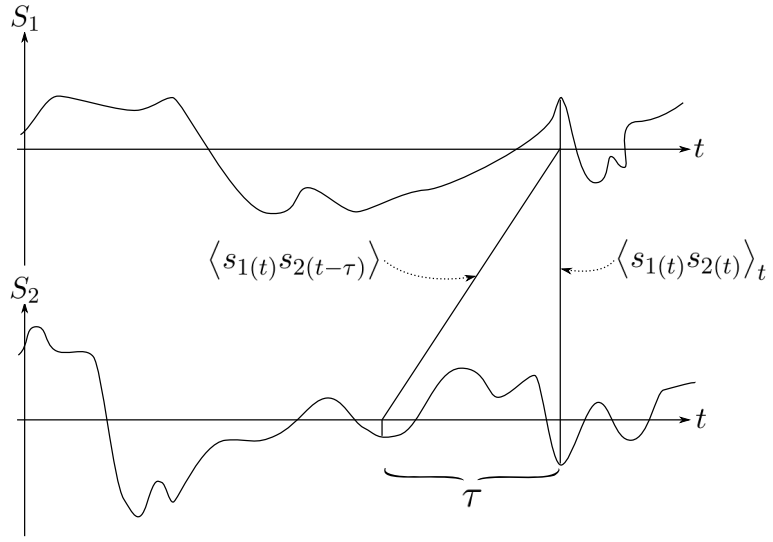


Figure 20: Source separation for time series data from two sources

Example: Source separation using two shifts
observations: $\underline{\mathbf{x}}_{(t)}$, recorded at different times (from $\underline{\mathbf{x}} = \underline{\mathbf{As}}$)

(1) PCA and sphering

[5]All the presented approaches assume a *linear* mixing model, i.e. $\underline{\mathbf{x}} \approx \underline{\mathbf{As}}$. In many scenarios, this might not be a good model. However, for small amplitudes of variation around an operating point $\underline{\mathbf{s}}_0$ this can still be used as a reasonable approximation for the more general relation $\underline{\mathbf{x}} = f(\underline{\mathbf{s}})$ as $\underline{\mathbf{x}} = f(\underline{\mathbf{s}}_1) = f(\underline{\mathbf{s_0} + \mathbf{\Delta s}}) \approx f(\underline{\mathbf{s_0}}) + \nabla f(\underline{\mathbf{s_0}}) \Delta s$.

$\rightsquigarrow$ "centering" of the data: $<\underline{\mathbf{x}}>=\underline{\mathbf{0}}$
usefull preprocessing for all ICA methods (often including dimension reduction)

$\rightsquigarrow$ solve the eigenvalue problem

$$\underline{\mathbf{C}}_{\mathrm{x}}^{(0)}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \text{ with } \big[\underline{\mathbf{C}}_{\mathrm{x}}^{(\overbrace{0}^{\tau=0})}\big]_{ij} = \Big\langle \mathrm{x}_{i(t)}\mathrm{x}_{j(t)} \Big\rangle_t \tag{2.113}$$

$\rightsquigarrow$ transformation into the eigenbasis and sphering

$$\underline{\mathbf{M}}_0 = \underbrace{\underline{\mathbf{\Lambda}}_0^{-1}}_{\substack{\text{diagonal matrix of}\\ \text{inverse eigenvalues}}} \cdot \underbrace{\underline{\mathbf{E}}_0}_{\substack{\text{matrix of}\\ \text{eigenvectors}}} \tag{2.114}$$

$$\underline{\mathbf{u}} = \underline{\mathbf{M}}_0\underline{\mathbf{x}} \tag{2.115}$$

(2) source separation requires one additional orthogonal transformation

$\rightsquigarrow$ Ansatz: $\underbrace{\underline{\mathbf{s}}}_{\substack{\text{true sources}\\ \text{(independent)}}} = \underline{\mathbf{B}}\underline{\mathbf{u}}$

$\rightsquigarrow$ for statistically independent sources we obtain

$$\begin{aligned} \Big\langle s_{i(t)}s_{j(t)} \Big\rangle_t &\underbrace{=}_{\substack{\text{statistical}\\ \text{independence}}} \delta_{ij} \\ &= \sum_{k,l=1}^{N} B_{ik} \underbrace{\Big\langle u_{k(t)}u_{l(t)} \Big\rangle_t}_{\substack{\overset{!}{=}\delta_{kl}\\ (cf.\ sphering)}} B_{lj}^T \\ &= \sum_{k=1}^{N} B_{ik}B_{kj}^T \end{aligned} \tag{2.116}$$

$$\underline{\mathbf{B}} \cdot \underline{\mathbf{B}}^T = \underline{\mathbf{1}} \rightsquigarrow \text{orthogonal transformation} \tag{2.117}$$

(3) determination of $\underline{\mathbf{B}}$ through diagonalization of a time-shifted cross-correlation matrix

$\rightsquigarrow$ solve the eigenvalue problem

$$\underline{\mathbf{C}}_u^{(\tau)}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \text{ with } \Big[\underline{\mathbf{C}}_u^{(\tau)}\Big]_{ij} = \Big\langle u_{i(t)}u_{j(t-\tau)} \Big\rangle_t \tag{2.118}$$

$\rightsquigarrow$ transformation into the eigenbasis

$$\widehat{\underline{\mathbf{s}}} = \underbrace{\underline{\mathbf{E}}_\tau}_{\substack{\text{matrix of}\\ \text{eigenvectors}}} \underline{\mathbf{u}} \tag{2.119}$$

Note: The matrix of Eigenvectors is orthonormal ($\underline{\mathbf{E}}_\tau^T \underline{\mathbf{E}}_\tau = \underline{\mathbf{I}}$) and therefore a candidate for $\underline{\mathbf{B}}$. Combining equations (2.114) and (2.119), this gives the transformation

$$\widehat{\underline{\mathbf{s}}} = \underline{\mathbf{E}}_\tau \underline{\mathbf{\Lambda}}_0^{-1/2} \underline{\mathbf{E}}_0^T \underline{\mathbf{x}} \tag{2.120}$$

Noise robust algorithms in the general case: Given a set of T zero-mean observations: $\underline{\mathbf{x}}_{(t)}$ and corresponding number of NxN covariance matrices indexed by τ

$$\left[\underline{\mathbf{C}}_{\mathrm{x}}^{(\tau)}\right]_{ij} = \frac{1}{T}\sum_{t=0}^{T-1} \mathrm{x}_{i(t)}\mathrm{x}_{j(t-\tau)} \tag{2.121}$$

- joint diagonalization of multiple cross-correlation matrices
- real world data: noise, approximate independence only $\rightsquigarrow$ only approximate diagonalization possible
- disturbances due to sensor noise can be minimized by ommitting $\tau = 0$

The following two algorithms implement these ideas using slightly different cost-functions. Both of them are implemented in the `jointDiag` package, available from CRAN.[6]

In order to find an $N \times N$ matrix $\underline{\mathbf{W}}$ that diagonalizes the set of the matrices $\underline{\mathbf{C}}^{(\tau)}, \tau = 0, 1, ...$ in a least squares sense, we consider a cost function given by the squared sum of all off-diagonal elements of $\underline{\mathbf{W}}\underline{\mathbf{C}}^{(\tau)}\underline{\mathbf{W}}^T$.

(1) QDIAG-algorithm

cost function: squared sum of non-diagonal elements:

$$E_{[\underline{\mathbf{W}}]}^T = \sum_\tau \underbrace{\alpha_\tau}_{\substack{\text{weighting}\\ \text{factors}\\ \text{(optional)}}} \sum_{i\neq j} \left(\underline{\mathbf{W}}\underline{\mathbf{C}}_{\mathrm{x}}^{(\tau)}\underline{\mathbf{W}}^T\right)_{ij}^2 \tag{2.122}$$

optimization problem:

$$\begin{aligned} E_{[\underline{\mathbf{W}}]}^T &\stackrel{!}{=} \min && \text{minimize cross-correlations} \\ \left(\underline{\mathbf{W}}\underline{\mathbf{C}}_{\mathrm{x}}^{(0)}\underline{\mathbf{W}}^T\right)_{ii} &= 1 \text{ for all } i && \text{avoid trivial solutions} \end{aligned} \tag{2.123}$$

Comments:

- *documentation:* **VollgrafObermayer2006**, Matlab-code implementing the algorithm can be found on the NI-website[7]

[6] `www.cran.r-project.org/web/packages/jointDiag/index.html`

[7] `www.ni.tu-berlin.de/menue/software/approximate_simultaneous_matrix_diagonalization_qdiag/`

- *computational complexity:* two versions with complexity $O(k \cdot N^3)$ or $O(N^5)$
- allows for arbitrary (rectangular) matrices $\underline{\mathbf{W}}$

(2) FFDIAG-algorithm

cost function: equally weighted

$$E^T_{[\underline{\mathbf{W}}]} = \sum_{\tau} \sum_{i \neq j} \left(\underline{\mathbf{W}} \underline{\mathbf{C}}^{(\tau)}_{\mathrm{x}} \underline{\mathbf{W}}^T \right)^2_{ij} \tag{2.124}$$

optimization problem:

$$\begin{aligned} E^T_{[\underline{\mathbf{W}}]} &\stackrel{!}{=} \min && \text{minimize cross-correlation} \\ & && \\ \text{invertability of } \underline{\mathbf{W}} & && \text{to avoid trivial solutions} \end{aligned} \tag{2.125}$$

Comments:

- *Documentation:* **ZieheEtAl2004**
- computational complexity: $O(K \cdot N^2)$ (approaching $O(N^3)$ for large N) requires square matrices $\underline{\mathbf{W}}$, no weighting

Preference for one or the other algorithm depends on problem size and kind of data (cf. **StetterEtAl2000**), □ Ex: optical imaging

2.4 Fast ICA

ICA and Projection Pursuit "interesting" complex features

⇝ variance criterion ⇒ PCA
important preprocessing method, but: scale sensitive solutions

⇝ higher moments: ok, but: Why not maximize non-Gaussianity?

ICA and non-Gaussianity

$$\underline{\mathbf{x}} = \underline{\mathbf{A}}\underline{\mathbf{s}} \qquad \text{mixing matrix } \underline{\mathbf{A}}\text{, statistically independent sources}$$

$$\widehat{s}_i = \underline{\mathbf{w}}_i^T \underline{\mathbf{x}} \qquad \text{extraction of one source through one row vector of an unmixing matrix}$$

$$\widehat{s}_i = \underbrace{\underline{\mathbf{w}}_i^T \underline{\mathbf{A}}}_{\underline{\mathbf{z}}_i^T} \underline{\mathbf{s}} = \underline{\mathbf{z}}_i^T \underline{\mathbf{s}} \qquad \text{linear combination of statistically independent variables}$$

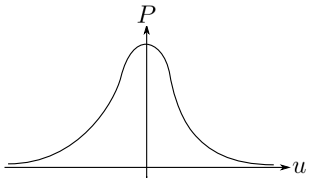
"The sum of independent random variables is 'more Gaussian' then the original variables"

⇒ interesting directions (projection pursuit) and individual components (ICA) can be extracted by maximizing non-Gaussianity w.r.t $\underline{\mathbf{w}}$

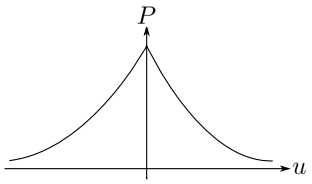
Measures for non-Gaussianity: The Gaussian is fully characterized by its first and second moments. Deviations from Gaussianity can therefore be quantified via the following measures:

(1) Kurtosis

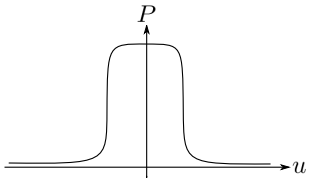
$$\mathrm{kurt}(u) = \left\langle u^4 \right\rangle_{P_u(u)} - 3 \underbrace{\left(\left\langle u^2 \right\rangle_{P_u(u)} \right)^2}_{\substack{\text{would be 1 for} \\ \text{sphered data}}} \tag{2.126}$$



$\mathrm{kurt}(u) = 0$ Gaussian PDF



$\mathrm{kurt}(u) > 0$ "super"-Gaussian PDF
$\rightarrow$ peaky, long tails (i.e. "outliers")
$\rightarrow$ e.g.: Laplace distribution



$\mathrm{kurt}(u) < 0$ "sub"-Gaussian PDF
$\rightarrow$ bulky, no "outliers"
$\rightarrow$ e.g. constant distribution

for independent random variables u_1 and u_2 we get:

$$\begin{aligned} \mathrm{kurt}(u_1 + u_2) &= \mathrm{kurt}(u_1) + \mathrm{kurt}(u_2) \\ \mathrm{kurt}(z_1 u_1) &= z_1^4 \, \mathrm{kurt}(u_1) \end{aligned} \tag{2.127}$$

Example: two statistically independent sources with $\left\langle s_i s_j \right\rangle = \delta_{ij}$

$$\begin{aligned} \widehat{s} &= \underline{\mathbf{z}}^T \underline{\mathbf{s}} \\ &= z_1 s_1 + z_2 s_2 \end{aligned} \tag{2.128}$$

$$\begin{aligned}\operatorname{var}(\widehat{s}) &= \Big\langle \big(z_1 s_1 + z_2 s_2\big)^2 \Big\rangle_{P_s(s)} \\ &= z_1^2 \langle s_1^2 \rangle + z_2^2 \langle s_2^2 \rangle \\ &= z_1^2 + z_2^2\end{aligned} \tag{2.129}$$

$$\operatorname{kurt}(\widehat{s}) = z_1^4 \operatorname{kurt}(s_1) + z_2^4 \operatorname{kurt}(s_2) \tag{2.130}$$

search for "interesting" directions

$$\begin{aligned}\operatorname{kurt}(\widehat{s}) &\stackrel{!}{=} \max_{\underline{\mathbf{z}}} \quad \leftarrow \text{ search for the direction of optimal kurtosis} \\ z_1^2 + z_2^2 &\stackrel{!}{=} 1 \quad \leftarrow \text{ such that data remained sphered}\end{aligned} \tag{2.131}$$

Result: (*see supplementary material*)

$$\underline{\mathbf{z}} = \begin{pmatrix} 0 \\ \pm 1 \end{pmatrix} \text{ or } \underline{\mathbf{z}} = \begin{pmatrix} \pm 1 \\ 0 \end{pmatrix} \tag{2.132}$$

independent sources correspond to extrema of the kurtosis

(2) Negentropy

$$J_{(u)} := \underbrace{H_{(u)}^{\text{Gauss}}}_{\substack{\text{entropy of Gaussian} \\ \text{with variance } \sigma^2}} - \underbrace{H_{(u)}}_{\substack{\text{entropy of true} \\ \text{distribution} \\ \text{(variance } \sigma^2\text{)}}} \tag{2.133}$$

- interesting, theoretically well founded measure
- but: hard to evaluate, optimization is computationally expensive (depends on full distribution)

(3) Approximations to the negentropy

$$J_{(u)} \approx \sum_{i=1}^{l} \underbrace{k_i}_{\substack{\text{some} \\ \text{constant}}} \left\{ \Big\langle G_{(u)} \Big\rangle_{\underbrace{P_u(u)}_{\substack{\text{true} \\ \text{density}}}} - \Big\langle G_{(u)} \Big\rangle_{\underbrace{\text{Gauss}}_{\substack{\text{reference:} \\ \text{Gaussian} \\ \text{density with} \\ \text{some variance}}}} \right\} \tag{2.134}$$

common contrast functions:

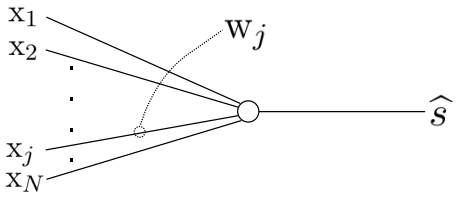
$$G_{1(u)} = \tfrac{1}{a} \log \cosh au \qquad \text{good general purpose function}$$

$$G_{2(u)} = -\exp\Big(-\tfrac{u^2}{2}\Big) \qquad \begin{matrix}\text{good only if sources are} \\ \text{highly "super"-Gaussian} \\ \text{i.e. many outliers}\end{matrix}$$

$$G_{3(u)} = \tfrac{1}{4} u^4 \qquad \begin{matrix}\text{kurtosis (see ①),} \\ \text{useful if components} \\ \text{are "sub"-Gaussian} \\ \text{i.e. few outliers}\end{matrix}$$

$\rightsquigarrow$ Clever choice of G allows to obtain robust approximations to negentropy **Hyvaerinen1997**, **HyvaerinenOja2000**.

Fixed point algorithm for one linear neuron: The following algorithm (alg. **??**) implements `fastICA` for standardized data (mean=0, sd=1) to learn a single weight vector $\underline{\mathbf{w}}$:



$$\hat{s} = \underbrace{\underline{\mathbf{w}}^T}_{\substack{\text{"interesting"} \\ \text{complex} \\ \text{features}}} \underline{\mathbf{x}}$$

Algorithm 3: fixed-point algorithm for fastICA: single component

randomly initialize weight vector $\underline{\mathbf{w}}$ of unit length

repeat

$$\underline{\mathbf{w}}^+ = \frac{1}{p}\left\{ \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} G'\left(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(\alpha)}\right) - \underline{\mathbf{w}} \sum_{\alpha=1}^{p} G''\left(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(\alpha)}\right) \right\}$$

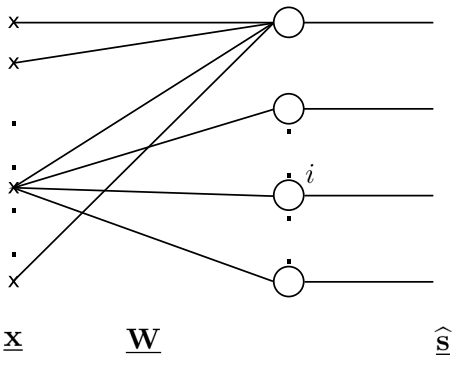
$$\underline{\mathbf{w}} = \frac{\underline{\mathbf{w}}^+}{\|\underline{\mathbf{w}}^+\|}$$

until *convergence*

- for details, see **Hyvaerinen1999** and **HyvaerinenEtAl2001**.
- convergence to direction of extremal "non-Gaussianity"

$\Rightarrow$ example of a so-called projection pursuit method

Fixed-point algorithm for a perceptron



Algorithm 4 implements fastICA for standardized data (mean=0, sd=1) to learn the $N \times N$ weight matrix $\underline{\mathbf{W}}$.

Algorithm 4: fixed-point algorithm for fastICA: multiple components

$t \leftarrow 0$

randomly initialize weight vectors $\underline{\mathbf{w}}_i^{(t)}$ for $i \in 1 \dots N$

repeat

symmetric orthogonalization:

$$\underline{\mathbf{W}}^{(t)} \leftarrow \frac{3}{2}\underline{\mathbf{W}}^{(t)} - \frac{1}{2}\underline{\mathbf{W}}^{(t)}\left(\underline{\mathbf{W}}^{(t)}\right)^T\underline{\mathbf{W}}^{(t)}$$

for $i \in 1 \dots N$ **do**

$$\underline{\mathbf{w}}_i^{(t)} \leftarrow \frac{1}{p}\left\{\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)}G'\left(\left(\underline{\mathbf{w}}_i^{(t)}\right)^T\underline{\mathbf{x}}^{(\alpha)}\right) - \underline{\mathbf{w}}_i^{(t)}\sum_{\alpha=1}^{p}G''\left(\left(\underline{\mathbf{w}}_i^{(t)}\right)^T\underline{\mathbf{x}}^{(\alpha)}\right)\right\}$$

$$\underline{\mathbf{w}}_i^{(t+1)} \leftarrow \frac{\underline{\mathbf{w}}_i^{(t)}}{\|\underline{\mathbf{w}}_i^{(t)}\|}$$

end

$t \leftarrow t + 1$

until *convergence*

Remark: The fastICA algorithm can be understood as a fixed point algorithm for maximum likelihood estimation of the ICA-model in which the learning rate for the different directions is adaptively adjusted (see **HyvaerinenOja2000**). For further details, see **HyvaerinenEtAl2001**.

code: `http://research.ics.aalto.fi/ica`

☐ sound demo: blind source separation

☐ natural images: van Hateren and van der Schaaf

3 Stochastic Optimization

Most supervised and unsupervised learning problems involve evaluation of a cost function E^T. For cost functions with real-valued arguments, gradient based techniques allow to find (locally) optimal solutions.

This chapter deals with methods to solve problems based on cost-functions with *discrete arguments* (e.g. cluster assignment) where gradient-based techniques are not directly applicable.

3.1 Simulated Annealing

Simulated annealing is a method for stochastic optimization and based on an analogy to "natural" optimization. The optimisation algorithm mimicks freezing or crystallization of a physical system during which (not necessarily global) optima regarding the energy of the system are reached. The process involves *slow cooling* (glass vs. crystal $\Rightarrow$ annealing) via a *computational temperature* T or "noise parameter" $\beta = \frac{1}{T}$.

Given a set of *discrete* variables: $\{s_i\}, i = 1, \ldots, N$ (with $s_i \in \mathcal{S}$) describing the *state*[8] of the system and a real-valued *cost function*:

$$E : \underline{\mathbf{s}} \mapsto E_{(\underline{\mathbf{s}})} \in \mathbb{R} \tag{3.1}$$

the *goal* is to find the (globally) optimal state $\underline{\mathbf{s}}^*$, such that

$$E \stackrel{!}{=} \min \tag{3.2}$$

"*Stochastic* simulated annealing" (algorithm **??**) implements an iterative procedure to find this optimal state.

Algorithm 5: Stochastic simulated annealing

initialization: $\underline{\mathbf{s}}_0, \tau, M, \beta_0$ small ($\rightsquigarrow$ high T)
begin Annealing loop: $t = 1, 2, \ldots$
  $\underline{\mathbf{s}}_t = \underline{\mathbf{s}}_{t-1}$ (initialization of inner loop)
  begin State Update loop: M iterations
    choose a new state $\underline{\mathbf{s}}$ randomly (local to $\underline{\mathbf{s}}_t$ – e.g. bit flip)
    calculate difference in cost: $\Delta E = E_{(\underline{\mathbf{s}})} - E_{(\underline{\mathbf{s}}_t)}$
    switch $\underline{\mathbf{s}}_t$ to $\underline{\mathbf{s}}$ with probability $\mathrm{W}_{(\underline{\mathbf{s}}_t \to \underline{\mathbf{s}})} = \frac{1}{1+\exp(\beta_t \Delta E)}$
    (otherwise keep the old state $\underline{\mathbf{s}}_t$)
  end
  $\beta_t = \tau \beta_{t-1}$ (practical but theoretically not optimal)
end

[8] we will use short-hand notation: $\underline{\mathbf{s}}$ ("state", but not necessarily a vector space)

Remark: $\underline{\mathbf{s}}$ is often chosen "similar" or "close" to the old state, e.g. by randomly "flipping" one variable rather than sampling a completely random new state.

The dynamics of such a switching process are affected by the transition probability function W and its dependence on the temperature parameter β:

- transition probability $\mathrm{W}_{(\underline{\mathbf{s}}_t \to \underline{\mathbf{s}})}$

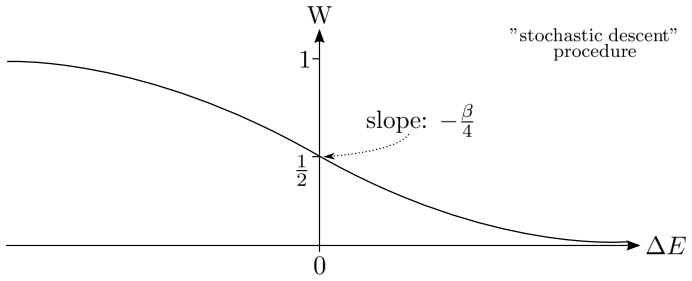


Figure 21: Transition probabilities

- limiting cases

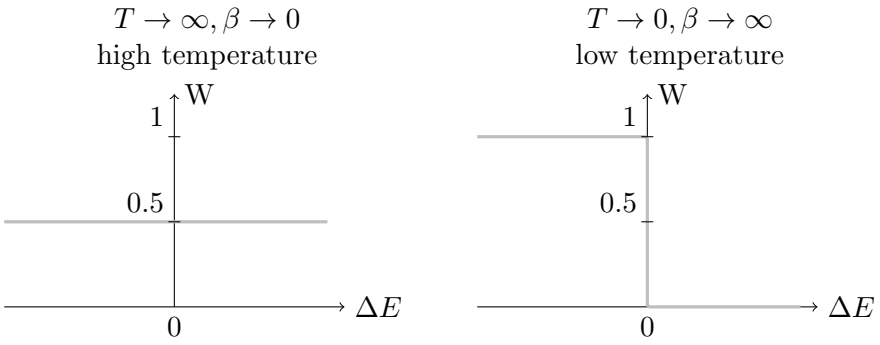


Figure 22: Transition probabilities in the two limiting cases

For many optimization problems, the cost function has local optima (see figure 23). In such cases, convergence to the global optimum of the cost function is guaranteed if:

$$\beta_t \sim \ln t \tag{3.3}$$

$\Rightarrow$ robust optimization procedure

$\Rightarrow$ but: $\beta_t \sim \ln t$ is too slow for practical problems

$\Rightarrow$ therefore: $\beta_{t+1} = \tau\beta_t, \tau = 1.01 \ldots 1.30$ (exponential annealing)

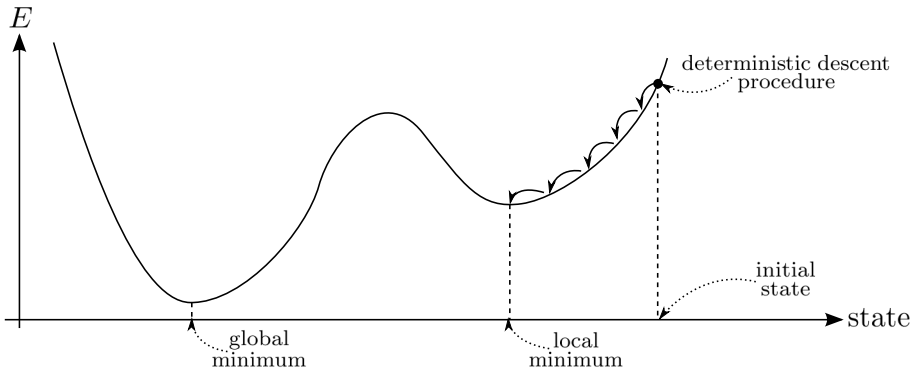
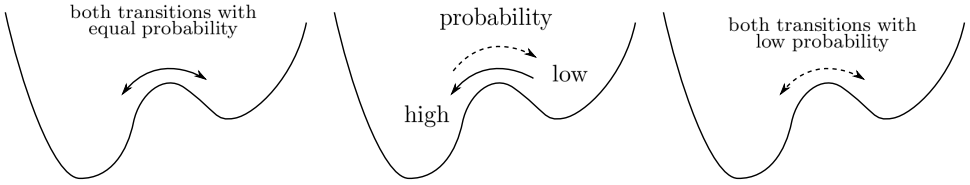


Figure 23: Cost function with local minima



high T, low β

intermediate values for T and β $\Rightarrow$ escape from local optima is possible

low T, high β

Figure 24: Effect of annealing on local minima

$\Rightarrow$ additionally: the `State Update loop` has to be iterated often enough, e.g. $M = 2000$ ($\rightsquigarrow$ thermal equilibrium), however also $M = 1$ can work if the temperature is decreased slowly enough

3.2 The Gibbs Distribution

The random (noisy) state changes cause state fluctuations even for constant T and β (more precisely we have a stochastic process $\underline{\mathbf{s}}_{t'}$ with the Markov property). Under certain conditions, these fluctuations result in a *stationary distribution* over all possible states, where the probability of observing a specific state depends on its energy (cost).

