Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI

Lecture Notes

Statistical Classification and Machine Learning

Prof. Dr.-Ing. Hermann Ney

Winter Semester 2017/2018

Version: Monday 9th October, 2017
Last modified: Yunsu Kim

This lecture note can be used for:

Bachelor: “Statistische Klassifikation und Maschinelles Lernen”
typically chapters 1-6 ($\sim$75% of the contents)
6 ECTS (V3/Ü2)

Master: “Statistische Klassifikation und Maschinelles Lernen”
full lecture
8 ECTS (V4/Ü2)

References

Textbooks on Statistical Classification and Learning (Pattern Recognition, Neural Networks, Data Mining, ...)

[Duda & Hart$^+$ 01] R. O. Duda, P. E. Hart, D. G. Stork:
Pattern Classification.
2nd ed., J. Wiley & Sons, New York, NY, 2001.
(best introduction including modern concepts)

[Ripley 96] B. D. Ripley:
Pattern Recognition and Neural Networks.
Cambridge University Press, Cambridge, England, 1996.
(emphasis on statistical concepts)

[Hastie & Tibshirani$^+$ 01] T. Hastie, R. Tibshirani, J. Friedman:
The Elements of Statistical Learning: Data Mining, Inference and Predictions.
Springer, New York, 2001.
(emphasis on modern statistical concepts)

[Devroye & Györfi$^+$ 96] L. Devroye, J. Györfi, G. Lugosi:
A Probabilistic Theory of Pattern Recognition.
Springer, New York, 1996.
(emphasis on theory and principles)

[Bishop 06] C. M. Bishop:
Pattern Recognition and Machine Learning.
Springer, New York, 2006.
(pattern recognition from a machine learning point-of-view)

Textbooks on Mathematical Toolbox (vector spaces and matrices, statistics, optimization methods, ...)

[Moon & Stirling 99] T. K. Moon, W. C. Stirling:
Mathematical Methods and Algorithms for Signal Processing.
Prentice Hall, Upper Saddle River, NJ, 1999.
(best overall summary)

[Press & Teukolsky$^+$ 92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery:
Numerical Recipes in C.
Cambridge Univ. Press, Cambridge, 2nd ed., 1992.
(good overview of numerical algorithms and implementations)

[Boyd 04 & Vandenberghe] S. Boyd, L. Vandenberghe:
Convex Optimization.
Cambridge Univ. Press, Cambridge, 2004.

Textbooks on Statistics and Bayesian Methods

[Casella & Berger 90] G. Casella, R. L. Berger:
Statistical Inference.
Wadsworth & Brooks/Cole, Pacific Grove, CA, 1990.
(good introduction into modern general statistics)

[Berger 85] J. O. Berger:
Statistical Decision Theory and Bayesian Analysis.
Springer, New York, 2nd ed., 1985.
(topic: Bayes Methods)

[Carlin & Louis 96] B. P. Carlin, T. A. Louis:
Bayes and Empirical Bayes Methods for Data Analysis.
Chapman & Hall, London, 1996.
(topic: Bayes and Empirical Bayes Methods)

Textbooks on Speech Recognition

[Rabiner & Juang 93] L. Rabiner, B. H. Juang:
Fundamentals of Speech Recognition.
Prentice Hall, Englewood Cliffs, NJ, 1993.
(emphasis on signal processing and small-vocabulary recognition)

[Jelinek 98] F. Jelinek:
Statistical Methods for Speech Recognition.
MIT Press, Cambridge, 1998.
(emphasis on large vocabulary and language modeling)

[Jurafsky & Martin 00] D. Jurafsky, J. H. Martin: Speech and Language Processing.
Prentice Hall, Englewood Cliffs, NJ, 2000.
(introduction to both speech and language)

[Mori 98] R. De Mori: Spoken Dialogues with Computers.
Academic Press, London, 1998
(advanced topics)

Textbooks on Natural Language Processing (using statistical/corpus-based methods)

[Manning & Schütze 99] C. D. Manning, H. Schütze:
Foundations of Statistical Natural Language Processing.
MIT Press, Cambridge, MA, 1999.
(best book on statistical methods for written language)

[Charniak 94] E. Charniak:
Statistical Language Learning.
MIT Press, Cambridge, MA, 1994.
(principles of statistical methods for written language)

[Jelinek 98] F. Jelinek:
Statistical Methods for Speech Recognition.
MIT Press, Cambridge, 1998.
(emphasis on language modelling and large vocabulary speech recognition)

Conventional Linguistics and Artificial Intelligence

[Jurafsky & Martin 00] D. Jurafsky, J. H. Martin:
Speech and Language Processing.
Prentice Hall, Englewood Cliffs, NJ, 2000.
(mixture of conventional and statistical approaches)

[Russel & Norvig 03] S. Russel, P. Norvig:
Artificial Intelligence.
2nd ed.,Prentice Hall, Upper Saddle River, NJ, 2003.
(comprehensive textbook on artificial intelligence:
– chapters 13-21: probabilistic approaches and learning,
– chapters 22-25: communication, language, perception, robotics)

[Nilsson 98] N. J. Nilsson:
Artificial Intelligence: A New Synthesis.
Morgan Kaufmann Pub., San Francisco, CA, 1998.
(standard textbook on artificial intelligence)

Additional Literature

[Bourlard & Morgan 93] H. A. Bourlard, N. Morgan:
Connectionist Speech Recognition – A Hybrid Approach.
Kluwer Academic Publishers, Boston, MA, 1993.

[Duda & Hart 73] R. O. Duda, P. E. Hart:
Pattern Classification and Scene Analysis.
J. Wiley, New York, NY, 1973.

[Devijver & Kittler 82] P. A. Devijver, J. Kittler:
Pattern Recognition: A Statistical Approach.
Prentice Hall, Englewood Cliffs, NJ, 1982.

[Fukunaga 90] K. Fukunaga:
Introduction to Statistical Pattern Recognition.
Academic Press, New York, NY, 1990.

[Niemann 83] H. Niemann:
Klassifikation von Mustern.
Springer, Berlin, Germany, 1983.

[Schalkoff 91] R. Schalkoff:
Pattern Recognition: Statistical Structural and Neural Approaches.
J. Wiley, New York, NY, 1991.

[Bishop 96] C. M. Bishop:
Neural Networks for Pattern Recognition.
Oxford University Press, Oxford, UK, 1996.

[Haykin 98] S. Haykin:
Neural Networks. A Comprehensive Foundation.
Prentice Hall, Englewood Cliffs, NJ, 1998.

[Kung 94] S. Y. Kung:
Digital Neural Networks.
Prentice Hall, Englewood Cliffs, NJ, 1994.

[Huang et al. 91] X. D. Huang, Y. Ariki, M. A. Jack:
Hidden Markov Models for Speech Recognition.
Edinburgh University Press, Edinburgh, UK, 1991.

[Krengel 90] U. Krengel:
Einführung in die Wahrscheinlichkeitstheorie und Statistik.
2nd ed., Vieweg, Braunschweig, Germany, 1990.

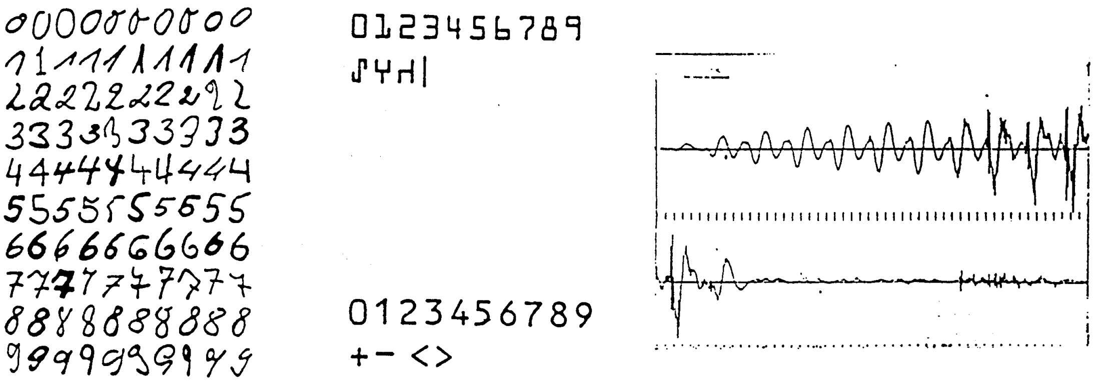
Extension / Application Lectures

- Statistical Methods in Natural Language Processing
- Automatic Speech Recognition
- Advanced Methods in Automatic Speech Recognition
- Advanced Topics in Machine Learning and Human Language Technology

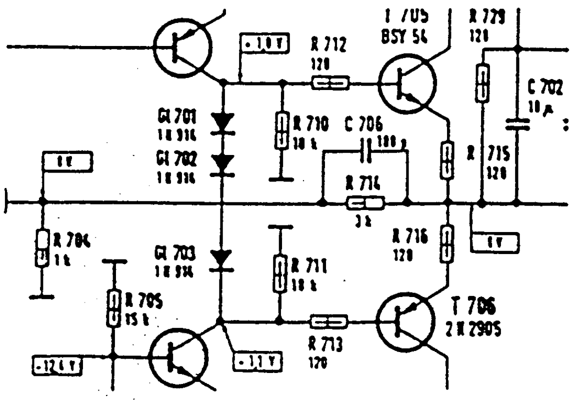
Contents

1 Introduction

Three examples of simple patterns:
left: hand-written digits
center: standardized digits
right: part of the microphone voltage curve for the German word “mit”



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



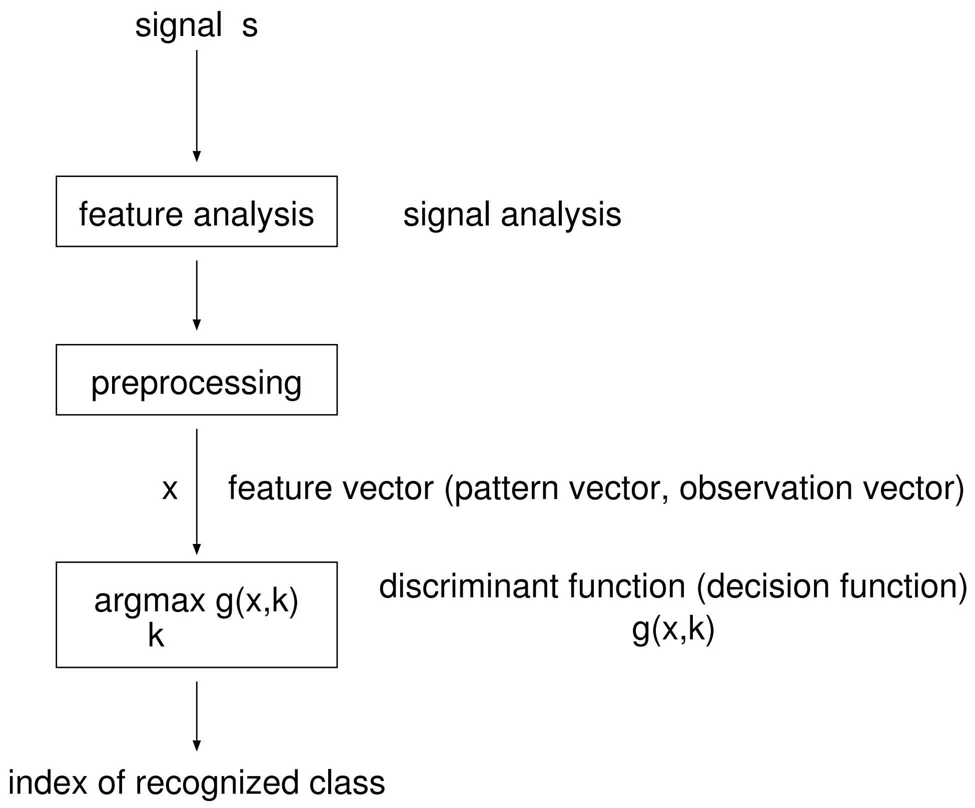
1.1 Pattern Recognition Domains

- Signals (“one-dimensional”)
  - acoustic signals (noise, motors, ...)
  - speech
  - ECG, 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)

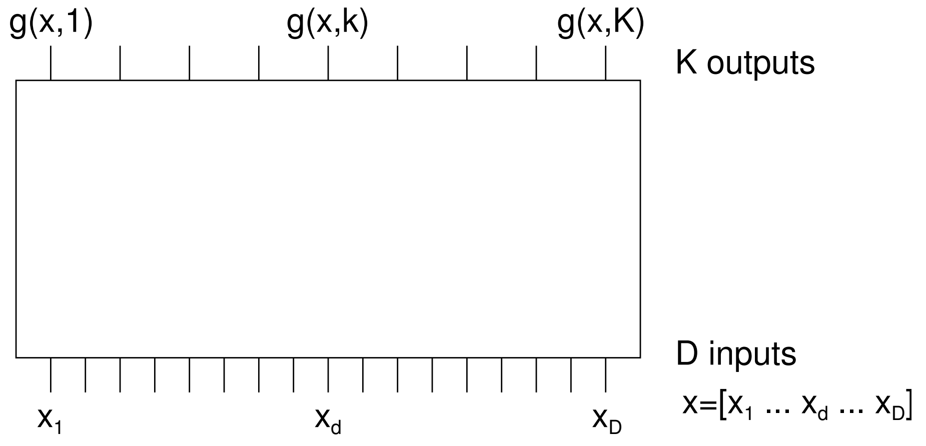

signal s
feature analysis
signal analysis
preprocessing
x
feature vector (pattern vector, observation vector)
argmax g(x,k)
k
discriminant function (decision function)
g(x,k)
index of recognized class

Given the observed signal s and its feature vector $x \in \mathbb{R}^D$, the corresponding class is to be determined. 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 $x \mapsto g(x,k), \quad k = 1, \ldots, K$:



There is no constraint for the functional dependency $x \mapsto g(x,k)$. Nevertheless, we often use in practice:

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

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

  $\downarrow$

  polynomial classifiers

- neural networks (ANN = artificial neural network), especially the *multilayer perceptron* (with one or two "hidden" layers)

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

$$\begin{array}{ll} g(x,k) \longmapsto 1 & \text{if } k \text{ is the "right" class} \\ g(x,k) \longmapsto 0 & \text{if } k \text{ is the "wrong" class} \end{array}$$

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

  The *a priori* probabilities are usually calculated as relative frequencies.

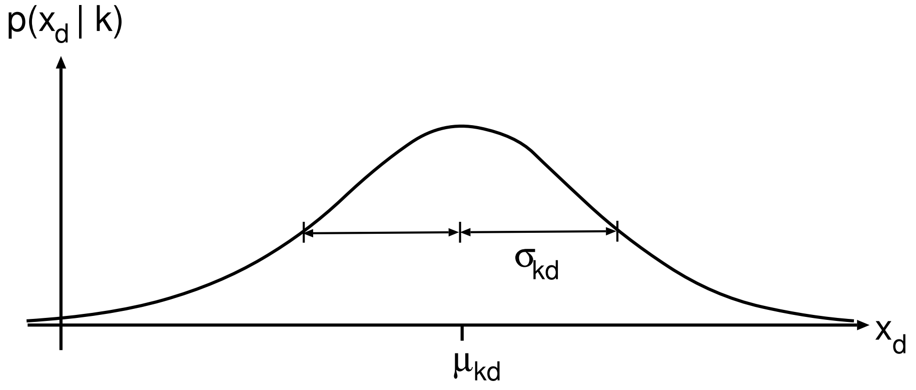
  Example - letter recognition: different letters have different frequencies.

