Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI

Lecture Notes

PATTERN RECOGNITION AND NEURAL NETWORKS

Prof. Dr.-Ing. H. Ney

Winter Term 2003/4

Prepared: A. Eiden, SS 1995
Corrected: R. Schlüter, SS 2002
Translated: M. Popović, R. Schlüter, WS 2003/2004
Last modified: R. Schlüter, October 14, 2003 (18:23 h)

Literature:

H.A. Bourlard, N. Morgan: *Connectionist Speech Recognition – A Hybrid Approach.*
Kluwer Academic Publishers, Boston, 1994

R.O. Duda, P.E. Hart, D.G. Storck: *Pattern Classification.*
2nd ed., J. Wiley, New York, 2001.

R.O. Duda, P.E. Hart: *Pattern Classification and Scene Analysis.*
J. Wiley, New York, 1973.

P.A. Devijver, J. Kittler: *Pattern Recognition: A Statistical Approach.*
Prentice-Hall, Englewood Cliffs, NJ, 1982.

K. Fukunaga: *Introduction to Statistical Pattern Recognition.*
Academic Press, New York, 1990.

H. Niemann: *Klassifikation von Mustern.*
Springer, Berlin, 1983.

R. Schalkoff: *Pattern Recognition: Statistical, Structural and Neural Approaches.*
J. Wiley, New York, 1992.

special: Neural Networks

C.M. Bishop: *Neural Networks for Pattern Recognition.*
Oxford University Press, Oxford, UK, 1995.

S. Haykin: *Neural Networks. A Comprehensive Foundation.*
Prentice-Hall, Englewood Cliffs, NJ, 1997.

S.Y. Kung: *Digital Neural Networks.*
Prentice-Hall, Englewood Cliffs, NJ, 1993.

B.D. Ripley: *Pattern Recognition and Neural Networks.*
Cambridge University Press, Cambridge, UK, 1996.

special: HMM

L.R. Rabiner, B.H. Juang: *Fundamentals of Speech Recognition.*
Prentice-Hall, Englewood Cliffs, NJ, 1993.

X.D. Huang, Y. Ariki, M.A. Jack: *Hidden Markov Models for Speech Recognition.*
Edinburgh Univ. Press, 1990.

special: Stohastic

G. Casella, R.L. Berger: *Statistical Inference.*
Wadsworth&Brooks/Cole, Pacific Grove, CA, 1990.

U. Krengel: *Einf"uhrung in die Wahrscheinlichkeitstheorie und Statistik.*
Vieweg, 1988.

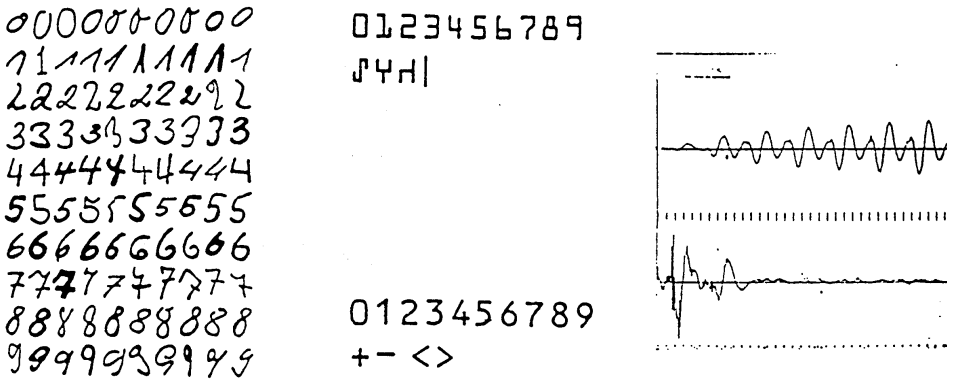
Extension/Application Lectures

- Speech Recognition and Search Procedures
- Digital Signal Processing for Speech and Images
- Language Modelling
- Stohastic Modelling in Pattern Recognition
- Grammatical Inference and Machine Learning
- Introduction to Artificial Intelligence

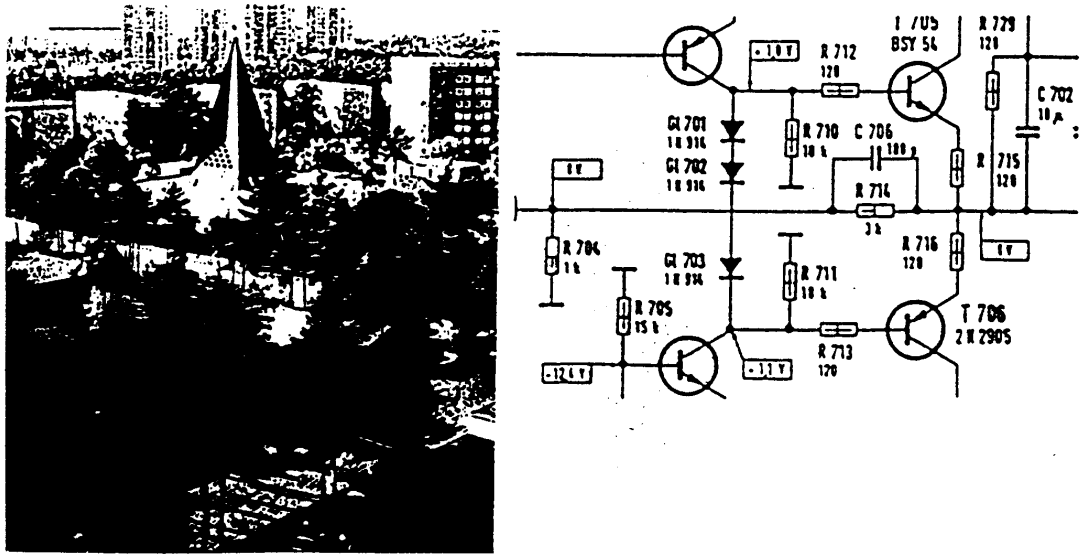
Contents

1 Introduction

Three examples of simple patterns:
left: hand-written digits; center: standardized digits; right: voltage flow on the microphone output for the German word “mit”.



Two examples of complex patterns:
left: part of a town; right: clip of an electronic circuit scheme.



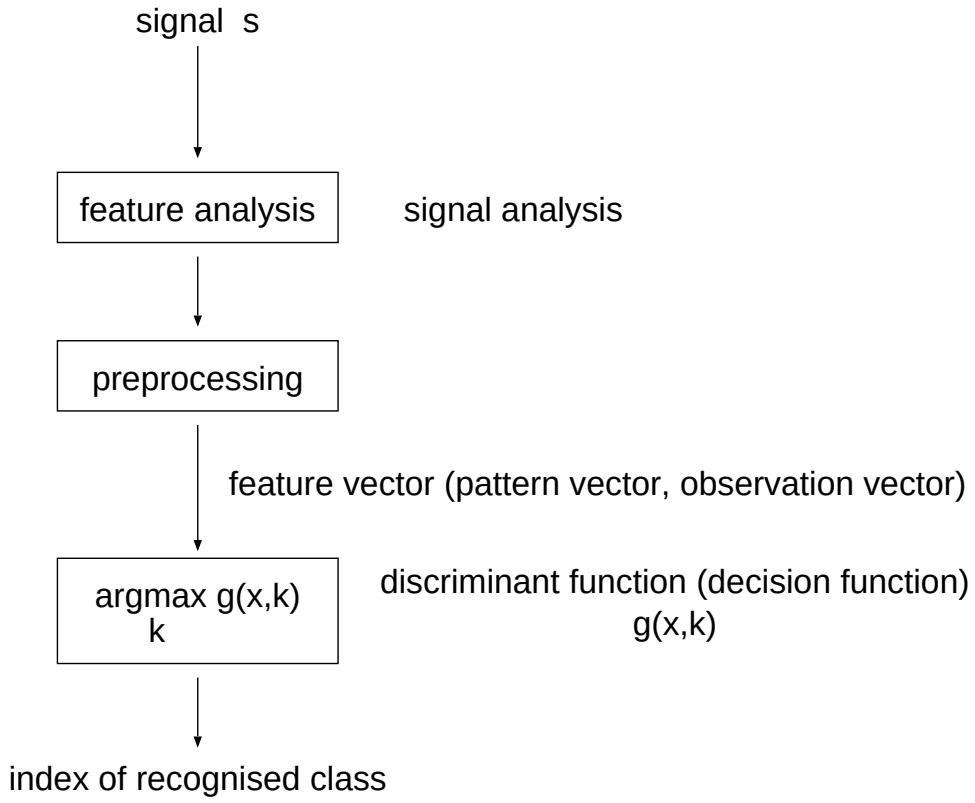
1.1 Pattern Recognition Domains

- Signals ("one-dimensional")
  - acoustic signals (noise, motors, ...)
  - speech
  - EKG, EEG, phonocardiogram
  - sonar, radar
- Images ("two-dimensional")
  - medical images (x-ray, cell, tomography, spine tomography images)
  - physics of elementary particles
  - production: electronic circuits, components, ...
  - satellite images
- written characters
  - OCR (Optical Character Recognition)
  - hand writing

1.2 Structure of a Recognition System

1.2.1 Approach with Discriminants

Classes $k = 1, \ldots, K$ (e.g. letters for character recognition)

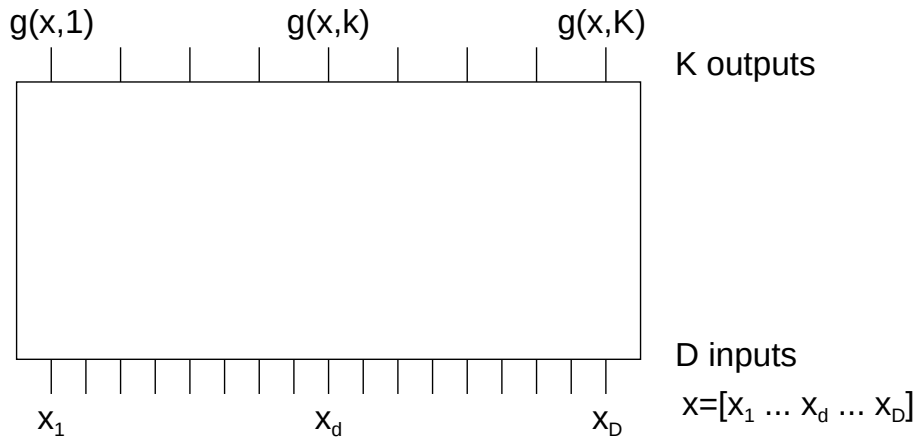


A corresponding class has to be found for a given observed signal s with the calculated feature vector $x \in \mathbb{R}^D$. In order to do this, we need a *decision rule* $x \mapsto r(x)$:

$$x \longmapsto r(x) = \underset{k}{\operatorname{argmax}} \{g(x,k)\}$$

$g(x,k)$ is called *discriminant function*.

Representation of the function $x \mapsto g(x,k), \quad k = 1, \ldots, K$:



In principle there is no constraint for the functional dependency $x \mapsto g(x,k)$. Often used in practice:

- linear functions of x : $\quad g(x,k) = \sum_{d=1}^{D} w_{kd} x_d + w_{k0}$

  quadratic functions, i.e. linear functions of components $x_d\ x_c$:

  $\downarrow$

  polynomial classifiers

- neural network (ANN = artificial neural network), especially the *Multilayer-Perceptron* (with one or two "hidden" layers)

Design criteria for discriminant function $g(x,k)$:

$g(x,k) \longmapsto 1$ if k is the "right" class
$g(x,k) \longmapsto 0$ if k is the "wrong" class

In practice, only approximations of these ideal values can be achieved, so (as we will see later) the *squared error criterion* is applied in the learning phase.

1.2.2 Statistical Approach

Classes $k = 1, \ldots, K$ (e.g. letters for character recognition)

- a-priori probabilities: $p(k), \quad \sum_{k=1}^{K} p(k) = 1.$

  A-priori probabilities are usually calculated as relative frequencies.

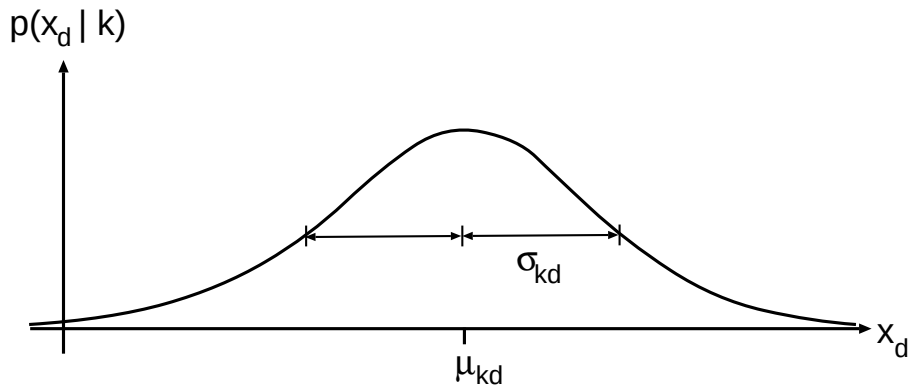
  Example - letter recognition: different letters have different frequencies.

- class-dependent probability densities: $p(x|k), \quad x \in \mathbb{R}^D.$

  Example:

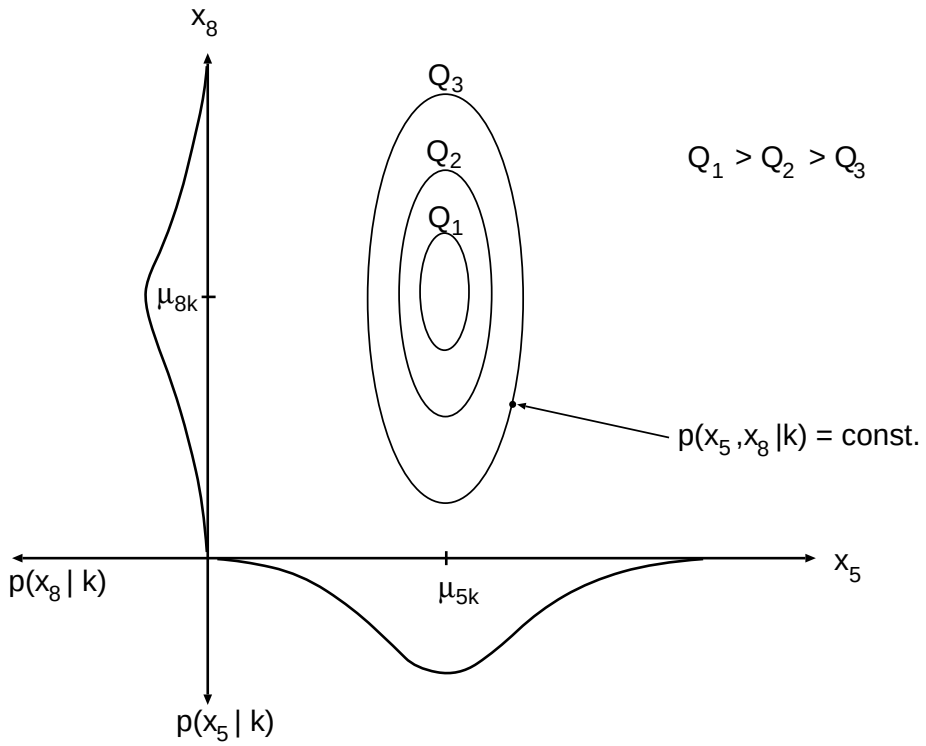
$$p(x|k) = \prod_{d=1}^{D} p(x_d|k)$$

$$p(x_d|k) : \text{univariate Gaussian distribution}$$



μ_{kd} : the average value of the component x_d for the class k
σ_{kd}^2 : the variance of the component x_d for the class k

Example (two-dimensional):



If we want to apply distributions $p(k)$ and $p(x|k)$, we have to know their parameters:

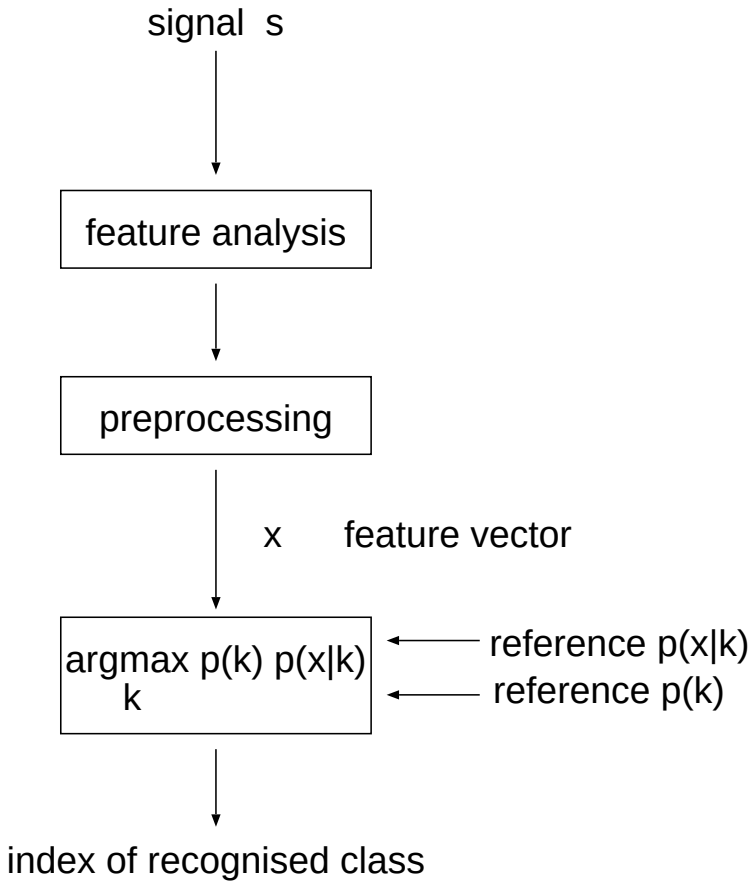
$p(k)$: relative frequencies

$p(x|k)$: Gaussian distribution $p(x|k) = \prod_{d=1}^{D} p(x_d|k)$

$$\left.\begin{array}{l}\text{average value } \mu_{kd} \\ \text{variance } \sigma_{kd}^2\end{array}\right\} \text{ for each component } x_d \text{ and class } k$$

Number of parameters:

$K-1$	values for $p(k)$	(normalization: $\sum_{k=1}^{K} p(k) = 1$)
$2 \cdot K \cdot D$	values for $p(x\|k)$	$\left[\mu_{kd}, \sigma_{kd}^2\right]$



In the statistical approach, Bayes decision rule is used for determining the corresponding class k for a given observation x

$$x \;\longmapsto\; r(x) = \operatorname*{argmax}_k \{p(k) \cdot p(x|k)\}.$$

During the course, the following connection between two approaches will be shown:

In ideal case, the discriminant function $g(x, k)$ approximates the *a-posteriori probability* $p(k|x)$.

$$g(x,k) \;\rightarrow\; p(k|x) := \frac{p(k)\; p(x|k)}{\sum_{k'=1}^{K} p(k')\; p(x|k')}$$

1.2.3 Typical Pattern Recognition Tasks

Typical tasks for both approaches are:

1. Defining and calculating suitable features; this is usually task-specific:
   - acoustic signals: spectral analysis, Fourier–transform, ...
   - writing: form-features, ...
   - images: texture, form-features, Fourier–transform, ...
2. Finding suitable models and structures for
   - $p(k)$ and $p(x|k)$ for the statistical approach
   - $g(x, k)$ for the discriminant approach
3. Defining suitable training criteria and algorithms in order to estimate free parameters from the training sample.
4. Search problem: maximization
   - not critical for 10 spoken digits or 36 written letters / digits
   - critical for 10000 words, and especially for $10000^{10} = 10^{40}$ sentences.

The tasks 1 and 4 are treated in the lecture *Algorithms for Speech Recognition* [Ney99a].

1.3 Random Variables and Distributions

The class index k is omitted in order to simplify the notation.

- Discrete random variable:

  Examples:

  - throwing coins: $X = \{0, 1\}$
  - throwing dice: $X = \{1, 2, 3, 4, 5, 6\}$

  Probability distribution:

$$p(x) \geq 0, \qquad \sum_{x \in X} p(x) = 1$$

  Expected value of function $x \longmapsto g(x)$:

$$E\{g(x)\} = \sum_{x \in X} p(x) \cdot g(x)$$

  Average value ("weighted averaging") of x:

$$E\{x\} = \sum_{x \in X} x \cdot p(x)$$

  Variance of x ("scattering around average value"):

$$\begin{aligned} Var\{x\} &= E\{(x - E\{x\})^2\} \\ &= E\{x^2 - 2x\ E\{x\} + E^2\{x\}\} \\ &= E\{x^2\} - 2\ E^2\{x\} + E^2\{x\} \\ &= E\{x^2\} - E^2\{x\} \end{aligned}$$

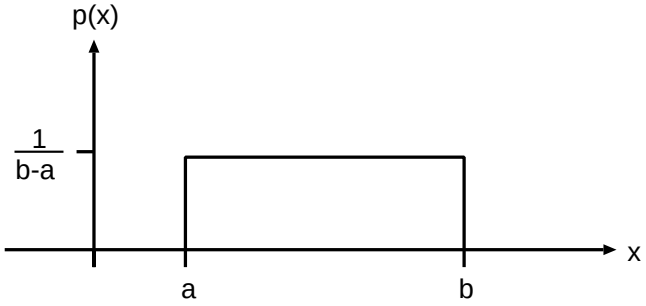
- One-dimensional, continuous random variable: $x \in \mathbb{R}$

  A one-dimensional, continuous random variable is also called a *univariate* random variable.

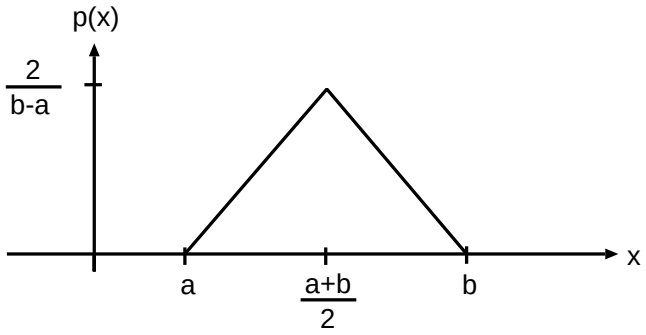
Distribution densities: $p(x) \geq 0, \qquad \int_{-\infty}^{\infty} dx\, p(x) = 1$

Examples:

– uniform distribution:



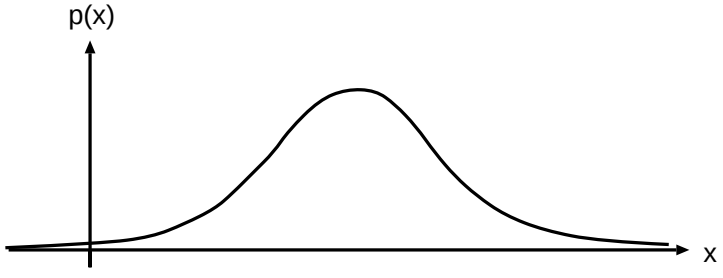
– triangular distribution:



Both examples above illustrate the probability density $p(x)$ whose values are equal to zero outside certain finite interval.

An example of a distribution whose values are always greater than zero:

– Gaussian distribution, normal distribution (bell curve):



Gaussian distribution:

$$\forall x : p(x) > 0, \qquad p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right).$$

The values $a, b, \mu, \sigma^2, \ldots$ are necessary for the full definition of densities and are called parameters of the distribution density.

Expected value, average value and variance for continuous random variables are defined by using the integral instead of the sum:

Expected value of a function $x \longmapsto g(x)$:

$$E\{g(x)\} = \int_{-\infty}^{\infty} dx\, p(x) \cdot g(x)$$

Average value:

$$E\{x\} = \int_{-\infty}^{\infty} dx\, x \cdot p(x)$$

Variance:

$$Var\{x\} = \int_{-\infty}^{\infty} dx\, (x - E\{x\})^2 \cdot p(x)$$

Exercise: Calculate the average value and the variance for the three previous examples (uniform distribution, triangular distribution, Gaussian distribution).

- multidimensional, continuous random variable: $x \in \mathbb{R}^D$

  Multidimensional, continuous random variables are called *multivariate* random variables.

  $x \in \mathbb{R}^D$: $\quad x = [x_1, \ldots, x_d, \ldots, x_D]$

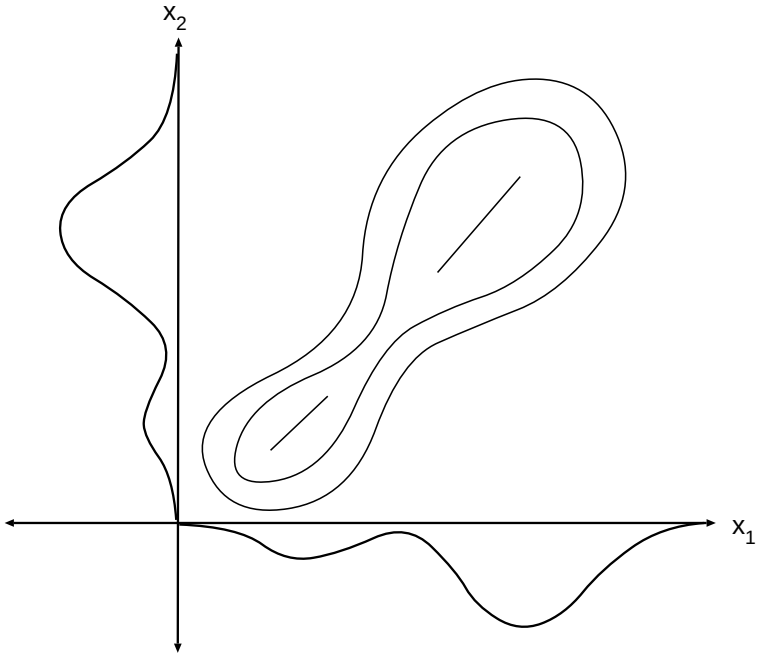
$$p(x) = p(x_1, \ldots, x_d, \ldots, x_D) \geq 0$$

$$\int_{\mathbb{R}^D} dx\, p(x) = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} dx_1 \ldots dx_d \ldots dx_D\, p(x_1, \ldots, x_d, \ldots, x_D) = 1$$

Illustration: two-dimensional case ($D = 2$) : $x = [x_1, x_2]$

$p(x_1, x_2)$ represents the joint distribution / distribution density of the joint event $[x_1, x_2]$.

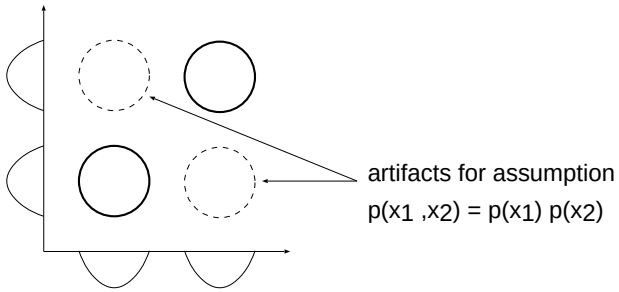
Representation of density $p(x_1, x_2)$ by iso-likelihood lines, i.e. lines with constant probability density value:



Marginal distributions $p(x_1)$ and $p(x_2)$ are obtained as *projections* on x_1 and x_2:

$$p(x_1) = \int_{-\infty}^{\infty} dx_2 \, p(x_1, x_2) \, ,$$
$$p(x_2) = \int_{-\infty}^{\infty} dx_1 \, p(x_1, x_2) \, .$$

Normally $p(x_1, x_2) \neq p(x_1) \cdot p(x_2)$.



Conditional distribution densities:

$$\begin{aligned} p(x_1|x_2) &= \frac{p(x_1, x_2)}{p(x_2)} \qquad \text{Bayes rule} \\ &= \frac{p(x_1, x_2)}{\int_{-\infty}^{\infty} dx_1' \; p(x_1', x_2)} \end{aligned}$$

Meaning: the knowledge about x_2 also includes *information* about x_1, i.e. x_1 can be better "predicted".

In words: x_2 and x_1 are correlated (or dependent).

(Stochastic) independence of x_1 and x_2:

$$p(x_1, x_2) = p(x_1) \cdot p(x_2) \qquad \forall \; x_1, x_2.$$

If x_1 and x_2 are stochastically independent, then:

$$\begin{aligned} p(x_1|x_2) &= p(x_1), \\ p(x_2|x_1) &= p(x_2). \end{aligned}$$

Stochastic Al independence of variables means that the knowledge about one of the two variables does not contain any information about the other. Similarly for $D > 2$.

Expected value of a function:

$$\begin{array}{rccc} g: & \mathbb{R}^D & \longrightarrow & \mathbb{R} \\ & x & \longmapsto & g(x) \end{array}$$

$$E\{g(x)\} = \int\limits_{\mathbb{R}^D} dx\; g(x) \cdot p(x), \qquad E\{g(x)\} \in \mathbb{R}$$

Average value of a random variable $x \in \mathbb{R}^D$:

$$\begin{aligned} E\{x_d\} &= \int\limits_{\mathbb{R}^D} dx\; x_d \cdot p(x) \\ &= \int\limits_{-\infty}^{\infty} dx_d\; x_d \cdot p(x_d) \end{aligned}$$

$E\{x_d\}$ defines the d-th component of the average value vector $E\{x\}$.

Covariance matrix Σ: the element Σ_{cd} is defined as:

$$\Sigma_{cd} = E\{(x_d - E\{x_d\}) \cdot (x_c - E\{x_c\})\}$$

The diagonal element σ_d^2 of the covariance matrix is the variance of the d-th component.

$$\begin{aligned} \sigma_d^2 &= \Sigma_{dd} = E\{(x_d - E\{x_d\})^2\} \\ &= \int\limits_{-\infty}^{\infty} dx_d\; p(x_d) \cdot (x_d - E\{x_d\})^2 \end{aligned}$$

1.4 Gaussian Distribution: Univariate and Multivariate

We consider the conditional probability (density) $p(x|k)$ and the observation vector $x \in \mathbb{R}^D$ for one fixed class k.

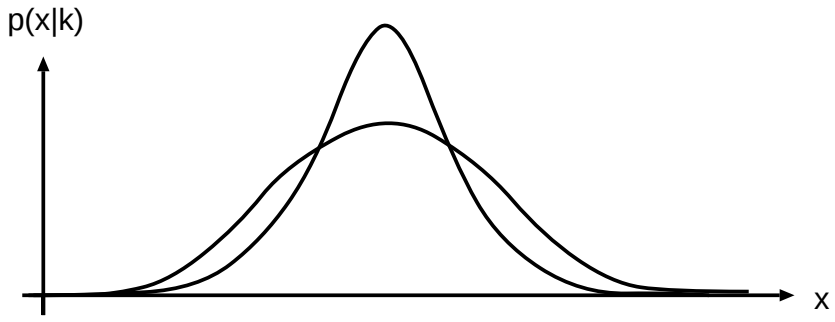
Observation vectors usually fluctuate around one "typical" prototype vector. These variations are often described by Gaussian distribution.

- Univariate Gaussian distribution:

  We fix one vector component d. The distribution of x_d is then:

$$p(x_d|k) = \frac{1}{\sqrt{2\pi\sigma_{kd}^2}} \exp\left[-\frac{1}{2}\left(\frac{x_d - \mu_{kd}}{\sigma_{kd}}\right)^2\right]$$

  with parameters μ_{kd} and σ_{kd}^2.



  For the parameters it holds:

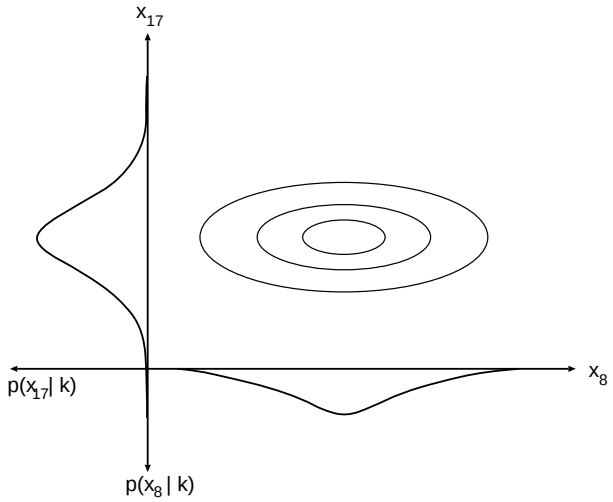
$$\int_{-\infty}^{+\infty} dx_d \; x_d \; p(x_d|k) = \mu_{kd} \qquad \text{average value}$$

$$\int_{-\infty}^{+\infty} dx_d \; (x_d - \mu_{kd})^2 \; p(x_d|k) = \sigma_{kd}^2 \qquad \text{variance}$$

- Multivariate Gaussian distribution with independent components:

$$\begin{aligned} p(x|k) &= p(x_1, \ldots, x_d, \ldots, x_D|k) = \prod_{d=1}^{D} p(x_d|k) \\ &= \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi\sigma_{kd}^2}} \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - \mu_{kd}}{\sigma_{kd}}\right)^2\right] \end{aligned}$$

Iso-likelihood lines $p(x|k) = const$ are hyper-ellipsoids whose main axes are parallel to the coordinate axes:



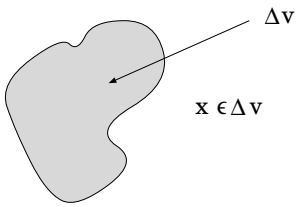
The negative logarithm of $p(x|k)$ can be interpreted as the distance between x and μ_k in $\mathbb{R}^D$:

$$-\log p(x|k) = \frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - \mu_{kd}}{\sigma_{kd}}\right)^2 + \frac{1}{2}\sum_{d=1}^{D}\log\left(2\pi\sigma_{kd}^2\right)$$

Notes about dimension:

1. Arguments in mathematical functions (e.g. exp, log, sin) have to be (actually) dimensionless values. This is achieved by predefining the dimension of the variance σ_{kd}^2.
2. Because of 1. it is possible to combine different dimensions (length, time, power, ...) in components x_d, $d = 1, \ldots, D$ without any dimension-related problems.

3. Dimension of the probability density $p(x|k)$:
   In order to obtain a true probability, integration over a volume element is necessary.



$$Pr(x \in \Delta \mathrm{v}|k) \;=\; \int\limits_{\Delta \mathrm{V}} dx\; p(x|k) \cong |\Delta \mathrm{v}| \cdot p(x|k)$$

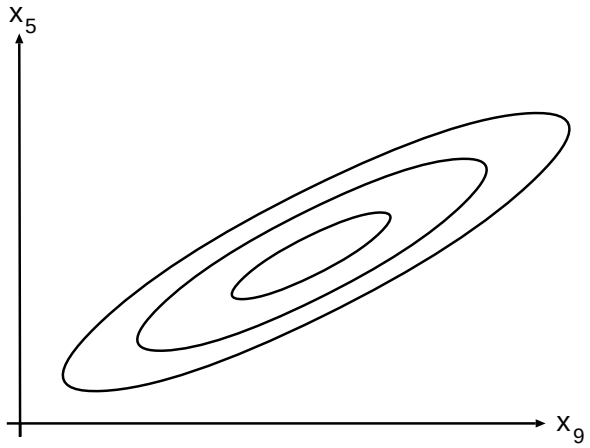
- multivariate Gaussian distribution:

  General multivariate Gaussian distribution is obtained if an arbitrary quadratic positive definite form is used in the exponent:

$$(x - \mu_k)^T\, \Sigma_k^{-1}\, (x - \mu_k)$$

  where μ_k is the average value vector
  Σ_k^{-1} is the inverted covariance matrix

  Iso-likelihood lines representation $p(x|k) = const$ results in the "rotated hyper-ellipsoid".