$\Pi_{(\underline{\mathbf{s}},t')}$: probability distribution across states where t' corresponds to the iteration count of the `State Update loop` of algorithm **??**.

$$\Pi_{(\underline{\mathbf{s}},t')} \rightarrow \underbrace{P_{(\underline{\mathbf{s}})}}_{\substack{\text{stationary}\\ \text{distribution}}} \quad \text{for } t' \rightarrow \infty \text{ (and constant } T, \beta) \tag{3.4}$$

calculation of $P_{(\underline{\mathbf{s}})}$ under the assumption of "detailed balance" (reversible Markov process):

$$\underbrace{\underbrace{\text{probability of transition } \underline{\mathbf{s}} \rightarrow \underline{\mathbf{s}}'}_{P_{(\underline{\mathbf{s}})} \mathrm{W}_{(\underline{\mathbf{s}} \rightarrow \underline{\mathbf{s}}')}}}_{\substack{=\\ \text{detailed balance}}} \underbrace{\underbrace{\text{probability of transition } \underline{\mathbf{s}}' \rightarrow \underline{\mathbf{s}}}_{P_{(\underline{\mathbf{s}}')} \mathrm{W}_{(\underline{\mathbf{s}}' \rightarrow \underline{\mathbf{s}})}}} \tag{3.5}$$

$$\begin{aligned}
\frac{P_{(\underline{\mathbf{s}})}}{P_{(\underline{\mathbf{s}}')}} &= \frac{\mathrm{W}_{(\underline{\mathbf{s}}' \rightarrow \underline{\mathbf{s}})}}{\mathrm{W}_{(\underline{\mathbf{s}} \rightarrow \underline{\mathbf{s}}')}} \\
&= \frac{1+\exp\left\{\beta\left(E_{(\underline{\mathbf{s}})} - E_{(\underline{\mathbf{s}}')}\right)\right\}}{1+\exp\left\{\beta\left(E_{(\underline{\mathbf{s}}')} - E_{(\underline{\mathbf{s}})}\right)\right\}} \\
&= \frac{1+\exp(\beta\Delta E)}{1+\exp(-\beta\Delta E)} \\
&= \exp(\beta\Delta E)\frac{1+\exp(-\beta\Delta E)}{1+\exp(-\beta\Delta E)} \\
&= \exp(\beta\Delta E)
\end{aligned} \tag{3.6}$$

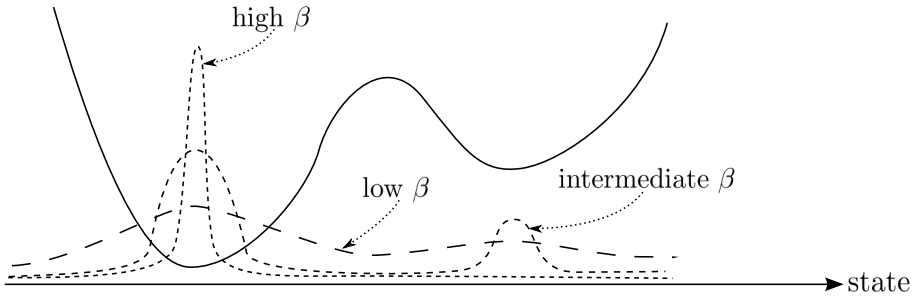
this condition is fulfilled for:

$$P_{(\underline{\mathbf{s}})} = \frac{1}{Z}\exp(-\beta E) \qquad \text{(Gibbs-Boltzmann-distribution)}$$

normalization constant / partition function (sum over all states):

$$Z = \sum_{\underline{\mathbf{s}}} \exp(-\beta E) \tag{3.7}$$

probability distribution depends on cost and temperature



$\beta \downarrow$: broad, "delocalized" distribution

$\beta \uparrow$: distribution localized around (global) minima

Figure 25: cost-dependent probability distributions

3.3 Mean-Field Annealing

Stochastic optimization can be computationally expensive: it depends on the cost function E and the cooling schedule.[9] so a possible strategy might seem to evaluate $P_{(\underline{\mathbf{s}})}$ directly ($P_{(\underline{\mathbf{s}})}$ is known! – the Gibbs-distribution). However:

- maxima of $P_{(\underline{\mathbf{s}})}$ are equally hard to obtain as minima of E
- moments of $P_{(\underline{\mathbf{s}})}$ can - in general - not be calculated analytically

Fortunately, a viable strategy is to approximate $P_{(\underline{\mathbf{s}})}$ by a computationally tractable distribution $Q_{(\underline{\mathbf{s}})}$.

Approximation: The distribution $Q_{(\underline{\mathbf{s}})}$ to approximate $P_{(\underline{\mathbf{s}})}$ is chosen as a *factorizing distribution* with costs E_Q linear in the state variable $\underline{\mathbf{s}}$:

$$Q_{(\underline{\mathbf{s}})} \quad = \quad \frac{1}{Z_Q} \exp\left\{-\beta E_Q\right\} \quad = \quad \frac{1}{Z_Q} \exp\Big\{ -\beta \sum_k \underbrace{e_k}_{\text{parameters}} \mathrm{s}_k \Big\} \tag{3.8}$$

$\rightarrow$ family of distributions parametrized by the *mean fields* e_k

$\rightarrow$ Goal: determine e_k such that this approximation is as good as possible

Calculation of moments: More generally, moments of a distribution P provide a concise description of P. For highly concentrated distributions (e.g. $\beta \rightarrow \infty$), the 1st moment (its *mean*) gives a good characterization of the distribution (e.g. if it is highly peaked also for the location of its

[9] so far, we have left E unspecified and – in many cases – it will be costly to compute

maximum). For the factorizing distribution $Q_{(\underline{\mathbf{s}})}$, these moments can be calculated easily:

$$\Big\langle f_{(\underline{\mathbf{s}}/s_l)} g_{(s_l)} \Big\rangle_Q = \frac{1}{Z_Q} \sum_{\underline{\mathbf{s}}} f_{(\underline{\mathbf{s}}/s_l)} g_{(s_l)} \exp\Big\{ -\beta \sum_k e_k s_k \Big\} \tag{3.9}$$

$$= \frac{1}{Z_Q} \Bigg[\sum_{\underline{\mathbf{s}}/s_l} f_{(\underline{\mathbf{s}}/s_l)} \exp\Big(-\beta \sum_{k \neq l} e_k s_k \Big) \Bigg] \Bigg[\sum_{s_l} g_{(s_l)} \exp\Big(-\beta e_l s_l \Big) \Bigg]$$

$$= \frac{1}{Z_Q} \Bigg[\sum_{\underline{\mathbf{s}}/s_l} f_{(\underline{\mathbf{s}}/s_l)} \exp\Big(-\beta \sum_{k \neq l} e_k s_k \Big) \Bigg] \frac{\sum\limits_{s_l} \exp(-\beta e_l s_l)}{\sum\limits_{s_l} \exp(-\beta e_l s_l)} \Bigg[\sum_{s_l} g_{(s_l)} \exp\Big(-\beta e_l s_l \Big) \Bigg]$$

$$= \Big\langle f_{(\underline{\mathbf{s}}/s_l)} \Big\rangle_Q \frac{\sum\limits_{s_l} g_{(s_l)} \exp(-\beta e_l s_l)}{\sum\limits_{s_l} \exp(-\beta e_l s_l)}$$

$$= \underbrace{\Big\langle f_{(\underline{\mathbf{s}}/s_l)} \Big\rangle_Q \cdot \Big\langle g_{(s_l)} \Big\rangle_Q}_{\substack{\text{factorization of moments} \\ \rightarrow \text{uncorrelated variables}}}$$

The first moments $\big\langle s_l \big\rangle_Q$ play a central role in the approximation of P by Q (see below) and usually have a tractable expression:

$$\big\langle s_l \big\rangle_Q = \frac{\sum\limits_{s_l \in \mathcal{S}} s_l \exp(-\beta e_l s_l)}{\sum\limits_{s_l \in \mathcal{S}} \exp(-\beta e_l s_l)} \tag{3.10}$$

The mean-field approximation:

$$\begin{aligned} P_{(\underline{\mathbf{s}})} &= \tfrac{1}{Z_p} \exp(-\beta E_p) && \text{true distribution} \\ Q_{(\underline{\mathbf{s}})} &= \tfrac{1}{Z_Q} \exp\Big(-\beta \overbrace{\sum_k e_k s_k}^{E_Q} \Big) && \text{approximation: family of factorizing distributions} \\ e_k : &\quad \textit{mean fields} && \text{parameters to be determined} \end{aligned} \tag{3.11}$$

minimization of the KL-divergence:

$$D_{KL}(Q||P) = \sum_{\underline{\mathbf{s}}} Q_{(\underline{\mathbf{s}})} \ln \frac{Q_{(\underline{\mathbf{s}})}}{P_{(\underline{\mathbf{s}})}} \stackrel{!}{=} \min_{\underline{\mathbf{e}}} \tag{3.12}$$

$$\begin{aligned}
\tfrac{\partial}{\partial e_l} D_{KL} &= \tfrac{\partial}{\partial e_l}\left\{\beta \sum_{\underline{\mathbf{s}}} Q_{(\underline{\mathbf{s}})} E_p - \beta \sum_{\underline{\mathbf{s}}} Q_{(\underline{\mathbf{s}})} E_Q + \ln Z_p - \ln Z_Q\right\} \\
&= \beta \tfrac{\partial}{\partial e_l}\big\langle E_p \big\rangle_Q \underbrace{-\beta \frac{\partial}{\partial e_l} \sum_{\underline{\mathbf{s}}} Q_{(\underline{\mathbf{s}})} \sum_k e_k s_k}_{-\beta \sum_k e_k \frac{\partial}{\partial e_l}\langle s_k\rangle_Q - \beta\langle s_l\rangle_Q} \underbrace{- \frac{1}{Z_Q} \sum_{\underline{\mathbf{s}}} \frac{\partial}{\partial e_l} \exp(-\beta \sum_k e_k s_k)}_{+\beta\langle s_l\rangle_Q} \\
&= \beta \tfrac{\partial}{\partial e_l}\big\langle E_p \big\rangle_Q - \beta \sum_k e_k \tfrac{\partial}{\partial e_l}\big\langle s_k \big\rangle_Q \quad \stackrel{!}{=} \quad 0
\end{aligned} \tag{3.13}$$

The solution to this equation determines the mean fields e_k minimzing D_{KL} and depends on the exact form of E_p. It can be found by solving the equations:

$$\boxed{\tfrac{\partial}{\partial e_l}\big\langle E_p \big\rangle_Q - \sum_k e_k \tfrac{\partial}{\partial e_l}\big\langle s_k \big\rangle_Q = 0} \tag{3.14}$$

The latter simplifies further to

$$\boxed{\tfrac{\partial}{\partial e_l}\big\langle E_p \big\rangle_Q - e_l \tfrac{\partial}{\partial e_l}\big\langle s_l \big\rangle_Q = 0} \tag{3.15}$$

because of the independent $\{s_k\}$ under the factorizing distribution Q.

Algorithm 6: Mean Field Annealing

initialization: $\langle \underline{\mathbf{s}} \rangle_0, \beta_0, t = 0$
begin Annealing loop
  repeat
    calculate mean-fields: $e_k, \quad k = 1, \ldots, N$
    calculate moments: $\big\langle s_k \big\rangle_Q, \quad k = 1, \ldots, N$
  until $|e_k^{\text{old}} - e_k^{\text{new}}| < \varepsilon$
  increase β
end

$\Rightarrow$ inner loop: fixed-point iteration for the mean-fields e_k

$\Rightarrow$ the moments $\big\langle s_k \big\rangle$ for not too small temperatures are in general not from the discrete set $\mathcal{S}$ but range continuously between the elements of $\mathcal{S}$

$\Rightarrow$ however: $\beta \to \infty (T \to 0) : \big\langle s_k \big\rangle \to s_k^*$
because $P_{(\underline{\mathbf{s}})}$ becomes singular at the state $\underline{\mathbf{s}}^*$ of minimal cost!

$\Rightarrow$ deterministic (fast) rather than stochastic (slow) optimization method (given that mean-field equations can be easily evaluated)

see **BilbroEtAl1989** and **Rose1998** for details.

Example (based on Ising model): Consider the quadratic cost function $E(s_1, \ldots, s_N)$ for binary state vectors, i.e., $s_k \in \mathcal{S} = \{+1, -1\}$,

$$E_p(\underline{\mathbf{s}}) = -\frac{1}{2} \sum_{\substack{i=1,j=1 \\ i \neq j}}^{N} W_{ij} s_i s_j, \tag{3.16}$$

with a real symmetric matrix $\underline{\mathbf{W}}$ with zeros on the diagonal ($\rightarrow$ no self-coupling).

The expressions required for the algorithm **??** can be directly calculated:

1. The first moment, cf. eq. (3.10), become:

$$\begin{aligned} \langle s_k \rangle_Q &= \frac{\sum_{s_k \in \mathcal{S}} s_k \exp(-\beta e_k s_k)}{\sum_{s_k \in \mathcal{S}} \exp(-\beta e_k s_k)} \\ &= \frac{+1 \exp(-\beta e_k) - 1 \exp(\beta e_k)}{\exp(-\beta e_k) + \exp(\beta e_k)} \\ &= \tanh(-\beta e_k) \end{aligned} \tag{3.17}$$

2. The mean-fields, cf. eq. (3.15), are given by

$$\begin{aligned} 0 &= \frac{\partial}{\partial e_k} \langle E_p \rangle_Q - e_k \frac{\partial}{\partial e_k} \langle s_k \rangle_Q \\ &= \frac{\partial}{\partial e_k} \left\langle -\frac{1}{2} \sum_{\substack{i=1,j=1 \\ i \neq j}}^{N} W_{ij} s_i s_j \right\rangle_Q - e_k \frac{\partial}{\partial e_k} \langle s_k \rangle_Q \\ &= -\frac{1}{2} \frac{\partial}{\partial e_k} \sum_{\substack{i=1,j=1 \\ i \neq j}}^{N} W_{ij} \langle s_i \rangle_Q \langle s_j \rangle_Q - e_k \frac{\partial}{\partial e_k} \langle s_k \rangle_Q \\ &= -\sum_{\substack{i=1 \\ i \neq k}}^{N} W_{ik} \langle s_i \rangle_Q \frac{\partial}{\partial e_k} \langle s_k \rangle_Q - e_k \frac{\partial}{\partial e_k} \langle s_k \rangle_Q \end{aligned} \tag{3.18}$$

   which implies

$$e_k = -\sum_{\substack{i=1 \\ i \neq k}}^{N} W_{ik} \langle s_i \rangle_Q. \tag{3.19}$$

   Remark: the fixed-point iteration for the mean-fields $\underline{\mathbf{e}}$ of this example converges (locally) which can be proven using the Banach fixed point theorem (at least for sufficiently large β).

4 Clustering and Embedding

While projection methods search for interesting directions / features along which data differ, clustering methods yield groupings of the data according to a specified similarity or distance measure.

While methods of central clustering typically also find representatives (e.g. group averages or prototypes) for these different groups, *pairwise clustering* methods just need estimates of distances (e.g. similiarity judgements) between pairs of data points.

4.1 K-means Clustering

4.1.1 The Clustering Problem

Goal: partitioning of observations $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$ according to similarity. This is illustrated in figure 26.

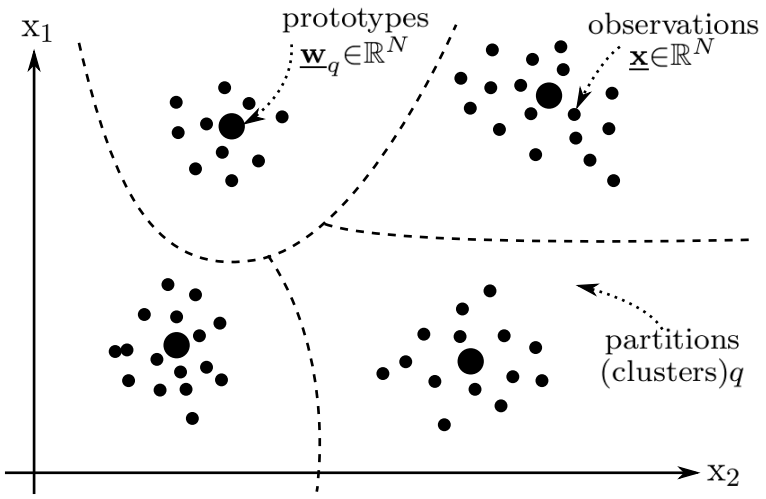


Figure 26: The clustering problem

$\Rightarrow$ unsupervised formation of categories (partitions, clusters) according to predefined criteria

$\Rightarrow$ description of clusters by prototypes $\leftarrow$ "central" clustering

K-means Clustering: "prototype-based clustering"

- based on average quadratic Euclidean distance between observations and prototypes
- simple & most common procedure for clustering of vectorial data

Cluster model:

prototypes: $\underline{\mathbf{w}}_q, q = 1, \ldots, M$ (M: number of clusters)

binary assignment variables m_q^α:

$$m_q^{(\alpha)} = \begin{cases} 1, & \text{if } \underline{\mathbf{x}}^{(\alpha)} \text{ belongs to cluster } q \\ 0, & \text{else} \end{cases} \tag{4.1}$$

normalization:

$$\sum_q m_q^{(\alpha)} = 1 \tag{4.2}$$

cost function ("empirical risk"):

$$E^T_{\left[\left\{m_q^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q\right\}\right]} = \frac{1}{2p} \sum_{q,\alpha} m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right)^2 \tag{4.3}$$

The cost function represents the average quadratic distance between observations and prototypes ("variance"). The choice of similarity/distance measure should be based on prior knowledge (if available).

Remark: If $\underline{\mathbf{w}}_q$ is center of mass $\implies E^T = \frac{1}{2} \cdot$ variance.

model selection:

$$E^T \stackrel{!}{=} \min \leftarrow \begin{matrix}\text{continous/discrete} \\ \text{optimization problem}\end{matrix} \tag{4.4}$$

$$\text{validation} \begin{cases} \text{no, if goal is "just" to describe the set of observations} \\ \text{yes, if goal is inference/prediction on future observations (e.g. calculating } m_q^{(\text{new})} \text{ for a new observation } \underline{\mathbf{x}}^{(\text{new})}) \end{cases}$$

The computations involved in the cost function for k-means clustering are illustrated in figure 27.

4.1.2 Model Selection

Optimization problem with continuous (cluster-centers) and discrete (cluster-assignment: binary) variables. Dissimilarity measure is Euclidean distance.

$\Rightarrow$ two step descent procedure (algorithm **??**)

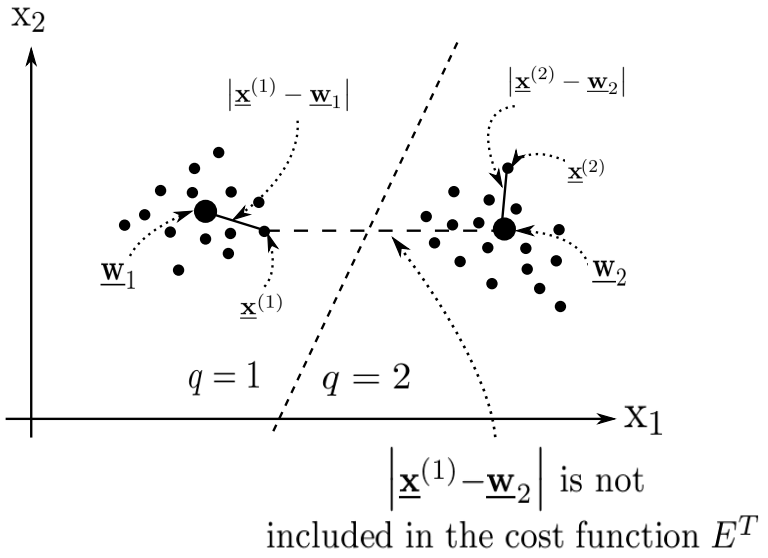


Figure 27: Illustration of the k-means cost function

Algorithm 7: batch k-means

randomly initialize prototypes e.g. around (i.e., +noise) the whole data's center of mass, e.g., using PCA for spread quantification

begin loop

(1) choose $m_q^{(\alpha)}$ ...
... such that E^T is minimal for the given prototypes

$$m_q^{(\alpha)} \begin{cases} 1, & \text{if } q = \text{argmin}_\gamma \left|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma\right| \\ 0, & \text{else} \end{cases}$$

$\Rightarrow$ assign every data point to its nearest prototype

(2) choose $\underline{\mathbf{w}}_q$ ...
... such that E^T is minimal for the -new- assignments

$$\underline{\mathbf{w}}_q = \frac{\sum\limits_\alpha m_q^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}{\sum\limits_\alpha m_q^{(\alpha)}}$$

$\Rightarrow$ set $\underline{\mathbf{w}}_q$ to the center of mass of its assigned data

end

center of mass is optimal: condition for extremal point:

$$\frac{\partial}{\partial \underline{\mathbf{w}}_q} \left\{ \frac{1}{2p} \sum_{q',\alpha} m_{q'}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{q'}\right)^2 \right\} = -\frac{1}{p} \sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right) \stackrel{!}{=} 0$$

$$\overset{64}{\rightsquigarrow} \underline{\mathbf{w}}_q = \frac{\sum\limits_\alpha m_q^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}{\sum\limits_\alpha m_q^{(\alpha)}} \tag{4.5}$$

condition for minimum:

$$\frac{\partial^2}{\partial \mathrm{w}_{qi}\partial \mathrm{w}_{q''j}}\left\{\frac{1}{2p}\sum_{q',\alpha} m_{q'}^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}-\underline{\mathbf{w}}_{q'}\right)^2\right\} = \frac{\partial}{\partial \mathrm{w}_{q''j}}\left\{-\frac{1}{p}\sum_{\alpha} m_q^{(\alpha)}\left(\mathrm{x}_i^{(\alpha)}-\underline{\mathbf{w}}_{qi}\right)\right\}$$
$$= \left(\frac{1}{p}\sum_{\alpha} m_q^{(\alpha)}\right)\delta_{ij}\delta_{qq''} \tag{4.6}$$

$\rightsquigarrow$ diagonal matrix with all positive entries

$\rightsquigarrow$ condition for minimum always fulfilled

- E^T is non-increasing in every step and E^T is bounded from below $\Rightarrow$ K-means clustering converges to a (local) optimum of E^T.
- E^T at the solution can be interpreted as the average variability within the groups.

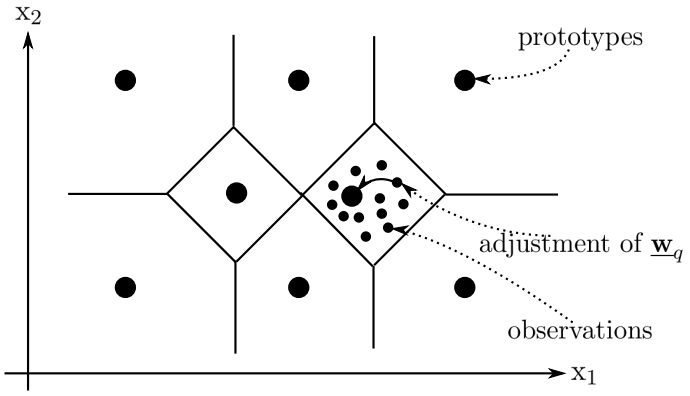


Figure 28: k-means and tesselation

tesselation cell: region of data space for which

$$q = \underset{\gamma}{\operatorname{argmin}} \left|\underline{\mathbf{x}} - \underline{\mathbf{w}}_\gamma\right| \tag{4.7}$$

This interpretation is illustrated in figure 28 and provides an alternative 2-step interpretation of k-means clustering in terms of an optimized tesselation of the input-space:

(1) construct tesselation of feature space

(2) adjust prototypes to the center of mass of all data points within the corresponding tesselation cell

"on-line" version of K-means Clustering: The k-means procedure can be implemented in an on-line fashion (algorithm **??**) that is often more robust wrt. local minima than batch-learning and can be useful for streaming-data.

Algorithm 8: on-line k-means

initialize prototypes e.g. around center of mass
select learning step $0 < \eta << 1$
begin loop
  choose a data point $\underline{\mathbf{x}}^{(\alpha)}$
  assign data point to its closest prototype q

$$q = \underset{\gamma}{\operatorname{argmin}} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma \right|$$

  change corresponding prototype according to

$$\Delta \underline{\mathbf{w}}_q = \eta \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma \right)$$

  change η
end

Goodness of the found solution depends on choosing an appropriate "annealing" schedule for η: Robbins-Monro conditions (*cf. MI I, section 1.4.1*). A typical schedule is shown in figure 29.

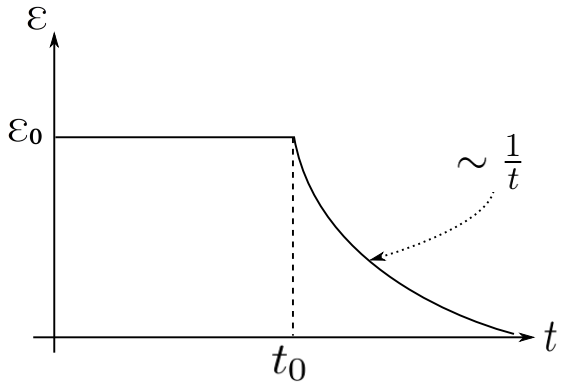
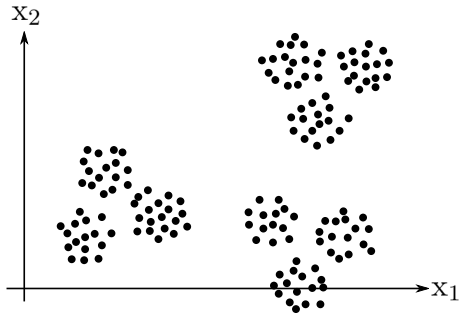


Figure 29: Annealing schedule (t is the number of iterations)

4.1.3 Number of Prototypes

The number of prototypes is a hyperparameter of k-means clustering. Knowledge about the data (noise amplitude) and robustness of the clustering solution can guide this choice.

choice of resolution



1 cluster
3 cluster
9 cluster
many cluster?

$\Rightarrow$ additional prior knowledge is needed!

E^T_{min}: average size of cluster (in terms of variance)

$\rightsquigarrow$ large for few clusters – small for many clusters

$\rightsquigarrow$ zero, if number of cluster $\widehat{=}$ number of data points

Note that E^T_{min} goes down if M increases.