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

  Example:

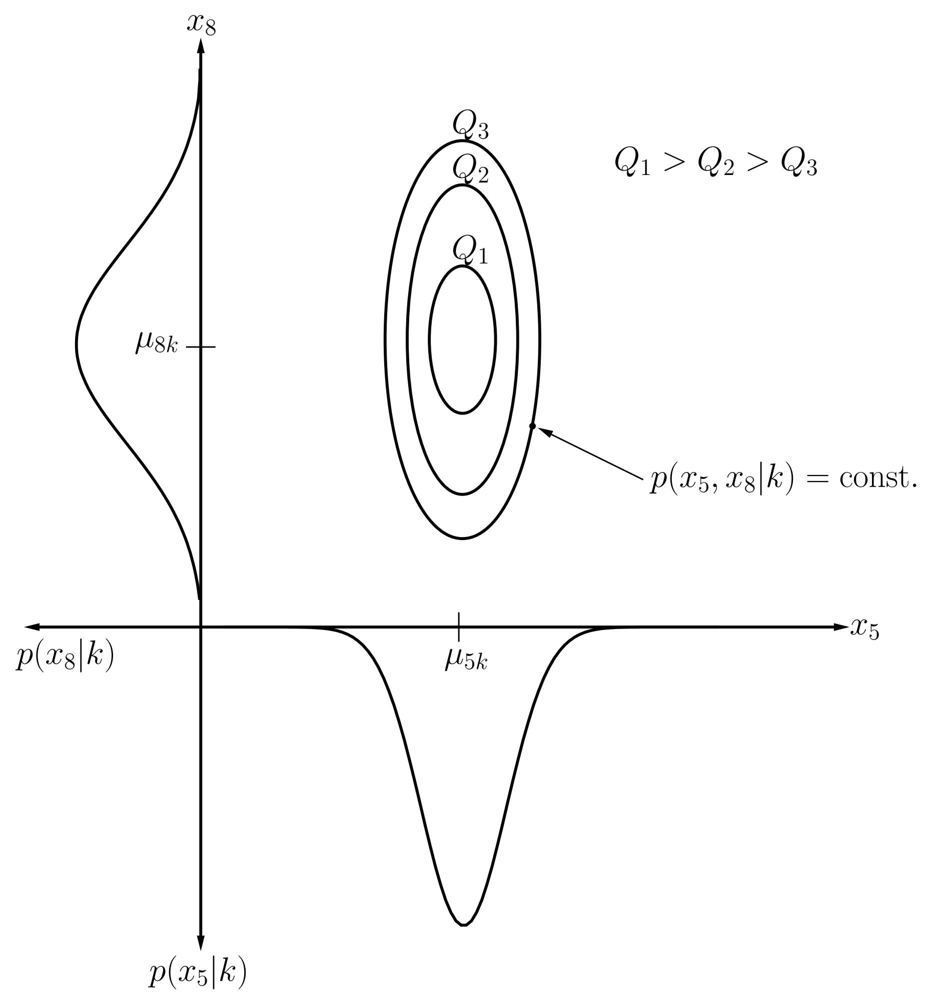
$$p(x|k) = \prod_{d=1}^{D} p(x_d|k)$$

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



μ_{kd} : mean of the component x_d for class k
σ^2_{kd} : variance of the component x_d for class k

Example (two-dimensional):



How do we estimate $p(k)$ and $p(x|k)$?

$p(k)$: relative frequencies

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

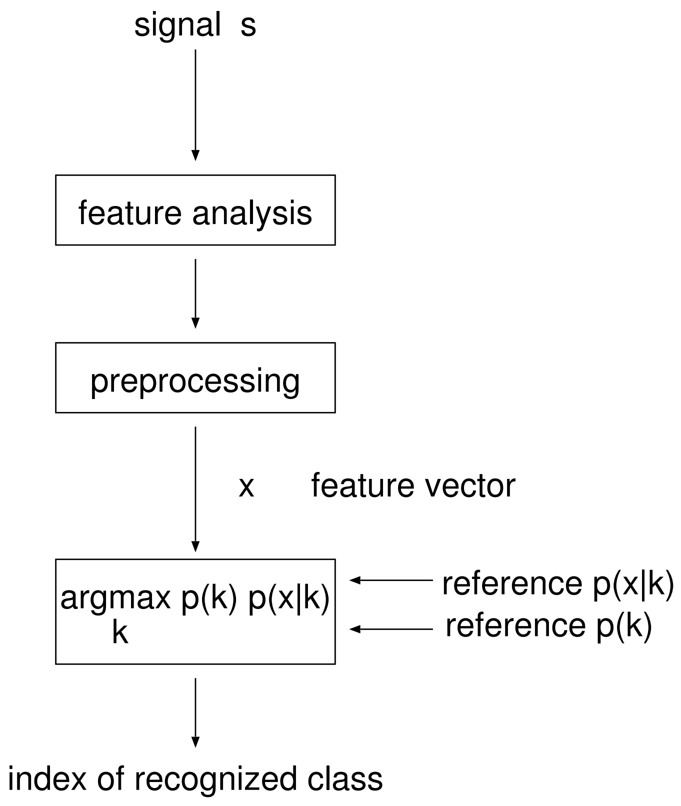
$$\left.\begin{array}{l}\text{mean } \mu_{kd} \\ \text{variance } \sigma_{kd}^2\end{array}\right\} \text{for each component } x_d \text{ and class } k$$

Number of parameters:

$K - 1$ values for $p(k)$ (normalization: $\sum_{k=1}^{K} p(k) = 1$)

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

The adapted pattern recognition scheme:



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

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

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

Ideally, the discriminant function $g(x,k)$ approximates the *a posteriori* probability $p(k|x)$.

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

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

Tasks 1 and 4 are investigated in the lecture *Speech Recognition* [Ney 99a].

1.3 Random Variables and Distributions

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

- Discrete random variable:

  Examples:

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

  Probability distribution:

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

  Expectation of function $x \longmapsto g(x)$:

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

  Mean ("weighted averaging") of x:

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

  Variance of x:

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

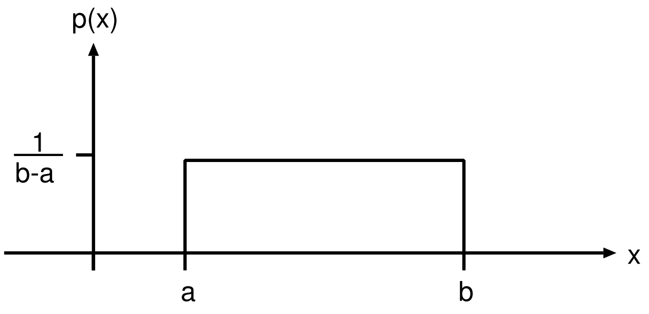
- One-dimensional, continuous random variable: $x \in \mathbb{R}$

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

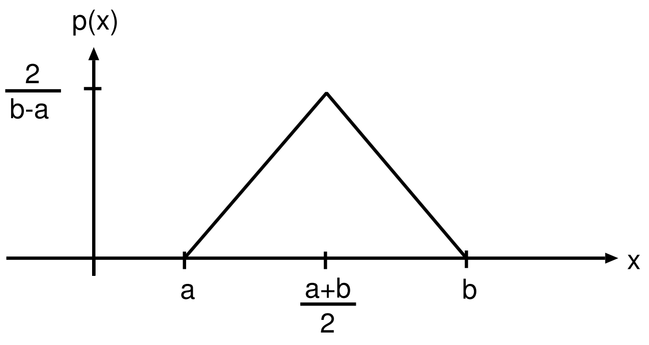
Distribution densities: $p(x) \geq 0, \qquad \int\limits_{-\infty}^{\infty} dx\, p(x) = 1$

Examples:

– uniform distribution:



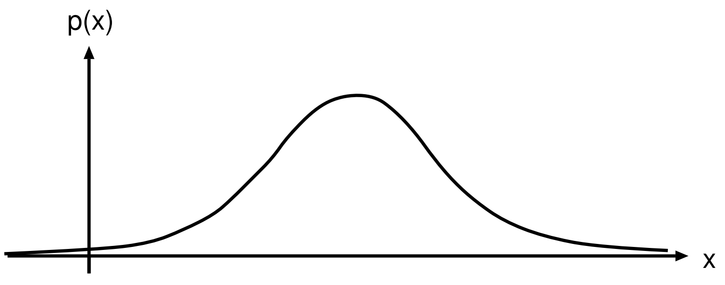
– triangular distribution:



These examples show a probability density $p(x)$ whose values are equal to zero outside a finite interval.

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

– Gaussian distribution, normal distribution (bell-shaped 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 parameters a, b or $\mu, \sigma^2, \ldots$ are necessary for the complete definition of the densities.

Expectation, mean and variance for continuous random variables are defined by using the integral instead of the sum:

Expectation of a function $x \longmapsto g(x)$:

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

Mean:

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

Variance:

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

Exercise: Calculate the mean 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.

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

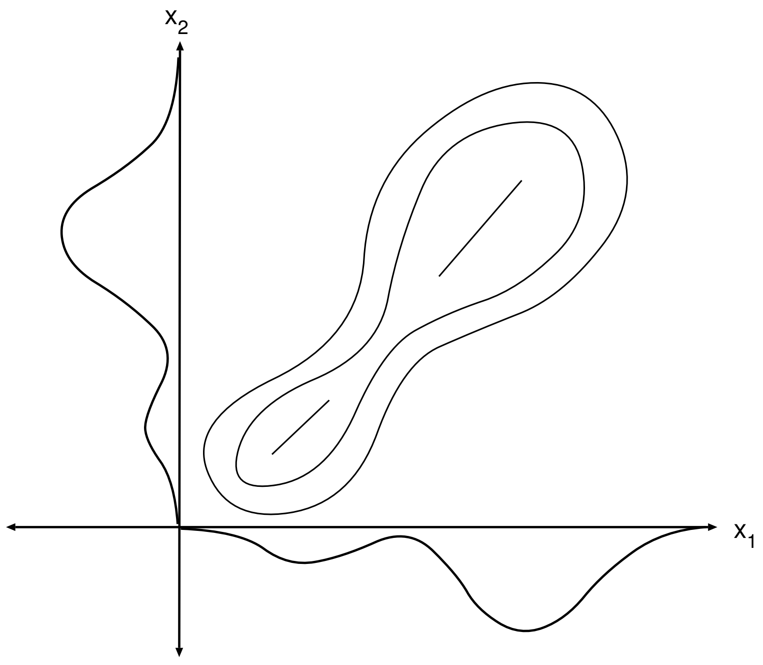
$$p(x) = p(x_1, \ldots, x_d, \ldots, x_D) \geq 0$$

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

To simplify matters, the following considerations use the two-dimensional space $(D = 2)$: $\quad x = [x_1, x_2]$

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

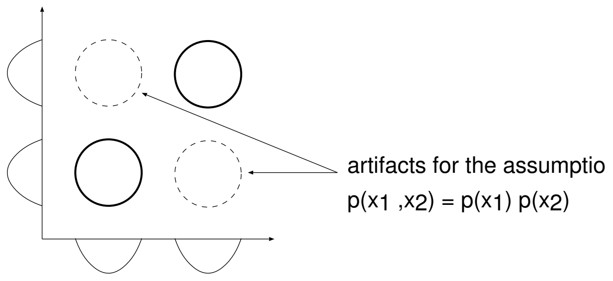
Representation of density $p(x_1, x_2)$ by iso-probability contours, i.e. lines with constant probability density value:



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

$$
\begin{aligned}
p(x_1) &= \int_{-\infty}^{\infty} dx_2 \, p(x_1, x_2) \; , \\
p(x_2) &= \int_{-\infty}^{\infty} dx_1 \, p(x_1, x_2) \; .
\end{aligned}
$$

In general, $p(x_1, x_2) \neq p(x_1) \cdot p(x_2)$.



Conditional distribution densities:

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

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

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

Stochastic independence of variables means that the knowledge about one of the two variables does not contain any information about the other. For $D > 2$, the definition is analogue.

Expectation of a function:

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

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

Mean 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 mean 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

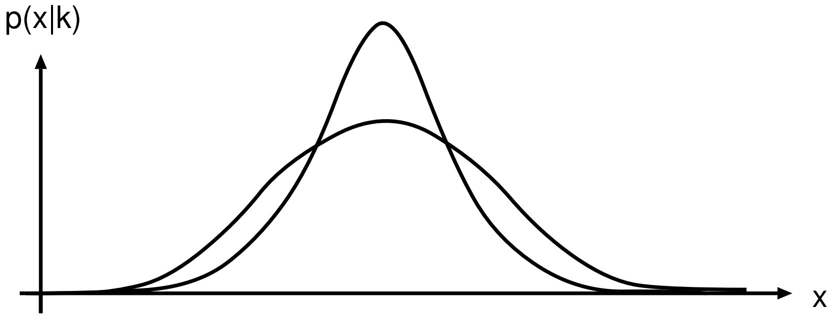
We consider the conditional probability (density) $p(x|k)$ and the observation vector $x \in \mathbb{R}^D$ for one fixed class k.

Observation vectors usually vary around one “typical” prototype vector. These variations are often described by Gaussian distributions.

- Univariate Gaussian distribution:

  We fix one vector component d. The distribution of x_d is defined as

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



The parameters μ_{kd} and σ_{kd}^2 are determined by

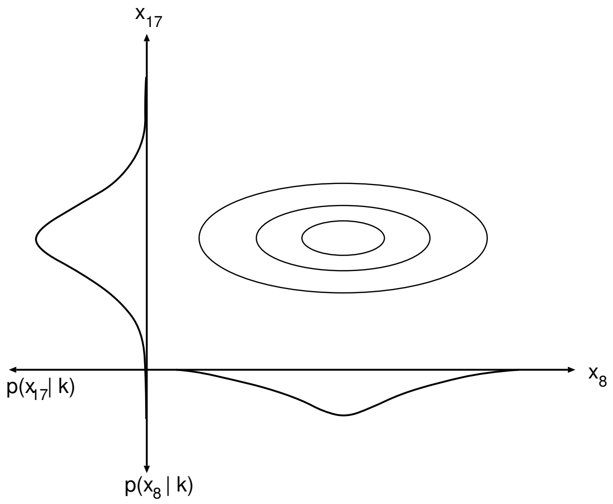
$$\int_{-\infty}^{+\infty} dx_d \; x_d \; p(x_d|k) \;=\; \mu_{kd} \qquad \text{mean}$$

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

- Multivariate Gaussian distribution with independent components:

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

The iso-probability contours $p(x|k) = \mathit{const}$ are hyperellipsoids whose main axes are parallel to the coordinate axes:



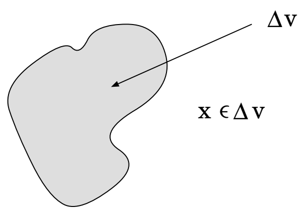
The negative logarithm of $p(x|k)$ can be interpreted as the distance between x and μ_k in $\mathbb{R}^D$:

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

Remarks concerning the dimension:

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

3. Dimension of the probability density $p(x|k)$:
   In order to obtain a probability, we have to integrate over a volume element.



$$Pr(x \in \Delta\text{v}|k) \;=\; \int_{\Delta\text{v}} dx'\, p(x'|k) \cong |\Delta\text{v}| \cdot p(x|k)$$

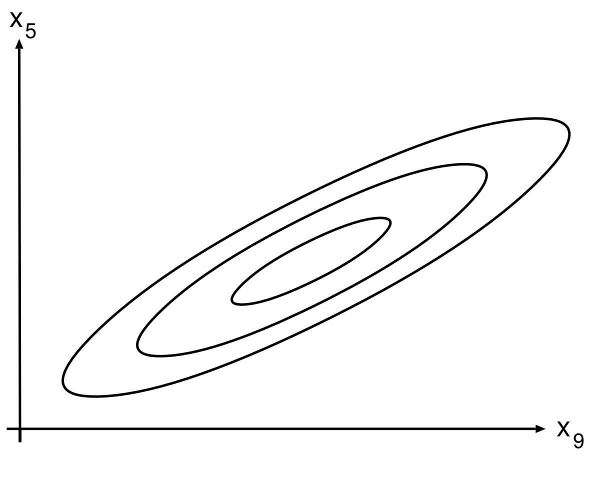
- multivariate Gaussian distribution:

  The general multivariate Gaussian distribution is obtained if the exponent has an arbitrary quadratic positive definite form.

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

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

The iso-probability contours $p(x|k) = const$ are *rotated hyperellipsoids.*



Taking the normalization $\int\limits_{\mathbb{R}^D} dx\; p(x|k) = 1$ into account, we get

$$\begin{aligned} p(x|k) &= \frac{1}{\sqrt{(2\pi)^D \;\det \Sigma_k}} \exp\left[-\frac{1}{2}(x-\mu_k)^T\, \Sigma_k^{-1}\, (x-\mu_k)\right] \\ &=: \mathcal{N}(x|\mu_k, \Sigma_k). \end{aligned}$$

The mean of the component d becomes

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

and the covariance of the 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 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.

Preview:

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 mean 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 $\mathbb{R}^D$

Now, we consider two methods to construct a distribution in $\mathbb{R}^D$.

1. Build a multivariate distribution from the product of univariate distributions. We assume pairwise independence of the univariate distributions.

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

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

   Example: Laplace distribution

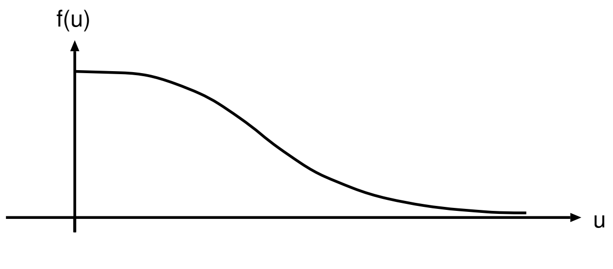
$$\begin{aligned} p(x_d|k) &= \frac{1}{2v_{kd}} \exp\left[-\frac{|x_d - \mu_{kd}|}{v_{kd}}\right] \\ p(x_1, \dots, x_D|k) &= \prod_{d=1}^{D} p(x_d|k) = \left[\prod_{d=1}^{D} \frac{1}{2v_{kd}}\right] \exp\left[-\sum_{d=1}^{D} \left|\frac{x_d - \mu_{kd}}{v_{kd}}\right|\right] \end{aligned}$$

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 monotonic 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

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

depending on matrix W_k.

