Machine Intelligence II

Prof. Dr. Klaus Obermayer

March 22, 2016

Contents

1 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)

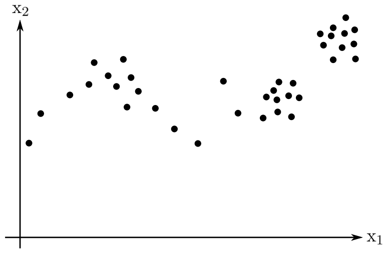
$\Rightarrow$ projection methods: search for "interesting" directions in feature space

$\Rightarrow$ clustering methods: grouping & categorization (and prototypes)

Note: Both PCA and ICA (discussed in chapters 2.1 and 2.4.1) are linear projection methods. In many cases (e.g. PCA) kernel methods allow to extend them to nonlinear problems.

2 Projection Methods

Projection methods provide important tools for high-dimensional datasets



p observations with N dims.:
$\{\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \ldots, \underline{\mathbf{x}}^{(p)} \in \mathbb{R}^N\}$

Figure 1: Projection and clustering methods for multidimensional data

2.1 Principal Component Analysis

2.1.1 The Covariance Matrix

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

Feature space:

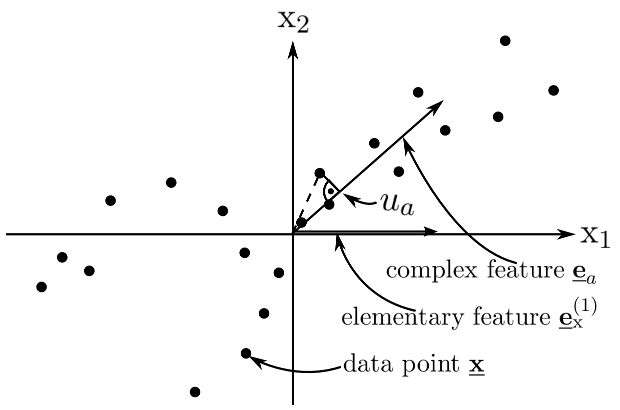

x2
x1
u_a
complex feature $\underline{\mathbf{e}}_a$
elementary feature $\underline{\mathbf{e}}_x^{(1)}$
data point $\underline{\mathbf{x}}$

Figure 2: Data, elementary and complex features

elementary features $\underline{\mathbf{e}}_x^{(1)}, \underline{\mathbf{e}}_x^{(2)}, \underline{\mathbf{e}}_x^{(3)}, \ldots \underline{\mathbf{e}}_x^{(N)}$: 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 $\underline{\mathbf{e}}_a$

$$\underbrace{u_a}_{\text{value}} = \underbrace{\underline{\mathbf{e}}_a^T}_{\text{feature}} \cdot \underbrace{\underline{\mathbf{x}}}_{\text{observation}} \tag{2.1}$$

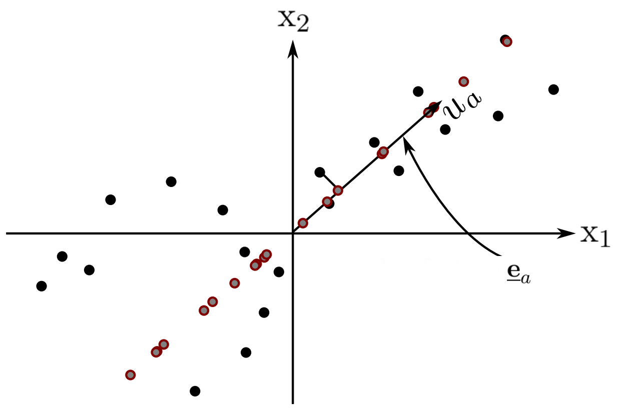


Figure 3: Same data as in Fig. 2, projected onto complex feature $\underline{\mathbf{e}}_a$ with feature values u_a

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

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

- *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):

$$\underline{\mathbf{m}} = \frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)} \tag{2.2}$$

Second moments (Variances and Covariances):

$$C_{ij} = \frac{1}{p}\sum_{\alpha=1}^{p}\underbrace{\left(\mathrm{x}_i^{(\alpha)} - m_i\right)}_{\substack{\text{deviations from}\\ \text{the mean}}}\underbrace{\left(\mathrm{x}_j^{(\alpha)} - m_j\right)}_{\substack{\text{component}\\ \text{indices } i,j}} \tag{2.3}$$

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

$$C_{ij} = \frac{1}{p} \sum_{\alpha=1}^{p} \mathrm{x}_i^{(\alpha)} \mathrm{x}_j^{(\alpha)} \tag{2.4}$$

and thus

$$\underline{\mathbf{C}} = \frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} \right)^T \tag{2.5}$$

Properties of the covariance matrix

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

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

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

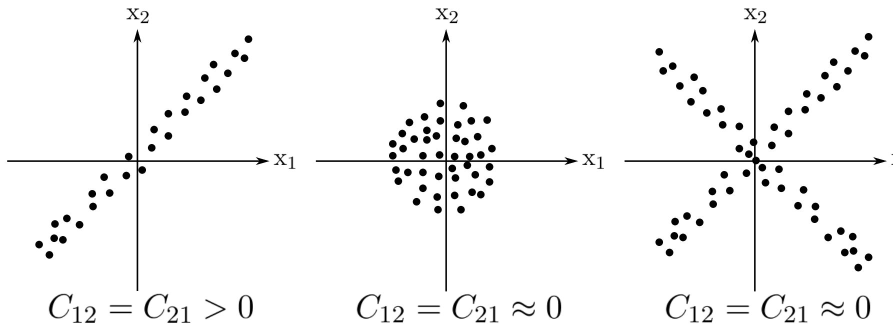


Figure 4: Illustration of data patterns: Note that $C_{ij} = 0$ does <u>not</u> imply that there are no dependencies between the variables. Clearly, $p(\mathrm{x}_1, \mathrm{x}_2) \neq p(\mathrm{x}_1)p(\mathrm{x}_2)$ in the third example.

[1] centering: subtract mean from all data points

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

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

Second moment along this direction:

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

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

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

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

(variance)

$$\boxed{\sigma_a^2 = \underline{\mathbf{e}}_a^T \mathbf{C} \underline{\mathbf{e}}_a} \qquad (2.6)$$

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

2.1.2 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

$\Rightarrow$ direction in feature space along which variance is maximal

$$\boxed{\sigma_a^2 = \max\left(\underline{\mathbf{e}}_a\right)}$$

optimisation under constraints

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

Solution is found using the method of Lagrange multipliers:

$$\underbrace{\underline{\mathbf{e}}_a^T \underline{\mathbf{C}} \underline{\mathbf{e}}_a}_{\text{objective}} - \underbrace{\lambda}_{\substack{\text{Lagrange}\\ \text{multiplier}}} \underbrace{\left(\underline{\mathbf{e}}_a^2 - 1\right)}_{\text{constraints}} \overset{!}{=} \max \tag{2.7}$$

$\rightsquigarrow$ family of solutions parametrized by λ

$\rightsquigarrow$ λ must be chosen such that constraints are fulfilled

$$f(\underline{\mathbf{e}}_a) := \sum_{i,j=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i C_{ij} \left(\underline{\mathbf{e}}_a\right)_j - \lambda \left\{ \sum_{i=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i^2 - 1 \right\} \overset{!}{=} \max \tag{2.8}$$

$$\frac{\partial f}{\partial \left(\underline{\mathbf{e}}_a\right)_k} \overset{!}{=} 0 \text{ for all components } k \tag{2.9}$$

$$\sum_{j=1}^{N} C_{kj} \left(\underline{\mathbf{e}}_a\right)_j + \sum_{i=1}^{N} \left(\underline{\mathbf{e}}_a\right)_i C_{ik} - 2\lambda \left(\underline{\mathbf{e}}_a\right)_k \overset{!}{=} 0 \tag{2.10}$$

$$2 \sum_{j=1}^{N} C_{kj} \left(\underline{\mathbf{e}}_a\right)_j - 2\lambda \left(\underline{\mathbf{e}}_a\right)_k \overset{!}{=} 0 \text{ because } \underline{\mathbf{C}} \text{ is symmetric} \tag{2.11}$$

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

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

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

$\rightsquigarrow$ variance $\sigma_a^2 = \underline{\mathbf{e}}_a^T \underline{\mathbf{C}} \underline{\mathbf{e}}_a = \lambda \underline{\mathbf{e}}_a^2 = \lambda$
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

2.1.3 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!)

$\rightarrow$ eigenvectors form an orthonormal basis:

$$\underline{\mathbf{e}}_i^T \cdot \underline{\mathbf{e}}_j = \delta_{ij} \leftarrow \text{ Kronecker-Delta } \delta_{ij} = \begin{cases} 1, & i = j \\ \\ 0, & \text{else} \end{cases} \tag{2.12}$$

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

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

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

$$\underline{\mathbf{M}}^T \underline{\mathbf{C}} \underline{\mathbf{M}} = \widehat{\underline{\mathbf{C}}} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ & 0 & & \ddots & \\ & & & & \lambda_N \end{pmatrix} \text{transformation into the eigenbasis} \tag{2.13}$$

$\rightarrow$ principal components are uncorrelated

$\rightarrow$ transformation into the eigenbasis leads to uncorrelated "complex" features

(3) interpretation of principal components w.r.t. variance [2]

$$\begin{array}{ccccccccccc} \lambda_1 & > & \lambda_2 & > & \lambda_3 & > & \ldots\ldots & > & \lambda_{N-1} & > & \lambda_N \\ \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\ \underline{\mathbf{e}}_1 & & \underline{\mathbf{e}}_2 & & \underline{\mathbf{e}}_3 & & & & \underline{\mathbf{e}}_{N-1} & & \underline{\mathbf{e}}_N \end{array} \tag{2.14}$$

[2]slide: interpretation of PCs

$$\boxed{\text{direction of largest variance}} \longrightarrow ? \longrightarrow \boxed{\text{direction of smallest variance}}$$

$\underline{\mathbf{e}}_j$ 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 $\underline{\mathbf{C}}$

$$\underline{\mathbf{x}} = \underbrace{a_1}_{\underline{\mathbf{e}}_1^T \underline{\mathbf{x}}} \underline{\mathbf{e}}_1 + \underbrace{a_2}_{\underline{\mathbf{e}}_2^T \underline{\mathbf{x}}} \underline{\mathbf{e}}_2 + \ldots + \underbrace{a_N}_{\underline{\mathbf{e}}_N^T \underline{\mathbf{x}}} \underline{\mathbf{e}}_N \tag{2.15}$$

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

$$\widetilde{\underline{\mathbf{x}}} = a_1 \underline{\mathbf{e}}_1 + a_2 \underline{\mathbf{e}}_2 + \ldots + a_M \underline{\mathbf{e}}_M \tag{2.16}$$

then yields an approximation error:

$$(\underline{\mathbf{x}} - \widetilde{\underline{\mathbf{x}}})^2 = \sum_{j=M+1}^{N} a_j^2 \tag{2.17}$$

$$\boxed{\text{This reconstruction error is minimal w.r.t. all possible projections into } M\text{-dimensional subspaces}}$$

2.1.4 Summary of <u>P</u>rincipal <u>C</u>omponent <u>A</u>nalysis (PCA)

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

(1) normalization to zero mean

$$\underline{\mathbf{m}} = \frac{1}{p} \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} \rightsquigarrow \widehat{\underline{\mathbf{x}}}^{(\alpha)} \leftarrow \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{m}} \tag{2.18}$$

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

$$C_{ij} = \frac{1}{p} \sum_{\alpha=1}^{p} \widehat{\mathrm{x}}_i^{(\alpha)} \widehat{\mathrm{x}}_j^{(\alpha)} \tag{2.19}$$

(3) solve the eigenvalue problem

$$\underline{\mathbf{C}}\underline{\mathbf{e}} = \lambda \underline{\mathbf{e}} \tag{2.20}$$

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.

Example – PCA of the Leptograpsus data: $\square$[3]

5 dimensions → 5 principal components (PCs)

$$\left.\begin{array}{ll} \text{PC1:} & \text{"size"} \\ \text{PC2:} & \text{"sex"} \\ \text{PC3:} & \text{"color/subspecies"} \end{array}\right\} \begin{array}{c}\text{relative importance of elementary} \\ \text{features can be "read out" from} \\ \text{the feature vectors}\end{array}$$

⇒ example for successful visualization

⇒ example for a successful preprocessing for classification

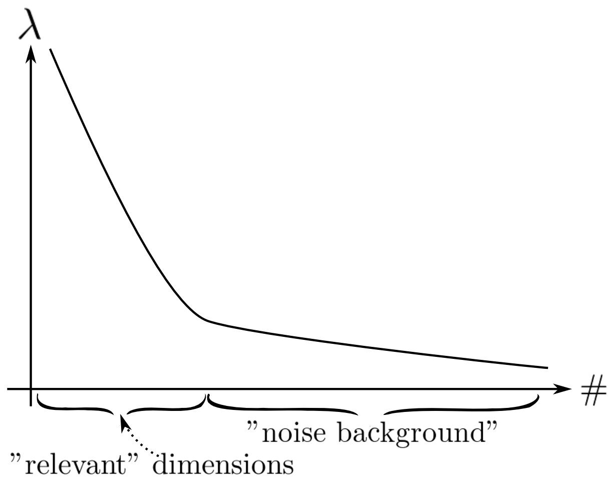


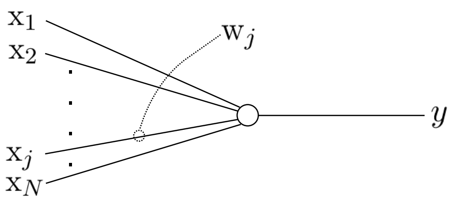
Figure 5: scree plots (ordering of eigenvalues by size):

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

[3]slide: projections for Leptograpsus data

2.2 Hebbian Learning for Linear Neurons



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

Figure 6: Linear connectionist neuron

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

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

```
initialization of weights (e.g. to small numbers)
choose learning rate $\varepsilon$
begin loop
    Choose an observation $\underline{\mathbf{x}}^{(\alpha)}$
    Change weights according to:
        $\underbrace{\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \underline{\mathbf{x}}^{(\alpha)}}_{\text{weights increase (decrease) if input and output are correlated (anticorrelated)}}$   (2.21)
end
```

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

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

$$\begin{aligned} \Delta \mathrm{w}_j &\approx \frac{\varepsilon}{p} \sum_{\alpha=1}^{p} y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{x}_j^{(\alpha)} \\ &= \frac{\varepsilon}{p} \sum_{\alpha=1}^{p} \sum_{k=1}^{N} \mathrm{w}_k \mathrm{x}_k^{(\alpha)} \mathrm{x}_j^{(\alpha)} \qquad (2.22) \\ &= \varepsilon \sum_{k=1}^{N} \mathrm{w}_k C_{kj} \end{aligned}$$

$$\Delta \underline{\mathbf{w}} = \varepsilon \underline{\mathbf{C}\mathbf{w}} \rightsquigarrow \text{"analysis" of the covariance matrix} \qquad (2.23)$$

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

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

$$\underline{\mathbf{w}} = a_1\underline{\mathbf{e}}_1 + a_2\underline{\mathbf{e}}_2 + \ldots + a_N\underline{\mathbf{e}}_N \leftarrow \text{ corresponding eigenvectors} \tag{2.24}$$

then:

$$\Delta a_j = \varepsilon\lambda_j a_j \tag{2.25}$$

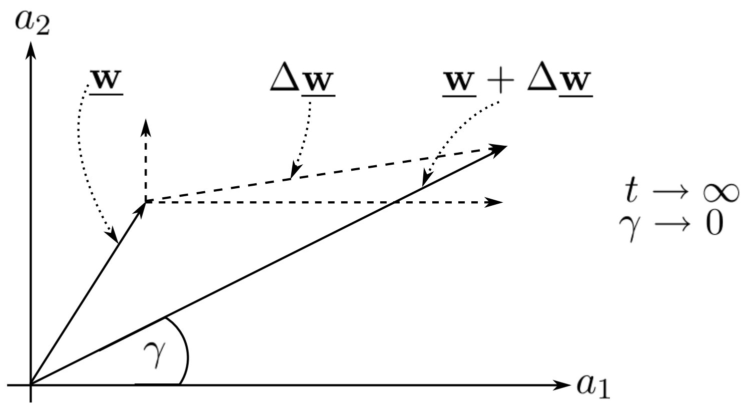


Figure 7: Learning via Ojas rule

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

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

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

$$\Delta \mathrm{w}_j = \varepsilon y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \left\{ \underbrace{\mathrm{x}_j^{(\alpha)}}_{\substack{\text{Hebbian}\\ \text{learning}}} \underbrace{- y_{\left(\underline{\mathbf{x}}^{(\alpha)};\underline{\mathbf{w}}\right)} \mathrm{w}_j}_{\substack{\text{decay}\\ \text{term}}} \right\} \tag{2.26}$$