The choice of resolution should depend prior knowledge on the average size of the cluster (e.g. clusters "smaller" than the variance of noise probably do not capture meaningful structure).

$\rightarrow$ exploit prior knowledge on E^T_{min} (e.g. $E^T_{\text{min}} \geq \sigma^2_{\text{noise}}$ which is a natural boundary on E^T_{min})

This prior knowledge can be exploited using the heuristic of *iterative refinement*, as illustrated in algorithm **??**.

Robustness of clustering solution

Idea: If the solution captures meaningful structure in the data, multiple runs of the same algorithm with different initial conditions should yield similar solutions.

Caveat: Assignment of labels to clusters is arbitrary: permutation of labels does neither change cost nor character of the solution.

$$1, 2, 3, \ldots, M$$
$$9, 1, M, \ldots, 7$$

This "permutation symmetry"leads to $M!$ trivially equivalent optima, so the robustness-criterion needs to account for such equivalence classes of optima. The number of equivalence class increases with increasing number of prototypes $\rightsquigarrow$ structurally different equivalent classes to be compared!

How many prototypes? $\rightarrow$ increase number until "overfitting" occurs

Algorithm 9: iterative k-means refinement

initialization: $\underline{\mathbf{w}}_1 = \frac{1}{p}\sum_{\alpha}\underline{\mathbf{x}}^{(\alpha)}$, $\underbrace{\left(E^T_{\min}\right)^*}_{\substack{\text{desired minimal}\\ \text{average variance}}}$, $M = 1$

begin loop

  if $E^T_{\min} < \left(E^T_{\min}\right)^*$ **then** STOP

  select partition $q \in \{1, \ldots, M\}$ with largest variance

$$q = \underset{\gamma}{\operatorname{argmax}} \left(\frac{\sum_{\alpha} m_{\gamma}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\gamma}\right)^2}{\sum_{\alpha} m_{\gamma}^{(\alpha)}} \right)$$

  add a new prototype: $\underline{\mathbf{w}}_{M+1} = \underline{\mathbf{w}}_q + \underbrace{\underline{\eta}_q}_{\substack{\text{small random}\\ \text{vector}}}$

  $M \leftarrow M + 1$

  do k-means clustering with these M prototypes

end

$\rightsquigarrow$ many equivalence classes with approximately equal cost

$\rightsquigarrow$ different initial conditions and different sequences of pattern presentation lead to clustering solutions from different equivalence classes

$\rightsquigarrow$ different sub-datasets lead to solutions from different equivalence classes

Validation measure

LangeEtAl2004 estimate expected dissimilarity between solutions for different samples from the same dataset $\rightsquigarrow$ "best" number of clusters yields most robust clustering (highest similiarity). For further details, see **Luxburg2010**.

- Model free approaches: Stability based validation
- **Idea:** taking to many or too few clusters leads to unstable partitions
- $\mathbf{X} = \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \alpha = 1, \ldots, p; \quad \underline{\mathbf{x}} \in \mathbb{R}^N$; A solution of the clustering algorithm is $\mathbf{Y} = (y_1, \ldots, y_p)$ where $y_i \in L := \{1, \ldots, M\}$
- Comparing clustering solutions Y_1 and Y_2:

$$d := \frac{1}{|\mathbf{Y_1}|} \sum_{\alpha} \mathbf{1}\left\{Y_{1,\alpha} \neq Y_{2,\alpha}\right\}$$

☐ slide: K-means Gaussian data

Algorithm 10: Validation measure

begin for each $M \in \{M_{min}, \ldots, M_{max}\}$

  begin loop for r splits of data

    Split data $\mathbf{X}$ into $\mathbf{X}_1^{(i)}$ and $\mathbf{X}_2^{(i)}$ and find corresponding clustering solutions $\mathbf{Y}_1^{(i)}$ and $\mathbf{Y}_2^{(i)}$

    Compute dissimilarity $d_i := \frac{1}{|\mathbf{X}_2|} \sum_\alpha \mathbf{1}\left\{\mathbf{Y_2}^{(\alpha)} \neq \phi_1[\mathbf{X}_2^{(\alpha)}]\right\}$

    Where ϕ_1 denotes the label of the nearest prototype taking into account permutations

  end

  Compute average dissimilarity: $\hat{S}_{clustering} = \frac{1}{r} \sum_r d_i$

  Sample s random clustering assignments and compute estimate average of the distances to estimate $\hat{S}_{random}$

  Calculate stability index: $\bar{S}_M = \frac{\hat{S}_{clustering}}{\hat{S}_{random}}$

end

Return $\hat{M} = \text{argmin}_M(\bar{S}_M)$

Alternative clustering approaches

- density based models ("model-based" ⇒ Gaussian Mixture algorithm)
- hierarchical (connectivity based) clustering
  - single linkage (∼ nearest neighbor)
  - complete linkage
  - average linkage / within group ssq (Ward criterion)
  - agglomerative vs. divisive clustering

4.2 Pairwise Clustering Methods

In some applications, direct measurements of the objects to be clustered are not available. If information regarding object similarities or distances between objects is available, clustering (∼ grouping objects that are similar to each other) is still possible. In this setting, too, mean field approaches provide effective means to find good clustering solutions (**HofmannBuhmann1997**).

4.2.1 The Clustering Problem

observations: set of p "objects" $\alpha, \alpha = 1, \ldots, p$

distance matrix $\{d_{\alpha\alpha'}\}$

	1	2	3		p
1	0	1.7	0.99		3.0
2	1.7	0	0.3	...	0.1
3	0.9	0.3	0		0.2
⋮		⋮		⋱	⋮
p	3.0	0.1	0.2	...	0

relational representation "pairwise data"

Commonly chosen constraints on the distance matrix e.g. are zero-diagonal, symmetry, or distances fulfilling the triangle-inequality. Examples are:

- distances directly determined by measurements (e.g. dissimilarity judgements in a psychophysics experiment, e.g. confusion matrices)
- distances determined through algorithms (e.g. dissimilarity of protein sequences through sequence alignment procedures, graph-similarity measures)
- distances derived from an underlying vector space representation

$$\left(d : \mathbb{R}^N \text{ x } \mathbb{R}^N \to \mathbb{R}_0^+, \text{ e.g. } d_{\alpha\alpha'} = \frac{1}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\right)^2\right) \tag{4.8}$$

- elements derived via a "kernel trick" (*cf. chapter 2.3.3*)

$$\underline{\phi} : \underline{\mathbf{x}}^{(\alpha)} \to \underline{\phi}_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} \equiv \underline{\phi}^{(\alpha)} \tag{4.9}$$

$$\begin{aligned} d_{\alpha\alpha'} &= \tfrac{1}{2}\left(\underline{\phi}^{(\alpha)} - \underline{\phi}^{(\alpha')}\right)^2 \\ &= \tfrac{1}{2}\left\{\left(\underline{\phi}^{(\alpha)}\right)^2 - 2\left(\underline{\phi}^{(\alpha)}\right)^T \underline{\phi}^{(\alpha')} + \left(\underline{\phi}^{(\alpha')}\right)^2\right\} \\ &= \tfrac{1}{2}\left\{k_{\left(\underline{\mathbf{x}}^{(\alpha)},\underline{\mathbf{x}}^{(\alpha)}\right)} + k_{\left(\underline{\mathbf{x}}^{(\alpha')},\underline{\mathbf{x}}^{(\alpha')}\right)} - 2k_{\left(\underline{\mathbf{x}}^{(\alpha)},\underline{\mathbf{x}}^{(\alpha')}\right)}\right\} \end{aligned} \tag{4.10}$$

cluster models

set of clusters (partitions): $q = 1, \ldots, M$

binary assignment variables:

$$m_q^{(\alpha)} \begin{cases} 1, & \text{if object } \alpha \text{ belongs to cluster } q \\ 0, & \text{else} \end{cases} \tag{4.11}$$

cost function:

$$E_{\left[\left\{m_q^{(\alpha)}\right\}\right]} = \frac{1}{2p}\sum_{q}^{M}\sum_{\alpha} m_q^{(\alpha)} \underbrace{\frac{\sum_{\alpha'} m_q^{(\alpha')} d_{\alpha\alpha'}}{\sum_{\alpha'} m_q^{(\alpha')}}}_{\substack{\text{av. distance between} \\ \alpha \text{ and other objects } \alpha' \\ \text{from the same cluster } q}} = \frac{1}{2p}\sum_{q} \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} d_{\alpha\alpha'}}{\sum_{\alpha} m_q^{(\alpha)}} \tag{4.12}$$

where $\sum_{\alpha} m_q^{(\alpha)}$ is simply the number of objects assigned to cluster q.

model selection:

$$E \stackrel{!}{=} \min \tag{4.13}$$

4.2.2 Pairwise Clustering with Euclidean Distances

observations (feature vectors): $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} = \mathbb{R}^N$

distance measure:

$$d_{\alpha\alpha'} = \frac{1}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\right)^2 \tag{4.14}$$

$$
\begin{aligned}
E_{\left[\left\{m_q^{(\alpha)}\right\}\right]} &= \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\right)^2}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} \left\{\left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 - 2\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\alpha')} + \left(\underline{\mathbf{x}}^{(\alpha')}\right)^2\right\}}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \left\{ \sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 - 2\Big(\sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T\Big) \underbrace{\frac{\sum_{\alpha'} m_q^{(\alpha')} \underline{\mathbf{x}}^{(\alpha')}}{\sum_{\alpha'} m_q^{(\alpha')}}}_{\substack{\stackrel{!}{=}\underline{\mathbf{w}}_q \\ \text{centroid = center of mass} \\ (cf.\ 4.1.2)}} + \sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 \right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left\{\left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 - \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{w}}_q\right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left\{\left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 - \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{w}}_q - \underbrace{\underline{\mathbf{w}}_q^2}_{= \frac{\sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T}{\sum_\alpha m_q^{(\alpha)}} \cdot \underline{\mathbf{w}}_q} + \underline{\mathbf{w}}_q^2 \right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left\{\left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 - 2\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{w}}_q + \underline{\mathbf{w}}_q^2\right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right)^2 \\
&= E_{\left[\left\{m_q^{(\alpha)}\right\}, \left\{\underline{\mathbf{w}}_q\right\}\right]} \widehat{=} \text{ cost function eq.(4.3)}
\end{aligned}
\tag{4.15}
$$

K-means Clustering $\widehat{=}$ Pairwise Clustering with squared Euclidean distance

4.2.3 The Mean-Field Approximation for Pairwise Clustering

discrete (binary) optimization problem:

$$
E_{\left[\left\{m_q^{(\alpha)}\right\}\right]} = \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} d_{\alpha\alpha'}}{\sum_{\alpha'} m_q^{(\alpha')}} \stackrel{!}{=} \min \tag{4.16}
$$

$\Rightarrow$ gradient-based methods are not applicable

$\Rightarrow$ methods from combinatorial optimization are needed

$$\underset{\text{straightforward but slow}}{\text{simulated annealing}} \quad \text{vs.} \quad \underset{\substack{\text{approximation}\\ \text{why? good and fast!}}}{\text{mean-field annealing}}$$

We can apply the framework of section 3.3 but:

$\rightsquigarrow$ variables $m_q^{(\alpha)}$ are *normalized* to $\sum_q m_q^{(\alpha)} = 1$

$\rightsquigarrow$ calculation of moments and mean-fields must be adapted because marginalized variables are not independent anymore

Nomenclature: Using the *set-product* $\otimes$ we define

$\{\underline{\mathbf{m}}^{(\alpha)}\}$: set of all M-dimensional binary vectors $\left(m_1^{(\alpha)}, m_2^{(\alpha)}, \ldots, m_M^{(\alpha)}\right)^T$ which fulfill the normalization condition: exactly one element equals 1.

$\mathscr{M}$: $\{\underline{\mathbf{m}}^{(1)}\} \otimes \{\underline{\mathbf{m}}^{(2)}\} \otimes \ldots \otimes \{\underline{\mathbf{m}}^{(p)}\}$
Kartesian product between all possible binary assignment variables i.e. all possible valid assignments for the full dataset

$\mathscr{M}_\gamma$: $\{\underline{\mathbf{m}}^{(1)}\} \otimes \ldots \otimes \{\underline{\mathbf{m}}^{(\gamma-1)}\} \otimes \{\underline{\mathbf{m}}^{(\gamma+1)}\} \otimes \ldots \otimes \{\underline{\mathbf{m}}^{(p)}\}$
i.e. set of all possible assignments for all datapoints except γ

assignment noise $\rightarrow$ Gibbs distribution

$$P_{\left(\{m_q^{(\alpha)}\}\right)} = \frac{1}{Z_p} \exp\Big\{ -\beta E^p_{\left[\{m_q^{(\alpha)}\}\right]} \Big\} \tag{4.17}$$

where

$$Z_p = \sum_{\mathscr{M}} \exp\Big\{ -\beta E^p_{\left[\{m_q^{(\alpha)}\}\right]} \Big\} \tag{4.18}$$

factorizing distribution

$$Q_{\left[\{m_q^{(\alpha)}\}\right]} = \frac{1}{Z_Q} \exp\Big\{ -\beta \sum_{p,\gamma} m_p^{(\gamma)} \underbrace{e_p^{(\gamma)}}_{\text{mean-fields}} \Big\} \tag{4.19}$$

where:

$$Z_Q = \sum_{\mathscr{M}} \exp\Big\{ -\beta \sum_{p,\gamma} m_p^{(\gamma)} e_p^{(\gamma)} \Big\} \tag{4.20}$$

this allows to calculate the *first moments* w.r.t Q ($\rightarrow$ assignment probabilities).

$$\left\langle m_q^{(\gamma)} \right\rangle_Q = \frac{1}{Z_Q} \sum_{\mathscr{M}} m_q^{(\gamma)} \exp\Big\{ -\beta \sum_{r,\delta} m_r^{(\delta)} e_r^{(\delta)} \Big\} \tag{4.21}$$

using the factorization eq. (3.9) regarding valid assignments $\{\underline{\mathbf{m}}^{(\gamma)}\}$ for observation γ and the rest of the variables this simplifies (for any functions f, g) to:

$$\sum_{\mathscr{M}} \left[f_{\left(\left\{m_p^{(\delta)}\middle|\delta\neq\gamma\right\}\right)} \cdot g_{\left\{m_p^{(\delta)}\middle|\delta=\gamma\right\}} \right] = \left[\sum_{\mathscr{M}_\gamma} f_{\left(\left\{m_p^{(\delta)}\middle|\delta\neq\gamma\right\}\right)} \right] \cdot \left[\sum_{\left\{\underline{\mathbf{m}}^{(\gamma)}\right\}} g_{\left\{m_p^{(\delta)}\middle|\delta=\gamma\right\}} \right] \tag{4.22}$$

and finally gives

$$\begin{aligned} \left\langle m_q^{(\gamma)} \right\rangle_Q &= \frac{\left[\sum\limits_{\mathscr{M}_\gamma} \exp\left\{ -\beta \sum\limits_{r,\delta\neq\gamma} m_r^{(\delta)} e_r^{(\delta)} \right\} \right] \cdot \left[\overbrace{\sum\limits_{\left\{\underline{\mathbf{m}}^{(\gamma)}\right\}} m_q^{(\gamma)}}^{\substack{\text{only term with} \\ m_q^{(\gamma)}=1 \\ \text{remains}}} \exp\left\{ -\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right\} \right]}{\underbrace{\left[\sum_{\mathscr{M}_\gamma} \exp\left\{ -\beta \sum_{r,\delta\neq\gamma} m_r^{(\delta)} e_r^{(\delta)} \right\} \right]}_{\text{first terms cancel}} \cdot \left[\sum_{\left\{\underline{\mathbf{m}}^{(\gamma)}\right\}} \exp\left\{ -\beta \underbrace{\sum_r m_r^{(\gamma)} e_r^{(\gamma)}}_{\substack{\text{only one term of this} \\ \text{sum remains for every} \\ \text{term of the qrevious sum}}} \right\} \right]} \\ &= \frac{\exp\left\{-\beta\, m_q^{(\gamma)} e_q^{(\gamma)}\right\}}{\sum_r \exp\left\{-\beta\, m_r^{(\gamma)} e_r^{(\gamma)}\right\}} \underbrace{=}_{\substack{\text{only the} \\ m_r^{(\gamma)}=1 \\ \text{stays}}} \underbrace{\frac{\exp\left\{ -\beta e_q^{(\gamma)} \right\}}{\sum\limits_r \exp\left\{ -\beta e_r^{(\gamma)} \right\}}}_{\text{soft-max of the mean-fields}} \end{aligned} \tag{4.23}$$

The $\left\langle m_q^{(\gamma)} \right\rangle_Q \in [0,1]$ represent assignment probabilities, i.e. they quantify the probability that an object belongs to a cluster (since we have $\sum_r \langle m_r^{(\gamma)} \rangle = 1$).

$\rightsquigarrow\ \beta \to \infty : \left\langle m_q^{(\gamma)} \right\rangle_Q \to \{0,1\}$ "hard assignments" (cmp. k-means)

minimization of the KL-divergence (eq. 3.14) leads to:

$$\frac{\partial \left\langle E^p \right\rangle_Q}{\partial e_q^{(\alpha)}} - \sum_{r,\gamma} \frac{\overbrace{\partial \left\langle m_r^{(\gamma)} \right\rangle_Q}^{\substack{\text{depends only on} \\ \text{data point } \gamma(\text{ not } \alpha)}}}{\partial e_q^{(\alpha)}} e_r^{(\gamma)} \stackrel{!}{=} 0 \tag{4.24}$$

$$\frac{\partial \left\langle E^p \right\rangle_Q}{\partial e_q^{(\alpha)}} - \sum_{r} \frac{\partial \left\langle m_r^{(\alpha)} \right\rangle_Q}{\partial e_q^{(\alpha)}} e_r^{(\alpha)} \stackrel{!}{=} 0 \tag{4.25}$$

for the mean-fields we obtain

$$\begin{aligned} e_q^{(\alpha)} &= \frac{1}{p}\Bigg[\frac{\sum_{\alpha',\gamma}}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q}\Bigg\{d_{\alpha\alpha'} - \frac{1}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q}\sum_{\delta\neq\alpha}\langle m_q^{(\gamma)}\rangle_Q d_{\delta\delta}\Bigg\} \\ &\quad + \frac{1}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q}\sum_{\delta\neq\alpha}\langle m_q^{(\delta)}\rangle_Q\Bigg\{\big(d_{\delta\alpha}+d_{\alpha\delta}\big) - \frac{1}{2}\frac{1}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q} \\ &\quad \sum_{\varepsilon\neq\delta,\alpha}\langle m_q^{(\varepsilon)}\rangle_Q\cdot\big(d_{\delta\varepsilon}+d_{\varepsilon\delta}\big)\Bigg\}\Bigg] \end{aligned} \tag{4.26}$$

proof: supplementary material

The terms $d_{\delta\alpha} + d_{\alpha\delta}$ and $d_{\delta\varepsilon} + d_{\varepsilon\delta}$ illustrate that the mean-field approximation leads to an evaluation of the symmetrized distance matrix.

Therefore, a common assumption wrt. the *distance matrix* is:

$\rightsquigarrow$ symmetric: $d_{\alpha,\alpha'} = d_{\alpha',\alpha}$

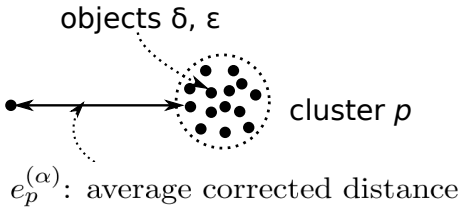
$\rightsquigarrow$ diagonal elements $d_{\alpha\alpha'} \stackrel{!}{=} 0$

Under this assumption, the mean-fields simplify as:

$$e_q^{(\alpha)} = \underbrace{\frac{2}{p}\frac{1}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q}\sum_\delta \langle m_q^{(\delta)}\rangle_Q \underbrace{\Bigg\{d_{\delta\alpha} - \frac{1}{2}\frac{1}{\sum_\gamma \langle m_q^{(\gamma)}\rangle_Q}\sum_\varepsilon \langle m_q^{(\varepsilon)}\rangle_Q d_{\varepsilon\delta}\Bigg\}}_{\substack{\text{distance between data objects } \alpha \text{ and } \delta, \\ \text{corrected by the average distance between} \\ \text{objects of the cluster, to which } \delta \text{ belongs (here: } q)}}}_{\substack{\text{average corrected distance between data objects} \\ \alpha \text{ and all objects } \delta \text{ of cluster } q}} \tag{4.27}$$

interpretation of the "mean fields":

$\Rightarrow$ "metric visualization" of the $e_q^{(\alpha)}$:



$e_p^{(\alpha)}$: average corrected distance

$\Rightarrow$ mean-fields depend on assignment probabilities $\langle m_q^{(\alpha)}\rangle_Q$ rather than on the "hard" binary assignments

- $\rightsquigarrow$ this is an effect of the underlying stochastic optimization procedure
- $\rightsquigarrow$ "fuzzy" memberships: objects contribute only weighted by their probability $\langle m_q^{(\alpha)} \rangle_Q$ of assignment

$\Rightarrow$ assignment probabilities $\langle m_1^{(\alpha)} \rangle_Q$ depend on the average normalized distance between the object α under consideration and the objects of cluster q (probability is high, if distance to α is small compared to average distance within cluster).

Euclidean distances: As shown above, using Euclidean distance as a pairwise similarity measure is a special case and yields particularly simple expressions for the mean fields and prototypes.

For observations (feature vectors): $\underline{\mathbf{x}}^{(\alpha)}; \alpha = 1, \ldots, p; \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$ and euclidean distance measure:

$$d_{\alpha\alpha'} = \frac{1}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\big)^2 \tag{4.28}$$

we then obtain:

$$e_q^{(\alpha)} = \frac{1}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\big)^2 \tag{4.29}$$

with

$$\underline{\mathbf{w}}_q = \underbrace{\frac{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q \underline{\mathbf{x}}^{(\gamma)}}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q}}_{\substack{\text{center of mass of all objects weighted} \\ \text{by their probability of assignment}}} \tag{4.30}$$

proof: supplementary material

- $\rightsquigarrow$ probabilistic cluster assignments. thus:
- $\rightsquigarrow$ natural extension of K-means clustering to the case of "fuzzy" assignments

4.2.4 General Mean-Field Algorithm for Pairwise Clustering

Algorithm **??** implements the stochastic optimization procedure (mean field annealing) from chapter 3.3 to find nearly optimal cluster centers and assignment (probabilities) of data points to these clusters.

Algorithm 11: soft k-means clustering for general distances

Initialization:
- choose number M of partitions (max no)
- choose initial (β_0) and final (β_f) values of the noise parameter
- choose annealing factor η and convergence criterion θ
- initialize mean-fields $e_q^{(\alpha)}$ with random numbers $\in [0,1]$
- set $\beta \leftarrow \beta_0$

while $\beta < \beta_f$ **do** annealing

  repeatEM fixpoint iteration

    compute assignment probabilities

$$\left\langle m_q^{(\alpha)} \right\rangle_Q = \frac{\exp\left\{ -\beta \left(e_q^{(\alpha)}\right)_{\text{old}} \right\}}{\sum\limits_r \exp\left\{ -\beta \left(e_r^{(\alpha)}\right)_{\text{old}} \right\}} \text{ for all } q, \alpha$$

    compute new mean-fields

$$\begin{aligned} \left(e_q^{(\alpha)}\right)_{\text{new}} &= \frac{2}{M} \frac{1}{\sum\limits_\gamma \left\langle m_q^{(\gamma)} \right\rangle_Q} \sum_\delta \left\langle m_q^{(\delta)} \right\rangle_Q \\ &\quad \cdot \left\{ d_{\delta\alpha} - \frac{1}{2} \frac{1}{\sum\limits_\gamma \left\langle m_q^{(\gamma)} \right\rangle_Q} \sum_\varepsilon \left\langle m_q^{(\varepsilon)} \right\rangle_Q d_{\varepsilon\delta} \right\} \text{ for all } q, \alpha \end{aligned}$$

  until $\left| \left(e_q^{(\alpha)}\right)_{\text{new}} - \left(e_q^{(\alpha)}\right)_{\text{old}} \right| < \theta$ *for all* q, α

  $\beta \leftarrow \eta \cdot \beta$

end

Application example: clustering of protein sequences

definition of a distance measure:

- $\rightsquigarrow$ pairwise sequence alignment
- $\rightsquigarrow$ definition of distance on the basis of number of insertions, deletions and amino acid exchanges ('edit' distance)

slide: Clustering of protein sequences

In a similar way, spiketrains can be clustered to identify groups of cells with similar response properties ($\rightsquigarrow$ spiketrain metrics).

4.2.5 Missing Data

The algorithm for pairwise data lends itself in a natural way do deal with *missing data* or subsets of data to reduce computational burden (e.g when computing distances is costly).

- $\rightsquigarrow$ number of matrix elements $\sim p^2$
- $\rightsquigarrow$ calculation or measurement of all distances may be computationally expensive or even unfeasible
- $\rightsquigarrow$ distance matrices are often "redundant" $\rightsquigarrow$ not all matrix entries might be needed (e.g., position on plane: 3 dists. enough)

As can be seen from the average distance of data object α to all data objects of cluster q:

$$\overline{d}_{q\alpha} = \frac{\sum\limits_{\gamma} \left\langle m_q^{(\gamma)} \right\rangle_Q d_{\gamma\alpha}}{\sum\limits_{\gamma} \left\langle m_q^{(\gamma)} \right\rangle_Q} \tag{4.31}$$

and the mean fields (see eq. 4.27) can be written in termns of these:

$$e_q^{(\alpha)} = \left\{ \overline{d}_{q\alpha} - \frac{1}{2} \sum_{\delta} \left\langle m_q^{(\delta)} \right\rangle_Q \overline{d}_{q\delta} \right\} \cdot \frac{2}{M} \tag{4.32}$$

These computations depend only on the *average* distances and therefore enable the following *missing data heuristics*:

- $\Rightarrow$ estimate average values $\overline{d}_{p\alpha}$ using the measured distances only
- $\Rightarrow$ perform summations within $\overline{d}_{p\alpha}$ only over the available distances

Good choice of a subset of data to compute the average values can significantly speed up clustering without strongly changing the solution.

4.2.6 Special case: "soft" K-means with Euclidean distances

Using the *squared euclidean distance* as the distance measure in the general procedure of algorithm **??** yields a particularly simple version of the soft clustering procedure that is described in algorithm **??**. It is robust, fast, and allows to choose k.