Example: t-distribution

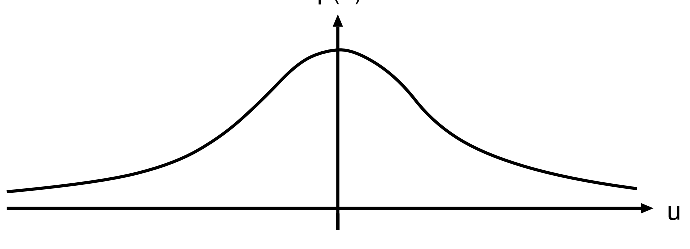
$$p(x|k) \;=\; \frac{1}{C_{NORM}}[1 + \frac{1}{M_k}(x-\mu_k)^T\, W_k\, (x-\mu_k)]^{-\left(\frac{M_k+1}{2}\right)},$$

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 the polynomial decrease of the t-distribution for $(x-\mu_k)^T\, W_k\, (x-\mu_k) \to +\infty$.

A special case of a t-distribution is the 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:

- The **binomial distribution** yields the distribution for the output of a sequence of independent Bernoulli experiments. The output of a Bernoulli experiment is binary (A: event occurs; $\overline{A}$: otherwise). Example: throwing a coin. If $p(A) = \theta$ is the probability that the event A occurs (and thus $p(\overline{A}) = 1 - \theta$) 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:

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

- The **multinomial distribution** (also polynomial distribution) generalizes the binomial distribution from binary events to I different events $A_1, ..., A_I$. Given the total number of events $N = \sum\limits_{i=1}^{I} n_i$, the probability that the event A_i occurs n_i times with success probability $p(A_i) = \theta_i$ is provided by the multinomial distribution:

$$p(n_1, n_2, \ldots, n_I|N) \;=\; N! \prod_{i=1}^{I} \frac{\theta_i^{n_i}}{n_i!}$$

where $\sum\limits_{i=1}^{I} \theta_i = 1$.

- The **Poisson distribution** is 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:

- The **univariate Gaussian distribution** (normal distribution) is a particular limit value of the binomial distribution for the case $n \to \infty$ when p is constant. The univariate Gaussian distribution with the mean $\mu \in \mathbb{R}$ and the variance $\sigma^2 \in \mathbb{R}^+$ is defined as:
$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- The **multivariate Gaussian distribution** is the multidimensional generalization of the Gaussian distribution with the mean vector $\mu \in \mathbb{R}^D$ and the covariance matrices $\Sigma \in \mathbb{R}^{D \times D}$:
$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \cdot \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right]$$

- The **Laplace distribution** with means μ_d and variances v_d for $d = 1, ..., D$ is defined as
$$p(x|\mu, v) = \prod_{d=1}^{D} \frac{1}{2v_d} \cdot \exp\left[-\frac{|x_d - \mu_d|}{v_d}\right]$$

- The χ^2 distribution is a distribution of a sum $\chi^2 = \sum_{i=1}^{n} x_i^2$ of n squared Gaussian-distributed random variables x_i with the mean 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):
$$p(\chi^2|n) = \frac{1}{2^{\frac{n}{2}} \cdot \Gamma\left(\frac{n}{2}\right)} \cdot (\chi^2)^{\frac{n}{2}-1} \cdot \exp\left(-\frac{\chi^2}{2}\right)$$

- The t-distribution (Student's distribution) is a distribution of quotients $t = x/\chi^2$ of a Gaussian-distributed random variable x with the mean $\mu = 0$ and the variance $\sigma^2 = 1$ and a 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 Distributions used in Statistical Classification

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

Assumption: The following distributions are completely 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 conditional distributions (or models) of feature vectors x: $p(x|k)$

Now, we are able to derive the 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 conditional distribution of x for 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:

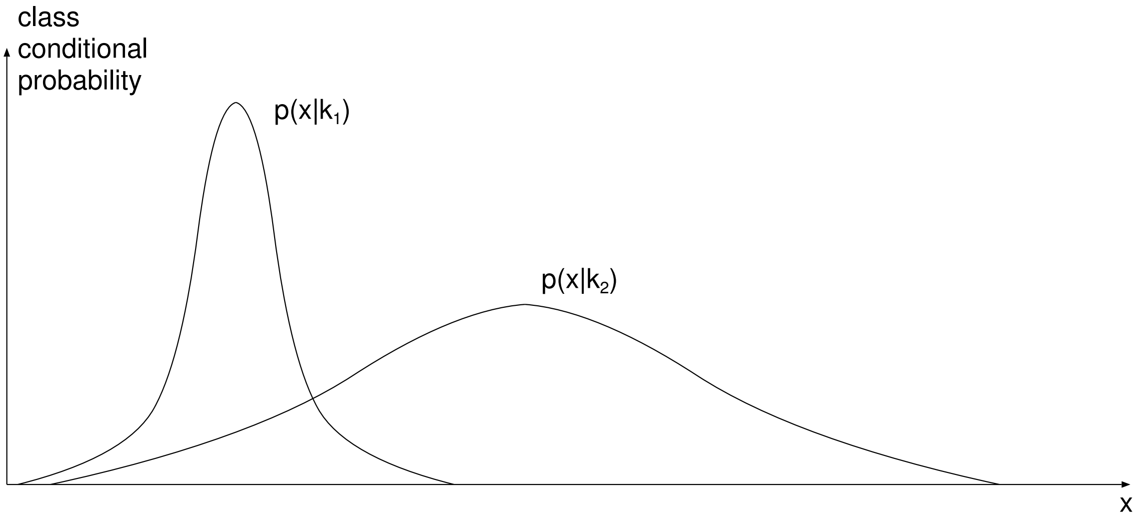
$$p(k|x) \;=\; \frac{p(x,k)}{p(x)} = \frac{p(k)\cdot p(x|k)}{\sum\limits_{c=1}^{K} p(c)\cdot p(x|c)}$$

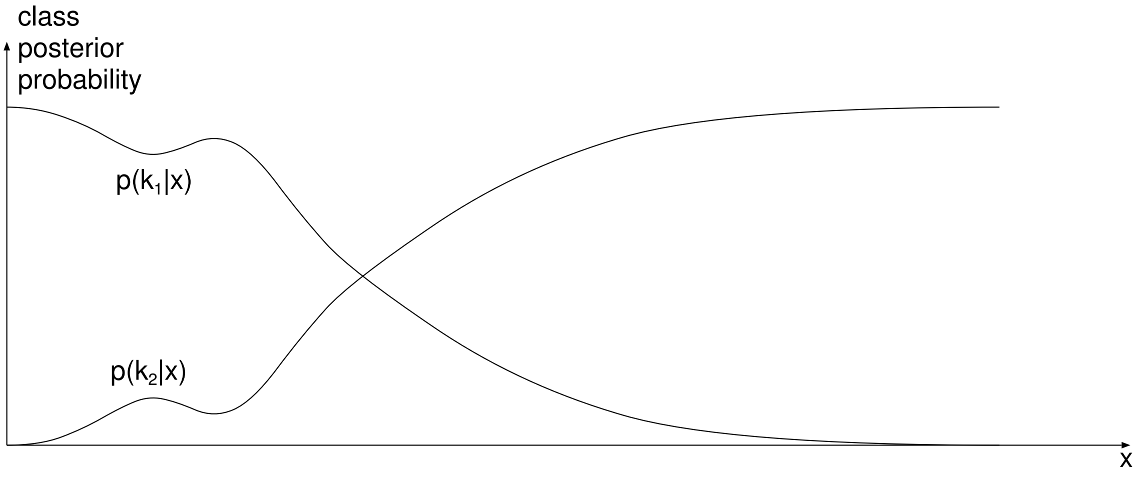
  We note that the normalization condition

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

  is fulfilled.

Example: $D = 1$ dimensions,
$K = 2$ classes,
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.

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

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

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

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

The costs for a given feature vector $x \in \mathbb{R}^D$ are minimized using the following decision rule:

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

("Bayes' decision rule in pattern recognition")

We are often only interested in the recognition errors. These errors can be defined 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

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

("Bayes' decision rule for minimum error rate")

Error rate of 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 of 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!
- Consider the conditions!

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-form expression for Bayes error rate is very rare because the speech recognition problem comprises
  - "too many" classes and
  - "too many" dimensions.

2.3 Discriminants and Decision Boundaries

The Bayes' decision rule is the starting point

$$x \mapsto r(x) = \operatorname*{argmax}_k \{p(k|x)\} \quad \text{where} \quad p(k|x) = \frac{p(k) \cdot p(x|k)}{\sum\limits_{c=1}^{K} p(c) \cdot p(x|c)}$$

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

$$x \mapsto r(x) \;=\; \operatorname*{argmax}_k \{g(x,k)\},$$

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. Therefore, the error rate is invariant to the choice of such a discriminant.

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

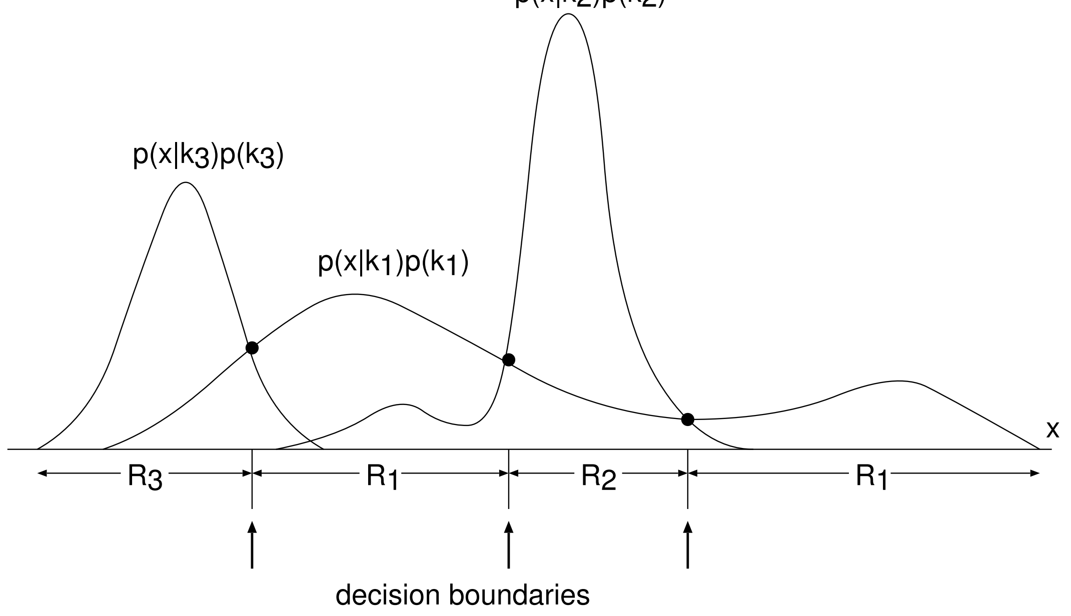
2. $g(x,k) \;:=\; \log p(x,k) = \log p(k) + \log p(x|k)$

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

Decision boundaries arise from discriminants. A decision or class boundary between the classes k and c is defined as follows:

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

Remark: for K classes, there are $\frac{K \cdot (K-1)}{2}$ possible class pairs and in the worst case the same number of decision boundaries.
Example: $D = 1, K = 3$
Consider the joint probability $p(x,k) = p(k) \cdot p(x|k)$:



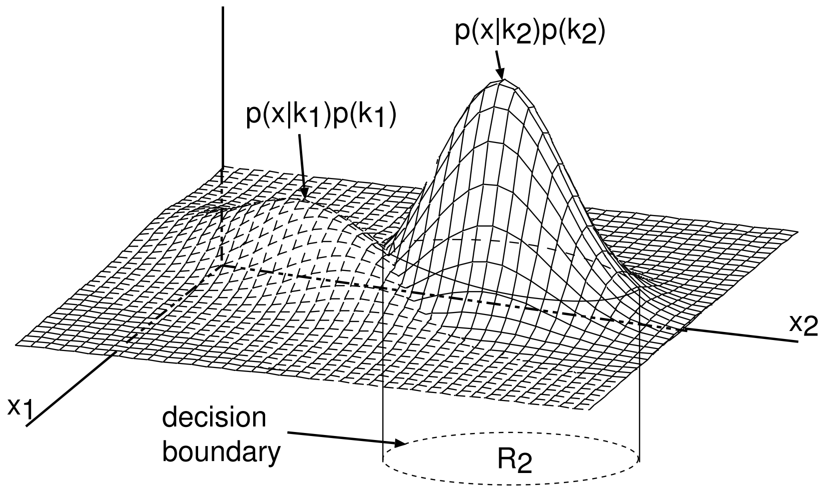
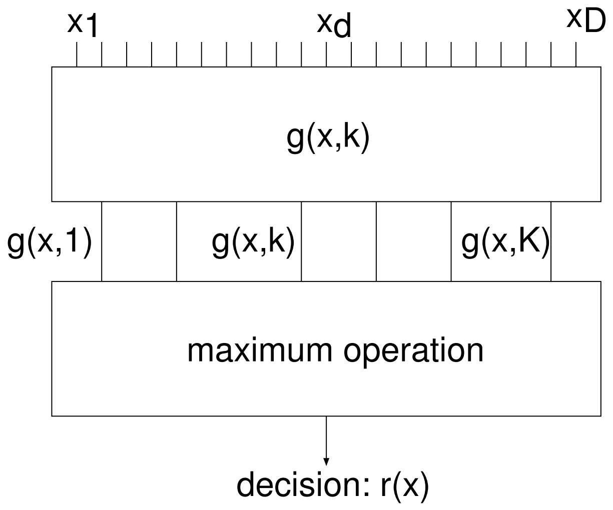


Illustration of the discriminant approach:

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



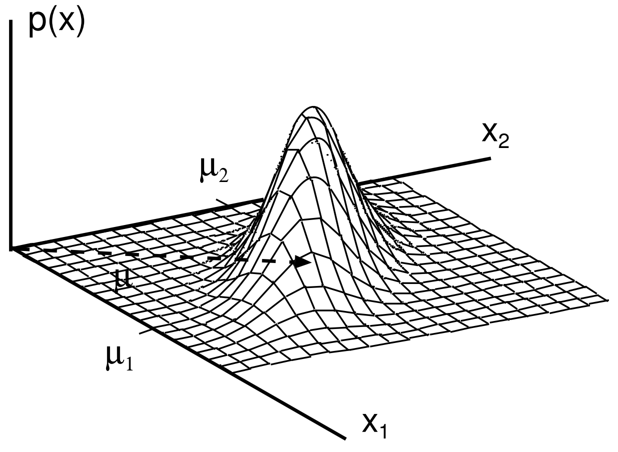
This concept consists of

- discriminant or evaluation functions $g(x,k)$,
- *arg max* operation.

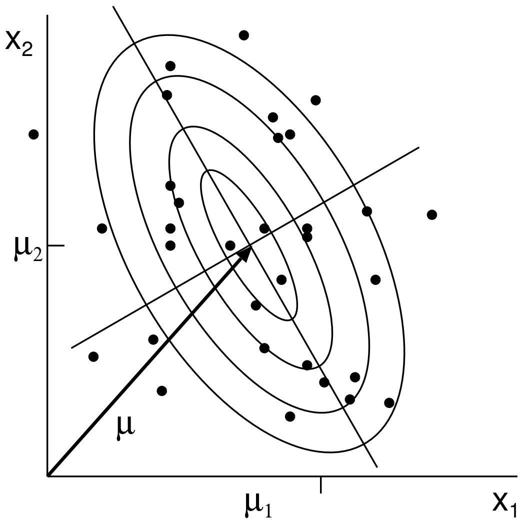
This concept can also be found in other ("non-statistical") approaches:

- neural networks,
- geometric classifiers.

2.4 Multivariate Gaussian Distribution



a) bivariate Gaussian distribution



b) scatter diagram

For each class k and each feature vector $x \in \mathbb{R}^D$, a class conditional probability can be modeled with multivariate Gaussian distribution:

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

where μ_k is the mean vector,
Σ_k is the covariance matrix.

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

We choose the following discriminant function:

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

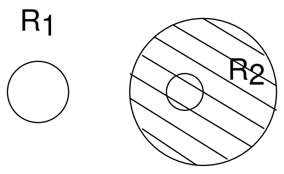
$g(x,k)$ is quadratic in x (*quadratic form*).

Case distinction for determining the decision boundaries:

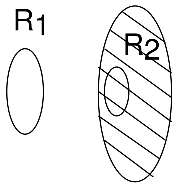
1. Σ_k arbitrary

   Most general case: the decision boundary $\{x \in \mathrm{I\!R}^D : g(x,k) = g(x,c)\}$ for the boundary between two classes k and c is also a quadratic form of x (hyperquadratics: planes, circles, ellipsoids, hyperboloids, paraboloids).

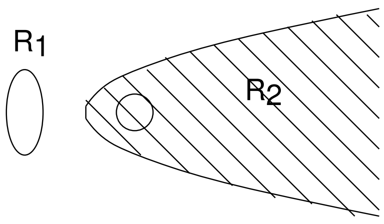
   Example: $x \in \mathrm{I\!R}^2, K = 2$ classes. The shaded areas indicate the decision $r(x) = 2$.