Taking normalization $\int\limits_{\mathbb{R}^D} dx\ p(x|k) = 1$ into account:

$$p(x|k) \;=\; \frac{1}{\sqrt{(2\pi)^D\ \det \Sigma_k}}\ \exp\left[-\frac{1}{2}(x-\mu_k)^T\ \Sigma_k^{-1}\ (x-\mu_k)\right]$$

$$=:\ \mathcal{N}(x|\mu_k, \Sigma_k).$$

For the average value of the component d it holds:

$$\int\limits_{\mathbb{R}^D} dx\ x_d\ p(x|k) \;=\; \mu_{kd}$$

and for the covariance of components c and d

$$\int\limits_{\mathbb{R}^D} dx\ (x_d - \mu_{kd})\ (x_c - \mu_{kc})\ p(x|k) \;=\; \Sigma_{k,cd}\ .$$

Exercise: Verify that the case of a diagonal covariance matrix:

$$\Sigma_k = \begin{bmatrix} \sigma_{k1}^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_{kD}^2 \end{bmatrix}$$

results in the Gaussian distribution with independent components.

Look-ahead:

A method for empirical estimation of model parameters μ_k and Σ_k:

We keep one class k fixed and collect all observation vectors $x_1, \ldots, x_N \in \mathbb{R}^D$ which belong to this class. The average value and the covariance are then estimated in the following way:

$$\begin{aligned} \hat{\mu}_d &:= \frac{1}{N} \sum_{n=1}^{N} x_{nd} \\ \hat{\Sigma}_{cd} &:= \frac{1}{N} \sum_{n=1}^{N} (x_{nd} - \hat{\mu}_d)\ (x_{nc} - \hat{\mu}_c) \end{aligned}$$

Matrix notation:

$$\hat{\Sigma} := \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{\mu})\ (x_n - \hat{\mu})^T$$

Note the difference between the scalar product and the matrix multiplication: $y \in \mathbb{R}^D$

$$\begin{aligned} y^T \cdot y = \sum_{d=1}^{D} y_d^2 \qquad & [1, D] \times [D, 1] = [1, 1] \\ \left[y \cdot y^T\right]_{cd} = y_c \cdot y_d \qquad & [D, 1] \times [1, D] = [D, D] \end{aligned}$$

1.5 Other Distributions in $\mathbb{R}^D$

We consider two methods for "construction" of a distribution in $\mathbb{R}^D$.

1. Building a multivariate distribution from the product of univariate distributions. The pairwise independence of univariate distributions is assumed.

   If $p(x_d|k)$ is the univariate distribution of the vector component x_d, then the multivariate distribution is composed as:

$$p(x_1, \ldots, x_D|k) = \prod_{d=1}^{D} p(x_d|k)$$

   Example: Laplace distribution

$$\begin{aligned} p(x_d|k) &= \frac{1}{2v_{kd}} \exp\left[-\frac{|x_d - \mu_{kd}|}{v_{kd}}\right] \\ p(x_1, \ldots, x_D|k) &= \prod_{d=1}^{D} p(x_d|k) = \left[\prod_{d=1}^{D} \frac{1}{2v_{kd}}\right] \exp\left[-\sum_{d=1}^{D} \left|\frac{x_d - \mu_{kd}}{v_{kd}}\right|\right] \end{aligned}$$

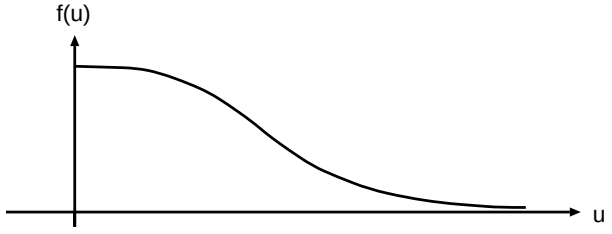
   Normalization: recalculate for yourself

2. Quadratic form (or distance):

$$(x - \mu_k)^T \, W_k \, (x - \mu_k) > 0 \qquad \forall x \in \mathbb{R}^D, x \neq \mu_k$$

   with a suitable matrix $W_k \in \mathbb{R}^D \times \mathbb{R}^D$.

   Choose a function $f : [0, \infty[\longrightarrow [0, \infty[$, typically monotone decreasing



and define the probability density

$$p(x|k) \;=\; \frac{1}{C_{NORM}} f[(x-\mu_k)^T\, W_k\, (x-\mu_k)],$$

where the normalization factor C_{NORM} results from the normalization condition

$$\int\limits_{\mathbb{R}^D} dx\; p(x|k) \;=\; 1$$

with the dependency on the matrix W_k.

Example: t-distribution

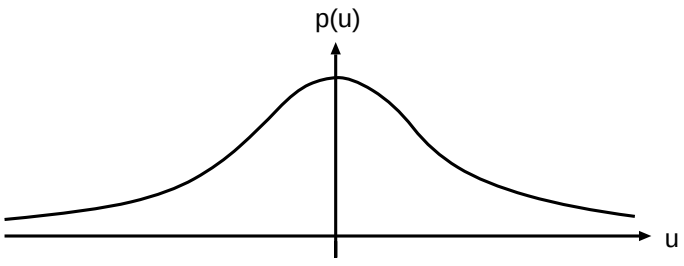
$$p(x|k) \;=\; \frac{1}{C_{NORM}} [1 + \frac{1}{M_k}(x-\mu_k)^T\, W_k\, (x-\mu_k)]^{-\left(\frac{M_k+1}{2}\right)},$$

where $\quad M_k \in \mathbb{N},\ \mu_k \in \mathbb{R}^D,\ W_k \in \mathbb{R}^{D\times D}$

A difference from the Gaussian distribution is in the polynomial decrease for $(x-\mu_k)^T\, W_k\, (x-\mu_k) \to +\infty$.

A special case of a t-distribution is Cauchy distribution, which results from setting $M_k = 1$ and $D = 1$:

$$p(u) \;=\; \frac{1}{C_{NORM}} \frac{1}{1+u^2}$$



Overview: distributions

Discrete events:

- Binomial distribution: gives the distribution for the output of a sequence of independent *Bernoulli*-experiments. Output of *Bernoulli*-experiment is binary (A: event occurs; $\overline{A}$: otherwise). Example: throwing a coin. If $p(A) = p$ is the probability that the event A occurs (and thus $p(\overline{A}) = 1 - p$) and N is the total number of observations in the considered *Bernoulli*-experiment, then the probability that event A occurs n times is given by the binomial distribution:

$$p(n|N) = \frac{N!}{n! \cdot (N-n)!} \cdot p^n \cdot (1-p)^{N-n}$$

- multinomial distribution (also polynomial distribution): generalization of binomial distribution over k different events $A_1, ..., A_K$. Probability that the event A_k with $p(A_k) = p_k$ occurs n_i times given the total number of events $N = \sum_{k=1}^{K} n_k$ is provided by the multinomial distribution:

$$p(n_1, n_2, \ldots, n_K|N) \;=\; N! \prod_{k=1}^{K} \frac{p_k^{n_k}}{n_k!}$$

  where $\sum_{k=1}^{K} p_k = 1$.

- Poisson distribution: the limit value of the binomial distribution for large values of n if $n \cdot p = \lambda$:

$$p(n) \;=\; \frac{\lambda^n}{n!} \exp(-\lambda)$$

Continuous events:

- Normal distribution / Gaussian distribution (univariate): the univariate Gaussian distribution (also "normal distribution") can be represented as a special limit value of the binomial distribution for the case $n \to \infty$ when p is constant. The univariate Gaussian distribution with the average value $\mu \in \mathbb{R}$ and the variance $\sigma^2 \in \mathbb{R}^+$ is defined as:

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Gaussian distribution (multivariate): multidimensional generalization of Gaussian distribution with the average value vector $\mu \in \mathbb{R}^D$ and the covariance matrices $\Sigma \in \mathbb{R}^{D \times D}$:

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \cdot \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

- Laplace distribution: with medians μ_d and variances v_d for $d = 1, ..., D$:

$$p(x|\mu, v) = \prod_{d=1}^{D} \frac{1}{2v_d} \cdot \exp\left[-\frac{|x_d - \mu_d|}{v_d}\right]$$

- χ^2-distribution: distribution of a sum $\chi^2 = \sum_{i=1}^{n} x_i^2$ of n squared Gaussian-distributed random variables x_i with the average values $\mu_i = 0$ and the identical unit covariance matrix $\Sigma_i = \Sigma = I$ (also called a χ^2-distribution with n degrees of freedom):

$$p(\chi^2|n) = \frac{1}{2^{\frac{n}{2}} \cdot \Gamma\left(\frac{n}{2}\right)} \cdot (\chi^2)^{\frac{n}{2}-1} \cdot \exp\left(-\frac{\chi^2}{2}\right)$$

- t-distribution or *Student* distribution: distribution of quotients $t = x/\chi^2$ of a Gaussian-distributed random variable x with the average value $\mu = 0$ and the variance $\sigma^2 = 1$ and χ^2-distributed random variable χ^2 with n degrees of freedom:

$$p(t|n) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{n} \cdot \pi \cdot \Gamma\left(\frac{n}{2}\right)} \cdot \frac{1}{\left(1 + \frac{t^2}{n}\right)^{\frac{n+1}{2}}} \qquad \text{for } n = 1, 2, ...$$

2 Bayes Decision Rule

2.1 Overview of different distributions

Observations or feature vectors: $x \in \mathbb{R}^D$
Classes: $k = 1, \ldots, K$

Assumption: following distributions are fully known, i.e. both the functional form and the parameters are known.

- a-priori distribution of classes: $p(k), \quad k = 1, \ldots, K$
- class conditioned distributions (or models) of feature vectors x: $p(x|k)$

Then it is possible to derive following distributions:

- joint distribution of pairs (x, k):

$$p(x, k) = p(k) \cdot p(x|k)$$

- distribution of x (independent of the class k):

$$p(x) = \sum_{k=1}^{K} p(x, k) = \sum_{k=1}^{K} p(k) \cdot p(x|k)$$

- a-priori distribution of k (independent of x):

$$p(k) = \int_{x \in \mathbb{R}^D} dx \; p(x, k)$$

- class conditioned distribution of x for the given class k:

$$p(x|k) = \frac{p(x, k)}{p(k)} = \frac{p(x, k)}{\int_{x' \in \mathbb{R}^D} dx' p(x', k)}$$

- a-posteriori distribution for each class k:

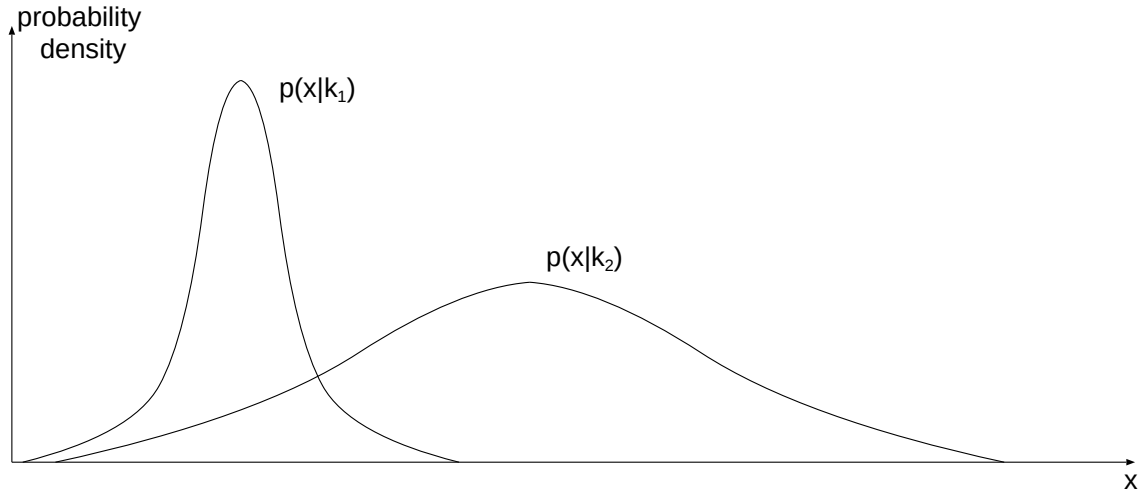
$$p(k|x) = \frac{p(x,k)}{p(x)} = \frac{p(k) \cdot p(x|k)}{\sum_{c=1}^{K} p(c) \cdot p(x|c)}$$

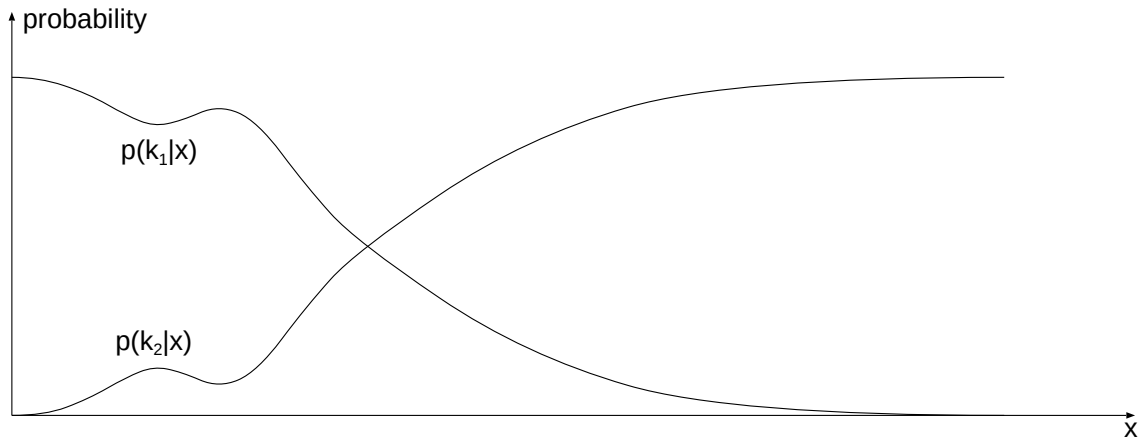
In the formula above it can be seen directly that the normalization condition

$$\sum_{k=1}^{K} p(k|x) = 1$$

is fulfilled.

Example: one-dimensional, $D = 1$
classes, $K = 2$
a-priori probabilities $p(k = 1) = \frac{2}{3}$ and $p(k = 2) = \frac{1}{3}$





2.2 Bayes Approach and Decision Rule

The goal is to find a decision rule which assigns a class k to each observation x.

$$\begin{array}{rcl} r : \mathbb{R}^D & \longrightarrow & \{1, \ldots, K\} \\ x & \mapsto & r(x) \end{array}$$

Assessment of the rule $x \mapsto r(x)$ results from a cost function (as usual for Bayes approaches):

- local costs, if one observation of the class c is assigned to one class k:

$$L[c,k] \;=\; \begin{cases} 0 & c = k \quad \text{“correct”} \\ \ldots & c \neq k \quad \text{“wrong”} \end{cases}$$

- total costs are obtained by integration of the local costs over the complete space:

$$\begin{aligned} R_{x \mapsto r(x)} &= \int_x dx \sum_{c=1}^{K} p(x,c) \cdot L[c, r(x)] \\ &= \int_x dx\; p(x) \sum_{c=1}^{K} p(c|x) \cdot L[c, r(x)] \end{aligned}$$

- minimization of total costs over the decision rule:

$$\begin{aligned} \min_{x \mapsto r(x)} R_{x \mapsto r(x)} &= \min_{x \mapsto r(x)} \int_x dx\; p(x) \sum_{c=1}^{K} p(c|x) \cdot L[c, r(x)] \\ &= \int_x dx\; p(x) \min_k \sum_{c=1}^{K} p(c|x) \cdot L[c,k] \end{aligned}$$

The costs for a given feature vector $x \in \mathbb{R}^D$ will be minimal if the following decision rule is chosen:

$$x \mapsto r(x) \;:=\; \operatorname*{argmin}_{k=1\ldots,K} \left\{ \sum_{c=1}^{K} p(c|x) \cdot L[c,k] \right\}$$

("Bayes decision rule in Pattern Recognition")

We are often interested only in recognition errors. These errors can be interpreted as costs using the following definition:

$$L[c,k] \;=\; \begin{cases} 0 & c = k \quad \text{"correct"} \\ 1 & c \neq k \quad \text{"wrong"} \end{cases}$$

This results in the following decision rule:

$$\begin{aligned} r(x) \;&:=\; \operatorname*{argmin}_{k} \left\{ \sum_{c=1}^{K} p(c|x) \cdot L[c,k] \right\} \\ &=\; \operatorname*{argmin}_{k} \left\{ \sum_{c=1}^{K} p(c|x) - p(k|x) \right\} \\ &=\; \operatorname*{argmin}_{k} \{1 - p(k|x)\} \\ &=\; \operatorname*{argmax}_{k} \{p(k|x)\} \\ &=\; \operatorname*{argmax}_{k} \{p(x,k)\} \end{aligned}$$

("Bayes decision rule for minimum error rate")

Error rate for a general decision rule:

$$p_{x \mapsto r(x)}(e) = \int_{\mathbb{R}^D} dx \; p(x) \left[1 - p(k = r(x)|x)\right].$$

Error rate for Bayes decision rule:

$$\begin{aligned} p_B(e) \;&=\; \int_{\mathbb{R}^D} dx \; p(x) \cdot \left[1 - \max_k p(k|x)\right] \\ &=\; 1 - \int_{\mathbb{R}^D} dx \; p(x) \cdot \max_k p(k|x). \end{aligned}$$

Hints:

- Note: guaranteed optimality of Bayes decision rule !
- Note the prerequisites !

Example: speech recognition

- classes: word sequences $k = w_1^N$
- features: sequences of acoustic vectors $x = y_1^T$
- cost function (for minimization of the sentence error rate):

$$C\left[w_1^N, v_1^M\right] = \begin{cases} 0, & w_1^N = v_1^M, \text{i.e. also } N = M \\ 1, & otherwise \end{cases}$$

- Bayes decision rule:

$$\underset{k}{\operatorname{argmax}} \{p(k) \cdot p(x|k)\} \longrightarrow \underset{N, w_1^N}{\operatorname{argmax}} \left\{p(w_1^N) \cdot p(y_1^T|w_1^N)\right\}$$

  acoustic model: $p(y_1^T|w_1^N)$

  language model: e.g. bigram $p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$

- A closed solution for Bayes error rate exists only rarely because the speech recognition problem comprises
  - too many classes, and
  - too many dimensions.

2.3 Discriminants and Limit Surfaces

Bayes decision rule is the starting point

$$r(x) = \underset{k}{\operatorname{argmax}} \{p(k|x)\} \quad \text{where} \quad p(k|x) = \frac{p(k) \cdot p(x|k)}{\sum_{c=1}^{K} p(c) \cdot p(x|c)}$$

A *discriminant* or *decision function* $g(x, k)$ is often used instead of $p(k|x)$:

$$r(x) \;=\; \underset{k}{\operatorname{argmax}} \{g(x, k)\},$$

where $g(x, k)$ can be derived from $p(k|x)$ using suitable transformations.

The following discriminants $g(x, k)$ do not change the decision of assigning a class k to a given feature vector x. The error rate is therefore invariant to the choice of such a discriminant.

1. $g(x, k) := p(x) \cdot p(k|x) = p(x, k) = p(k) \cdot p(x|k)$

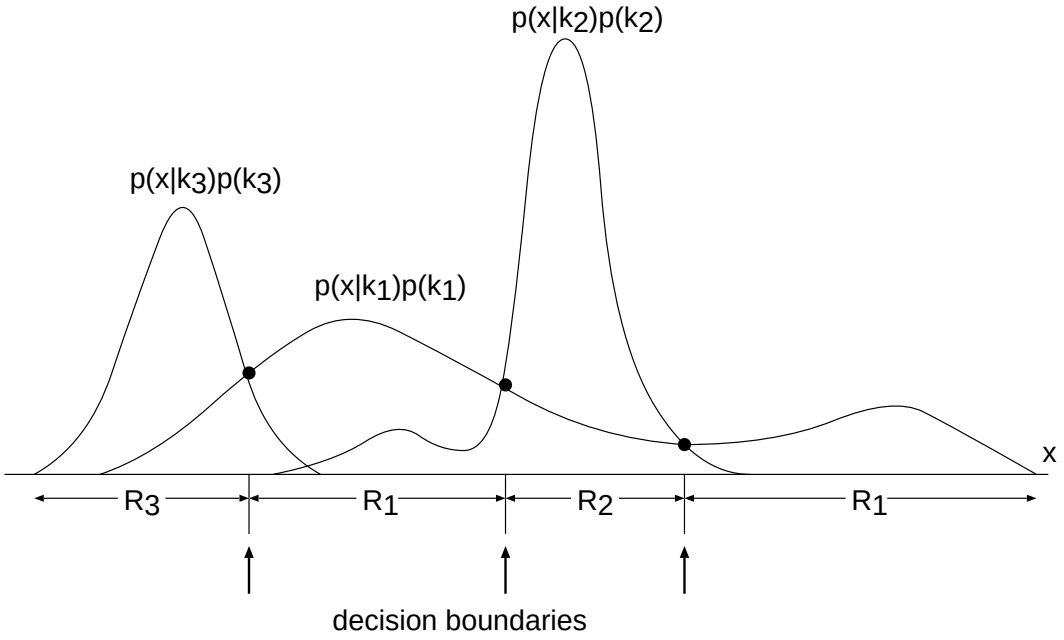
2. $g(x, k) := \log p(x, k) = \log p(k) + \log p(x|k)$

3. $$\begin{aligned} g(x, k) &:= \log p(k|x) \\ &= \log [p(k) \cdot p(x|k)] - \log \left[\sum_{c=1}^{K} p(c) \cdot p(x|c) \right] \end{aligned}$$

Limit surfaces arise from discriminants. A limit surface between the classes k and c is defined as follows:

$$\{x \in \mathbb{R}^D \ : \ g(x,k) = g(x,c)\}.$$

Note: for K classes there are $\frac{K \cdot (K-1)}{2}$ possible class pairs and the corresponding number of limit surfaces (in the maximal case).
Example: $D = 1, K = 3$
Consider the joint probability $p(x,k) = p(k) \cdot p(x|k)$:



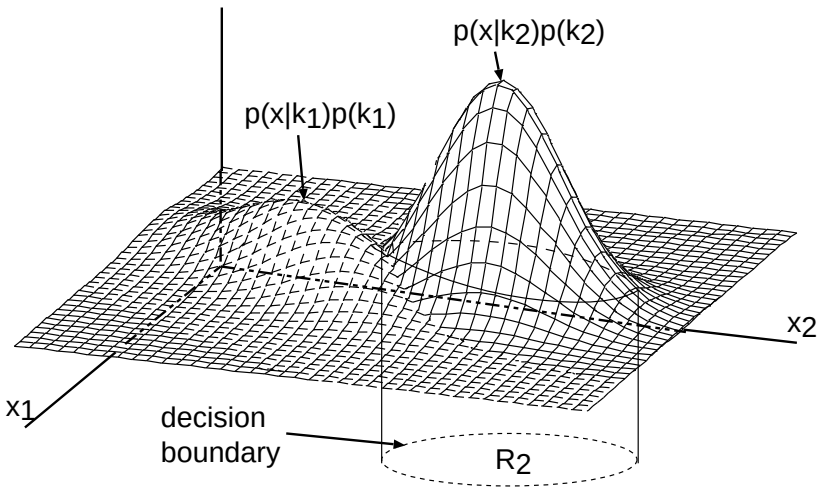
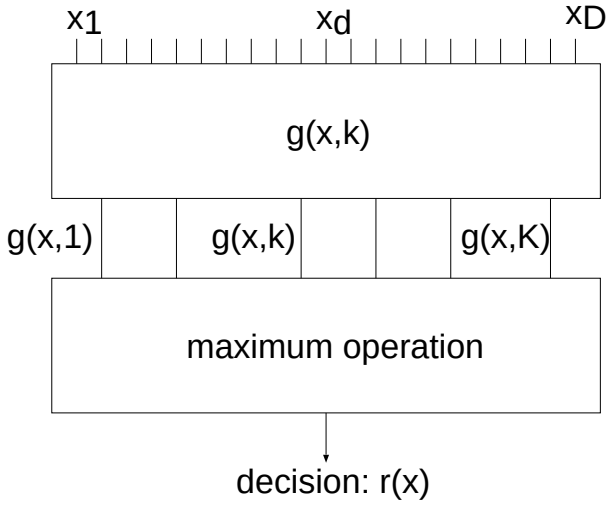


Illustration of the discriminant approach:

$$r(x) \;=\; \underset{k}{\operatorname{argmax}} \{g(x,k)\}$$



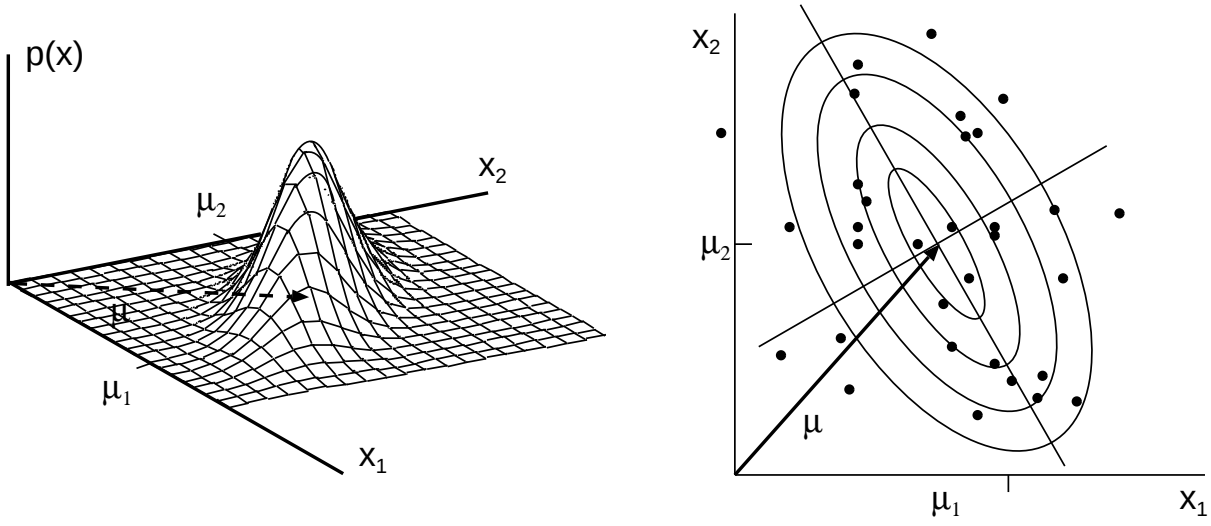
This concept consists of

- discriminant or evaluation functions $g(x,k)$,
- maximum-operation.

This concept can be also found in other ("non-statistical") approaches:

- neural networks,
- geometric classifiers.

2.4 Multivariate Gaussian Distribution



a) bivariate normal density b) scatter diagram

For each class k and each feature vector $x \in \mathbb{R}^D$ there is a class conditioned probability

$$p(x|k) = \left[(2\pi)^D \det \Sigma_k\right]^{-\frac{1}{2}} \cdot \exp\left[-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\right]$$

where μ_k is the average value vector,
Σ_k is the covariance matrix.

It holds: $(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \geq 0 \qquad \forall x \in \mathbb{R}^D$.
Thus this term can be interpreted as a distance.

We choose the following discriminant function:

$$\begin{aligned} g(x,k) &:= \log p(k) + \log p(x|k) \\ &= \log p(k) - \frac{1}{2}\log\left[(2\pi)^D \det \Sigma_k\right] - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \end{aligned}$$

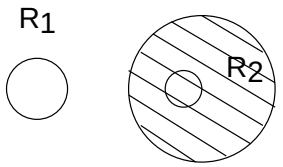
$g(x,k)$ is quadratic in x (*quadratic form*).

Different cases for determining limit surfaces:

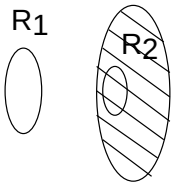
1. Σ_k arbitrary

   Most general case: the limit surface $\{x \in \mathbb{R}^D : g(x,k) = g(x,c)\}$ for the boundary between two classes k and c is also a quadratic form of x (hyper-quadratics: -planes, -circles, -ellipsoids, -hyperboloids, -paraboloids).

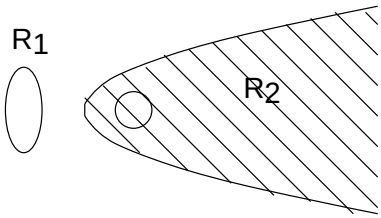
   Example: $x \in \mathbb{R}^2, K = 2$ classes. Shaded surface belongs to the class k=2.



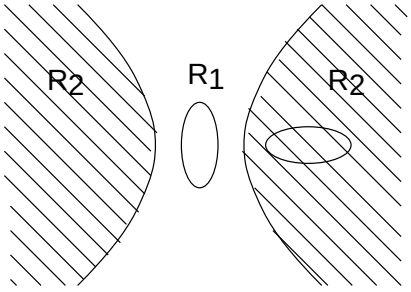
a) circle



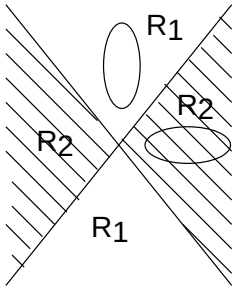
b) ellipse



c) parabola



d) hyperbola



e) straight lines

2. $\Sigma_k = \Sigma$, class independent covariance matrix (*pooled covariance matrix*)

   Limit surfaces are linear functions of x, i.e. hyper-planes, because quadratic terms of x in the equation $g(x, c) = g(x, k)$ are canceled.

   Hint: By applying the main axes transformation (diagonalization of the covariance matrix Σ) and scaling of the coordinate axes, this case can be reduced to:

$$\Sigma_k = \sigma^2 \, I$$

   (this corresponds to a *Euclidean distance*) classifier.

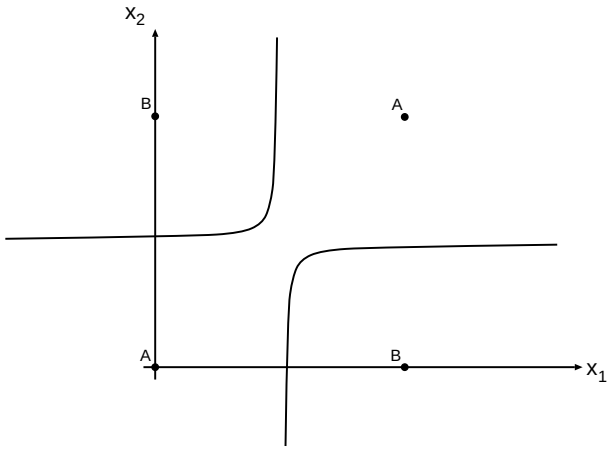
Exercise:

1. Calculate explicitly the class boundary in the case:

$$\Sigma_k \;=\; \Sigma \qquad \forall\, k$$

2. Calculate the "images" for yourself.

3. Consider the *XOR*-problem, which is often treated in context of neural networks:



   Is it possible to solve the *XOR*-problem with a Gaussian approach?

   If yes, how?

2.5 Distance or Geometric Classifiers

Starting point: Gaussian distribution with $\Sigma_k = \sigma^2 \, I$

For the discriminant function $g(x,k) = \log\left[p(k) \cdot p(x|k)\right]$ we get the following expression by omitting the terms which are constant w. r. t. k:

$$\begin{aligned} g(x,k) &= -\frac{1}{2\sigma^2}(x-\mu_k)^T(x-\mu_k) + \log p(k) \\ &= -\frac{1}{2\sigma^2}\sum_{d=1}^{D}(x_d-\mu_{kd})^2 + \log p(k) \end{aligned}$$

$\sum\limits_{d=1}^{D}(x_d-\mu_{kd})^2$ is the squared Euclidean distance between x and μ_k.

Variant: we get the *correlation classifier* by omitting the quadratic terms of x:

$$\tilde{g}(x,k) = -\frac{1}{2\sigma^2}\left[\sum_{d=1}^{D}\mu_{kd}^2 - 2\sum_{d=1}^{D}\mu_{kd}\,x_d\right] + \log p(k)$$

The only term dependent on x which remains is the *correlation* between x and μ_k:

$$\mu_k^T \, x = \sum_{d=1}^{D}\mu_{kd}\,x_d$$

(This method used to be important for "fast" hardware implementations, e.g. radar.)

Generalization: Gaussian distribution with $\Sigma_k = \Sigma$ class independent

By diagonalization of Σ and scaling of the coordinate axes, this case can be reduced to:

$$\Sigma_k = \sigma^2 \, I$$

(This transformation is also called *Whitening Transformation*)

Distance classificators:

Distance classifiers are based on a distance function of x and μ_k:

$$d(x, \mu_k) \geq 0$$

A-priori probability $p(k)$ is ignored and the following function is defined

$$g(x, k) = -d(x, \mu_k),$$

so that the decision rules becomes:

$$r(x) = \underset{k}{\operatorname{argmin}} \{d(x, \mu_k)\}$$

The terminology is not unique; the following notations are possible:

- minimum distance,
- nearest neighbor *(attention: here normally another meaning!)*,
- nearest prototype (center, mean).

l_p-*norms* (where $p \in \mathbb{N}$) are often used:

$$\begin{aligned} d_p(x, \mu_k) &:= ||x - \mu_k||_p \\ &= \left[\sum_{d=1}^{D} |x_d - \mu_{kd}|^p \right]^{1/p} \end{aligned}$$

l_p-norms fulfill the norm criteria, especially the triangle inequality, and thus define a distance measure.

Special cases

- $p = 1$: absolute value distance (city-block distance, chess board distance, comparable to Laplace distribution)
$$d_1(x, \mu_k) = \sum_{d=1}^{D} |x_d - \mu_{kd}|$$

- $p = 2$: Euclidean distance (comparable to Gaussian distribution, but notice the root)
$$d_2(x, \mu_k) = \sqrt{\sum_{d=1}^{D} |x_d - \mu_{kd}|^2}$$

- $p = \infty$: "maximum distance" (Chebyshev distance)
$$d_\infty(x, \mu_k) = \max_d |x_d - \mu_{kd}|$$

Exercise: For $x \in \mathbb{R}^2$ and $K = 3$ classes calculate the limit surfaces for the case $p = 1, p = 2$ and $p = \infty$.

Refinements: distance classifiers are sometimes refined with class- and axis-dependent scaling factors v_{kd} (*variances*). In the end, again a statistical method is obtained.

Determining the error rate for the special case:

- Gaussian distribution with $\Sigma_k = \Sigma$
- $K = 2$ classes

A closed solution exists for this special case.

Because of the assumption $\Sigma_1 = \Sigma_2$, a linear limit surface exists as it is known from chapter 2.4.

Difference of the two discriminants $g(x, 1)$ and $g(x, 2)$:

$$g(x,1) - g(x,2) = \log \frac{p(k=1)}{p(k=2)} + \underbrace{(\mu_2 - \mu_1)^T \Sigma^{-1} x + \frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2)}_{h(x)}$$

For an observation $x \in \mathbb{R}^D$ it is sufficient to calculate distance $h(x)$ between x and a linear limit surface and to integrate over this distance – instead of integrating over the total space $\mathbb{R}^D$. Since $h(x)$ is a linear function of x, it also has the Gaussian distribution (according to a general theorem for the Gaussian distribution).