Algorithm 12: soft k-means clustering for Euclidean distances

Initialization:
- choose no. M of partitions (max no)
- choose initial (β_0) and final (β_f) values of the noise parameter
- initialize prototypes: $\underline{\mathbf{w}}_q = \frac{1}{p}\sum_{\alpha} \underline{\mathbf{x}}^{(\alpha)} + \underline{\eta}_q$ (small random vector)
- choose annealing factor η
- choose convergence criterion θ
- $\beta \leftarrow \beta_0$

while $\beta < \beta_f$ *(annealing)* **do**

  repeat EM

    compute assignment probabilities

$$\langle m_q^{(\alpha)} \rangle_Q = \frac{\exp\left\{ -\frac{\beta}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}}\right)^2 \right\}}{\sum_r \exp\left\{ -\frac{\beta}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r^{\text{old}}\right)^2 \right\}} \quad \text{for all } \alpha, q$$

    compute new prototypes

$$\underbrace{\underline{\mathbf{w}}_q^{\text{new}} = \frac{\sum_{\alpha} \langle m_q^{(\alpha)} \rangle_Q \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha} \langle m_q^{(\alpha)} \rangle_Q}}_{\substack{\text{center of mass of the data points} \\ \text{which belong to cluster } q \text{ - weighted} \\ \text{by assignment probability}}} \quad \text{for all } q$$

  until $\left|\underline{\mathbf{w}}_q^{\text{new}} - \underline{\mathbf{w}}_q^{\text{old}}\right| < \theta$ *for all* q

  $\beta \leftarrow \eta\beta$

end

Comments

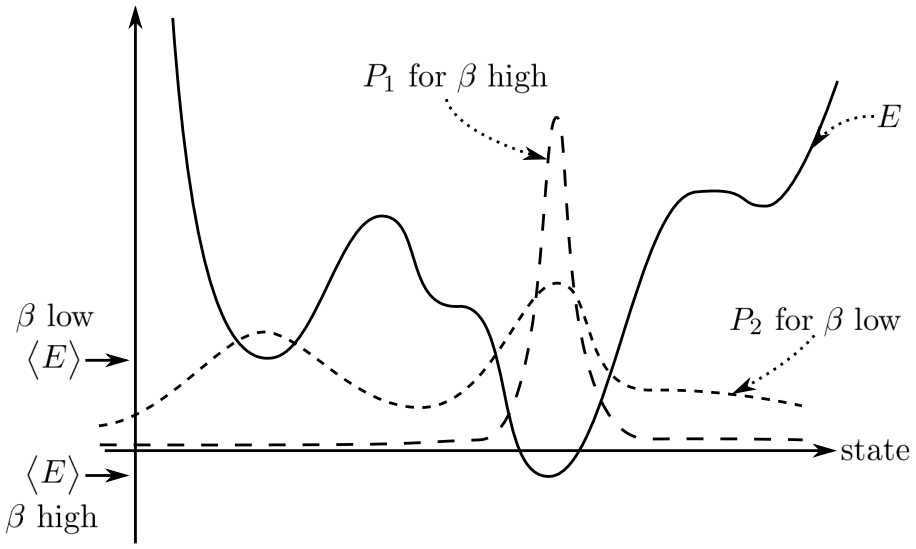
- for an "on-line" version: replace inner loop by:
  `choose observation` $\underline{\mathbf{x}}^{(\alpha)}$
  `compute assignment probabilities` $\langle m_p^{(\alpha)} \rangle_Q$ for all p
  `change all prototypes according to`

$$\Delta \underline{\mathbf{w}}_p = \varepsilon \langle m_p^{(\alpha)} \rangle_Q \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_p\right)$$

- principled alternative to fuzzy clustering methods
- mean-field annealing is robust against convergence to local optima
- choice of nooise parameter $\beta \Rightarrow$ "resolution" of cluster analysis

average cost (Gibb's distribution of simulated annealing):

$$\langle E \rangle = \frac{1}{Z} \sum_{\left\{m_p^{(\alpha)}\right\}} E \exp\left\{ -\beta E \right\} \tag{4.33}$$



increase of β implies:

$\rightarrow$ decrease of average cost

$\rightarrow$ decrease of cluster size

$\rightarrow$ increase in spatial resolution ($\rightsquigarrow$ hierarchical clustering)

$\Rightarrow$ β controls the "complexity" of the clustering solution

$\rightarrow$ sudden increase in number of clusters hints at 'overfitting'

For further details, see **RoseEtAl1990** and **Rose1998**.

4.3 Self-Organising Maps

Self-organizing maps (SOMs) are an example of local *embedding* techniques motivated by principles of neural development & plasticity. Although there are more efficient embedding methods ($\rightsquigarrow$ BigData) SOMs are useful tools to extract and visualize statistical structure of high-dimensional data. For a detailed exposition of SOMs, see **Kohonen2001**, for alternative embedding techniques, see e.g. *MultiDimensional Scaling* (MDS, Sammon mapping), ISOMAP **TenenbaumEtAl2000** or Local Linear Embedding **RoweisSaul2000**.

4.3.1 Kohonen Networks

observations: $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p$

goal:

- $\rightsquigarrow$ clustering of data based on similarity
- $\rightsquigarrow$ low-dimensional and **neighborhood preserving** representation for the purpose of visualization

distance measure: squared Euclidean distance

neural network: Set of units with a geometrical structure (see figure 30).

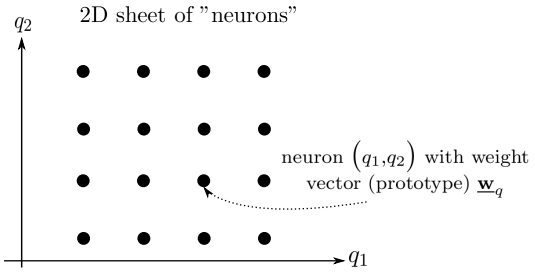


Figure 30: Units of a neural network are arranged on a map with 'coordinates' q_1 and q_2 and each represent a prototype.

The units in this network are binary "neurons" with "activities" (assignment to prototype $\underline{\mathbf{q}} = (q_1, q_2)^T$):

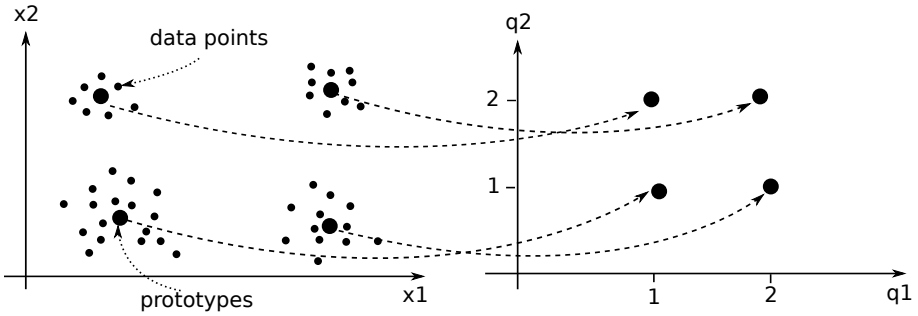
$$m_{\underline{\mathbf{q}}}^{(\alpha)} = \begin{cases} 1, & \text{if } \underline{\mathbf{q}} = \text{argmin}_{\underline{\mathbf{r}}} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{r}}} \right| \\ 0, & \text{else} \end{cases} \tag{4.34}$$

$\Rightarrow$ "winner-takes-all" network (WTA)

$\Rightarrow$ "competitive" network

Topographic maps: arrangement of groups on a 'map' such that data-points in *neighboring groups* are similar.

→ "neigboring" neurons (within the neural network or map) should represent closeby data points (similar in data/feature space)



"map" of data space
- visualization
- dimension reduction
- preprocessing for prediction

Notes regarding algorithm ??

- a common choice to initialise the prototypes is to use the data-mean and small vectors $\underline{\eta}_q$ along the first 2 PCs.
- learning in topographic maps can be understood as a modification of the K-means clustering method that breaks permutation symmetry $\rightsquigarrow$ neighboring neurons should undergo similar changes of prototypes during learning

Interpretation of the neighborhood function $h_{\underline{\mathbf{qp}}}$

- large for neighboring neurons in the neural network
- enforces similar learning steps for neighboring neurons
- typical choice:

$$h_{\underline{\mathbf{qp}}} = \exp\left\{ -\frac{(\underline{\mathbf{q}} - \underline{\mathbf{p}})^2}{2\sigma^2} \right\} \qquad \text{(Gauss function)}$$

Algorithm 13: Online learning for SOMs (Kohonen map) – "Vanilla Kohonen"

Initialization:
- choose no. M of partitions ("neurons")
- choose annealing schedule for ε and σ
- initialize prototypes, e.g.: $\underline{\mathbf{w}}_{\underline{\mathbf{q}}} = \underline{\mathbf{x}} + \underline{\eta}_{\underline{\mathbf{q}}}$ (faster convergence: center of mass + random in plane of first two principal components)

begin

choose a data point $\underline{\mathbf{x}}^{(\alpha)}$

determine the closest prototype:

$$\underline{\mathbf{p}} = \underset{\underline{\mathbf{r}}}{\operatorname{argmin}} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{r}}}^{\text{old}} \right|$$

change prototypes according to:

$$\Delta \underline{\mathbf{w}}_{\underline{\mathbf{q}}} = \varepsilon \, h_{\underline{\mathbf{qp}}} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}^{\text{old}} \right)$$

end

$\rightsquigarrow$ using $\delta_{\underline{\mathbf{qp}}}$ as the neighborhood function $h_{\underline{\mathbf{qp}}}$ in algorithm (**??**) results in standard k-means

- *annealing of the parameter σ:* Start with σ large ($\rightsquigarrow$ neighborhood function convex over its support) and decrease linearly or exponentially (but "slow") during learning. Solution will depend on final value of σ:
  - $\rightarrow$ $\sigma = 0$: solution corresponds to a minimum of the K-means clustering cost function (cf. section 4.1.1) but will be *neighborhood preserving* ($\rightsquigarrow$ permutation symmetry)
  - $\rightarrow$ σ small but finite: better visualization capabilities at the expense of a non-optimal clustering cost

Dimension reducing mappings: For observations $\underline{\mathbf{x}} \in \mathbb{R}^N, N$ typically larger than 2, 2D Self-Organizing Map can be used for visualization purposes. How this reduction of dimensionality is performed is illustrated in figure 31 for a 1D example with finite range σ of the neighborhood function.

$\Rightarrow$ automatic selection of relevant feature dimension

$\Rightarrow$ hierarchical maps, semantic maps, OR/OD maps

$\Rightarrow$ depending on the purpose, number of network units should be chosen small ($\rightsquigarrow$ clustering) or large enough ($\rightsquigarrow$ visualisation).

3D surface, $\square$Leptograpsus data

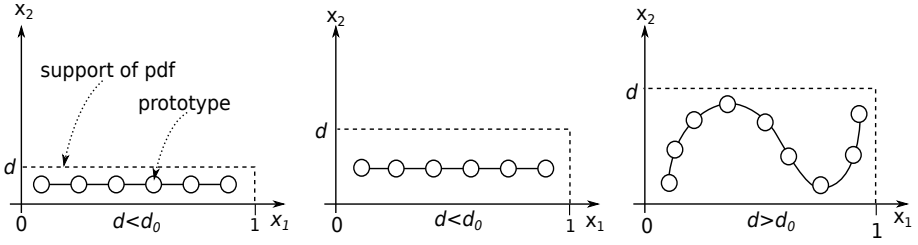


Figure 31: Dimensionality reduction with SOMs: For small d, the one-dimensional topographic map naturally covers variability of the data. For large d, the map starts to meander to capture both directions of variability in the data.

4.3.2 Self-Organizing Maps for Pairwise Data

Data: distance matrix $\underline{\mathbf{D}}$ specifying the distances or 'dissimilarities' $d_{\alpha\alpha'}$ between a set of p "objects" $\alpha, \alpha' = 1, \ldots, p$

Model: set of M clusters (partitions) $\underline{\mathbf{q}}$ with a geometrical structure (e.g. 1-d line or 2-d grid).

binary assignment variables (normalized):

$$m_{\underline{\mathbf{q}}}^{(\alpha)} = \begin{cases} 1, & \text{if object } \alpha \text{ belongs to cluster } \underline{\mathbf{q}} \\ 0, & \text{else} \end{cases} \tag{4.35}$$

cost function:

$$E_{\left[\left\{m_{\underline{\mathbf{q}}}^{(\alpha)}\right\}\right]} = \frac{1}{p} \sum_{\underline{\mathbf{r}}} \frac{\sum\limits_{\alpha,\alpha'} \Big(\sum\limits_{\underline{\mathbf{q}}} \overbrace{h_{\underline{\mathbf{r}}\underline{\mathbf{q}}}}^{\substack{\text{neighborhood} \\ \text{function}}} m_{\underline{\mathbf{q}}}^{(\alpha)} \Big) \Big(\sum\limits_{\underline{\mathbf{q}}} h_{\underline{\mathbf{r}}\underline{\mathbf{q}}} m_{\underline{\mathbf{q}}}^{(\alpha')} \Big) d_{\alpha\alpha'}}{\sum\limits_{\alpha} \Big(\sum\limits_{\underline{\mathbf{q}}} h_{\underline{\mathbf{r}}\underline{\mathbf{q}}} m_{\underline{\mathbf{q}}}^{(\alpha)} \Big)} \tag{4.36}$$

$\rightsquigarrow$ E corresponds to the average cost (min. data pt. distance) of groups that are close by

$\rightsquigarrow$ replace $m_q^{(\alpha)}$ of pairwise clustering by $\sum\limits_{\underline{\mathbf{q}}} h_{\underline{\mathbf{r}}\underline{\mathbf{q}}} m_{\underline{\mathbf{q}}}^{(\alpha)}$

$\rightsquigarrow$ "neighboring" cluster (w.r.t. $h_{\underline{\mathbf{r}}\underline{\mathbf{q}}}$) contribute to the total average distance

$\rightsquigarrow$ "neighborhood preserving maps" induce lower cost

model selection

$$E_{\left[\left\{m_{\underline{\mathbf{q}}}^{(\alpha)}\right\}\right]} \stackrel{!}{=} \min \tag{4.37}$$

Optimization: similar to pairwise mean field clustering (alg. **??**), algorithm **??** implements mean-field annealing to train SOMs on pairwise data (see **GraepelObermayer1999**).

Algorithm 14: Meanfield EM-learning for SOMs with pairwise data

Initialization:
- choose no. of partitions
- choose initial (β_0) and final (β_f) values of the noise parameter
- choose annealing factor η
- choose convergence criterion θ
- initialization of mean-fields $e_{\underline{\mathbf{q}}}^{(\alpha)}$: random numbers $\in [0,1]$

$\beta \leftarrow \beta_0$

while $\beta < \beta_e$ **do** annealing

  repeat EM

    compute assignment probabilities

$$\left\langle m_{\underline{\mathbf{q}}}^{(\alpha)} \right\rangle_Q = \frac{\exp\left\{-\beta\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{old}}\right\}}{\sum\limits_r \exp\left\{-\beta\left(e_{\underline{\mathbf{r}}}^{(\alpha)}\right)_{\text{old}}\right\}} \text{ for all } \underline{\mathbf{q}}, \alpha$$

    compute new mean-fields

$$\begin{aligned}\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{new}} \;=\; & \tfrac{1}{p} \sum_{\underline{\mathbf{s}}} \underbrace{h_{\underline{\mathbf{sq}}}}_{\circledast} \left[\frac{1}{\sum\limits_\gamma \left(\sum\limits_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sq}}} \left\langle m_{\underline{\mathbf{r}}}^{(\gamma)} \right\rangle_Q \right)} \sum_\delta \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \left\langle m_{\underline{\mathbf{r}}}^{(\delta)} \right\rangle_Q \right. \right. \\ & \left. \left. \cdot \left\{ d_{\delta\alpha} - \tfrac{1}{2} \frac{1}{\sum\limits_\gamma \left(\sum\limits_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \left\langle m_{\underline{\mathbf{r}}}^{(\gamma)} \right\rangle_Q \right)} \sum_\varepsilon \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \left\langle m_{\underline{\mathbf{r}}}^{(\varepsilon)} \right\rangle_Q \right) d_{\varepsilon\delta} \right\} \right] \\ & \text{for all } \underline{\mathbf{q}}, \alpha\end{aligned}$$

  until $\left|\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{new}} - \left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{old}}\right| < \theta$ *for all* $\underline{\mathbf{q}}, \alpha$

  $\beta \leftarrow \eta\beta$

end

Comments

- replacing $h_{\underline{\mathbf{sp}}}$ by $\delta_{\underline{\mathbf{sp}}}$ in algorithm **??** recovers standard pairwise clustering (see section 4.2.4)

- replacing $h_{\underline{\mathbf{sp}}}$ by $\delta_{\underline{\mathbf{sp}}}$ for the neighborhood function (only) at $\circledast$ in algorithm **??** is called the "Kohonen-approximation" (because the original algorithm suggested by T. Kohonen is recovered for squared Euclidean distances $d_{\alpha\alpha'}$ and $\beta \to \infty$)

  $\to$ reduction of computational cost

  $\to$ visualization properties remain

- no need to anneal σ since mean-field annealing ("robust method")

noisy spiral, ☐ brain connectivity pattern

4.3.3 Euclidean Distances

observations (feature vectors): $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p, \underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$

distance measure:

$$d_{\alpha\alpha'} = \frac{1}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\big)^2 \tag{4.38}$$

cost function:

$$\begin{aligned} E_{\left[\left\{m_{\underline{\mathbf{q}}}^{(\alpha)}\right\}\right]} &= \tfrac{1}{p}\sum_{\underline{\mathbf{q}},\alpha}\Big(\sum_{\underline{\mathbf{p}}} h_{\underline{\mathbf{qp}}} m_{\underline{\mathbf{p}}}^{(\alpha)}\Big)\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\big)^2 \\ &= \tfrac{1}{p}\sum_{\underline{\mathbf{p}}\alpha} m_{\underline{\mathbf{p}}}^{(\alpha)} \sum_{\underline{\mathbf{q}}} h_{\underline{\mathbf{qp}}}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}\big)^2 \qquad (4.39) \\ &= E^T_{\left[\left\{m_{\underline{\mathbf{q}}}^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_{\underline{\mathbf{q}}}\right\}\right]} \end{aligned}$$

$\widehat{=}$ K-means cost function, but with a different distance measure:

$$\sum_q h_{\underline{\mathbf{qp}}}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}\big)^2 \tag{4.40}$$

instead of: $\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{p}}}\big)^2$

where:

$$\underline{\mathbf{w}}_{\underline{\mathbf{q}}} = \underbrace{\frac{\sum_{\alpha'}\Big(\sum_{\underline{\mathbf{p}}} h_{\underline{\mathbf{qp}}} m_{\underline{\mathbf{p}}}^{(\alpha')}\Big)\underline{\mathbf{x}}^{(\alpha')}}{\sum_{\alpha'}\Big(\sum_{\underline{\mathbf{p}}} h_{\underline{\mathbf{qp}}} m_{\underline{\mathbf{p}}}^{(\alpha')}\Big)}}_{\substack{\text{center of mass of all data}\\ \text{which belongs to cluster } \underline{\mathbf{p}},\\ \text{weighted by the neigborhood}\\ \text{function } h_{\underline{\mathbf{qp}}}}} \tag{4.41}$$

proof: replace $m_q^{(\alpha)}$ by $\sum_{\underline{\mathbf{p}}} h_{\underline{\mathbf{qp}}} m_{\underline{\mathbf{p}}}^{(\alpha)}$ in the corresponding derivation in section 4.2.2.

On-line Minimization of $E_{\left[\left\{m_q^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q\right\}\right]}$: Using pairwise squared distances yields a simple on-line version (Algorithm **??**) of the learning algorithm for topographic maps.

Algorithm 15: Online learning for SOMs and Euclidean distances

Initialization of prototypes
Select learning step η
begin
choose a data point $\underline{\mathbf{x}}^{(\alpha)}$
assign data point to the 'closest' prototype

$$\underline{\mathbf{p}} = \underset{\underline{\mathbf{r}}}{\operatorname{argmin}} \sum_{\underline{\mathbf{p}}} \underbrace{h_{\underline{\mathbf{rp}}}}_{\circledast} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{p}}}^{\text{old}}\right)^2$$

change all prototypes according to

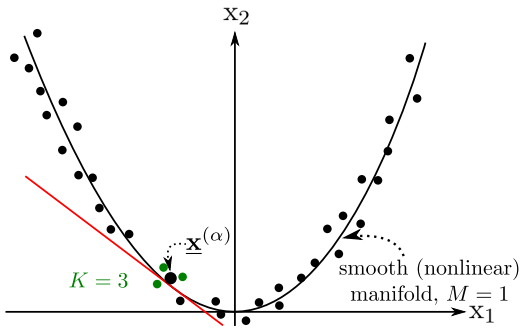
$$\Delta \underline{\mathbf{w}}_{\underline{\mathbf{q}}} = \eta h_{\underline{\mathbf{pq}}} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}\right) \text{ for all } \underline{\mathbf{q}}$$

end

Comments

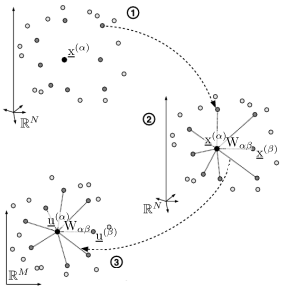
- cost-function based approach to the Self-Organizing Map
- $h_{\underline{\mathbf{qr}}} \to \delta_{\underline{\mathbf{qr}}}$ at $\circledast$ (see above) is called the Kohonen-approximation
- using the Kohonen approximation and Euclidean distance in section 4.3.2 leads to a "deterministic annealing" version of the standard Self-Organizing Map

4.4 Locally Linear Embedding



Project the data into the tangential (linear) space of the data manifold.

- data points $\underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^N$
- embedded data points $\underline{\mathbf{u}}^{(\alpha)} \in \mathbb{R}^M, \quad M < N$



For each data point $\underline{\mathbf{x}}^{(\alpha)}$

- find the K nearest neighbors
- calculate reconstruction weights $\underline{\mathbf{W}}$ s.t. $\underline{\mathbf{x}}^{(\alpha)} \approx \sum_{\beta \in \mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)})} \mathrm{W}_{\alpha\beta} \cdot \underline{\mathbf{x}}^{(\beta)}$
- obtain embedding $\underline{\mathbf{u}}^{(\alpha)} \in \mathbb{R}^M$ s.t. $\underline{\mathbf{u}}^{(\alpha)} \approx \sum_{\beta \in \mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)})} \mathrm{W}_{\alpha\beta} \cdot \underline{\mathbf{u}}^{(\beta)}$

Step 1: find K nearest neighbors choice: Euclidean distance

$$
\begin{aligned}
\beta_1^{(\alpha)} &= \arg\min_{\beta} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta)} \right| \\
\beta_2^{(\alpha)} &= \arg\min_{\beta \neq \beta_1^{(\alpha)}} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta)} \right| \\
&\vdots \\
\beta_K^{(\alpha)} &= \arg\min_{\beta \neq \beta_1^{(\alpha)}, k=1,\ldots,K-1} \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta)} \right| \\
\mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)}) &= \left\{ \beta_1^{(\alpha)}, \beta_2^{(\alpha)}, \ldots, \beta_K^{(\alpha)} \right\} \qquad \text{(not necessarily unique)}
\end{aligned}
$$

linear data structure (e.g. data matrix): $\mathcal{O}(Np^2)$
k-d tree (balanced search tree): $\mathcal{O}(Np \log p)$

Step 2: calculate reconstruction weights
minimize cost function:

$$E(\underline{\mathbf{W}}) = \sum_{\alpha=1}^{p} \underbrace{\left| \underline{\mathbf{x}}^{(\alpha)} - \sum_{\beta=1}^{p} \mathrm{W}_{\alpha\beta} \underline{\mathbf{x}}^{(\beta)} \right|^2}_{\substack{\text{reconstruct } \underline{\mathbf{x}}^{(\alpha)} \text{ by its} \\ K \text{ nearest neighbors only}}} \stackrel{!}{=} \min \quad \text{s.t.} \quad \mathrm{W}_{\alpha\beta} = 0 \text{ if } \beta \notin \mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)}), \quad \sum_{\beta=1}^{p} \mathrm{W}_{\alpha\beta} = 1$$

weight matrix $\underline{\mathbf{W}}$:

- sparse: (up to) K nonzero elements per row
- not symmetric: nearest neighbors of a data point can have closer neighbors

optimal weights are invariant to:

- rotation $\underline{\mathbf{R}}$: $E\left[\underline{\mathbf{R}} \cdot \underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{R}} \cdot \underline{\mathbf{x}}^{(p)}\right] \stackrel{\text{orthog. } \underline{\mathbf{R}}}{=} E\left[\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)}\right]$
- scaling $\gamma > 0$: $E\left[\gamma \underline{\mathbf{x}}^{(1)}, \ldots, \gamma \underline{\mathbf{x}}^{(p)}\right] = \gamma^2 E\left[\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)}\right]$
- translation $\Delta\underline{\mathbf{x}}$: $E\left[\underline{\mathbf{x}}^{(1)} + \Delta\underline{\mathbf{x}}, \ldots, \underline{\mathbf{x}}^{(p)} + \Delta\underline{\mathbf{x}}\right] \stackrel{\sum_\beta \mathrm{W}_{\alpha\beta}=1}{=} E\left[\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)}\right]$

for each data point $\underline{\mathbf{x}}^{(\alpha)}$:

- local "covariance" matrix (symmetric & positive semidefinite) $\underline{\mathbf{C}}^{(\alpha)} \in \mathbb{R}^{K,K}$:
$$\mathrm{C}_{jk} = \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta_j^{(\alpha)})}\right)^T \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta_k^{(\alpha)})}\right)$$
- solve linear system $\underline{\mathbf{C}}^{(\alpha)} \underline{\widetilde{\mathbf{w}}}^{(\alpha)} = (1, \ldots, 1)^T$
- rescale weights: $\mathrm{W}_{\alpha\beta_j^{(\alpha)}} = \widetilde{\mathrm{w}}_j^{(\alpha)} / \sum_{k=1}^{K} \widetilde{\mathrm{w}}_k^{(\alpha)}$ to fulfill constraint

$\Rightarrow \underline{\mathbf{W}}$ contains the optimal weights with $\mathrm{W}_{\alpha\beta} = 0$ for $\beta \notin \mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)})$
$\Rightarrow p$ dense K-dim. linear systems have to be solved: $\mathcal{O}(pK^3)$

Step 3: obtain embedding coordinates For any M-dimensional manifold there exist linear mappings of each local "patch" onto M-dimensional coordinates in a linear space

- linear mapping: rotation, scaling, translation
- weights $\underline{\mathbf{W}}$ can be used to optimally reconstruct the data points in the lower-dimensional embedding space

idea:

- cut N-d manifold into small patches
- "glue" them together in M-d using only rotation, scaling, translation for each patch

given $M \ll N$ and $\underline{\mathbf{W}}$: find optimal coordinates $\underbrace{\underline{\mathbf{u}}^{(1)}, \ldots, \underline{\mathbf{u}}^{(p)}}_{=\underline{\mathbf{U}} \in \mathbb{R}^{M,p}} \in \mathbb{R}^M$

cost function:

$$F(\underline{\mathbf{U}}) = \sum_{\alpha=1}^{p} \left| \underline{\mathbf{u}}^{(\alpha)} - \sum_{\beta=1}^{p} \mathrm{W}_{\alpha\beta} \underline{\mathbf{u}}^{(\beta)} \right|^2$$

equivalent quadratic form:

$$F(\underline{\mathbf{U}}) = \sum_{\alpha,\beta=1}^{p} g_{\alpha\beta} (\underline{\mathbf{u}}^{(\alpha)})^T \underline{\mathbf{u}}^{(\beta)}$$

where $g_{\alpha\beta} =$ **see blackboard**

$$= \delta_{\alpha\beta} - \mathrm{W}_{\alpha\beta} - \mathrm{W}_{\beta\alpha} + \sum_{\gamma=1}^{p} \mathrm{W}_{\gamma\alpha} \mathrm{W}_{\gamma\beta}$$