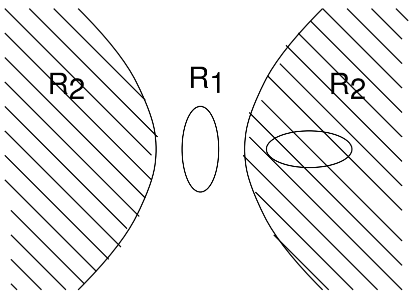
a) circle



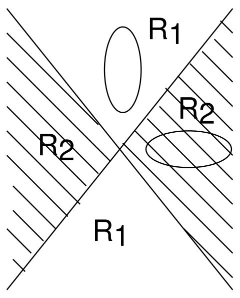
b) ellipse



c) parabola



d) hyperbola



e) straight lines

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

   Decision boundaries are linear functions of x, i.e. hyperplanes, because quadratic terms of x in the equation $g(x,c) = g(x,k)$ cancel out.

   Hint: By applying the principal axis 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 classification with squared *Euclidean distance*)

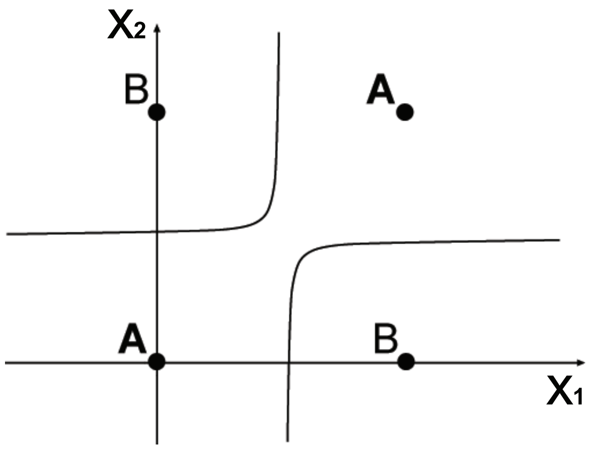
Exercise:

1. Calculate explicitly the class boundary in the case:

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

2. Calculate the above figures for yourself.

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



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

   If yes, how?

2.5 Distance Classifiers 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 with respect to k:

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

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

Distance classifiers:

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

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

We ignore the *a priori* probability $p(k)$ and define

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

so that the decision rule becomes

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

The terminology is not unique; there are the following terms

- minimum distance,
- nearest neighbor,
- 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. Thus, they can be used to define a distance measure.

Special cases

- $p = 1$: absolute value distance (city block 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 decision boundary of l_p-norm classifiers 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*). Finally, this again results in a statistical method.

Determining the error rate for the special case

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

For this special case, a closed-form solution exists.

As shown in chapter 2.4, the assumption $\Sigma_1 = \Sigma_2$ results in a linear decision boundary.

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 $x \in \mathrm{I\!R}^D$ it is sufficient to calculate the distance $h(x)$ between x and the linear decision boundary and to integrate over this distance – instead of integrating over the total space $\mathrm{I\!R}^D$. Since $h(x)$ is a linear function of x, it is also Gaussian-distributed (according to a general theorem about Gaussian distributions).

We need the mean u_k and the variance v_k^2 of $h(x)$ for each class $k = 1, 2$. This results in ([Fukunaga 90], 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 plays an important role and is called *Mahalanobis distance* (between μ_1 and μ_2).

Let $P(E_{1\to2}|k = 1)$ be the conditional probability that the classifier assigns a wrong class $k = 2$ to an observation which actually belongs to the class $k = 1$. $\mathcal{R}_2$ is the set of all observations assigned to class $k = 2$:

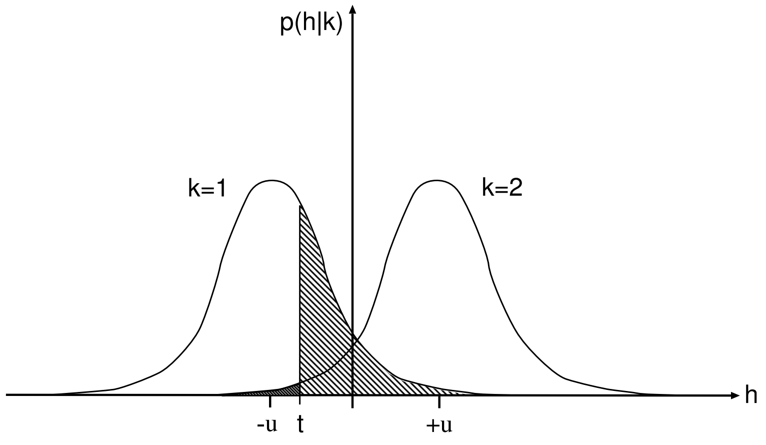
$$\mathcal{R}_2 := \{x \in \mathrm{I\!R}^D | p(2) \cdot p(x|2) > p(1) \cdot p(x|1)\} \ .$$

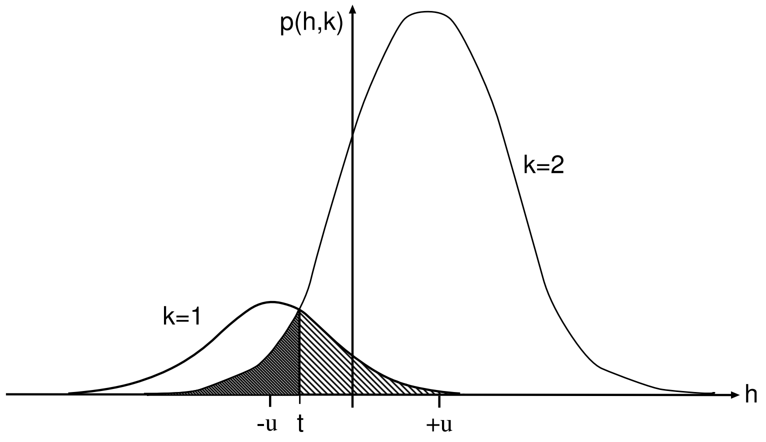
For the class conditional error probability of the class $k = 1$, we get:

$$P(E_{1\to2}|k = 1) = \int_{\mathcal{R}_2} dx \; p(x|k = 1)$$

The class conditional error probability $P(E_{2\to1}|k = 2)$ for the class $k = 2$ can be derived similarly.

The substitution by the distance $h(x)$ to the class boundary enables us to reduce these integrals to one-dimensional integrals as shown in the following figure (shaded and filled). The integration limits result from the class boundary.





Thus, the total error rate $P(E)$ is given by:

$$\begin{aligned}
P(E) &= \int\limits_{-\infty}^{t} dh\, p(h, k=1) + \int\limits_{t}^{\infty} dh\, p(h, k=2) \\
&= p(k=1) \int\limits_{-\infty}^{t} dh\ \underbrace{\mathcal{N}(h|-u, v)}_{p(h|k=1)} + p(k=2) \int\limits_{t}^{\infty} dh\ \underbrace{\mathcal{N}(h|+u, v)}_{p(h|k=2)} \\
&\quad \text{where } t := \log \frac{p(k=1)}{p(k=2)}
\end{aligned}$$

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

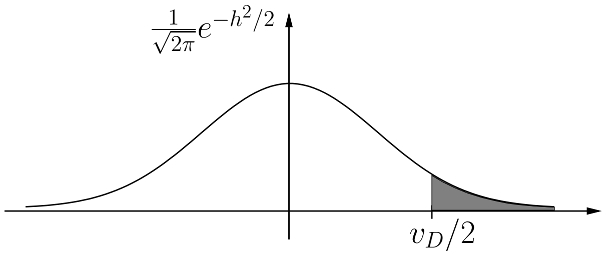
- the ratio of the *a priori* probabilities $\dfrac{p(k=1)}{p(k=2)}$
- and the Mahalanobis distance $v^2 = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1)$.

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

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

i.e. the total error rate $P(E)$ depends only on the Mahalanobis distance. What happens to 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:

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



By means of the above figure it can be easily seen that

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

provided that, for increasing dimension D, we have

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

In this case, the error rate $P(E|v_D^2)$ can be arbitrarily 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 feature space with 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 reduced to zero.

2.6 Binary Features

So far, we have considered random vectors $x \in \mathbb{R}^D$. The Bayes' rule is also applicable to 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

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

In particular, binary features are defined as:

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

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

2.6.1 Independent Binary Features

If $x = [x_1, \ldots, x_d, \ldots, x_D]$ is a binary feature set, then the (class conditional) 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, we have:

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

where ϑ_{kd} is the class conditional *success probability.*
For the whole feature set x, we get

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

We use the discriminant

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

and, by inserting the class conditional 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: Show that, for ternary features $x_d \in \{-1,0,1\}$, we get a quadratic discriminant.

2.6.2 Dependent Binary Features

Rademacher-Walsh Expansion

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

To represent the model $p(x|k)$, 2^D possible probabilities (actually $2^D - 1$ due to the normalization) are necessary.

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 basis functions $\varphi_i(x)$ capture more and more complex dependencies between the features x_d when i increases. They are defined as follows

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

The basis functions are orthogonal, i.e.

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

here, the summation is carried out over all 2^D configurations. Due to the orthogonality, the coefficients $a_i(k)$ become

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

Preview: The training of $a_i(k)$ is performed by *empirical averaging* of the training data $x_1, \ldots, x_N$ for the class k:

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

Remarks to the orthogonal series expansion:

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

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

related methods ([Duda & Hart 73], p.111-114):

- Bahadur-Lazarsfeld expansion
- *dependence tree* by Chow

2.6.3 Decision Rule and Error Rate for a Special Case

Assumptions:

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

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

For the discriminants $g(x,1)$ and $g(x,2)$, we have

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

The difference of the discriminants yields

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

From $\lambda > \frac{1}{2}$ follows $\log \frac{\lambda}{1-\lambda} > 0$, and we obtain the decision rule $r(x)$

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

Determination of the error rate for odd feature vector dimensions D:

case distinction:

1. An observation x which belongs to class 1 is assigned by mistake to class 2. d (with $0 \leq d \leq \frac{D-1}{2}$) of the D components are equal to 1. The probability of this event is

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

The number of different configurations (cf. binomial distribution) is

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

Hence, 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 class 2 is assigned by mistake to class 1. d (with $0 \leq d \leq \frac{D-1}{2}$) of the D components are equal to 0.

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

Consequently, the total error rate $P(E|D,\lambda)$ is

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

We obtain two limiting cases:

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

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

Proof of the case $\lim\limits_{D \to \infty} P(E|D,\lambda) = 0$:

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

Because of

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

it follows that

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

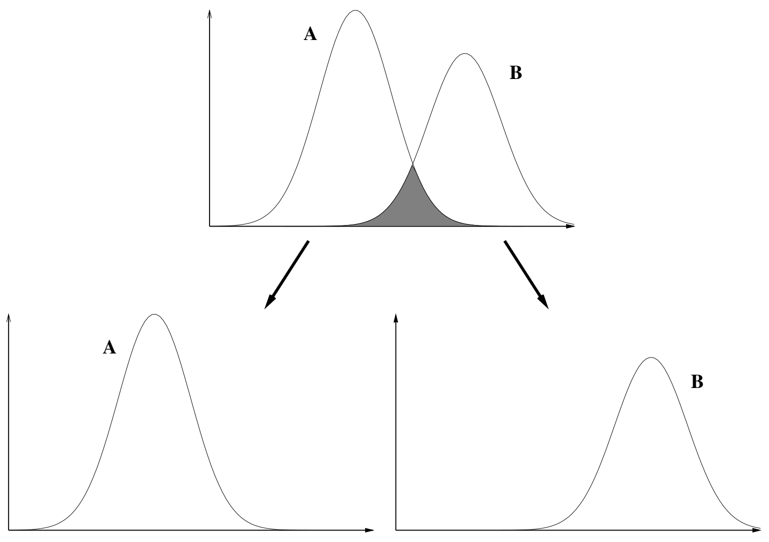
and, consequently,

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

3 Training and Learning

Generally, we do not have *one ideal* learning/estimation method; the choice of a method depends on the task conditions.

In this chapter, we discuss class-individual and nondiscriminative methods, i.e., the methods train the classes A and B independently of each other.



3.1 Task Formulation

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

Within the statistical approach, we utilized

$p(k)$	(*a priori* probabilities) and
$p(x\|k)$	(class conditional distributions or models).

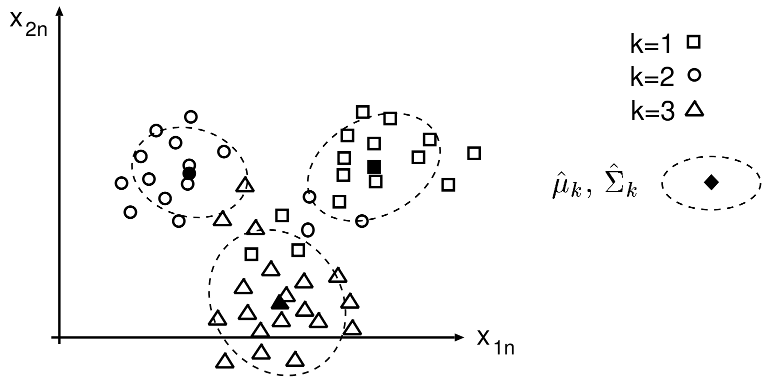
In the first two chapters, we presumed that

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

are known.

In practice, the parameters of the models $p(x|k)$ are not known, and they have to be learned or estimated from training data.
Here, the training data are pairs (x_n, k_n) with $n = 1, \ldots, N$, where 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 quantized with 8 bits for each dimension.
Number of possible values: $256^{16} = 2^{128} \approx 10^{38}$
$\Rightarrow$ there are much fewer observations than possible values.

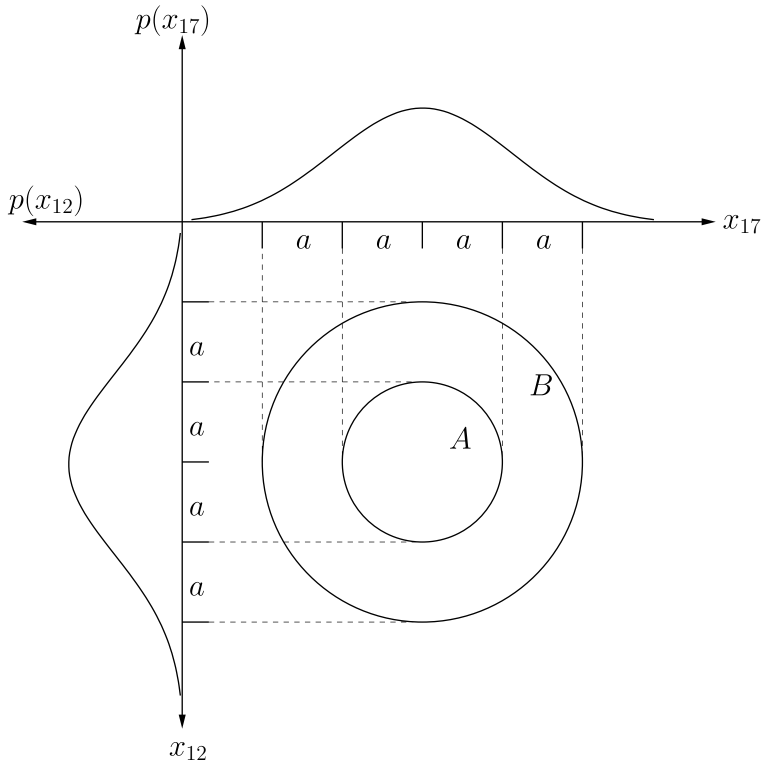
Example: Gaussian distribution

If it is possible to estimate mean 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 reason for 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 $D \gg 1$.
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 compared:

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

For that purpose, we approximate the probability 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 and
	$\langle p(x\|k) \rangle_A$	is the average distribution density in A.

For B, we obtain

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

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 $\frac{\langle p(x|k)\rangle_B}{\langle p(x|k)\rangle_A} = \frac{1}{10}$

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

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 the training data for the high-dimensional feature 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

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

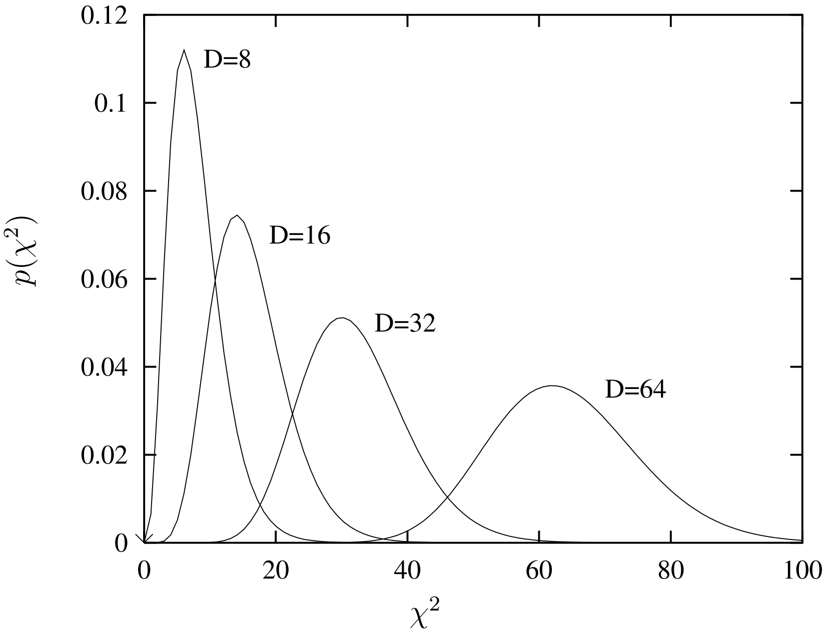
can be calculated if the vectors x are Gaussian-distributed. The derivation of this result can be found in statistics books (e.g. [Krengel 91], p.170-173).