We need the average value u_k and the variance v_k^2 of $h(x)$ for each class $k = 1, 2$. This results in ([Fuk90], p.85/86):

$$\begin{aligned} u_1 &= -u_2 = u \\ v_1^2 &= v_2^2 = v^2 \\ \text{where } v^2 &= 2\,u = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1) \end{aligned}$$

The value v^2 has an important role and is called *Mahalanobis distance* (between μ_1 and μ_2).

Let $P(E_{1\to 2})$ be the conditioned probability that the classifier assigns a wrong class $k = 2$ to an observation which actually belongs to the class $k = 1$. Define $\mathcal{R}_2$ as the set of all observations to which the classifier assigns the class $k = 2$:

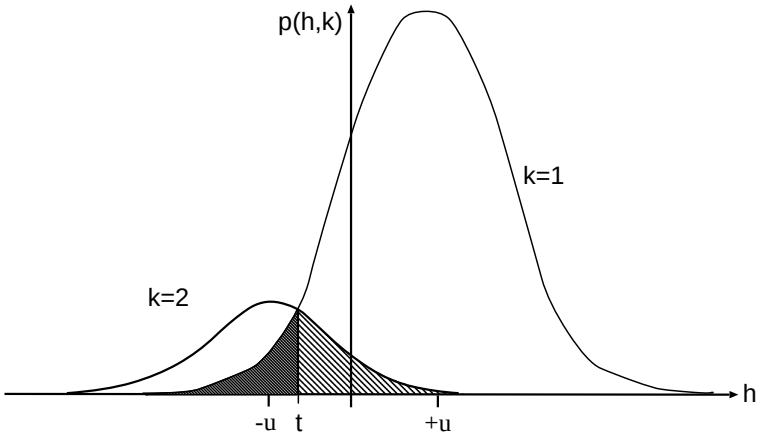
$$\mathcal{R}_2 := \{x \in \mathbb{R}^D | p(2) \cdot p(x|2) > p(1) \cdot p(x|1)\} \; .$$

Then for the class conditioned error probability of the class $k = 1$ we get:

$$P(E_{1\to 2}) = \int\limits_{\mathcal{R}_2} dx \; p(x|k = 1)$$

The class conditioned error probability for the class $k = 2$ $P(E_{2\to 1})$ can be derived analogically.

By applying the substitution with the distance $h(x)$ to the class boundary, these integrals can be reduced to one-dimensional integrals as shown in the following figure (shaded). The integration limits result from the class boundary.



The total error rate probability $P(E)$ is thus given by:

$$\begin{aligned} P(E) &= p(k=1)\ P(E_{1\to 2}) + p(k=2)\ P(E_{2\to 1}) \\ &= p(k=1)\frac{1}{\sqrt{2\pi}}\int\limits_{\frac{u+t}{v}}^{\infty} dh\ \exp\left(-\frac{h^2}{2}\right) + p(k=2)\frac{1}{\sqrt{2\pi}}\int\limits_{\frac{u-t}{v}}^{\infty} dh\ \exp\left(-\frac{h^2}{2}\right) \\ &\quad \text{where } t := \log\frac{p(k=1)}{p(k=2)} \end{aligned}$$

This equation for $P(E)$ shows that the error rate $P(E)$ depends only on

- a-priori-probabilities ratio $\dfrac{p(k=1)}{p(k=2)}$
- and Mahalanobis distance $v^2 = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1)$.

For the special case $p(k=1) = p(k=2)$, we get the following simplified representation for the total error $P(E)$:

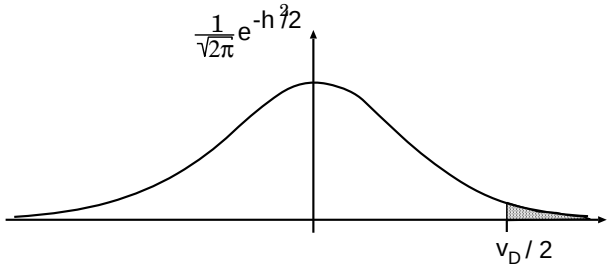
$$P(E) = \frac{1}{\sqrt{2\pi}}\int\limits_{\frac{v}{2}}^{\infty} dh\ \exp\left(-\frac{h^2}{2}\right)$$

i.e. the total error $P(E)$ depends only on the Mahalanobis distance.

What happens with the error rate $P(E)$ if the dimension D of the feature vectors increases?

In order to represent the total error rate as a function of the dimension D, we write:

$$P(E|v_D^2) \;=\; \frac{1}{\sqrt{2\pi}} \int\limits_{v_D/2}^{\infty} dh \;\exp\left(-\frac{h^2}{2}\right)$$



In the figure above it can be easily seen that:

$$\lim_{D\to\infty} P(E|v_D^2) = 0,$$

provided that for the increasing dimension D holds:

$$\lim_{D\to\infty} v_D^2 = \infty.$$

In this case the error rate $P(E|v_D^2)$ can be infinitely reduced.
To illustrate this, we assume that

$\Sigma \in \mathbb{R}^{D\times D}$: is a diagonal matrix with diagonal elements $\sigma_d^2 > 0$

Then the Mahalanobis distance v_D^2 for the dimension D is:

$$v_D^2 \;=\; \sum_{d=1}^{D} \left(\frac{\mu_{1d} - \mu_{2d}}{\sigma_d}\right)^2$$

i.e. if each new dimension contributes sufficiently to this distance, the error rate can be infinitely reduced.

2.6 Binary Features

So far we have considered random vectors $x \in \mathbb{R}^D$, but the Bayes rule also holds for discrete observations which can be obtained by discretization of measured values.
Strictly speaking it is better to call such values feature sets instead of feature vectors:

$$x = [x_1, \ldots, x_d, \ldots, x_D]$$

The case of binary features is especially simple:

$$x_d \in \{0, 1\}$$

In this case there are 2^D possible configurations of feature sets.

2.6.1 Independent Binary Features

If $x = [x_1, \ldots, x_d, \ldots, x_D]$ where $x_d \in \{0, 1\}$ is a binary feature set, then the (class conditioned) independence of features means:

$$p(x|k) = \prod_{d=1}^{D} p(x_d|k) \qquad \text{for each class } k = 1, \ldots, K$$

For a single binary feature x_d it holds:

$$\begin{aligned} p(x_d|k) &= \begin{cases} \vartheta_{kd} & x_d = 1 \\ 1 - \vartheta_{kd} & x_d = 0 \end{cases} \\ &= (\vartheta_{kd})^{x_d} \cdot (1 - \vartheta_{kd})^{1-x_d} \end{aligned}$$

where ϑ_{kd} is the class conditioned *hitting quote* .
For the whole feature set x it follows:

$$\begin{aligned} p(x|k) &= \prod_{d=1}^{D} \left[(\vartheta_{kd})^{x_d} (1 - \vartheta_{kd})^{1-x_d} \right] \\ &= \prod_{d=1}^{D} \left[\left(\frac{\vartheta_{kd}}{1 - \vartheta_{kd}} \right)^{x_d} (1 - \vartheta_{kd}) \right] \end{aligned}$$

We use the following discriminant function

$$g(x,k) \;:=\; \log\left[p(k)\cdot p(x|k)\right]$$

and by inserting the class conditioned distribution $p(x|k)$ we obtain:

$$g(x,k) \;=\; \sum_{d=1}^{D} x_d \cdot \log\frac{\vartheta_{kd}}{1-\vartheta_{kd}} + \sum_{d=1}^{D}\log\left(1-\vartheta_{kd}\right) + \log p(k)$$

This discriminant is a linear function of $[x_1,\ldots,x_D]$.

Exercise: For ternary features $x_d \in \{-1,0,1\}$ we get a quadratic discriminant function.

2.6.2 Dependent Binary Features

Rademacher-Walsh expansion

We consider again the binary feature set $x = [x_1,\ldots,x_d,\ldots,x_D]$ where $x_d \in \{0,1\}$.

For representation of the model $p(x|k)$, 2^D possible probabilities (more correct: 2^D-1 due to normalization) are needed.

Rademacher-Walsh approach:

$$p(x|k) \;=\; \sum_{i=0}^{2^D-1} a_i(k)\cdot\varphi_i(x) \qquad \text{where } \varphi_i(x)\in\{-1,1\}.$$

The base functions $\varphi_i(x)$ are capturing more and more complex dependencies between features x_d when i is increasing, and they are defined as follows:

$$\varphi_i(x) \;=\; \begin{cases} 1 & i = 0 \\ 2x_1 - 1 & i = 1 \\ \vdots & \vdots \\ 2x_D - 1 & i = D \\ (2x_1 - 1)(2x_2 - 1) & i = D + 1 \\ \vdots & \vdots \\ (2x_{D-1} - 1)(2x_D - 1) & i = D + \dfrac{D(D-1)}{2} \\ (2x_1 - 1)(2x_2 - 1)(2x_3 - 1) & \\ \vdots & \vdots \\ (2x_1 - 1)(2x_2 - 1)\ldots(2x_D - 1) & i = 2^D - 1 \end{cases}$$

The base functions are orthogonal, i.e.

$$\sum_x \varphi_i(x) \cdot \varphi_j(x) \;=\; \begin{cases} 2^D & , i = j \\ 0 & , i \neq j \end{cases} ,$$

whereby the summation is carried out over all 2^D configurations. Due to the orthogonality, for the coefficients $a_i(k)$ it follows:

$$\begin{aligned} a_i(k) \;&=\; \frac{1}{2^D} \sum_x \varphi_i(x) \cdot p(x|k) \\ &=\; \frac{1}{2^D}\; E_k\left[\varphi_i(x)\right] \end{aligned}$$

Look-ahead: The training of $a_i(k)$ is performed by *empirical averaging* of the training data $x_1, \ldots, x_N$ belonging to the class k:

$$\hat{a}_i(k) \;=\; \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2^D} \varphi_i(x_n)$$

Notes to the orthogonal series expansion:

- the series can be truncated so that only the most important dependencies between x_{kd} are captured.
- but: the resulting function $\tilde{p}(x|k)$ can become negative!
- alternative approach which enforces that $\tilde{p}(x|k)$ is positive:

$$\log p(x|k) \;=\; \sum_{i=0}^{2^D-1} b_i(k) \varphi_i(x)$$

similar methods ([DH73], p.111-114):

- expansion by Bahadur - Lazarsfeld
- *Dependence Tree* by Chow

2.6.3 Decision Rule and Error Rate for a special case

Assumptions:

- 2 classes: $k = 1, 2$ where $p(1) = p(2) = \frac{1}{2}$
- independent binary features with $\lambda > \frac{1}{2}$ and

$$p(x_d|k=1) = \begin{cases} \lambda & x_d = 1 \\ (1-\lambda) & x_d = 0 \end{cases}$$
$$p(x_d|k=2) = \begin{cases} (1-\lambda) & x_d = 1 \\ \lambda & x_d = 0 \end{cases}$$

Then for both discriminants $g(x,1)$ and $g(x,2)$ it holds:

$$\begin{aligned} g(x,1) &= \sum_{d=1}^{D} x_d \cdot \log \frac{\lambda}{1-\lambda} + \sum_{d=1}^{D} \log(1-\lambda) + \log(\frac{1}{2}) \\ &= \sum_{d=1}^{D} x_d \cdot \log \frac{\lambda}{1-\lambda} + D \cdot \log(1-\lambda) + \log(\frac{1}{2}) \\ g(x,2) &= \sum_{d=1}^{D} x_d \cdot \log \frac{1-\lambda}{\lambda} + D \cdot \log \lambda + \log(\frac{1}{2}) \end{aligned}$$

The difference of the discriminants is:

$$g(x,1) - g(x,2) = \sum_{d=1}^{D} (2x_d - 1) \cdot \log \frac{\lambda}{1-\lambda}$$

Because of $\lambda > \frac{1}{2}$ it holds $\log \frac{\lambda}{1-\lambda} > 0$, and we obtain the decision rule $r(x)$:

$$r(x) = \begin{cases} k = 1 & \text{if } \sum_{d=1}^{D} x_d > \frac{D}{2} \\ k = 2 & \text{if } \sum_{d=1}^{D} x_d < \frac{D}{2} \\ \text{arbitrary} & \text{if } \sum_{d=1}^{D} x_d = \frac{D}{2} \quad \text{(occurs only for even } D\text{)} \end{cases}$$

Calculating the error rate for odd feature vector dimensions D:

Different cases:

1. An observation x which belongs to the class 1 is by mistake assigned to the class 2.

   d (where $0 \leq d \leq \frac{D-1}{2}$) of the D components are equal to 1:

   Probability for that

   $$\lambda^d \cdot (1-\lambda)^{D-d}$$

   Number of different configurations (see binomial distribution):

   $$\binom{D}{d} = \frac{D!}{d! \cdot (D-d)!}$$

   In this way we obtain:

   $$P(E_{1\to 2}|D,\lambda) = \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \cdot \lambda^d \cdot (1-\lambda)^{D-d}$$

2. An observation x which belongs to the class 2 is by mistake assigned to the class 1.

   d (where $0 \leq d \leq \frac{D-1}{2}$) of the D components are equal to 0:

   $$P(E_{2\to 1}|D,\lambda) = \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \cdot \lambda^d \cdot (1-\lambda)^{D-d}$$

For the total error rate $P(E|D,\lambda)$ it follows:

$$\begin{aligned} P(E|D,\lambda) &= p(k=1) \cdot P(E_{1\to 2}|D,\lambda) + p(k=2) \cdot P(E_{2\to 1}|D,\lambda) \\ &= \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \cdot \lambda^d \cdot (1-\lambda)^{D-d} \end{aligned}$$

We obtain the following limit cases:

1. $\lim_{\lambda \to \frac{1}{2}} P(E|D,\lambda) = \frac{1}{2}$ (proof: exercise)
2. $\lim_{D \to \infty} P(E|D,\lambda) = 0$

That means, that additional features with $p(x_d|k) = \lambda^{x_d} \cdot (1-\lambda)^{1-x_d}$ are guaranteed to reduce the error rate.

Proof for the limit case $\lim_{D \to \infty} P(E|D,\lambda) = 0$:

$$\begin{aligned}
P(E|D,\lambda) &= \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \cdot \lambda^d \cdot (1-\lambda)^{D-d} \qquad \text{where } \lambda > \frac{1}{2} \\
&= \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \cdot \lambda^{\frac{D}{2}} \cdot (1-\lambda)^{\frac{D}{2}} \cdot \underbrace{\left(\frac{1-\lambda}{\lambda}\right)^{\frac{D}{2}-d}}_{\leq 1 \quad \forall\, 0 \leq d \leq \frac{D-1}{2}} \\
&\leq \lambda^{\frac{D}{2}} \cdot (1-\lambda)^{\frac{D}{2}} \cdot \sum_{d=0}^{\frac{D-1}{2}} \binom{D}{d} \\
&< \lambda^{\frac{D}{2}} \cdot (1-\lambda)^{\frac{D}{2}} \cdot \underbrace{\sum_{d=0}^{D} \binom{D}{d}}_{=2^D = 4^{D/2}} \\
&= [4\,\lambda(1-\lambda)]^{\frac{D}{2}}
\end{aligned}$$

Because of

$$4\,\lambda(1-\lambda) < 1$$

it follows that

$$\lim_{D \to \infty} [4\,\lambda(1-\lambda)]^{\frac{D}{2}} = 0$$

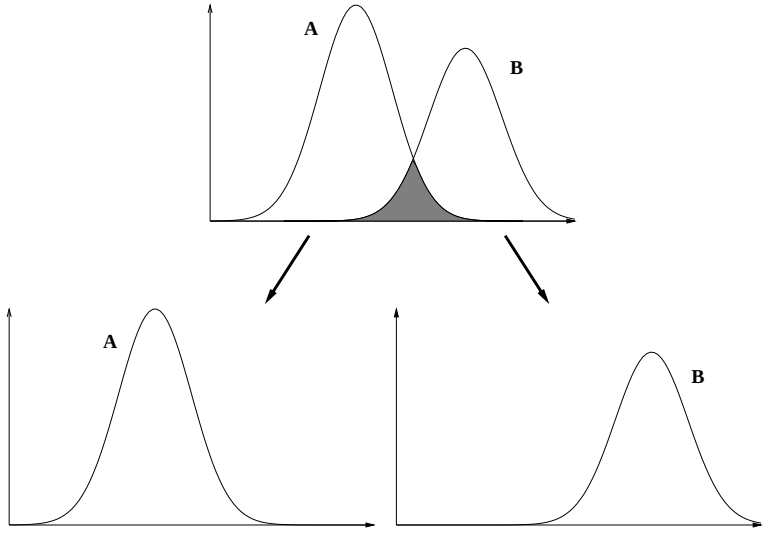
and consequently the statement

$$\lim_{D \to \infty} P(E|D,\lambda) \;=\; 0.$$

3 Training and Learning

Generally it holds: "the one" learning/estimation method does not exist; the choice of a method depends on the existing task conditions.

In this chapter: class individual and non-discriminative methods, i.e. the learning for class A and the learning for class B are fully independent of each other:



3.1 Task Formulation

Notation:	$k = 1, \ldots, K$	classes
	$x \in \mathbb{R}^D$	feature vector

In the statistical approach we started from:

$p(k)$	a-priori probability
$p(x\|k)$	class conditioned distributions or models

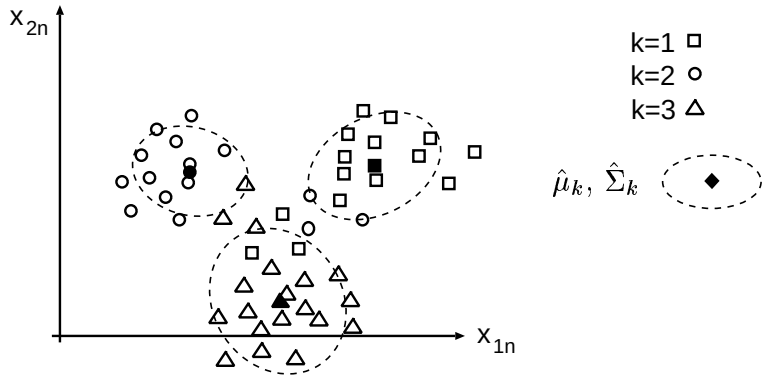
In the first two chapters we presumed that

- the functional form, e.g. Gaussian $x \mapsto p(x|k)$,
- as well as the parameters of the models, e.g. for Gaussian distribution
  - average value vector μ_k – estimated $\hat{\mu}_k$
  - covariance matrix Σ_k – estimated $\hat{\Sigma}_k$

are known.

In practice, the parameters of models $p(x|k)$ are not known, they have to be learned or estimated from training data.
For doing that, the training pairs (x_n, k_n) where $n = 1, \ldots, N$ are used, whereby x_n is the n-th feature vector and $k_n \in \{1, \ldots, K\}$ is the corresponding class.

Example: $x_n \in \mathbb{R}^2$



Example: $x_n \in \mathbb{R}^{16}$, dimension $D = 16$

Assume $N = 10^5$ observations quantified with 8 bits.
Number of possible values: $256^{16} = 2^{128} \approx 10^{38}$
$\Rightarrow$ there are much fewer observations than possible values.

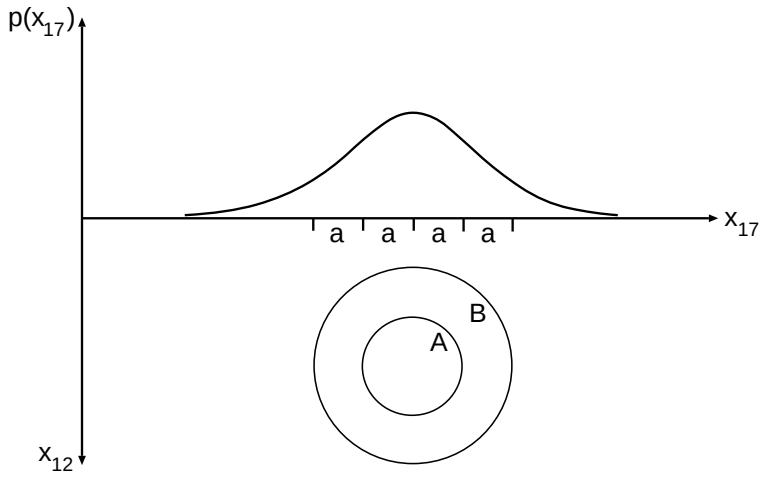
Example: Gaussian distribution

If it is possible to estimate average value vector μ_k and covariance matrix Σ_k from training data for each class, then the (estimated) class limits are also known.

Overlap between classes (or the corresponding training data) is the cause of recognition errors.

3.2 Distribution of Distances in $\mathbb{R}^D$ when $D \gg 1$

In contrast to the two- or three-dimensional case, the space is very sparsely filled with training data if $D \gg 1$ (starting from about $D = 10$).
Let us assume that the class k is fixed and the distribution $p(x|k)$ (e.g. Gaussian) for $x \in \mathbb{R}^D$ is given.



The following probabilities are to be compared:

$$Pr(x \in A) \text{ and } Pr(x \in B).$$

For that purpose we calculate the probability approximatively as the product of the volume content and the average distribution density:

$$Pr(x \in A) \;=\; c_D \cdot a^D \cdot \langle p(x|k) \rangle_A$$

where	c_D	is a constant depending on the dimension D
	$\langle p(x\|k) \rangle_A$	is the average distribution density in A

For B we obtain analogically:

$$\begin{aligned} Pr(x \in B) &= c_D \cdot \left[(2a)^D - a^D\right] \cdot \langle p(x|k)\rangle_B \\ &= c_D \cdot \left[2^D - 1\right] \cdot a^D \cdot \langle p(x|k)\rangle_B \end{aligned}$$

Hence we get the following probability ratio:

$$\frac{Pr(x \in B)}{Pr(x \in A)} = \left[2^D - 1\right] \cdot \frac{\langle p(x|k)\rangle_B}{\langle p(x|k)\rangle_A}$$

Example: $D = 16$ and $\dfrac{\langle p(x|k)\rangle_B}{\langle p(x|k)\rangle_A} = \dfrac{1}{10}$

$$\frac{Pr(x \in B)}{Pr(x \in A)} = \left[2^{16} - 1\right] \cdot \frac{1}{10} \cong 6400$$

This example shows that the number of observations in the domain A, i.e. in *the center of the distribution*, is negligible in comparison to the domain B, i.e. *far away from the center of the distribution.*

Key word: *sparseness of high-dimensional space*

The exact distribution of the distances between a vector $x \in \mathbb{R}^D$ and a class center μ_k of the class k

$$\|x - \mu_k\|^2 := \sum_{d=1}^{D} (x_d - \mu_{kd})^2$$

can be calculated if vectors x have Gaussian distribution. The derivation of this result can be found in statistics books (e.g. [Kre91], p.170-173).

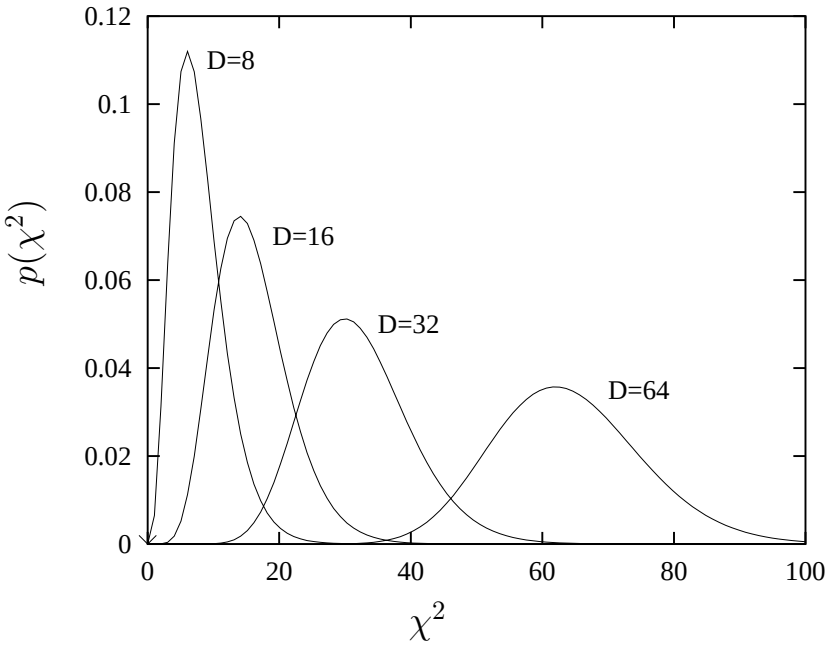
We consider one class k with Gaussian distribution $p(x|k)$ and covariance matrix $\Sigma_k = I$. The general case with an arbitrary covariance matrix can be reduced to this special case by diagonalization of the matrix and scaling of the coordinate axes.

The result is the χ^2-*distribution* ("Chi-Square") with D degrees of freedom:

$$\chi^2 = \sum_{d=1}^{D} (x_d - \mu_{kd})^2$$

$$p(\chi^2) = \frac{1}{\Gamma(\frac{D}{2})\, 2^{\frac{D}{2}}} \cdot \left(\chi^2\right)^{\frac{D}{2}-1} \cdot \exp\left(-\frac{1}{2}\chi^2\right)$$

Graphical representation of the χ^2-distribution:



Properties (without proofs):

- Average value: $E\{\chi^2\} = D$
- Variance: $Var\{\chi^2\} = 2D$
- Limit case $D \to \infty$:

$$p\left(\frac{\chi^2 - D}{\sqrt{2D}}\right) \longrightarrow \text{Gaussian distribution with } (\mu_{\chi^2} = 0, \sigma^2_{\chi^2} = 1)$$

From the graphical representation of the χ^2-distribution for $D = 8, 16, 32, 64$ it can be perceived that the concentration around the average value decreases as the dimension D increases.

3.3 Moment Method

Assume that N_k training vectors for each class k are given:

$$x_{1k}, \ldots, x_{nk}, \ldots, x_{N_k k} \qquad \in \mathbb{R}^D$$

Approach: the expected value of a function $f : \mathbb{R}^D \to \mathbb{R}$

$$\vartheta_k \;=\; \int\limits_{\mathbb{R}^D} dx\; f(x) \cdot p(x|k)$$

is replaced by *empirical averaging* of the training data:

$$\hat{\vartheta}_k \;=\; \frac{1}{N_k} \sum_{n=1}^{N_k} f(x_{nk})$$

In the moment method, the moments or the centered moments are chosen for $f(x)$:

- moments: $f(x) = x^\alpha$ for $x \in \mathbb{R}, \alpha \in \mathbb{N} > 0$
- centered moments: $f(x) = (x - \mu)^\alpha$ where $\mu = E\{x\}$

Then for the average value μ_k

$$\mu_k \;=\; \int\limits_{\mathbb{R}^D} dx\; x \cdot p(x|k)$$

we obtain the estimated value $\hat{\mu}_k$

$$\hat{\mu}_k \;=\; \frac{1}{N_k} \sum_{n=1}^{N_k} x_{nk}$$

and the estimated value for the elements $\hat{\Sigma}_{cd}^{(k)} \quad (c, d = 1, \ldots, D)$ of the covariance matrix $\hat{\Sigma}_k$:

$$\hat{\Sigma}_{cd}^{(k)} \;=\; \frac{1}{N_k} \sum_{n=1}^{N_k} (x_{nk,c} - \hat{\mu}_{k,c}) \cdot (x_{nk,d} - \hat{\mu}_{k,d}).$$

For the variances $\hat{\sigma}_{kd}^2$ (diagonal elements of $\hat{\Sigma}_k$) it holds:

$$\hat{\sigma}_{kd}^2 \;=\; \frac{1}{N_k} \sum_{n=1}^{N_k} (x_{nk,d} - \hat{\mu}_{k,d})^2.$$

Higher moments can be estimated only unreliably and therefore are seldom utilized.

3.4 Maximum-Likelihood Method

The method of moments fails for complex models. A general widely used method is the *Maximum-Likelihood method*.
The dependency of the model $p(x|k)$ on a parameter ϑ is written as

$$p_\vartheta(x|k) \quad \text{or} \quad p(x|k,\vartheta).$$

The parameter ϑ is often class dependent so that $p(x|k,\vartheta)$ is replaced by

$$p(x|k,\vartheta_k) \quad .$$

N_k training vectors are given for each class k:

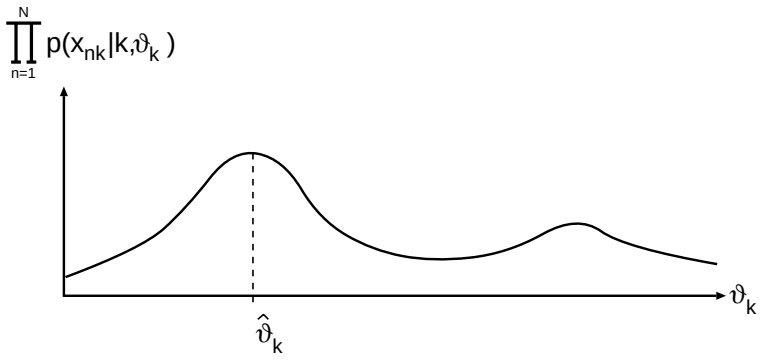
$$x_{1k}, \ldots, x_{nk}, \ldots, x_{N_k k} \qquad \in \mathbb{R}^D$$

The likelihood-function is then defined as follows:

$$\vartheta_k \;\longrightarrow\; \prod_{n=1}^{N_k} p(x_{nk}|k,\vartheta_k)$$

and also the log-likelihood-function:

$$\vartheta_k \;\longrightarrow\; \sum_{n=1}^{N_k} \log p(x_{nk}|k,\vartheta_k).$$



The Maximum-Likelihood estimate $\hat{\vartheta}_k$ is defined as:

$$\hat{\vartheta}_k \;:=\; \underset{\vartheta_k}{\operatorname{argmax}} \left\{ \prod_{n=1}^{N_k} p(x_{nk}|k,\vartheta_k) \right\}.$$

In words:
"$\hat{\vartheta}_k$ maximizes the probability of the model $p(x|k,\vartheta_k)$ for the given training data."
"$\hat{\vartheta}_k$ yields the *best* explanation of given the training data for the model $p(x|k,\vartheta_k)$."

$\hat{\vartheta}_k$ is often estimated by setting the derivation value to zero:

$$\sum_{n=1}^{N_k} \frac{\partial}{\partial \vartheta_k} \log p(x_{nk}|k,\vartheta_k) \;\stackrel{!}{=}\; 0$$

or also

$$\sum_{n=1}^{N_k} \frac{1}{p(x_{nk}|k,\vartheta_k)} \cdot \frac{\partial p(x_{nk}|k,\vartheta_k)}{\partial \vartheta_k} \;\stackrel{!}{=}\; 0$$

Strictly speaking, it should be checked at this point if a global maximum really exists.

Example: Gaussian distribution with diagonal covariance matrix

$x \in \mathbb{R}^D$ with average value vector $\mu_k \in \mathbb{R}^D$
variance vector $\sigma_k^2 \in \mathbb{R}^D$

$$p(x|k,\mu_k,\sigma_k^2) \;=\; \frac{1}{\prod_{d=1}^{D}\sqrt{2\pi\sigma_{kd}^2}}\exp\left[-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d-\mu_{kd}}{\sigma_{kd}}\right)^2\right]$$

Training data: $x_{1k},\ldots,x_{nk},\ldots,x_{N_kk}$

Log-Likelihood for class k:

$$\sum_{n=1}^{N_k}\log p(x_{nk}|k,\mu_k,\sigma_k^2) \;=\; -\frac{1}{2}\sum_{n=1}^{N_k}\left[\sum_{d=1}^{D}\left(\frac{x_{nk,d}-\mu_{kd}}{\sigma_{kd}}\right)^2+\sum_{d=1}^{D}\log\left(2\pi\sigma_{kd}^2\right)\right]$$

Derivation over μ_{kd}:

$$\frac{\partial}{\partial\mu_{kd}} \;=\; \sum_{n=1}^{N_k}\left(\frac{x_{nk,d}-\mu_{kd}}{\sigma_{kd}^2}\right)\stackrel{!}{=}0$$

which results in

$$\hat{\mu}_{kd} \;=\; \frac{1}{N_k}\sum_{n=1}^{N_k}x_{nk,d}$$

Derivation with respect to σ_{kd}^2:

$$\frac{\partial}{\partial\sigma_{kd}^2} \;=\; \frac{1}{2\sigma_{kd}^4}\sum_{n=1}^{N_k}(x_{nk,d}-\mu_{kd})^2-\sum_{n=1}^{N_k}\frac{1}{2\sigma_{kd}^2}\stackrel{!}{=}0$$

after inserting the estimate $\hat{\mu}_{kd}$ of μ_{kd}:

$$\hat{\sigma}_{kd}^2 \;=\; \frac{1}{N_k}\sum_{n=1}^{N_k}(x_{nk,d}-\hat{\mu}_{kd})^2$$

The usual estimates of average value and variance are obtained, which are identical to the estimates obtained using the method of moments (though $\frac{1}{N_k-1}$ is often used for the variance instead of $\frac{1}{N_k}$).

The estimated values resulted from a Gaussian distribution with an arbitrary covariance matrix Σ_k are the same as from the method of moments.

$$\begin{aligned} \hat{\mu}_{kd} &= \frac{1}{N_k} \sum_{n=1}^{N_k} x_{nk,d} \\ \hat{\Sigma}_{kc,d} &= \frac{1}{N_k} \sum_{n=1}^{N_k} (x_{nk,d} - \hat{\mu}_{kd})(x_{nk,c} - \hat{\mu}_{kc}) \end{aligned}$$

Derivation using the formula

$$\frac{\partial}{\partial A_{ij}} detA = (A^{-1})_{ij} \cdot detA$$

for an invertible matrix A.

As we will see later, estimating Σ_k can lead to a singular matrix $\hat{\Sigma}_k$. As a remedy, it is possible to presume a class independent covariance matrix Σ.

$$p(x|k, \mu_k, \Sigma) \; : \quad \text{Gaussian distribution with } \mu_k \text{ and } \Sigma$$

The following likelihood-function is obtained

$$\Sigma \quad \rightarrow \quad \prod_{K=1}^{K} \prod_{n=1}^{N_k} p(x_{nk}|k, \mu_k, \Sigma)$$

Maximum-Likelihood estimation results in:

$$\begin{aligned} \hat{\mu}_{kd} &= \frac{1}{N_k} \sum_{n=1}^{N_k} x_{nk,d} \\ \hat{\Sigma}_{cd} &= \frac{1}{N} \sum_{k=1}^{K} \left[\sum_{n=1}^{N_k} (x_{nk,d} - \hat{\mu}_{kd})(x_{nk,c} - \hat{\mu}_{kc}) \right] \\ & \text{where } N = \sum_{k=1}^{K} N_k \end{aligned}$$

Hints:

1. Each observation x_{nk} enters only once, after subtraction of the corresponding average value. Consequently, a contribution of a class with many observations is stronger than a contribution of a class with view observations; if this is not desired (e.g. because this class is *not* represented stronger in the test data), weights like $\sqrt[N_k]{\ldots}$ or $\frac{1}{N_k}$ should be used.

2. Interpretation: $\hat{\Sigma}$ results from the weighted averaging of class individual covariance matrices $\hat{\Sigma}_k$.