$\underline{\mathbf{G}} = \left\{ g_{\alpha\beta} \right\} \in \mathbb{R}^{p,p}$ is symmetric and positive semidefinite
minimize cost function:

$$F(\underline{\mathbf{U}}) = \sum_{\alpha,\beta=1}^{p} g_{\alpha\beta} (\underline{\mathbf{u}}^{(\alpha)})^T \underline{\mathbf{u}}^{(\beta)}$$

$$\text{s.t.} \quad \sum_{\alpha=1}^{p} \underline{\mathbf{u}}^{(\alpha)} = 0, \quad \text{(remove translation freedom)}$$

$$\frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{u}}^{(\alpha)} (\underline{\mathbf{u}}^{(\alpha)})^T = \underline{\mathbf{I}} \quad \text{(prevent trivial solutions e.g., } \underline{\mathbf{u}}^{(\alpha)} = \underline{\mathbf{0}})$$

$\rightsquigarrow$ w.l.o.g. as $F(\underline{\mathbf{U}})$ invariant to rotation, scaling, translation

$\rightsquigarrow$ implies that reconstruction errors for different coordinates are measured on the same scale

Step 3: obtain embedding coordinates solution:
compute the $M+1$ eigenvectors of $\underline{\mathbf{G}}$ with the lowest eigenvalues but discard the eigenvector $\underline{\mathbf{e}}_p = \frac{1}{p}(1, \ldots, 1)^T$ with eigenvalue 0 (corresponding to translation)

$$\underline{\mathbf{U}} = \begin{pmatrix} \underline{\mathbf{e}}_{p-M}^T \\ \vdots \\ \underline{\mathbf{e}}_{p-1}^T \end{pmatrix} = \left(\underline{\mathbf{u}}^{(1)}, \ldots, \underline{\mathbf{u}}^{(p)} \right) \quad \text{(solution satisfies both constraints)}$$

implementation:

- store $\underline{\mathbf{W}}$ in sparse matrix format (at most $K \cdot p$ non-zero elements) and calculate $\underline{\mathbf{G}} = \left(\underline{\mathbf{I}} - \underline{\mathbf{W}}^T \right) \left(\underline{\mathbf{I}} - \underline{\mathbf{W}} \right) \in \mathbb{R}^{p,p}$
- use sparse eigenvalue solvers (eigsh)
  $\underline{\mathbf{v}} \mapsto \underline{\mathbf{G}} \cdot \underline{\mathbf{v}} = \left(\underline{\mathbf{I}} - \underline{\mathbf{W}}^T \right) \left[\left(\underline{\mathbf{I}} - \underline{\mathbf{W}} \right) \underline{\mathbf{v}} \right]$

□ Ex: LLE vs PCA for human faces translated in 2D

Summary of the LLE algorithm
parameters: K, M

1. find K nearest neighbors $\mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)}) = \left\{\beta_1^{(\alpha)}, \ldots, \beta_K^{(\alpha)}\right\}\ \forall \alpha = 1, \ldots, p$
2. calculate (locally invariant) reconstruction weights $\underline{\mathbf{W}}$:

$$\underline{\mathbf{C}}^{(\alpha)} \underline{\widetilde{\mathbf{w}}}^{(\alpha)} = (1, \ldots, 1)^T, \quad \forall \alpha = 1, \ldots, p$$
$$\mathrm{W}_{\alpha\beta_j^{(\alpha)}} = \frac{\widetilde{\mathrm{w}}_j^{(\alpha)}}{\sum_{k=1}^{K} \widetilde{\mathrm{w}}_k^{(\alpha)}}$$

3. calculate the embedding coordinates $\underline{\mathbf{U}}$:
   compute the $M+1$ eigenvectors $\left(\underline{\mathbf{e}}_p, \ldots, \underline{\mathbf{e}}_{p-M}\right)$ of $\underline{\mathbf{G}}$
   with the smallest eigenvalues

$$g_{\alpha\beta} = \delta_{\alpha\beta} - \mathrm{W}_{\alpha\beta} - \mathrm{W}_{\beta\alpha} + \sum_{\gamma=1}^{p} \mathrm{W}_{\gamma\alpha}\mathrm{W}_{\gamma\beta}$$

$$\underline{\mathbf{G}} \cdot \underline{\mathbf{e}}_j = \lambda_j \underline{\mathbf{e}}_j \qquad \underline{\mathbf{U}} = \begin{pmatrix} \underline{\mathbf{e}}_{p-M}^T \\ \vdots \\ \underline{\mathbf{e}}_{p-1}^T \end{pmatrix} = \left(\underline{\mathbf{u}}^{(1)}, \ldots, \underline{\mathbf{u}}^{(p)}\right)$$

Remarks

- efficient & robust algorithm
- parameters: number K of neighbors, embedding dimension M
- convex optimization problem, standard (sparse) linear algebra methods suffice
- for $K > N$ regularization is required (singular covariance matrix $\underline{\mathbf{C}}^{(\alpha)}$)

$$\underline{\mathbf{C}}^{(\alpha)} \leftarrow \underline{\mathbf{C}}^{(\alpha)} + \varepsilon \underline{\mathbf{I}}$$

- extension for pairwise data available via non-Euclidean distances $d_{\alpha\alpha'}$ in $\underline{\mathbf{C}}^{(\alpha)}$
- alternative methods available (e.g. Laplacian eigenmaps, t-stochastic neighbor embedding, isomap, Kernel PCA)

5 Probability Density Estimation

This chapter introduces the concept of a probability density and illustrates both *non-parametric* (→ section 5.1) and *parametric* (→ section 5.2) approaches to estimate a density function given limited amounts of data. Good estimators can often be constructed using the *Maximum Likelihood* principle, which is illustrated in section 5.2.3.

Probabilities

discrete random variable:	X
values:	$x \in X = \{x_1, x_2, \ldots, x_k\}$
$P(x)$:	$X \to \mathbb{R}$
positivity and normalization:	$0 \leq P(x) \leq 1$ and $\sum_i P(x_i) = 1$

Probability Density

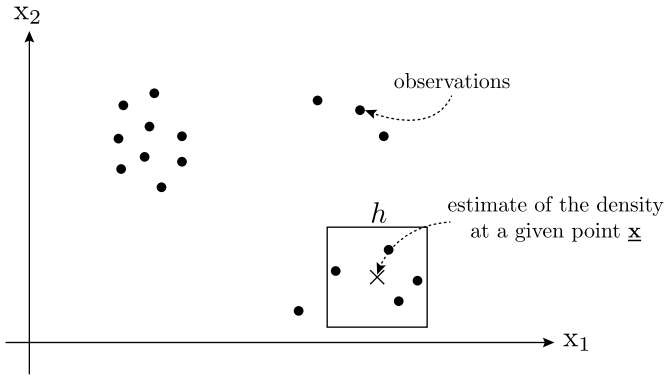
continous random variable:	X
values:	$x \in \mathbb{R}$ (can be generalised to $\mathbb{R}^n, P(\underline{\mathbf{x}})$)
probability density:	$P(x) : \mathbb{R} \to \mathbb{R}_0^+$ $\rightsquigarrow$ non-negative function $\rightsquigarrow$ normalized to $\int_{\text{support(x)}} dx P(x) = 1$ $\rightsquigarrow P(x)$ can be larger than 1 for some x
probability:	$\underbrace{P(x)^{(\text{vol.})} = \int_{\text{vol.}} dx P(x)}_{\underline{\text{p}}\text{robability } \underline{\text{d}}\text{ensity } \underline{\text{f}}\text{unction (pdf)}}$
probability density:	$\frac{\text{probability}}{\text{volume}}$

Density Estimation

Given observations $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \ldots, p$ drawn (iid) from a (generally unknown) distribution, *inductive learning* can be applied to get good estimates for both prior densities $P(x)$ and conditional densities $P(y|x)$.

"good" estimate/model $\widehat{P}(\underline{\mathbf{x}})$ {
- non-parametric methods (e.g. histograms)
- parametric methods: $\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}})$
  $\rightsquigarrow$ estimate a "good" parameter vector $\underline{\mathbf{w}}$

5.1 Kernel Density Estimation



Histogram: count the number of data points in a volume of a given size V centered on $\underline{\mathbf{x}}$. For $u_j = \underline{\mathbf{x}}_j - \underline{\mathbf{x}}_j^{(\alpha)}$

$$V = h^n \qquad \text{volume}$$

$$H(\underline{\mathbf{u}}) = \begin{cases} 1, & |u_j| < \frac{1}{2}, \forall j \in 1, \ldots, n \\ 0, & \text{else} \end{cases} \qquad \begin{array}{c}\text{histogram "kernel"} \\ \text{(here: equals to 1} \\ \text{if } \underline{\mathbf{u}} \text{ is located within} \\ \text{the unit cube)}\end{array} \tag{5.42}$$

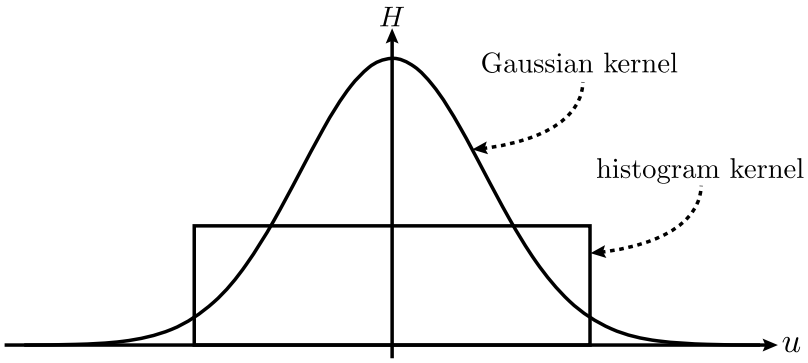
Density estimate ("gliding histogram")

$$\widehat{P}(\underline{\mathbf{x}}) = \underbrace{\frac{1}{h^n}}_{\substack{\text{normalization} \\ \text{("density"!)}}} \cdot \underbrace{\frac{1}{p} \overbrace{\sum_{\alpha=1}^{p} H\left(\frac{\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}}{h}\right)}^{\substack{\text{number of data points} \\ \text{within volume } V \text{ around } \underline{\mathbf{x}}}}}_{\text{fraction of data points}} \tag{5.43}$$

Problem: Histogram kernel leads to discontinous pdf estimates $\rightsquigarrow$ use other kernel than the 'box' to get smooth pdf-estimates.

- $\rightsquigarrow$ weighted sum of data points
- $\rightsquigarrow$ e.g. through a Gaussian kernel

$$H(\underline{\mathbf{u}}) = \frac{1}{(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{\underline{\mathbf{u}}^2}{2}\right) \tag{5.44}$$



Density estimate:

$$\begin{aligned}\widehat{P}(\underline{\mathbf{x}}) &= \frac{1}{h^n} \cdot \frac{1}{p} \sum_{\alpha=1}^{p} H\Big(\frac{\underline{\mathbf{x}}-\underline{\mathbf{x}}^{(\alpha)}}{h}\Big) \\ &= \frac{1}{p} \sum_{\alpha=1}^{p} \frac{1}{\big(2\pi h^2\big)^{\frac{n}{2}}} \exp\left\{ - \frac{\big(\underline{\mathbf{x}}-\underline{\mathbf{x}}^{(\alpha)}\big)^2}{2h^2} \right\}\end{aligned} \tag{5.45}$$

h: hyperparameter $\rightsquigarrow$ determines smoothness

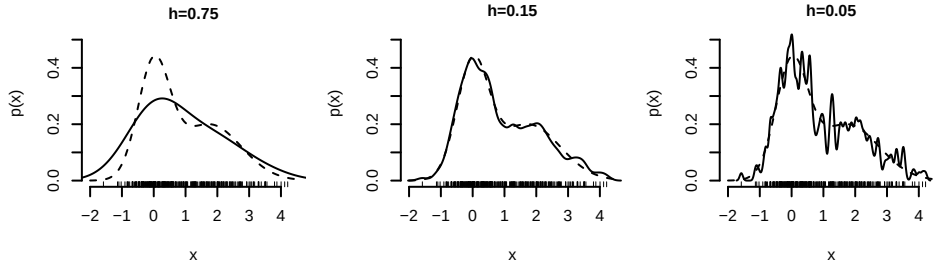


Figure 32: Impact of kernel-width on density estimate.

optimal choice of h? $\rightarrow$ see section 5.2.2

5.2 Parametric Density Estimation

Given a set of observations $\big\{\underline{\mathbf{x}}^{(\alpha)}\big\}, \alpha = 1, \ldots, p$, we assume the data has been produced by a specific *generative model*, e.g. a parameterized family of pdfs: $\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}})$

generative model of a data source $\longrightarrow$ observations

description of the generation process

Comment:

MI 2: models $\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}})$ for unconditional densities $P(\underline{\mathbf{x}})$ $\leftarrow$ unsupervised learning

MI I: models $\widehat{P}(y|\underline{\mathbf{x}}; \underline{\mathbf{w}})$ for conditional densities $P(y|\underline{\mathbf{x}})$ $\leftarrow$ supervised learning

5.2.1 Model Selection and Cost function

Select the model which is most similar to the true density!

Kullback-Leibler-Divergence

$$D_{KL} = \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln \frac{P(\underline{\mathbf{x}})}{\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}})} \tag{5.46}$$

$\rightarrow$ distance measure between probability distributions

$D_{KL} \geq 0$ and $D_{KL} = 0$ iff $\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}}) = P(\underline{\mathbf{x}})$

Criterion for model selection:

$$D_{KL} \stackrel{!}{=} \min_{(\underline{\mathbf{w}})} \tag{5.47}$$

$$\begin{aligned} \underline{\mathbf{w}}^* &= \text{argmin}_{\underline{\mathbf{w}}} \left\{ \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln P(\underline{\mathbf{x}}) - \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln \widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}}) \right\} \\ &= \text{argmin}_{\underline{\mathbf{w}}} \Big\{ \underbrace{- \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln \widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}})}_{E^G_{[\underline{\mathbf{w}}]}} \Big\} \qquad \text{"cross entropy"} \end{aligned} \tag{5.48}$$

$$E^G \stackrel{!}{=} \min_{(\underline{\mathbf{w}})} \tag{5.49}$$

problem: $P(\underline{\mathbf{x}})$ is unknown.

5.2.2 Principle of empirical risk minimization (ERM)

$$\boxed{\begin{array}{c}\text{mathematical} \\ \text{expectation } E^G\end{array}} \longrightarrow \boxed{\begin{array}{c}\text{empirical} \\ \text{average } E^T\end{array}}$$

"generalization cost" "training cost"

cost function:

$$E^T = -\frac{1}{p} \sum_{\alpha=1}^{p} \ln \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big) \tag{5.50}$$

but when is this a reasonable procedure? $\rightsquigarrow$ statistical learning theory
(cf. MI I, section 2.1)

criterion for model selection:

$$\boxed{E^T = -\tfrac{1}{p} \sum_{\alpha=1}^{p} \ln \widehat{P}\big((\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big) \stackrel{!}{=} \min_{(\underline{\mathbf{w}})}} \tag{5.51}$$

Optimization

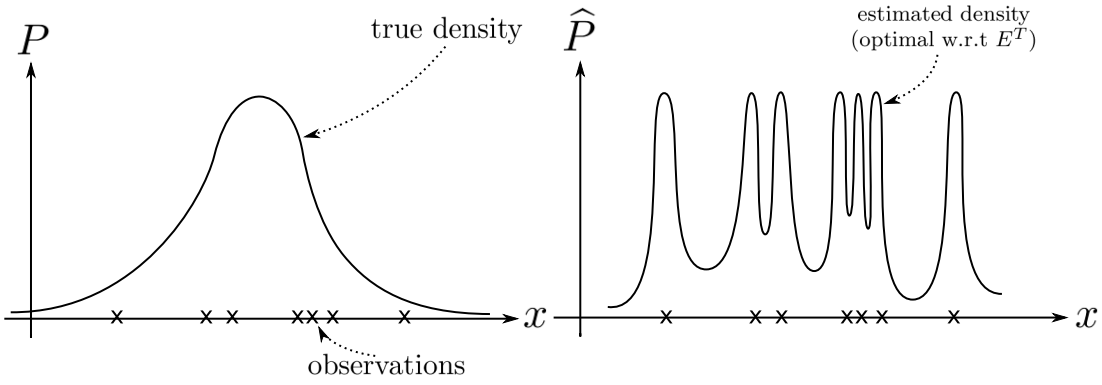
$$\underbrace{E^T_{[\underline{\mathbf{w}}]}}_{\substack{\text{total}\\ \text{cost}}} = \frac{1}{p}\sum_{\alpha=1}^{p} \underbrace{e^{(\alpha)}_{[\underline{\mathbf{w}}]}}_{\substack{\text{individual}\\ \text{cost}}} \tag{5.52}$$

standard gradient-descent procedures *(cf. MI I, sections 1.3.4 and 1.4.1-3)*

$$\left.\begin{array}{ll} \text{"batch"-learning:} & \Delta\underline{\mathbf{w}} = -\eta\frac{\partial E^T}{\partial \underline{\mathbf{w}}} \\ \text{"on-line"-learning:} & \Delta\underline{\mathbf{w}} = -\eta\frac{\partial e^{(\alpha)}}{\partial \underline{\mathbf{w}}} \end{array}\right\} \begin{array}{l}\text{examples for}\\ \text{gradient-based}\\ \text{methods}\end{array} \tag{5.53}$$

Validation

Motivation: $E^T_{[\underline{\mathbf{w}}]} \stackrel{!}{=} \min$ instead of $E^G_{[\underline{\mathbf{w}}]} \stackrel{!}{=} \min$ may lead to overfitting.

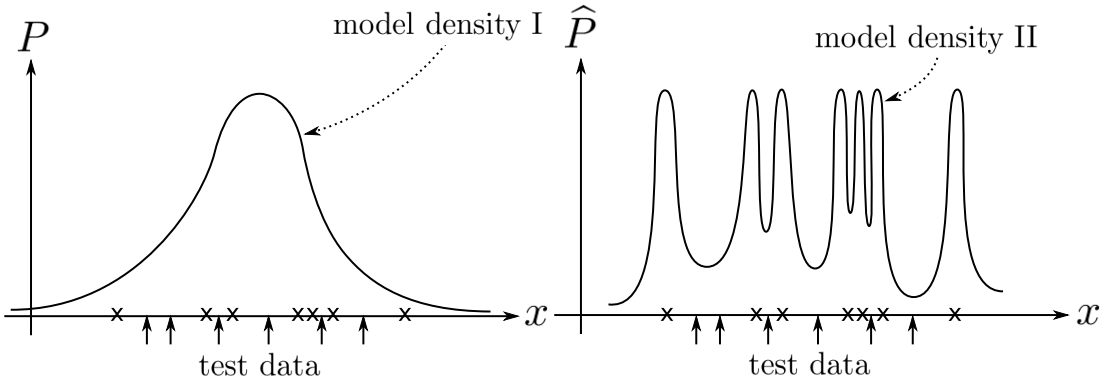


E^T small but E^G large may indicate overfitting

Testset method:

$$\text{observations}\left\{\begin{array}{ll} \text{training data} & \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \alpha = 1,\ldots,p \\ \text{test data} & \left\{\underline{\mathbf{x}}^{(\beta)}\right\}, \beta = 1,\ldots,q \end{array}\right.$$

$$\widehat{E}^G = \frac{1}{q}\sum_{\beta=1}^{q} e^{(\beta)} \leftarrow \text{ estimate of } E^G \tag{5.54}$$



$\Rightarrow\ E^T_{(I)} > E^T_{(II)}$ but $E^G_{(I)} << E^G_{(II)}$

alternative to the "test-set-method": n-fold cross-validation *(cf. MI I, section 1.3.7)*

Comment: Validation methods can also be used to estimate hyperparameters for non-parametric methods (i.e. kernel density estimate).

5.2.3 The Principle of Maximum Likelihood

generative model

$$\widehat{P}(\underline{\mathbf{x}};\underline{\mathbf{w}}) \quad \begin{matrix}\text{probability density for the}\\ \text{generation of one data point}\end{matrix} \tag{5.55}$$

likelihood of the observations (iid assumption)

$$\widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\};\underline{\mathbf{w}}\big) = \prod_{\alpha=1}^{p} \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\big) \quad \begin{matrix}\text{probability density for the}\\ \text{generation of the whole}\\ \text{observed data set}\end{matrix} \tag{5.56}$$

model selection via the maximum likelihood principle

$$\widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\};\underline{\mathbf{w}}\big) \stackrel{!}{=} \max_{(\underline{\mathbf{w}})} \tag{5.57}$$

Idea: pick the model under which the probability of observing the data is maximal

In practice: minimization of the negative log-likelihood

$$\begin{aligned} p \cdot E^T_{[\underline{\mathbf{w}}]} &= -\ln \widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\};\underline{\mathbf{w}}\big) \\ &= -\sum_{\alpha=1}^{p} \ln \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\big) \\ &\stackrel{!}{=} \min \end{aligned} \tag{5.58}$$

$\Rightarrow$ fully equivalent to the minimization of the KL-divergence via ERM.

The multivariate Gaussian

$$\widehat{P}\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\};\underline{\boldsymbol{\mu}},\underline{\boldsymbol{\Sigma}}\right)=\left(\frac{1}{\sqrt{(2\pi)^N\det\underline{\boldsymbol{\Sigma}}}}\right)^p\cdot\prod_{\alpha=1}^{p}\exp\left(-\frac{1}{2}\left(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}\right)^T\underline{\boldsymbol{\Sigma}}^{-1}\left(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}\right)\right)$$

$$\begin{aligned}E^T\left(\underline{\boldsymbol{\mu}},\underline{\boldsymbol{\Sigma}}\right)&=-\ln\widehat{P}\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\};\underline{\boldsymbol{\mu}},\underline{\boldsymbol{\Sigma}}\right)\\&=\frac{p\cdot N}{2}\ln(2\pi)+\frac{p}{2}\ln(\det\underline{\boldsymbol{\Sigma}})+\frac{1}{2}\sum_{\alpha=1}^{p}\left(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}\right)^T\underline{\boldsymbol{\Sigma}}^{-1}\left(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}\right)\end{aligned}$$

minimization of E^T (necessary conditions):

$$\frac{\partial E^T}{\partial\underline{\boldsymbol{\mu}}}=\underline{\mathbf{0}}\quad\Rightarrow\quad\underline{\boldsymbol{\mu}}^*=\frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)}\qquad\text{(empirical average)}$$

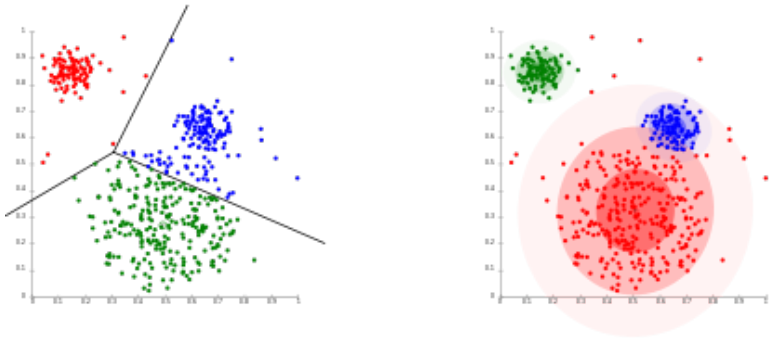
$$\frac{\partial E^T}{\partial\underline{\boldsymbol{\Sigma}}}=\underline{\mathbf{0}}\quad\Rightarrow\quad\underline{\boldsymbol{\Sigma}}^*=\frac{1}{p}\sum_{\alpha=1}^{p}(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}^*)(\underline{\mathbf{x}}^{(\alpha)}-\underline{\boldsymbol{\mu}}^*)^T\qquad\text{(empirical covariance matrix)}$$

remark: $\underline{\boldsymbol{\mu}}^*$ is unbiased, but $\underline{\boldsymbol{\Sigma}}^*$ is a biased estimator (cf. section: 6.1 Model fitting)

6 Mixture Models and the EM-Algorithm

6.1 Mixture Models

6.1.1 Motivation



(a) K-Means Clustering (b) Mixture of Gaussians

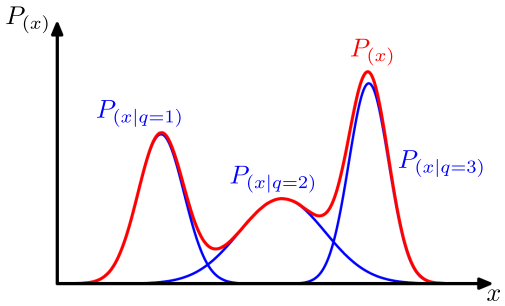
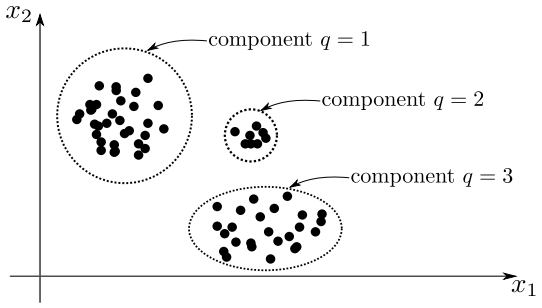


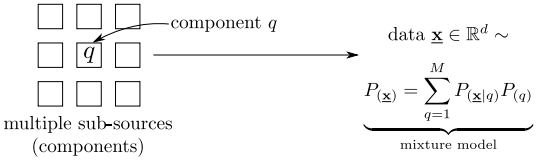
Figure 34: Component-based modeling of complex densities(**Bishop2006**)

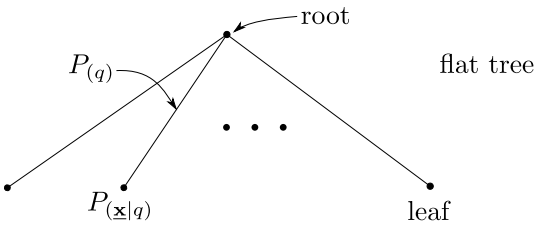
Data sources & representation
Data source → data $\underline{\mathbf{x}} \in \mathbb{R}^N \sim P_{(\underline{\mathbf{x}})}$
⇒ Assumption: Data is generated by multiple sources / classes as represented by



components in figure **??**

Model class
$P_{(\underline{\mathbf{x}}|q)}$: components: probability density, that data point $\underline{\mathbf{x}}$ was created by component q
$P_{(q)}$: mixture parameters: probability, that component q creates a data point





$\rightarrow$ deeper trees possible: hierarchical mixture-models
$\rightarrow$ neural networks at the leaves: mixture of experts