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

proof: supplementary material

2.3 Kernel Principal Component Analysis

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

2.3.1 Non-linear Manifolds

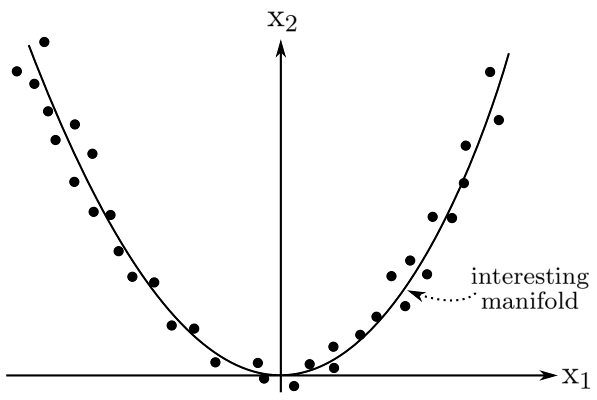


Figure 8: Example of a nonlinear dependency

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

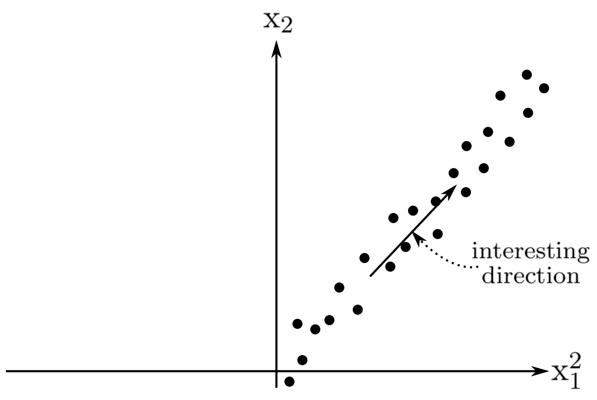


Figure 9: Nonlinear dependency becomes linear in transformed space

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

(1) transformation into an "appropriate" feature space

$$\underline{\phi} : \underline{\mathbf{x}} \to \underline{\phi}_{(\underline{\mathbf{x}})} \tag{2.27}$$

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

Problem: Some feature spaces may be extremely high-dimensional

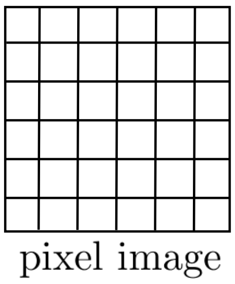


Figure 10: Pixel image as a high dimensional patterns

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

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

$$\text{e.g. } d = 2 : \mathrm{x}_1^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_3, \mathrm{x}_2\mathrm{x}_3, \mathrm{x}_3^2, \dots \tag{2.28}$$

$\rightsquigarrow$ number of dimensions

$$\frac{(N + d - 1)!}{d!(N - 1)!} = O_{\left(N^d\right)} \tag{2.29}$$

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*)

2.3.2 The Kernel Trick

Idea: perform PCA in a suitable feature space

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}_{(\underline{\mathbf{x}})} \tag{2.30}$$

Kernel trick:

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

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

$$\underline{\phi}^T_{(\underline{\mathbf{x}})}\underline{\phi}_{(\underline{\mathbf{x}}')} \longleftrightarrow k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} \tag{2.31}$$

Mercer's theorem: every positive definite kernel k corresponds to a scalar product in some metric feature space[4]

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*
$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \frac{1}{\lvert\underline{\mathbf{x}}-\underline{\mathbf{x}}'+\varepsilon\rvert^N}$	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

2.3.3 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:

$$\frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)} = \underline{\mathbf{0}} \rightsquigarrow \text{ zero mean data} \tag{2.32}$$

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

$$C_{ij} = \frac{1}{p}\sum_{\alpha=1}^{p}\mathrm{x}_i^{(\alpha)}\mathrm{x}_j^{(\alpha)} \Rightarrow \underline{\mathbf{C}} = \frac{1}{p}\sum_{\alpha=1}^{p}\underline{\mathbf{x}}^{(\alpha)}\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \tag{2.33}$$

[4]for details see supplementary material and MI I, chapt. 2.2.4

PCA requires a solution of the eigenvalue problem:

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \tag{2.34}$$

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

$$\underbrace{\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)}}_{\substack{\text{always possible, because principal components}\\ \text{lie in the subspace of the data}}} \tag{2.35}$$

Insertion into the eigenvalue equation leads to:

$$\frac{1}{p}\sum_{\alpha,\beta=1}^{p} a_k^{(\beta)} \left[\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)}\right] \underline{\mathbf{x}}^{(\alpha)} = \lambda_k \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)} \tag{2.36}$$

Projections onto the directions given by data points $\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T$:

$$\frac{1}{p}\sum_{\alpha,\beta=1}^{p} a_k^{(\beta)} \left[\left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)}\right]\left[\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T \underline{\mathbf{x}}^{(\alpha)}\right] = \lambda_k \sum_{\beta=1}^{p} a_k^{(\beta)} \left[\left(\underline{\mathbf{x}}^{(\gamma)}\right)^T \underline{\mathbf{x}}^{(\beta)}\right] \tag{2.37}$$

Formulation of the eigenvalue problem in terms of scalar products:

$$K_{\alpha\beta} := \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)} \tag{2.38}$$

in matrix notation:

$$\underline{\mathbf{K}}^2 \underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{K}}\underline{\mathbf{a}}_k \tag{2.39}$$

Remark: $\underline{\mathbf{K}}$ is a positive semidefinite matrix! This means, it has only non-negative eigenvalues (but potentially 0) $\rightarrow$ can be inverted $\rightarrow$ multiply by K^{-1} from the left. $\rightarrow$ solutions of the transformed problem differ only by components orthogonal to the solution space (see e.g. **Bishop2006**).

Arbitrary vector $\underline{\mathbf{y}}$:

$$\begin{aligned}\underline{\mathbf{y}}^T \underline{\mathbf{K}}\underline{\mathbf{y}} &= \sum_{\alpha,\beta=1}^{p} y^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)} y^{(\beta)} \\ &= \left(\sum_{\alpha=1}^{p} y^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}\right)^2 \\ &\geq 0\end{aligned} \tag{2.40}$$

Translated eigenvalue problem:

$$\boxed{\underline{\mathbf{K}}\underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{a}}_k} \tag{2.41}$$

$\underline{\mathbf{K}}$:	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

$$\begin{aligned} \underline{\mathbf{e}}_k^2 &= \sum_{\alpha,\beta=1}^{p} a_k^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^T \underline{\mathbf{x}}^{(\beta)} a_k^{(\beta)} \\ &= \underline{\mathbf{a}}_k^T \underline{\mathbf{K}} \underline{\mathbf{a}}_k \\ &= \lambda_k \underline{\mathbf{a}}_k^2 p \\ &\overset{!}{=} 1 \end{aligned} \tag{2.42}$$

$$\boxed{\underline{\mathbf{a}}_k^{\text{norm.}} = \frac{1}{\sqrt{p}\sqrt{\lambda_k}|\underline{\mathbf{a_k}}|}\underline{\mathbf{a}}_k} \tag{2.43}$$

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

$$\begin{aligned} u_k &= \underline{\mathbf{e}}_k^T \cdot \underline{\mathbf{x}} \\ &= \sum_{\beta=1}^{p} a_k^{(\beta)} \underbrace{\left[\left(\underline{\mathbf{x}}^{(\beta)}\right)^T \cdot \underline{\mathbf{x}}\right]}_{\text{scalar product}} \end{aligned} \tag{2.44}$$

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

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

$$\frac{1}{p}\sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} \overset{!}{=} 0 \nrightarrow \frac{1}{p}\sum_{\alpha=1}^{p} \underline{\phi}_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} = \underline{\mathbf{0}} \tag{2.45}$$

Solution: calculation of "centered" kernel matrix elements

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

$$\underbrace{\underline{\phi}_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)}}_{\substack{\text{"centered"} \\ \text{feature vectors}}} = \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} - \frac{1}{p}\sum_{\gamma=1}^{p} \underbrace{\underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\gamma)}\right)}}_{\substack{\text{uncentered} \\ \text{feature vectors}}} \tag{2.46}$$

$$\begin{aligned}
K_{\alpha\beta} &= \underline{\phi}^T_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} \cdot \underline{\phi}_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} \\
&= \left(\underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} - \tfrac{1}{p} \sum_{\gamma=1}^{p} \underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\gamma)}\right)} \right) \left(\underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} - \tfrac{1}{p} \sum_{\delta=1}^{p} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\delta)}\right)} \right) \\
&= \underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} - \tfrac{1}{p} \sum_{\gamma=1}^{p} \underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\gamma)}\right)} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} \qquad (2.47) \\
&\quad - \tfrac{1}{p} \sum_{\delta=1}^{p} \underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\alpha)}\right)} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\delta)}\right)} + \tfrac{1}{p^2} \sum_{\delta=1}^{p} \underline{\widetilde{\phi}}^T_{\left(\underline{\mathbf{x}}^{(\gamma)}\right)} \underline{\widetilde{\phi}}_{\left(\underline{\mathbf{x}}^{(\delta)}\right)} \\
&= \widetilde{K}_{\alpha\beta} - \tfrac{1}{p} \sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} - \tfrac{1}{p} \sum_{\gamma=1}^{p} \widetilde{K}_{\alpha\gamma} + \tfrac{1}{p^2} \sum_{\gamma,\delta} \widetilde{K}_{\gamma\delta}
\end{aligned}$$

2.3.4 Summary of the Kernel-PCA Method

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

$$\widetilde{K}_{\alpha\beta} = \underbrace{k_{\left(\underline{\mathbf{x}}^{(\alpha)},\underline{\mathbf{x}}^{(\beta)}\right)}}_{\substack{\text{kernel}\\ \text{function}}} \qquad (2.48)$$

(2) normalize to zero mean

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p} \sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} - \frac{1}{p} \sum_{\gamma=1}^{p} \widetilde{K}_{\alpha\gamma} + \frac{1}{p^2} \sum_{\gamma,\delta} \widetilde{K}_{\gamma\delta} \qquad (2.49)$$

(3) solve the eigenvalue problem

$$\underline{\mathbf{K}} \underline{\widetilde{\mathbf{a}}}_k = p\lambda_k \underline{\widetilde{\mathbf{a}}}_k \qquad (2.50)$$

(4) normalize eigenvectors to unit length

$$\underline{\mathbf{a}}_k = \frac{1}{\sqrt{p\lambda_k}||\underline{\widetilde{\mathbf{a}}}_k|} |\underline{\widetilde{\mathbf{a}}}_k \qquad (2.51)$$

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

$$u_{k(\underline{\mathbf{x}}^{\delta})} = \sum_{\beta=1}^{p} a_k^{(\beta)} K_{\beta\delta} \leftarrow \substack{\text{use normalized}\\ \text{matrix element!}} \qquad (2.52)$$

Comments:

- kernel-PCA $\widehat{=}$ PCA in feature space

  $\rightarrow$ "principal components" are uncorrelated

  $\rightarrow$ λ_l: variance of the data along principal component k

  $\rightarrow$ mean-squared approximation error 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

- 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 $q < p$:

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

  $$\underline{\widehat{\mathbf{e}}}_k = \sum_{\gamma=1}^{q} \widehat{a}_k^{(\gamma)} \underline{\phi}\underbrace{\big(\underline{\mathbf{z}}^{(\gamma)}\big)}_{\substack{\text{new data}\\ \text{point}}} \qquad \text{approximation}$$

  quadratic error φ:

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

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

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

  e.g. using gradient-based methods

- kernel matrices may be very large

  $\rightarrow$ only eigenvectors to the largest eigenvalues are of interest

  $\rightarrow$ use specialized routines (**PressEtAl2007** chapt. 11.5 - 11.7)

$\square$[5]

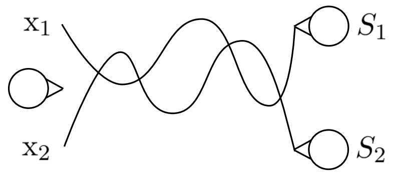
[5]slide: toy example (Schölkopf et al., MPI Biol. Cybern. Technical Report 44, Fig. 2)

2.4 Model-based Source Separation Techniques

2.4.1 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



$$\underbrace{\underline{\mathbf{x}}}_{\text{observations}} = \underbrace{\underline{\mathbf{A}}}_{\substack{\text{mixing}\\ \text{matrix}}} \underbrace{\underline{\mathbf{s}}}_{\text{sources}} \tag{2.53}$$

Figure 11: 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 A	$\{X^{(\alpha)}\},\ P_X(X^{(\alpha)})$ Data	*Approach B*
	$\Downarrow$	
$P_S(\hat{S}),\ \hat{S} = WX$	Models	$\hat{U}_i = f_i(e^T WX)$
	$\Downarrow$	
$D_{\mathrm{KL}}[P_S(\hat{S}), \prod_i P_S(\hat{S}_i)]$	Performance measure	$H(\hat{U})$
	$\Downarrow$	
$\min D_{\mathrm{KL}}$	Optimisation	$\max H(\hat{U})$

Table 1: Overview of the 2 approaches: Note that the two approaches are equivalent if the "transition functions" f_i match the marginal distributions 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

$$\underbrace{\widehat{\underline{\mathbf{s}}}}_{\text{"estimated sources"}} = \underbrace{\underline{\mathbf{W}}}_{\text{"unmixing matrix"}} \underbrace{\underline{\mathbf{x}}}_{\text{observations}} \tag{2.54}$$

such that the joint distribution of sources factorizes:

$$P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}}) = \prod_{i=1}^{N} P_{\underline{\mathbf{s}}}(\widehat{s_i}) \tag{2.55}$$

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

$$\underline{\mathbf{W}} = \underline{\mathbf{A}}^{-1} \tag{2.56}$$

Why not PCA? Decorrelation is not independence!

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

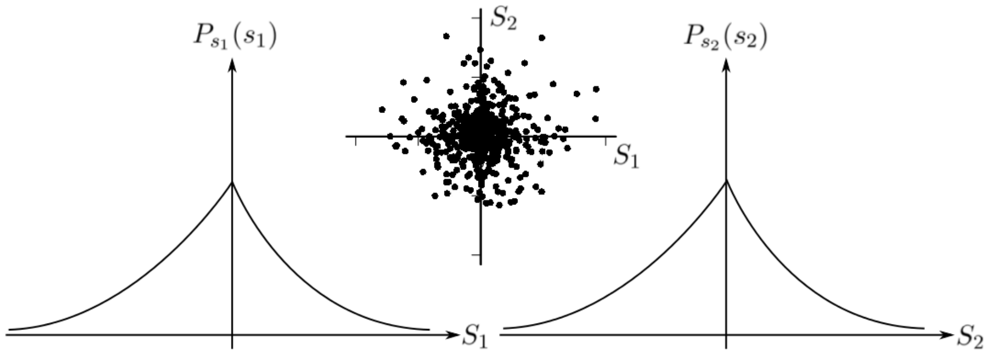
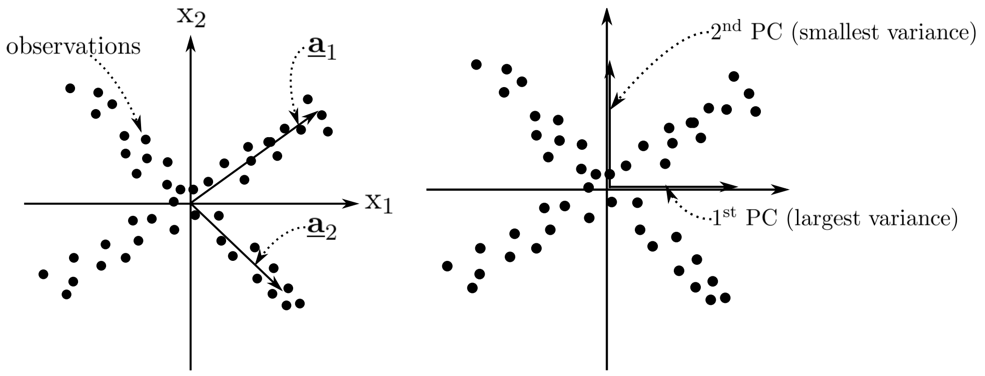


Figure 12: Marginal and joint distribution of independent variables

$$\underline{\mathbf{x}} = \underline{\mathbf{A}}\underline{\mathbf{s}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} a_{11}s_1 + a_{12}s_2 \\ a_{21}s_1 + a_{22}s_2 \end{pmatrix} \tag{2.57}$$

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

$$\begin{aligned} \underline{\mathbf{s}} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightsquigarrow \underline{\mathbf{x}} = \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} = \underline{\mathbf{a}}_1 \quad \text{1st source} \\ \underline{\mathbf{s}} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightsquigarrow \underline{\mathbf{x}} = \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} = \underline{\mathbf{a}}_2 \quad \text{2nd source} \end{aligned} \tag{2.58}$$



x_2
observations
$\underline{\mathbf{a}}_1$
x_1
$\underline{\mathbf{a}}_2$
2nd PC (smallest variance)
1st PC (largest variance)