Other distributions for which simple estimates are obtained by the Maximum-Likelihood method:

- Laplace distribution:
$$p(x|\mu, v) = \frac{1}{2v_k} e^{-\frac{|x-\mu|}{v}},$$

- Binomial distribution:
$$p_N(n|\vartheta) = \left(\begin{array}{c} \mathrm{N} \\ \mathrm{n} \end{array} \right) \vartheta^n (1-\vartheta)^{N-n},$$

- Multinomial distribution:
$$\text{if} \quad \sum_{i=1}^{k} \vartheta_i = 1 \quad \text{and} \quad \sum_{i=1}^{k} n_i = N$$
$$p_N(n_1, \ldots, n_k|\vartheta_1, \ldots, \vartheta_k) = \frac{N!}{n_1! \cdot \ldots \cdot n_k!} \cdot \vartheta_1^{n_1} \cdot \ldots \cdot \vartheta_k^{n_k},$$

- Poisson distribution:
$$p(n|\lambda) = \frac{\lambda^n}{n!} e^{-\lambda}.$$

Exercise: Calculate the corresponding estimates.

Application of the multinomial distribution:

Language modeling (application examples: speech recognition, translation)
Consider a word sequence $w_1^N = w_1 \dots w_N$ and an acoustic vector $x_1^T = x_1 \dots x_T$
Determine

$$\underset{w_1^N}{\operatorname{argmax}}\{p(w_1^N) \cdot p(x_1^T|w_1^N)\}$$

with the language model probability

$$\begin{aligned} p(w_1^N) &= \prod_{n=1}^{N} p(w_n|w_1 \dots w_{n-1}) \\ &= \begin{cases} \prod_{n=1}^{N} p(w_n) & \text{for a unigram language model} \\ \prod_{n=1}^{N} p(w_n|w_{n-1}) & \text{for a bigram language model} \end{cases} \end{aligned}$$

If $N(w)$ is the frequency of the word w, then the log-likelihood function for an unigram-model is

$$\sum_{w=1}^{W} N(w) \log p(w) \ .$$

The frequency of a word sequence v, w is $N(v, w)$, and it has multinomial distribution. Probabilities should be normalized so that $\sum_w p(w|v) = 1 \ \forall v$. (There are different methods for assigning a probability > 0 also to unseen events, see the lecture *Language Modeling*.) The log-likelihood function for a bigram-model is

$$\begin{aligned} \sum_{n=1}^{N} \log p(w_n|w_{n-1}) &= \sum_{v,w} N(v, w) \log p(w|v) \\ &= \sum_{v} N(v) \sum_{w} \frac{N(v, w)}{N(v)} \log p(w|v) \ . \end{aligned}$$

$\frac{N(v,w)}{N(v)}$ is a distribution since $\sum_w \frac{N(v,w)}{N(v)} = 1$

$\rightsquigarrow$ ML-estimate $\hat{p}(w|v) = \frac{N(v,w)}{N(v)}$.

Poisson distribution: Modeling of "rare" events (e.g. word counts in a text)
Consider an unigram language model. If the count n_w for each word w has Poisson distribution with parameter λ_w, i.e.

$$p(n_w|\lambda_w) = \frac{\lambda_w^{n_w}}{n_w!} e^{-\lambda_w} \ .$$

then

$$\begin{aligned} p(n_1, \ldots, n_W|\lambda_1, \ldots, \lambda_W) &= \prod_{w=1}^{W} p(n_w|\lambda_w) = \prod_w (e^{-\lambda_w} \frac{\lambda_w^{n_w}}{n_w!}) \\ &= p(N|\lambda_1, \ldots, \lambda_W) \cdot \underbrace{p(n_1, \ldots, n_W|N, \lambda_1, \ldots, \lambda_W)}_{\text{multinomial distribution}} \end{aligned}$$

where $N := \sum_{w=1}^{W} n_w$.

Example: text classification
Text classes: $k = 1, \ldots, K$
New text: $w_1 \ldots w_N$ (where N is between 1000 and 10000)
Multinomial distribution: success probability ϑ_{wk} where $\sum_w \vartheta_{wk} = 1 \ \forall k$
Reduce the text to the counts $n_1 \ldots n_w \ldots n_W := n_1^W$.

$$\begin{aligned} \operatorname*{argmax}_k p(k|n_1^W) &= \operatorname*{argmax}_k \{p(k) \cdot p(n_1^W|k)\} \\ &= \operatorname*{argmax}_k \{p(k) \cdot p(n_1^W|\vartheta_{w=1;k}^W \ , N)\} \\ &\qquad \text{where } N = \sum_{w=1}^{W} n_w \text{ (multinomial distribution)} \\ &= \operatorname*{argmax}_k \{p(k) \cdot N! \prod_w \frac{\vartheta_{wk}^{n_k}}{n_w!}\} \\ &= \operatorname*{argmax}_k \{\log p(k) + \sum_w n_w \log \vartheta_{wk}\} \end{aligned}$$

(since $\log p(n_1^W|k) = \log N! + \sum_w n_w \log \vartheta_{wk} - \sum_w \log n_w!$)

3.5 Practical Aspects

1. Typical problems

   Typical problems will be shown on the example of a Gaussian model:

   - The variability of the learning sample is too small, so that the estimation of the variance is too small.
   - Due to this underestimation of the variance and the exponential decrease of the Gaussian model, the probability mass is too concentrated in the center.
   - The covariance matrix $\hat{\Sigma}_k$ can easily become singular:
     If we have N_k observation vectors, $\hat{\Sigma}_k$ is constructed of N_k vectors $(x_{nk} - \hat{\mu}_k)$, from those only $(N_k - 1)$ can be independent because of $\hat{\mu}_k$. That means that $\hat{\Sigma}_k$ has the rank $(N_k - 1)$ and is definitely singular if $N_k \leq D$.
     In practice it is required that:
     $$N_k \ \geq \ (10, \ldots, 100) \cdot D$$
     This singularity of $\hat{\Sigma}_k$ is the main reason why often only diagonal covariance matrices are utilized.

2. Counter measures in the case of singular $\hat{\Sigma}_k$

   - class independent covariance matrix $\hat{\Sigma}$:
     In this case a transformation of the coordinates can be advisable.
   - Smoothing $\hat{\Sigma}_k$ using $\hat{\Sigma}$:
     $$\tilde{\Sigma}_k := (1 - \lambda_k)\hat{\Sigma}_k + \lambda_k \hat{\Sigma} \ ,$$
     where λ_k is a class dependent weight parameter.
   - Emphasizing the diagonal elements of $\hat{\Sigma}_k$:
     $$\tilde{\Sigma}_k := (1 - \lambda_k)\hat{\Sigma}_k + \lambda_k \cdot diag(\hat{\Sigma}_k) \ ,$$
     assumed that all diagonal elements are greater than zero.

3. Structure of a system (using Gaussian models)

   - Calculate average values $\hat{\mu}_k$ and covariance matrices $\hat{\Sigma}_k$ for each class k.
   - Check if the covariance matrix $\hat{\Sigma}_k$ is singular or tends to be singular (by means of eigenvalues).
   - Smooth the covariance matrix $\hat{\Sigma}_k$ or calculate a class independent covariance matrix $\hat{\Sigma}$ (possibly including coordinate axes transformation).
   - if necessary:

     Emphasize diagonal elements of covariance matrices.
   - still open: choice of the weight parameter λ_k

     finally: Analyze the effect on the error rate (either for the training data or for another sample).
   - possible refinement (or unimodality test):

     Nearest-Neighbor classifier:

     Retain all training data and determine the next training vector for a test vector x which has to be classified. This requires a distance measure which contains the covariance matrix or its diagonal elements.

3.6 Evaluation Criteria: Empirical Error Rate

The most important evaluation criterion is the error rate of the system when it is practically applied. There are 2 sample types:

- the training or learning sample which is used for the design of the classifier, i.e. the choice of the model $p(x|k, \vartheta_k)$ and the estimation of parameters ϑ_k;
- the test sample which is used to measure the reliability of the classifier by means of

$$\text{(empirical) error rate} = \frac{\text{number of recognition errors}}{\text{number of recognition tests}}$$

A Theoretical calculation of the error rate without a test sample is difficult because generally the form of the models $p(x|k, \vartheta_k)$ is not known.

Considerations:

- The strict separation of training and test sample is absolutely necessary for an objective determination of the error rate.

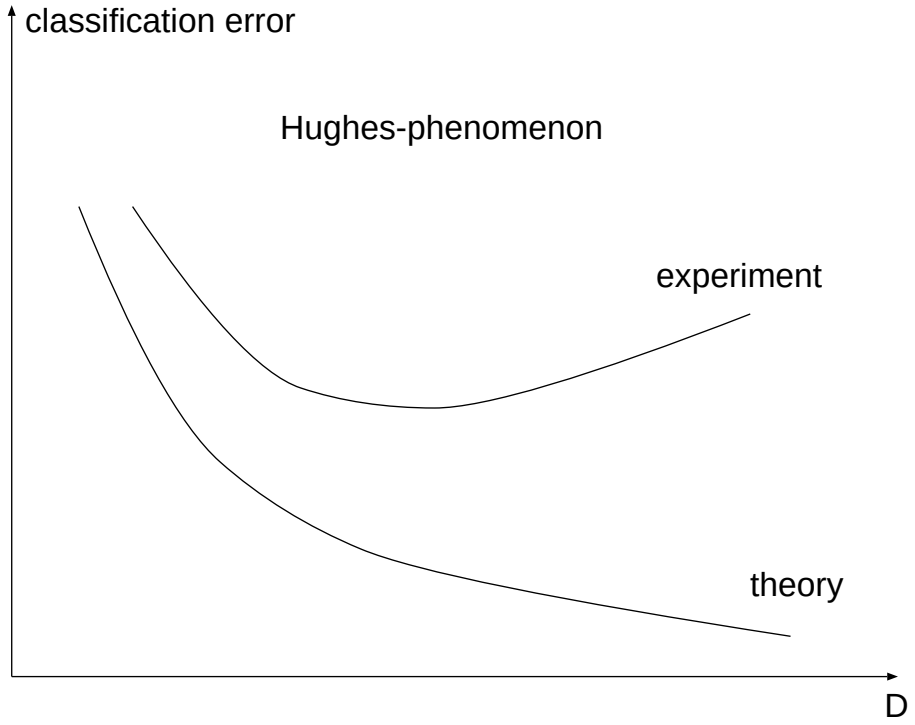
  Example: The Nearest-Neighbor classifier has always a zero error rate on the training sample.

- Repeated tests on the same test data can sometimes be delusive because the system is then optimized on this special test data. This effect is called *training on the testing data*.

- For the purpose of scientific objectivity, the error rate must be measured on a so far unseen sample because the error rate should enable a prediction for new tests. In statistics, such methods are known as *Cross-Validation*.

3.7 Dependency on Dimension off the Error Rate

Consider the following behavior of the theoretically expected and experimentally obtained error rate in dependence on the number of feature components D:



"Hughes-Phenomenon": For the constant training set, the error rate first decreases as the number of feature components increases. From a certain point it starts to increase although according to the theory dimension increase leads to a error rate reduction.

Explanation: if more parameters have to be estimated, more errors can occur so that the classification error rate can increase due to wrong estimations.

Theoretical view of the error rate: additional feature component:

$$x_1^D \in I\!R^D \rightarrow x_1^{D+1} \in I\!R^{D+1}$$

Bayes Error Rate:

$$p_D(e) = 1 - \int\limits_{x_1^D} dx_1^D \max_k p(x_1^D, k)$$

$$\begin{aligned}
1 - p_D(e) &= \int\limits_{x_1^D} dx_1^D \max_k p(x_1^D, k) \\
&= \int\limits_{x_1^D} dx_1^D \max_k [\ p(x_1^D, k) \underbrace{\int\limits_{x_{D+1}} dx_{D+1}\ p(x_{D+1}|x_1^D, k)}_{=1}\ ] \\
&= \int\limits_{x_1^D} dx_1^D \max_k [\ \int\limits_{x_{D+1}} dx_{D+1}\ p(x_1^{D+1}, k)\ ] \\
&\quad (\text{where} \max_u \sum_i g_i(u) \leq \sum_i \max_u g_i(u)\ ) \\
&\leq \int\limits_{x_1^D} dx_1^D \int\limits_{x_{D+1}} dx_{D+1} \max_k p(x_1^{D+1}, k) \\
&= \int\limits_{x_1^{D+1}} dx_1^{D+1} \max_k p(x_1^{D+1}, k) \\
&= 1 - p_{D+1}(e)
\end{aligned}$$

That results in:

$$p_{D+1}(e) \ \leq \ p_D(e)$$

This means, when the number of feature components increases, the error rate can theoretically only be reduced or kept unchanged. Equality is trues especially when x_{D+1} depends on x_1^D, i.e. $x_{D+1} = f(x_1^D)$.

3.8 Bayes Learning

The following two assumptions are the starting point for this method:

1. ϑ_k in $p(x|k, \vartheta_k)$ is treated itself as a random variable.
2. There is an a-priori distribution $p(\vartheta_k)$ for the parameter i.e. the random variable ϑ_k (this is given here).

   a-priori $\hat{=}$ before the actual measuring

This a-priori distribution $p(\vartheta_k)$ is based on

- previous measurements or experiments,
- or previous knowledge about the task that is also acquired from previous experiments.

Example:
Many systems for writing or speech recognition are user-dependent, i.e. each user has to train the system personally. Such a system can be trained with a sample spoken by many speakers so that the distribution $p(\vartheta_k)$ of the speaker-dependent parameter ϑ_k can be estimated.

Advantage: such a system can be efficiently trained with only a small set of speaker-dependent training material.

The Bayes parameter estimation itself results from the a-posteriori probability
$p(\vartheta_k|x_{1k}, \ldots, x_{N_k k})$ for the training data x_{nk} of class k where $n = 1, \ldots, N_k$.

Distribution of the joint events $(\vartheta_k; x_{1k}, \ldots, x_{N_k k})$:

$$\begin{aligned} p(\vartheta_k; x_{1k}, \ldots, x_{N_k k}) &= p(\vartheta_k) \cdot p(x_{1k}, \ldots, x_{N_k k}|\vartheta_k) \\ &= p(\vartheta_k) \cdot \prod_{n=1}^{N_k} p(x_{nk}|\vartheta_k) \end{aligned}$$

with the model $p(x_{nk}|\vartheta_k)$

Definition and calculation of the a-posteriori distribution:

$$\begin{aligned} p(\vartheta_k|x_{1k},\ldots,x_{N_k k}) &= \frac{p(\vartheta_k)\prod_{n=1}^{N_k} p(x_{nk}|\vartheta_k)}{\int d\vartheta'_k\; p(\vartheta'_k)\prod_{n=1}^{N_k} p(x_{nk}|\vartheta'_k)} \\ &= \frac{p(\vartheta_k)\prod_{n=1}^{N_k} p(x_{nk}|\vartheta_k)}{const(\vartheta_k)} \end{aligned}$$

Only the numerator is directly dependent on ϑ_k, and it is equal to the product of a-priori distribution and likelihood.
The a-posteriori distribution is often reduced to one single value; following values are especially used:

- Posterior mean:

$$\hat{\vartheta}_k^{Mean} = \int d\vartheta_k\; \vartheta_k\; p(\vartheta_k|x_{1k},\ldots,x_{N_k k})$$

($\hat{\vartheta}_k^{Mean}$ minimizes the cost function $\int d\vartheta_k\; [\vartheta_k - \hat{\vartheta}_k]^2\; p(\vartheta_k|x_{1k},\ldots,x_{N_k k})$)

- Posterior maximum: (also called *MAP-estimate*, *Maximum-A-Posteriori estimate*)

$$\hat{\vartheta}_k^{Max} = \underset{\vartheta_k}{\operatorname{argmax}}\; \{p(\vartheta_k|x_{1k},\ldots,x_{N_k k})\}$$

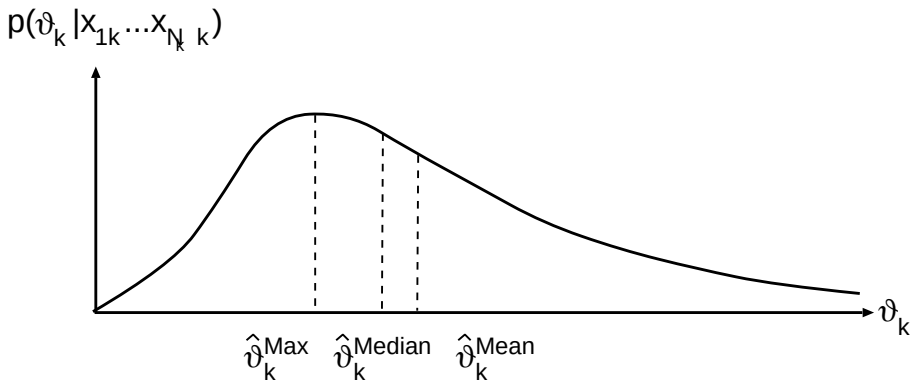
- Posterior median:

$$\hat{\vartheta}_k^{Median} = \underset{\vartheta_k}{\operatorname{argmin}} \int d\vartheta_k\; |\vartheta_k - \hat{\vartheta}_k|\; p(\vartheta_k|x_{1k},\ldots,x_{N_k k})$$

If a symmetric distribution has global optimum in the center of symmetry (like e.g. Gaussian distribution), then holds:

$$\hat{\vartheta}_k^{Mean} = \hat{\vartheta}_k^{Max} = \hat{\vartheta}_k^{Median}$$

Illustration:



Special case: if $p(\vartheta_k) = const(\vartheta_k)$, then holds

$$\hat{\vartheta}_k^{Max} \;=\; \hat{\vartheta}_k^{Max.Lik.}.$$

Example: univariate Gaussian distribution
Class index k is omitted for simplification.
Assumptions:

- Gaussian distribution: $p(x|\mu, \sigma^2), \quad x \in \mathbb{R}$
  with unknown μ and known σ^2.
- The a-priori distribution $p(\mu)$ for the parameter μ is also Gaussian:
  $p(\mu|\mu_0, \sigma_0^2)$ with *hyper-parameters* μ_0 and σ_0^2.

We calculate the a-posteriori distribution for the training data $x_1, \ldots, x_N$:

$$\begin{aligned} p(\mu|x_1,\ldots,x_N) &= const(\mu)\cdot p(\mu|\mu_0,\sigma_0^2)\cdot\prod_{n=1}^{N} p(x_n|\mu,\sigma^2) \\ &= const'(\mu)\cdot\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2-\frac{1}{2}\sum_{n=1}^{N}\left(\frac{x_n-\mu}{\sigma}\right)^2\right] \quad (*) \\ &= const'(\mu)\cdot \\ &\quad \exp\left[-\frac{1}{2}\left\{\frac{\mu^2}{\sigma_0^2}-\frac{2\mu\mu_0}{\sigma_0^2}+\frac{\mu_0^2}{\sigma_0^2}+\frac{N\mu^2}{\sigma^2}-\frac{2\mu\sum_{n=1}^{N}x_n}{\sigma^2}+\frac{\sum_{n=1}^{N}x_n^2}{\sigma^2}\right\}\right] \end{aligned}$$

Equation $(*)$ shows that the hyper-parameters (μ_0, σ_0^2), similar to the pairs (x_n, σ^2), are determining the a-posteriori distribution (x_n, σ^2), i.e. from measurements x_n and variance σ^2.

Transformation of the exponent to a quadratic expression of μ results in:

$$p(\mu|x_1,\ldots,x_N) = \frac{1}{\sqrt{2\pi\sigma_N^2}}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_N}{\sigma_N}\right)^2\right]$$

with suitable expressions

$$\begin{aligned} \sigma_N^2 &\equiv \sigma_N^2(\mu_0,\sigma_0^2;x_1,\ldots,x_N;N), \\ \mu_N &\equiv \mu_N(\mu_0;\sigma_0^2,x_1,\ldots,x_N;N). \end{aligned}$$

Comparison of coefficients for μ^2:

$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2}+\frac{N}{\sigma^2}$$

We obtain

$$\sigma_N^2 = \frac{\sigma^2 \sigma_0^2}{\sigma^2 + N\sigma_0^2}$$

Comparison of coefficients for μ:

$$\frac{\mu_N}{\sigma_N^2} = \frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2} \sum_n x_n$$

Transformation and inserting σ_N^2 results in:

$$\mu_N = \left(1 - \frac{\sigma^2}{\sigma^2 + N\sigma_0^2}\right) \frac{1}{N} \sum_{n=1}^{N} x_n + \frac{\sigma^2}{\sigma^2 + N\sigma_0^2} \cdot \mu_0$$

This expression describes an averaging between

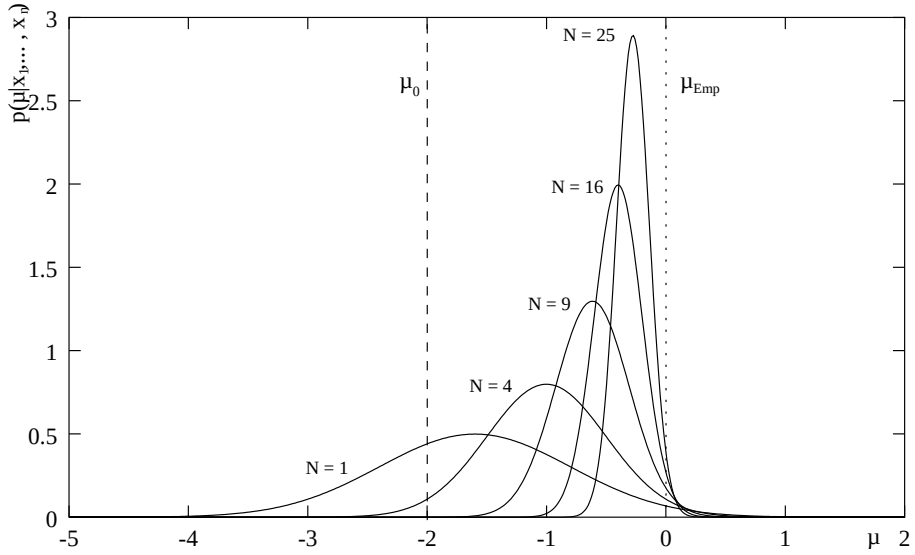
- empirical average value $\frac{1}{N}\sum_{n=1}^{N} x_n$ with weight $\left(1 - \frac{\sigma^2}{\sigma^2 + N\sigma_0^2}\right)$
- a-priori average value μ_0 with weight $\frac{\sigma^2}{\sigma^2 + N\sigma_0^2}$.

If $N \to \infty$, influence of the a-priori distribution $p(\mu|\mu_0, \sigma_0^2)$ tends to zero, and the a-posteriori distribution $p(\mu|\mu_N, \sigma_N^2)$ results in the Delta-function:

$$N \to \infty \quad \begin{array}{l} p(\mu|\mu_N, \sigma_N^2) \\ \Big\downarrow \\ \text{Delta-function} \end{array} \quad \begin{array}{l} \mu_N \to \frac{1}{N}\sum_{n=1}^{N} x_n \\ \\ \sigma_N^2 \to 0 \end{array}$$

Graphical representation: as N increases, the a-priori distribution becomes more concentrated around μ_N.
Example:

$p(\mu|x_1, \ldots, x_N)$ where $\mu_0 = -2,\ \sigma_0 = 1,\ \sigma = 2,\ \mu_{Emp} = \frac{1}{N}\sum_{n=1}^{N} x_n = 0$



The general case is more difficult:

- multivariate (instead of univariate) Gaussian distribution;
- μ and Σ are both unknown at the same time;
- in general the covariance matrix is not diagonal.

Details can be found in:

- [Fuk90], p.389-394
- [Kee65]

In this and in similar cases, the new estimation is obtained (exact or as approximation) by *linear interpolation* of:

- an estimate which is based on real measurements,
- and a suitable parameter of the a-priori distribution.

How is the Bayes learning utilized in recognition?
We mention two methods:

1. So far the *plug-in method* is (implicitly) presumed:

   apply $p(x|k,\hat{\vartheta}_k)$ with $\hat{\vartheta}_k = \hat{\vartheta}_k(x_{1k},\ldots,x_{N_kk})$,

   i.e. with an estimate $\hat{\vartheta}_k$ which is obtained using Bayes estimation method from the training data $x_{1k},\ldots,x_{N_kk}$ of class k.

2. Another method (though rarely utilized) is based on the *predictive distribution* $p(x|k;x_{1k},\ldots,x_{N_kk})$:

$$p(x|k;x_{1k},\ldots,x_{N_kk}) \;:=\; \int\limits_{\vartheta_k} d\vartheta_k \; p(x|k,\vartheta_k)\cdot p(\vartheta_k|k;x_{1k},\ldots,x_{N_kk}) \;,$$

   where $p(\vartheta_k|k;x_{1k},\ldots,x_{N_kk})$ is the a-posteriori distribution of the parameter ϑ_k, now with additional class index k.

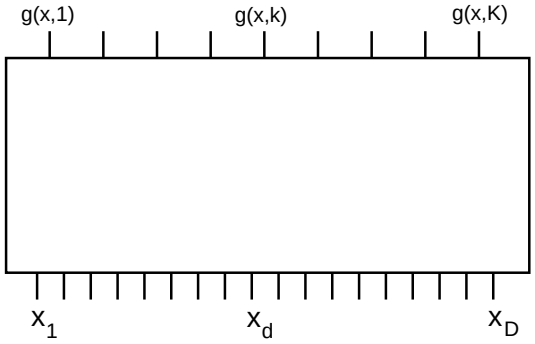
   In practice, closed solutions for the integrals which occur in this method exist only in exceptional cases. Details can be found in the already mentioned books: [DH73], p.55-59, [Kee65] and [Ney99b].

4 Discriminants and Neural Networks

4.1 Squared Error Criterion

The learning methods which are treated in Chapter 3 are developed for distribution models $p(x|k, \vartheta_k)$. At first, there is no underlying statistical model for general discriminants $g(x, k)$.

Reminder: discriminant approach



Decision rule: $r(x) = \underset{k}{\operatorname{argmax}} \{g(x, k)\}$

The function $g(x, k)$ depends on the parameter ϑ:

$$g(x, k) = g_\vartheta(x, k).$$

These parameters are obtained in learning phase.

In this phase, the ideal output values are assumed:

$$\begin{aligned} g(x, k) &\stackrel{!}{=} 1, && \text{if } x \text{ belongs to the class } k, \\ g(x, k) &\stackrel{!}{=} 0, && \text{if } x \text{ does not belong to the class } k. \end{aligned}$$

Normally, these ideal values can be achieved only approximatively. During the learning phase the squared error criterion is (normally) used for a training vector x_n which belongs to class k_n:

$$\sum_{c=1}^{K} [g(x_n, c) - \delta(k_n, c)]^2$$

with the Kronecker-function:

$$\delta(k,c) = \begin{cases} 1\,, & c = k \quad \text{(correct)} \\ 0\,, & c \neq k \quad \text{(wrong)} \end{cases} \quad .$$

Instead of one single training vector, there is a whole set:

$$\text{training vectors:} \qquad (x_1,k_1),\ldots,(x_n,k_n),\ldots,(x_N,k_N)$$

(Pay attention to the notation changes in comparison to the Chapter 3.)

We sum up the errors over all training vectors and define:

$$F_N(g) \;:=\; \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{K}\left[g(x_n,c)-\delta(k_n,c)\right]^2$$

The writing $F_N(g)$ should express:

- the dependency on the function $(x,k) \longrightarrow g(x,k)$, i.e. its functional form and its parameters;
- the dependency on the training data (x_n,k_n) where $n = 1,\ldots,N$.

For the limit case $N \longrightarrow \infty$ ensues:

$$\begin{aligned} F(g) &= \lim_{N\longrightarrow\infty}\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{K}\left[g(x_n,c)-\delta(k_n,c)\right]^2 \\ &= \int_x dx\;\sum_{k=1}^{K} p(x,k)\cdot\sum_{c=1}^{K}\left[g(x,c)-\delta(k,c)\right]^2 \\ &= \int_x dx\;p(x)\cdot\underbrace{\sum_{k=1}^{K} p(k|x)\cdot\sum_{c=1}^{K}\left[g(x,c)-\delta(k,c)\right]^2}_{=:\,e(x)} \end{aligned}$$

$e(x)$ is the local error at the point x.

Transforming expression for the local error $e(x)$:

$$\begin{aligned} e(x) &= \sum_{k=1}^{K} p(k|x) \cdot \sum_{c=1}^{K} \left[g(x,c) - \delta(k,c)\right]^2 \\ &= \sum_{k=1}^{K} p(k|x) \cdot \left(\left[g(x,k) - 1\right]^2 + \sum_{c \neq k} g^2(x,c) \right) \\ &= \sum_{k=1}^{K} p(k|x) \cdot \left(1 - 2\, g(x,k) + \sum_{c=1}^{K} g^2(x,c) \right) \\ &= 1 - 2 \sum_{c=1}^{K} p(c|x) \cdot g(x,c) + \sum_{c=1}^{K} g^2(x,c) \\ &= 1 - \sum_{c=1}^{K} p^2(c|x) + \sum_{c=1}^{K} \left[p(c|x) - g(x,c)\right]^2 \end{aligned}$$

According to the equation above, $e(x)$ consists of two parts:

- $1 - \sum_{c=1}^{K} p^2(c|x)$: independent of the discriminant function $g(x,k)$ (appears in other cases as well; in context of Gini-criterion, Nearest-Neighbor error)
  Upper limit for Bayes error:

$$\begin{aligned} p_B(e|x) &= 1 - \max_k p(k|x) \\ &= 1 - \sum_{c=1}^{K} p(c|x) \max_k p(k|x) \\ &\leq 1 - \sum_{c=1}^{K} p(c|x) p(c|x) \\ &= 1 - \sum_{c=1}^{K} p^2(c|x) \end{aligned}$$

- $\sum_{c=1}^{K} [p(c|x) - g(x,c)]^2$: squared error criterion between $g(x,c)$ and the a-posteriori distribution $p(c|x)$ at the point $x \in \mathbb{R}^D$.

The expression for $F(g)$ is:

$$F(g) \;=\; \int\limits_x dx\; p(x) \cdot \left(1 - \sum_{c=1}^{K} p^2(c|x) + \sum_{c=1}^{K} [p(c|x) - g(x,c)]^2\right)$$

Notes about $F(g)$:

- Global optimum is achieved if $g(x,c) = p(c|x) \quad \forall (x,c)$, i.e. if the values of the discriminant function are *exactly* identical to the a-posteriori class probabilities.

  For the global optimum g_0 holds:

  $$F(g_0) = \int\limits_x p(x) \cdot \left(1 - \sum_{c=1}^{K} p^2(c|x)\right)\; dx\;,$$

  i.e. it is not possible to fall below this value (as error criterion).

- To achieve the global optimum (approximatively), the discriminant function should be suitably chosen in view of
  - functional form (*structure*) and
  - number of free parameters ϑ.

- If the equality $g(x,c) = p(c|x)$ is not achievable for each (x,c), the error will be minimized primarily at points x at which $p(x)$ has a large value.

- Finding the (global) optimum is a difficult mathematical problem; iterative methods are used for it:
  - Gradient method (numerical mathematics)
  - Error Back Propagation (*stochastic gradient*)

4.2 Structures and Multilayer-Perceptron

Simple structures for the discriminant function $(x, k) \longrightarrow g(x, k)$ are (see Chapter 1):

$$\text{linear} \quad : \quad g(x,k) = \alpha_{k0} + \sum_{d=1}^{D} \alpha_{kd} \cdot x_d$$

$$\text{quadratic} : \quad g(x,k) = \alpha_{k0} + \sum_{d=1}^{D} \alpha_{kd} \cdot x_d + \sum_{d=1}^{D} \sum_{e=1}^{D} \beta_{kde} \cdot x_d x_e$$

with the parameters α_{kd} and β_{kde}.

These structures can be analogically defined as higher polynoms of x, in this way the *polynomial or regression classifiers* are obtained.
(*Regression* because of the squared error criterion).

Advantage: There is an efficient mathematical formalism for the parameter estimation.

Disadvantage: Polynoms do not have good interpolation capabilities (known from the approximation of functions). They are not smooth enough and they easily fail for new data not seen before in training.

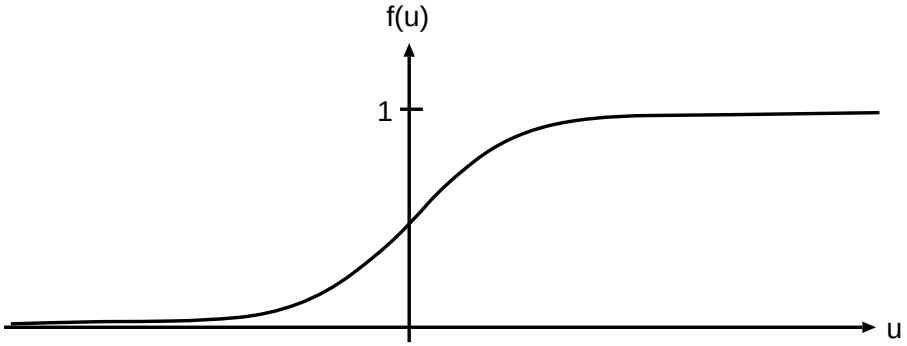
The disadvantages are (significantly) avoided using the *Multilayer-Perceptron.*

Structure of the Multilayer-Perceptron:

- There are several layers with nodes (*multilayer*):
  - output layer (outputs $g(x, k), \quad k = 1, \ldots, K$);
  - hidden layers;
  - input layer (inputs $x_d, \quad d = 1, \ldots, D$).
- In each node i (*neuron*) of a layer l, the corresponding inputs $y_j^{(l-1)}$ are linearly combined with weights $\alpha_{ij}^{(l)}$ (*scalar product*):

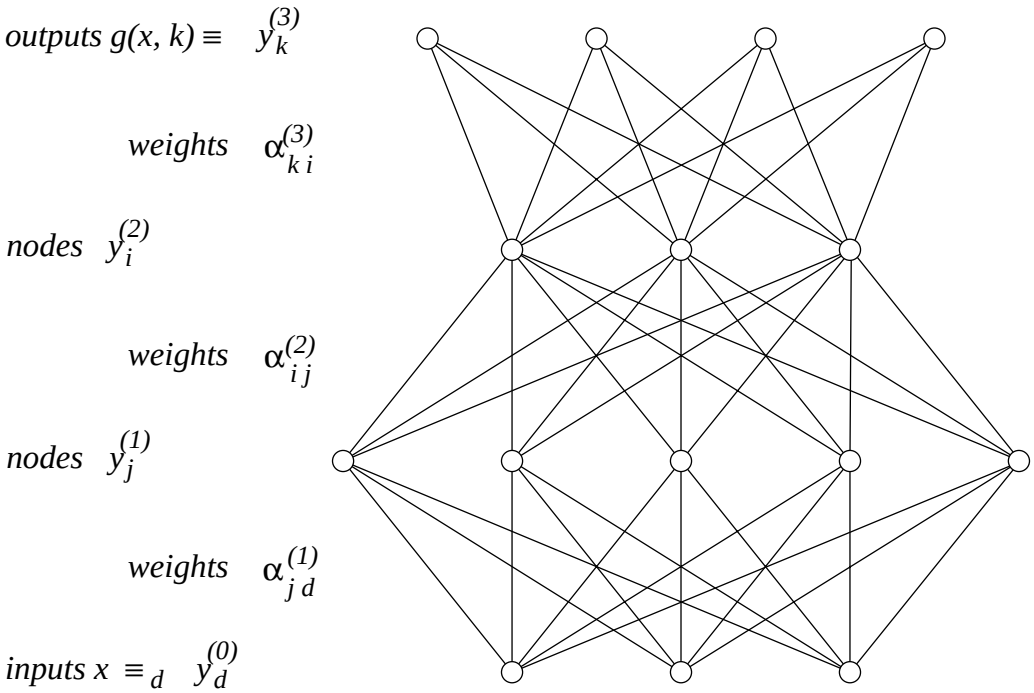
$$\sum_j \alpha_{ij}^{(l)} \cdot y_j^{(l-1)}$$

Each such linear combination passes through a non-linear function $u \longrightarrow f(u)$:



A linear discriminant $x \longrightarrow g(x, k)$ is also called (linear) perceptron.