Choice of basis functions

$$P_{(\underline{\mathbf{x}})} = \sum_{q=1}^{M} P_{(q)} P_{(\underline{\mathbf{x}}|q)} \tag{6.59}$$

$$P_{(\underline{\mathbf{x}}|q)} = \mathcal{N}\left(\underline{\mathbf{x}}; \underline{\mathbf{w}}_q, \sigma_q^2\right) = \frac{1}{(2\pi\sigma_q^2)^{N/2}} \exp\left\{-\frac{(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q)^2}{2\sigma_q^2}\right\} \tag{6.60}$$

$$\rightsquigarrow \text{ (Gaussian mixture model)} \tag{6.61}$$

parameters $P_{(q)}$, $\underline{\mathbf{w}}_q$ and σ_q must be determined for all components q

different basis functions are possible (problem specific)

Performance measure Probability, that the dataset $\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}$ was generated by the model:

$$P_{\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}\right)} = \prod_{\alpha=1}^{p} P_{(\underline{\mathbf{x}}^{(\alpha)})} = \prod_{\alpha=1}^{p} \left\{\sum_{q=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)}\right\} \tag{6.62}$$

$$= \prod_{\alpha=1}^{p} \left\{\sum_{q=1}^{M} \mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}_q, \sigma_q^2\right) P_{(q)}\right\} \tag{6.63}$$

Principle of maximum likelihood:

$$P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\}\right)} \stackrel{!}{=} \max \quad \text{w.r.t. parameters} \tag{6.64}$$

Minimization of negative log-likelihood instead:

$$E^T = -\ln P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\}\right)} = -\sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)} \stackrel{!}{=} \min \quad \text{w.r.t parameters} \tag{6.65}$$

Relation to Soft-Clustering methods Assumptions:

- Gaussian mixture model with M components
- same widths $\sigma_q^2 = \sigma^2 := \underbrace{\frac{1}{\beta}}_{\text{given}}$ for all basis functions
- same mixture parameters $P_{(q)} = \frac{1}{M}$

Cost function:

$$P_{(\underline{\mathbf{x}}^{(\alpha)})} = \sum_{q=1}^{M} \mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}_q, \sigma^2\right) P_{(q)} = \frac{1}{M} \left(\frac{\beta}{2\pi}\right)^{N/2} \sum_{q=1}^{M} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right)^2\right\} \tag{6.66}$$

$$E^T = -\ln P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\}\right)} = -\ln \prod_{\alpha=1}^{p} \left\{\sum_{q=1}^{M} \mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}_q, \sigma^2\right) P_{(q)}\right\} \tag{6.67}$$

$$= -\sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2\right\} + \text{const}_{(\underline{\mathbf{w}}_q)} \tag{6.68}$$

Assignment probabilities:
$P_{(q|\underline{\mathbf{x}})}$: posterior probability of component q having generated a given data point $\underline{\mathbf{x}}$

$$P_{(q|\underline{\mathbf{x}})} = \frac{P_{(\underline{\mathbf{x}}|q)} P_{(q)}}{P_{(\underline{\mathbf{x}})}} \qquad \text{(Bayes' theorem)} \tag{6.69}$$

$\rightsquigarrow$ given the simplified Gaussian mixture model we obtain:

$$P_{(q|\underline{\mathbf{x}})} = \frac{\left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2\right\} \cdot \frac{1}{M}}{\left(\frac{\beta}{2\pi}\right)^{N/2} \frac{1}{M} \sum_\gamma \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_\gamma\right)^2\right\}} \tag{6.70}$$

$$= \frac{\exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2\right\}}{\sum_{\gamma=1}^{M} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_\gamma\right)^2\right\}} \tag{6.71}$$

$\Rightarrow$ assignment probability for Soft-Clustering

$$E^T = -\sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2\right\} + \text{const}_{(\underline{\mathbf{w}}_q)} \tag{6.72}$$

Minimization of the cost function w.r.t. the weights:

$$\frac{\partial E^T}{\partial \underline{\mathbf{w}}_r} = -\sum_{\alpha=1}^{p} \frac{\exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_r\right)^2\right\}}{\sum_{q=1}^{M} \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2\right\}} \beta\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_r\right) \stackrel{!}{=} 0 \tag{6.73}$$

$$\underline{\mathbf{w}}_r = \frac{\sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})} \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})}} \quad \rightsquigarrow \text{ center of mass condition for Soft-Clustering!} \tag{6.74}$$

$\Rightarrow$ Gaussian mixture model with components of equal size and strength is equivalent to Soft-Clustering

New interpretation of Soft-Clustering:

- parameter estimation for a Gaussian mixture model with components of equal widths and strengths
- β is given
- implicit assumption: every cluster contains the same number of data points

Mixture models can be viewed as an extension of Soft-Clustering methods:

- clusters with different widths
- clusters with different number of data points

Optimization for Gaussian mixtures

Supporting equations:

$$\frac{\partial}{\partial \underline{\mathbf{w}}_q} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} = \frac{\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)}{\sigma_q^2} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} \qquad \frac{\partial}{\partial \sigma_q} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} = \left\{-\frac{N}{\sigma_q} + \frac{\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2}{\sigma_q^3}\right\} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)}$$

Cost function: $E^T = -\ln P_{\{\underline{\mathbf{x}}^{(\alpha)}\}} = -\sum_{\alpha=1}^{p} \ln\left(\sum_{q=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)}\right) \stackrel{!}{=} \min$

Minimization w.r.t. weights:

$$\frac{\partial E^T}{\partial \underline{\mathbf{w}}_r} = -\sum_{\alpha=1}^{p} \frac{\frac{\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r\right)}{\sigma_r^2} P_{(\underline{\mathbf{x}}^{(\alpha)}|r)} P_{(r)}}{\sum_{q=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)}}$$

$$= \sum_{\alpha=1}^{p} \frac{\left(\underline{\mathbf{w}}_r - \underline{\mathbf{x}}^{(\alpha)}\right)}{\sigma_r^2} P_{(r|\underline{\mathbf{x}}^{(\alpha)})} \stackrel{!}{=} 0$$

$$\boxed{\underline{\mathbf{w}}_r = \frac{\sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})} \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})}}}$$

$\Rightarrow$ mean of the data $\underline{\mathbf{x}}$ assigned to cluster r

Minimization w.r.t. width of components:

$$\frac{\partial E^T}{\partial \sigma_r} = -\sum_{\alpha=1}^{p} \frac{\left\{ -\frac{N}{\sigma_r} + \frac{\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r\right)^2}{\sigma_r^3} \right\} P_{(\underline{\mathbf{x}}^{(\alpha)}|r)} P_{(r)}}{\sum_{q=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)}}$$

$$= \frac{1}{\sigma_r} \sum_{\alpha=1}^{p} \left\{ N - \frac{\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r\right)^2}{\sigma_r^2} \right\} P_{(r|\underline{\mathbf{x}}^{(\alpha)})} \stackrel{!}{=} 0$$

$$\boxed{\sigma_r^2 = \frac{1}{N} \frac{\sum_{\alpha=1}^{p} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r\right)^2 P_{(r|\underline{\mathbf{x}}^{(\alpha)})}}{\sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})}}}$$

$\Rightarrow$ width of cluster: variance of data

Minimization w.r.t. mixture parameters using Lagrange multipliers:

$$\frac{\partial}{\partial P_{(r)}} \left\{ E^T + \lambda \left(\sum_{q=1}^{M} P_{(q)} - 1 \right) \right\} \stackrel{!}{=} 0$$

$$= -\sum_{\alpha=1}^{p} \frac{P_{(\underline{\mathbf{x}}^{(\alpha)}|r)}}{\sum_{q=1}^{M} \underbrace{P_{(\underline{\mathbf{x}}^{(\alpha)}|q)} P_{(q)}}_{P_{(\underline{\mathbf{x}}^{(\alpha)})}}} + \lambda \stackrel{\frac{P_{(\underline{\mathbf{x}}|r)}}{P_{(\underline{\mathbf{x}})}} = \frac{P_{(r|\underline{\mathbf{x}})}}{P_{(r)}}}{=} -\sum_{\alpha=1}^{p} \frac{P_{(r|\underline{\mathbf{x}}^{(\alpha)})}}{P_{(r)}} + \lambda \stackrel{!}{=} 0$$

$P_{(r)} = \frac{1}{\lambda} \sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})}$, from $\sum_{r=1}^{M} P_{(r)} = p \stackrel{!}{=} 1$ follows $\lambda = p$ and

$$\boxed{P_{(r)} = \frac{1}{p} \sum_{\alpha=1}^{p} P_{(r|\underline{\mathbf{x}}^{(\alpha)})}}$$

$\Rightarrow$ "number" of data points per cluster (weighted by probability)

Algorithm 16: Fixed-point iteration (Expectation-Maximization-algorithm)

initialization: $P_{(q)}^{\text{old}} = \frac{1}{M}, \quad \underline{\boldsymbol{\mu}} = \frac{1}{p}\sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)}, \quad \underline{\mathbf{w}}_q^{\text{old}} = \underline{\boldsymbol{\mu}} + \underline{\eta}_q,$

$$(\sigma_q^2)^{\text{old}} = \frac{1}{p}\sum_{\alpha=1}^{p}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\boldsymbol{\mu}}\right)^2 + \varepsilon_q, \quad \underline{\eta}_q, \underline{\varepsilon}_q\text{: small random vectors}$$

repeat

1. E-Step: Calculation of the assignment probabilities for $q = 1, \ldots, M$

$$P_{(q|\underline{\mathbf{x}}^{(\alpha)})} \overset{\text{Bayes}}{=} \frac{P_{(\underline{\mathbf{x}}^{(\alpha)}|q)}^{\text{old}} P_{(q)}^{\text{old}}}{\sum_{r=1}^{M} P_{(\underline{\mathbf{x}}^{(\alpha)}|r)}^{\text{old}} P_{(r)}^{\text{old}}} \qquad P_{(\underline{\mathbf{x}}^{(\alpha)}|q)}^{\text{old}} = \mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}_q^{\text{old}}, (\sigma_q^2)^{\text{old}}\right)$$

2. M-Step: Calculation of the new parameter values for $q = 1, \ldots, M$

$$\underline{\mathbf{w}}_q^{\text{new}} = \frac{\sum_{\alpha=1}^{p} P_{(q|\underline{\mathbf{x}}^{(\alpha)})} \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha=1}^{p} P_{(q|\underline{\mathbf{x}}^{(\alpha)})}}$$

$$\left(\sigma_q^2\right)^{\text{new}} = \frac{1}{N} \frac{\sum_{\alpha=1}^{p}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}}\right)^2 P_{(q|\underline{\mathbf{x}}^{(\alpha)})}}{\sum_{\alpha=1}^{p} P_{(q|\underline{\mathbf{x}}^{(\alpha)})}}$$

$$P_{(q)}^{\text{new}} = \frac{1}{p}\sum_{\alpha=1}^{p} P_{(q|\underline{\mathbf{x}}^{(\alpha)})}$$

until *parameter values converge*

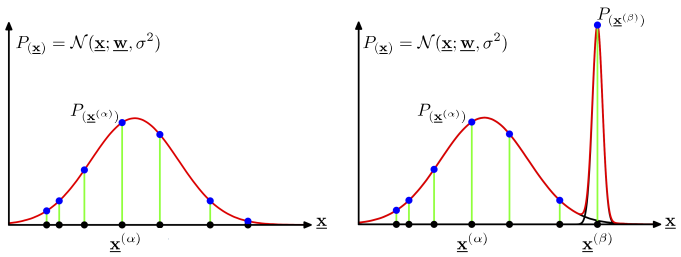


Figure 35: Degenerated solutions

Degenerated solutions: "collapse" of components

$$\mathcal{N}(\underline{\mathbf{x}}^{(\beta)}; \underline{\mathbf{w}}_q, \sigma_q^2) = \mathcal{N}(\underline{\mathbf{x}}^{(\beta)}; \underline{\mathbf{x}}^{(\beta)}, \sigma_q^2)$$

$$= \frac{1}{\left(2\pi\sigma_q^2\right)^{1/2}} \exp\left\{-\frac{\left(\underline{\mathbf{x}}^{(\beta)} - \underline{\mathbf{x}}^{(\beta)}\right)^2}{2\sigma_q^2}\right\} = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_q} \overset{\sigma_q^2 \to 0}{\longrightarrow} \infty$$

$\rightsquigarrow$ model validation (testset method) to detect overfitting

$\rightsquigarrow$ Maximum-a-posteriori instead of maximum likelihood approaches using a prior for each component which penalizes components with small variance σ_q^2

Hierarchical Gaussian mixtures

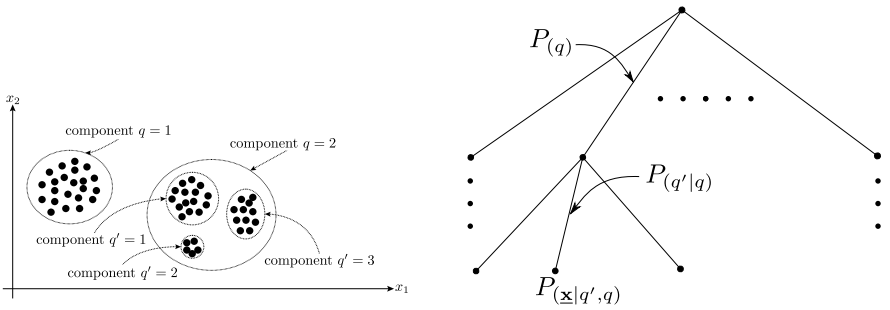


Figure 36: Hierarchical Gaussian mixtures

$$P_{(\underline{\mathbf{x}})} = \sum_q P_{(q)} \sum_{q'} P_{(q'|q)} P_{(\underline{\mathbf{x}}|q',q)}$$

Summary

Gaussian mixture model:

$$P_{(\underline{\mathbf{x}})} = \sum_{q=1}^{M} P_{(q)} P_{(\underline{\mathbf{x}}|q)}$$

$$P_{(\underline{\mathbf{x}}|q)} = \frac{1}{\left(2\pi\sigma_q^2\right)^{N/2}} \exp\left\{ -\frac{\left(\underline{\mathbf{x}} - \underline{\mathbf{w}}_q\right)^2}{2\sigma_q^2} \right\}$$

Maximum likelihood:

$$P_{(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}|\text{ parameter})} \stackrel{!}{=} \max$$

Relation to Soft-Clustering:

- cost functions are identical, if: $\sigma_q^2 = \text{const.}_{(q)} = \frac{1}{\beta}$ and $P_{(q)} = \text{const.}_{(q)} = \frac{1}{M}$
- new interpretation of Soft-Clustering:
  - estimation of parameter $\underline{\mathbf{w}}_q$ of a Gaussian mixture model
  - β defines the size of the cluster $\widehat{=}$ resolution

Remarks

- equivalent solutions (permutation of components)
- improved initialization by application of the (much faster) K-means method:
  - prototypes $\rightsquigarrow$ component means $\underline{\mathbf{w}}_q$
  - intra-cluster spreads $\rightsquigarrow$ component variances σ_q^2

- extension to general Gaussian components ($\sigma_q^2 \to \underline{\mathbf{\Sigma}}_q$) straightforward (cf. Bishop 2006)
- mixture model: example of latent variable model

6.2 The Expectation-Maximization Algorithm

Latent variables

Example: mixture model

- observed data set $\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)} \in \mathbb{R}^N$
- every data point $\underline{\mathbf{x}}^{(\alpha)}$ is generated by one component $q = 1, \ldots, M$

  $\rightsquigarrow$ assignment variables: $\underline{\mathbf{m}}^{(\alpha)} = \left(m_1^{(\alpha)}, \ldots, m_M^{(\alpha)}\right)^T \in \{0,1\}^M$

  $$m_q^{(\alpha)} = \begin{cases} 1, & \text{if component } q \text{ has generated data point} \\ 0, & \text{otherwise} \end{cases} \qquad \sum_{q=1}^{M} m_q^{(\alpha)} = 1$$

- complete data set: $\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{m}}^{(1)} \ldots, \underline{\mathbf{x}}^{(p)}, \underline{\mathbf{m}}^{(p)}$
- hidden / latent variables: $\underline{\mathbf{m}}^{(1)}, \ldots, \underline{\mathbf{m}}^{(p)}$

Latent variable models and maximum likelihood

Calculation of the likelihood of the observed data
requires marginalization of $P(\underline{\mathbf{x}}, \underline{\mathbf{m}}|\underline{\mathbf{w}})$:

$$P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}|\underline{\mathbf{w}}\right) \overset{iid}{=} \prod_{\alpha=1}^{p} P\left(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}\right) = \prod_{\alpha=1}^{p} \sum_{\underline{\mathbf{m}}} P\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{m}}|\underline{\mathbf{w}}\right)$$

Log-likelihood is computationally costly to maximize / no closed-form solution due to sum in logarithm:

$$\ln P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}|\underline{\mathbf{w}}\right) = \sum_{\alpha=1}^{p} \ln\left(\sum_{\underline{\mathbf{m}}} P\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{m}}|\underline{\mathbf{w}}\right)\right)$$

The Expectation-Maximization (EM) algorithm

Maximize the joint distribution over observed and latent variables (specifically useful if $P(\underline{\mathbf{x}}, \underline{\mathbf{m}}|\underline{\mathbf{w}})$ is from the exponential family: Gaussian, Bernoulli etc.)

$$\ln P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \underline{\mathbf{w}}\right) \overset{!}{=} \max_{(\underline{\mathbf{w}})}$$

Problem: values of the hidden variables are unknown.

The Expectation-Maximization (EM) algorithm

choose initial values for the parameters $\underline{\mathbf{w}}_{\text{old}}$ (e.g., by random) and tolerance θ

repeat

1. Evaluation of posterior distribution: $P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \mid \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \underline{\mathbf{w}}_{\text{old}}\right)$
2. E-Step: Compute expectation of complete data log-likelihood w.r.t posterior of $\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}$

$$\mathcal{Q}(\underline{\mathbf{w}}, \underline{\mathbf{w}}_{\text{old}}) = \sum_{\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}} P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \underline{\mathbf{w}}_{\text{old}}\right) \ln P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \underline{\mathbf{w}}\right)$$

3. M-Step: Determine new parameters that maximize the expectation

$$\underline{\mathbf{w}}_{\text{new}} = \arg\max_{(\underline{\mathbf{w}})} \mathcal{Q}(\underline{\mathbf{w}}, \underline{\mathbf{w}}_{\text{old}})$$

$$\underline{\mathbf{w}}_{\text{old}} \leftarrow \underline{\mathbf{w}}_{\text{new}}$$

until $|\underline{\mathbf{w}}_{old} - \underline{\mathbf{w}}_{new}| < \theta$

Remarks

- The EM algorithm converges to a local maximum of the log-likelihood function (cf. Bishop 2006)
- local optima (e.g., multimodal likelihood function) $\rightsquigarrow$ different initial conditions or simulated annealing methods
- EM is applicable to many latent variable problems: e.g., hidden Markov models, missing data situations
- EM is particularly efficient if the complete data distribution is from exponential family (log of exp)
- further extensions:
  - continuous latent variables (replace sums by integrals in marginalization / expectation)
  - maximum a posteriori estimation using a prior distribution $P_0(\underline{\mathbf{w}})$
  - non-tractable E- or M-steps: approximate inference or generalized EM-algorithms

Gaussian mixtures revisited

$$P(\underline{\mathbf{x}}) = \sum_{q=1}^{M} \rho(q) \mathcal{N}(\underline{\mathbf{x}} | \underline{\mathbf{w}}_q, \sigma_q^2) \stackrel{!}{=} \sum_{\underline{\mathbf{m}}} P(\underline{\mathbf{x}}, \underline{\mathbf{m}}) = \sum_{\underline{\mathbf{m}}} P(\underline{\mathbf{m}}) P(\underline{\mathbf{x}} | \underline{\mathbf{m}})$$

mixture parameters: $\rho(q), \quad 0 \leq \rho(q) \leq 1, \sum_{q=1}^{M} \rho(q) = 1$
prior distribution of latent variables

$$P(\underline{\mathbf{m}}) = \prod_{q=1}^{M} \rho(q)^{m_q}$$

conditional distribution of the observed variables given the latent variables

$$P(\underline{\mathbf{x}}|\underline{\mathbf{m}}) = \prod_{q=1}^{M} \mathcal{N}^{m_q}(\underline{\mathbf{x}}|\underline{\mathbf{w}}_q, \sigma_q^2)$$

joint distribution

$$P(\underline{\mathbf{x}}, \underline{\mathbf{m}}) = P(\underline{\mathbf{x}}|\underline{\mathbf{m}}) \cdot P(\underline{\mathbf{m}}) = \prod_{q=1}^{M} \rho(q)^{m_q} \mathcal{N}^{m_q}(\underline{\mathbf{x}}|\underline{\mathbf{w}}_q, \sigma_q^2)$$

Gaussian mixtures & the EM algorithm

joint distribution: $P(\underline{\mathbf{x}}, \underline{\mathbf{m}}) = \prod_{q=1}^{M} \rho(q)^{m_q} \mathcal{N}^{m_q}(\underline{\mathbf{x}}|\underline{\mathbf{w}}_q, \sigma_q^2)$

likelihood:

$$P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}\right) = \prod_{\alpha=1}^{p} \prod_{q=1}^{M} \rho(q)^{m_q^{(\alpha)}} \mathcal{N}^{m_q^{(\alpha)}}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_q, \sigma_q^2)$$

log-likelihood:

$$\ln P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}\right) = \sum_{\alpha=1}^{p} \sum_{q=1}^{M} m_q^{(\alpha)} \left(\ln \rho(q) + \ln \mathcal{N}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_q, \sigma_q^2)\right)$$

log within sum & log of normal: much easier to handle posterior distribution:

$$P(\underline{\mathbf{m}}|\underline{\mathbf{x}}) = \frac{P(\underline{\mathbf{x}}, \underline{\mathbf{m}})}{P(\underline{\mathbf{x}})} = \frac{\prod_{q=1}^{M} \left[\rho(q)\mathcal{N}(\underline{\mathbf{x}}|\underline{\mathbf{w}}_q, \sigma_q^2)\right]^{m_q}}{\sum_{q=1}^{M} \rho(q)\mathcal{N}(\underline{\mathbf{x}}|\underline{\mathbf{w}}_q, \sigma_q^2)}$$

posterior distribution of hidden data given observed:

$$\begin{aligned} & P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}\right) \\ & \overset{\text{iid. data}}{=} \prod_{\alpha=1}^{p} \frac{P\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{m}}^{(\alpha)} \middle| \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}\right)}{P\left(\underline{\mathbf{x}}^{(\alpha)} \middle| \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}\right)} \\ & = \prod_{\alpha=1}^{p} \frac{\prod_{q=1}^{M} \left[\rho(q)\mathcal{N}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_q, \sigma_q^2)\right]^{m_q^{(\alpha)}}}{\sum_{q=1}^{M} \rho(q)\mathcal{N}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_q, \sigma_q^2)} \end{aligned}$$

expected value under posterior:

$$\begin{aligned}\langle m_q^{(\alpha)} \rangle_{P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}\middle|\left\{\underline{\mathbf{x}}^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}\right)} &= \textbf{see blackboard} \\ &= \frac{\rho(q)\mathcal{N}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_q,\sigma_q^2)}{\sum_{r=1}^{M}\rho(r)\mathcal{N}(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_r,\sigma_r^2)} \\ &= \rho(q|\underline{\mathbf{x}}^{(\alpha)}) \text{ (from mixture EM-Algorithm)}\end{aligned}$$

using this we can evaluate

$$\begin{aligned}&\mathcal{Q}\left(\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right) \\ &= \left\langle \ln P\left(\left\{\underline{\mathbf{x}}^{(\alpha)}\right\},\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}\middle|\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}\right)\right\rangle_{P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}\middle|\left\{\underline{\mathbf{x}}^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right)} \\ &= \left\langle \sum_{\alpha=1}^{p}\sum_{q=1}^{M} m_q^{(\alpha)}\left(\ln\rho(q)+\ln\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}\middle|\underline{\mathbf{w}}_q,\sigma_q^2\right)\right)\right\rangle_{P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}\middle|\left\{\underline{\mathbf{x}}^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right)} \\ &= \sum_{\alpha=1}^{p}\sum_{q=1}^{M}\left[\langle m_q^{(\alpha)}\rangle_{P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\}\middle|\left\{\underline{\mathbf{x}}^{(\alpha)}\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right)}\cdot\left(\ln\rho(q)+\ln\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}\middle|\underline{\mathbf{w}}_q,\sigma_q^2\right)\right)\right] \\ &= \sum_{\alpha=1}^{p}\sum_{q=1}^{M}\left[\rho\left(q|\underline{\mathbf{x}}^{(\alpha)}\right)\cdot\left(\ln\rho(q)+\ln\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}\middle|\underline{\mathbf{w}}_q,\sigma_q^2\right)\right)\right]\end{aligned}$$

$\rightsquigarrow$ E-step: calculation of

$$\rho\left(q|\underline{\mathbf{x}}^{(\alpha)}\right) = \frac{\rho(q)_{\text{OLD}}\cdot\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_{q,\text{OLD}},\sigma_{q,\text{OLD}}^2\right)}{\sum_{r=1}^{M}\rho(r)_{\text{OLD}}\cdot\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}|\underline{\mathbf{w}}_{r,\text{OLD}},\sigma_{r,\text{OLD}}^2\right)}$$

calculation of new parameters:

$$\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{new}} = \underset{\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}}{\text{argmax}}\ \mathcal{Q}\left(\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right)$$

$$\begin{aligned}&\mathcal{Q}\left(\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\},\left\{\underline{\mathbf{w}}_q,\sigma_q^2,\rho(q)\right\}_{\text{OLD}}\right) \\ &= \sum_{\alpha=1}^{p}\sum_{q=1}^{M}\left[\rho\left(q|\underline{\mathbf{x}}^{(\alpha)}\right)\cdot\left(\ln\rho(q)+\ln\mathcal{N}\left(\underline{\mathbf{x}}^{(\alpha)}\middle|\underline{\mathbf{w}}_q,\sigma_q^2\right)\right)\right]\end{aligned}$$

$$\left.\begin{array}{lll} \frac{\partial \mathcal{Q}}{\partial \underline{\mathbf{w}}_q} = 0 & \Rightarrow & \underline{\mathbf{w}}_{q,\text{new}} = \frac{\sum_{\alpha=1}^{p} \rho(q|\underline{\mathbf{x}}^{(\alpha)}) \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha=1}^{p} \rho(q|\underline{\mathbf{x}}^{(\alpha)})} \\ \frac{\partial \mathcal{Q}}{\partial \sigma_q^2} = 0 & \Rightarrow & \sigma^2_{q,\text{new}} = \frac{1}{N} \frac{\sum_{\alpha=1}^{p} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{q,\text{OLD}}\right)^2 \rho(q|\underline{\mathbf{x}}^{(\alpha)})}{\sum_{\alpha=1}^{p} \rho(q|\underline{\mathbf{x}}^{(\alpha)})} \\ \frac{\partial \mathcal{Q}}{\partial \rho(q)} = 0 & \Rightarrow & \rho(q)_{\text{new}} = \frac{1}{p} \sum_{\alpha=1}^{p} \rho(q|\underline{\mathbf{x}}^{(\alpha)}) \end{array}\right\} \text{expressions from mixture EM-algorithm recovered}$$

$\rightsquigarrow$ M-step: optimal parameters for given

$$\langle m_q^{(\alpha)} \rangle_{P\left(\left\{\underline{\mathbf{m}}^{(\alpha)}\right\} \middle| \left\{\underline{\mathbf{x}}^{(\alpha)}\right\}, \left\{\underline{\mathbf{w}}_q, \sigma_q^2, \rho(q)\right\}_{\text{OLD}}\right)} = \rho\left(q|\underline{\mathbf{x}}^{(\alpha)}\right)$$

7 Model-fitting

7.1 Maximum Likelihood and Estimation Theory

An estimator $\hat{P}(X)$ is a function that maps from its sample space X (data) to a set of *sample estimates* W
An estimator ...