Figure 13: 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

$$\begin{pmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{pmatrix} = \begin{pmatrix} \mathrm{w}_{11} & \mathrm{w}_{12} \\ \mathrm{w}_{21} & \mathrm{w}_{22} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix} \quad \widehat{=} \quad \begin{pmatrix} \widehat{s}_2 \\ \widehat{s}_1 \end{pmatrix} = \begin{pmatrix} \mathrm{w}_{21} & \mathrm{w}_{22} \\ \mathrm{w}_{11} & \mathrm{w}_{12} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix}$$

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

both alternatives are solutions to the unmixing problem

(2) amplitude of the sources

$$\begin{pmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{pmatrix} = \begin{pmatrix} \mathrm{w}_{11} & \mathrm{w}_{12} \\ \mathrm{w}_{21} & \mathrm{w}_{22} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix} \quad \widehat{=} \quad \begin{pmatrix} a\widehat{s}_1 \\ b\widehat{s}_2 \end{pmatrix} = \begin{pmatrix} a\mathrm{w}_{11} & a\mathrm{w}_{12} \\ b\mathrm{w}_{21} & b\mathrm{w}_{22} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix}$$

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

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.

$$\begin{aligned} P_{\underline{\mathbf{s}}}(\underline{\widehat{\mathbf{s}}}) &= \tfrac{1}{2\pi} \exp\left\{ - \tfrac{\underline{\widehat{\mathbf{s}}}^2}{2} \right\} \\ &= \underbrace{\left[\frac{1}{\sqrt{2\pi}} \exp\left\{ - \frac{\widehat{s}_1{}^2}{2} \right\} \right] \left[\frac{1}{\sqrt{2\pi}} \exp\left\{ - \frac{\widehat{s}_2{}^2}{2} \right\} \right]}_{\text{solution to the unmixing problem}} \end{aligned} \tag{2.59}$$

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

$$\begin{aligned} \underline{\widetilde{\mathbf{s}}} &= \underline{\mathbf{B}}\underline{\widehat{\mathbf{s}}} \\ &= \underline{\mathbf{B}}\underline{\mathbf{W}}\underline{\mathbf{x}} \\ &= \underline{\mathbf{W}}'\underline{\mathbf{x}} \end{aligned} \tag{2.60}$$

$$\begin{aligned} \underline{\widetilde{\mathbf{s}}}^2 := \|\underline{\widetilde{\mathbf{s}}}\|_2^2 &= \sum_{i=1}^{2} \widetilde{s}_i^2 \\ &= \sum_{i,k,l=1}^{2} B_{ik}\widehat{s}_k B_{il}\widehat{s}_l \\ &= \sum_{k,l=1}^{2} \widehat{s}_k \left(\sum_{i=1}^{2} B_{ki}^T B_{il} \right) \widehat{s}_l \\ &= \underline{\widehat{\mathbf{s}}}^T \underbrace{\left(\underline{\mathbf{B}}^T \underline{\mathbf{B}} \right)}_{\overset{!}{=}\underline{\mathbf{1}}} \underline{\widehat{\mathbf{s}}} \\ &= \underline{\widehat{\mathbf{s}}}^2 \end{aligned} \tag{2.61}$$

$$\underline{\widetilde{\mathbf{s}}} = \underline{\mathbf{W}}'\underline{\mathbf{x}} \tag{2.62}$$

$$\begin{aligned} P_{\underline{\mathbf{s}}}(\underline{\widetilde{\mathbf{s}}}) &= \tfrac{1}{2\pi} \exp\left\{ - \tfrac{\underline{\widetilde{\mathbf{s}}}^2}{2} \right\} \\ &= \underbrace{\left[\frac{1}{\sqrt{2\pi}} \exp\left\{ - \frac{\widetilde{s}_1{}^2}{2} \right\} \right] \left[\frac{1}{\sqrt{2\pi}} \exp\left\{ - \frac{\widetilde{s}_2{}^2}{2} \right\} \right]}_{\text{also solution to the unmixing problem}} \end{aligned} \tag{2.63}$$

2.4.2 The Infomax Principle

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

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

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

estimation of $\underline{\mathbf{W}} \widehat{=}$ estimation of the probability density $P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}})$

$\rightsquigarrow$ parametrized density estimate

Approach 1: direct model selection (cf. section 5.3)

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

$$\widehat{P}_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}}) = \prod_{i=1}^{N} \widehat{P}_{s_i}(\widehat{s}_i) \leftarrow \text{ ICA assumption} \tag{2.64}$$

This naturally leads to formulation of the following cost function

$$D_{KL} = \int d\widehat{\underline{\mathbf{s}}} P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}}) \ln \frac{P_{\underline{\mathbf{s}}}(\widehat{\underline{\mathbf{s}}})}{\prod_{i=1}^{N} \widehat{P}_{s_i}(\widehat{s}_i)} \overset{!}{=} \min \tag{2.65}$$

which could be directly optimized.

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.[6] 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*

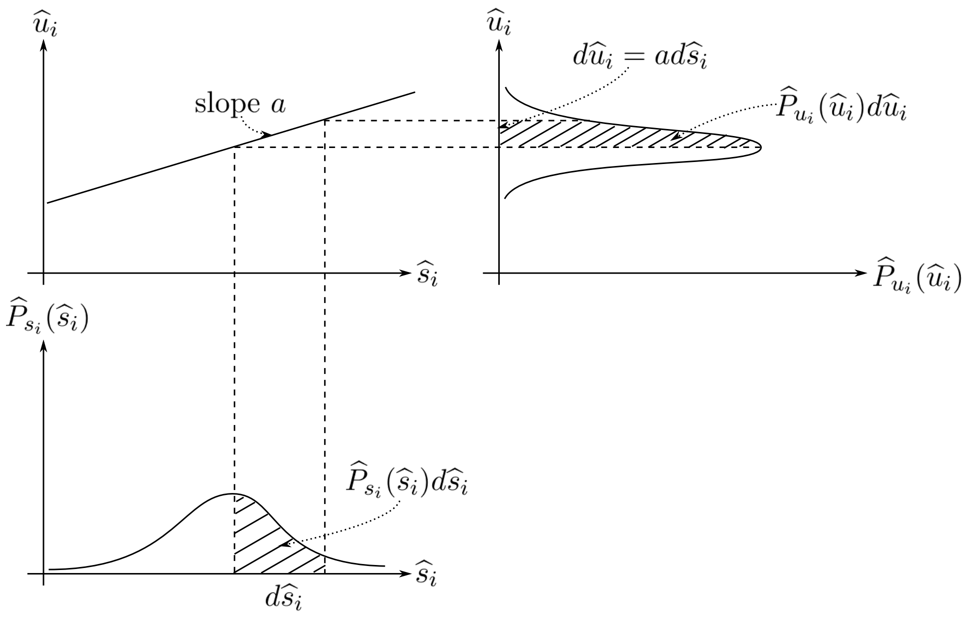
$$\widehat{P}_{u_i}(\widehat{u}_i) d\widehat{u}_i \ = \ \widehat{P}_{s_i}(\widehat{s}_i) d\widehat{s}_i. \tag{2.66}$$

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

$$\widehat{P}_{u_i}(\widehat{u}_i) \quad = \quad \left| \frac{d\widehat{s}_i}{d\widehat{u}_i} \right| \widehat{P}_{s_i}(\widehat{s}_i) \quad = \quad \frac{1}{\left| \widehat{f}_i'(\widehat{s}_i) \right|} \widehat{P}_{s_i}(\widehat{s}_i) \tag{2.67}$$

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 14. Applying the principle

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



$\widehat{u}_i$
slope a
$\widehat{s}_i$
$\widehat{u}_i$
$d\widehat{u}_i = a d\widehat{s}_i$
$\widehat{P}_{u_i}(\widehat{u}_i) d\widehat{u}_i$
$\widehat{P}_{u_i}(\widehat{u}_i)$
$\widehat{P}_{s_i}(\widehat{s}_i)$
$\widehat{P}_{s_i}(\widehat{s}_i) d\widehat{s}_i$
$\widehat{s}_i$
$d\widehat{s}_i$

Figure 14: 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:

$$\widehat{P}_{u_i}(\widehat{u}_i) \;=\; \frac{1}{\left|\widehat{f}_i'(\widehat{s}_i)\right|}\widehat{P}_{s_i}(\widehat{s}_i) \;\overset{!}{=}\; const \qquad \Rightarrow \qquad \left|\widehat{f}_i'(\widehat{s}_i)\right| = a\widehat{P}_{s_i}(\widehat{s}_i) \quad (2.68)$$

and therefore

$$\Rightarrow \widehat{f}_i(\widehat{s}_i) = \int_{-\infty}^{\widehat{s}_i} dy\; a\widehat{P}_{s_i}(y) \quad (2.69)$$

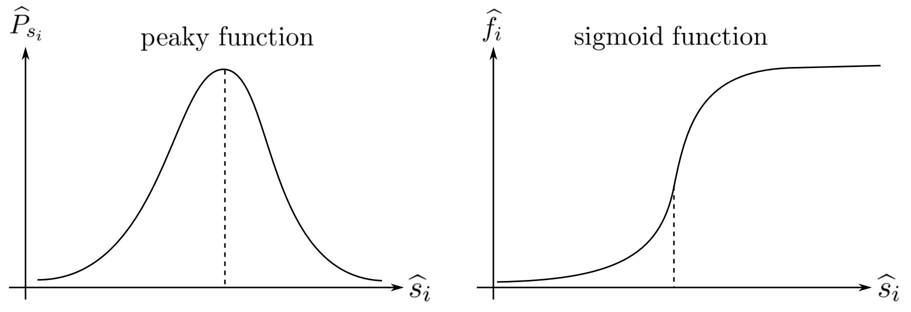


Figure 15: density (pdf) and corresponding distribution function (cdf)

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

$$\begin{aligned} D_{KL} &= \int d\underline{\mathbf{s}} P_{\widehat{\underline{\mathbf{s}}}} \ln \frac{P_{\widehat{\underline{\mathbf{s}}}}}{\prod_i P_{\widehat{s}_i}} && (2.70)\\ &= \int d\underline{\mathbf{s}} P_{\widehat{\underline{\mathbf{s}}}} \ln \frac{P_{\widehat{\underline{\mathbf{s}}}} \quad \prod_i \frac{1}{f_i'}}{\prod_i P_{\widehat{s}_i} \quad \frac{1}{f_i'}} \quad = \int d\underline{\mathbf{s}} P_{\widehat{\underline{\mathbf{s}}}} \ln \frac{P_{\widehat{\underline{\mathbf{u}}}}}{\prod_i P_{\widehat{u}_i}} && (2.71)\\ &= \underbrace{\int d\widehat{\underline{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}})}_{\substack{\text{negative entropy}\\ \text{of the transformed}\\ \text{(true) density}}} - \int d\widehat{\underline{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}}) \left(\underbrace{\ln \prod_{i=1}^{N} \widehat{P}_{u_i}(\widehat{u}_i)}_{\text{const. for 'flat' } u_i} \right) && (2.72) \end{aligned}$$

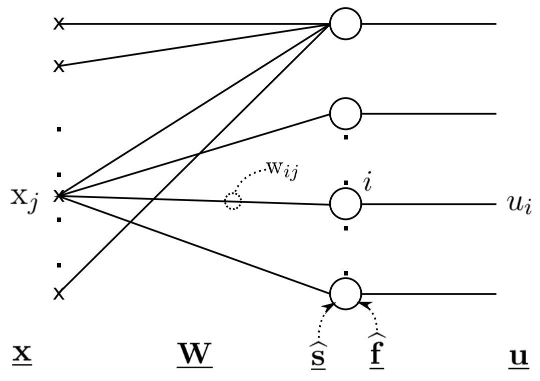
This motivates the socalled *Infomax principle* (**BellSejnowski1995**)

$$H = - \int d\widehat{\underline{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}}) \overset{!}{=} \max \tag{2.73}$$

using the transformed estimated sources

$$\widehat{u}_i = \widehat{f}_i \underbrace{\left(\underline{\mathbf{e}}_i^T \underline{\mathbf{W}} \underline{\mathbf{x}} \right)}_{\widehat{\underline{\mathbf{s}}}} \tag{2.74}$$

2.4.3 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

$$P_{\underline{\mathbf{u}}}(\widehat{\underline{\mathbf{u}}}) d\widehat{\underline{\mathbf{u}}} = P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \tag{2.75}$$

$$\begin{aligned} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) &= \left|\tfrac{d\underline{\mathbf{x}}}{d\underline{\widehat{\mathbf{u}}}}\right| P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \\ &= \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{\left|\frac{d\underline{\widehat{\mathbf{u}}}}{d\underline{\mathbf{x}}}\right|} = \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{|\mathbf{M}|} \end{aligned} \tag{2.76}$$

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

$$\begin{aligned} \mathbf{M}_{ij} = \tfrac{\partial \widehat{u}_i}{\partial \mathrm{x}_j} &= \tfrac{\partial}{\partial \mathrm{x}_j} \widehat{f}_i\Bigg(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k\Bigg) \\ &= \mathrm{w}_{ij} \widehat{f}_i'\Bigg(\sum_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k\Bigg). \end{aligned} \tag{2.77}$$

We therefore obtain for the value of the functional determinant

$$|\mathbf{M}| = \left|\frac{\partial \underline{\widehat{\mathbf{u}}}}{\partial \underline{\mathbf{x}}}\right| = |\underline{\mathbf{w}}| \prod_{l=1}^{N} \widehat{f}_l'\left(\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k\right). \tag{2.78}$$

Inserting this into the Infomax cost function (2.73) gives

$$H = -\int d\underline{\widehat{\mathbf{u}}} P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \ln P_{\underline{\mathbf{u}}}(\underline{\widehat{\mathbf{u}}}) \tag{2.79}$$

$$= -\int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln \frac{P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}{|\mathbf{M}|} \tag{2.80}$$

$$= \underbrace{-\int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}})}_{\text{constant w.r.t. } \underline{\mathbf{w}}} + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln |\mathbf{M}| \tag{2.81}$$

and with (2.78) can be written in terms explicitly depending on w:

$$H = const + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \ln |\underline{\mathbf{w}}| + \int d\underline{\mathbf{x}} P_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \sum_{l=1}^{N} \ln \widehat{f}_l'\left(\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k\right). \tag{2.82}$$

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*

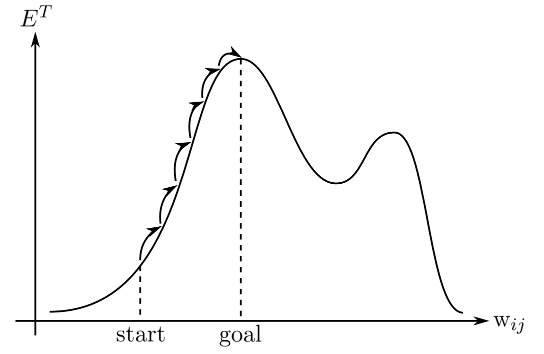
$$E^T = \ln |\underline{\mathbf{w}}| + \frac{1}{p} \sum_{\alpha=1}^{p} \sum_{l=1}^{N} \ln \widehat{f}_l'\left(\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right) \tag{2.83}$$

can be used for model selection using empirical data

$$\boxed{E^T \overset{!}{=} \max} \tag{2.84}$$

2.4.4 Learning by Gradient Ascent

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



$$\Delta \mathrm{w}_{ij} = \underbrace{\eta}_{\text{learning step}} \frac{\partial E^T}{\partial \mathrm{w}_{ij}}$$

Figure 16: Gradient descent using the training cost

Taking partial derivatives of the training cost (2.83) wrt. the model parameters w_{ij} yields

$$\frac{\partial E^T}{\partial \mathrm{w}_{ij}} = \underbrace{\frac{1}{p}\sum_{\alpha=1}^{p}\sum_{l=1}^{N}\frac{\partial}{\partial \mathrm{w}_{ij}}\left\{\ln \widehat{f}_l'\left(\sum_{k=1}^{N}\mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right)\right\}}_{\frac{1}{p}\sum\limits_{\alpha=1}^{p}\frac{\widehat{f}_i''\left(\sum\limits_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum\limits_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}\cdot \mathrm{x}_j^{(\alpha)}} + \underbrace{\frac{\partial}{\partial \mathrm{w}_{ij}}\left(\ln|\underline{\mathbf{w}}|\right)}_{\left(\underline{\mathbf{w}}^{-1}\right)_{ji}} \tag{2.85}$$

with individual costs:

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

$$\frac{\partial e^{(\alpha)}}{\partial \mathrm{w}_{ij}} = \underbrace{\left(\underline{\mathbf{w}}^{-1}\right)_{ji}}_{\text{costly computation}} + \underbrace{\frac{\widehat{f}_i''\left(\sum\limits_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum\limits_{k=1}^{N}\mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}}_{:=\varphi_i^{(\alpha)}}\cdot \mathrm{x}_j^{(\alpha)} \tag{2.87}$$

this can be used for *batch-learning*:

$$\Delta \mathrm{w}_{ij} = \frac{\eta}{p}\sum_{\alpha=1}^{p}\frac{\partial e^{(\alpha)}}{\partial \mathrm{w}_{ij}} \tag{2.88}$$

or using *on-line-learning* (see algorithm 2).[7]

[7]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 $\mathrm{w}_{ij}$ according to: $\Delta \mathrm{w}_{ij}^{(t)} = \eta_t \frac{\partial e_t^{(\alpha)}}{\partial \mathrm{w}_{ij}}$
    $t \leftarrow t + 1$
end
```

2.4.5 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: $d\underline{\mathbf{w}}, \underline{\mathbf{w}}$

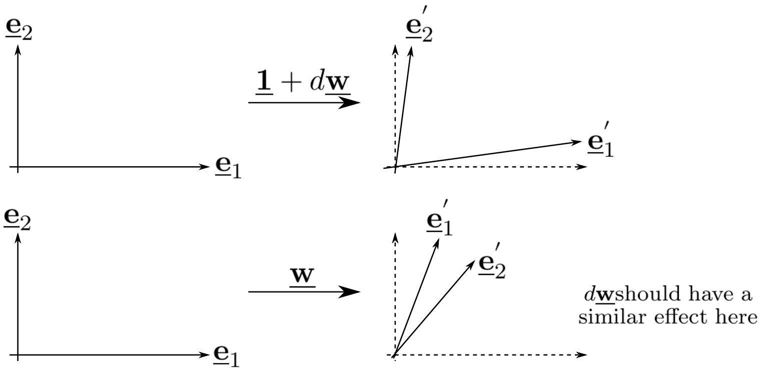

$\underline{\mathbf{e}}_2$
$\underline{\mathbf{e}}_1$
$\underline{\mathbf{1}} + d\underline{\mathbf{w}}$
$\underline{\mathbf{e}}_2'$
$\underline{\mathbf{e}}_1'$
$\underline{\mathbf{e}}_2$
$\underline{\mathbf{e}}_1$
$\underline{\mathbf{w}}$
$\underline{\mathbf{e}}_1'$
$\underline{\mathbf{e}}_2'$
$d\underline{\mathbf{w}}$should have a similar effect here

Figure 17: Illustration of gradient descent in transformed coordinate system

$$d\underline{\mathbf{Z}} = \underbrace{d\underline{\mathbf{w}}}_{\text{then do } d\underline{\mathbf{w}}} \cdot \underbrace{\underline{\mathbf{w}}^{-1}}_{\substack{\text{transfer back to } \underline{\mathbf{1}} \\ \Rightarrow \text{ makes learning} \\ \text{steps "comparable"}}} \tag{2.89}$$

This allows to do steepest ascent under normalized step size:

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

$$e_{(\underline{\mathbf{w}}+d\underline{\mathbf{w}})} = e_{(\underline{\mathbf{w}})} + \nabla e_{(\underline{\mathbf{w}})}^T dw \tag{2.90}$$

$$= e_{(\underline{\mathbf{w}})} + \underbrace{\eta}_{d\underline{\mathbf{w}}=\eta\underline{\mathbf{z}}_{\mathrm{w}}} \left[\nabla e_{(\underline{\mathbf{w}})}\right]^T \cdot \underbrace{\underline{\mathbf{z}}_{\mathrm{w}}}_{d\underline{\mathbf{w}}=\eta\underline{\mathbf{z}}_{\mathrm{w}}} \tag{2.91}$$

learning step:

$$\begin{aligned} d\underline{\mathbf{Z}} &= d\underline{\mathbf{w}} \cdot \underline{\mathbf{w}}^{-1} \\ &= \eta \underline{\mathbf{z}}_{\mathrm{w}} \cdot \underline{\mathbf{w}}^{-1} \end{aligned} \tag{2.92}$$

direction of steepest ascent under normalized step-size:

$$\left[\nabla e_{(\underline{\mathbf{w}})}\right]^T \underline{\mathbf{z}}_{\mathrm{w}} \overset{!}{=} \max \tag{2.93}$$

$$\left(\underline{\mathbf{z}}_{\mathrm{w}} \cdot \underline{\mathbf{w}}^{-1}\right)^2 \overset{!}{=} 1 \tag{2.94}$$

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

$$\sum_{i,j=1}^{N} \frac{\partial e}{\partial \mathrm{w}_{ij}} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ij} - \lambda \sum_{i,j,k,l=1}^{N} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ij} \left(\underline{\mathbf{w}}^{-1}\right)_{jl} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{ik} \left(\underline{\mathbf{w}}^{-1}\right)_{kl} \overset{!}{=} \max \tag{2.95}$$

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

$$\frac{\partial e}{\partial w_{\gamma s}} - 2\lambda \sum_{k=1}^{N} \left(\underline{\mathbf{z}}_{\mathrm{w}}\right)_{\gamma k} \sum_{l=1}^{N} \left(\underline{\mathbf{w}}^{-1}\right)_{kl} \left(\underline{\mathbf{w}}^{-1}\right)_{ls} \overset{!}{=} 0 \tag{2.96}$$

$$\frac{\partial e}{\partial \underline{\mathbf{w}}} = 2\lambda \underline{\mathbf{z}}_{\mathrm{w}} \underline{\mathbf{w}}^{-1} \left(\underline{\mathbf{w}}^{-1}\right)^T \tag{2.97}$$

$$\underline{\mathbf{z}}_{\mathrm{w}} = \frac{1}{2\lambda} \frac{\partial e}{\partial \underline{\mathbf{w}}} \underline{\mathbf{w}}^T \underline{\mathbf{w}} \tag{2.98}$$

yielding the direction for "natural" gradient ascent

$$\Delta \underline{\mathbf{w}} = \eta \overbrace{\frac{\partial e}{\partial \underline{\mathbf{w}}}}^{\text{"original" gradient}} \underbrace{\underline{\mathbf{w}}^T \underline{\mathbf{w}}}_{\text{normalization of step size}} \tag{2.99}$$

$$\boxed{\Delta \mathrm{w}_{ij} = \eta \sum_{l=1}^{N} \left\{ \delta_{il} + \frac{\widehat{f}_i''\left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum_{k=1}^{N} \mathrm{w}_{ik} \mathrm{x}_k^{(\alpha)}\right)} \sum_{k=1}^{N} \mathrm{w}_{lk} \mathrm{x}_k^{(\alpha)} \right\} \mathrm{w}_{lj}} \tag{2.100}$$

- 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

2.4.6 Some Practical Aspects

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

Bell-Sejnowski solution:

$$\Delta \mathrm{w}_{ii} = 0 \text{ and } \mathrm{w}_{ii} = 1 \text{ for all } i \tag{2.101}$$

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

$$\Delta \mathrm{w}_{ij} = \eta \frac{\widehat{f}_i''\left(\sum\limits_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum\limits_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)} \sum_{l \neq i}^{N} \left(\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}\right) \mathrm{w}_{lj} \tag{2.102}$$

(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:

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

$$\frac{\widehat{f}''_{(y)}}{\widehat{f}'_{(y)}} = 1 - 2\widehat{f}_{(y)} \tag{2.103}$$

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

Ansatz: Ignoring scaling of $\hat{s}_i$, i.e. using (2.100): for stationary state of natural gradient ascent (batch) the following has to hold:

$$\Delta \underline{\mathbf{w}}_{ij} \overset{!}{=} 0 \tag{2.104}$$

$$\delta_{il} \overset{!}{=} -\frac{1}{p} \sum_{\alpha=1}^{p} \underbrace{\frac{\widehat{f}_i''\left(\sum\limits_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}{\widehat{f}_i'\left(\sum\limits_{k=1}^{N} \mathrm{w}_{ik}\mathrm{x}_k^{(\alpha)}\right)}}_{\varphi_i\left(\widehat{s}_i^{(\alpha)}\right)} \cdot \underbrace{\sum_{k=1}^{N} \mathrm{w}_{lk}\mathrm{x}_k^{(\alpha)}}_{\widehat{s}_l^{(\alpha)}} \tag{2.105}$$

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

$i = l$:

$$-\frac{1}{p} \sum_{\alpha=1}^{p} \varphi_i\left(\widehat{s}_i^{(\alpha)}\right) \lambda_i s_i^{(\alpha)} \overset{!}{=} 1 \tag{2.106}$$

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:

$$\delta_{il} \frac{1}{p} \sum_{\alpha=1}^{p} \varphi_i\left(\widehat{s}_i^{(\alpha)}\right) \lambda_l s_l^{(\alpha)} \rightarrow \left\langle \varphi_i\left(\widehat{s}_i^{(\alpha)}\right) \lambda_l s_l^{(\alpha)} \right\rangle_{P_{\underline{\mathbf{s}}}} \tag{2.107}$$

$$\left\langle \varphi_i\left(\lambda_i s_i\right) \lambda_l s_l \right\rangle \underbrace{=}_{\substack{\text{statistical} \\ \text{independence}}} \left\langle \varphi_i\left(\lambda_i s_i\right) \right\rangle \left\langle \lambda_l s_l \right\rangle \overset{!}{=} 0 \tag{2.108}$$

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 $\underline{\mathbf{w}}$

2.5 Cost Function Based Source Separation Techniques

2.5.1 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[8]

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

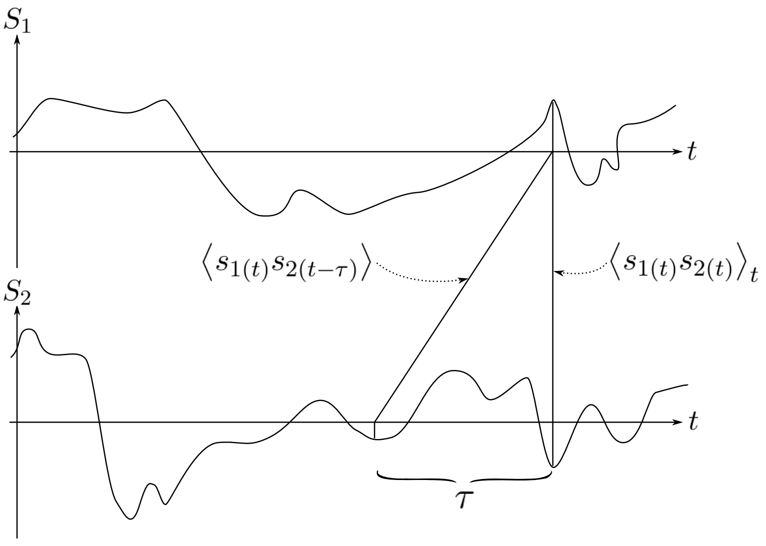

S_1
t
$\langle s_{1(t)} s_{2(t-\tau)} \rangle$
$\langle s_{1(t)} s_{2(t)} \rangle_t$
S_2
t
τ

Figure 18: 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 $\underline{\mathbf{x}} = \underline{\mathbf{A}}\underline{\mathbf{s}}$)

[8]All the presented approaches assume a *linear* mixing model, i.e. $\underline{\mathbf{x}} \approx \underline{\mathbf{A}}\underline{\mathbf{s}}$. 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 $\underline{\mathbf{x}} = f(\underline{\mathbf{s}}_1) = f(\underline{\mathbf{s_0} + \mathbf{\Delta s}}) \approx f(\underline{\mathbf{s_0}}) + \nabla f(\underline{\mathbf{s_0}})\Delta s$.

(1) PCA and sphering

- $\rightsquigarrow$ "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

$$\underline{\mathbf{C}}_{\mathrm{x}}^{(0)}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \text{ with } \big[\underline{\mathbf{C}}_{\mathrm{x}}^{\overbrace{(0)}^{\tau=0}}\big]_{ij} = \left\langle \mathrm{x}_{i(t)}\mathrm{x}_{j(t)} \right\rangle_t \tag{2.109}$$

- $\rightsquigarrow$ transformation into the eigenbasis and sphering

$$\underline{\mathbf{M}}_0 = \underbrace{\underline{\mathbf{\Lambda}}_0^{-1}}_{\substack{\text{diagonal matrix of}\\ \text{inverse eigenvalues}}} \cdot \underbrace{\underline{\mathbf{E}}_0}_{\substack{\text{matrix of}\\ \text{eigenvectors}}} \tag{2.110}$$

$$\underline{\mathbf{u}} = \underline{\mathbf{M}}_0 \underline{\mathbf{x}} \tag{2.111}$$

(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

$$\begin{aligned}
\left\langle s_{i(t)} s_{j(t)} \right\rangle_t &\underbrace{=}_{\substack{\text{statistical}\\ \text{independence}}} \delta_{ij} \\
&= \sum_{k,l=1}^{N} B_{ik} \underbrace{\left\langle u_{k(t)} u_{l(t)} \right\rangle_t}_{\substack{\overset{!}{=}\delta_{kl}\\ (\textit{cf. sphering})}} B_{lj}^T \\
&= \sum_{k=1}^{N} B_{ik} B_{kj}^T
\end{aligned} \tag{2.112}$$

$$\underline{\mathbf{B}} \cdot \underline{\mathbf{B}}^T = \underline{\mathbf{1}} \rightsquigarrow \text{orthogonal transformation} \tag{2.113}$$

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

- $\rightsquigarrow$ solve the eigenvalue problem

$$\underline{\mathbf{C}}_u^{(\tau)}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \text{ with } \left[\underline{\mathbf{C}}_u^{(\tau)}\right]_{ij} = \left\langle u_{i(t)} u_{j(t-\tau)} \right\rangle_t \tag{2.114}$$

$\rightsquigarrow$ transformation into the eigenbasis

$$\underline{\widehat{\mathbf{s}}} = \underbrace{\underline{\mathbf{E}}_\tau}_{\substack{\text{matrix of}\\ \text{eigenvectors}}} \underline{\mathbf{u}} \tag{2.115}$$

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 (2.110) and (2.115), this gives the transformation

$$\underline{\widehat{\mathbf{s}}} = \underline{\mathbf{E}}_\tau \underline{\mathbf{\Lambda}}_0^{-} \underline{\mathbf{E}}_0 \underline{\mathbf{x}} \tag{2.116}$$

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

$$\left[\underline{\mathbf{C}}_{\mathrm{x}}^{(\tau)}\right]_{ij} = \frac{1}{T} \sum_{t=0}^{T-1} \mathrm{x}_{i(t)} \mathrm{x}_{j(t-\tau)} \tag{2.117}$$

- 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.[9]

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:

$$E^T_{[\underline{\mathbf{W}}]} = \sum_\tau \underbrace{\alpha_\tau}_{\substack{\text{weighting}\\ \text{factors}\\ \text{(optional)}}} \sum_{i \neq j} \left(\underline{\mathbf{W}}\underline{\mathbf{C}}_{\mathrm{x}}^{(\tau)}\underline{\mathbf{W}}^T\right)^2_{ij} \tag{2.118}$$

optimization problem:

$$\begin{aligned} E^T_{[\underline{\mathbf{W}}]} &\overset{!}{=} \min && \text{minimize cross-correlations} \\ \left(\underline{\mathbf{W}}\underline{\mathbf{C}}_{\mathrm{x}}^{(0)}\underline{\mathbf{W}}^T\right)_{ii} &= 1 \text{ for all } i && \text{avoid trivial solutions} \end{aligned} \tag{2.119}$$

Comments:

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

- *documentation:* **VollgrafObermayer2006** Matlab-code implementing the algorithm can be found on the NI-website[10]
- *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

$$E^T_{[\underline{\mathbf{W}}]} = \sum_\tau \sum_{i \neq j} \left(\underline{\mathbf{W}} \underline{\mathbf{C}}_{\mathrm{x}}^{(\tau)} \underline{\mathbf{W}}^T \right)^2_{ij} \tag{2.120}$$

optimization problem:

$$\begin{aligned} &E^T_{[\underline{\mathbf{W}}]} \overset{!}{=} \min && \text{minimize cross-correlation} \\ &\text{invertability of } \underline{\mathbf{W}} && \text{to avoid trivial solutions} \end{aligned} \tag{2.121}$$

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**), □[11]

2.5.2 ICA and Projection Pursuit

"interesting" complex features

$\rightsquigarrow$ variance criterion $\Rightarrow$ PCA
important preprocessing method, but: scale sensitive solutions

$\rightsquigarrow$ higher moments: ok, but: Why not maximize non-Gaussianity?

ICA and non-Gaussianity

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

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

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

[10]www.ni.tu-berlin.de/menue/software/approximate_simultaneous_matrix_diagonalization_qdiag/
[11]slide: optical recording example

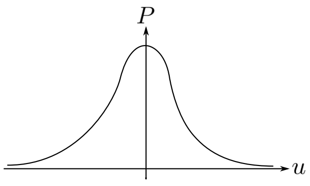
"The sum of independent random variables is 'more Gaussian' then the original variables"

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

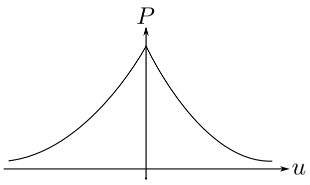
Measures for non-Gaussianity: The Gaussian is fully characterized by its first and second moments. Deviations from Gaussianity can therefore be quantified via the following measures:

(1) Kurtosis

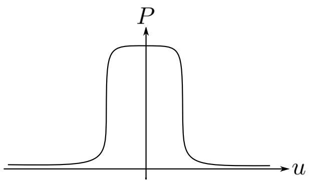
$$\text{kurt}(u) = \left\langle u^4 \right\rangle_{P_u(u)} - 3 \underbrace{\left(\left\langle u^2 \right\rangle_{P_u(u)} \right)^2}_{\substack{\text{would be 1 for} \\ \text{sphered data}}} \tag{2.122}$$



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



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



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

for independent random variables u_1 and u_2 we get:

$$\begin{aligned} \text{kurt}(u_1 + u_2) &= \text{kurt}(u_1) + \text{kurt}(u_2) \\ \text{kurt}(z_1 u_1) &= z_1^4 \, \text{kurt}(u_1) \end{aligned} \tag{2.123}$$

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

$$\begin{aligned} \widehat{s} &= \underline{\mathbf{z}}^T \underline{\mathbf{s}} \\ &= z_1 s_1 + z_2 s_2 \end{aligned} \tag{2.124}$$

$$\begin{aligned} \operatorname{var}(\widehat{s}) &= \left\langle \left(z_1 s_1 + z_2 s_2\right)^2 \right\rangle_{P_s(s)} \\ &= z_1^2 \langle s_1^2 \rangle + z_2^2 \langle s_2^2 \rangle \qquad (2.125) \\ &= z_1^2 + z_2^2 \end{aligned}$$

$$\operatorname{kurt}(\widehat{s}) = z_1^4 \operatorname{kurt}(s_1) + z_2^4 \operatorname{kurt}(s_2) \qquad (2.126)$$

search for "interesting" directions

$$\begin{aligned} \operatorname{kurt}(\widehat{s}) &\overset{!}{=} \max_{\underline{\mathbf{z}}} &&\leftarrow \text{search for the direction of optimal kurtosis} \\ z_1^2 + z_2^2 &\overset{!}{=} 1 &&\leftarrow \text{such that data remained sphered} \end{aligned} \qquad (2.127)$$

Result: (*see supplementary material*)

$$\underline{\mathbf{z}} = \begin{pmatrix} 0 \\ \pm 1 \end{pmatrix} \text{ or } \underline{\mathbf{z}} = \begin{pmatrix} \pm 1 \\ 0 \end{pmatrix} \qquad (2.128)$$

independent sources correspond to extrema of the kurtosis

(2) Negentropy

$$J_{(u)} := \underbrace{H^{\text{Gauss}}_{(u)}}_{\substack{\text{entropy of Gaussian} \\ \text{with variance } \sigma^2}} - \underbrace{H_{(u)}}_{\substack{\text{entropy of true} \\ \text{distribution} \\ \text{(variance } \sigma^2)}} \qquad (2.129)$$

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

(3) Approximations to the negentropy

$$J_{(u)} \approx \sum_{i=1}^{l} \underbrace{k_i}_{\substack{\text{some} \\ \text{constant}}} \left\{ \left\langle G_{(u)} \right\rangle_{\underbrace{P_u(u)}_{\substack{\text{true} \\ \text{density}}}} - \left\langle G_{(u)} \right\rangle_{\underbrace{\text{Gauss}}_{\substack{\text{reference:} \\ \text{Gaussian} \\ \text{density with} \\ \text{some variance}}}} \right\} \qquad (2.130)$$

common contrast functions:

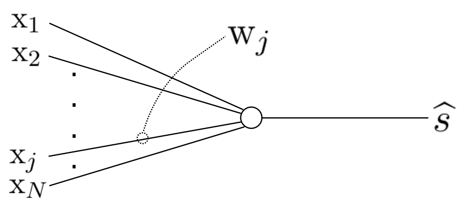
$$G_{1(u)} = \frac{1}{a} \log \cosh au \qquad \text{good general purpose function}$$

$$G_{2(u)} = -\exp\left(-\frac{u^2}{2}\right) \qquad \text{good only if sources are highly "super"-Gaussian i.e. many outliers}$$

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

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

Fixed point algorithm for one linear neuron: The following algorithm (alg. 3) implements `fastICA` for standardized data (mean=0, sd=1) to learn a single weight vector $\underline{\mathbf{w}}$:



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

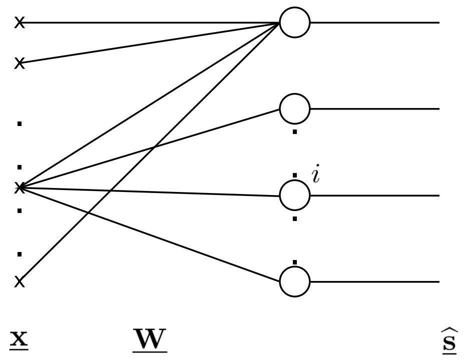
Algorithm 3: fixed-point algorithm for fastICA: single component