Illustration of a Multilayer-Perceptron (example with 2 hidden layers):



$$\begin{aligned}
\text{Outputs (layer 3)} \quad &: \quad y_k^{(3)} = f\left(\sum_i \alpha_{ki}^{(3)} \cdot y_i^{(2)}\right) \\
\text{Node layer 2} \quad &: \quad y_i^{(2)} = f\left(\sum_j \alpha_{ij}^{(2)} \cdot y_j^{(1)}\right) \\
\text{Node layer 1} \quad &: \quad y_j^{(1)} = f\left(\sum_d \alpha_{jd}^{(1)} \cdot y_d^{(0)}\right) \\
\text{Inputs (layer 0)} \quad &: \quad y_d^{(0)} = x_d
\end{aligned}$$

Thus the L-layer Multilayer-Perceptron realizes a discriminant function:

$$\begin{aligned}
g(\cdot, k): \; & \mathbb{R}^D \longrightarrow \mathbb{R} \\
& x \longmapsto g(x,k) = y_k^{(L)}(x)
\end{aligned}$$

In order to include a constant (*bias, offset*) in the linear combination, a fictive constant node value $y_0^{(l)} \equiv 1$ is added in each layer, e.g.:

$$\begin{aligned}
\sum_j \alpha_{ij}^{(l)} \cdot y_j^{(l-1)} &= \sum_{j=0}^{J} \alpha_{ij}^{(l)} \cdot y_j^{(l-1)} \\
&= \alpha_{i0}^{(l)} + \sum_{j=1}^{J} \alpha_{ij}^{(l)} \cdot y_j^{(l-1)}.
\end{aligned}$$

4.2.1 Non-Linearity

The *Sigmoid-function* is typically chosen as function f:

$$f(u) = \frac{1}{1+e^{-u}} \quad .$$

And its derivation is calculated as:

$$f'(u) = \frac{e^{-u}}{(1+e^{-u})^2} = f(u) \cdot (1 - f(u)).$$

In probability theory/statistics, the function $f(u)$ is known as *logistic* distribution.

The arc tangent can also be used as alternative to the Sigmoid-function (quadratic decrease of derivation):

$$\begin{aligned} f'(u) &= \frac{1}{\pi} \cdot \frac{1}{1+u^2} \\ f(u) &= \frac{1}{\pi} \cdot \arctan(u) + \frac{1}{2} \; . \end{aligned}$$

Other variants are also possible, but are seldom utilized:

- Gaussian model:

$$\begin{aligned} f'(u) &= \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{u^2}{2}\right) \\ f(u) &= \int_{-\infty}^{u} dz \; f'(z) \; . \end{aligned}$$

- *elementary* functions:

$$\begin{aligned} f(u) &= \frac{1}{2}\left[1 + \frac{u}{1+|u|}\right] \\ f'(u) &= \frac{1}{2}\frac{1}{(1+|u|)^2} \qquad \text{for } u \neq 0 \; . \end{aligned}$$

The non-linearity of the Sigmoid-function can be easily illustrated. The following series expansion is known:

$$\frac{1}{1+z} = 1 - z + z^2 - z^3 + \ldots \qquad\qquad |z| < 1 \; .$$

Application to the Sigmoid-function (expansion about the point $u_0 = 0$):

$$\begin{aligned} f(u) &= \frac{1}{1+e^{-u}} \\ &= \frac{1}{2} \cdot \frac{1}{1+\frac{1}{2}(e^{-u}-1)} \\ &\tilde{=} \frac{1}{2}\left[1 - \left(\frac{e^{-u}-1}{2}\right) + \left(\frac{e^{-u}-1}{2}\right)^2 - \left(\frac{e^{-u}-1}{2}\right)^3 + \dots\right] \end{aligned}$$

Interpretation: The Sigmoid-function generates the non-linear factors in a *specific* manner.

4.2.2 Notes about the Multilayer-Perceptron (MLP)

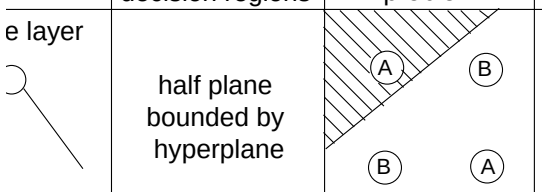
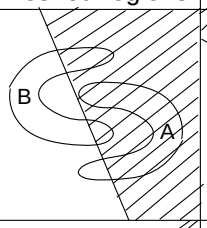
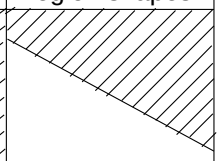
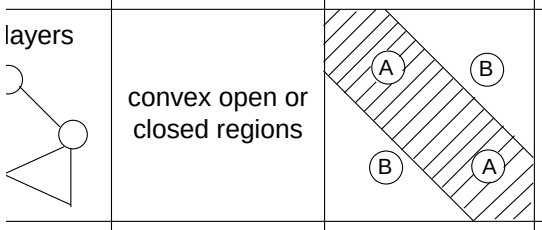
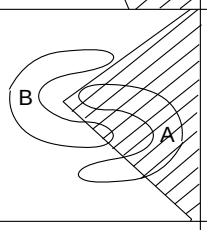
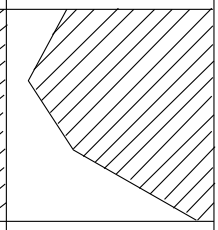
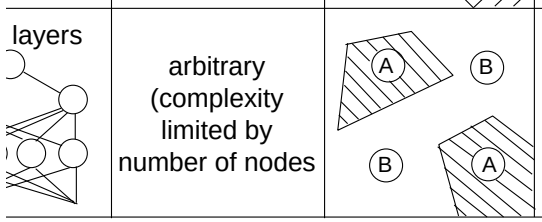
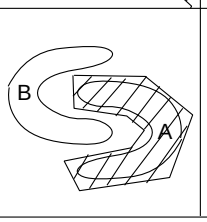
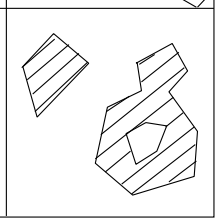
- The Multilayer-Perceptron is the prototype of an *artificial* neural network. The attribute *artificial* signifies the difference from biological neural networks.
- The advantages of (artificial) neural networks are often summarized in the following way:
  - orientation to the biological example,
  - massive parallelism ("pdp" $\hat{=}$ parallel distributed processing),
  - discriminative training,
  - approximation of arbitrary input-output dependencies.
- The same advantages also hold for many non-neural approaches, but the idea about discriminative training was revived through neural networks, especially in speech recognition.

Structure of the Multilayer-Perceptron: For the choice of

- number of hidden layers – already one hidden layer can bring significant improvements – and
- number of nodes per layer

there are basically only empirical statements.

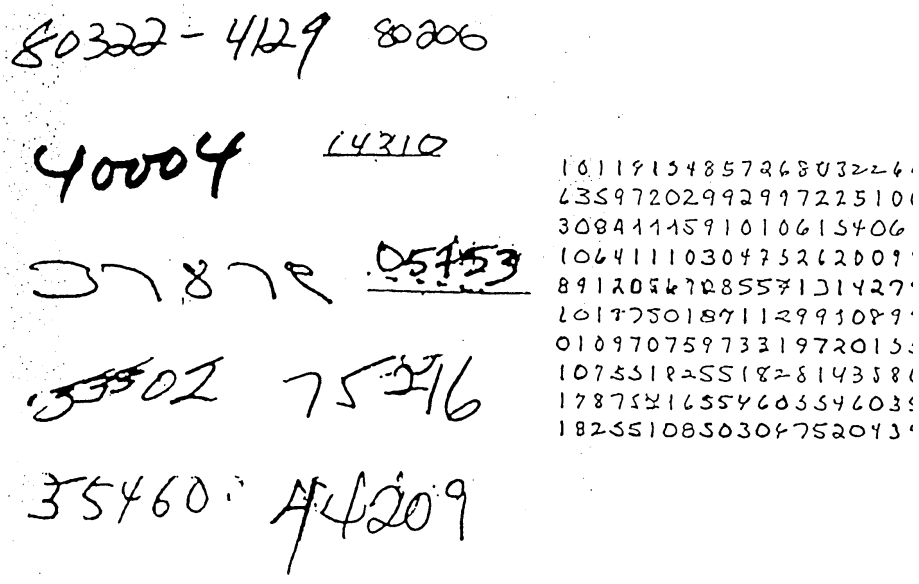
Formal inspection of the number of layers (1987)

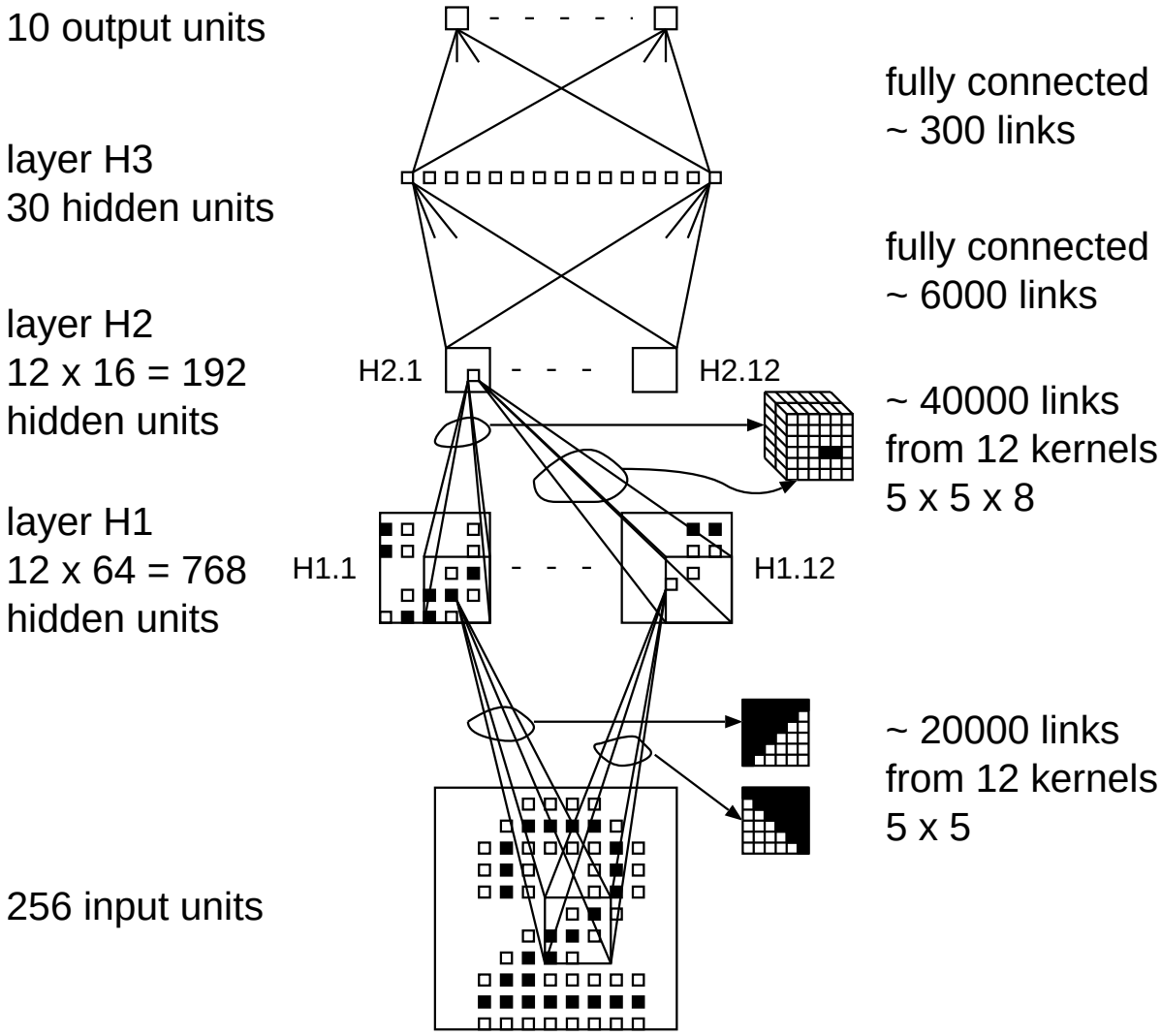
cture	type of decision regions	exclusive OR problem	classes with meshed regions	most general region shapes
e layer	half plane bounded by hyperplane			
layers	convex open or closed regions			
layers	arbitrary (complexity limited by number of nodes			

Note: In general there are more than 2 classes; accordingly the class boundaries are getting more complex.

It is useful to prescribe symmetric dependencies between weights if that results from the problem definition.

Example: Multilayer-Perceptron for character recognition by Yan le Cun.




10 output units
layer H3
30 hidden units
layer H2
12 x 16 = 192
hidden units
layer H1
12 x 64 = 768
hidden units
256 input units
H2.1
H2.12
H1.1
H1.12
fully connected
~ 300 links
fully connected
~ 6000 links
~ 40000 links
from 12 kernels
5 x 5 x 8
~ 20000 links
from 12 kernels
5 x 5

Building the Multilayer-Perceptron:

Input layer:	16×16 pixels (grey values)
Layer H1:	Each node sees an image part of 5×5 pixels. The shifting of 2 pixels results in 8×8 cells. 12 planes are assumed, each with 8×8 cells. Parameters per plane: 25 weights (independent of the cell) 64 biases (offsets) total: $12 \times (25 + 64) = 1068$ free parameters
Layer H2:	also consists of 12 planes, each with 4×4 cells. Each H2-cell sees 8 of the 12 H1-planes, 5×5 cells in the 8 planes. Parameters per plane: $8 \cdot 5 \cdot 5 = 200$ weights (independent of the cell) $4 \cdot 4 = 16$ biases (offsets) total: $12 \times (200 + 16) = 2592$ free parameters

Idea:
H2 can build more complex features using features from H1.

Layer H3:	consists of 30 cells, each of them sees all $12 \cdot 16 = 192$ cells of H2.
Output layer:	consists of 10 cells, each of them sees all 30 cells of H3.

hidden nodes:

H1:	768	nodes
H2:	192	nodes
H3:	30	nodes
total:	990	nodes

Weights (*links*):

H1:	$768 \cdot 25 =$	19200
H2:	$192 \cdot 200 =$	38400
H3:	$30 \cdot 192 =$	5760
output layer:	$10 \cdot 30 =$	300
total:		63660

independent parameters:

H1:	1068
H2:	2592
H3:	5790
output layer:	310
total:	9760

This example shows the building of structures in the Multilayer-Perception.

4.2.3 Multilayer-Perceptron Similar Structures for Statistical Classificators

Independent of neural networks, there are approaches which are leading to similar structures for the a-posteriori probabilities $p(k|x)$.
From Chapter 2 the following is known:

$$\begin{aligned} p_\vartheta(k|x) &= \frac{p_\vartheta(k) \cdot p_\vartheta(x|k)}{\sum\limits_{c=1}^{K} p_\vartheta(c) \cdot p_\vartheta(x|c)} \\ &= \frac{1}{1 + \sum\limits_{c \neq k} \frac{p_\vartheta(c) \cdot p_\vartheta(x|c)}{p_\vartheta(k) \cdot p_\vartheta(x|k)}} \\ &= \frac{1}{1 + \sum\limits_{c \neq k} \frac{p_\vartheta(c)}{p_\vartheta(k)} \cdot \exp\left[\log p_\vartheta(x|c) - \log p_\vartheta(x|k)\right]} \end{aligned}$$

Note: $p_\vartheta(k, x)$ is only a model distribution – there are no statements about the true probability $p(k, x)$ here.

If the difference $[\log p_\vartheta(x|c) - \log p_\vartheta(x|k)]$ is a linear function of x, then there are operations resulting from that are similar to the operations within a Multilayer-Perceptron node. This can be achieved with:

1. log-linear models: $\log p_\vartheta(x|k) = \sum\limits_d \alpha_{kd} \cdot x_d$

   (Note: x is mostly discrete, because otherwise some problems considering normalization $\int\limits_{\mathbb{R}^D} p_\vartheta(x|k)\, dx = 1$ usually occur.)

2. Gaussian models with $\Sigma_k = \Sigma$ (class-independent)

Exercise: Specify the neuronal structures for 1. and 2.

4.3 Error Back Propagation

Error Back Propagation is the most widely used method for the training of a Multilayer-Perceptron and it was suggested by Rumelhart, Hinton and Williams in the 1980s. There was also previous work: [Ama67], [Wer74], [Par82].
Shortly summarized, it holds:

Error Back Propagation = stochastic gradient and chain rule

Starting point is the squared error criterion for the training data (x_n, c_n) where $n = 1, \ldots, N$:

$$\begin{aligned} E &= \sum_{n=1}^{N} E_n && \text{(global squared error)} \\ E_n &= \frac{1}{2} \sum_{k=1}^{K} \Big[y_k^{(L)}(x_n) - \delta(k, c_n) \Big]^2 && \text{(local squared error)} \end{aligned}$$

If for $1 \leq l \leq L$ holds: $\quad y_i^{(l)} = f(z_i^{(l)}) \quad$ where $\quad z_i^{(l)} = \sum_j \alpha_{ij}^{(l)} \cdot y_j^{(l-1)}$,

Then $\dfrac{\partial z_i^{(l)}}{\partial \alpha_{ij}^{(l)}} = y_j^{(l-1)}$.

Different cases:

a) Output layer L:

$$\begin{aligned} \frac{\partial E_n}{\partial \alpha_{ki}^{(L)}} &= \frac{\partial E_n}{\partial y_k^{(L)}} \cdot \frac{\partial y_k^{(L)}}{\partial \alpha_{ki}^{(L)}} \\ &= \frac{\partial E_n}{\partial y_k^{(L)}} \cdot \frac{\partial y_k^{(L)}}{\partial z_k^{(L)}} \cdot \frac{\partial z_k^{(L)}}{\partial \alpha_{ki}^{(L)}} \\ &= \Big[y_k^{(L)}(x_n) - \delta(k, c_n) \Big] \cdot f'(z_k^{(L)}) \cdot y_i^{(L-1)} \end{aligned}$$

$$\text{where } \frac{\partial E_n}{\partial y_k^{(L)}} = y_k^{(L)}(x_n) - \delta(k, c_n)$$

b) hidden layer $1 \leq l < L$:

At first for the highest hidden layer $L-1$:

$$\begin{aligned}
\frac{\partial E_n}{\partial \alpha_{ij}^{(L-1)}} &= \frac{\partial E_n}{\partial y_i^{(L-1)}} \cdot \frac{\partial y_i^{(L-1)}}{\partial \alpha_{ij}^{(L-1)}} \\
&= \frac{\partial E_n}{\partial y_i^{(L-1)}} \cdot \frac{\partial y_i^{(L-1)}}{\partial z_i^{(L-1)}} \cdot \frac{\partial z_i^{(L-1)}}{\partial \alpha_{ij}^{(L-1)}} \\
&= \frac{\partial E_n}{\partial y_i^{(L-1)}} \cdot f'(z_i^{(L-1)}) \cdot y_j^{(L-2)}
\end{aligned}$$

Determining the partial derivation $\partial E_n / \partial y_j^{(L-1)}$ of the local squared error E_n after the output $y_i^{(L-1)}$ of the node i in the hidden layer $L-1$ is now more complicated. For the calculation of $\partial E_n / \partial y_j^{(L-1)}$ the following functional dependency has to be taken into account:

$$y_j^{(L-1)} \longrightarrow y_{1\ldots K}^{(L)}(y_j^{(L-1)}) \longrightarrow E_n\big(y_{1\ldots K}^{(L)}(y_j^{(L-1)})\big).$$

That means, if there is a change in $y_i^{(L-1)}$ i.e. in the inner node i of the hidden layer $L-1$, all outputs $y_k^{(L)}$ are changed. The following derivation results from that:

$$\begin{aligned}
\frac{\partial E_n}{\partial y_i^{(L-1)}} &= \sum_k \frac{\partial E_n}{\partial y_k^{(L)}} \cdot \frac{\partial y_k^{(L)}}{\partial z_k^{(L)}} \cdot \frac{\partial z_k^{(L)}}{\partial y_i^{(L-1)}} \\
&= \sum_k \frac{\partial E_n}{\partial y_k^{(L)}} \cdot f'(z_k^{(L)}) \cdot \frac{\partial}{\partial y_i^{(L-1)}} \sum_{i'} \alpha_{ki'}^{(L)} \cdot y_{i'}^{(L-1)} \\
&= \sum_k \frac{\partial E_n}{\partial y_k^{(L)}} \cdot f'(z_k^{(L)}) \cdot \alpha_{ki}^{(L)}
\end{aligned}$$

The derivations of the local error E_n in respect to output in deeper nodes ensue analogically by further recursive application of the chain rule, which is illustrated in the following example:

$$y^{(0)}_{1\ldots D} \longrightarrow y^{(1)}_{1\ldots J} \longrightarrow \ldots \longrightarrow y^{(L-1)}_{1\ldots I} \longrightarrow y^{(L)}_{1\ldots K} \longrightarrow E_n$$

$$\frac{\partial E_n}{\partial y^{(L)}_k}$$

$$\downarrow$$

$$\frac{\partial E_n}{\partial y^{(L-1)}_i} = \sum_k \frac{\partial E_n}{\partial y^{(L)}_k} \frac{\partial y^{(L)}_k}{\partial y^{(L-1)}_i}$$

$$\downarrow$$

$$\vdots$$

$$\frac{\partial E_n}{\partial y^{(1)}_j} = \cdots$$

$$\downarrow$$

$$\frac{\partial E_n}{\partial y^{(0)}_d} = \sum_j \frac{\partial E_n}{\partial y^{(1)}_j} \frac{\partial y^{(1)}_j}{\partial y^{(0)}_d}$$

Note: The derivations $\dfrac{\partial E_n}{\partial y^{(l)}_j}$ are calculated *recursively*, first for the output layer and then step by step for each layer (*Error Back Propagation).*

Update of the parameters $\alpha^{(l)}_{ij}$:

$$\bar{\alpha}^{(l)}_{ij} = \alpha^{(l)}_{ij} - \gamma \cdot \frac{\partial E_n}{\partial \alpha^{(l)}_{ij}} \qquad \text{with step length } \gamma \ .$$

The practical implementation of the Error Back Propagation requires suitable heuristics, regarding:

- choice of initial values,
- choice of step length (usually decreasing when the number of iteration increases),
- avoiding bad local optima,
- controlling the convergence speed.

Altogether, the realization and implementation require considerable care.

4.4 Discriminative Training for Statistical Classificators

Discriminative training denotes criteria or methods which try to use a-posteriori probability as learning criterion, e.g.

$$\vartheta \;\longrightarrow\; \prod_{n=1}^{N} p(k_n|x_n,\vartheta)$$

or logarithmized

$$\vartheta \;\longrightarrow\; \sum_{n=1}^{N} \log p(k_n|x_n,\vartheta),$$

where ϑ denotes the unknown parameters.

Note the analogy and also the difference in comparison to Maximum-Likelihood:

$$\begin{aligned} \operatorname*{argmax}_{\vartheta} \sum_{n} \log p(k_n|x_n,\vartheta) \qquad & \textit{Maximum Class Posterior Probability} \\ \hat{=} \quad & \textit{Maximum Mutual Information, MMI} \end{aligned}$$

$$\operatorname*{argmax}_{\vartheta} \sum_{n} \log p(x_n|k_n,\vartheta) \qquad \textit{Maximum Likelihood}$$

For $p(k_n|x_n, \vartheta)$ holds:

$$p(k_n|x_n, \vartheta) = \frac{p(k_n) \cdot p(x_n|k_n, \vartheta)}{\sum_{c=1}^{K} p(c) \cdot p(x_n|c, \vartheta)}$$

Terminology:
Discriminative, since $p(k_n|x_n, \vartheta)$ captures the class overlap.

For the calculation of ϑ approximative methods have to be utilized, e.g. corresponding modification of the Error Back Propagation. As initial value for ϑ e.g. the Maximum-Likelihood estimate can be used.

Exercise: Formulate the Error Back Propagation for the criterion above where $p(x|k, \mu_k, \Sigma_k)$ are Gaussian distributions with diagonal covariance matrices Σ_k.

4.5 Error Rate and Discriminative Training

Model distribution: $p_\vartheta(k|x)$ (true distribution: p) with $x \in \mathbb{R}^D$, class k, parameters ϑ and normalization $\sum_k p_\vartheta(k|x) = 1 \;\; \forall\, x, \vartheta$.

Decision rule r:

$$x \;\longmapsto\; r_x \;=\; \operatorname*{argmax}_k p_\vartheta(k|x)$$

Error rate for decision rule $x \longmapsto r_x$ (local error):

- decision correct (with probability):
  $p(r_x|x)$ where $p(k|x)$ is *true distribution*
- local error rate for decision rule r:

$$\boxed{p_r(''e''|x) \;=\; 1 - p(r_x|x)}$$

- Bayes error rate:

$$\begin{aligned} p_B(''e''|x) &= \min_{x \mapsto r_x} [1 - p(r_x|x)] \\ &= 1 - \max_{x \mapsto r_x} p(r_x|x) \\ &= 1 - \max_k p(k|x) \end{aligned}$$

- global error rate for decision rule r:

$$\begin{aligned} p_r(''e'') &= \int\limits_{\mathbb{R}^D} dx p(x) \cdot p_r(''e''|x) \\ &\left[= \int\limits_{\mathbb{R}^D} p_r(''e'', x) \right] \end{aligned}$$

Maximum Mutual Information

Estimation of the Bayes error rate for class log-posterior probabilities (see *Maximum-Mutual-Information* criterion, Chapter 4.4):

- **local:**

$$\begin{aligned} p_B(''e''|x) &= 1 - \max_k p(k|x) \\ &= 1 - \underbrace{\sum_c p_\vartheta(c|x)}_{=1} \cdot \max_k p(k|x) \\ &\leq 1 - \sum_c p_\vartheta(c|x) \cdot p(c|x) \\ &= \sum_c p(c|x) \ [1 - p_\vartheta(c|x)] \\ &\leq -\sum_c p(c|x) \ \log p_\vartheta(c|x) \qquad \text{since } \log y \leq y-1 \end{aligned}$$

- **global:**

$$\begin{aligned} p_B(''e'') &\leq \int\limits_{\mathbb{R}^D} dx \ p(x) \sum_c p(c|x) \ [-\log p_\vartheta(c|x)] \\ &= \int\limits_{\mathbb{R}^D} dx \sum_c p(c, x) \ [-\log p_\vartheta(c|x)] \end{aligned}$$

Consider training data (x_n, k_n), $n = 1, \ldots, N$. This corresponds to the empirical expected value or a empirical estimation of the error rate for class log-posterior probabilities:

$$\begin{aligned} & \hat{p}_B(''e'') \;\leq\; \frac{1}{N} \sum_{n=1}^{N} (-\log p_\vartheta(k_n|x_n)) \qquad \forall\, \vartheta \\ \Rightarrow \quad & \hat{p}_B(''e'') \;\leq\; \min_\vartheta \left[-\frac{1}{N} \sum_{n=1}^{N} \log p_\vartheta(k_n|x_n) \right] \\ & \qquad\quad = \; \frac{1}{N} \max_\vartheta \sum_{n=1}^{N} \log p_\vartheta(k_n|x_n) \end{aligned} \Bigg\} \Rightarrow \text{ closer to the Bayes error rate}$$

$$\min_{p_\vartheta(c|x)} \left\{ -\sum_c p(c|x) \log p(c|x) \right\} \quad \text{results in } \hat{p}_\vartheta(c|x) \;=\; p(c|x)$$

(comparable with divergence inequality $\sum_i p_i \log q_i \;\leq\; \sum_i p_i \log p_i$)

Squared error criterion

local:

$$\begin{aligned} p_B(''e''|x) \;&=\; 1 - \max_k p(k|x) \\ &=\; 1 - \sum_c p(c|x) \max_k p(k|x) \\ &\leq\; 1 - \sum_c p^2(c|x) \end{aligned}$$

NN with $p_\vartheta(c|x) \mathrel{\hat{=}} g(x, c)$

$$e(x) \;=\; 1 - \sum_c p^2(c|x) + \underbrace{\sum_c [p(c|x) - p_\vartheta(c|x)]^2}_{\geq 0}$$

$$\begin{aligned} p_B(''e''|x) &\leq 1 - \sum_c p^2(c|x) + \sum_c [p(c|x) - p_\vartheta(c|x)]^2 \\ &= \sum_k p(k|x) \cdot \sum_c [p_\vartheta(c|x) - \delta(c, k_n)]^2 \end{aligned}$$

With training data (x_n, k_n):

$$\boxed{p_B(''e'') \leq \frac{1}{N} \sum_{n=1}^{N} \sum_c [p_\vartheta(c|x) - \delta(c, k_n)]^2}$$

Training criterion yields upper limit for Bayes error rate.

5 Model-free methods

5.1 Introduction

Terminology: Model-free methods are also called "non-parametric".

For random numbers $x \in \mathbb{R}^D$ in Chapter 2 and 3 we have used special models $p(x|k, \vartheta_k)$,

e.g. Gaussian or similar distributions.

Main characteristic of those methods is the strong reduction of the training data to few parameters using

$$\text{estimated values} \qquad \hat{\vartheta}_k = \hat{\vartheta}_k(x_{1k}, \ldots, x_{N_k k}).$$

The other extreme is achieved by the model-free methods with the following properties:

- there are (almost) no special assumptions about the distribution;
- the true distributions can possibly be approximated arbitrarily exact.

A possible disadvantage is the increased use of CPU-time and memory requirement.

Distributions which can be described by model-free methods:

- class-dependent distributions $p(x|k)$,
- a-posteriori distributions $p(k|x)$,
- ...

Typical model-free methods:

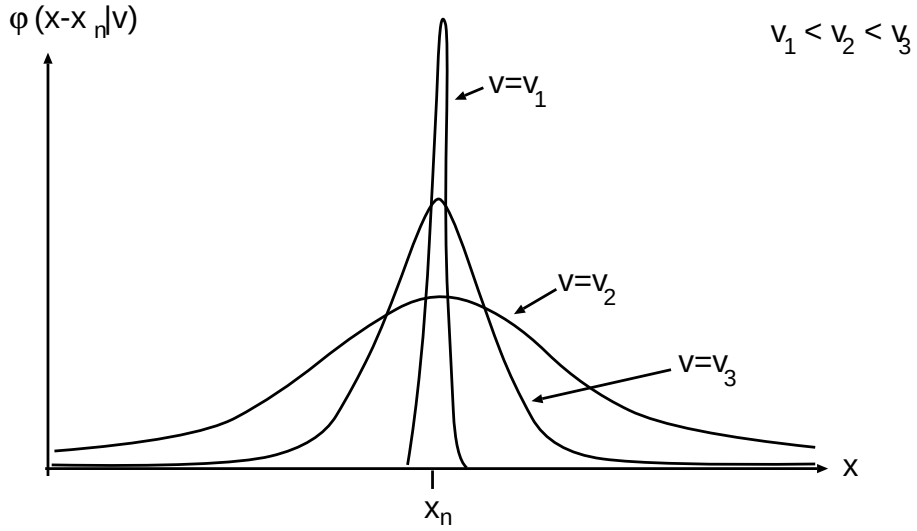
- *Kernel Densities* or *Parzen Densities*;
- Next-Neighbor rule for the classification;
- Discriminative methods, e.g. polynomial classificators or Multilayer-Perceptron (Chapter 4);
- Mixture distributions (Chapter 6).

The transitions between *model-free* and *model-based* methods are of course fluent. If a Multilayer-Perceptron has too few nodes (parameters), then strictly speaking it is not a model-free method anymore.

5.2 Kernel Densities

The starting point is a sort of histogram approximation of the training data x_n with $n = 1, \ldots, N$ in $\mathbb{R}^D$.

Let $x_n \in \mathbb{R}$ for simplicity.



The contribution of a training vector x_n is *distributed* over its environment. This distribution is described by a function φ which is called *Kernel Density, Parzen Density* or *Parzen Window*:

$$\begin{aligned} \varphi : \mathbb{R}^D &\longrightarrow [0, \infty[\\ x &\longmapsto \varphi(x) \end{aligned}$$

where

- $\varphi(x) \geq 0$,
- $\int\limits_{\mathbb{R}^D} \varphi(x)\, dx = 1.$

$\varphi(x)$ itself is a distribution which is chosen "smooth" and unimodal.

In general, the distribution φ can depend on the class k, then it is written $\varphi_k(x)$. Typical choice for $\varphi_k(x)$ is the Gaussian distribution:

$$\varphi_k(x) \;=\; \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi v_{kd}^2}} \exp\left[-\frac{1}{2}\left(\frac{x_d}{v_{kd}}\right)^2\right] = \varphi(x|v_k)$$

with suitable class-dependent variances v_{kd}.

An estimate of the distribution $p(x|k)$ is chosen as

$$\hat{p}(x|k) \;=\; \frac{1}{N_k} \sum_{n=1}^{N_k} \varphi_k(x - x_{nk}) \;,$$

where $x_{1k}, \ldots, x_{N_k k}$ are the training data for the class k.

The variances v_{kd} still have to be estimated. The Maximum-Likelihood estimation does not work because the maximum is obtained for $v_{kd} \to 0$.

Remedy:
Combination with *leaving-one-out*, but rather unmanageable equations might occur.

Possible way out is to try to introduce a class-dependent factor α_k and to multiply the usual estimated variances with this factor α_k:

$$v_{kd}(\alpha_k) \;=\; \alpha_k \cdot \frac{1}{N_k} \sum_{n=1}^{N_k} (x_{nk,d} - \hat{\mu}_{kd})^2 \;.$$

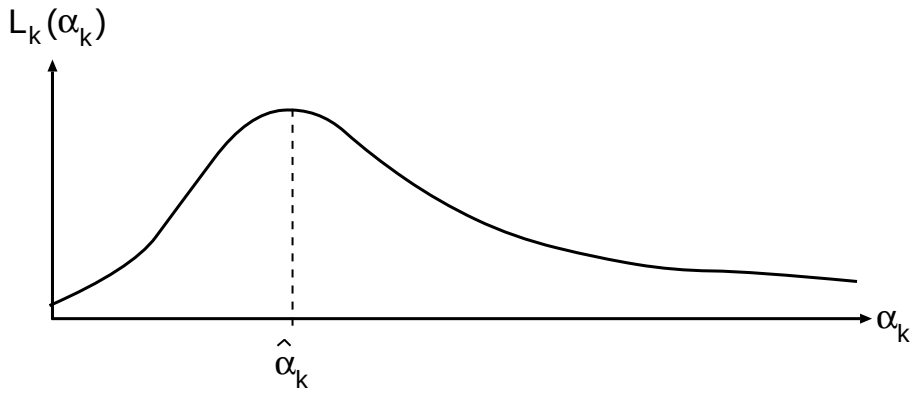
The optimal value for α_k can be now found by the simple method (in combination with leaving-one-out):

$$\alpha_k \;\longrightarrow\; L_k(\alpha_k) := \sum_{m=1}^{N_k} \log f_m(x_m)$$

$$\text{where} \quad f_m(x) := \frac{1}{N_k - 1} \sum_{\substack{n=1 \\ n \neq m}}^{N_k} \varphi(x - x_n | v_k(\alpha_k)) \;.$$

An approximative solution can be obtained using the maximum-approximation, i.e. by replacing the sum $f_m(x)$ with the maximum:

$$f_m(x) \;\tilde{=}\; \frac{1}{N_k - 1} \max_{\substack{n=1,\ldots,N_k \\ n \neq m}} \{\varphi(x - x_n | v_k(\alpha_k))\}$$



Exercise:
Calculate the factors α_k for the case that the Kernel Density is a Gaussian-distribution with diagonal covariance matrix.