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:

$$\begin{aligned}\chi^2 &= \sum_{d=1}^{D} (x_d - \mu_{kd})^2 \\ p(\chi^2) &= \frac{1}{\Gamma(\frac{D}{2})\, 2^{\frac{D}{2}}} \cdot \left(\chi^2\right)^{\frac{D}{2}-1} \cdot \exp\left(-\frac{1}{2}\chi^2\right)\end{aligned}$$

Graphical representation of the χ^2 distribution:



Properties (without proofs):

- mean: $E\{\chi^2\} = D$
- variance: $Var\{\chi^2\} = 2D$
- limit for $D \to \infty$:

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

The graphical representation of the χ^2 distribution for $D = 8, 16, 32, 64$ shows that the concentration around the mean decreases if D increases.

3.3 Moment Method

We are given N_k training vectors for each class k

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

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

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

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

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

The moment method assigns the moments or central moments to $f(x)$

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

Then, for the mean μ_k

$$\mu_k \ = \ \int_{\mathbb{R}^D} dx \ x \cdot p(x|k) \ ,$$

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

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

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

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

The variances $\hat{\sigma}_{kd}^2$ (diagonal elements of $\hat{\Sigma}_k$) become

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

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

3.4 Maximum Likelihood Method

The moment method fails for complex models. The general, widely used *maximum likelihood method* tries to take this into account.
The dependency of the model $p(x|k)$ on a parameter ϑ is written as

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

Often, the parameter ϑ is class-dependent so that $p(x|k,\vartheta)$ is replaced by

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

For each class k, N_k training vectors are given

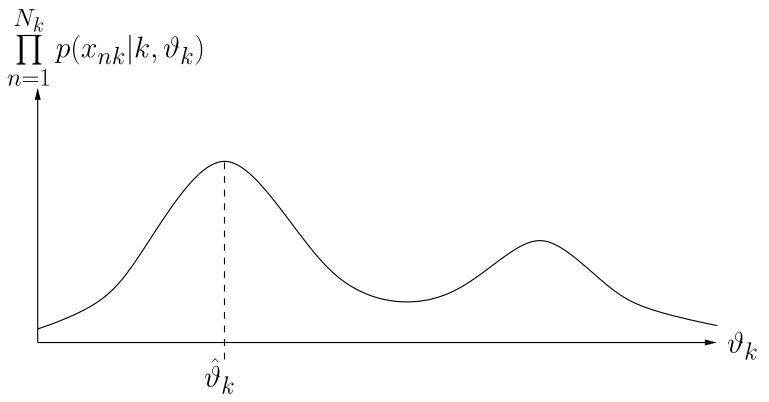
$$x_{1k}, \ldots, x_{nk}, \ldots, x_{N_k k} \qquad \in \mathbb{R}^D \ .$$

Then, the likelihood function is defined as

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

or the log likelihood function, respectively

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



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

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

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

or

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

At this point, it should be checked if a global maximum really exists.

Example: Gaussian distribution with diagonal covariance matrix

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

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

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

Log likelihood for class k:

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

Derivative with respect to μ_{kd}:

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

This results in

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

Derivative with respect to σ_{kd}^2:

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

After inserting the estimate $\hat{\mu}_{kd}$ for μ_{kd}, we get

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

We obtain the usual estimates for mean and variance. They are identical to the estimates by the moment method (instead of $\frac{1}{N_k}$, for the variance, $\frac{1}{N_k-1}$ is often used).

For a Gaussian distribution with an arbitrary covariance matrix Σ_k, we obtain the same estimated values as for the moment method.

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

Derivative using the formula

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

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, we can presume a class independent covariance matrix Σ.

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

As likelihood function, we obtain

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

The maximum likelihood estimation results in

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

Hints:

1. After subtraction of the corresponding mean, each observation x_{nk} is considered only once. Consequently, the contribution of a class with many observations is stronger than that of a class with few observations. If this is not desired, weights like $\sqrt[N_k]{\ldots}$ or $\frac{1}{N_k}$, respectively, have to be used.

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

The maximum likelihood method also yields simple estimates for other distributions:

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

- Binomial distribution:
$$p_N(n|\vartheta) = \binom{N}{n} \vartheta^n (1-\vartheta)^{N-n},$$

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

- 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

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

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 is multinomially distributed. The probabilities have to be normalized: $\sum\limits_{w} p(w|v) = 1 \ \forall \ v$. (There are different methods for assigning a probability > 0 also to unseen events, cf. lecture *Language Modeling.*) Hence, the log likelihood function for a bigram model is

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

$\dfrac{N(v,w)}{N(v)}$ is a distribution since $\sum\limits_{w} \dfrac{N(v,w)}{N(v)} = 1$

$\rightsquigarrow$ maximum likelihood estimate $\hat{p}(w|v) = \dfrac{N(v,w)}{N(v)}$.

Poisson distribution: Modeling of “rare” events (e.g. word counts in a text)
Consider a unigram language model. If the count n_w for each word w is Poisson-distributed with the 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 $N := \sum\limits_{w=1}^{W} n_w$.

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} with $\sum\limits_w \vartheta_{wk} = 1 \ \forall k$
Reduce the text to the counts $n_1 \ldots n_w \ldots n_W := n_1^W$.

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

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

3.5 Practical Aspects

1. Typical problems

   A Gaussian model serves as example for typical problems:

   - The variability of the training sample is too small. Consequently, the variance is underestimated.
   - This underestimation of the variance and the exponential decrease of the Gaussian model lead to a concentration of the probability mass in the center.
   - Often, the covariance matrix $\hat{\Sigma}_k$ becomes singular:

     If we have N_k observation vectors, $\hat{\Sigma}_k$ is constructed of N_k vectors $(x_{nk} - \hat{\mu}_k)$. Only $(N_k - 1)$ of them can be independent because of $\hat{\mu}_k$. I.e., $\hat{\Sigma}_k$ 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. Remedies for singular $\hat{\Sigma}_k$

   - Class-independent covariance matrix $\hat{\Sigma}$:

     In this case, a coordinate transformation can be appropriate.

   - Smoothing $\hat{\Sigma}_k$ using $\hat{\Sigma}$:

     $$\tilde{\Sigma}_k := (1 - \lambda_k)\hat{\Sigma}_k + \lambda_k \hat{\Sigma} \; ,$$

     Here, 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 means $\hat{\mu}_k$ 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 transformation).
   - if necessary:

     Emphasize the diagonal elements of covariance matrices.
   - still to be done: 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 should consider 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. for 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, in general, the form of the models $p(x|k, \vartheta_k)$ is not known.

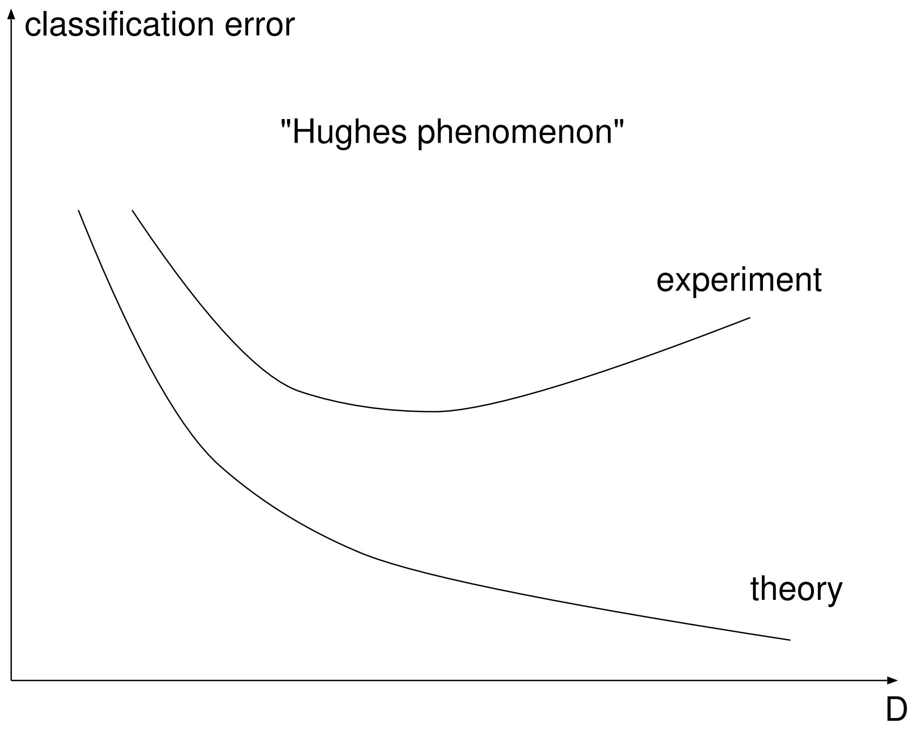
Hints:

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

  Example: On the training sample, the nearest neighbor classifier does not produce errors at all.
- Repeated tests on the same test data can sometimes be delusive because the system is 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 an unseen sample because the error rate should enable a prediction for new tests. In statistics, similar methods are known as *cross-validation.*

3.7 Dependency of the Error Rate on the Dimension

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, first, the error rate decreases when the number of feature components increases. From a certain point it starts to increase although, according to the theory, a dimension increase leads to an error rate reduction.

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

Theoretical considerations on the error rate: additional feature component: $x_1^D \in \mathbb{R}^D \rightarrow x_1^{D+1} \in \mathbb{R}^{D+1}$

Bayes error rate:

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

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

This results in

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

i.e., when the number of feature components increases, in theory, the error rate is reduced or, at least, kept unchanged. In particular, we have equality if 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 as random variable.
2. There is an *a priori* (or *prior*) distribution $p(\vartheta_k)$ for the parameter ϑ_k.

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 OCR or speech recognition are user-dependent. Such a system can be trained beforehand 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 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 with $n = 1, \ldots, N_k$.

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

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

Definition and calculation of the *a posteriori* distribution:

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

Only the numerator is directly dependent on ϑ_k. It is equal to the product of the *a priori* distribution and the likelihood.
The *a posteriori* distribution is often reduced to one single value. In particular, we use

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

- the posterior maximum: (also called *MAP estimate*, *maximum a posteriori estimate*)

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

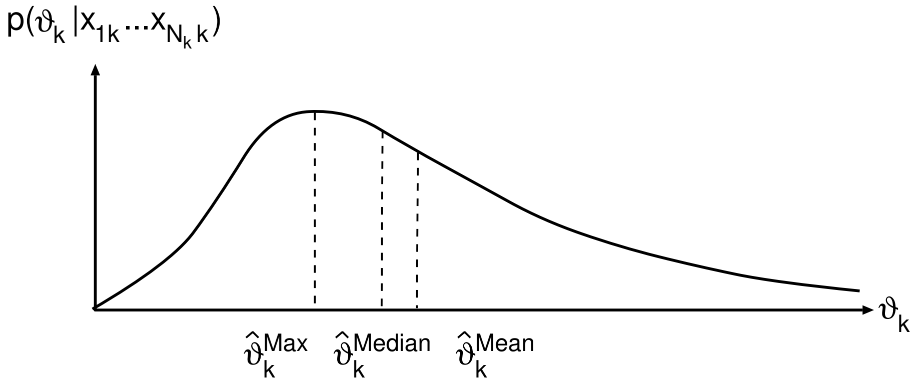
- the posterior median:

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

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

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

Illustration:



Special case: if $p(\vartheta_k) = const(\vartheta_k)$, we have

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

Example: univariate Gaussian distribution
To simplify matters, the class index k is omitted.
Assumptions:

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

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

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

Equation $(*)$ shows that the hyperparameters (μ_0, σ_0^2) determine the *a posteriori* distribution of μ, similar to the pairs (x_n, 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:

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

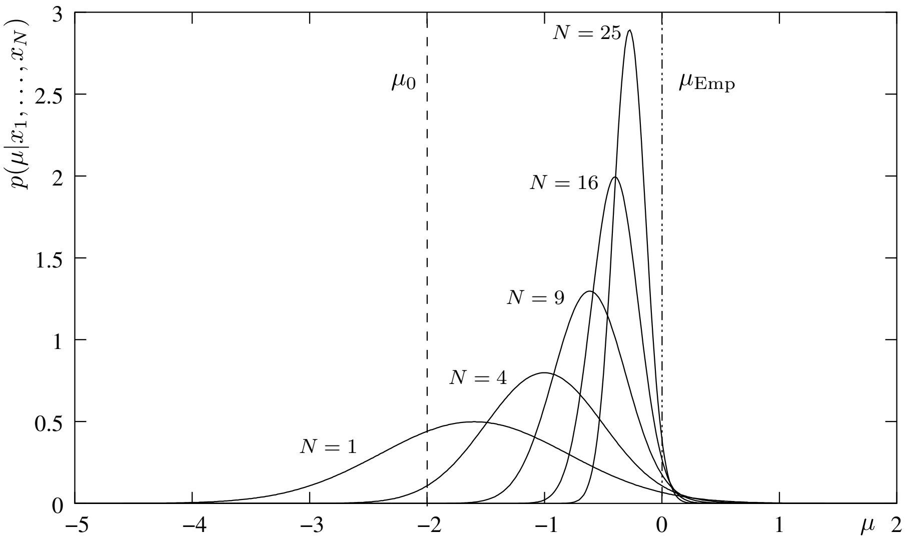
This expression describes an averaging between

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

For $N \to \infty$, the 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:

$$\begin{tikzcd}[row sep=large] & p(\mu|\mu_N,\sigma_N^2) \arrow[d] & \mu_N \to \frac{1}{N}\sum_{n=1}^N x_n \\ N \to \infty & & \\ & \text{delta function} & \sigma_N^2 \to 0 \end{tikzcd}$$

Graphical representation: if N increases, the *a priori* distribution concentrates 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;
- both μ and Σ are unknown at the same time;
- in general, the covariance matrix is not diagonal.

Details can be found in:

- [Fukunaga 90], p.389-394
- [Keehn 65]

In this case and in similar ones, the new estimation is obtained (exactly 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 method utilized in recognition?
Two methods:

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

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

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

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

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

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

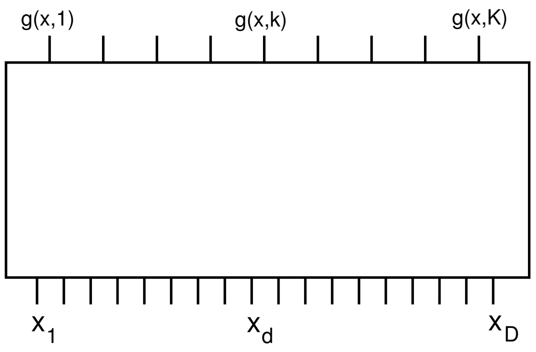
   In practice, closed solutions for the integrals which occur in this method exist only in exceptional cases. Details can be found in the already mentioned books: [Duda & Hart 73], p.55-59, [Keehn 65] and [Ney 99b].

4 Discriminants and Neural Networks

4.1 Squared Error Criterion

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

Remember the discriminant approach



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

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

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

These parameters are determined during the learning phase.

In this phase, the ideal output values are provided:

$$\begin{aligned} g(x, k) &\overset{!}{=} 1, && \text{if } x \text{ belongs to class } k, \\ g(x, k) &\overset{!}{=} 0, && \text{otherwise.} \end{aligned}$$

Normally, these ideal values can only be approximated. During the learning phase, usually, the squared error criterion is used for the 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:

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

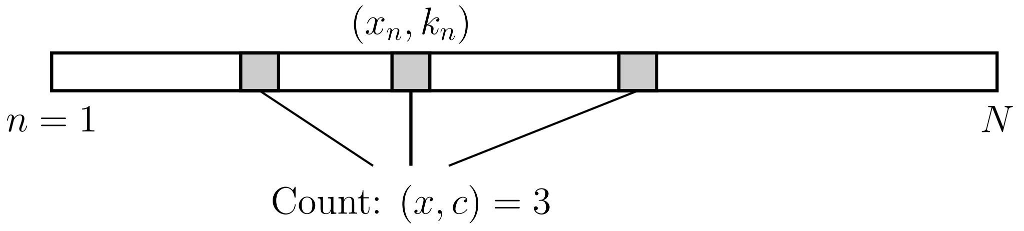
Instead of one single training vector, there is a whole set:

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

In the following, *pr* denotes the *empirical distribution*, i.e. relative frequency in the training data. For the joint distribution of observation x and class k, it is defined as:

$$\begin{aligned} pr(x,k) &:= \frac{1}{N} \sum_{\substack{n:x_n=x \\ k_n=k}} 1 \\ &= \frac{1}{N} \sum_n \delta(x,x_n) \cdot \delta(k,k_n) \end{aligned}$$

The figure below illustrates the counting observation-class pairs for calculating empirical distributions:



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

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

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

The notation $E_N(g)$ is to 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 $n = 1,\ldots,N$.

$$\begin{aligned}
E(g) &= \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{K}\left[g(x_n,c)-\delta(k_n,c)\right]^2 \\
&= \frac{1}{N}\sum_{x,k}\sum_{\substack{n:x_n=x\\k_n=k}}\sum_{c=1}^{K}\left[g(x,c)-\delta(k,c)\right]^2 \\
&= \sum_{x,k} pr(x,k)\sum_{c=1}^{K}\left[g(x,c)-\delta(k,c)\right]^2 \\
&= \sum_{x} pr(x)\cdot \underbrace{\sum_{k=1}^{K} pr(k|x)\cdot\sum_{c=1}^{K}\left[g(x,c)-\delta(k,c)\right]^2}_{=:\,e(x)}
\end{aligned}$$

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

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

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

- $1 - \sum\limits_{c=1}^{K} pr^2(c|x)$: independent of the discriminant function $g(x,k)$ (it also appears in the context of the Gini criterion or the nearest neighbor error)
  Upper limit for the Bayes error:

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

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

Consequently, the expression for $E(g)$ becomes

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

Remarks with respect to $E(g)$:

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

  For the global optimum g_0, we have

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

  i.e., as error criterion, we never obtain smaller values.