```
randomly initialize weight vector $\underline{\mathbf{w}}$ of unit length
begin
    $\underline{\mathbf{w}}^+ = \frac{1}{p}\left\{ \sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} G'_{\left(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(\alpha)}\right)} - \underline{\mathbf{w}} \sum_{\alpha=1}^{p} G''_{\left(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(\alpha)}\right)} \right\}$
    $\underline{\mathbf{w}} = \frac{\underline{\mathbf{w}}^+}{\|\underline{\mathbf{w}}^+\|}$
end
```

- 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



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

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 $i \in 1 \dots N$
begin
    repeat "symmetric orthogonalisation"
        $\underline{\mathbf{W}}^{(t)} \leftarrow \frac{3}{2}\underline{\mathbf{W}}^{(t)} - \frac{1}{2}\underline{\mathbf{W}}^{(t)}\left(\underline{\mathbf{W}}^{(t)}\right)^T \underline{\mathbf{W}}^{(t)}$
    until convergence
    for $i \in 1 \dots N$ do
        $\widehat{\underline{\mathbf{w}}}_i^{(t)} = \frac{1}{p}\left\{\sum_{\alpha=1}^{p} \underline{\mathbf{x}}^{(\alpha)} G'\left(\left(\underline{\mathbf{w}}_i^{(t)}\right)^T \underline{\mathbf{x}}^{(\alpha)}\right) - \underline{\mathbf{w}}_i^{(t)} \sum_{\alpha=1}^{p} G''\left(\left(\underline{\mathbf{w}}_i^{(t)}\right)^T \underline{\mathbf{x}}^{(\alpha)}\right)\right\}$
        $\underline{\mathbf{w}}_i^{(t+1)} = \frac{\widehat{\underline{\mathbf{w}}}_i^{(t)}}{\|\widehat{\underline{\mathbf{w}}}_i^{(t)}\|}$
    end
end
```

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://www.cis.hut.fi/projects/ica/fastica/index.shtml`
□[12] □[13]

[12]slide: sound demo: blind source separation
[13]slide: 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: $\{S_i\}, i = 1, \ldots, N$ describing the *state*[14] of the system and a *cost function*:

$$E : \underline{\mathbf{S}} \to \mathbb{R} \tag{3.1}$$

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

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

"*Stochastic* simulated annealing" (algorithm 5) implements an iterative procedure to find this optimal state.

Algorithm 5: Stochastic simulated annealing

```
initialization: $\underline{\mathbf{S}}_0, \beta_0, \tau, t = 0$
begin Annealing loop
    $t \leftarrow t + 1$
    begin State Update loop
        choose a new state $\underline{\mathbf{S}}$ randomly
        calculate difference in cost: $\Delta E = E_{(\underline{\mathbf{S}})} - E_{(\underline{\mathbf{S}}_{t-1})}$
        switch to $\underline{\mathbf{S}}$ with $p(\underline{\mathbf{S}}_{t-1} \to \underline{\mathbf{S}}) = \frac{1}{1+\exp(\beta_t \Delta E)}$
        otherwise keep the old state ($\underline{\mathbf{S}}_t = \underline{\mathbf{S}}_{t-1}$)
    end
    $\beta_t \leftarrow \tau \beta_{t-1}$ (practical but theoretically not optimal)
end
```

[14] 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 p and its dependence on the temperature parameter β:

- transition probability p

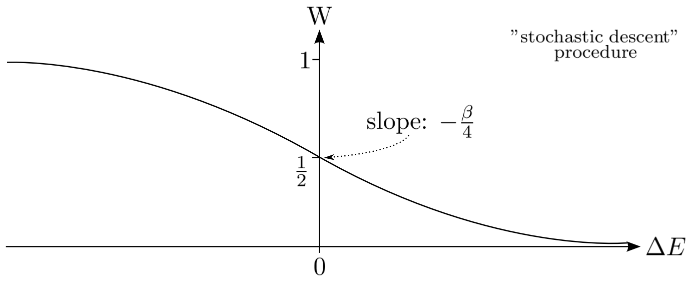


Figure 19: Transition probabilities

- limiting cases

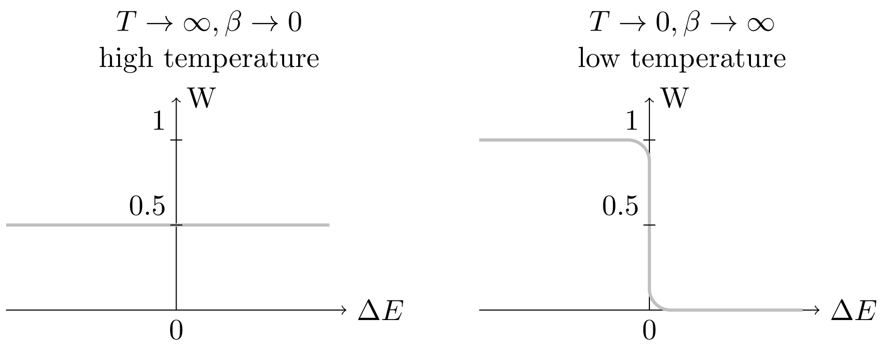


Figure 20: Transition probabilities in the two limiting cases

For many optimization problems, the cost function has local optima (see figure 21). 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)

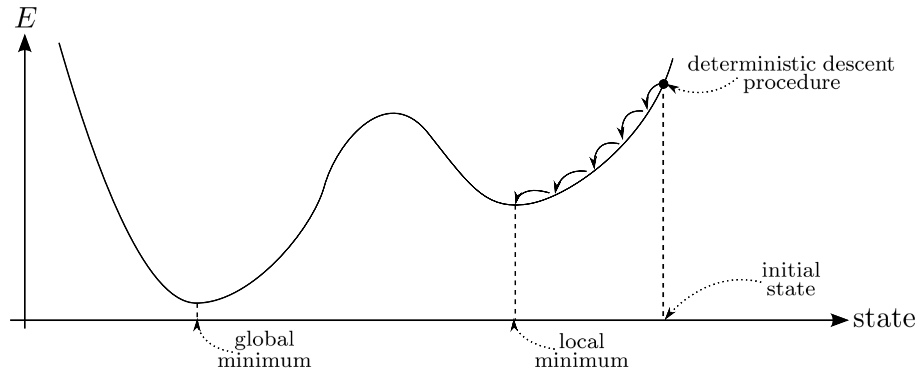


Figure 21: Cost function with local minima

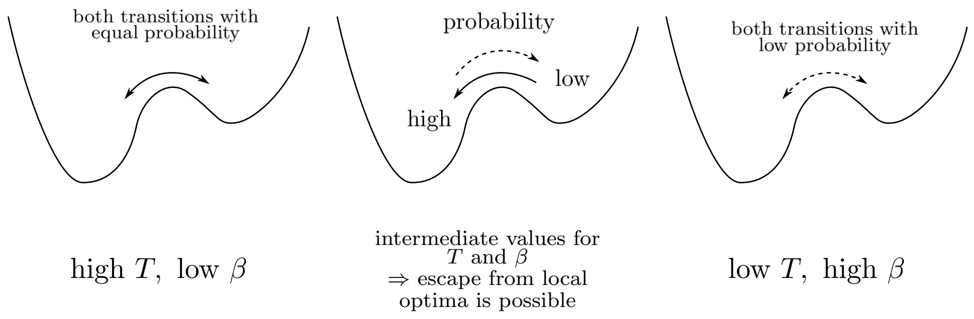

both transitions with equal probability
probability
low
high
both transitions with low probability
high T, low β
intermediate values for T and β ⇒ escape from local optima is possible
low T, high β

Figure 22: Effect of annealing on local minima

3.2 The Gibbs Distribution

The random (noisy) state changes cause state fluctuations even for constant T and β. 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

$$\Pi_{(\underline{\mathbf{S}},t)} \to \underbrace{P_{(\underline{\mathbf{S}})}}_{\text{stationary density}} \quad \text{for } t \to \infty \text{ (and constant } T, \beta) \tag{3.4}$$

calculation of $P_{(\underline{\mathbf{S}})}$ under the assumption of "detailed balance":

$$\underbrace{\overset{\text{probability of}}{\text{transition } \underline{\mathbf{S}} \to \underline{\mathbf{S}}'}}_{P_{(\underline{\mathbf{S}})} \mathrm{W}_{(\underline{\mathbf{S}} \to \underline{\mathbf{S}}')}} \underbrace{=}_{\text{detailed balance}} \underbrace{\overset{\text{probability of}}{\text{transition } \underline{\mathbf{S}}' \to \underline{\mathbf{S}}}}_{P_{(\underline{\mathbf{S}}')} \mathrm{W}_{(\underline{\mathbf{S}}' \to \underline{\mathbf{S}})}} \tag{3.5}$$

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

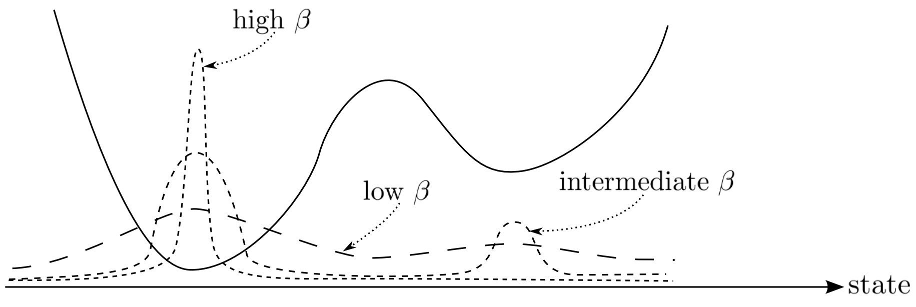
this condition is fulfilled for:

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

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 23: 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.[15] So a possible strategy might seem to evaluate $P_{(\underline{\mathbf{S}})}$ directly ($P_{(\underline{\mathbf{S}})}$ is known!). 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 $\underline{\mathbf{S}}$:

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

→ family of distributions parametrized by the *mean fields* e_k

→ 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. $\beta \to \infty$), 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 maximum). For the factorizing distribution $Q_{(\underline{\mathbf{S}})}$, these moments can be calculated easily:

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

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

$$
\begin{aligned}
&= \frac{1}{Z_Q}\left[\sum_{\underline{\mathbf{S}}/S_l} f_{(\underline{\mathbf{S}}/S_l)} \exp\Big(-\beta\sum_{k\neq l} e_k S_k\Big)\right]\left[\sum_{S_l} g_{(S_l)} \exp\Big(-\beta e_l S_l\Big)\right] \\
&= \frac{1}{Z_Q}\left[\sum_{\underline{\mathbf{S}}/S_l} f_{(\underline{\mathbf{S}}/S_l)} \exp\Big(-\beta\sum_{k\neq l} e_k S_k\Big)\right]\frac{\sum\limits_{S_l}\exp(-\beta e_l S_l)}{\sum\limits_{S_l}\exp(-\beta e_l S_l)}\left[\sum_{S_l} g_{(S_l)} \exp\Big(-\beta e_l S_l\Big)\right] \\
&= \Big\langle f_{(\underline{\mathbf{S}}/S_l)}\Big\rangle_Q \frac{\sum\limits_{S_l} g_{(S_l)}\exp(-\beta e_l S_l)}{\sum\limits_{S_l}\exp(-\beta e_l S_l)} \\
&= \underbrace{\Big\langle f_{(\underline{\mathbf{S}}/S_l)}\Big\rangle_Q \cdot \Big\langle g_{(S_l)}\Big\rangle_Q}_{\substack{\text{factorization of moments}\\ \rightarrow\text{uncorrelated variables}}}
\end{aligned}
$$

The first moments $\langle S_l\rangle_Q$ play a central role in the approximation of P by Q (see below) and usually have a tractable expression. E.g. for $\mathcal{S}=\{0,1\}$

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

The mean-field approximation:

$$
\begin{aligned}
P_{(\underline{\mathbf{S}})} &= \tfrac{1}{Z_p}\exp(-\beta E_p) && \text{true distribution} \\
Q_{(\underline{\mathbf{S}})} &= \tfrac{1}{Z_Q}\exp\big(-\beta\sum_k e_k S_k\big) && \substack{\text{approximation: family of}\\ \text{factorising distributions}} \\
e_k :&\quad \text{"mean fields"} && \substack{\text{parameters to}\\ \text{be determined}}
\end{aligned}
\tag{3.11}
$$

minimization of the KL-divergence:

$$
D_{KL} = \sum_{\underline{\mathbf{S}}} Q_{(\underline{\mathbf{S}})} \ln\frac{Q_{(\underline{\mathbf{S}})}}{P_{(\underline{\mathbf{S}})}} \overset{!}{=} \min \tag{3.12}
$$

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

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:

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

Algorithm 6: Mean Field Annealing

```
initialization: $\underline{\mathbf{S}}_0, \beta_0, t = 0$
begin Annealing loop
    calculate mean-fields: $e_k, k = 1, \ldots, N$
    calculate moments: $\langle S_k \rangle_Q, k = 1, \ldots, N$
    increase $\beta$
end
```

- $\Rightarrow$ $\beta \to \infty (T \to 0) : \langle S_k \rangle \to S_k^{\text{opt.}}$
  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.