Refinements:
Like for Gaussian distributions in Chapter 2, it is possible to define the dependency $u \longrightarrow \varphi_k(u)$ using distances ($u \in \mathbb{R}^D$):

- radially symmetric:

$$\varphi_k(u) \;:=\; \varphi_k(u^T u) = \varphi_k(||u||^2)$$

- elliptically symmetric with weight matrix A_k:

$$\varphi_k(u) \;:=\; \varphi_k((A_k u)^T (A_k u)) = \varphi_k(u^T A_k^T A_k u)$$

Examples for Kernel Densities in radially symmetric formalism (without variances for simplicity):

- Gauss: $\varphi(u) = \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2}\|u\|^2\right)$;
- t-distribution: $\varphi(u) = \frac{A}{(B + ||u||^2)^M}$, $A, B, M > 0$.

The method is closely related to the potential-function where the Coulomb-potential (from electrical engineering) is often applied:

$$\varphi(u) = \frac{1}{\epsilon + \|u\|} , \qquad \epsilon > 0 ,$$

(problematic because of the normalization).

5.3 Next Neighbor Rule (NN-rule)

- more precise: *nearest neighbor decision rule*, and also 1-NN in contrast to k-NN.
- not to be mixed up with *Minimum distance* e.g. for Gaussian distribution (see Chapter 2.5, S. 37) – for these methods the distance between test vector and average value of the training vectors is considered, but for 1-NN the distance between test vector and each single training vector is considered

We derive the Next Neighbor rule from the Kernel Densities (another derivation can be found in literature).

Training data for each class k: $x_{1k}, \ldots, x_{N_k k}$

As estimate for $p(x|k)$ we use the Kernel Density:

$$\hat{p}(x|k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \varphi(||x - x_{nk}||),$$

with $\varphi : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ and suitable distance $||\ ||$.
Further on, we assume for simplicity that the Kernel Density φ is class-independent.

Estimated value of $p(k)$:

$$\hat{p}(k) \;=\; \frac{N_k}{N} \qquad \text{where } \; N = \sum_{k=1}^{K} N_k \; .$$

We insert $\hat{p}(x|k)$ and $\hat{p}(k)$ in the Bayes decision rule:

$$\begin{aligned}
r(x) \;&=\; \underset{k}{\text{argmax}} \, \{\hat{p}(k) \cdot \hat{p}(x|k)\} \\
&=\; \underset{k}{\text{argmax}} \, \{\frac{N_k}{N} \cdot \frac{1}{N_k} \sum_{n=1}^{N_k} \varphi(||x - x_{nk}||)\} \\
&=\; \underset{k}{\text{argmax}} \, \{\frac{1}{N} \sum_{n=1}^{N_k} \varphi(||x - x_{nk}||)\} \\
&=\; \underset{k}{\text{argmax}} \, \{\sum_{n=1}^{N_k} \varphi(||x - x_{nk}||)\} \\
&\qquad \text{(maximum approximation: replace } \textstyle\sum \text{ with max)} \\
&\approx\; \underset{k}{\text{argmax}} \, \{\max_{n=1,\ldots,N_k} \varphi(||x - x_{nk}||)\} \\
&\qquad (z \longrightarrow \varphi(z) \text{ is monotonously decreasing}) \\
&=\; \underset{k}{\text{argmin}} \, \{\min_{n=1,\ldots,N_k} ||x - x_{nk}||\}
\end{aligned}$$

This approximation is called Next Neighbor Rule, because the class chosen for the vector x is the class of the next neighbor in the training data x_{nk}:

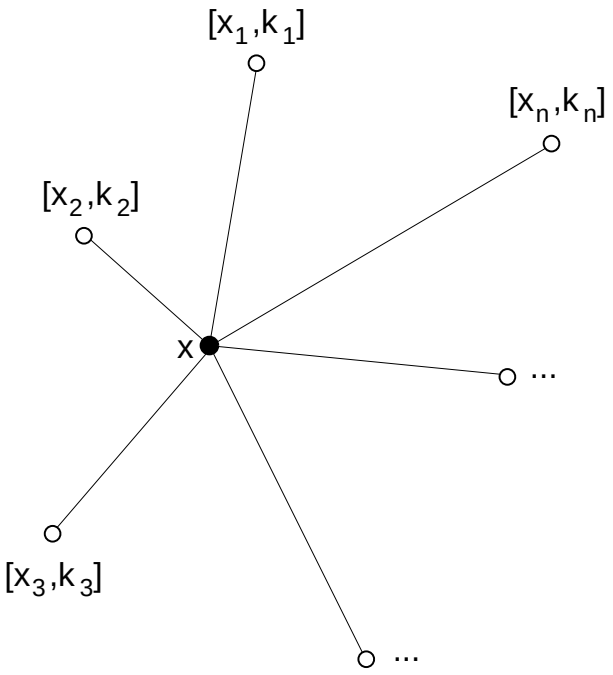
$$r(x) \;=\; \underset{k}{\text{argmin}} \, \{\min_{n=1,\ldots,N_k} ||x - x_{nk}||\}$$

Note:

- The NN-rule depends only on the chosen distance measure $||x - y||$: due to the maximum approximation of the sum and the monotony of φ the slope of φ does not have any influence on the result.
- Kernel Densities generally yield better results

- Furthermore, the maximum operation itself does not reduce computational time imperatively, since for the calculation of the maximum all distances have to be determined anyway. The computational time can be reduced only by using additional "tricks", see below.

Illustration of the NN-rule for training vectors $[x_n, k_n]$
where $n = 1, \ldots, N$ and with a test vector x:



There are (successful and unsuccessful) approaches for reducing computational time, i.e. the number of distance calculations:

1. early truncation by summation over the dimensions (*partial distance*);

2. triangular equality;

3. hierarchical structuring of the training vectors;

4. k-dimensional binary search trees (*k-d trees*); partitioning search space into disjunct hyper-cuboids (see [Ben75], [FBF77])

5. reducing the number of training vectors, e.g. "editing": consideration only vectors nearby the class boundaries

5.4 Error Rate of Next Neighbor Rule

More precise: error rate in asymptotic Al limit case when the number N of the training data tends to infinity [CT91].

Error rate for arbitrary decision rule:

$$\begin{aligned} p(error) &= \int dx\; p(x) \sum_k p(k|x)\, p(error|k,x) \\ &\qquad \left(\quad p(e|k,x) \;=\; \begin{cases} 1 & \text{if } k \neq r_x \\ 0 & \text{if } k = r_x \end{cases} \quad \right) \\ &= \int dx\; p(x) \sum_{k \neq r_x} p(k|x) \\ &= \int dx\; p(x)\; [1 - p(r_x|x)] \end{aligned}$$

At first we consider the error rate for the Bayes decision rule (Chapter 2):

$$r_B(x) \;=\; \underset{k}{\operatorname{argmax}} \{p(k|x)\} \qquad \text{for } x \in \mathbb{R}^D.$$

For the local error $p_B(e|x)$ we obtain:

$$\begin{aligned} p_B(e|x) &= "p_{Bayes}(error|x)" \\ &= 1 - p_B(correct|x) \\ &= 1 - \max_k p(k|x) \\ &= 1 - p(r_B(x)|x) \end{aligned}$$

The total error rate $p_B(e)$ results from integration over the whole space:

$$\begin{aligned} p_B(e) &= \int_x dx\ p(x) \cdot p_B(e|x) \\ &= \int_x dx\ \left[1 - \max_k p(k|x)\right] \cdot p(x) \\ &= \int_x dx\ 1 \cdot p(x) - \int_x dx\ \max_k p(k|x) \cdot p(x) \\ &= 1 - \int_x dx\ \max_k p(k,x) \end{aligned}$$

For the Next Neighbor (NN) rule we define:

x_{NN} : next neighbor of the considered test vector x, i.e.

$$x_{NN} := \underset{x_n : n=1,\dots,N}{\operatorname{argmin}} \{||x - x_n||\},$$

with the training vectors $[x_n, k_n], \quad n = 1, \dots, N.$

For the local error rate $p_{NN}(e|x, x_{NN})$ of the NN-rule for given x and hence calculated x_{NN} results from

$$\begin{aligned} p_{NN}(e|x, x_{NN}) &= \sum_k \left[p(k|x) \cdot \sum_{c \neq k} p(c|x_{NN}) \right] \\ &= \sum_k p(k|x) \cdot [1 - p(k|x_{NN})] \\ &= 1 - \sum_k p(k|x) \cdot p(k|x_{NN}) \ . \end{aligned}$$

In limit case $N \longrightarrow \infty$, i.e. with enough training data holds

$$p(k|x_{NN}) = p(k|x) \ .$$

In this way the local error rate is simplified

$$p_{NN}(e|x) = 1 - \sum_k p^2(k|x) \ ,$$

with the definition:

$$p_{NN}(e|x) := \lim_{N \to \infty} p_{NN}(e|x, x_{NN}).$$

We estimate the local error rate of the asymptotic NN-rule using the Bayes error rate in the following way:

- estimating the lower bound of the NN-error:

$$\begin{aligned} p_B(e|x) &= 1 - \max_k p(k|x) \\ &= 1 - \sum_c p(c|x) \cdot \max_k p(k|x) \\ &\leq 1 - \sum_k p^2(k|x) \\ &= p_{NN}(e|x) \ . \end{aligned}$$

- estimating the upper bound of the NN-error:

$$\begin{aligned} p_{NN}(e|x) &= 1 - \sum_k p^2(k|x) \\ &= 1 - p^2(r_B(x)|x) - \sum_{k \neq r_B(x)} p^2(k|x) \end{aligned}$$

We apply a special case of the Cauchy-Schwarz inequality:

$$\begin{aligned} \left[\sum_{k \neq r_B(x)} p(k|x) \right]^2 &= \left[\sum_{k \neq r_B(x)} 1 \cdot p(k|x) \right]^2 \\ &\leq \sum_{k \neq r_B(x)} 1^2 \cdot \sum_{k \neq r_B(x)} p^2(k|x) \\ &= (K-1) \cdot \sum_{k \neq r_B(x)} p^2(k|x) \end{aligned}$$

and therefore

$$\begin{aligned} \sum_{k \neq r_B(x)} p^2(k|x) &\geq \frac{1}{K-1} \left[\sum_{k \neq r_B(x)} p(k|x) \right]^2 \\ &= \frac{1}{K-1} \cdot \left[1 - p(r_B(x)|x) \right]^2 \ . \end{aligned}$$

Inserting the inequality in $p_{NN}(e|x)$ results in:

$$\begin{aligned} p_{NN}(e|x) &\leq 1 - p^2(r_B(x)|x) - \frac{1}{K-1} \cdot [1 - p(r_B(x)|x)]^2 \\ &= \ldots \\ &= 2 \cdot [1 - p(r_B(x)|x)] - \frac{K}{K-1} \cdot [1 - p(r_B(x)|x)]^2 \end{aligned}$$

$$p_{NN}(e|x) \leq 2 \cdot p_B(e|x) - \frac{K}{K-1} \cdot p_B^2(e|x)$$

For the total error rate we need an additional inequality: if $f(x)$ is a function of random variable x, then holds:

$$Var\{f(x)\} = E\{f^2(x)\} - E^2\{f(x)\} \geq 0$$

and thus

$$E^2\{f(x)\} \leq E\{f^2(x)\} \, .$$

Integration of both estimates over the whole space of both estimates results in:

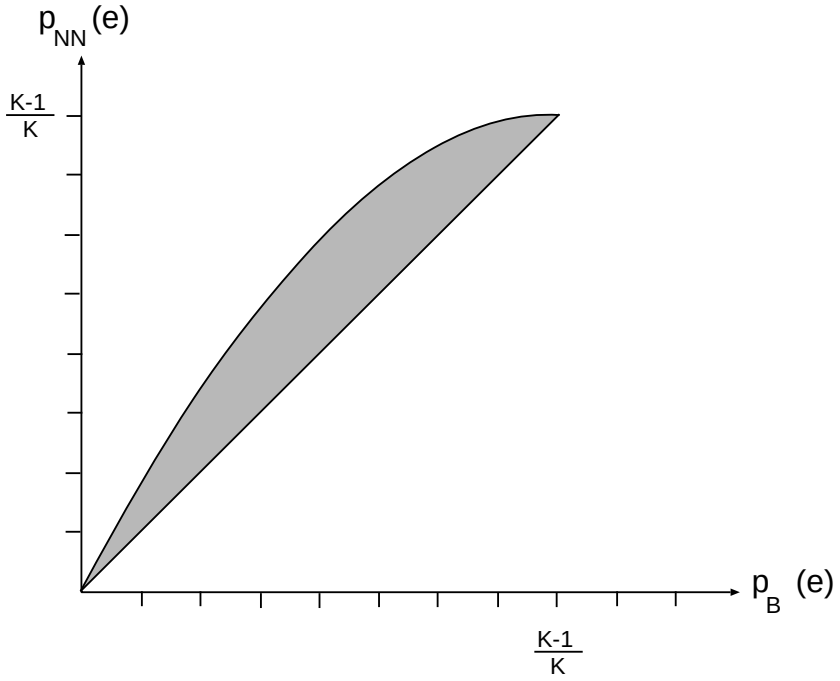
$$p_B(e) \leq p_{NN}(e) \leq 2 \cdot p_B(e) - \frac{K}{K-1} \cdot p_B^2(e) \leq 2 \cdot p_B(e)$$

Exercises:

1. In which case is the upper limit exact?
2. How does the upper limit look like for $p_{NN}(e)$ in case of an arbitrary (instead of the Bayes) decision rule?

Illustration of the inequality for $p_{NN}(e)$:

$$p_B(e) \leq \; p_{NN}(e) \; \leq p_B(e) \cdot \left[2 - \frac{K}{K-1} \cdot p_B(e)\right]$$



The shaded area shows the possible values of $p_{NN}(e)$ for given $p_B(e)$.

5.5 Outlook

The inequality for $p_{NN}(e)$ derived in Chapter 5.4 holds independently of Next Neighbor rule. Actually, $p_{NN}(e)$ is identical to the Gini-criterion which is mentioned in Chapter 4.1. In this way a connection between recognition error and squared error criterion for discriminants is obtained:

Combination of definitions:

$$\begin{aligned} p_B(e|x) &= 1 - \max_k p(k|x) \\ p_{NN}(e|x) &= 1 - \sum_{k=1}^{K} p^2(k|x) \\ f(g;x) &= \sum_{k=1}^{K} p(k|x) \cdot \sum_{c=1}^{K} [g(x,c) - t(k,c)]^2 \\ &= \left[1 - \sum_{k=1}^{K} p^2(k|x)\right] + \sum_{c=1}^{K} [g(x,c) - p(c|x)]^2 \end{aligned}$$

(local squared error criterion on the output of a neural network or discriminant, see 4.1, page 76)
Thus for each point $x \in \mathbb{R}^D$ holds:

$$p_B(e|x) \leq p_{NN}(e|x) \leq f(g;x)$$

and if $g(x,k) \equiv p(k|x)$ for each k:

$$p_{NN}(e|x) = f(g;x).$$

A corresponding inequality/equality holds also for the global expected values:

$$\begin{aligned} p_B(e) &:= \int dx\, p_B(e|x) \cdot p(x) \\ p_{NN}(e) &:= \int dx\, p_{NN}(e|x) \cdot p(x) \\ f(g) &:= \int dx\, f(g;x) \cdot p(x) \end{aligned}$$

6 Mixture Densities and Cluster Analysis

6.1 Mixture Densities

6.1.1 Introduction

So far, two extremes have been introduced:

typical models (Gaussian unimodal distributions)

$\uparrow$

$\downarrow$ $\longleftarrow$ mixture densities

Kernel Densities (model-free)

Compromise:

- multi-modal instead of unimodal
- data reduction in comparison to Kernel Densities
- arbitrarily exact approximation for arbitrary distributions

Standard model: $x \in \mathbb{R}^D$ (class k fixed)

$$p(x) = \sum_{i=1}^{I} c_i \cdot p(x|i, \vartheta_i)$$

(weighted sum of elementary densities)

with weights (*mixture weights*) c_i, for which

$$c_i \geq 0, \quad \sum_i c_i = 1$$

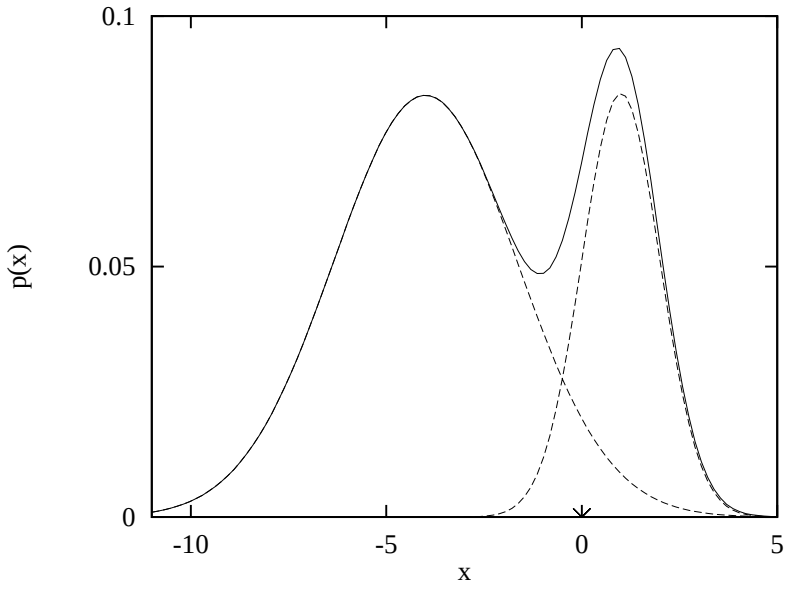
and basis densities, single densities (*component densities*)

$$p(x|i, \vartheta_i).$$

Single densities are typically unimodal, mostly Gaussian:

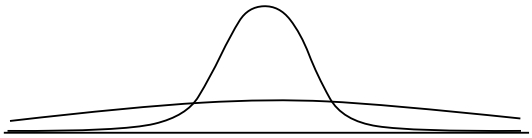
$$\vartheta_i = (\mu_i, \Sigma_i)$$

Example of a bimodal Gaussian distribution (both weighted unimodal distributions are drawn as well):



Similar applications are:

- Runaway-model (Tukey-Model)



Goal: ϵ-probability for non-central area

$$p(x) \;=\; (1-\epsilon)\cdot p_{\text{Gauß}}(x|\mu,\Sigma) + \epsilon \cdot p_{\text{Gauß}}(x|\mu,\alpha\Sigma)$$

$\alpha \gg 1$: augmentation factor for scattering

- Interpolation for language models, e.g. bigram

  $N(v, w)$: Counts of word pair v, w in the training data

  Approach:

$$p(w|v) \;=\; \frac{N(v,w)}{N(v)}$$

  Many pairs are not seen in training, i.e. $N(v, w) = 0$, which means that the corresponding conditional probability is $p(w|v) = 0$.

  Improved approach with interpolation:

$$p(w|v) \;=\; (1-\lambda)\cdot\frac{N(v,w)}{N(v)} \;+\; \lambda\cdot p(w)$$

  with λ : interpolation parameter
  $p(w)$: unigram distribution

  Normalization: $\sum_w p(w|v) \;=\; (1-\lambda)\underbrace{\sum_w \frac{N(v,w)}{N(v)}}_{=1} \;+\; \lambda\underbrace{\sum_w p(w)}_{=1} \;=1$

- Gaussian mixture densities

$$p(x) \;=\; \sum_{i=1}^{I} c_i\,\mathcal{N}(x|\mu_i,\Sigma_i)$$

  Normalization:

$$\int dx\, p(x) = \sum c_i \underbrace{\int dx\, \mathcal{N}(x|\mu_i,\Sigma_i)}_{=1} \;=\; \sum c_i \;=\; 1$$

$$p(x|k) \;=\; \sum_{i=1}^{I} \underbrace{p(i|k)}_{=c_{ik}}\; \mathcal{N}(\mu_{ik},\Sigma_{ik})$$

covariance separately for each density i of each class k

$\downarrow$

$\mathcal{N}(\mu_{ik},\Sigma_k)$ covariance pooled over one class k

$\downarrow$

$\mathcal{N}(\mu_{ik},\Sigma)$ covariance pooled over all classes

6.1.2 EM Algorithm for Mixture Densities

Special case: Maximum Likelihood for Mixture Densities

We combine the unknown parameters c_i and ϑ_i $(i \in I)$ of all I single densities under the notation λ:

$$\lambda \equiv \{c_i, \vartheta_i\}$$

Model: $p_\lambda(x)$ or $p(x|\lambda)$

Training data: $x_1, \ldots, x_n, \ldots, x_N$

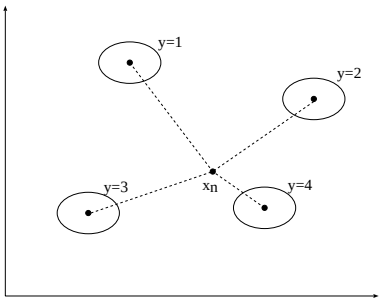
Maximum-Likelihood criterion:

$$\underset{\lambda}{\operatorname{argmax}} \prod_{n=1}^{N} p(x_n|\lambda) \quad \text{or} \quad \underset{\lambda}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(x_n|\lambda)$$

In this way, we obtain an unmanageable optimizing problem. A closed solution for mixture densities and similar complex model distributions does not exist.
The goal of the EM algorithm is to develop

- a simple, consistent iterative method
- without heuristics, and
- without (explicit) gradients.

The EM algorithm is based on the concept of a *hidden variable y* (discrete random variable):



An affiliation between an observation and a value of the hidden variable y is not known/irrelevant. Therefore *all* values of the hidden variable are considered.

Rewrite:

$$\begin{aligned} p(x_n|\lambda) &= \sum_y p(x_n, y|\lambda) \\ &= \sum_y p(y|\lambda) \cdot p(x_n|y, \lambda) \end{aligned}$$

Mixture density: $y \hat{=} i$

$$p(x_n|\lambda) = \sum_i \underbrace{p(i|\lambda)}_{\hat{=} c_i} \cdot p(x_n|i, \vartheta_i)$$

i is not observable

Applications of the EM algorithm:

- Mixture Densities
- smoothing in Language Modeling
- Hidden Markov Models
- IBM translation models
- ...

We consider the difference of log-likelihood functions for two parameter sets λ and $\tilde{\lambda}$:

$$\sum_{n=1}^{N} \log \frac{p(x_n|\tilde{\lambda})}{p(x_n|\lambda)} = \sum_{n=1}^{N} \underbrace{\sum_y p(y|x_n,\lambda)}_{\equiv 1} \cdot \log \frac{p(x_n|\tilde{\lambda})}{p(x_n|\lambda)}$$

$$\text{transform: } p(x_n|\tilde{\lambda}) \cdot p(y|x_n,\tilde{\lambda}) = p(x_n,y|\tilde{\lambda})$$

$$= \sum_{n=1}^{N} \sum_y p(y|x_n,\lambda) \cdot \log \left[\frac{p(x_n,y|\tilde{\lambda})}{p(x_n,y|\lambda)} \cdot \frac{p(y|x_n,\lambda)}{p(y|x_n,\tilde{\lambda})} \right]$$

$$= \sum_{n=1}^{N} \sum_y p(y|x_n,\lambda) \cdot \log \frac{p(x_n,y|\tilde{\lambda})}{p(x_n,y|\lambda)} + \underbrace{\sum_{n=1}^{N} \sum_y p(y|x_n,\lambda) \cdot \log \frac{p(y|x_n,\lambda)}{p(y|x_n,\tilde{\lambda})}}_{(*) \qquad \geq 0 \qquad \forall\, \lambda,\tilde{\lambda}}$$

Define auxiliary function (corresponds to weighted Maximum-Likelihood approach):

$$Q(\lambda,\tilde{\lambda}) := \sum_{n=1}^{N} \sum_y \underbrace{p(y|x_n,\lambda)}_{\text{weight}} \cdot \underbrace{\log p(x_n,y|\tilde{\lambda})}_{\text{LL-function}}$$

$$= \sum_{n=1}^{N} \frac{1}{p(x_n|\lambda)} \sum_y p(y,x_n|\lambda) \cdot \log p(x_n,y|\tilde{\lambda})$$

Then

$$\sum_{n=1}^{N} \log \frac{p(x_n|\tilde{\lambda})}{p(x_n|\lambda)} \geq Q(\lambda,\tilde{\lambda}) - Q(\lambda,\lambda)$$

holds since $(*) \geq 0$.

$(*)$ is the divergence inequality from information theory.

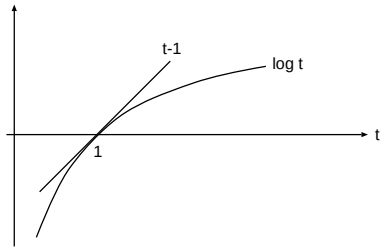
Proof of the divergence inequality:

$$\sum_{n=1}^{N} \sum_{y} p(y|x_n, \lambda) \cdot \log \frac{p(y|x_n, \lambda)}{p(y|x_n, \tilde{\lambda})} \;\geq\; 0$$

It is enough to show:

$$\sum_{y} p(y) \cdot \log \frac{p(y)}{q(y)} \geq 0 \quad \text{where} \quad p(y) > 0,\ \sum_{y} p(y) = 1,$$
$$q(y) > 0,\ \sum_{y} q(y) = 1.$$

For the logarithm:



$$\log t \;\leq\; t - 1, \qquad t > 0$$

Then

$$\begin{aligned} \sum_{y} p(y) \cdot \log \frac{q(y)}{p(y)} \;&\leq\; \sum_{y} p(y) \left[\frac{q(y)}{p(y)} - 1 \right] \\ &=\; \sum_{y} q(y) - \sum_{y} p(y) = 1 - 1 = 0 \end{aligned}$$

and the divergence inequality follows.

EM iterative method:

Initial value λ
Iterations:
1. Expectation (over y): calculate $Q(\lambda, \tilde{\lambda})$
2. Maximization (over $\tilde{\lambda}$): find $\underset{\tilde{\lambda}}{\text{argmax}} \{Q(\lambda, \tilde{\lambda})\}$

next iteration with $\lambda := \tilde{\lambda}$

Application on Mixture Densities:

$$\begin{aligned} p(x_n|\lambda) &= \sum_i p(x_n, i|\lambda) \\ &= \sum_i p(i|\lambda) \cdot p(x_n|i, \lambda) \\ \text{Model} &\hat{=} \sum_i c_i \cdot p(x_n|i, \vartheta_i) \end{aligned}$$

with constraint $\sum_i \tilde{c}_i = 1$.
Abbreviated notation for parameter set: $\lambda \equiv (\{c_i\}, \{\vartheta_i\})$
Using the Lagrange multiplicator for the constraint results in:

$$\begin{aligned} Q(\lambda, \tilde{\lambda}) &= \sum_n \sum_i p(i|x_n, \lambda) \cdot \log\left[\tilde{c}_i \cdot p(x_n|i, \tilde{\vartheta}_i)\right] - \gamma[\sum_i \tilde{c}_i - 1] \\ &= \sum_n \sum_i p(i|x_n, \lambda) \cdot \log \tilde{c}_i \\ &\quad + \sum_n \sum_i p(i|x_n, \lambda) \cdot \log p(x_n|i, \tilde{\vartheta}_i) \; - \; \gamma[\sum_i \tilde{c}_i - 1] \end{aligned}$$

$$\text{where } p(i|x_n, \lambda) = \frac{c_i \cdot p(x_n|i, \vartheta_i)}{\sum_{i'} c_{i'} \cdot p(x_n|i', \vartheta_{i'})}$$

Taking derivatives for determining $\tilde{\lambda}$ as a function of λ:

$$\text{a) } \frac{\partial Q}{\partial \gamma} = \sum_i \tilde{c}_i - 1 \stackrel{!}{=} 0 \quad \Leftrightarrow \quad \sum_i \tilde{c}_i = 1 \text{ (constraint)}$$

b)

$$\frac{\partial Q}{\partial \tilde{c}_i} = \sum_n p(i|x_n, \lambda) \cdot \frac{1}{\tilde{c}_i} - \gamma \stackrel{!}{=} 0$$

Result:

$$\tilde{c}_i = \frac{1}{N} \sum_{n=1}^{N} p(i|x_n, \lambda) = \frac{1}{N} \sum_{n=1}^{N} \frac{c_i \cdot p(x_n|i, \vartheta_i)}{\sum_{i'} c_{i'} \cdot p(x_n|i', \vartheta_{i'})}$$

c)

$$\frac{\partial Q}{\partial \tilde{\vartheta}_i} = \sum_{n=1}^{N} p(i|x_n, \lambda) \cdot \frac{\partial}{\partial \tilde{\vartheta}_i} \log p(x_n|i, \tilde{\vartheta}_i) \stackrel{!}{=} 0$$

in particular Gauß: $p(x_n|i, \mu_i, \Sigma_i)$

with μ_i = average value vector
Σ_i = covariance matrix

$$\begin{aligned} \frac{\partial Q}{\partial \tilde{\mu}_{id}} &= \sum_n p(i|x_n, \lambda) \cdot [\tilde{\mu}_{id} - x_{nd}] \stackrel{!}{=} 0 \\ \Leftrightarrow \quad \tilde{\mu}_{id} &= \frac{\sum_n p(i|x_n, \lambda) \cdot x_{nd}}{\sum_n p(i|x_n, \lambda)} \\ &= \sum_n \frac{p(i|x_n, \lambda)}{\sum_{n'} p(i|x_{n'}, \lambda)} \cdot x_{nd} \end{aligned}$$

Rewriting into vector equation with weights $\gamma_i(n)$ results in

$$\tilde{\mu}_i = \sum_{n=1}^{N} \gamma_i(n) \cdot x_n \quad \text{where} \quad \gamma_i(n) = \frac{p(i|x_n, \lambda)}{\sum_{n'} p(i|x_{n'}, \lambda)}.$$

The weights $\gamma_i(n)$ express the contribution rate of observation x_n for the estimated value $\tilde{\mu}_i$.

Note: $\sum_{n=1}^{N} \gamma_i(n) = 1$ holds due to definition.

Corresponding result for the covariance matrix:

$$\tilde{\Sigma}_i \;=\; \sum_{n=1}^{N} \gamma_i(n) \cdot [x_n - \tilde{\mu}_i][x_n - \tilde{\mu}_i]^T.$$

Interpretation:
Weighted Maximum-Likelihood estimation where the weights are depending on $p(i|x_n, \lambda)$ and are changing during the iterations:

- μ_i : weighted empirical average value vector
- Σ_i : weighted empirical covariance matrix
- c_i : weighted relative frequency

note:

- only local convergence;
- open choice of initial values.

Algorithm (estimation of Gaussian mixture densities):

choose suitable initial values μ_i, Σ_i, c_i

Loop: $$p(i|x_n, \{c_i, \mu_i, \Sigma_i\}) := \frac{c_i \cdot p(x_n|i, \mu_i, \Sigma_i)}{\sum_{i'} c_{i'} \cdot p(x_n|i', \mu_{i'}, \Sigma_{i'})}$$

$$\gamma_i(n) := \frac{p(i|x_n, \{c_i, \mu_i, \Sigma_i\})}{\sum_n p(i|x_n, \{c_i, \mu_i, \Sigma_i\})}$$

$$\tilde{c}_i := \frac{1}{N} \sum_n p(i|x_n, \{c_i, \mu_i, \Sigma_i\})$$

$$\tilde{\mu}_i := \sum_n \gamma_i(n) \cdot x_n$$

$$\tilde{\Sigma}_i := \sum_n \gamma_i(n) \cdot [x_n - \tilde{\mu}_i][x_n - \tilde{\mu}_i]^T$$

update: $c_i = \tilde{c}_i \qquad \mu_i = \tilde{\mu}_i \qquad \Sigma_i = \tilde{\Sigma}_i$

GOTO Loop

Exercises:

1. Write a C-program for the example above.
2. Calculate formulae for the EM algorithm for the Gaussian distribution with *tied covariance matrix*, i.e. $p(x_n|i, \mu_i, \Sigma)$ with Σ independent of i.

6.1.3 Mixture Densities for K classes

Different cases:

a) no coupling between classes:

for each class k: weight $c_{ik} = p(i|k)$, $\sum_{i=1}^{I_k} p(i|k) = 1$

parameters of conditioned distributions ϑ_{ik}

Model: $p(x|k) = \sum_{i=1}^{I_k} c_{ik} \cdot p(x|i, k, \vartheta_{ik})$

Training can be done separately for each class k.

b) *Tied mixture densities*:
Coupling between models $p(x|k)$ through class-independent parameters ϑ_i.

Model: $p(x|k) = \sum_{i=1}^{I} c_{ik} \cdot p(x|i, \vartheta_i)$

i.e. single densities $p(x|i, \vartheta_i)$ act as *basis distributions*.

Exercise:

Calculate the EM-iteration formula for the model above i.e. *tied mixture densities* for Gaussian distributions.

Approach:

$$Q(\lambda, \tilde{\lambda}) = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \sum_{i=1}^{I} p(i|x_{nk}, k, \lambda) \cdot \log\left[\tilde{c}_{ik} \cdot p(x_{nk}|i, \tilde{\vartheta}_i)\right]$$

$$\text{with} \quad p(i|x_{nk}, k, \lambda) = \frac{c_{ik} \cdot p(x_{nk}|i, \vartheta_i)}{\sum_{i'=1}^{I} c_{i'k} \cdot p(x_{nk}|i', \vartheta_{i'})}$$

and training vectors $x_{1k}, \ldots, x_{nk}, \ldots, x_{N_k k}$ for class k.

6.1.4 Maximum Approximation

(without class index k)

Replace the sum with maximum:

$$p(x) \;=\; \max_i \,[c_i \cdot \underbrace{p(x|i,\mu_i,\Sigma_i)}_{\text{Gaussian distribution}}\,].$$

Justification: the Gaussian distribution decreases exponentially such that one sum element is expectedly dominating.

Training method:

choose suitable initial values μ_i, Σ_i, c_i

Loop: determine the *best* single density for each observation x_n

$$\operatorname*{argmax}_i [c_i \cdot p(x_n|i,\mu_i,\Sigma_i)]$$

Observation x_n contributes only to the training of one single density, namely the best one.