- To approximate the global optimum, the discriminant function should be suitably chosen with regard to
  - the functional form (*structure*) and
  - the number of free parameters ϑ.

- If the equality $g(x, c) = pr(c|x)$ is not achievable for each (x, c), the error will be minimized primarily at points x at which $pr(x)$ has a large value.
- Finding the (global) optimum is a difficult mathematical problem. There are two popular, iterative methods:
  - the gradient method (numerical mathematics)
  - the error back propagation (*stochastic gradient*)
- In discriminant approaches, we usually do not deal with normalization issues.

4.2 Output Normalization and Training Criteria

We consider three types of normalization $q(x, c)$ for the output of discriminant function $g(x, c)$ and the associated training criteria.

- Squared error
  This criterion has already been studied in the previous section. Here we have
  $$q(x, c) \equiv g(x, c) \in \mathbb{R}$$
  which means the output is not normalized explicitly. The optimal discriminant function $\hat{q}(x, c)$ for minimizing the squared error is given by the empirical class posterior probability $pr(c|x)$.
- Binary divergence
  The output is normalized using the *sigmoid* function:
  $$\begin{aligned} q(x, c) &= \frac{e^{g(x,c)}}{1 + e^{g(x,c)}} \\ &= \frac{1}{1 + e^{-g(x,c)}} \in [0, 1] \end{aligned}$$
  The training criterion is defined as
  $$\begin{aligned} E &= \frac{1}{N}\sum_{n=1}^{N}\Big(\log q(x_n, c_n) + \sum_{c \neq c_n} \log[1 - q(x_n, c)]\Big) \\ &= \ldots\ (*) \\ &= \sum_x\sum_c\Big(pr(c|x) \log q(x, c) + [1 - pr(c|x)] \log[1 - q(x, c)]\Big) \end{aligned}$$

Since the output should be large for the correct class and small for the incorrect classes the criterion needs to be maximized.

Exercise: Derive the missing steps for $(*)$ and show that $\hat{q}(x,c) = pr(c|x)$ for the binary divergence criterion.

- Cross-entropy
  The output is normalized using the *softmax* function:

$$q(c|x) = \frac{e^{g(x,c)}}{\sum\limits_{\tilde{c}} e^{g(x,\tilde{c})}}$$

This defines a probability distribution $q(c|x)$ over the classes c, where $\sum\limits_{c} q(c|x) = 1$. The respective criterion is

$$\begin{aligned} E &= \frac{1}{N}\sum_n \log q(c_n|x_n) \\ &= \sum_x pr(x) \sum_c pr(c|x) \log q(c|x) \end{aligned}$$

which needs to be maximized during training. Using the divergence inequality

$$\sum_u p_u \log q_u \leq \sum_u p_u \log p_u$$

where p and q are probability distributions defined over u, the optimal result is given by $\hat{q}(c|x) = pr(c|x)$.

4.3 Structures and Multilayer Perceptron

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

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

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

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

These structures can also be defined as higher polynoms of x. One obtains the *polynomial classifiers* or *regression classifiers* (*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 data not seen in training.

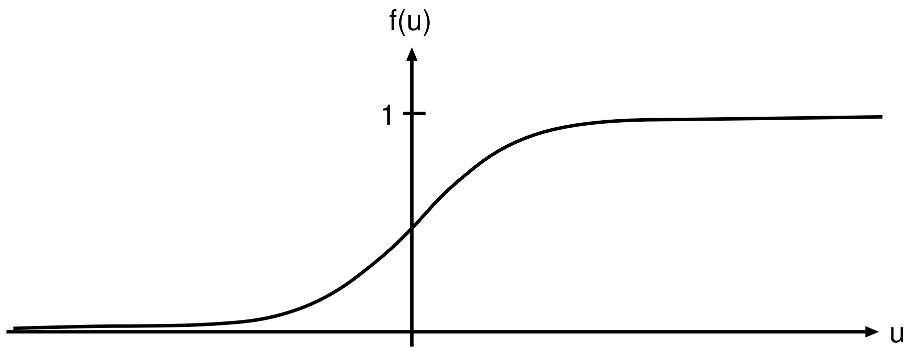
The disadvantages are to be avoided using the *multilayer perceptron*:

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*):

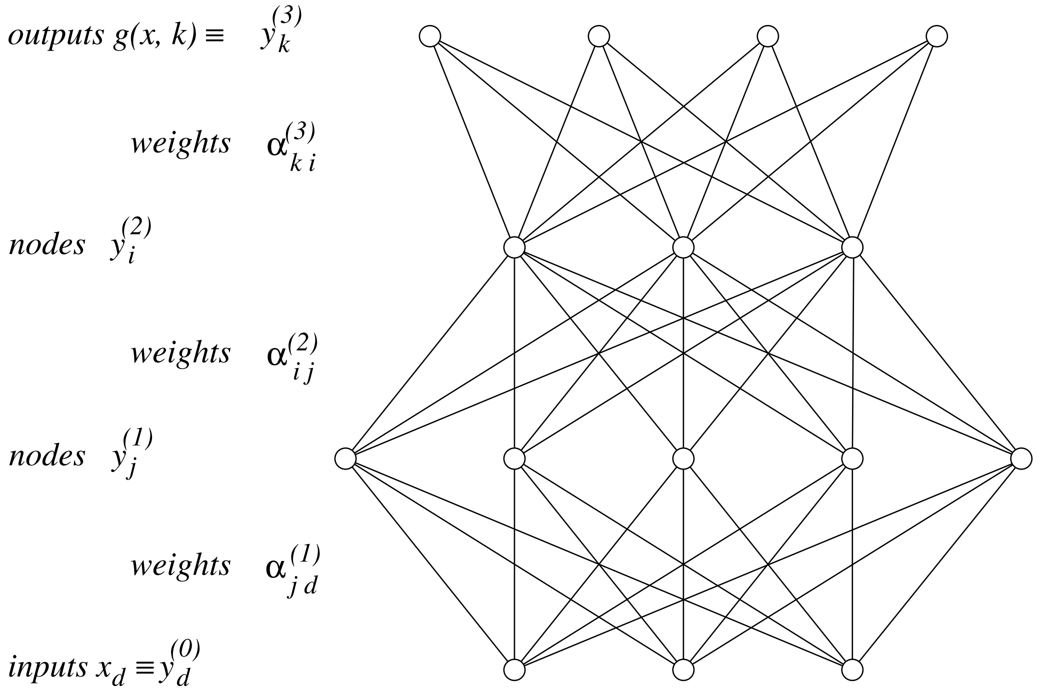
  $$\sum_j \alpha_{ij}^{(l)} \cdot y_j^{(l-1)}$$

  Each such linear combination passes 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):


outputs $g(x, k) \equiv y_k^{(3)}$
weights $\alpha_{ki}^{(3)}$
nodes $y_i^{(2)}$
weights $\alpha_{ij}^{(2)}$
nodes $y_j^{(1)}$
weights $\alpha_{jd}^{(1)}$
inputs $x_d \equiv y_d^{(0)}$

$$\text{outputs (layer 3)} \quad : \quad y_k^{(3)} = f\left(\sum_i \alpha_{ki}^{(3)} \cdot y_i^{(2)}\right)$$

$$\text{node layer 2} \quad : \quad y_i^{(2)} = f\left(\sum_j \alpha_{ij}^{(2)} \cdot y_j^{(1)}\right)$$

$$\text{node layer 1} \quad : \quad y_j^{(1)} = f\left(\sum_d \alpha_{jd}^{(1)} \cdot y_d^{(0)}\right)$$

$$\text{inputs (layer 0)} \quad : \quad y_d^{(0)} = x_d$$

Thus, the L-layer 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 input value $y_0^{(l)} \equiv 1$ is added in each layer, e.g.:

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

4.3.1 Non-Linearity

Usually, the function f is represented by the *sigmoid function*:

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

And its derivative is calculated as:

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

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

The arc tangent can also be used as alternative to the sigmoid function (polynomial decay of the derivative):

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

Other variants are possible as well, but they 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. Applying the series expansion

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

to the sigmoid function (expansion about the point $u_0 = 0$), we obtain

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

Interpretation: The sigmoid function generates the non-linear terms in a *specific* manner.

4.3.2 Remarks about the Multilayer Perceptron (MLP)

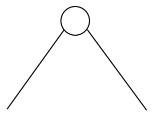
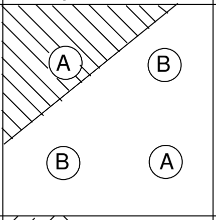
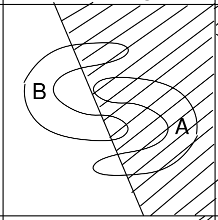
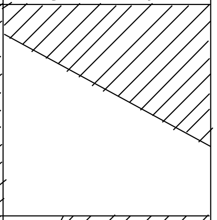
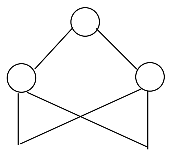
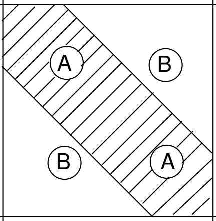
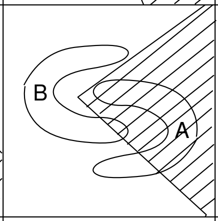
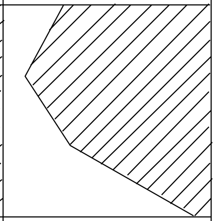
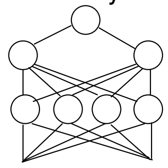
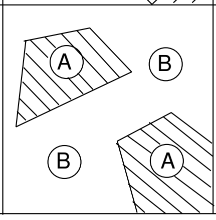
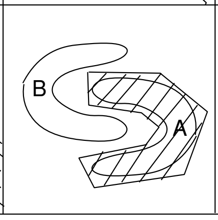
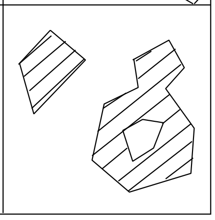
- The multilayer perceptron is the prototype of an *artificial* neural network. The attribute *artificial* is utilized to distinguish them from biological neural networks.
- The advantages of (artificial) neural networks are often summarized as follows:
  - similarity to the biological example,
  - massive parallelism ("pdp" $\hat{=}$ parallel distributed processing),
  - discriminative training,
  - approximation of arbitrary input-output dependencies.

- Many non-neural approaches also have the same advantages, but the idea of the discriminative training was revived by the neural networks, especially in speech recognition.

Structure of the multilayer perceptron: Basically, for the choice of

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

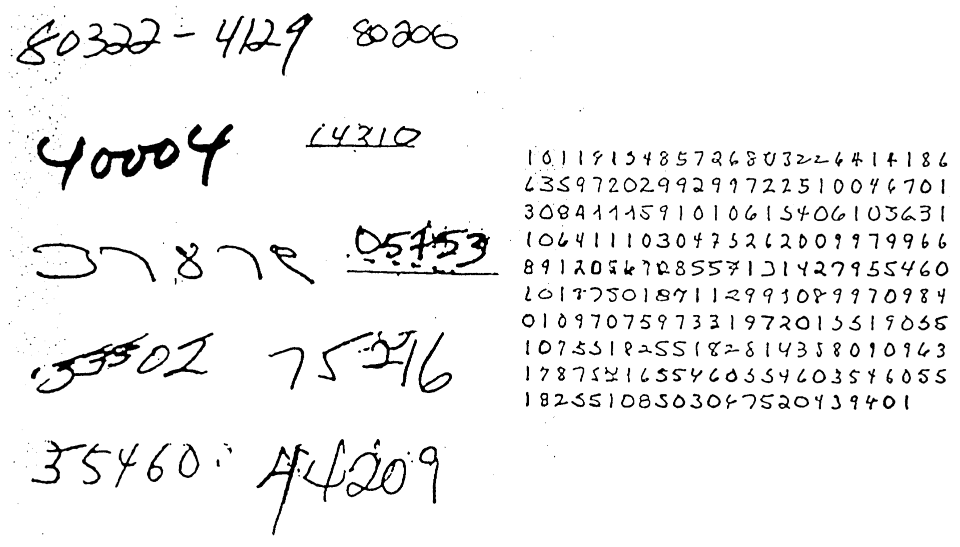
there are only empirical methods.
Formal inspection of the number of layers (1987)

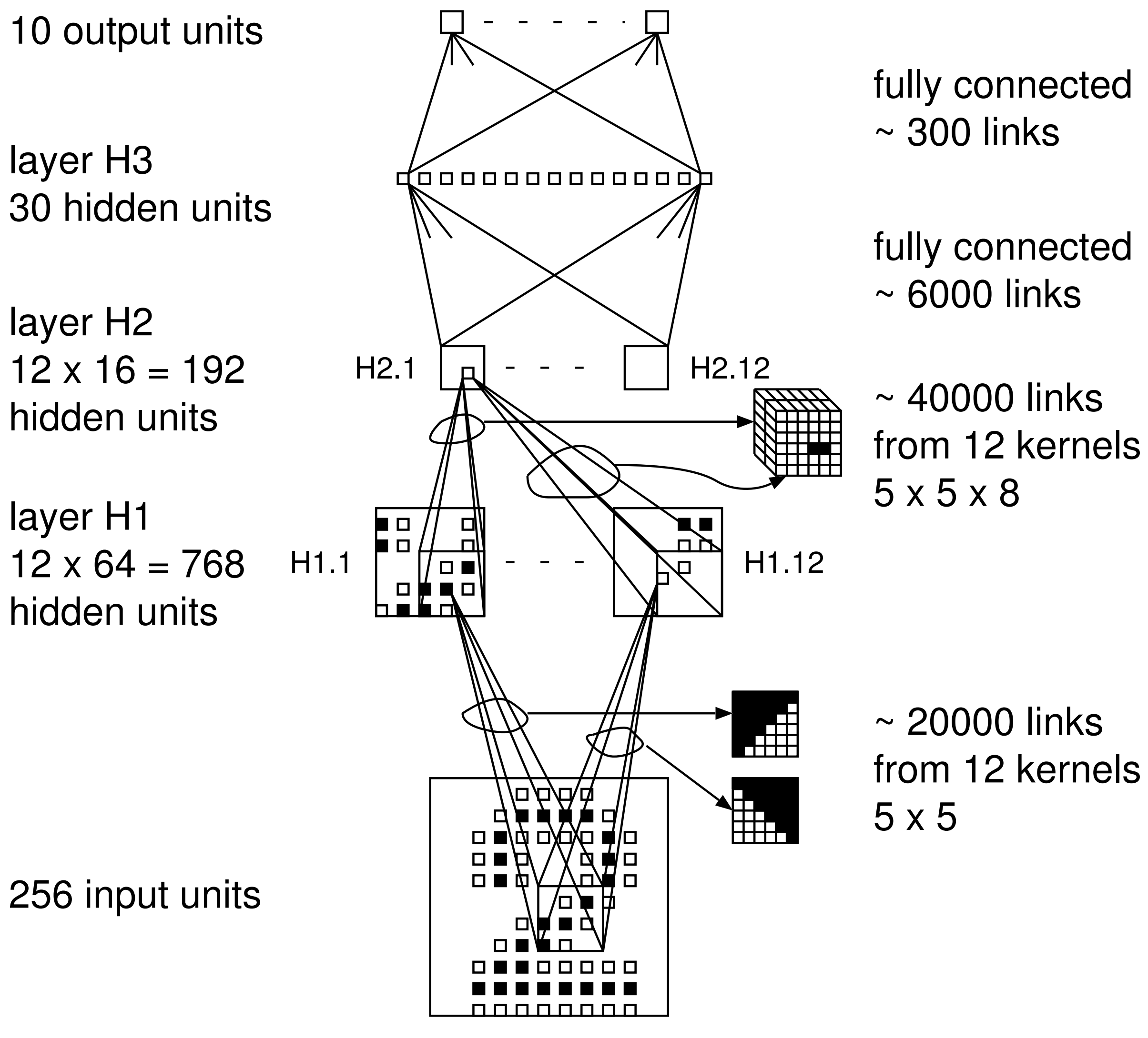
structure	type of decision regions	exclusive OR problem	classes with meshed regions	most general region shapes
single layer 	half plane bounded by hyperplane			
two layers 	convex open or closed regions			
three layers 	arbitrary (complexity limited by number of nodes			

Note: In general, there are more than 2 classes. Therefore, the class boundaries become more complex.