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 24.

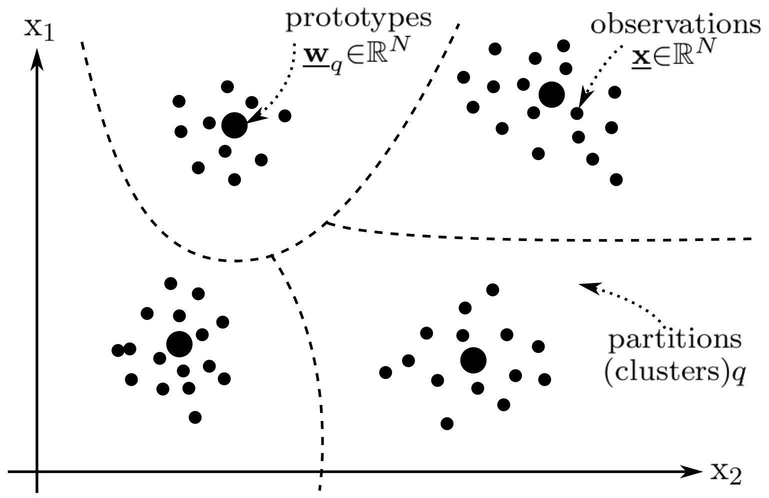

x1
prototypes
$\underline{\mathbf{w}}_q \in \mathbb{R}^N$
observations
$\underline{\mathbf{x}} \in \mathbb{R}^N$
partitions
(clusters)q
x2

Figure 24: The clustering problem

⇒ unsupervised formation of categories (partitions, clusters) according to predefined criteria

⇒ description of clusters by prototypes ← "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: $\underline{\mathbf{w}}_q, q = 1, \ldots, M$

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.

model selection:

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

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

The computations involved in the cost function for k-means clustering are illustrated in figure 25.

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 7)

center of mass is optimal: condition for extremal point:

$$\begin{aligned} \frac{\partial}{\partial \underline{\mathbf{w}}_q} \left\{ \frac{1}{2p} \sum_{q',\alpha} m_{q'}^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{q'}\right)^2 \right\} &= -\frac{1}{p} \sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right) \overset{!}{=} 0 \\ &\rightsquigarrow \underline{\mathbf{w}}_q = \frac{\sum_\alpha m_q^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}{\sum_\alpha m_q^{(\alpha)}} \end{aligned} \tag{4.5}$$

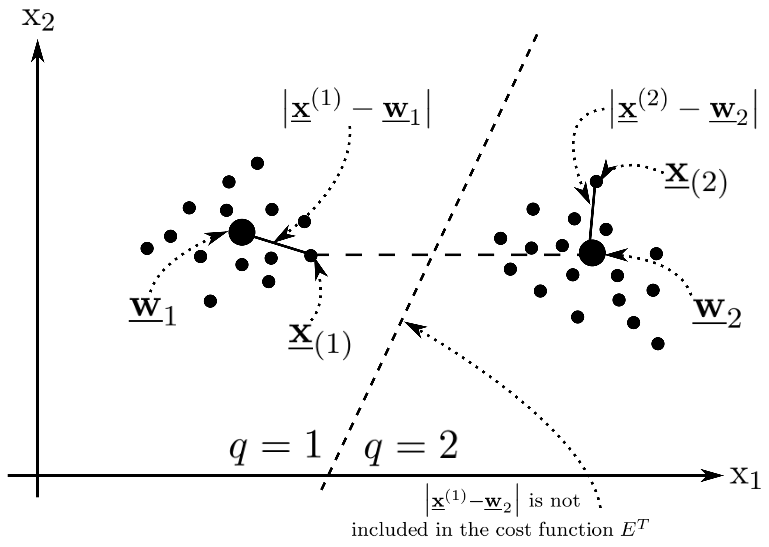

x_2
x_1
$|\underline{\mathbf{x}}^{(1)} - \underline{\mathbf{w}}_1|$
$|\underline{\mathbf{x}}^{(2)} - \underline{\mathbf{w}}_2|$
$\underline{\mathbf{x}}_{(2)}$
$\underline{\mathbf{w}}_1$
$\underline{\mathbf{x}}_{(1)}$
$\underline{\mathbf{w}}_2$
$q = 1$
$q = 2$
$|\underline{\mathbf{x}}^{(1)} - \underline{\mathbf{w}}_2|$ is not included in the cost function E^T

Figure 25: Illustration of the k-means cost function

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.

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 26 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

Algorithm 7: batch k-means

```
randomly initialize prototypes e.g. around the data's center of mass
begin loop
    (1) choose $m_q^{(\alpha)}$ ...
    ... such that $E^T$ is minimal for the given prototypes
        $m_q^{(\alpha)} \begin{cases} 1, & \text{if } q = \text{argmin}_\gamma \left|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma\right| \\ 0, & \text{else} \end{cases}$
    ⇒ assign every data point to its nearest prototype

    (2) choose $\underline{\mathbf{w}}_q$ ...
    ... such that $E^T$ is minimal for the -new- assignments
        $\underline{\mathbf{w}}_q = \frac{\sum_\alpha m_q^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}{\sum_\alpha m_q^{(\alpha)}}$
    ⇒ set $\underline{\mathbf{w}}_q$ to the center of mass of its assigned data
end
```

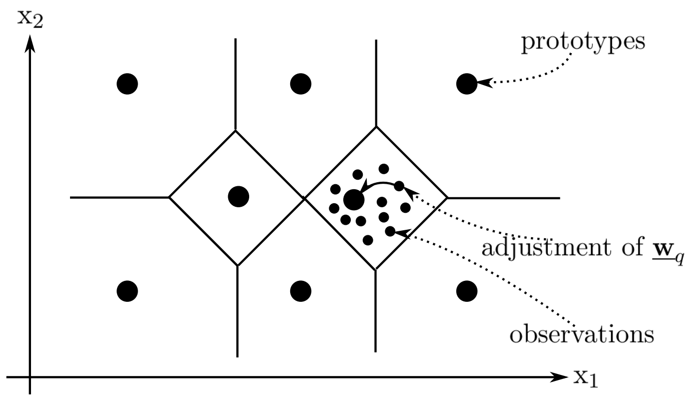


Figure 26: k-means and tesselation

"on-line" version of K-means Clustering: The k-means procedure can be implemented in an on-line fashion (algorithm 8) that is often more robust wrt. local minima than batch-learning and can be useful for streaming-data.

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 27.

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 $q$
        $$q = \operatorname*{argmin}_{\gamma} \left|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\gamma}\right|$$
    change corresponding prototype according to
        $$\Delta\underline{\mathbf{w}}_q = \eta\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\gamma}\big)$$
    change $\eta$
end
```

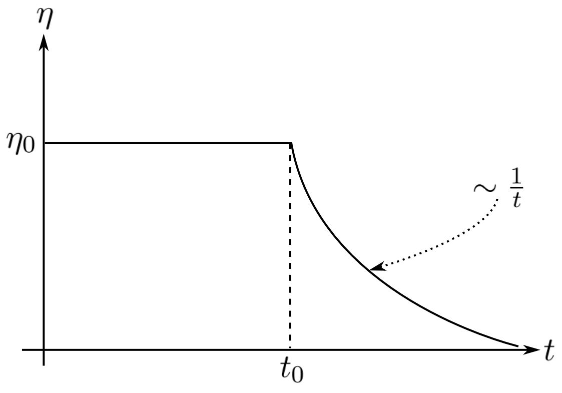
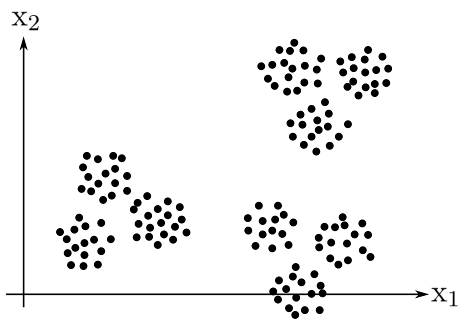


Figure 27: Annealing schedule

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

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

$\rightarrow$ exploit prior knowledge on E^T_{min} (e.g. $E^T_{\text{min}} \geq \sigma_{\text{noise}}$)

This prior knowledge can be exploited using the heuristic of *iterative refinement*, as illustrated in algorithm 9.

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.

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**

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.

Algorithm 9: iterative k-means refinement

```
initialization: $\underline{\mathbf{w}}_1 = \frac{1}{p}\sum_{\alpha} \underline{\mathbf{x}}^{(\alpha)}, \quad \underbrace{\left(E^T_{\min}\right)^*}_{\text{desired minimal average variance}}, M = 1$
begin loop
    if $E^T_{\min} < \left(E^T_{\min}\right)^*$ then STOP
    select partition $q \in \{1, \ldots, M\}$ with largest variance
        $q = \operatorname*{argmax}_{\gamma} \left( \frac{\sum_{\alpha} m_\gamma^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma\right)^2}{\sum_{\alpha} m_\gamma^{(\alpha)}} \right)$
    add a new prototype: $\underline{\mathbf{w}}_{M+1} = \underline{\mathbf{w}}_q + \underbrace{\underline{\eta}_q}_{\text{small random vector}}$
    $M \leftarrow M + 1$
    do k-means clustering with these $M$ prototypes
end
```

The number of equivalence class increases with increasing number of prototypes

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

- $\leadsto$ many equivalence classes with approximately equal cost
- $\leadsto$ different initial conditions and different sequences of pattern presentation lead to clustering solutions from different equivalence classes
- $\leadsto$ different sub-datasets lead to solutions from different equivalence classes

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 ($\sim$ 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 $\{d_{\alpha\alpha'}\}$

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & & p \\
\hline
1 & 0 & 1.7 & 0.99 & & 3.0 \\
2 & 1.7 & 0 & 0.3 & \dots & 0.1 \\
3 & 0.9 & 0.3 & 0 & & 0.2 \\
\vdots & & \vdots & & \ddots & \vdots \\
p & 3.0 & 0.1 & 0.2 & \dots & 0
\end{array}$$

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

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

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

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

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

cluster models

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

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:

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

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

model selection:

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

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}{2M}\sum_q \frac{\sum\limits_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\alpha')}\right)^2}{\sum\limits_{\alpha} m_q^{(\alpha)}} \\
&= \frac{1}{2M}\sum_q \frac{\sum\limits_{\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\limits_{\alpha} m_q^{(\alpha)}} \\
&= \frac{1}{2M}\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\limits_{\alpha'} m_q^{(\alpha')} \underline{\mathbf{x}}^{(\alpha')}}{\sum\limits_{\alpha'} m_q^{(\alpha')}}}_{\substack{\overset{!}{=}\underline{\mathbf{w}}_q \\ (\textit{cf. 4.1.2})}} + \sum_\alpha m_q^{(\alpha)} \left(\underline{\mathbf{x}}^{(\alpha)}\right)^2 \right\} \\
&= \frac{1}{M}\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}{M}\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}{M}\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}{M}\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]} \mathrel{\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:

$$E_{\left[\left\{m_q^{(\alpha)}\right\}\right]} = \frac{1}{M}\sum_q \frac{\sum\limits_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} d_{\alpha\alpha'}}{\sum\limits_{\alpha'} m_q^{(\alpha'}} \overset{!}{=} \min \tag{4.16}$$

$\Rightarrow$ 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 $\sum\limits_q m_q^{(\alpha)} = 1$

$\rightsquigarrow$ calculation of moments and mean-fields must be adapted

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

$\{\underline{\mathbf{m}}^{(\alpha)}\}$:	set of all M-dimensional binary vectors $(m_1^{(\alpha)}, m_2^{(\alpha)}, \dots, m_M^{(\alpha)})^T$ which fulfill the normalization condition: exactly one element equals 1.
$\mathscr{M}$:	$\{\underline{\mathbf{m}}^{(1)}\} \otimes \{\underline{\mathbf{m}}^{(2)}\} \otimes \ldots \otimes \{\underline{\mathbf{m}}^{(p)}\}$ 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

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

where

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

factorizing distribution

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

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).

$$\langle m_p^{(\gamma)} \rangle_Q = \frac{1}{Z_Q} \sum_{\mathscr{M}} m_p^{(\gamma)} \exp\Big\{ -\beta \sum_{q,\delta} m_q^{(\delta)} e_q^{(\delta)} \Big\} \tag{4.21}$$

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

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

and finally gives

$$\left\langle m_p^{(\gamma)}\right\rangle_Q = \frac{\left[\sum\limits_{\mathscr{M}_\gamma} \exp\left\{-\beta \sum\limits_{q,\delta\neq\gamma} m_q^{(\delta)} e_q^{(\delta)}\right\}\right] \cdot \left[\overbrace{\sum\limits_{\left\{\underline{\mathbf{m}}^{(\gamma)}\right\}} m_p^{(\gamma)}}^{\substack{\text{only term with}\\ m_p^{(\gamma)}=1\\ \text{remains}}} \exp\left\{-\beta \sum\limits_q m_q^{(\gamma)} e_q^{(\gamma)}\right\}\right]}{\underbrace{\left[\sum\limits_{\mathscr{M}_\gamma} \exp\left\{-\beta \sum\limits_{q,\delta\neq\gamma} m_q^{(\delta)} e_q^{(\delta)}\right\}\right]}_{\text{first terms cancel}} \cdot \left[\sum\limits_{\left\{\underline{\mathbf{m}}^{(\gamma)}\right\}} \exp\left\{-\beta \underbrace{\sum\limits_q m_q^{(\gamma)} e_q^{(\gamma)}}_{\substack{\text{only one term of this}\\ \text{sum remains for every}\\ \text{term of the previous sum}}}\right\}\right]}$$

$$= \frac{\exp\left\{-\beta m_p^{(\gamma)} e_p^{(\gamma)}\right\}}{\sum\limits_q \exp\left\{-\beta m_q^{(\gamma)} e_q^{(\gamma)}\right\}} = \frac{\exp\left\{-\beta e_p^{(\gamma)}\right\}}{\sum\limits_q \exp\left\{-\beta e_q^{(\gamma)}\right\}} \tag{4.23}$$

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

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

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

$$\frac{\partial\left\langle E^p\right\rangle_Q}{\partial e_q^{(\alpha)}} - \sum_{p,\gamma} \frac{\overbrace{\partial\left\langle m_p^{(\gamma)}\right\rangle_Q}^{\substack{\text{depends only on}\\ \text{data point } \gamma}}}{\partial e_q^{(\alpha)}} e_p^{(\gamma)} \overset{!}{=} 0 \tag{4.24}$$

$$\frac{\partial\left\langle E^p\right\rangle_Q}{\partial e_q^{(\alpha)}} - \sum_{p} \frac{\partial\left\langle m_p^{(\alpha)}\right\rangle_Q}{\partial e_q^{(\alpha)}} e_p^{(\alpha)} \overset{!}{=} 0 \tag{4.25}$$

for the mean-fields we obtain

$$\begin{aligned} e_p^{(\alpha)} \quad = \quad & \frac{1}{M}\left[\frac{1}{\sum\limits_\gamma \left\langle m_p^{(\gamma)}\right\rangle_Q}\left\{d_{\alpha\alpha'} - \frac{1}{\sum\limits_\gamma \left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\gamma)}\right\rangle_Q d_{\delta\delta}\right\}\right. \\ & + \frac{1}{\sum\limits_\gamma \left\langle m_p^{(\gamma)}\right\rangle_Q}\sum_{\delta\neq\alpha}\left\langle m_p^{(\delta)}\right\rangle_Q\left\{\left(d_{\delta\alpha}+d_{\alpha\delta}\right) - \frac{1}{2}\frac{1}{\sum\limits_\gamma \left\langle m_p^{(\gamma)}\right\rangle_Q}\right. \\ & \left.\left.\sum_{\varepsilon\neq\delta,\alpha}\left\langle m_p^{(\varepsilon)}\right\rangle_Q\cdot\left(d_{\delta\varepsilon}+d_{\varepsilon\delta}\right)\right\}\right] \end{aligned} \tag{4.26}$$

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:

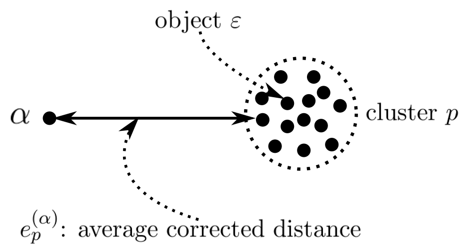
- $\rightsquigarrow$ symmetric: $d_{\alpha,\alpha'} = d_{\alpha',\alpha}$
- $\rightsquigarrow$ diagonal elements $d_{\alpha\alpha'} \overset{!}{=} 0$

Under this assumption, the mean-fields simplify as:

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

interpretation of the "mean fields":

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



$\Rightarrow$ mean-fields depend on assignment probabilities $\langle m_p^{(\alpha)} \rangle_Q$ 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 $\langle m_p^{(\alpha)} \rangle_Q$ of assignment

$\Rightarrow$ assignment probabilities $\langle m_p^{(\alpha)} \rangle_Q$ depend on the average normalized distance between the object α under consideration and the objects of cluster p (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:

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

we then obtain:

$$e_p^{(\alpha)} = \frac{1}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_p\right)^2 \tag{4.29}$$

with

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

proof: supplementary material

$\rightsquigarrow$ natural extension of K-means clustering to the case of "fuzzy" assignments

4.2.4 General Mean-Field Algorithm for Pairwise Clustering

Algorithm 10 implements the stochastic optimization procedure (mean field annealing) from chapter 3.3 to find nearly optimal cluster centers and assignment of data points to these clusters.

Algorithm 10: soft k-means clustering for general distances