Parameter estimation:

c_i = relative frequency of choosing i as best single density

μ_i = average value for each single density i of assigned observations x_n

Σ_i = covariance matrix for each single density i of assigned observations x_n

GOTO Loop

Interpretation: maximum approximation results from "specialization":

$$p(i|x_n,\{c_i,\mu_i,\Sigma_i\}) = \begin{cases} 1, & i = \textit{best} \text{ single density}, \\ 0, & \text{otherwise.} \end{cases}$$

6.2 Cluster Analysis

6.2.1 Overview

Cluster = aggregation, grouping (of observations, usually points in $\mathbb{R}^D$)

Goal: find (and interpret) clusters (sub-groups) for a given set of observations.

In many cases, there is no clear optimization criterion for this task and it is not possible to define the best optimization strategy. The approach developed in Chapter 6.1 can (partly) avoid these difficulties in the way that the task is defined as a clear statistical model.

In literature about cluster analysis, a lot of different methods are proposed which can be arranged according to different criteria:

- distance measures (similarity measures) $\leftrightarrow$ statistical methods (mixture densities), fuzzy clustering
- discrete $\leftrightarrow$ continuous observations
- hierarchical $\leftrightarrow$ non-hierarchical
- fully automatic $\leftrightarrow$ interactive

Combinatorial problem:

- For a given set of N vectors there is an exponential number of partitions (pairwise different).

  Vectors $x_1, \ldots, x_N$ with label 0 or 1, i.e. 2 clusters

  $\Longrightarrow \dfrac{2^N - 2}{2} = 2^{N-1} - 1$ partitions
- in principle, all partitions have to be considered in order to choose the best (e.g. Maximum-Likelihood criterion).

$S(N, I)$ is the number of partitions for I clusters and N (pairwise different) vectors.

$$S(N, I) \;=\; \frac{1}{I!} \cdot \sum_{i=1}^{I} \binom{I}{i} (-1)^{I-i} \cdot i^N \cong \frac{1}{I!} I^N \;.$$

Note: Stirling numbers of the second kind

Recursion:

$$\begin{aligned} S(N+1, I) \;&= S(N, I-1) && x_{N+1} \text{ defines new } I\text{-th cluster} \\ &+\; I \cdot S(N, I) && x_{N+1} \text{ will be inserted in each} \\ & && \text{of the } I \text{ old clusters} \end{aligned}$$

Examples:

- $I = 2 \Longrightarrow S(N, 2) = \dfrac{2^N - 2}{2} = 2^{N-1} - 1$
- $N = 5, I = 3 \Longrightarrow S(5, 3) = 25$

  decompositions: $\{a, b, c\} \cup \{d\} \cup \{e\}$ $\quad \binom{5}{3} = 10$

  $\{a, b\} \cup \{c, d\} \cup \{e\}$ $\quad 5 \cdot 3 = 15$

In practice:

- heuristic methods;
- iterative methods: improving an initial partition step-by-step until a local optimum is achieved (see EM algorithm).

6.2.2 Squared Error Criterion and Exchange Method

(often called variance criterion; minimum-distance method)

Names of iteration methods for optimizing the squared error criterion:

Nearest Mean, K-Means, Iso-data
(here: $K = I \hat{=}$ number of clusters)

Assumption: everything is reduced to one (Euclidean) distance measure (after suitably scaling the axes).

If A_i are clusters where $i = 1, \ldots, I$, then

$$\begin{aligned} G_{TOT} &:= \sum_{i=1}^{I} G_i \\ &\quad \text{with } G_i := \sum_{x \in A_i} ||x - \mu_i||^2 \\ &\quad \text{and } \mu_i := \frac{1}{N_i} \sum_{x \in A_i} x \\ N_i &:= |A_i| \quad \text{number of elements in } A_i \end{aligned}$$

Exchange operation: move x from cluster A_i into cluster A_j.

How are the centers μ_i and μ_j and scores G_i and G_j changed after an exchange operation? (calculate for yourself, s.a. [DH73], p. 227):

$$\begin{aligned} \mu_j^* &= \mu_j + \frac{x - \mu_j}{N_j + 1} \\ \mu_i^* &= \mu_i - \frac{x - \mu_i}{N_i - 1} \\ G_j^* &= G_j + \frac{N_j}{N_j + 1} ||x - \mu_j||^2 \\ G_i^* &= G_i - \frac{N_i}{N_i - 1} ||x - \mu_i||^2 \end{aligned}$$

Variant: introduction of cluster-dependent variances.

Algorithms:

NEAREST MEAN (*sequential, adaptive, on-line*)

choose initial partition $A_1, \ldots, A_I$ and calculate G_i and μ_i

Loop: choose next observation x; let $x \in A_i$
calculate G_j^* for all clusters A_j and move x to the best cluster

update: μ_j, G_j, μ_i, G_i

if (G_{TOT} does not change in N passes) STOP
otherwise GOTO Loop

NEAREST MEAN (*batch, off-line*)

choose initial partition $A_1, \ldots, A_I$ and calculate G_i and μ_i

Loop: for each observation x_n where $n = 1, \ldots, N$ determine the nearest cluster A_i and memorize the cluster index i

for each cluster A_i where $i = 1, \ldots, I$ calculate new average value μ_i from the observations x_n assigned to this cluster

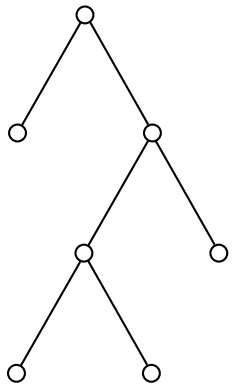
if (no change) STOP
otherwise GOTO Loop

Notes:

1. it is clear that both algorithms can be transferred to other "average value" models.
2. empirical experience: batch-variant is less susceptible to bad local minima.
3. still open: how to choose initial partition?
   $\longrightarrow$ hierarchical clusters

6.2.3 Hierarchical Cluster Analysis

Hierarchy is modeled with binary tree:



Criteria:

- distance measure or likelihood
- structuring/refinement of the tree

Two variants:

1. Top-down, divisive method (*splitting*)
2. Bottom-up, agglomerative method (*merging*)

Top-down method: basic concept

Start: all data $x_1, \ldots, x_N$ build one cluster

Loop: choose the cluster with the worst score,
split this cluster into two new clusters

improve parameters of all clusters by iterative training (NEAREST-MEAN-STYLE) until stop-criterion is achieved

check the quality of total partition

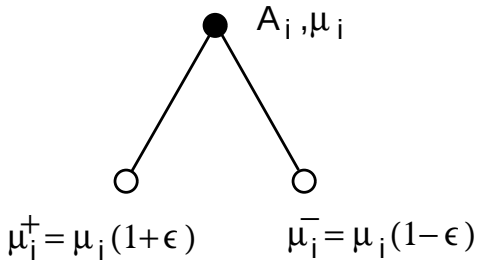
if "good" STOP,

otherwise GOTO Loop

Problem:
By splitting a cluster which contains m elements there are $2^{m-1} - 1$ possibilities to build 2 clusters. Due to computing time issues it is usually not possible to evaluate all of them.

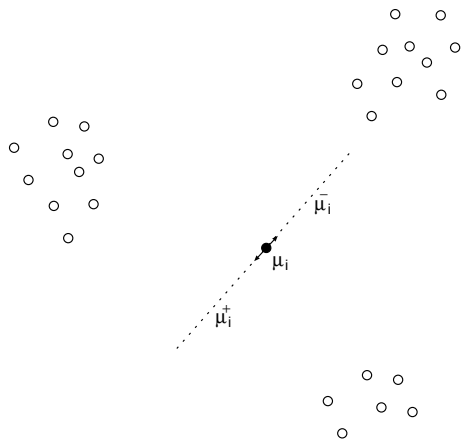
Heuristic for the "distance method":

1. *infinitesimal* perturbation of the average value μ_i



2. iterative training to improve μ_i^- and μ_i^+

   Example:



   Perturbation is normally performed in standard direction because the direction is not critical.

   Speech processing: Linde, Buzo, Gray [LBG80] "LBG algorithm"

Note:

- Extension to mixture densities as an extension of distance models is obvious
- it is suitable for vector spaces, equivalent for language modeling is not known.

Bottom-up method (agglomerative hierarchical clustering)

Start: each observation is its own cluster

Loop: determine the cluster pair with minimal distance $(*)$

merge the two clusters

if (number of clusters > 1) GOTO Loop

else STOP

Complexity:

- For m clusters, $\frac{m(m-1)}{2}$ steps are necessary to find the optimal pair.
- sum over all $m = N, N-1, \ldots, 1$

$$\sum_{m=1}^{N} \frac{m(m-1)}{2} \cong \frac{1}{6} N^3$$

cubic complexity $\Longrightarrow$ from $N \cong 10000$, computational time is critical.

improved variant $(*)$:

Loop: find the cluster pair whose merging produces minimal deterioration of the global criterion.

Standard methods:

- [Gor81], p.46
- [Spa77], p.170
- [DH73], p.230

Distance measures between clusters:

given: cluster A_i with vectors x_i and cluster A_j with vectors x_j,
$||.||$ Euclidean distance (not squared)

- single link

$$d(A_i, A_j) := \min_{i,j} ||x_i - x_j||$$

- complete link

$$d(A_i, A_j) := \max_{i,j} ||x_i - x_j||$$

- group average link

$$d(A_i, A_j) := \frac{1}{N_i \cdot N_j} \sum_i \sum_j ||x_i - x_j||$$

- weighted average link

$$d(A_i, A_j) := \sum_i \sum_j ||x_i - x_j||$$

- centroid link

$$d(A_i, A_j) := ||\mu_i - \mu_j||$$

- median link Medians $\tilde{\mu}_i, \tilde{\mu}_j$

$$d(A_i, A_j) := ||\tilde{\mu}_i - \tilde{\mu}_j||$$

- sum of squares (Ward algorithm)

$$d(A_i, A_j) := \frac{N_i \cdot N_j}{N_i + N_j} ||\mu_i - \mu_j||^2$$

(consistent with Gaussian model)

Exercise:

Specify Gaussian model
(but: in the first step, each vector represents its own cluster i.e. it is not possible to calculate variances.
therefore: pooled covariance matrix in the first step)

Note:
Efficient reformulations can be found in the literature.

Hints:

- preferably avoid (often in literature): "reading tea leaves"
- preferably define objective criteria for new test data:
  - likelihood
  - error rate (for classification tasks)

Iterative optimal algorithm for hierarchical agglomerative clustering

Start: each observation is one cluster

Loop: find the cluster pair whose merging produces minimal deterioration of global criterion
merge those clusters

if (number of clusters > 1) GOTO Loop

Notes:

1. Computational effort increases scarcely (factor 2 at most) in comparison to standard methods.
2. "optimum" is achieved with each merging.
3. The Ward algorithm ("sum of squares") already has this optimality property (see [DH73], p.260, Exercise(Lesson) 19).

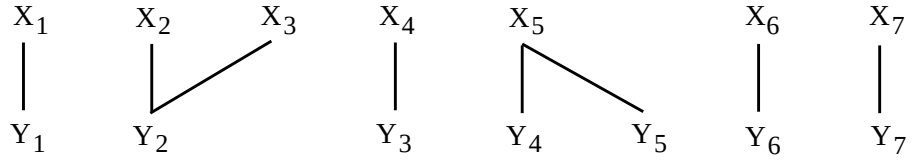
7 Stochastic Finite Automata

7.1 Motivation and Model

We consider a time sequence of observations:

$$x_1, \ldots, x_t, \ldots, x_T, \qquad x_t \in \mathbb{R}^D, \quad t = \text{time} .$$

The mapping of corresponding events (e.g. phonemes in speech) is difficult without a fixed time basis:



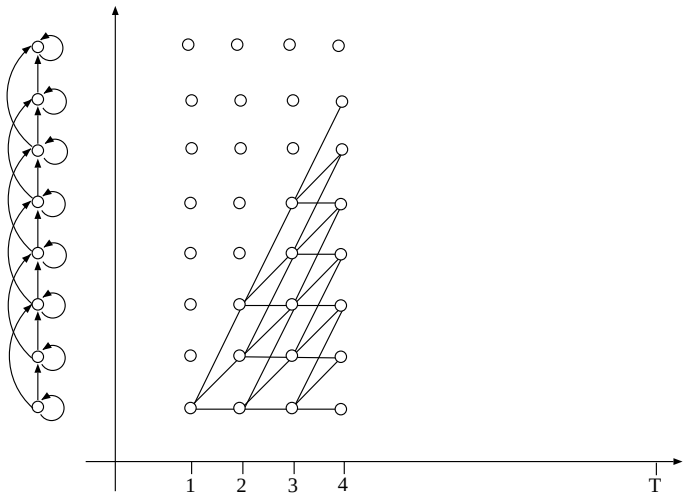
In other words:
The sequence as a whole (practically) cannot be described as an element of a suitable vector space because a mapping of observations x_t to corresponding points on coordinate axes is difficult.

Linear scaling and normalization of the time axis is helpful but not sufficient.

Analogy:

- Formal grammars are introduced in order to compactly describe the chains of discrete symbols $\Rightarrow$ extension of probability statements.
- Stochastic finite automata are introduced in order to compactly describe continuous vectors (or discrete *noisy* symbols).
- *Syntactic Pattern Recognition* (probability scores)

Stochastic finite automaton
= stochastic regular grammar
= Hidden Markov Model (HMM)



- Also: *lattice / trellis*
- Path corresponds to unfolding of a regular grammar
- Recombination of different paths in one point

Estimating the number of possible paths for the standard $(0, 1, 2)$-model in the figure above which is used in speech recognition:

> For a sufficiently long word model, i.e. $S \geq \frac{T}{2}$, there are 3^{T-1} possible paths for T observation vectors $x_1, \ldots, x_t, \ldots x_T$ if all $2T-1$ reachable end states are allowed. But actually only one end state is allowed. If we assume the simplification that all end states can be reached through an equal number of different paths (raw approximation!), we obtain approximately $3^{T-1}/(2T-1)$ possible paths for one allowed end state.

abstract (=hidden) states: $s, \sigma \in \{1, \ldots, S\}$

HMM specification:

- Transitions: specifying pairs $\sigma \longrightarrow s$ and their probabilities $p(s|\sigma)$ with constraint:

$$\sum_s p(s|\sigma) = 1, \qquad \forall \sigma.$$

  $(0, 1, 2)$-model:

$$\sum_{i=0}^{2} p(\sigma + i|\sigma) \; = \; 1$$

- Emissions ($\hat{=}$ observations x_t) during transition $\sigma \longrightarrow s$:

  $p(x_t|\sigma, s) =$ "usual model" (e.g. Gauss, mixture densities, ...).

  Note the normalization: $\int p(x_t|\sigma, s) \; = \; 1$

- Specifying start and end state.

Examples:

1. Speech:

   x_t = energy distribution (over frequency axis), calculated every 10 ms using Fourier-analysis.

   HMMs extremely important:

   - extreme non-linear variations of the speaking rate;
   - avoids local decisions (e.g. about the identity of one phoneme).

   similar: signal flow over time-axis
   (e.g. gradients, EKG, EEG, ...)

2. Character recognition:

   Approach:

   (a) time axis $\hat{=}$ horizontal axis

   (b) decomposition of the image using window function

   (c) feature vector x_t for each window

3. Processing "erroneous" symbol chains

   - *Levenshtein* distance: deletions, insertions, substitutions
   - special: written language

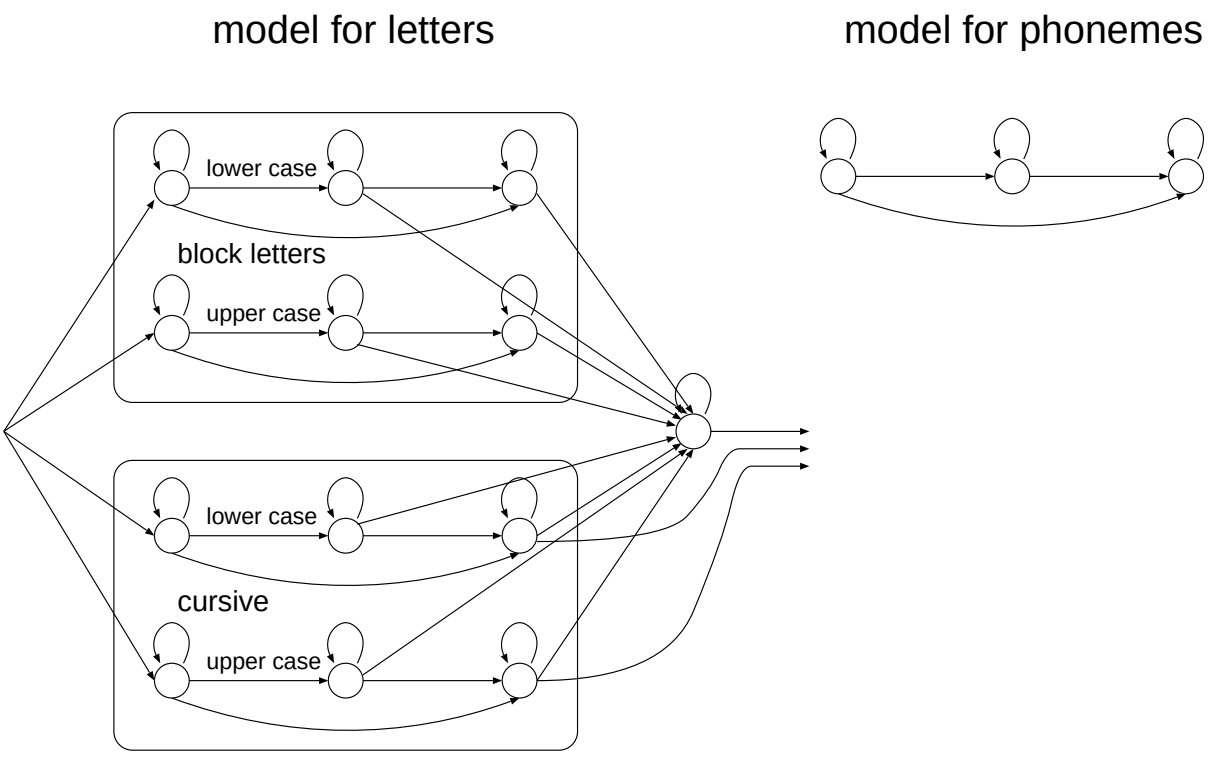


Figure: Hidden Markov Models for character and speech recognition.

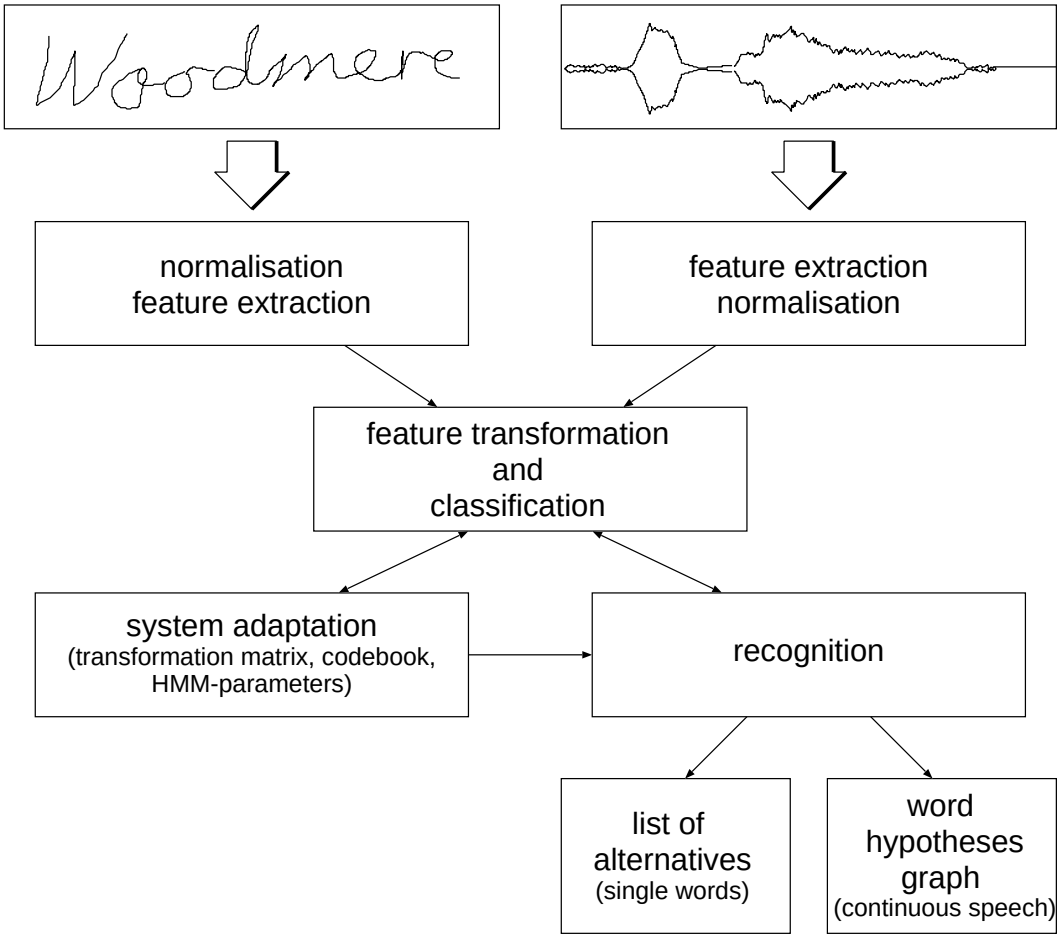


Figure: Block-diagram of a recognition system with feature extraction, feature transformation and classification, training of HMM parameters and Viterbi recognition.

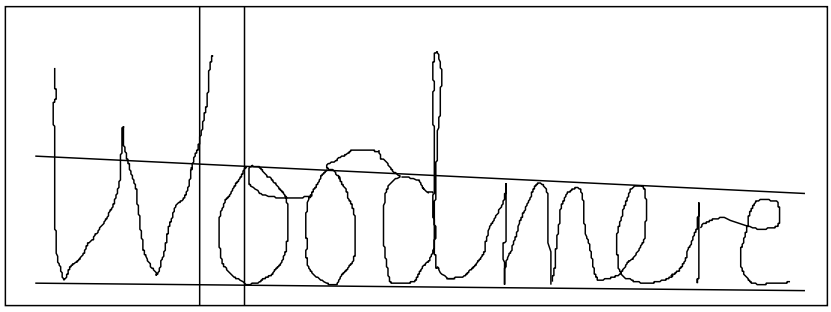


Figure: Address word after preprocessing and normalization.

7.2 Mathematical Formalism

(for Hidden Markov Models in general)

In practical systems

x_1^T : time sequence of vectors $x_1, \ldots, x_T$

s_1^T : time sequence of states $s_1, \ldots, s_T$

$p(x_1^T)$: probability of whole sequence. Strictly speaking, this should be considered separately for each class $k = 1, \ldots, K$, i.e. p has the form $p(x_1^T|k)$. In the following, we will use only the shortened form.

$$p(x_1^T) = \sum_{[s_1^T]} p(x_1^T, s_1^T) \qquad \text{(introducing states)} \quad (*)$$

$$p(x_1^T, s_1^T) = \prod_{t=1}^{T} p(x_t, s_t | x_1^{t-1}, s_1^{t-1})$$

Model assumptions:

- Dependency only on states (*hidden*):

$$\boxed{p(x_t, s_t | x_1^{t-1}, s_1^{t-1}) = p(x_t, s_t | s_1^{t-1})}$$

- Dependency only on the first previous state (*first-order Markov*):

$$\boxed{p(x_t, s_t | s_1^{t-1}) = p(x_t, s_t | s_{t-1}) = p(s_t | s_{t-1}) \cdot p(x_t | s_{t-1}, s_t)}$$

($\hat{=}$ Transition Probability $\cdot$ Emission Probability (density))

Inserting in $(*)$ results in:

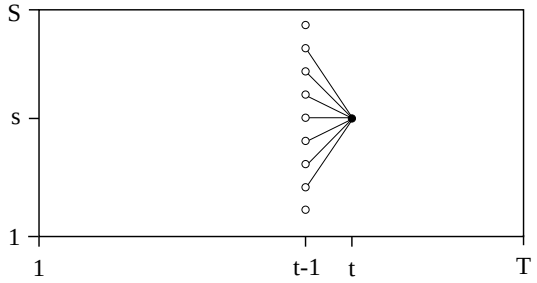
$$\begin{aligned} p(x_1^T) &= \sum_{[s_1^T]} \prod_{t=1}^{T} [p(s_t|s_{t-1}) \cdot p(x_t|s_{t-1}, s_t)] \\ &= \sum_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}) \end{aligned}$$

A direct calculation of the sum in the previous equation is time-consuming because we have to sum over $3^{T-1}/(2T-1)$ (for the standard $(0,1,2)$-model) or S^{T-1} (in general) paths. However, there is one simple and efficient method which we will derive in the following (see also [Ney99a]):

7.2.1 Baum Recursion

Define auxiliary function $Q(t,s)$ as "sum over all paths passing through (t,s)".

$$Q(t,s) \;:=\; \sum_{s_1^t : s_t = s} \prod_{\tau=1}^{t} p(x_\tau, s_\tau | s_{\tau-1})$$



Decomposition: $[\cdots \longrightarrow (s,t)] = [\cdots \longrightarrow (\sigma, t-1)]\ [(\sigma, t-1) \longrightarrow (s,t)]$

$$Q(t,s) \;=\; \sum_{\sigma} \Big[\underbrace{\sum_{s_1^{t-1} : s_{t-1} = \sigma} \prod_{\tau=1}^{t-1} p(x_\tau, s_\tau | s_{\tau-1})}_{=Q(t-1,\sigma)} \cdot p(x_t, s | \sigma) \Big]$$

$$\begin{aligned} Q(t,s) &= \sum_{\sigma} p(x_t, s|\sigma) \cdot Q(t-1,\sigma) \\ p(x_1^T) &= Q(T,S) \end{aligned}$$

Recursion equation of type *divide and conquer* (*dynamic programming*)

The computational time is proportional to $T \cdot S^2$ (instead of $3^{T-1}/(2 \cdot (T-1)) \cong$ as the number of paths in the standard $(0,1,2)$-model or S^{T-1} as the number of paths in general).

This equation is (for historical reasons) called *Baum Recursion.*

- Variants: *Baum-Welch Recursion*, *Forward-Backward Algorithm* (here, only the *Forward*-direction is considered)

7.2.2 Viterbi Algorithm

(also: Maximum Approximation)
This approximation means restriction to the *best* path through (t, s):

$$p(x_1^T) \;\cong\; \max_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}) \; .$$

This is a good approximation of the sum if one path is "dominating".

Analogously to the Baum(-Welch) recursion, define $Q(t,s)$ as:

$$Q(t,s) := \max_{s_1^t : s_t = s} \prod_{\tau=1}^{t} p(x_\tau, s_\tau | s_{\tau-1}) \; .$$

The decomposition results in:

$$Q(t,s) = \max_{\sigma} \left[p(x_t, s|\sigma) \cdot \underbrace{\max_{s_1^{t-1} : s_{t-1} = \sigma} \prod_{\tau=1}^{t-1} p(x_\tau, s_\tau | s_{\tau-1})}_{= Q(\tau-1, \sigma)} \right]$$

$$\boxed{Q(t,s) = \max_{\sigma} \left[p(x_t, s|\sigma) \cdot Q(t-1, \sigma) \right]}$$

Recursive equation of dynamic programming

As in the previous case, the computational time is proportional to $T \cdot S^2$ (instead of $3^{T-1}/(2 \cdot (T-1))$).

Local decisions are stored for $t = 1, \ldots, T$ and for each state s. After reaching the end-time point ($t = T$), the best path is traced back.

Analogy to *formal languages*:

Recognizer	$\hat{=}$	Baum algorithm	:	all parse trees
Parser	$\hat{=}$	Viterbi algorithm	:	best parse tree

7.3 Incorporating into Bayes Decision Rule

Each class k has its own automaton with end-state $S(k)$.
$Q_k(T, S(k))$ is calculated for a vector sequence x_1^T using the recursive equation (either exact or DP):

$$Q_k(T, S(k)) \qquad \text{for each class (automaton) } k = 1, \ldots, K$$

Therefore, the following holds per definition:

$$p(x_1^T|k) \;=\; Q_k(T, S(k)),$$

and the decision rule is:

$$\begin{aligned} x_1^T \longrightarrow r(x_1^T) \;&=\; \operatorname*{argmax}_k \{p(k) \cdot p(x_1^T|k)\} \\ &=\; \operatorname*{argmax}_k \{p(k) \cdot Q_k(T, S(k))\} \end{aligned}$$

7.4 Maximum-Likelihood Training

Same approach as for mixture densities, i.e. Baum algorithm, EM algorithm

Vector sequence: x_1^T (only one observation; for more: summation over n)

$y = s_1^T =$ *hidden variable*

$$\begin{aligned} Q(\lambda, \tilde{\lambda}) &= \sum_{[s_1^T]} p(s_1^T | x_1^T, \lambda) \cdot \log p(x_1^T, s_1^T | \tilde{\lambda}) \\ &= \frac{1}{p(x_1^T|\lambda)} \cdot \sum_{[s_1^T]} p(x_1^T, s_1^T | \lambda) \cdot \log p(x_1^T, s_1^T | \tilde{\lambda}) \end{aligned}$$

- Model: $p(x_1^T, s_1^T | \tilde{\lambda}) = \prod_{t=1}^{T} \left[p(s_t | s_{t-1}, \tilde{\lambda}) \cdot p(x_t | s_{t-1}, s_t, \tilde{\lambda}) \right]$
- Parameter λ: $\alpha(s|\sigma)$, $\sum_s \alpha(s|\sigma) = 1$, $\forall \sigma$ instead of $p(s|\sigma)$.

$p(x|\sigma, s; \vartheta) = p(x|s; \vartheta_s)$ e.g. Gaussian or mixture density with parameters ϑ_s

Parameter set: $\lambda = \{\alpha(s|\sigma), \vartheta_s\}$

Regrouping of terms results in:

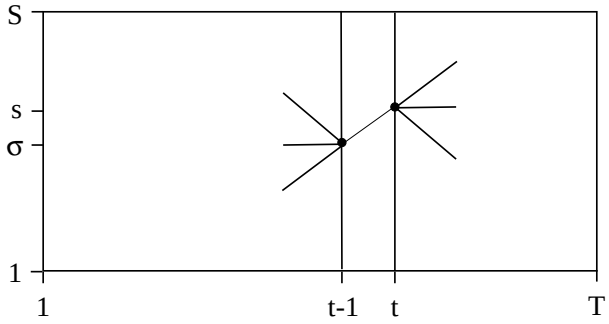
$$\begin{aligned} Q(\lambda, \tilde{\lambda}) = \frac{1}{p(x_1^T|\lambda)} \cdot \Bigg[& \sum_{s,\sigma} \sum_t p(x_1^T, s_{t-1} = \sigma, s_t = s | \lambda) \cdot \log \tilde{\alpha}(s|\sigma) \\ & + \sum_s \sum_t p(x_1^T, s_t = s | \lambda) \cdot \log p(x_t | s, \tilde{\vartheta}_s) \Bigg] \end{aligned}$$

Note:
An initial distribution $\pi(s)$ which is often found in literature is omitted here; it can be easily modeled by insertion of a fictitious state $s = 0$ and probability $\pi(s) := p(s|0)$.

$$p(x_1^T|\lambda) = \sum_{[s_1^T]} p(x_1^T, s_1^T|\lambda)$$

$$p(x_1^T, s_{t-1}=\sigma, s_t=s|\lambda) := \sum_{\substack{[s_1^T];\\ s_{t-1}=\sigma; s_t=s}} p(x_1^T, s_1^T|\lambda)$$

"Paths going through $(t-1,\sigma)$ and (t,s)"



Analogous definition for $p(x_1^T, s_t - 1 = s|\lambda)$.
Defining the "posterior" auxiliary functions:

$$\gamma_t(\sigma,s) = p(s_{t-1}=\sigma, s_t=s|x_1^T,\lambda) = \frac{p(x_1^T, s_{t-1}=\sigma, s_t=s|\lambda)}{p(x_1^T|\lambda)}$$

$$= \frac{\sum\limits_{\substack{[s_1^T];\\ s_{t-1}=\sigma; s_t=s}} p(x_1^T, s_1^T|\lambda)}{\sum\limits_{s_1^T} p(x_1^T, s_1^T|\lambda)}$$

$$\gamma_t(s) = p(s_t=s|x_1^T,\lambda) = \frac{p(x_1^T, s_t=s|\lambda)}{p(x_1^T|\lambda)}$$

Model: Gaussian distribution $\vartheta_s = [\mu_s, \Sigma_s]$ for each state s

Iteration formulae (reestimation):

$$\tilde{\alpha}(s|\sigma) = \frac{\sum_t \gamma_t(\sigma, s)}{\sum_t \sum_{s'} \gamma_t(\sigma, s')} = \frac{\sum_t \gamma_t(\sigma, s)}{\sum_t \gamma_t(\sigma)}$$

$$\Rightarrow \quad Q(\lambda, \tilde{\lambda}) = \sum_{s,\sigma} \sum_t \gamma_t(\sigma, s) \cdot \log \tilde{\alpha}(s|\sigma) + \sum_{s,\sigma} \sum_t \gamma_t(s) \cdot \log p(x_t|s, \tilde{\theta}_s)$$

$$p(x_t|s, \tilde{\theta}_s) \stackrel{!}{=} \mathcal{N}(x_t|\tilde{\mu}_s, \tilde{\Sigma}_s)$$

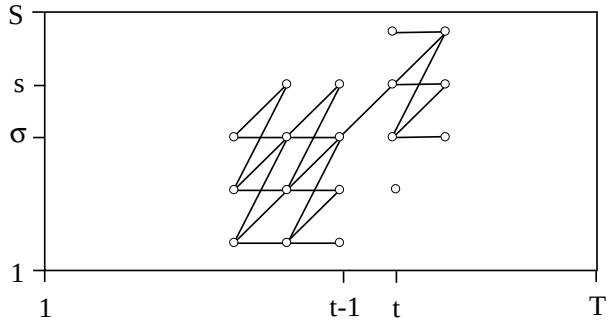
$$\Rightarrow \quad \tilde{\mu}_s = \frac{\sum_t \gamma_t(s) \cdot x_t}{\sum_t \gamma_t(s)}$$

$$\tilde{\Sigma}_s = \frac{\sum_t \gamma_t(s) \cdot [x_t - \tilde{\mu}_s] \cdot [x_t - \tilde{\mu}_s]^T}{\sum_t \gamma_t(s)}$$

Note:

1. Mixture densities: analogous result, but more complicated proof of the EM method (exercise)
2. Necessary: extension to many vector sequences
   $\longrightarrow$ "analogous" iteration formulae for one long vector sequence

Explanations about training:



$\gamma_t(\sigma, s)$ "forces" paths through σ and s in order to isolate their contribution ($\gamma_t(s)$ analogously).