It is useful to prescribe symmetric dependencies between the weights if they result from the problem definition.

Example: multilayer perceptron for character recognition by Yan le Cun.




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

Construction of the multilayer perceptron:

Input layer:	16×16 pixels (gray values)
Layer H1:	consists of 12 planes (H1.1–H1.12). Each plane has an 8×8 square of nodes (= 64 hidden units), and all nodes (also called cells) in a plane share the same weights. Each node sees an image part of 5×5 pixels. Neighboring nodes see 5×5 grids shifted by 2 pixels. (therefore: 8 nodes, since $8 \cdot 2 = 16$ (image height/width)) Parameters per plane: 25 weights (independent of the node) 64 biases (node dependent offsets) Total: $12 \times (25 + 64) = 1068$ free parameters

Idea: each of the planes of H1 learns to be a grid of (identical) feature detectors.

Layer H2:	also consists of 12 planes (H2.1–H2.12). Each plane has 4×4 nodes (= 16 hidden units). Each H2-node looks at only 8 of the 12 H1-planes and sees a 5×5 neighborhood in each of those 8 planes. Parameters per plane: $8 \cdot 5 \cdot 5 = 200$ weights (node independent) $4 \cdot 4 = 16$ biases (node dependent) Total: $12 \times (200 + 16) = 2592$ free parameters

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

Layer H3:	consists of 30 nodes, each of them sees all $12 \cdot 16 = 192$ nodes of H2.
Output layer:	consists of 10 nodes, each of them sees all 30 nodes 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 how structures are included in the multilayer perceptron.

4.3.3 Structures of Statistical Classifiers Similar to the Multilayer Perceptron

In addition to neural networks, there are approaches which lead to similar structures for the *a posteriori* probability $p(k|x)$.
From Chapter 2, we know that

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

Note: $p_\vartheta(k,x)$ is only a model distribution – it must not be mistaken for the distribution $p(k,x)$.

If the difference $[\log p_\vartheta(x|c)-\log p_\vartheta(x|k)]$ is a linear function of x, then we obtain operations which are similar to those within a multilayer perceptron node. This can be achieved by

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

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

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

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

4.3.4 Equivalence of log-linear model and posterior Gaussian

From posterior Gaussian to log-linear model

We start with the transformation from posterior Gaussian to log-linear model. Given Gaussian model for $x \in \mathbb{R}^D$ and class c:

$$\begin{aligned}
p_\Theta(c|x) &= \frac{1}{p_\Theta(x)} \cdot p_\Theta(c)\, \mathcal{N}(x|\mu_c, \Sigma_c) \\
&= \frac{1}{p_\Theta(x)} \cdot \frac{p_\Theta(c)}{\sqrt{\det(2\pi\Sigma_c)}} \exp\left(-\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1}(x-\mu_c)\right) \\
&= \frac{1}{p_\Theta(x)} \cdot \exp\Bigg(\log p_\Theta(c) - \frac{1}{2}\log\det(2\pi\Sigma_c) - \frac{1}{2}\mu_c^T \Sigma_c^{-1}\mu_c \\
&\quad + \mu_c^T \Sigma_c^{-1} x - \frac{1}{2} x^T \Sigma_c^{-1} x\Bigg) \\
&= \frac{1}{p_\Theta(x)} \cdot \exp\left(\alpha_c + \lambda_c^T x + x^T \Lambda_c x\right)
\end{aligned}$$

with the parameters

$$\Theta := \left\{\alpha_c \in \mathbb{R}, \lambda_c \in \mathbb{R}^D, \Lambda_c \in \mathbb{R}^{D\times D}\right\}$$

This corresponds to a log-linear model (log-linear in parameters, log-quadratic in observations):

$$p_\Theta(c|x) = \frac{\exp\left(\alpha_c + \lambda_c^T x + x^T \Lambda_c x\right)}{\sum_{c'} \exp\left(\alpha_{c'} + \lambda_{c'}^T x + x^T \Lambda_{c'} x\right)}$$

From log-linear model to posterior Gaussian

Log-linear models are invariant under additive transformations:

$$\begin{aligned}
\alpha_c &\to \alpha_c + \alpha_0 \in \mathbb{R} \\
\lambda_c &\to \lambda_c + \lambda_0 \in \mathbb{R} \\
\Lambda_c &\to \Lambda_c + \Lambda_0 \in \mathbb{R}
\end{aligned}$$

These invariances can be used to transform the log-linear model back to a Gaussian model such that the constraints of a Gaussian model are satisfied ($p(c)$ is normalized, Σ_c is positive definite and invertible).

Note: When going from a generative Gaussian model to its posterior form the parameters are no longer unique.
As a consequence of the equivalence the discriminative training criterion for Gaussian models defines a convex optimization criterion:

$$\underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_r \log p_\vartheta(c_n|x_r) \right\}$$

4.4 Error Back Propagation

Error back propagation is the most widely used method for the training of a multilayer perceptron. It was suggested by Rumelhart, Hinton and Williams in the 1980s. However, several publications of the 1960s and 1970s already dealt with this subject: [Amari 67], [Werbos 74], [Parker 82]. Shortly summarized:

error back propagation = stochastic gradient and chain rule

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

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

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

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

Case distinction:

a) Output layer L: In order to compute $\frac{\partial E_n}{\partial \alpha_{ki}^{(L)}}$ we need to consider the following chain of dependencies:

$$\alpha_{ki}^{(L)} \longrightarrow z_k^{(L)} \longrightarrow y_k^{(L)} \longrightarrow E_n(\alpha_{ki}^{(L)})$$

This leads to:

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

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

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

At first, we consider the layer $L-1$. Here we need to consider the following chain of dependencies:

$$\alpha_{ij}^{(L-1)} \longrightarrow z_i^{(L-1)} \longrightarrow y_i^{(L-1)} \longrightarrow E_n(\alpha_{ij}^{(L-1)})$$

This leads to:

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

Determining the partial derivative $\partial E_n/\partial y_j^{(L-1)}$ 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:

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

I.e., if there is a change in $y_i^{(L-1)}$ (in the inner node i of the hidden layer $L-1$), all outputs $y_k^{(L)}$ are changed. Thus, we derive

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

The derivatives of the local error E_n to outputs in deeper nodes yield by further recursive applications of the chain rule, which is illustrated in the following example:

$$
\begin{array}{lll}
\textit{Layers} & E_n & \\
 & \uparrow & \\
L & y^{(L)}_{1\ldots K} & \dfrac{\partial E_n}{\partial y^{(L)}_k} \\
 & \uparrow & \qquad\qquad\searrow \\
L-1 & y^{(L)}_{1\ldots I} & \dfrac{\partial E_n}{\partial y^{(L-1)}_i} = \sum_k \dfrac{\partial E_n}{\partial y^{(L)}_k} \cdot \dfrac{\partial y^{(L)}_k}{\partial y^{(L-1)}_i} \\
 & & \qquad\qquad\searrow \\
L-2 & & \dfrac{\partial E_n}{\partial y^{(L-2)}_j} = \sum_i \dfrac{\partial E_n}{\partial y^{(L-1)}_i} \cdot \dfrac{\partial y^{(L-1)}_i}{\partial y^{(L-2)}_i} \\
\vdots & \vdots & \quad\vdots \\
 & \uparrow & \\
1 & y^{(1)}_{1\ldots H} & \dfrac{\partial E_n}{\partial y^{(1)}_h} \\
 & \uparrow & \qquad\qquad\searrow \\
0 & y^{(0)}_{1\ldots D} & \dfrac{\partial E_n}{\partial y^{(0)}_d} = \sum_h \dfrac{\partial E_n}{\partial y^{(1)}_h} \cdot \dfrac{\partial y^{(1)}_h}{\partial y^{(0)}_d}
\end{array}
$$

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

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

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

The practical implementation of the error back propagation requires suitable heuristics regarding:

- the choice of initial values,
- the choice of the learning rate (usually decreasing when the number of iteration increases),
- the avoidance of poor local optima,
- the control of the convergence speed.

Altogether, the realization and implementation require considerable care.

4.5 Discriminative Training for Statistical Classifiers

Discriminative training denotes criteria or methods which try to use the 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 the difference in comparison to the maximum likelihood:

$$\begin{array}{lll} \underset{\vartheta}{\operatorname{argmax}} \sum_n \log p(k_n|x_n,\vartheta) & & \textit{maximum class posterior probability} \\ & \hat{=} & \textit{maximum mutual information, MMI} \\ & & \\ \underset{\vartheta}{\operatorname{argmax}} \sum_n \log p(x_n|k_n,\vartheta) & & \textit{maximum likelihood} \end{array}$$

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

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

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.

4.6 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):
  $p(r_x|x)$ with $p(k|x)$ *true distribution*
- local error rate for decision rule r:

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

- Bayes error rate:

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

- global error rate for decision rule r:

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

maximum mutual information

Estimation of the Bayes error rate for class log posterior probabilities (cf. *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:**

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

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

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

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

(cf. 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):

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

The training criterion yields the upper limit for the Bayes error rate.

4.7 Sequence Processing: Recurrent Neural Networks (RNN)

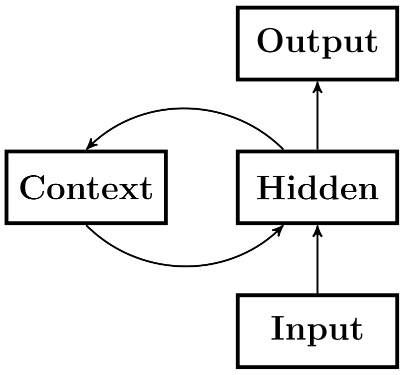
So far (output,input)-pairs (c, x) have been handled in isolation without internal structures in c or x. In this chapter we consider output and input sequences synchronized over time t

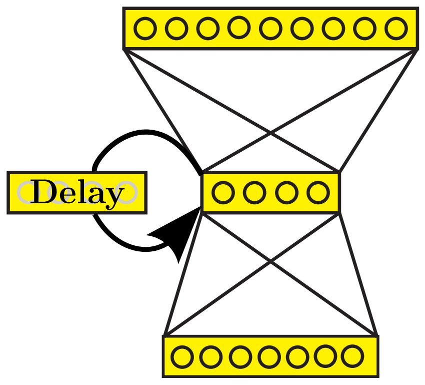
$$(c_t, x_t), \quad t = 1, \dots, T$$

where x_t are input vectors and c_t the class labels. The goal is to model the conditional probability $p(c_1^T|x_1^T)$ of the string c_1^T (assuming causality and a special start symbol c_0):

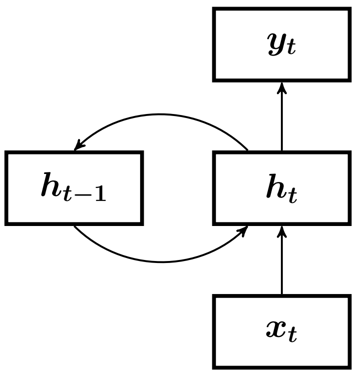
$$p(c_1^T|x_1^T) = \prod_{t=1}^{T} p(c_t|c_0^{t-1}, x_1^t)$$

with an output vector $y_t = p(c|\cdot)$ at each point in time t. This is realized using *recurrent neural networks* where a memory (or context) component is introduced to keep track of history:

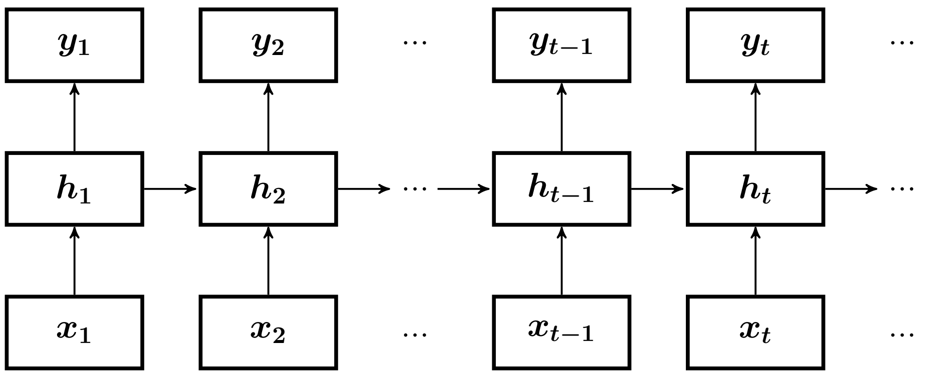




As a result, there are two types of input at time t, the observation x_t and the memory h_{t-1}:



The architecture of an RNN can be unfolded over time, resulting in a feedforward network with a special deep structure:



The application of the backpropagation algorithm to this unfolded network is called *backpropagation through time.*
The recurrence can be expressed with the equations:

$$\begin{aligned} h_t &= H\big(h_{t-1}, x_t\big) \\ y_t &= Y\big(h_t\big) \end{aligned}$$

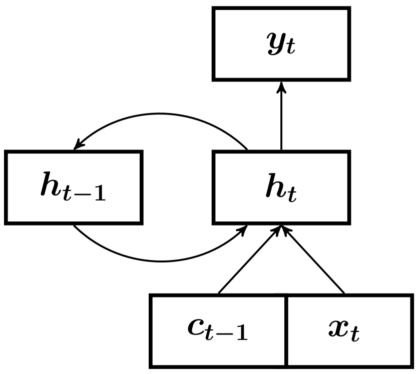
with time independent operations $H(\cdot)$ and $Y(\cdot)$.
Interpretation: The RNN models the conditional probability $p(c_t|x_1^t)$:

$$\begin{aligned} p\big(c_t|x_1^t\big) &= p\big(c_t|h_{t-1}, x_t\big) \\ &= p\big(c_t|h_t\big) \end{aligned}$$

with output vector $y_t = p(c_t|\cdot)$ at each time t.

4.7.1 Recurrent Neural Networks with additional input

At time t the preceding class output c_{t-1} can also be used as additional input, thus the input to the RNN are pairs (c_{t-1}, x_t):



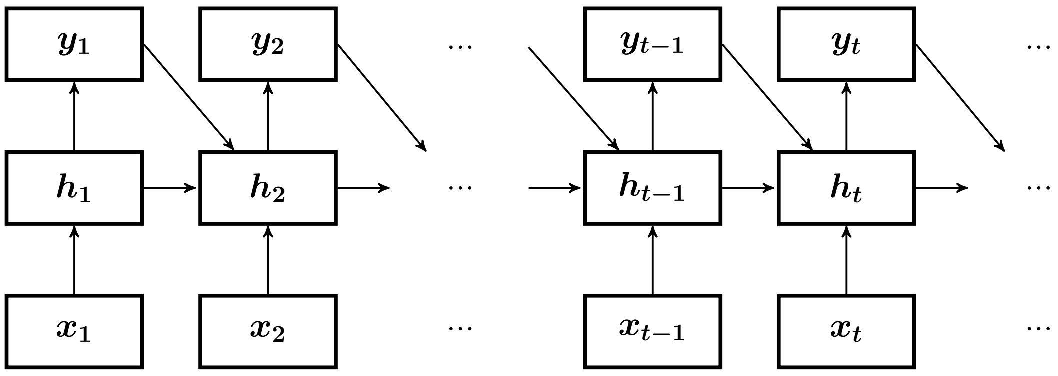
This leads to the recurrence equations:

$$\begin{aligned} h_t &= H\big(h_{t-1}, c_{t-1}, x_t\big) \\ y_t &= Y\big(h_t\big) \end{aligned}$$

with time independent operations $H(\cdot)$ and $Y(\cdot)$.
Interpretation: RNN models the conditional probability $p(c_t|c_0^{t-1}, x_1^t)$:

$$\begin{aligned} p\big(c_t|c_0^{t-1}, x_1^t\big) &= p\big(c_t|h_{t-1}, c_{t-1}, x_t\big) \\ &= p\big(c_t|h_t\big) \end{aligned}$$

with output vector $y_t = p(c_t|\cdot)$ at each time t.
Unfolding over time leads to the following deep structure:



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 distributions or similar ones.