```
Initialization:
- choose number $M$ of partitions
- choose initial ($\beta_0$) and final ($\beta_f$) values of the noise parameter
- choose annealing factor $\eta$ and convergence criterion $\theta$
- initialize mean-fields $e_p^{(\alpha)}$ with random numbers $\in [0,1]$
- set $\beta \leftarrow \beta_0$
while $\beta < \beta_f$ do annealing
    repeatEM fixpoint iteration
        compute assignment probabilities
            $$\big\langle m_p^{(\alpha)} \big\rangle_Q = \frac{\exp\big\{ -\beta \big(e_p^{(\alpha)}\big)_{\text{old}}\big\}}{\sum_q \exp\big\{ -\beta \big(e_q^{(\alpha)}\big)_{\text{old}}\big\}} \quad \text{for all } p, \alpha$$
        compute new mean-fields
            $$\begin{aligned} \big(e_p^{(\alpha)}\big)_{\text{new}} &= \tfrac{2}{M} \frac{1}{\sum_\gamma \big\langle m_p^{(\gamma)} \big\rangle_Q} \sum_\delta \big\langle m_p^{(\delta)} \big\rangle_Q \\ &\quad \cdot \Bigg\{ d_{\delta\alpha} - \tfrac{1}{2} \frac{1}{\sum_\gamma \big\langle m_p^{(\gamma)} \big\rangle_Q} \sum_\varepsilon \big\langle m_p^{(\varepsilon)} \big\rangle_Q d_{\varepsilon\delta} \Bigg\} \quad \text{for all } p, \alpha \end{aligned}$$
    until $\big|\big(e_p^{(\alpha)}\big)_{\text{new}} - \big(e_p^{(\alpha)}\big)_{\text{old}}\big| < \theta$ for all $p, \alpha$
    $\beta \leftarrow \eta \cdot \beta$
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)

In a similar way, spiketrains can be clustered to identify groups of cells with similar response properties ($\rightsquigarrow$ spiketrain metrics).
$\square$[16]

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

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

$$\overline{d}_{p\alpha} = \frac{\sum\limits_{\gamma} \langle m_p^{(\gamma)} \rangle_Q d_{\gamma\alpha}}{\sum\limits_{\gamma} \langle m_p^{(\gamma)} \rangle_Q} \tag{4.31}$$

and the mean fields (see eq. 4.27):

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

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.

[16]slide: Skyer, Sequence clustering data; p.15, 17

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 10 yields a particularly simple version of the soft clustering procedure that is described in algorithm 11. It is robust, fast, and allows to choose k.

Algorithm 11: soft k-means clustering for Euclidean distances

```
Initialization:
- choose no. $M$ of partitions
- choose initial ($\beta_0$) and final ($\beta_f$) 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 $\eta$
- choose convergence criterion $\theta$
- $\beta \leftarrow \beta_0$
while $\beta < \beta_f$ (annealing) do
    repeat EM
        compute assignment probabilities
            $$\big\langle m_p^{(\alpha)}\big\rangle_Q = \frac{\exp\big\{-\frac{\beta}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_p^{\text{old}}\big)^2\big\}}{\sum_q \exp\big\{-\frac{\beta}{2}\big(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}}\big)^2\big\}} \quad \text{for all } \alpha, p$$
        compute new prototypes
            $$\underbrace{\underline{\mathbf{w}}_p^{\text{new}} = \frac{\sum_\alpha \big\langle m_p^{(\alpha)}\big\rangle_Q \underline{\mathbf{x}}^{(\alpha)}}{\sum_\alpha \big\langle m_p^{(\alpha)}\big\rangle_Q}}_{\substack{\text{center of mass of the data points}\\ \text{which belong to cluster } p \text{ - weighted}\\ \text{by assignment probability}}} \quad \text{for all } p$$
    until $\big|\underline{\mathbf{w}}_q^{\text{new}} - \underline{\mathbf{w}}_q^{\text{old}}\big| < \theta$ for all $q$
    $\beta \leftarrow \eta\beta$
end
```

Comments

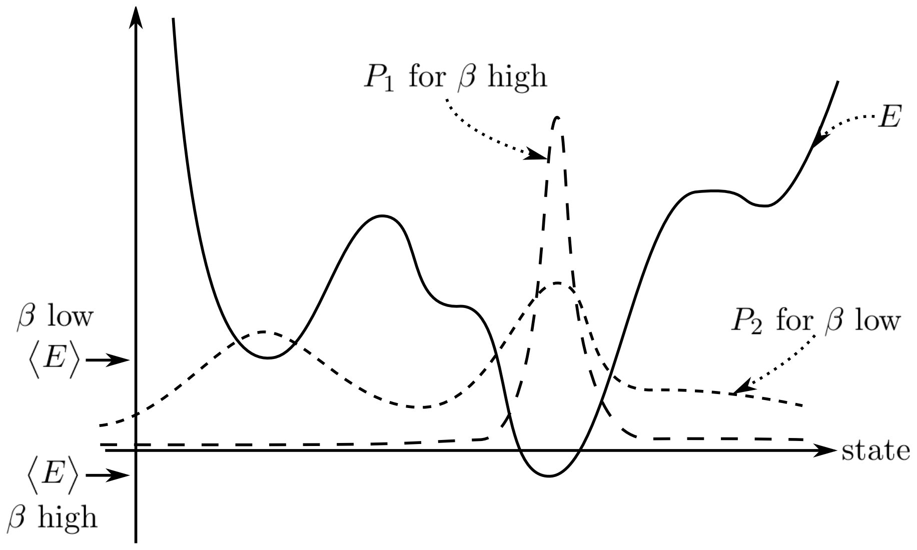
- for an "on-line" version: replace inner loop by:
  `choose observation` $\underline{\mathbf{x}}^{(\alpha)}$
  `compute assignment probabilities` $\big\langle m_p^{(\alpha)}\big\rangle_Q$ for all p
  `change all prototypes according to`

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

- 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:

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



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: quadratic Euclidean measure

neural network: Set of units with a geometrical structure (see figure 28).

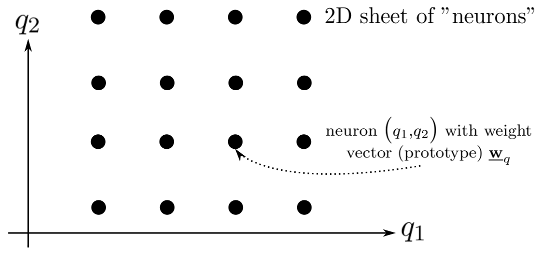

q_2
2D sheet of "neurons"
neuron (q_1,q_2) with weight vector (prototype) $\underline{\mathbf{w}}_q$
q_1

Figure 28: 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":

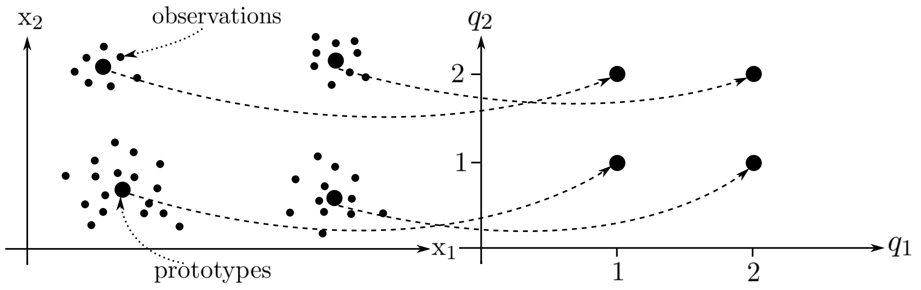
$$m_{\underline{\mathbf{p}}}^{(\alpha)} = \begin{cases} 1, & \text{if } p = \operatorname{argmin}_\gamma \left|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_\gamma\right| \\ 0, & \text{else} \end{cases} \tag{4.34}$$

$\Rightarrow$ "$\underline{\text{w}}$inner-$\underline{\text{t}}$akes-$\underline{\text{a}}$ll" network (WTA)

$\Rightarrow$ "competitive" network

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

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



$$\text{"map" of data space} \begin{cases} \text{visualization} \\ \text{dimension reduction} \\ \text{preprocessing for prediction} \end{cases}$$

Notes regarding algorithm 12

- 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:

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

$\rightsquigarrow$ using $\delta_{\underline{\mathbf{qp}}}$ as the neighborhood function $h_{\underline{\mathbf{qp}}}$ in algorithm (12) results in standard k-means

Algorithm 12: Online learning for SOMs (Kohonen map)

```
Initialization:
- choose no. $M$ of partitions
- choose annealing schedule for $\varepsilon$ and $\sigma$
- initialize prototypes, e.g.: $\underline{\mathbf{w}}_q = \underline{\mathbf{x}} + \underline{\eta}_q$
begin
    choose a data point $\underline{\mathbf{x}}^{(\alpha)}$
    determine the closest prototype:
        $\underline{\mathbf{p}} = \operatorname{argmin}_q \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}} \right|$
    change prototypes according to:
        $\Delta \underline{\mathbf{w}}_q = \varepsilon \, h_{\underline{\mathbf{qp}}} \left( \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}} \right)$
end
```

- *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 29 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 $\square$[17]

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

[17]slide: 3D surface, Leptograpsus data

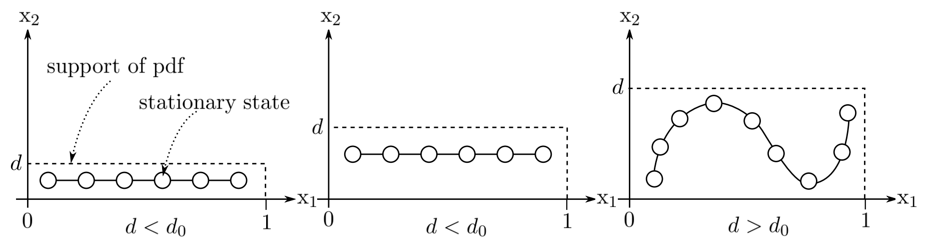


Figure 29: Dimensionality reduction with SOMs: For small d, the one-dimensional topographic map naturally covers variability of the data. For large d, the map starts to meander to capture both directions of variability in the data.

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" $\alpha, \alpha = 1, \dots, p$

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):

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

cost function:

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

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

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

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

model selection

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

Optimization: similar to pairwise mean field clustering (alg. 10), algorithm 13 implements mean-field annealing to train SOMs on pairwise data (see **GraepelObermayer1999**).

Algorithm 13: Meanfield EM-learning for SOMs with pairwise data

```
Initialization:
- choose no. of partitions
- choose initial ($\beta_0$) and final ($\beta_f$) values of the noise parameter
- choose annealing factor $\eta$
- choose convergence criterion $\gamma$
- initialization of mean-fields $e_p^{(\alpha)}$ : random numbers $\in [0,1]$
$\beta \leftarrow \beta_0$
while $\beta < \beta_e$ do annealing
    repeat EM
        compute assignment probabilities
            $$\left\langle m_{\underline{\mathbf{p}}}^{(\alpha)} \right\rangle_Q = \frac{\exp\left\{ -\beta \left( e_{\underline{\mathbf{p}}}^{(\alpha)} \right)_{\text{old}} \right\}}{\sum_q \exp\left\{ -\beta \left( e_{\underline{\mathbf{q}}}^{(\alpha)} \right)_{\text{old}} \right\}} \quad \text{for all } \underline{\mathbf{p}}, \alpha$$
        compute new mean-fields
            $$\left( e_{\underline{\mathbf{p}}}^{(\alpha)} \right)_{\text{new}} = \frac{1}{M} \sum_{\underline{\mathbf{s}}} \underbrace{h_{\underline{\mathbf{sp}}}}_{\circledast} \Bigg[ \frac{1}{\sum_\gamma \left( \sum_{\underline{\mathbf{q}}} h_{\underline{\mathbf{sq}}} \left\langle m_{\underline{\mathbf{q}}}^{(\gamma)} \right\rangle_Q \right)} \sum_\delta \Bigg( \sum_{\underline{\mathbf{q}}} h_{\underline{\mathbf{sq}}} \left\langle m_{\underline{\mathbf{q}}}^{(\delta)} \right\rangle_Q \Bigg]$$
            $$\cdot \left\{ d_{\delta\alpha} - \frac{1}{2} \frac{1}{\sum_\gamma \left( \sum_{\underline{\mathbf{q}}} h_{\underline{\mathbf{sq}}} \left\langle m_{\underline{\mathbf{q}}}^{(\gamma)} \right\rangle_Q \right)} \sum_\varepsilon \left( \sum_{\underline{\mathbf{q}}} h_{\underline{\mathbf{sq}}} \left\langle m_{\underline{\mathbf{q}}}^{(\varepsilon)} \right\rangle_Q \right) d_{\varepsilon\delta} \right\} \Bigg]$$
            for all $\underline{\mathbf{p}}, \alpha$
    until $\left| \left( e_{\underline{\mathbf{p}}}^{(\alpha)} \right)_{\text{new}} - \left( e_{\underline{\mathbf{p}}}^{(\alpha)} \right)_{\text{old}} \right| < \gamma$ for all $\underline{\mathbf{p}}, \alpha$
    $\beta \leftarrow \eta\beta$
end
```

Comments

- replacing $h_{\underline{\mathbf{sp}}}$ by $\delta_{\underline{\mathbf{sp}}}$ in algorithm 13 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 13 is called the "Kohonen-approximation" (because the original algorithm suggested by T. Kohonen is recovered for squared Euclidean distances $d_{\alpha\alpha'}$ and $\beta \to \infty$)

  $\rightarrow$ reduction of computational cost

  $\rightarrow$ visualization properties remain

$\square$[18]

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:

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

cost function:

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

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

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

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

where:

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

[18]slide: noisy spiral, brain connectivity pattern

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 14) of the learning algorithm for topographic maps.

Algorithm 14: Online learning for SOMs and Euclidean distances

```
Initialization of prototypes
Select learning step $\eta$
begin
    choose a data point $\underline{\mathbf{x}}^{(\alpha)}$
    assign data point to the 'closest' prototype
        $\underline{\mathbf{p}} = \operatorname{argmin}_{\underline{\gamma}} \sum_{\underline{\mathbf{q}}} \underbrace{h_{\gamma \underline{\mathbf{q}}}}_{\circledast} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}^{\text{old}}\right)^2$
    change all prototypes according to
        $\Delta \underline{\mathbf{w}}_{\underline{\mathbf{q}}} = \eta h_{\underline{\mathbf{pq}}} \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_{\underline{\mathbf{q}}}\right)$ for all $\underline{\mathbf{q}}$
end
```

Comments

- cost-function based approach to the Self-Organizing Map
- $h_{\underline{\mathbf{q}\gamma}} \to \delta_{\underline{\mathbf{q}\gamma}}$ 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

5 Probability Density Estimation

This chapter introduces the concept of a probability density and illustrates both *non-parametric* ($\rightarrow$ section 5.2) and *parametric* ($\rightarrow$ section 5.3) 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 5.4.