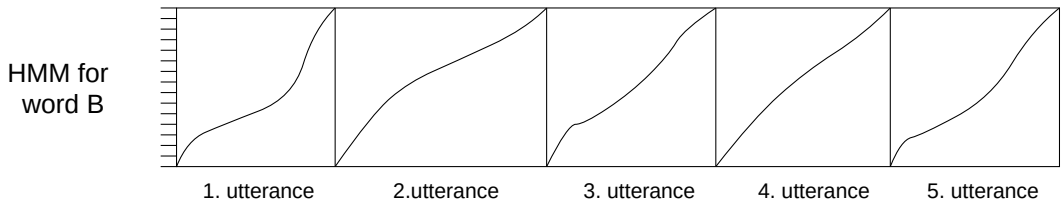
The computation of $\gamma_t(\sigma, s)$ and $\gamma_t(s)$ can be efficiently determined by combining the forward and backward variants of the Baum recursion, i.e. the *Forward-Backward algorithm*.
The principle is simple, but an implementation has to be done using tricks due to CPU and memory requirements as well as possible overflow.

Instead of that, we will consider the *Maximum Approximation* (also called *Viterbi training*), which uses only the best path:

$$\gamma_t(\sigma, s) = \begin{cases} 1, & \text{best path through } \sigma, s \\ 0, & \text{otherwise} \end{cases}$$

$$\gamma_t(s) = \begin{cases} 1, & \text{best path through } s \\ 0, & \text{otherwise} \end{cases}$$

Training using Maximum Approximation: (see [Ney99a])
Speech training samples: utterances of a word B



Iterative training in two steps:

1. Time alignment: estimating the best path
2. Estimation of the model parameters:
   (a) Collecting observations for each state and each transition
   (b) Maximum-Likelihood estimation, e.g.
   Gauss: $\hat{\mu}$ = empirical average value
   $\hat{\Sigma}$ = empirical covariance

   transition probability = relative frequency

Note:

1. This is a further example of *decision-directed learning* (as for cluster analysis).
2. Extension to word chains instead of single words is possible in a simple way.

7.5 Stochastic Grammars (overview)

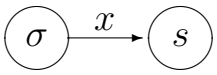
Goal: transferring the HMM concept to general context-free grammars.

Notation:

$A, B, C, \ldots, Z$	non-terminals
$x_1, \ldots, x_t, \ldots, x_T$	terminals (observations, discrete or $\in \mathbb{R}^D$)

- *Stochastic regular grammar*:

  The probability that the terminal x is observed during a transition from state σ into state s

  is denoted as $p(x, s|\sigma)$ $(= p(s|\sigma) \cdot p(x|\sigma, s))$.

  This corresponds to a production rule of type $s \longrightarrow \sigma x$ or $\sigma \longrightarrow xs$ (according to interpretation).

  The whole regular grammar can be built from this type of rule (except start-states).

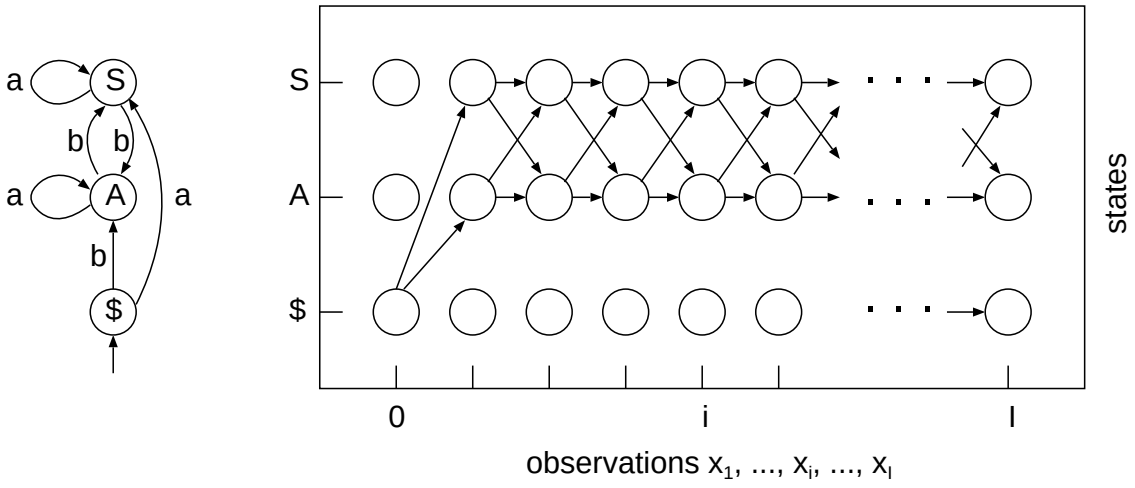
- *Stochastic context-free grammar* (Chomsky Normal Form):

structure	$A \longrightarrow BC$	$p(BC\|A)$
observation	$A \longrightarrow x$	$p(x\|A)$

$p(BC|A)$ corresponds to the probability of applying rule $A \longrightarrow BC$ if non-terminal A is given.

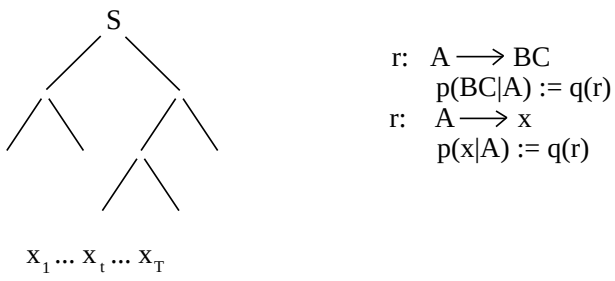
Note:

For significantly distinctive structures only a small number of probabilities $p(BC|A) > 0$.



Consider the chain of rules $r_1 \dots r_M$ (in Chomsky Normal Form: $M = 2T - 1$) :

$$S \xrightarrow{r_1} \cdots \xrightarrow{r_m} \cdots \xrightarrow{r_M} [x_1 \dots x_t \dots x_T]$$



Model: $q(r_m|r_{m-1}) = q(r_m)$ "arbitrary order"

$$\boxed{p(x_1^T, r_1^M) = \prod_{m=1}^{M} q(r_m)}$$

$$\begin{aligned} p(x_1^T) &= \sum_{[r_1^M]} p(x_1^T, r_1^M) && \text{"all derivations" (recognizer)} \\ &\cong \max_{[r_1^M]} p(x_1^T, r_1^M) && \text{"best derivations" (parser)} \end{aligned}$$

Notes:

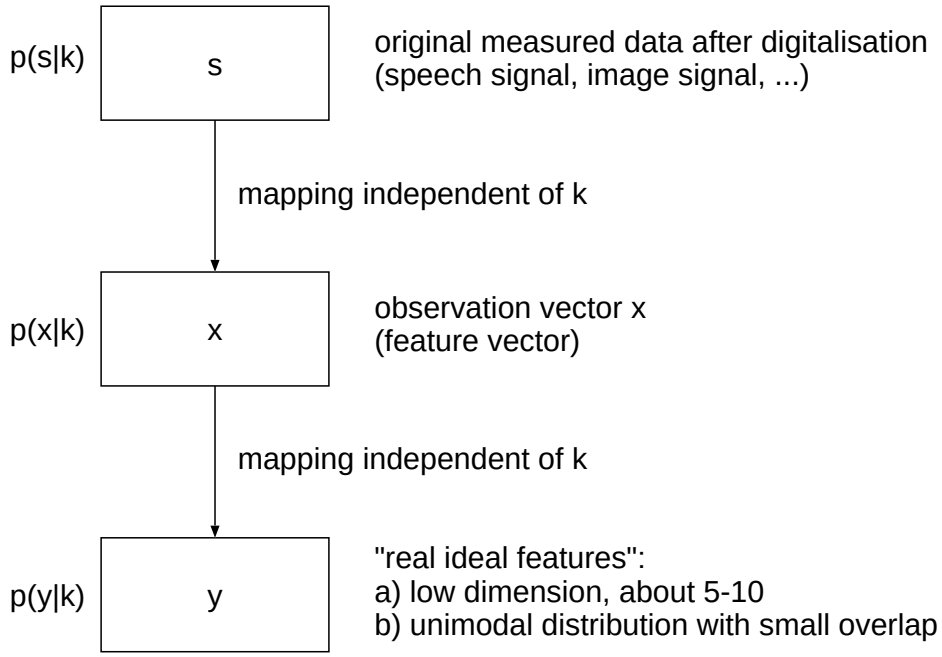
1. Efficient algorithms based on dynamic programming (or divide and conquer) exist for both variants $(\sum, \max)$.
2. In this way, a stochastic variant of Cocke-Younger-Kasami or Earley algorithm is obtained (Bottom-Up or Top-Down Parser for context-free grammars).
3. Training:
   - extensions to the reestimation equations exist
   - so far not much experience with real problems
   - open and actual problem: learning the rules (*grammatical inference*)
4. Hierarchical structure is possible:
   - word level: combining word observations
   - sentence level: combining words into sentences

   (Example: introducing parity checking in digit-sequence recognition)

8 Feature Extraction and Linear Mapping

8.1 Ideal Feature Extraction

Different data types (k = class index):



Statements:

1. For *completely and exactly* known distributions

   $$p(s|k),\ p(x|k),\ p(y|k)$$

   and *invertible* transformations $s \longleftrightarrow x \longleftrightarrow y$ there is no difference in the error rate when the Bayes decision rule is used, at most in computational time (original data s are always extremely extensive).

   Mappings $s \longrightarrow x \longrightarrow y$ are usually connected with data reduction, therefore there is no gain in reducing the error rate for given distributions.

2. Distributions are not known in practice, so feature extraction has the following advantages:

   (a) less amount of data;

   (b) fixed dimension of observation vector $x, y \in \mathbb{R}^D$;

(c) Dependencies between components of x or y can be captured easier (see problem: estimation of covariance matrix).

3. (Possible) strategy:

(a) Fully automatic determination of a mapping $s \longrightarrow x$ (or $x \longrightarrow y$) requires a good criterion, e.g. the empirical error rate (on the training data).

$\Longrightarrow$ the resulting optimization problem is almost always too complex

(b) Choice of mapping $s \longrightarrow x$ is task-specific, based on specific knowledge about the problem (e.g. spectral analysis for speech, "primitives" for character recognition, textures for images).

(c) For the last step $x \longrightarrow y$, a linear mapping is specified:

$$\begin{aligned} V: \ \mathbb{R}^D &\longrightarrow \mathbb{R}^d, \qquad d < D \\ x &\longmapsto y = Vx \end{aligned}$$

The mapping V is determined using an optimization criterion which will be considered in the following.

8.2 Linear Mappings

8.2.1 Effect for given Gaussian Distribution

Given μ and Σ:

(example for illustration)
$x \in \mathbb{R}^D$: $p(x|\mu_x, \Sigma_x)$
If $V \in \mathbb{R}^{D \times D}$ is invertible, "distance ratios" remain unchanged.

Proof:

$$y = V \cdot x \implies \mu_y = E\{y\} = V \cdot \mu_x$$
$$\Sigma_y = \cdots = V \cdot \Sigma_x \cdot V^T$$

$$\begin{aligned} d_x^2 &\equiv (x-\mu_x)^T \Sigma_x^{-1} (x-\mu_x) \quad \text{(exponent in } p(x|\mu_x, \Sigma_x)) \\ d_y^2 &\equiv (y-\mu_y)^T \Sigma_y^{-1} (y-\mu_y) \\ &= (x-\mu_x)^T \cdot \underbrace{V^T \Sigma_y^{-1} V}_{\substack{V^T (V^T)^{-1} \Sigma_x^{-1} V^{-1} V \\ = \Sigma_x^{-1}}} \cdot (x-\mu_x) \end{aligned}$$

$$\implies d_x^2 \equiv d_y^2$$

8.2.2 Minimization of Representation Error: Karhunen-Loève Transformation

([Fuk90], p.417)

other names: KL expansion, Principal Component Analysis, Hotelling transformation [also principal axes transformation, diagonalization]

$x \in \mathbb{R}^D$ [class affiliation is not given]
Basis vectors, orthogonal and normalized:
$V = [v_1, \ldots, v_i, \ldots, v_D], \quad v_i \in \mathbb{R}^D, \quad i = 1, \ldots, D$
where

$$v_i^T v_j = \begin{cases} 1, & \text{i=j} \\ 0, & \text{otherwise} \end{cases}$$

Decomposition:

$$x = \sum_{i=1}^{D} \alpha_i(x) \cdot v_i, \qquad \text{where } \boxed{\alpha_i(x) = v_i^T \cdot x}$$

Approach:
Starting from $i > d$ replace coefficients $\alpha_i(x)$ with constant β_i (independent of x):

$$\begin{aligned} x &= \sum_{i=1}^{d} \alpha_i(x) \cdot v_i + \sum_{i=d+1}^{D} \alpha_i(x) \cdot v_i \\ &\Downarrow \\ \hat{x} &= \sum_{i=1}^{d} \alpha_i(x) \cdot v_i + \sum_{i=d+1}^{D} \beta_i \cdot v_i \end{aligned}$$

Criterion: choose basis vectors so that the representation error

$$\begin{aligned} F(d) &= E_x\{\|x - \hat{x}\|^2\} \\ &= \int_x dx \; ||x - \hat{x}||^2 \cdot p(x) \\ &= E_x\{\| \sum_{i=d+1}^{D} (\alpha_i(x) - \beta_i) \cdot v_i\|^2\} \end{aligned}$$

is minimized.

$$\begin{aligned} F(d) &= E_x\{\sum_{i>d} \sum_{j>d} (\alpha_i(x) - \beta_i) \cdot (\alpha_j(x) - \beta_j) \cdot v_i^T v_j\} \\ &= E_x\{\sum_{i>d} (\alpha_i(x) - \beta_i)^2\} \\ &= \sum_{i>d} E_x\{(\alpha_i(x) - \beta_i)^2\} \end{aligned}$$

Optimal choice for β_i (v_i = average value of $\alpha_i(x)$, $\mu := E_x\{x\}$):

$$\beta_i \;=\; E_x\{\alpha_i(x)\} = E_x\{v_i^T \cdot x\} = v_i^T \cdot E_x\{x\} = v_i^T \cdot \mu$$

$$\boxed{\beta_i = v_i^T \cdot \mu}$$

Rewriting $F(d)$ results in:

$$\begin{aligned} F(d) &= \sum_{i>d} E_x\{[v_i^T \cdot (x-\mu)]^2\} \\ &= \sum_{i>d} E_x\{v_i^T \cdot (x-\mu) \cdot (x-\mu)^T \cdot v_i\} \\ &= \sum_{i>d} v_i^T \cdot \underbrace{E_x\{(x-\mu) \cdot (x-\mu)^T\}}_{\Sigma} \cdot v_i \\ &= \sum_{i>d} v_i^T \cdot \Sigma \cdot v_i \end{aligned}$$

Minimization of $F(d)$ in terms of $v_i \in \mathbb{R}^D$ where $\|v_i\|^2 = 1$ results in the equation for eigenvalues:

$$\Sigma \cdot v_i \;=\; \lambda_i \cdot v_i, \qquad\qquad \lambda_i > 0.$$

(Derivation: either clever estimation using linear algebra or derivation with the aid of the Lagrange formalism)

Sort the eigenvalues in descending order: $\lambda_1 > \lambda_2 > \ldots > \lambda_D$. Then set

$$F(d) := \sum_{i>d} \lambda_i$$

$[v_1, \ldots, v_{d^*}]$ is then the matrix for dimension reduction (with suitably chosen d^*).

Note:
The same derivation is also possible when the representation error is averaged over one sample (difference: $\lambda_i \geq 0$).

Karhunen-Loève-transformation:

1. Calculate (empirical) covariance matrix Σ;
2. Diagonalize Σ;
3. Arrange eigenvectors v_i according to eigenvalues λ_i;
4. Choose (*trial-and-error*) new dimension d; sometimes, the relative error as a criterion is helpful:

$$E_d = \frac{F(d)}{F(0)} = \frac{\sum\limits_{i>d} \lambda_i}{\sum\limits_{i=1}^{D} \lambda_i} \; .$$

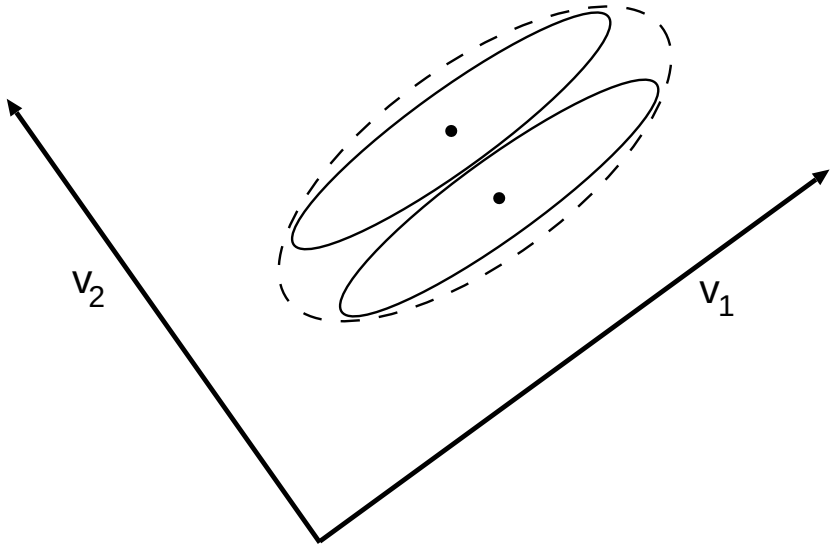
Notes:

1. The term "class" or "class separability" does not occur at all in the derivation.
2. Therefore there is no reason to expect any optimality for class separation (*warning:* in the literature often unclear or wrong).
3. Derivation is possible without any assumption about distributions.
4. New features $\alpha_i(x) = v_i^T \cdot x$ have diagonal covariance matrix, i.e. they are "decorrelated".
5. If (in context of time sequence analysis/signal analysis, see [Ney99a])

$$\Sigma_{ij} \;\; = \;\; R(i-j)$$

holds, i.e. if Σ_{ij} has a *Töplitz structure*, the diagonalization is achieved using a Fourier/cos transformation.

Counter-example (for point 2):
"Karhunen-Loève transformation is good for class separability."



- v_1 : important for representation error (Karhunen-Loève transformation)
- v_2 : important for class separation

8.2.3 Linear Discriminant Analysis (LDA; Fisher's LDA)

([DH73], p.118-120)

Given: training data $x_n \in \mathbb{R}^D$ where $n = 1, \ldots, N$:

$$\mu := \frac{1}{N} \sum_x x$$

For each class A_k:

$$\mu_k := \frac{1}{n_k} \sum_{x \in A_k} x$$

$$n_k := |A_k| = \text{number of observations belonging to class } k$$

Desired: linear mapping $x \longrightarrow y = V^T x$ such that

i) distances within one class remain constant;

ii) distances between class centers μ_k are maximized.

Let $\mathrm{tr}(X) := \sum_{d=1}^{D} X_{dd}$ be the trace of a matrix $X \in \mathbb{R}^{D \times D}$,
such that $u^T u = \|u\|^2 = \sum_d u_d^2 = \mathrm{tr}(uu^T)$ for $u \in \mathbb{R}^D$.
Then:

for i)

$$\begin{aligned} D_W(V) &:= \sum_k \sum_{x \in A_k} \|V^T(x - \mu_k)\|^2 \\ &= \mathrm{tr}\Big[\sum_k \sum_{x \in A_k} V^T (x - \mu_k)(x - \mu_k)^T V\Big] \\ &= \mathrm{tr}\Big[V^T \underbrace{\sum_k \sum_{x \in A_k} (x - \mu_k)(x - \mu_k)^T}_{=:W} V\Big] \end{aligned}$$

W : Within-Class Scatter Matrix

for ii)

$$\begin{aligned} D_B(V) &:= \sum_k n_k \|V^T(\mu_k - \mu)\|^2 \\ &\vdots \\ &= \mathrm{tr}\Big[\sum_k n_k\, V^T (\mu_k - \mu)(\mu_k - \mu)^T V\Big] \\ &= \mathrm{tr}\Big[V^T \underbrace{\sum_k n_k\, (\mu_k - \mu)(\mu_k - \mu)^T}_{=:B} V\Big] \end{aligned}$$

B : Between-Class Scatter Matrix

The result is the *Total Scatter Matrix*

$$T := \sum_x (x-\mu)(x-\mu)^T$$

independent of class affiliations.

Then the following holds:

$$\boxed{T = W + B}$$

Transition trace → determinant:
Within-Class Scatter Matrix is given by the volume of a hyper-ellipsoid; the volume is described by the determinant of the matrix:

$$\text{volume} = const \cdot \pi \cdot \prod_{i=1}^{D} \text{axis}_i$$

Optimization criterion:
Choose a dimension reducing mapping V so that

$$F(V) = \frac{\det(V^T B V)}{\det(V^T W V)} \stackrel{!}{=} \text{maximum}$$

$$\text{resp.} \quad F(V) = \frac{\text{tr}(V^T B V)}{\text{tr}(V^T W V)} \stackrel{!}{=} \text{maximum}$$

or alternatively

$$a^T W a \stackrel{!}{=} \text{minimum} \quad \text{and} \quad a^T B a \stackrel{!}{=} \text{maximum}$$

resp.

$$a^T W a \stackrel{!}{=} \text{minimum} \quad \text{with constraint} \quad a^T B a = 1$$

or

$$a^T B a \stackrel{!}{=} \text{maximum} \quad \text{with constraint} \quad a^T W a = 1$$

Solution: generalized eigenvectors $[v_d]_1^D$
with constraint $v_d^T W v_D = 1 \quad \forall d = 1, \ldots, D.$

$$B \cdot v_d = \lambda_d \cdot W \cdot v_d, \quad \lambda_d \in \mathbb{R}, \quad d = 1 \ldots D, \quad v_d \in \mathbb{R}^D.$$

due to $T = W + B$, it follows:

$$\boxed{T \cdot v_d = (1 + \lambda_d) \cdot W \cdot v_d}$$

a) determine eigenvectors v_d
b) sort according to eigenvalues λ_d

Direct derivation:

Mapping $V = [v_1, \ldots, v_d, \ldots, v_D]$ where $v_d \in \mathbb{R}^D$, choose each vector v_d so that:

- $v_d^T \cdot B \cdot v_d =$ maximum
- with constraint: $v_d^T \cdot W \cdot v_d =$ *const*

$$\text{(equivalent to: } \frac{v_d^T \cdot B \cdot v_d}{v_d^T \cdot W \cdot v_d} = maximum)$$

Lagrange: $f(v_d, \lambda_d) = v_d^T \cdot B \cdot v_d - \lambda_d (v_d^T \cdot W \cdot v_d - 1)$

Calculation (linear algebra, differentiation) results in:

$$B \cdot v_d = \lambda_d \cdot W \cdot v_d.$$

Notes:

1. Solution for V is not unique due to the properties of functions tr and det:

   an invertible mapping leaves the optimum invariant.

2. Without dimension reduction, i.e. $V \in \mathbb{R}^{D \times D}$, the following holds:

$$F(I) = F(V)$$

   In words:
   LDA does not make sense without dimension reduction. A regular mapping in the new feature space leaves the criterion $F(V)$ invariant.

3. Eigenvalues λ_d are invariant with respect to invertible linear transformations. The following holds ([DH73], p.223):

$$\begin{aligned} \mathrm{tr}(W^{-1}B) &= \sum_d \lambda_d & tr(W^{-1}T) &= \sum_d (1+\lambda_d) \\ \det(W^{-1}B) &= \prod_d \lambda_d & det(W^{-1}T) &= \prod_d (1+\lambda_d) \end{aligned}$$

   $(= 0$ because of 4., if $D \geq K)$

4. Between-Class Scatter Matrix B has the rank $(K-1)$ according to the definition (see covariance matrix for Gaussian model), i.e. only $(K-1)$ eigenvalues λ_d are not equal to zero.

5. Special case: $W = I$ (*Isotropy*)

   Then holds: $\{\mu_k - \mu;\ k = 1, \ldots, K\}$

   $\Downarrow$

   Gram-Schmidt orthogonalization

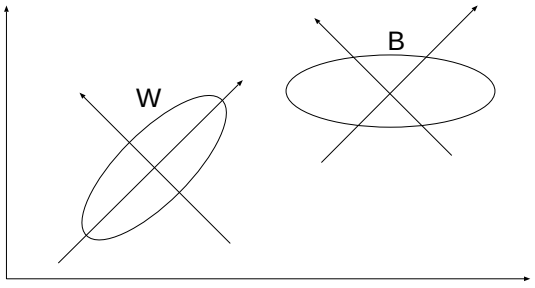
   $\Downarrow$

   Eigenvectors v_k

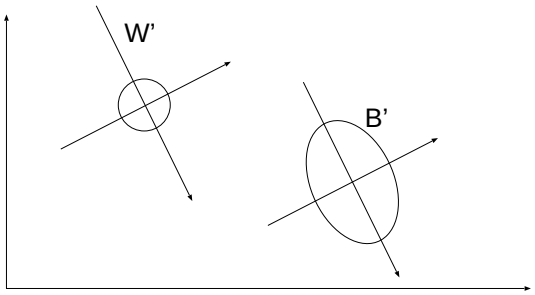
6. The derivation of LDA does not use any Gaussian model, only distances.

Illustration ("simultaneous diagonalization") ([Fuk90],p.31)

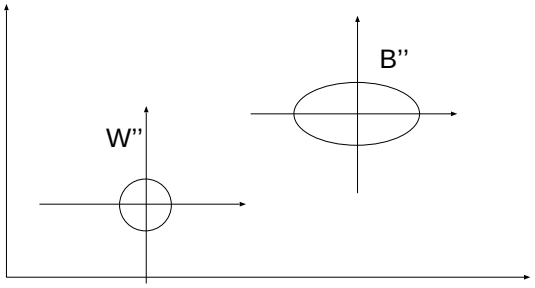
1. Determine principal axes of W and assign them to B.



2. Scale axes so that W is a hyper-sphere. Determine principal axes of B and assign them to W.



3. Rotate the axes appropriately.



4. Reduce dimension according to new axes.

8.2.4 Motivation for Determinant Criterion

- Basics
  Gaussian model: $\mathcal{N}(x|\mu,\Sigma),\ x \in IR^D$
  Data: $x_1,\ldots,x_N$

$$
\begin{aligned}
F(\mu,\Sigma) &:= \sum_n \log \mathcal{N}(x_n|\mu,\Sigma) \\
&= -\frac{1}{2}\sum_n [\log\det(2\pi\Sigma) + (x_n-\mu)^T\Sigma^{-1}(x_n-\mu)] \\
&\quad \text{(where } a^Tb = tr(ab^T) = tr(ba^T) \\
&\quad \text{and } tr(A+B) = tr(A) + tr(B)\ ) \\
&= -\frac{N}{2}\log\det(2\pi\Sigma) - \frac{1}{2}\sum_n tr(\Sigma^{-1}(x_n-\mu)(x_n-\mu)^T)] \\
&\quad \text{(insert ML estimate } \hat{\mu} \text{ and } \hat{\Sigma} = \frac{1}{N}\sum_n (x_n-\hat{\mu})(x_n-\hat{\mu})^T) \\
&= -\frac{N}{2}[\log\det(2\pi\hat{\Sigma}) + tr(I_D)] \\
&= -\frac{N}{2}[\log\det(2\pi\hat{\Sigma}) + D] \\
&= -\frac{N}{2}\log\det(2\pi e\hat{\Sigma})
\end{aligned}
$$

- Model score
  class-dependent models: $p(x|k) = \mathcal{N}(x|\mu_k, W)$

  Score: N samples
  $$-\frac{N}{2}\log\det(2\pi eW)\ (\star)$$

  $p(x) = \mathcal{N}(x|\mu,T)$ is used instead of $p(x) = \sum_k p(k)\cdot p(x|k)$, where T is the total scatter matrix.
  Score: $-\frac{N}{2}\log\det(2\pi e\tilde{T})$ with $\tilde{T} = V^TTV$, where V^T is the transpose of V.
  This value is optimized, and the value $(\star)$ is kept constant.

8.3 Linear Classificators/Discriminants

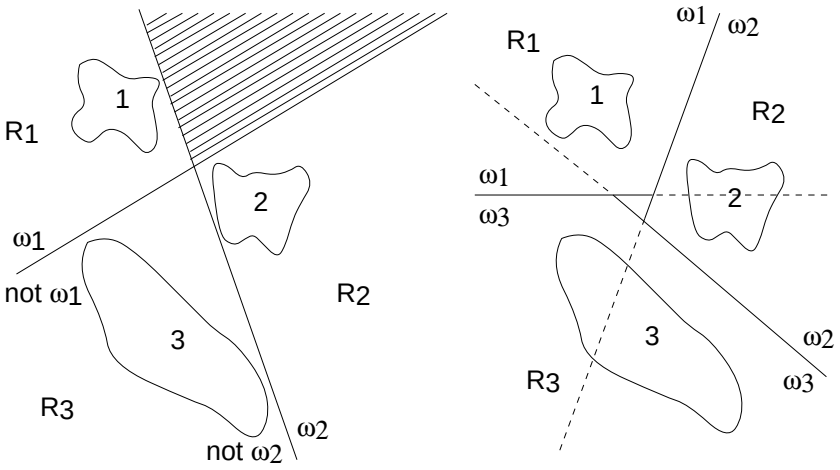
(see Chapter 4.2)

Number of classes K is arbitrary (i.e. not only 2 classes).

Reminder: Gaussian model with $\Sigma_k = \Sigma$ leads to linear limit surfaces.

Method (different from typical ones in the literature):

1. general case $K > 2$;
2. no assumptions about class separability;
3. one single criterion instead of considering class pairs ([DH73], p.151,177).



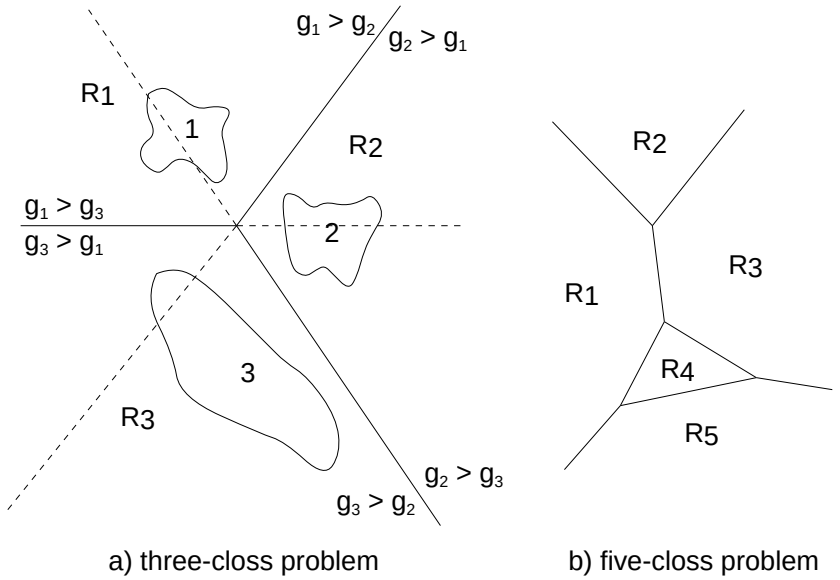


Figure: linear classification boundaries.

Model:
Decision rule:

$$r(x) \;:=\; \operatorname*{argmax}_k \{g(x,k)\}$$

Discriminant:

$$g(x,k) \;=\; v_k^T \cdot x, \qquad x, v_k \in \mathbb{R}^D.$$

Training: (ideal values)

$$\begin{aligned} v_k^T x &\stackrel{!}{=} 1\ , && \text{if } x \text{ belongs to class } k, \\ v_k^T x &\stackrel{!}{=} 0\ , && \text{if } x \text{ does not belong to class } k\ . \end{aligned}$$

In general, ideal demands can be realized only approximatively. Therefore the squared error criterion is usually chosen:

$$F(v_1 \ldots v_k) = \sum_n \sum_c [v_c^T x_n - \delta(c, k_n)]^2,$$

with training data (x_n, k_n) and Kronecker function δ:

$$\delta(c,k) = \begin{cases} 1, & c = k \\ 0, & c \neq k \end{cases}\ .$$

Solution:

F is a quadratic function of v_k; thus resulting in a linear equation system for v_k where $k = 1, \ldots, K$. The structure of this equation system is similar to normal equations and pseudo-inverses.

generalized features:

1. $g(x,k) = b_k + v_k^T x,$ $\qquad b_k \in \mathbb{R},\ v_k \in \mathbb{R}^D$
2. $g(x,k) = b_k + v_k^T x + x^T W_k x,$ $\qquad v_k \in \mathbb{R}^D,\ W_k \in \mathbb{R}^{D \times D}$
3. $g(x,k) = v_k^T f(x),$ $\qquad f : \mathbb{R}^D \to \mathbb{R}^{D'},\ v_k \in \mathbb{R}^{D'}$

Notation: *polynomial discriminant function, regression classificator*

Index

References

[Ama67] S. I. Amari. A Theory of Adaptive Pattern Classifiers. *IEEE Trans. on Elec. Com.*, EC 16:279–307, 1967.

[Ben75] J.L. Bentley. Multidimensional Binary search Trees used for Associative Searching. *Communications of ACM*, 18(9):509–517, September 1975.

[BM94] Bourlard and Morgan. *Connectionist Speech Recognition - A Hybird Approach.* Kluwer Academic Publishers, 1994.

[CB90] G. Casella and R.L. Berger. *Statistical Inference.* Wadsworth & Brooks/Cole, Pacific Grove, CA, 1990.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley series in telecommunications. John Wiley & Sons, New York, NY, 1991.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, NY, 1973.

[DK82] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[FBF77] J.H. Friedman, J.L. Bentley, and R.A. Finkel. Finding Best Matches in Logarithmic Expected Time. *ACM Trans. on Mathematical Software*, 3(3):209–226, September 1977.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Computer Science and Scientific Computing. Academic Press, Inc., San Diego, CA, 2nd edition, 1990.

[Gor81] A. D. Gordon. *Classification.* Monographs on Applied Probability and Statistics. Chapman and Hall, London, 1981.

[HAJ90] X. D. Huang, Y. Ariki, and M. A. Jack. *Hidden Markov Models for Speech Recognition.* Edingurgh Information Technologie Series. Edinburgh University Press, 22 George Square, Edinburgh, 1990.

[Kee65] D. G. Keehn. A note of learning for Gaussian properties. *IEEE Trans. on Information Theory*, IT-11:126–132, 1965.

[Kre91] U. Krengel. *Einf"uhrung in die Wahrscheinlichkeitstheorie und Statistik.* Vieweg Studium, Aufbaukurs Mathematik. Vieweg, Braunschweig, 3rd edition, 1991.

[Kun93] S.Y. Kung. *Digital Neural Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Trans. Communications*, 28:84–95, January 1980.

[Lip87] R. P. Lippmann. An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, pages 4 – 22, April 1987.

[Ney99a] H. Ney. Vorlesungsmitschrift Spracherkennung und Suchverfahren, WS 1999/2000, 1999.

[Ney99b] H. Ney. Vorlesungsmitschrift stochastische modellierung in der mustererkennung ss 1999, 1999.

[Nie83] H. Niemann. *Klassifikation von Mustern.* Springer-Verlag, Berlin, 1983.

[Par82] D.B. Parker. Learning Logic. Invention Report, S81-64, File 1, Office of Technology Licensing, Stanford University, 1982.

[RJ93] L.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Sch92] R. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches.* John Wiley & Sons, New York, NY, 1992.

[Spa77] H. Spath. *Clusteranalyse-Algorithmen.* R. Oldenbourg Verlag, M"unchen, 2nd edition, 1977.

[Wer74] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.