- is a function of a random variable
- is a random variable
- can be statistically characterized via its moments (mean, variance, ...)
  $\rightsquigarrow$ quality criteria: unbiasedness, efficiency

set of observations: $\{\underline{\mathbf{x}}^{(\alpha)}\}, \alpha = 1, \ldots, p$

true distribution (normalized):

$$P\big(\{\underline{\mathbf{x}}^{(\alpha)}\};\ \underbrace{\underline{\mathbf{w}}^*}_{\substack{\text{true}\\ \text{parameter}\\ \text{value}}}\ \big) \equiv P \tag{7.75}$$

$\rightsquigarrow$ true model is member of the model class

$\rightsquigarrow$ structure of the postulated model has to be valid

model selection $\Rightarrow$ estimation of the "true" values $\underline{\mathbf{w}}^*$ from the observed data

estimator $\widehat{\underline{\mathbf{w}}}$:

$$\widehat{\underline{\mathbf{w}}} = \widehat{\underline{\mathbf{w}}}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}\big) \tag{7.76}$$

$\rightsquigarrow$ procedure for the determination of $\underline{\mathbf{w}}^*$ given the observed data

$\rightsquigarrow$ $\underline{\mathbf{w}}^*$ is a function of $\big(\{\underline{\mathbf{x}}^{(\alpha)}\}\big)$

$\rightsquigarrow$ $\underline{\mathbf{x}}^{(\alpha)}$ are random variables $\rightarrow$ $\widehat{\underline{\mathbf{w}}}$ is a random variable

The Maximum Likelihood estimator

the likelihood function

$$\widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big)$$

the log**-likelihood function**

$$\ln \widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big) = \sum_{\alpha=1}^{p} \ln \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big)$$

the Maximum Likelihood estimator

$$\widehat{\underline{\mathbf{w}}} = \underset{\underline{\mathbf{w}}}{\operatorname{argmax}} \widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big)$$

quality criteria for estimators: (see also MI I, section 1.4.5)

$$\begin{aligned} &\text{bias:} && \underline{\mathbf{b}} = \underbrace{\left\langle \underline{\widehat{\mathbf{w}}} \right\rangle_p}_{\substack{\text{expectation} \\ \text{w.r.t } \underline{\text{true}} \\ \text{distribution}}} - \underline{\mathbf{w}}^* \\ &\text{variance:} && \underline{\boldsymbol{\Sigma}} = \left\langle (\underline{\widehat{\mathbf{w}}} - \underline{\mathbf{w}}^*)(\underline{\widehat{\mathbf{w}}} - \underline{\mathbf{w}}^*)^T \right\rangle_p \end{aligned} \tag{7.77}$$

optimal estimators:

$$\begin{aligned} &\text{no bias:} && \underline{\mathbf{b}} \stackrel{!}{=} 0 && \leftarrow \text{only possible if true model within model class} \\ &\text{minimal variance:} && |\underline{\boldsymbol{\Sigma}}| \stackrel{!}{=} \min && \leftarrow \text{smallest average deviation of } \underline{\widehat{\mathbf{w}}} \text{ from } \underline{\mathbf{w}}^* \end{aligned} \tag{7.78}$$

Example

sample mean

N observations $x^{(\alpha)} = A + \epsilon^{(\alpha)}$ with $\epsilon^{(\alpha)} \sim N(0, \sigma^2)$

Examples for estimators for A:

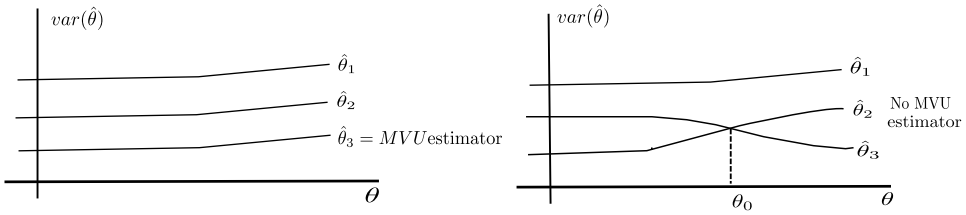
$\hat{A} = \frac{1}{N} \sum x^{(\alpha)}$	unbiased
$\tilde{A} = \frac{1}{2N} \sum x^{(\alpha)}$	biased for$A \neq 0$
$\tilde{A} = k$	minimum variance but biased

The minimum variance unbiased estimator

Optimal estimators:

$$\begin{aligned} &\text{no bias:} && \underline{\mathbf{b}} \stackrel{!}{=} 0 && \leftarrow \text{only possible if true model within model class} \\ &\text{minimal variance:} && |\underline{\boldsymbol{\Sigma}}| \stackrel{!}{=} \min \end{aligned}$$

MVU: criteria have to hold for ALL possible values of $\underline{\mathbf{w}}^*$!



MVUs do not always exist

given just observed sample conditionally independent observations with the 2 pdfs

$$x[0] \sim \mathcal{N}(\theta, 1) \qquad x[1] \sim \begin{cases} \mathcal{N}(\theta, 1) & \text{if } \theta \geq 0 \\ \mathcal{N}(\theta, 2) & \text{if } \theta < 0 \end{cases}$$

two estimators

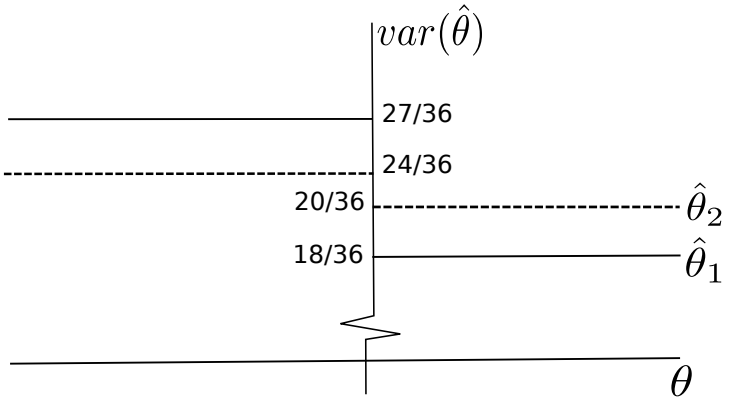
$$\hat{\theta}_1 = \frac{1}{2}(x[0] + x[1]) \qquad \text{and} \qquad \hat{\theta}_2 = \frac{2}{3}x[0] + \frac{1}{3}x[1]$$

variances:

$$var(\hat{\theta}_1) = \frac{1}{4}(var(x[0]) + var(x[1])) \qquad \begin{cases} \frac{18}{36} & \text{if } \theta \geq 0 \\ \frac{27}{36} & \text{if } \theta < 0 \end{cases}$$

$$var(\hat{\theta}_2) = \frac{4}{9}var(x[0]) + \frac{1}{9}var(x[1]) \qquad \begin{cases} \frac{20}{36} & \text{if } \theta \geq 0 \\ \frac{24}{36} & \text{if } \theta < 0 \end{cases}$$

Example for the non-existence of MVUs (Kay, 1993)



MVU vs. minimal mean squared error

$$MSE(\hat{\underline{\mathbf{w}}}) = E[(\hat{\underline{\mathbf{w}}} - \underline{\mathbf{w}^*})^2]$$

This however does not yield a realizable estimator because

$$\begin{aligned} MSE(\hat{w}) &= E\left\{[(\hat{w} - E(\hat{w})) + (E(\hat{w}) - w^*)]^2\right\} \\ &= var(\hat{w}) + [E(\hat{w}) - w^*]^2 \\ &= variance + bias^2 \end{aligned}$$

MSE trades bias against variance.

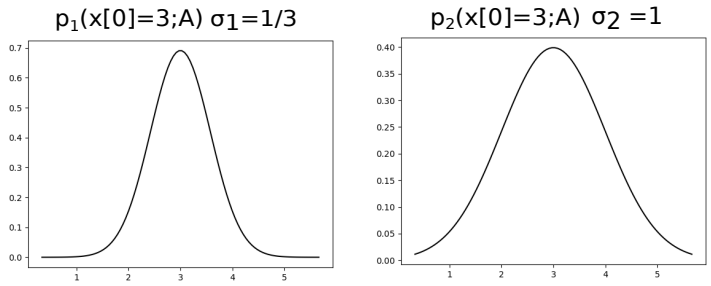
7.2 Cramer-Rao Bound

Cramer-Rao bound for unbiased estimators

The stronger a PDF depends on its parameters, the more accurate will their estimates be.

N observations $x^{(\alpha)}$ with $\epsilon^{(\alpha)} \sim N(0, \sigma^2)$

$$x^{(\alpha)} = A + \epsilon^{(\alpha)}, \qquad \hat{A} = \frac{1}{N}\sum x^{(\alpha)}$$



Accuracy can be measured by the 'sharpness' of the likelihood function ($\rightsquigarrow$ 2nd derivative of the neg. log likelihood).

Cramer-Rao bound for unbiased estimators:

$$M_{ij} = -\left\langle \frac{\partial^2 \ln P}{\partial \mathrm{w}_i \partial \mathrm{w}_j} \right\rangle_p \Bigg|_{\underline{\mathbf{w}}^*} \qquad \left(\substack{\text{Fisher infor-}\\\text{mation matrix}}\right)$$

then for all unbiased estimators:

$$\underline{\mathbf{\Sigma}} - \left(\underline{\mathbf{M}}^{-1}\right) \text{ is a positive semidefinite matrix}$$

proof: see supplementary material it follows:

$$var(\hat{w}_i) \geq [H^{-1}]_{ii} \text{ for all } i$$

Variance of an estimator $> 1/$ Fisher Information
This is a universal lower bound on the variance of estimators. The bound is tight.

$\Rightarrow$ example: one scalar parameter w:

$$\sigma_{\mathrm{w}}^2 - \left\{ - \left\langle \frac{d^2 \ln P}{d\mathrm{w}^2} \right\rangle_p \Bigg|_{\underline{\mathbf{w}}^*} \right\}^{-1} > 0 \qquad \left(\substack{\text{"positive}\\\text{definite"}}\right)$$

$$\sigma_{\mathrm{w}}^2 > -\underbrace{\frac{1}{\left\langle \frac{d^2 \ln P}{d\mathrm{w}^2} \right\rangle_p \Big|_{\underline{\mathbf{w}}^*}}}_{\substack{\text{Fisher}\\\text{information}}} \tag{7.79}$$

$\Rightarrow$ Fisher information is an interesting measure for evaluating data representations

good estimators:

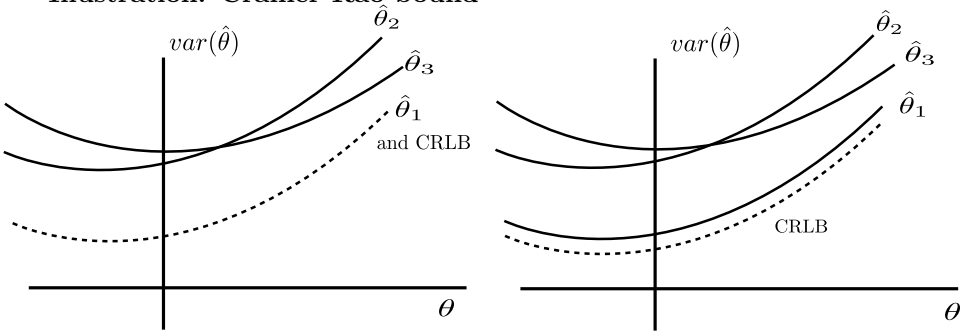
efficient estimator: $\underline{\mathbf{b}} = \underline{\mathbf{0}}$ and $\underline{\mathbf{\Sigma}} = \underline{\mathbf{M}}^{-1}$ $\leftarrow$ variance assumes lower bound

$\substack{\text{unbiased minimum}\\\text{variance estimator:}}$ $\underline{\mathbf{b}} = \underline{\mathbf{0}}$ and $\left|\underline{\mathbf{\Sigma}} - \underline{\mathbf{M}}^{-1}\right| \stackrel{!}{=} \min(\text{all estimators})$ (7.80)

$\Rightarrow$ these estimates may not exist

$\Rightarrow$ even if they exist, they may be difficult to find
(see Kay, 1993, figures. 3.2 and 3.3)

Illustration: Cramer-Rao bound



Asymptotic optimality: An estimator is said to be asymptotically unbiased if for $p \to \infty$ (limit of infinite sample size):

$$E(\hat{\mathrm{w}}) \to \mathrm{w}^*$$

An estimator is said to be asymptotically efficient if for $p \to \infty$:

$$var(\hat{\mathrm{w}}) \to \text{Cramer Rao lower bound}$$

An estimator is said to be consistent if it converges to the true value for $p \to \infty$ and is asymptotically unbiased.

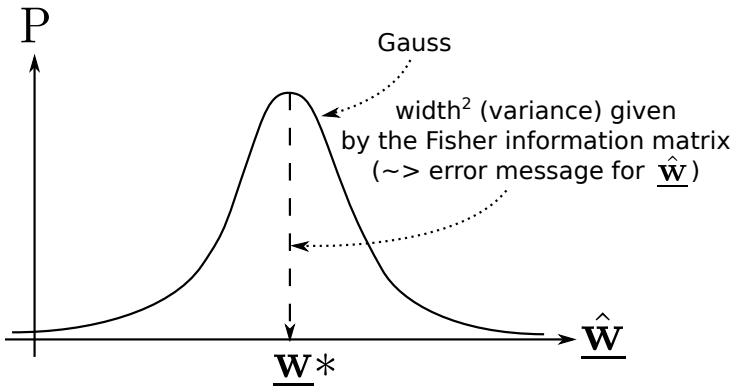
Results for the maximum likelihood estimator

$$\begin{aligned} & P\left(\{\underline{\mathbf{x}}^{(\alpha)}\};\underline{\mathbf{w}}\right) && \text{normalized and two times differentiable} \\ & M_{ij} = -\left\langle \frac{\partial^2 \ln P}{\partial \mathrm{w}_i \partial \mathrm{w}_j} \right\rangle_p && \text{Fisher information matrix} \end{aligned} \tag{7.81}$$

then:

$$\underline{\hat{\mathbf{w}}} \sim \mathcal{N}\left(\underline{\mathbf{w}}^*, \underline{\mathbf{M}}^{-1}_{(\underline{\mathbf{w}}^*)}\right) \text{ asymptotically Gaussian distributed} \tag{7.82}$$

for a proof, see e.g. **Rao1973**



$\Rightarrow$ "maximum likelihood" estimator is asymptotically efficient (unbiased & approaches the Cramer-Rao bound)

$\Rightarrow$ finite number of observations:
maximum likelihood estimator is efficient - if an efficient estimator exists
proof: see supplementary material

$\Rightarrow$ maximum likelihood procedure can often be implemented

$\rightsquigarrow$ very practical estimator

Summary

- An estimator is a random variable.
- It can only be analyzed statistically (e.g. mean, variance, shape of distribution).
- biased & unbiased estimators
- minimum variance unbiased estimator (MVU) has smallest variance for **all values** of the true parameter

MVUs and the Cramer-Rao bound

- minimum variance unbiased estimators do not always exist
- Cramer Rao Bound provides a universal bound but may not be realizable

Outlook

Inclusion of prior knowledge

- MLEs: no prior knowledge regarding 'reasonable' parameter values
- Maximum a Posteriori estimates (MAP) incorporate such knowledge via Bayes Theorem ($\rightsquigarrow$ regularisation)

$$p(\underline{\mathbf{w}}|\underline{\mathbf{x}}) \propto p(\underline{\mathbf{x}}|\underline{\mathbf{w}})p(\underline{\mathbf{w}})$$

- Beyond point estimates: Bayesian statistics. A complete (probabilistic) treatment should exploit the degrees of belief in a given model (set of parameters)

8 Supplementary Material

8.1 Proof of the Cramer-Rao bound

Normalization of the probability density:

$$\int d^p\underline{\mathbf{x}}\, \underbrace{P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\};\underline{\mathbf{w}}\right)}}_{\equiv P} = 1 \tag{8.1}$$

$$\begin{aligned} 0 &= \tfrac{\partial}{\partial \underline{\mathbf{w}}} \int d^p\underline{\mathbf{x}} P \\ &= \int d^p\underline{\mathbf{x}} \tfrac{\partial P}{\partial \underline{\mathbf{w}}} \\ &= \int d^p P \tfrac{\partial \ln P}{\partial \underline{\mathbf{w}}} \\ &= \left\langle \tfrac{\partial \ln P}{\partial \underline{\mathbf{w}}} \right\rangle_p \end{aligned} \tag{8.2}$$

we then obtain

$$\begin{aligned} \left\langle \left(\hat{\mathrm{w}}_i - \mathrm{w}_i^*\right) \tfrac{\partial \ln P}{\partial \mathrm{w}_j}\Big|_{\underline{\mathbf{w}}^*} \right\rangle_p &= \left\langle \hat{\mathrm{w}}_i \overbrace{\tfrac{\partial \ln P}{\partial \mathrm{w}_j}\Big|_{\underline{\mathbf{w}}^*}}^{=\frac{1}{P}\frac{\partial P}{\partial \mathrm{w}_j}} \right\rangle_p - \mathrm{w}_i^* \underbrace{\left\langle \tfrac{\partial \ln P}{\partial \mathrm{w}_j}\Big|_{\underline{\mathbf{w}}^*} \right\rangle_p}_{=0} \\ &= \tfrac{\partial <\hat{\mathrm{w}}_i>_p}{\partial \mathrm{w}_j}\Big|_{\underline{\mathbf{w}}^*} \\ &= \tfrac{\partial \mathrm{w}_i}{\partial \mathrm{w}_j}\Big|_{\underline{\mathbf{w}}^*} \\ &\uparrow \text{estimator without bias} \\ &= \delta_{ij} \end{aligned} \tag{8.3}$$

let $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ be arbitrary vectors, then:

$$\left\langle \underline{\mathbf{a}}^T \left(\hat{\underline{\mathbf{w}}} - \underline{\mathbf{w}}^*\right) \left(\frac{\partial \ln P}{\partial \underline{\mathbf{w}}}\right)^T \underline{\mathbf{b}} \right\rangle\Big|_{\underline{\mathbf{w}}^*} = \underline{\mathbf{a}}^T \underline{\mathbf{b}} \tag{8.4}$$

Application of the Cauchy-Schwarz inequality:

$$\left\{ \int f_{(\underline{\mathbf{D}})} g_{(\underline{\mathbf{D}})} h_{(\underline{\mathbf{D}})} d\underline{\mathbf{D}} \right\}^2 \leq \left\{ \int f_{(\underline{\mathbf{D}})} g^2_{(\underline{\mathbf{D}})} d\underline{\mathbf{D}} \right\} \left\{ \int f_{(\underline{\mathbf{D}})} h^2_{(\underline{\mathbf{D}})} d\underline{\mathbf{D}} \right\} \tag{8.5}$$

with:

$$\begin{aligned} \underline{\mathbf{D}} &= \left\{\underline{\mathbf{x}}^{(\alpha)}\right\} \\ f_{(\underline{\mathbf{D}})} &= P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\},\underline{\mathbf{w}}\right)} \\ g_{(\underline{\mathbf{D}})} &= \underline{\mathbf{a}}^T \left(\hat{\underline{\mathbf{w}}} - \underline{\mathbf{w}}^*\right) \\ h_{(\underline{\mathbf{D}})} &= \left(\frac{\partial \ln P_{\left(\{\underline{\mathbf{x}}^{(\alpha)}\}\right)}}{\partial \underline{\mathbf{w}}} \right)^T \underline{\mathbf{b}} \end{aligned}$$

yields:

$$\left(\underline{\mathbf{a}}^T\underline{\mathbf{b}}\right)^2 \leq \underline{\mathbf{a}}^T\Big\langle\left(\hat{\underline{\mathbf{w}}}-\underline{\mathbf{w}}^*\right)\left(\hat{\underline{\mathbf{w}}}-\underline{\mathbf{w}}^*\right)^T\Big\rangle_p\underline{\mathbf{a}}\underline{\mathbf{b}}^T\Big\langle\frac{\partial\ln P}{\partial\underline{\mathbf{w}}}\Big(\frac{\partial\ln P}{\partial\underline{\mathbf{w}}}\Big)^T\Big\rangle_p\Big|_{\underline{\mathbf{w}}^*}\underline{\mathbf{b}} \tag{8.6}$$

Using:

$$\begin{aligned}\Big\langle\tfrac{\partial^2\ln P}{\partial \mathrm{w}_i\partial \mathrm{w}_j}\Big\rangle_p &= \Big\langle\tfrac{\partial}{\partial \mathrm{w}_i}\Big(\tfrac{1}{P}\tfrac{\partial P}{\partial \mathrm{w}_i}\Big)\Big\rangle_p\\ &= \Big\langle\Big\{-\tfrac{1}{P^2}\tfrac{\partial P}{\partial \mathrm{w}_i}\tfrac{\partial P}{\partial \mathrm{w}_j}+\tfrac{1}{P}\tfrac{\partial}{\partial \mathrm{w}_i}\tfrac{\partial P}{\partial \mathrm{w}_j}\Big\}\Big\rangle_p\\ &= -\Big\langle\tfrac{\partial\ln P}{\partial \mathrm{w}_i}\tfrac{\partial\ln P}{\partial \mathrm{w}_j}\Big\rangle_p+\Big\langle\tfrac{1}{P}\tfrac{\partial}{\partial \mathrm{w}_i}\Big(P\tfrac{\partial\ln P}{\partial \mathrm{w}_j}\Big)\Big\rangle_p\\ &= -\Big\langle\tfrac{\partial\ln P}{\partial \mathrm{w}_i}\tfrac{\partial\ln P}{\partial \mathrm{w}_j}\Big\rangle_p+\tfrac{\partial}{\partial \mathrm{w}_i}\underbrace{\Big\langle\frac{\partial\ln P}{\partial \mathrm{w}_j}\Big\rangle_p}_{=0}\end{aligned} \tag{8.7}$$

we obtain:

$$\left(\underline{\mathbf{a}}^T\underline{\mathbf{b}}\right)^2 \leq \left(\underline{\mathbf{a}}^T\underline{\sum}\underline{\mathbf{a}}\right)\left(\underline{\mathbf{b}}^T\underline{\mathbf{M}}\underline{\mathbf{b}}\right) \tag{8.8}$$

let $\underline{\mathbf{b}}=\underline{\mathbf{M}}^{-1}\underline{\mathbf{a}}$ (ok, because $\underline{\mathbf{b}}$ can be an arbitrary vector), then:

$$\left(\underline{\mathbf{a}}^T\underline{\mathbf{M}}^{-1}\underline{\mathbf{a}}\right)^2 \leq \left(\underline{\mathbf{a}}^T\underline{\sum}\underline{\mathbf{a}}\right)\left(\underline{\mathbf{b}}^T\underline{\mathbf{M}}^{-1}\underline{\mathbf{b}}\right) \tag{8.9}$$

$$\left(\underline{\mathbf{a}}^T\underline{\mathbf{M}}^{-1}\underline{\mathbf{a}}\right) \leq \left(\underline{\mathbf{a}}^T\underline{\sum}\underline{\mathbf{a}}\right) \text{ for all vectors } \underline{\mathbf{a}} \tag{8.10}$$

Consequence:

$$\underline{\sum} - \underline{\mathbf{M}}^{-1} \text{ is positive semidefinite} \tag{8.11}$$

8.2 Maximum likelihood estimator is efficient

Proof of the claim:

Maximum likelihood estimator is efficient - if an efficient estimator exists.

From the proof of the Cramer-Rao bound it follows, that
$\Rightarrow$ the equality sign in the Cauchy-Schwarz inequality (*) holds, if

$$g_{(\underline{\mathbf{D}})} \sim h_{(\underline{\mathbf{D}})}$$

$\Rightarrow$ using the definitions below eq. (*) one obtains

$$\begin{aligned}\underbrace{\underline{\mathbf{a}}^T\overbrace{(\hat{\underline{\mathbf{w}}}-\underline{\mathbf{w}})}^{\text{efficient estimator}}}_{g_{(\underline{\mathbf{w}})}} &= \gamma_{(\underline{\mathbf{w}}^*)}\underbrace{\Big(\frac{\partial\ln P}{\partial\underline{\mathbf{w}}}\Big)^T\underline{\mathbf{b}}}_{h_{(\underline{\mathbf{D}})}}\\ &= \gamma\Big(\tfrac{\partial\ln P}{\partial\underline{\mathbf{w}}}\Big)^T\underbrace{\underline{\mathbf{M}}^{-1}\underline{\mathbf{a}}}_{\text{particular cl??? of }\underline{\mathbf{b}}}\end{aligned} \tag{8.12}$$

since $\underline{\mathbf{a}}$ is an arbitrary vector, we obtain

$$\frac{\partial \ln P}{\partial \underline{\mathbf{w}}} = \frac{1}{\gamma}\underline{\mathbf{M}}(\underline{\hat{\mathbf{w}}} - \underline{\mathbf{w}}) \tag{8.13}$$

calculation of γ:

$$\frac{\partial \ln P}{\partial \mathrm{w}_j} = \sum_k \frac{\mathrm{M}_{jk}}{\gamma}(\hat{\mathrm{w}}_k - \mathrm{w}_k) \tag{8.14}$$

$$\frac{\partial^2 \ln P}{\partial \mathrm{w}_i \partial \mathrm{w}_j} = \sum_k \left\{ -\frac{\mathrm{M}_{jk}}{\gamma} \underbrace{\delta_{ik}}_{\mathrm{w}_k} + (\hat{\mathrm{w}}_k - \mathrm{w}_k)\overbrace{\frac{\partial}{\partial \mathrm{w}_i}}^{\gamma}\left(\frac{\mathrm{M}_{jk}}{\gamma}\right)\right\} \tag{8.15}$$

$$\begin{aligned} \mathrm{M}_{ij} &= \left\langle \tfrac{\partial^2 \ln P}{\partial \mathrm{w}_i \partial \mathrm{w}_j} \right\rangle_p \Big|_{\underline{\mathbf{w}}^*} \\ &= \tfrac{\mathrm{M}_{ji}}{\gamma} \text{ because } <\hat{\mathrm{w}}_k>_p = \mathrm{w}_k^* \end{aligned} \tag{8.16}$$

We obtain:

$$\frac{\partial \ln P}{\partial \underline{\mathbf{w}}} = \underline{\mathbf{M}}(\underline{\hat{\mathbf{w}}} - \underline{\mathbf{w}}) \text{ for all vectors } \underline{\mathbf{w}} \tag{8.17}$$

For the maximum likelihood estimator we get

$$\underbrace{\frac{\partial \ln P}{\partial \underline{\mathbf{w}}} \stackrel{!}{=} 0}_{\text{max. of likelihood}} \quad \Rightarrow \underline{\mathbf{w}} = \underline{\hat{\mathbf{w}}} \tag{8.18}$$

Since $\underline{\hat{\mathbf{w}}}$ is efficient, this also holds for the maximum likelihood estimator.