5.1 Probability Densities and Problem Statement

Probabilities

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

Probability Density

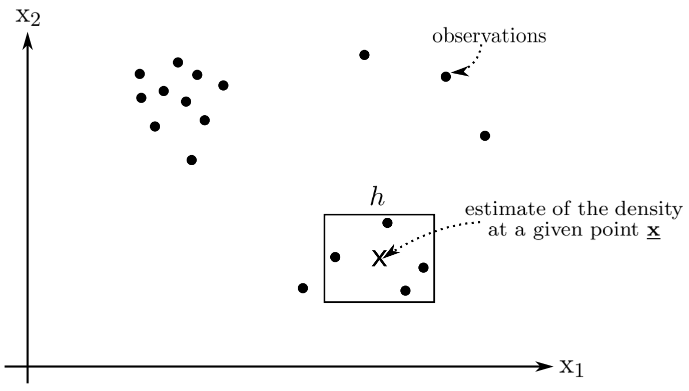
continous random variable:	X
values:	$x \in \mathbb{R}$ (can be generalised to $\mathbb{R}^n, P(\underline{\mathbf{x}})$)
probability density:	$P(x) : \mathbb{R} \rightarrow \mathbb{R}_0^+$ $\rightsquigarrow$ non-negative function $\rightsquigarrow$ normalized to $\int_{\text{support(x)}} dx P(x) = 1$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

Density Estimation

Given observations $\underline{\mathbf{x}}^{(\alpha)}, \alpha = 1, \dots, p$ 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 $P(y|x)$.

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

5.2 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")

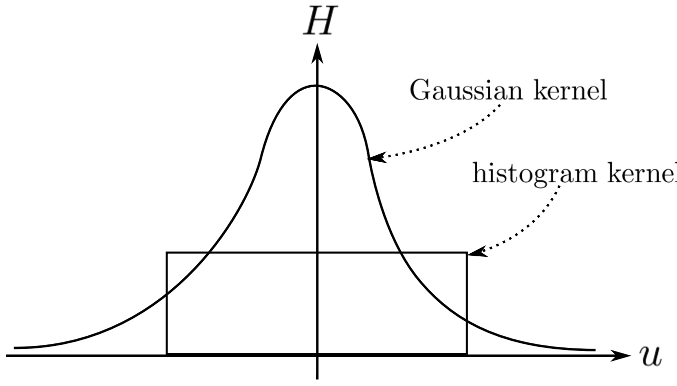
$$\widehat{P}(\underline{\mathbf{x}}) = \underbrace{\frac{1}{h^n}}_{\substack{\text{normalization} \\ \text{("density"!)}}} \cdot \underbrace{\frac{1}{p} \overbrace{\sum_{\alpha=1}^{p} H\left(\frac{\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}}{h}\right)}^{\substack{\text{number of data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

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:

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

h: hyperparameter $\rightsquigarrow$ determines smoothness

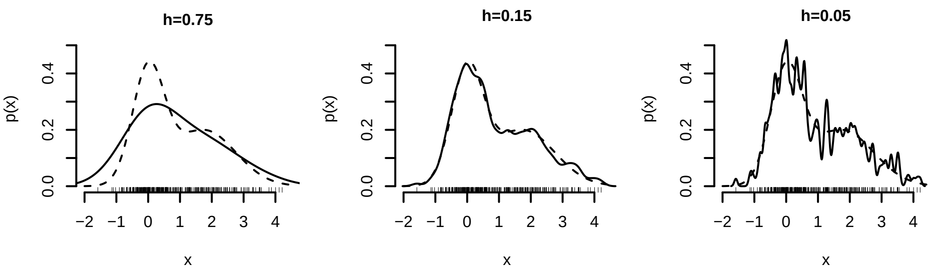


Figure 30: Impact of kernel-width on density estimate.

optimal choice of h? $\rightarrow$ see section 5.3.2

5.3 Parametric Density Estimation

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

$$\boxed{\begin{array}{c}\text{generative model}\\ \text{of a data source}\end{array}} \longrightarrow \boxed{\text{observations}}$$

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

5.3.1 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:

$$D_{KL} \overset{!}{=} \min_{(\underline{\mathbf{w}})} \tag{5.47}$$

$$\begin{aligned}\underline{\mathbf{w}}^* &= \text{argmin}_{\underline{\mathbf{w}}} \left\{ \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln P(\underline{\mathbf{x}}) - \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln \widehat{P}(\underline{\mathbf{x}};\underline{\mathbf{w}}) \right\} \\ &= \text{argmin}_{\underline{\mathbf{w}}} \left\{ \underbrace{- \int d\underline{\mathbf{x}} P(\underline{\mathbf{x}}) \ln \widehat{P}(\underline{\mathbf{x}};\underline{\mathbf{w}})}_{E^G_{[\underline{\mathbf{w}}]}} \right\} \qquad \text{"cross entropy"}\end{aligned} \tag{5.48}$$

$$E^G \overset{!}{=} \min_{(\underline{\mathbf{w}})} \tag{5.49}$$

problem: $P(\underline{\mathbf{x}})$ is unknown.

5.3.2 Principle of empirical risk minimization (ERM)

$$\boxed{\begin{matrix}\text{mathematical}\\ \text{expectation } E^G\end{matrix}} \longrightarrow \boxed{\begin{matrix}\text{empirical}\\ \text{average } E^T\end{matrix}}$$

"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:

$$\boxed{E^T = -\frac{1}{p}\sum_{\alpha=1}^{p} \ln \widehat{P}\big((\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big) \overset{!}{=} \min_{(\underline{\mathbf{w}})}} \tag{5.51}$$

Optimization

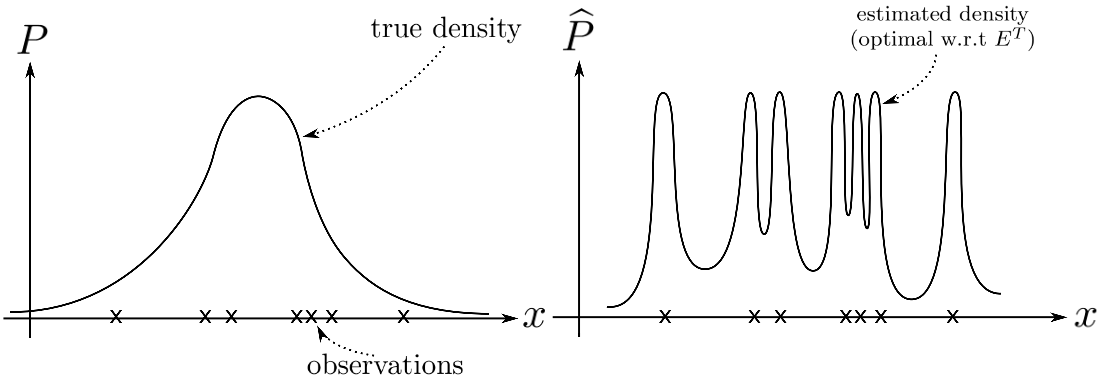
$$\underbrace{E^T_{[\underline{\mathbf{w}}]}}_{\substack{\text{total}\\ \text{cost}}} = \frac{1}{p}\sum_{\alpha=1}^{p} \underbrace{e^{(\alpha)}_{[\underline{\mathbf{w}}]}}_{\substack{\text{individual}\\ \text{cost}}} \tag{5.52}$$

standard gradient-descent procedures *(cf. MI I, sections 1.3.4 and 1.4.1-3)*

$$\left.\begin{array}{ll} \text{"batch"-learning:} & \Delta\underline{\mathbf{w}} = -\eta\frac{\partial E^T}{\partial \underline{\mathbf{w}}} \\ \text{"on-line"-learning:} & \Delta\underline{\mathbf{w}} = -\eta\frac{\partial e^{(\alpha)}}{\partial \underline{\mathbf{w}}} \end{array}\right\} \begin{array}{c}\text{examples for}\\ \text{gradient-based}\\ \text{methods}\end{array} \tag{5.53}$$

Validation

Motivation: $E^T_{[\underline{\mathbf{w}}]} \overset{!}{=} \min$ instead of $E^G_{[\underline{\mathbf{w}}]} \overset{!}{=} \min$ may lead to overfitting.

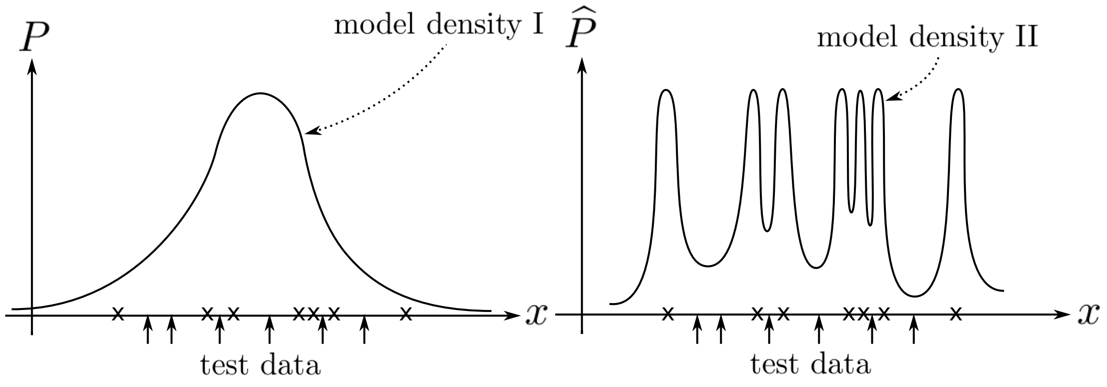


E^T small but E^G large may indicate overfitting

Testset method:

$$\text{observations}\begin{cases} \text{training data} & \{\underline{\mathbf{x}}^{(\alpha)}\}, \alpha = 1, \ldots, p \\ \text{test data} & \{\underline{\mathbf{x}}^{(\beta)}\}, \beta = 1, \ldots, q \end{cases}$$

$$\widehat{E}^G = \frac{1}{q}\sum_{\beta=1}^{q} e^{(\beta)} \leftarrow \text{ estimate of } E^G \tag{5.54}$$



$\Rightarrow \; E^T_{(I)} > E^T_{(II)}$ <u>but</u> $E^G_{(I)} << E^G_{(II)}$

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).

5.4 The Principle of Maximum Likelihood

generative model

$$\widehat{P}(\underline{\mathbf{x}}; \underline{\mathbf{w}}) \quad \substack{\text{probability density for the} \\ \text{generation of one data point}} \tag{5.55}$$

likelihood of the observations (iid assumption)

$$\widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big) = \prod_{\alpha=1}^{p} \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big) \quad \substack{\text{probability density for the} \\ \text{generation of the whole} \\ \text{observed data set}} \tag{5.56}$$

model selection via the maximum likelihood principle

$$\widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big) \overset{!}{=} \max_{(\underline{\mathbf{w}})} \tag{5.57}$$

Idea: pick the model under which the probability of observing the data is maximal

In practice: minimization of the negative log-likelihood

$$\begin{aligned} p \cdot E^T_{[\underline{\mathbf{w}}]} &= -\ln \widehat{P}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big) \\ &= -\sum_{\alpha=1}^{p} \ln \widehat{P}\big(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}\big) \qquad (5.58) \\ &\overset{!}{=} \min \end{aligned}$$

$\Rightarrow$ fully equivalent to the minimization of the KL-divergence via ERM.

5.5 Maximum Likelihood and Estimation Theory

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

true distribution (normalized):

$$P\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underbrace{\underline{\mathbf{w}}^*}_{\substack{\text{true}\\\text{parameter}\\\text{value}}}\big) \equiv P \tag{5.59}$$

$\rightsquigarrow$ 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 $\widehat{\underline{\mathbf{w}}}$:

$$\widehat{\underline{\mathbf{w}}} = \widehat{\underline{\mathbf{w}}}\big(\{\underline{\mathbf{x}}^{(\alpha)}\}\big) \tag{5.60}$$

$\rightsquigarrow$ 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 $\to$ $\widehat{\underline{\mathbf{w}}}$ is a random variable

quality criteria for estimators: (see also MI I, section 1.4.5)

$$\begin{aligned} &\text{bias:} && \underline{\mathbf{b}} = \underbrace{\big\langle \widehat{\underline{\mathbf{w}}} \big\rangle_p}_{\substack{\text{expectation}\\\text{w.r.t } \underline{\text{true}}\\\text{distribution}}} - \underline{\mathbf{w}}^* \\ &\text{variance:} && \underline{\mathbf{\Sigma}} = \big\langle (\widehat{\underline{\mathbf{w}}} - \underline{\mathbf{w}}^*)(\widehat{\underline{\mathbf{w}}} - \underline{\mathbf{w}}^*)^T \big\rangle_p \end{aligned} \tag{5.61}$$

optimal estimators:

$$\begin{aligned} &\text{no bias:} && \underline{\mathbf{b}} \overset{!}{=} 0 \quad \leftarrow \substack{\text{only possible if true model}\\\text{within model class}} \\ &\text{minimal variance:} && |\underline{\mathbf{\Sigma}}| \overset{!}{=} \min \quad \leftarrow \substack{\text{smallest average deviation}\\\text{of } \widehat{\underline{\mathbf{w}}} \text{ from } \underline{\mathbf{w}}^*} \end{aligned} \tag{5.62}$$

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:

$$\underline{\mathbf{\Sigma}} - \big(\underline{\mathbf{M}}^{-1}\big) \text{ is a positive semidefinite matrix}$$

proof: see supplementary material

$\Rightarrow$ <u>universal</u> lower bound on the variance of estimators

$\Rightarrow$ example: one scalar parameter w:

$$\sigma_{\mathrm{w}}^2 - \left\{ - \left\langle \frac{d^2 \ln P}{d\mathrm{w}^2} \right\rangle_p \bigg|_{\underline{\mathbf{w}}^*} \right\}^{-1} > 0 \qquad \binom{\text{"positive}}{\text{definite"}}$$

$$\sigma_{\mathrm{w}}^2 > - \frac{1}{\underbrace{\left\langle \frac{d^2 \ln P}{d\mathrm{w}^2} \right\rangle_p \bigg|_{\underline{\mathbf{w}}^*}}_{\substack{\text{Fisher}\\ \text{information}}}} \tag{5.63}$$

$\Rightarrow$ Fisher information is an interesting measure for evaluating data representations

good estimators:

efficient estimator: $\underline{\mathbf{b}} = \underline{\mathbf{0}}$ and $\underline{\mathbf{\Sigma}} = \underline{\mathbf{M}}^{-1}$ $\quad\leftarrow$ variance assumes lower bound

unbiased minimum variance estimator: $\underline{\mathbf{b}} = \underline{\mathbf{0}}$ and $\left|\underline{\mathbf{\Sigma}} - \underline{\mathbf{M}}^{-1}\right| \overset{!}{=} \min(\text{all estimators})$

(5.64)

$\Rightarrow$ 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)

results for the maximum likelihood estimator

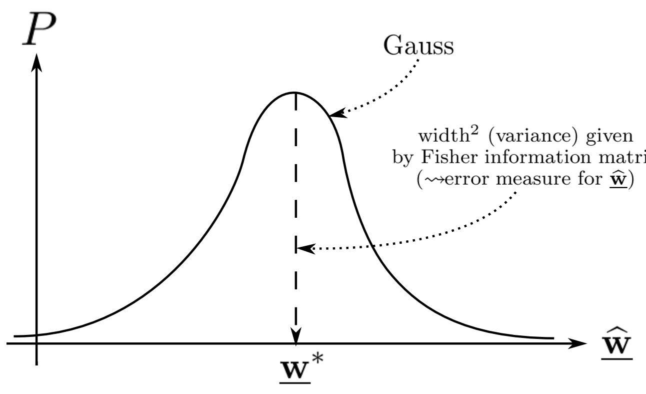
$$P\big(\{\underline{\mathbf{x}}^{(\alpha)}\}; \underline{\mathbf{w}}\big) \qquad \text{normalized and two times differentiable}$$

$$M_{ij} = - \left\langle \frac{\partial^2 \ln P}{\partial \mathrm{w}_i \partial \mathrm{w}_j} \right\rangle_p \qquad \text{Fisher information matrix} \tag{5.65}$$

then:

$$\underline{\widehat{\mathbf{w}}} \sim \mathcal{N}\big(\underline{\mathbf{w}}^*, \underline{\mathbf{M}}_{(\underline{\mathbf{w}}^*)}^{-1}\big) \quad \substack{\text{asymptotically}\\ \text{Gaussian}\\ \text{distributed}} \tag{5.66}$$

for a proof, see e.g. **Rao1973**


P
Gauss
width^2 (variance) given by Fisher information matrix ($\rightsquigarrow$error measure for $\underline{\widehat{\mathbf{w}}}$)
$\underline{\mathbf{w}}^*$
$\underline{\widehat{\mathbf{w}}}$

$\Rightarrow$ "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