8.3 Convergence Properties of Oja's Rule

① small learning steps $\rightsquigarrow$ average over all patterns

$$\begin{aligned} \Delta \mathrm{w}_j &= \tfrac{\epsilon}{p} \sum_{\alpha=1}^{p} \left\{ \sum_{k=1}^{N} \mathrm{w}_k \mathrm{x}_k^{(\alpha)} \mathrm{x}_j^{(\alpha)} - \mathrm{w}_j \sum_{k,l=1}^{N} \mathrm{w}_k \mathrm{w}_l \mathrm{x}_k^{(\alpha)} \mathrm{x}_l^{(\alpha)} \right\} \\ &= \epsilon \left\{ \sum_{k=1}^{N} \mathrm{w}_k C_{kj} - \mathrm{w}_j \sum_{k,l=1}^{N} \mathrm{w}_k \mathrm{w}_l C_{kl} \right\} \\ \Delta \underline{\mathbf{w}} &= \epsilon \Big\{ \underbrace{\underline{\mathbf{C}}\underline{\mathbf{w}}}_{\substack{\text{Hebbian} \\ \text{rule}}} - \underbrace{\underbrace{(\underline{\mathbf{w}}^T \underline{\mathbf{C}}\underline{\mathbf{w}})}_{\substack{\text{always} \\ \text{positive}}} \underline{\mathbf{w}}}_{\text{decay term}} \Big\} \end{aligned} \tag{8.19}$$

② stationary states $\underline{\mathbf{w}}^*$ of Oja's rule
$\widehat{=}$ normalized eigenvectors $\underline{\mathbf{e}}_j$ of the correlation matrix $\underline{\mathbf{C}}$

Proof:

stationary state: $\Delta \underline{\mathbf{w}} \stackrel{!}{=} \underline{\mathbf{0}}$

ansatz: $\underline{\mathbf{w}}^* = C\underline{\mathbf{e}}_j \sim \underline{\mathbf{e}}_j$

insertion into Oja's rule:

$$\begin{aligned} &\Delta\underline{\mathbf{w}} = \epsilon\Big\{C\lambda_j\underline{\mathbf{e}}_j - C^3\lambda_j\underline{\mathbf{e}}_j\Big\} \stackrel{!}{=} O \\ &\Rightarrow C^2 = 1 \\ &\Rightarrow \underline{\mathbf{w}}^* = \pm\underline{\mathbf{e}}_j \end{aligned} \tag{8.20}$$

③ The stationary state $\underline{\mathbf{w}}^* = \underline{\mathbf{e}}_j$ is stable if and only if $\underline{\mathbf{e}}_j = \underline{\mathbf{e}}_1$, i.e. if $\underline{\mathbf{e}}_j$ is the eigenvactor with the largest eigenvalue.

Proof: linear stability analysis

$\lambda_1 > \lambda_2 > \ldots > \lambda_N$ eigenvalues of $\underline{\mathbf{C}}$

$\underline{\mathbf{w}} = \underline{\mathbf{e}}_j + \underline{\eta} \leftarrow$ small deviation from the stationary state

$$\begin{aligned} \Delta\underline{\mathbf{w}} = \quad \Delta\underline{\eta} \quad &= \epsilon\underline{\mathbf{C}}(\underline{\mathbf{e}}_j + \underline{\eta}) - \epsilon\Big\{(\underline{\mathbf{e}}_j + \underline{\eta})^T\underline{\mathbf{C}}(\underline{\mathbf{e}}_j + \underline{\eta})\Big\}(\underline{\mathbf{e}}_j + \underline{\eta}) \\ &= \epsilon\Big\{\lambda_j\underline{\mathbf{e}}_j + \underline{\mathbf{C}}\underline{\eta} - \lambda_j\underline{\mathbf{e}}_j - \lambda_j\underline{\eta} - 2\lambda_j(\underline{\mathbf{e}}_j^T\underline{\eta})(\underline{\mathbf{e}}_j + \underline{\eta}) - (\underline{\eta}^T\underline{\mathbf{C}}\underline{\eta})(\underline{\mathbf{e}}_j + \underline{\eta})\Big\} \\ \Delta\eta \quad &= \epsilon\Big\{ - 2\lambda_j(\underline{\mathbf{e}}_j^T\underline{\eta})\underline{\mathbf{e}}_j + \underline{\mathbf{C}}\underline{\eta} - \lambda_j\underline{\eta}\Big\} + \underbrace{O_{(\underline{\eta}^2)}}_{\substack{\text{discarded for} \\ |\underline{\eta}|\to 0}} \end{aligned} \tag{8.21}$$

Projection onto the eigenvectors $\underline{\mathbf{e}}_k$:

$$\underline{\mathbf{e}}_k^T\Delta\underline{\eta} = \epsilon\Big\{(\lambda_k - \lambda_j)\underline{\mathbf{e}}_k^T\underline{\eta} - 2\lambda_k(\underline{\mathbf{e}}_k^T\underline{\eta}) \underbrace{\delta_{kj}}_{\substack{\text{Kronecker-} \\ \text{Delta}}} \Big\} \tag{8.22}$$

<u>case I:</u> $k = j$

$$\begin{aligned} &\underline{\mathbf{e}}_k^T\Delta\underline{\eta} = \underbrace{-2\epsilon\lambda_k}_{\substack{\text{always} \\ \text{negative}}}\underline{\mathbf{e}}_k^T\underline{\eta} \\ &\Rightarrow \underline{\mathbf{e}}_k^T\underline{\eta} \to 0 \end{aligned} \tag{8.23}$$

<u>case II:</u> $k \neq j$

$$\begin{aligned} \underline{\mathbf{e}}_k^T\Delta\underline{\eta} = \quad &\underbrace{\epsilon(\lambda_k - \lambda_j)}_{\substack{\text{factor is negative} \\ \text{only if } \lambda_j \text{ is} \\ \text{the largest eigenvalue}}} \quad \underline{\mathbf{e}}_k^T\underline{\eta} \\ &\Rightarrow \text{ if } \lambda_j = \lambda_1, \text{ then } \underline{\mathbf{e}}_k^T\underline{\eta} \to 0 \end{aligned} \tag{8.24}$$

8.4 Mercer's Theorem

Preliminaries:

χ : compact subset of $\mathbb{R}^N$

K : $\chi \times \chi \to \mathbb{R}, k \in L_\alpha,$ symmetric function ("kernel")

T_k : $L_{2(\chi)} \to L_{2(\chi)}$

$(T_k f)_{(\underline{\mathbf{x}})} := \int_\chi K_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} f_{(\underline{\mathbf{x}}')} d\underline{\mathbf{x}}'$

} corresponding integral operator

λ_j : eigenvalues

$\psi_j \in L_{2(\chi)}$: normalized eigenfunction

} of T_k

essential condition: T_k positive definite

$$\int_{\chi\times\chi} K_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} f_{(\underline{\mathbf{x}})} f_{(\underline{\mathbf{x}}')} d\underline{\mathbf{x}} d\underline{\mathbf{x}}' > 0 \qquad \forall f \in L2(\chi) \tag{8.25}$$

then:

$$K_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \sum_{j=1}^{n} \lambda_j \underbrace{\psi_{j(\underline{\mathbf{x}})} \psi_{j(\underline{\mathbf{x}}')}}_{\substack{\text{eigenvalue} \\ \text{decomposition}}} \tag{8.26}$$

$n \to \infty$: absolute and uniform convergence (non-trivial part)

consequences of Mercer's theorem:

$$\begin{aligned} &\underline{\phi} : \underline{\mathbf{x}} \to \left(\sqrt{\lambda_1}\psi_{1(\underline{\mathbf{x}})}, \sqrt{\lambda_2}\psi_{2(\underline{\mathbf{x}})}, \ldots, \sqrt{\lambda_n}\psi_{n(\underline{\mathbf{x}})}\right)^T \\ &K_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \underline{\phi}^T_{(\underline{\mathbf{x}})} \underline{\phi}_{(\underline{\mathbf{x}}')} \end{aligned} \tag{8.27}$$

8.5 Natural gradient - alternate derivation

$$\begin{aligned} de &= \sum_{i,j} \tfrac{\partial e}{\partial \mathrm{w}_{ij}} d\mathrm{w}_{ij} \\ &= \sum_{i,j} \varphi_i \mathrm{x}_j d\mathrm{w}_{ij} + \sum_{i,j} \left(\underline{\mathbf{w}}^{-1}\right)_{ji} d\mathrm{w}_{ij} \\ &= \underline{\varphi}^T d\underline{\mathbf{w}}\underline{\mathbf{x}} + \mathrm{Tr}\left(d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1}\right) \\ &= \underline{\varphi}^T \underbrace{\left(d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1}\right) \overbrace{\hat{\underline{\mathbf{s}}}}^{\hat{\underline{\mathbf{s}}}=\underline{\mathbf{w}}\underline{\mathbf{x}}} + \mathrm{Tr}\left(d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1}\right)}_{d\underline{\mathbf{Z}}} \end{aligned} \tag{8.28}$$

with

$$d\underline{\mathbf{Z}} = d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1} \tag{8.29}$$

we obtain

$$de = \underline{\varphi}^T \underline{\widehat{\mathbf{s}}} d\underline{\mathbf{Z}} + \mathrm{Tr}(d\underline{\mathbf{Z}}) \tag{8.30}$$

gradient ascent learning

$$\begin{aligned} \Delta \mathrm{Z}_{ij} &= \eta \tfrac{\partial e}{\partial \mathrm{z}_{ij}} \\ &= \eta\big(\varphi_i \widehat{s}_j + \delta_{ij}\big) \\ &= \eta\Big(\varphi_i \sum_k \mathrm{w}_{jk}\mathrm{x}_k + \delta_{ij}\Big) \\ &= \sum_k \Delta \mathrm{w}_{jk} \big(\underline{\mathbf{w}}^{-1}\big)_{kj} \end{aligned} \tag{8.31}$$

$$\Delta \mathrm{w}_{il} = \eta\Big(\mathrm{w}_{il} + \varphi_i \sum_{k,j} \mathrm{w}_{jl}\mathrm{w}_{jk}\mathrm{x}_k\Big) \tag{8.32}$$

8.6 Kurtosis optimization

Two statistically independent sources with $\langle \mathrm{s}_i \mathrm{s}_j \rangle = \delta_{ij}$
Extraction from observations (one source)

$$\begin{aligned} \widehat{\mathrm{s}} &= \underline{\mathbf{z}}^T \underline{\mathbf{s}} \\ &= \mathrm{z}_1\mathrm{s}_1 + \mathrm{z}_2\mathrm{s}_2 \end{aligned} \tag{8.33}$$

optimization problem

$$\begin{aligned} \mathrm{kurt}(\widehat{\mathrm{s}}) &\stackrel{!}{=} \max_{\underline{\mathbf{z}}} \\ \mathrm{z}_1^2 + \mathrm{z}_2^2 &\stackrel{!}{=} 1 \end{aligned} \tag{8.34}$$

method of lagrange multiplies

$$\mathrm{kurt}(\widehat{\mathrm{s}}) = \mathrm{z}_1^4 \underbrace{\mathrm{kurt}(\mathrm{s}_1)}_{=a} + \mathrm{z}_2^2 \underbrace{\mathrm{kurt}(\mathrm{s}_2)}_{=b} \tag{8.35}$$

let $a, b > 0$ (special case)

$$a\mathrm{z}_1^4 + b\mathrm{z}_2^4 - \lambda\big(\mathrm{z}_1^2 + \mathrm{z}_2^2\big) \stackrel{!}{=} \text{exts.} \tag{8.36}$$

$$\begin{aligned} 4a\mathrm{z}_1^3 - 2\lambda\mathrm{z}_1 = 0 \quad &\rightsquigarrow \mathrm{z}_1\big(4a\mathrm{z}_1^2 - 2\lambda\big) = 0 \quad \rightsquigarrow \mathrm{z}_1 = \pm\tfrac{\lambda}{2a}, 0 \\ 4b\mathrm{z}_2^3 - 2\lambda\mathrm{z}_2 = 0 \quad &\rightsquigarrow \mathrm{z}_2\big(4a\mathrm{z}_2^2 - 2\lambda\big) = 0 \quad \rightsquigarrow \mathrm{z}_2 = \pm\tfrac{\lambda}{2b}, 0 \end{aligned} \tag{8.37}$$

matrix of second derivatives

$$\left| \begin{pmatrix} 12a\mathrm{z}_1^2 - 2\lambda & 0 \\ 0 & 12b\mathrm{z}_2^2 - 2\lambda \end{pmatrix} \right| = \big(12a\mathrm{z}_1^2 - 2\lambda\big)\big(12b\mathrm{z}_2^2 - 2\lambda\big) \tag{8.38}$$

solutions

z_1	z_2	λ	$1\%1$	$a\mathrm{z}_1^4 + b\mathrm{z}_2^4$
0	± 1	$2b$	$-32b^2$	b
± 1	0	$2a$	$-32a^2$	a
$\pm\frac{b}{\sqrt{a^2+b^2}}$	$\pm\frac{a}{\sqrt{a^2+b^2}}$	$\frac{2ab}{\sqrt{a^2+b^2}}$	depending on values for a,b	at most local optima: $\frac{ab}{(a^2+b^2)^2}(a^3+b^3)$

8.7 Mean-field for pairwise clustering

Self-consistent equations for the mean-field ⊖

$$\frac{\partial\langle E^p\rangle_Q}{\partial e_r^{(\alpha)}} = \sum_p \frac{\partial\left\langle m_p^{(\alpha)}\right\rangle_Q}{\partial e_r^{(\alpha)}} e_r^{(\alpha)} \tag{8.39}$$

useful relations:
①

$$\frac{m_p^{(\alpha)}}{\sum\limits_\gamma m_p^{(\gamma)}} = \frac{m_p^{(\alpha)}}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)} + 1} \tag{8.40}$$

② from $\frac{1}{a+b} = \frac{1}{a}\left(1 - \frac{b}{a+b}\right)$ it follows

$$\begin{aligned}
\frac{1}{\sum\limits_\gamma m_p^{(\gamma)}} &= \frac{1}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)} + m_p^{(\alpha)}} \\
&= \frac{1}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}}\left\{1 - \frac{m_p^{(\alpha)}}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)} + m_p^{(\alpha)}}\right\} \\
&= \frac{1}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}}\left\{1 - \frac{m_p^{(\alpha)}}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)} + 1}\right\}
\end{aligned} \tag{8.41}$$

derivations of the expected cost:

$$\begin{aligned}
\frac{\partial\left\langle E^p\right\rangle_Q}{\partial e_r^{(\alpha)}} &= \frac{1}{M}\sum_{p,\delta,\varepsilon} \underbrace{\frac{\partial}{\partial e_r^{(\alpha)}}\left\langle \frac{m_p^{(\delta)} m_p^{(\varepsilon)}}{\sum\limits_\gamma m_p^{(\gamma)}}\right\rangle_Q d_{\delta\varepsilon}}_{\circledast} \\
&= \frac{1}{M}\sum_p \left\{ \sum_{\substack{\delta,\varepsilon \\ \delta\neq\varepsilon;\delta,\varepsilon\neq\alpha}} \circledast + \sum_{\substack{\delta \\ \delta\neq\alpha;\varepsilon=\alpha}} \circledast + \sum_{\substack{\varepsilon \\ \varepsilon\neq\alpha;\delta=\alpha}} \circledast + \sum_{\substack{\delta,\varepsilon \\ \delta=\varepsilon;\delta,\varepsilon\neq\alpha}} \circledast + \sum_{\substack{\delta,\varepsilon \\ \delta=\varepsilon=\alpha}} \circledast \right\}
\end{aligned} \tag{8.42}$$

calculation of the individual expectations

① $\delta, \varepsilon : \delta \neq \varepsilon; \delta, \varepsilon \neq \alpha$

$$\begin{aligned}\left\langle \frac{m_p^{(\delta)} m_p^{(\epsilon)}}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q &\underbrace{=}_{\text{using ②}} \left\langle \frac{m_p^{(\delta)} m_p^{(\varepsilon)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}} \left\{ 1 - \frac{m_p^{(\alpha)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\} \right\rangle_Q \\ &= \left\langle \frac{m_p^{(\delta)} m_p^{(\varepsilon)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}} \right\rangle_Q - \left\langle \frac{m_p^{(\delta)} m_p^{(\varepsilon)}}{\left(\sum_{\gamma\neq\alpha} m_p^{(\gamma)}\right)\left(\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1\right)} \right\rangle_Q \left\langle m_p^{(\alpha)} \right\rangle_Q \end{aligned} \tag{8.43}$$

② $\delta : \delta \neq \alpha; \varepsilon = \alpha$

$$\begin{aligned}\left\langle \frac{m_p^{(\delta)} m_p^{(\alpha)}}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q &\underbrace{=}_{\text{using ①}} \left\langle \frac{m_p^{(\delta)} m_p^{(\alpha)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q \\ &= \left\langle \frac{m_p^{(\delta)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q \left\langle m_p^{(\alpha)} \right\rangle_Q \end{aligned} \tag{8.44}$$

③ $\varepsilon : \varepsilon \neq \alpha; \delta = \alpha$

$$\left\langle \frac{m_p^{(\alpha)} m_p^{(\varepsilon)}}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q \underbrace{=}_{\substack{\text{in analogy} \\ \text{to ②}}} \left\langle \frac{m_p^{(\varepsilon)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q \left\langle m_p^{(\alpha)} \right\rangle_Q \tag{8.45}$$

④ $\delta, \varepsilon : \delta = \varepsilon; \delta, \varepsilon \neq \alpha$

$$\begin{aligned}\left\langle \frac{\left(m_p^{(\varepsilon)}\right)^2}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q &= \left\langle \frac{m_p^{(\varepsilon)}}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q \\ &\underbrace{=}_{\text{using ②}} \left\{ \left\langle \frac{m_p^{(\varepsilon)}}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}} \right\rangle_Q - \left\langle \frac{m_p^{(\varepsilon)}}{\left(\sum_{\gamma\neq\alpha} m_p^{(\gamma)}\right)\left(\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1\right)} \right\rangle_Q \left\langle m_p^{(\alpha)} \right\rangle_Q \right\} \end{aligned} \tag{8.46}$$

⑤ $\delta, \varepsilon : \delta = \varepsilon = \alpha$

$$\begin{aligned}\left\langle \frac{\left(m_p^{(\alpha)}\right)^2}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q &= \left\langle \frac{m_p^{(\alpha)}}{\sum_\gamma m_p^{(\gamma)}} \right\rangle_Q \\ &\underbrace{=}_{\text{using ①}} \left\langle \frac{1}{\sum_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q \left\langle m_p^{(\alpha)} \right\rangle_Q \end{aligned} \tag{8.47}$$

inserting this into the derivative we obtain:

$$\begin{aligned}
\frac{\partial\left\langle E^p\right\rangle_Q}{\partial e_r^{(\alpha)}} &= \frac{1}{M}\sum_p \frac{\partial\left\langle m_p^{(\alpha)}\right\rangle_Q}{\partial e_r^{(\alpha)}}\cdot \\
&\Bigg\{\underbrace{-\sum_{\substack{\delta,\varepsilon\\ \delta\neq\varepsilon;\delta,\varepsilon\neq\alpha}}\left\langle\frac{m_p^{(\delta)}m_p^{(\varepsilon)}}{\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}\Big)\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1\Big)}\right\rangle_Q d_{\delta\varepsilon}}_{\text{from }①} \\
&\underbrace{+\sum_{\substack{\delta\\ \delta\neq\alpha}}\left\langle\frac{m_p^{(\delta)}}{\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1}\right\rangle_Q\left(d_{\delta\alpha}+d_{\alpha\delta}\right)}_{\text{from }② \text{ and } ③} \\
&\underbrace{-\sum_{\substack{\delta\\ \delta\neq\alpha}}\left\langle\frac{m_p^{(\delta)}}{\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}\Big)\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1\Big)}\right\rangle_Q\left\langle m_p^{(\alpha)}\right\rangle_Q d_{\delta\delta}}_{\text{from }④} \\
&\underbrace{+\left\langle\frac{1}{\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1}\right\rangle_Q d_{\alpha\alpha}}_{\text{from }⑤}\Bigg\}
\end{aligned} \tag{8.48}$$

using eq. ⊖ and comparing coefficients, we find

$$\begin{aligned}
e_p^{(\alpha)} &= \frac{1}{M}\left\langle\frac{1}{\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1}\right\rangle_Q d_{\alpha\alpha}+\frac{1}{M}\sum_{\delta\neq\alpha}\left\langle\frac{m_p^{(\delta)}}{\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1}\right\rangle_Q\left\{d_{\delta\alpha}+d_{\alpha\delta}\right\} \\
&\quad -\frac{1}{M}\sum_{\delta\neq\alpha}\left\langle\frac{m_p^{(\delta)}}{\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}\Big)\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1\Big)}\right\rangle_Q d_{\delta\delta} \\
&\quad -\frac{1}{M}\sum_{\substack{\delta,\varepsilon\\ \delta\neq\varepsilon;\delta,\varepsilon\neq\alpha}}\left\langle\frac{m_p^{(\delta)}m_p^{(\varepsilon)}}{\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}\Big)\Big(\sum\limits_{\gamma\neq\alpha}m_p^{(\gamma)}+1\Big)}\right\rangle_Q d_{\delta\varepsilon}
\end{aligned} \tag{8.49}$$

approximations for large numbers of data points (neglecting terms Q_{y_p})

$$\begin{aligned}
\left\langle \frac{1}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q &\approx \frac{1}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} && \text{proof via Taylor expansion} \\
\left\langle \frac{m_p^{(\delta)}}{\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}+1} \right\rangle_Q &\approx \frac{\left\langle m_p^{(\delta)} \right\rangle_Q}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} && \text{using above and relation ①} \\
\left\langle \frac{m_p^{(\delta)} m_p^{(\varepsilon)}}{\left(\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}\right)\left(\sum\limits_{\gamma\neq\alpha} m_p^{(\gamma)}+1\right)} \right\rangle_Q &\approx \frac{\left\langle m_p^{(\delta)} \right\rangle_Q \left\langle m_p^{(\varepsilon)} \right\rangle_Q}{\left(\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q\right)^2} && \text{using above and relation ①}
\end{aligned} \tag{8.50}$$

leads to the self-consistent equations for the mean-fields:

$$\begin{aligned}
e_p^{(\alpha)} = \frac{1}{M}\Bigg[& \frac{1}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} \Bigg\{ d_{\alpha\alpha} - \frac{1}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} \sum_{\delta\neq\alpha} \left\langle m_p^{(\delta)} \right\rangle_Q d_{\delta\delta} \Bigg\} \\
& + \frac{1}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} \sum_{\delta\neq\alpha} \left\langle m_p^{(\delta)} \right\rangle_Q \Bigg\{ \left(d_{\delta\alpha} + d_{\alpha\delta}\right) \\
& - \frac{1}{2} \frac{1}{\sum\limits_{\gamma} \left\langle m_p^{(\gamma)} \right\rangle_Q} \sum_{\substack{\delta,\varepsilon \\ \delta\neq\varepsilon;\delta,\varepsilon\neq\alpha}} \left\langle m_p^{(\varepsilon)} \right\rangle_Q \left(d_{\delta\varepsilon} + d_{\varepsilon\delta}\right) \Bigg\} \Bigg]
\end{aligned} \tag{8.51}$$

8.8 Mean-fields for central clustering

$$d_{\alpha\gamma} = \frac{1}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\gamma)}\big)^2, \text{ approximation to } O_{\left(\frac{1}{p}\right)} \tag{8.52}$$

$$\begin{aligned}
e_p^{(\alpha)} &= \frac{1}{M\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\big(\underline{\mathbf{x}}^{(\delta)} - \underline{\mathbf{x}}^{(\alpha)}\big)^2 \\
&\quad -\frac{1}{M\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\sum_{\varepsilon\neq\alpha}\left\langle m_p^{(\varepsilon)}\right\rangle_Q\big(\underline{\mathbf{x}}^{(\delta)} - \underline{\mathbf{x}}^{(\varepsilon)}\big)^2 \\
&= \frac{1}{M\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\left\{\underline{\mathbf{x}}^{(\delta)} - 2\big(\underline{\mathbf{x}}^{(\delta)}\big)^T\underline{\mathbf{x}}^{(\alpha)} + \big(\underline{\mathbf{x}}^{(\alpha)}\big)^2\right\} \\
&\quad -\frac{1}{M\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\sum_{\varepsilon\neq\alpha}\left\langle m_p^{(\varepsilon)}\right\rangle_Q \\
&\quad \cdot\left\{\big(\underline{\mathbf{x}}^{(\delta)}\big)^2 - 2\big(\underline{\mathbf{x}}^{(\delta)}\big)^T\underline{\mathbf{x}}^{(\varepsilon)} + \big(\underline{\mathbf{x}}^{(\varepsilon)}\big)^2\right\} \\
&= \frac{1}{M}\left[\frac{\sum\limits_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\big(\underline{\mathbf{x}}^{(\delta)}\big)^2}{\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q} - 2\frac{\big(\underline{\mathbf{x}}^{(\alpha)}\big)^T\sum\limits_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\underline{\mathbf{x}}^{(\delta)}}{\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q} + \big(\underline{\mathbf{x}}^{(\alpha)}\big)^2\right] \\
&\quad -\frac{1}{M\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\left\{\frac{\sum\limits_{\varepsilon\neq\alpha}\left\langle m_p^{(\varepsilon)}\right\rangle_Q\big(\underline{\mathbf{x}}^{(\varepsilon)}\big)^2}{\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q}\right. \\
&\quad \left.-2\frac{\big(\underline{\mathbf{x}}^{(\delta)}\big)^T\sum\limits_{\varepsilon\neq\alpha}\left\langle m_p^{(\varepsilon)}\right\rangle_Q\underline{\mathbf{x}}^{(\varepsilon)}}{\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q} + \big(\underline{\mathbf{x}}^{(\delta)}\big)^2\right\} \\
&\approx \frac{1}{M}\left[\big(\underline{\mathbf{x}}^{(\alpha)}\big)^2 - 2\big(\underline{\mathbf{x}}^{(\alpha)}\big)^T\underline{\mathbf{w}}_p + \underline{\mathbf{w}}_p^2\right]
\end{aligned} \tag{8.53}$$

with:

$$\underline{\mathbf{w}}_p = \frac{\sum\limits_{p}\left\langle m_p^{(\gamma)}\right\rangle_Q\underline{\mathbf{x}}^{(\gamma)}}{\sum\limits_{\gamma}\left\langle m_p^{(\gamma)}\right\rangle_Q} \tag{8.54}$$