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

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

The other extreme are model-free methods with the following properties:

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

A possible disadvantage is the increasing 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:

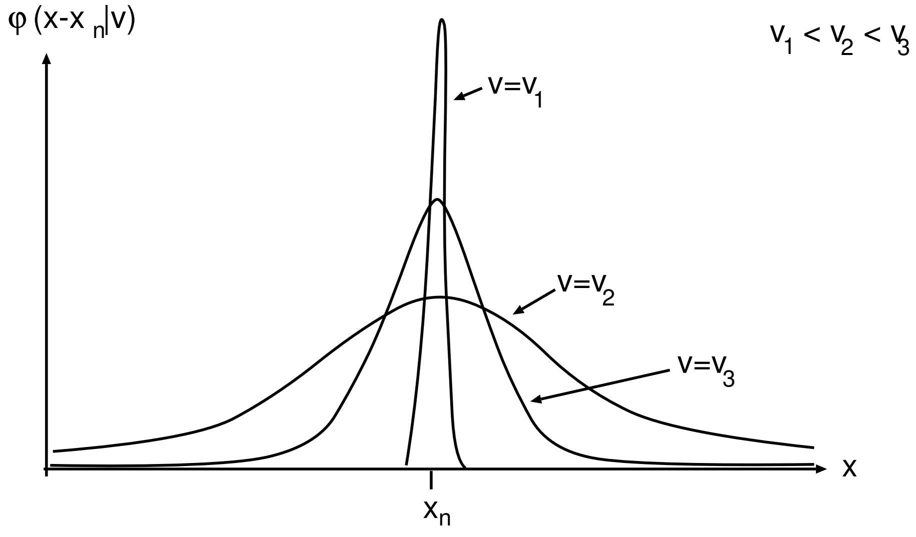
- *kernel densities* or *Parzen densities*;
- nearest neighbor rule for classification;
- discriminative methods, e.g. polynomial classifiers or multilayer perceptron (Chapter 4);
- mixture distributions (Chapter 6);
- Classification And Regression Trees (CART);
- Support Vector Machines (SVM).

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 kind of histogram approximation in $\mathbb{R}^D$ for the training data x_n with $n = 1, \ldots, N$.

To simplify matters, let $x_n \in \mathbb{R}$.



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

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

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

In general, the distribution φ can depend on the class k. In this case, we use the notation $\varphi_k(x)$. A 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}.

The distribution $p(x|k)$ is estimated by

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

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 is not directly applicable because the maximum is assumed at $v_{kd} \to 0$.

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

A solution of this problem is to introduce a class-dependent factor α_k and to multiply the common estimated variances with α_k:

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

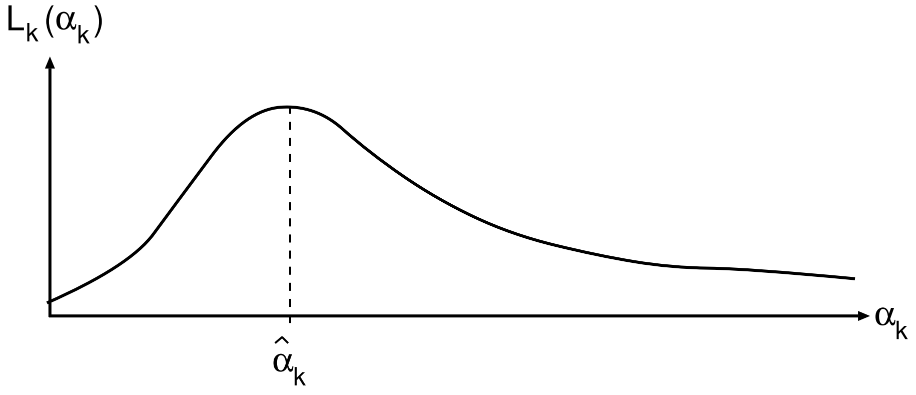
Now, the optimal value can be found by step-by-step increasing the factor α_k (in combination with leaving-one-out, assuming pairwise different training data):

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

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

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:
As for the 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 radial symmetry (to simplify matters, without variances):

- 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. Often, the Coulomb potential (from the electrical engineering) is applied:

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

(difficult because of the normalization).

5.3 Nearest Neighbor Rule (NN rule)

- more precise: *nearest neighbor decision rule*; 1-NN in contrast to k-NN.
- It must not to be mistaken for the *minimum distance*, e.g. for Gaussian distributions (cf. Chapter 2.5, S. 36) – The latter considers the distance between the test vector and the mean of the training vectors. However, the 1-NN takes the distance between the test vector and each single training vector into account.

We derive the nearest 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 a suitable distance $||\;||$.
Besides, to simplify matters, we assume that the kernel density φ is class-independent.

Estimated value for $p(k)$:

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

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

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

This approximation is called nearest neighbor rule, because the class chosen for the vector x is the class of the nearest neighbor in the training data x_{nk}:

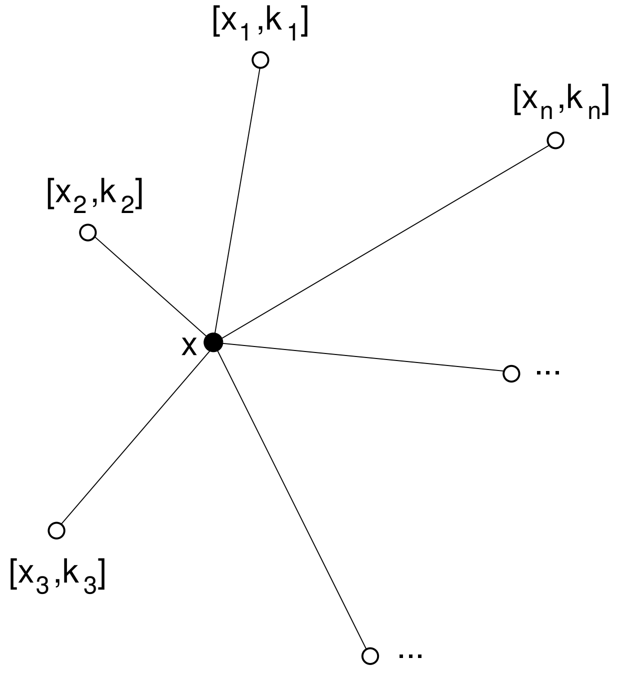
$$x \mapsto r(x) \;=\; \operatorname*{argmin}_k \{\min_{n=1,\ldots,N_k} ||x - x_{nk}||\}$$

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 affect the result.
- kernel densities generally yield better results

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

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



There are (successful and unsuccessful) approaches to reduce the computing time, i.e. the number of distance calculations:

1. early abort the summation over the dimensions (*partial distance*);
2. triangle inequality;
3. hierarchical structuring of the training vectors;
4. k-dimensional binary search trees (*k-d trees*); partitioning the search space into disjunct hypercuboids (cf. [Bentley 75], [Friedman & Bentley$^+$ 77])
5. reducing the number of training vectors, e.g. "editing": consider only vectors close to the class boundaries

5.4 Error Rate of the Nearest Neighbor Rule

More precise: error rate in the asymptotic limit case if the number N of the training data approaches infinity [Cover & Thomas 91].

Error rate for an 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

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

The above lines denote no actual derivation. Basically, Bayes' decision rule is simply plugged in for r_x in $p(r_x|x)$, arriving at $p(r_B(x)|x)$.

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 nearest neighbor (NN) rule, we define:

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

$$x_{NN} := \operatorname*{argmin}_{x_n : n=1,\ldots,N} \{||x - x_n||\},$$

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

Given the test vector x and its nearest neighbor x_{NN}, the local error rate $p_{NN}(e|x, x_{NN})$ of the NN rule yields

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

For $N \longrightarrow \infty$, i.e. with enough training data, we obtain

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

Hence, the local error rate is simplified to

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

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:

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

- 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 the random variable x, then we have:

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

and thus

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

Integrating both estimates over the whole space yields

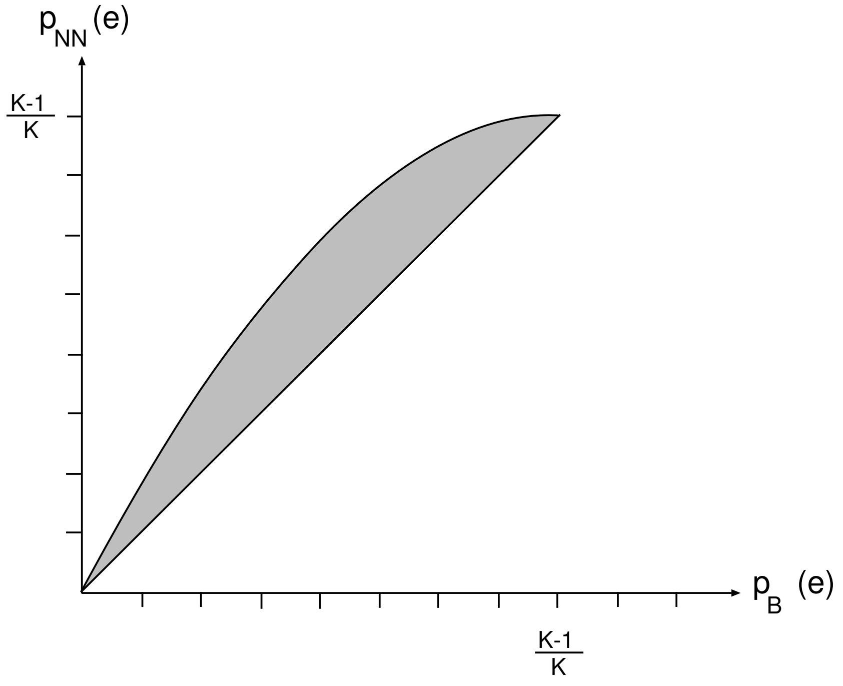
$$p_B(e) \leq \; p_{NN}(e) \; \leq 2 \cdot p_B(e) - \frac{K}{K-1} \cdot p_B^2(e) \leq 2 \cdot p_B(e) \; .$$

Exercises:

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

Independent of the nearest neighbor rule, the inequality for $p_{NN}(e)$ derived in Chapter 5.4 can be applied. Actually, $p_{NN}(e)$ is identical to the Gini criterion mentioned in Chapter 4.1. Thus, we obtain a connection between the recognition error and the squared error criterion for discriminants:

collection of the definitions:

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

(local squared error criterion at the output of a neural network or a discriminant, cf. Chapter 4.1, page 75)
Thus, for each point $x \in \mathbb{R}^D$, we have:

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

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

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

A corresponding inequality/equality can be defined for the global expectations:

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

6 Mixture Densities and Cluster Analysis

6.1 Mixture Densities

6.1.1 Introduction

So far, two extremes:

$$\begin{tikzcd}
\text{typical models (Gaussian, unimodal distributions)} \arrow[dd, leftrightarrow] & \\
{} & \text{mixture densities} \arrow[l] \\
\text{kernel densities (model-free)} &
\end{tikzcd}$$

Compromise:

- multimodal 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 with

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

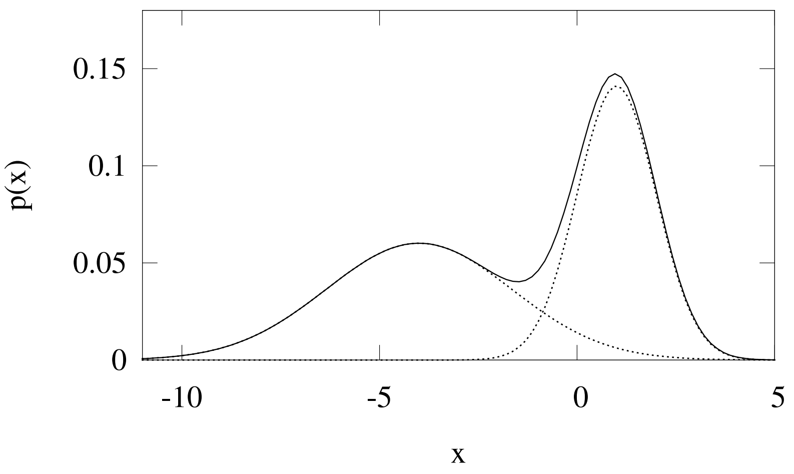
and basis distributions, single densities (*component densities*)

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

Single densities are typically unimodal, in most cases Gaussian distributions:

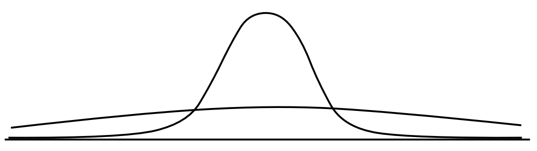
$$\vartheta_i = (\mu_i, \Sigma_i)$$

Example of a bimodal Gaussian distribution (the unimodal distributions are drawn as well; equal weights are used):



Similar applications are:

- the Tukey Model



purpose: an ϵ-probability for the non-central region

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

$\alpha \gg 1$: augmentation factor for the variance

- Interpolation for language models, e.g. bigram

  $N(v,w)$: Counts of the 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. $N(v,w) = 0$. Consequently, 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: $\sum\limits_w p(w|v) \; = \; (1-\lambda) \underbrace{\sum_w \frac{N(v,w)}{N(v)}}_{=1} \; + \; \lambda \underbrace{\sum_w p(w)}_{=1} \; = 1$

- Gaussian mixture densities

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

  Normalization:

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

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

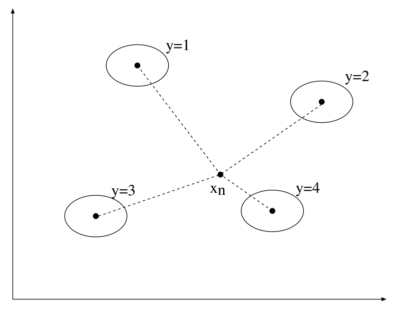
$$\operatorname*{argmax}_{\lambda} \prod_{n=1}^{N} p(x_n|\lambda) \quad \text{or} \quad \operatorname*{argmax}_{\lambda} \sum_{n=1}^{N} \log p(x_n|\lambda)$$

Thus, we obtain an unmanageable optimization 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):



A membership of an observation to a value of the hidden variable y is not known or is irrelevant, respectively. 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 $\tilde{\lambda}$:

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

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

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

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

Now, we define the auxiliary function (corresponds to the 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,

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

because of the divergence inequality of the information theory $(*)$.

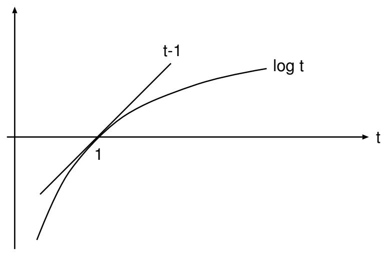
Proof of the divergence inequality:

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

It is enough to show:

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

For the logarithm, we have



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

Hence,

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

this leads to the divergence inequality.

EM iterative method:

```
Initial value $\lambda$
Iterations:
  1. "E-step" (expectation over $y$):                  calculate $Q(\lambda, \tilde{\lambda})$
  2. "M-step" (maximization over $\tilde{\lambda}$):   find $\underset{\tilde{\lambda}}{\operatorname{argmax}} \{Q(\lambda, \tilde{\lambda})\}$
     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 $\sum\limits_i \tilde{c}_i = 1$.
Abbreviated notation for parameter set: $\lambda \equiv (\{c_i\}, \{\vartheta_i\})$
Using the Lagrange multiplier for the constraint results in:

$$\begin{aligned} Q(\lambda, \tilde{\lambda}) &= \sum_n \sum_i p(i|x_n, \lambda) \cdot \log\left[\tilde{c}_i \cdot p(x_n|i, \tilde{\vartheta}_i)\right] - \gamma\Big[\sum_i \tilde{c}_i - 1\Big] \\ &= \sum_n \sum_i p(i|x_n, \lambda) \cdot \log \tilde{c}_i \\ &\quad + \sum_n \sum_i p(i|x_n, \lambda) \cdot \log p(x_n|i, \tilde{\vartheta}_i) \;-\; \gamma\Big[\sum_i \tilde{c}_i - 1\Big] \\ \text{with } \; p(i|x_n, \lambda) &= \frac{c_i \cdot p(x_n|i, \vartheta_i)}{\sum\limits_{i'} c_{i'} \cdot p(x_n|i', \vartheta_{i'})} \; . \end{aligned}$$

Taking derivatives to determine $\tilde{\lambda}$ as a function of λ:

$$\text{a)} \; \frac{\partial Q}{\partial \gamma} = \sum_i \tilde{c}_i - 1 \overset{!}{=} 0 \quad \Leftrightarrow \quad \sum_i \tilde{c}_i = 1 \text{ (constraint)}$$

b)

$$\frac{\partial Q}{\partial \tilde{c}_i} \;=\; \sum_n p(i|x_n,\lambda) \cdot \frac{1}{\tilde{c}_i} \;-\; \gamma \;\overset{!}{=}\; 0$$

Result:

$$\tilde{c}_i \;=\; \frac{1}{N}\sum_{n=1}^{N} p(i|x_n,\lambda) = \frac{1}{N}\sum_{n=1}^{N} \frac{c_i \cdot p(x_n|i,\vartheta_i)}{\sum\limits_{i'} c_{i'} \cdot p(x_n|i',\vartheta_{i'})}$$

c)

$$\frac{\partial Q}{\partial \tilde{\vartheta}_i} \;=\; \sum_{n=1}^{N} p(i|x_n,\lambda) \cdot \frac{\partial}{\partial \tilde{\vartheta}_i} \log p(x_n|i,\tilde{\vartheta}_i) \;\overset{!}{=}\; 0$$

in particular, Gaussian distributions: $p(x_n|i,\mu_i,\Sigma_i)$

with μ_i = mean vector
Σ_i = covariance matrix

$$\begin{aligned}
\frac{\partial Q}{\partial \tilde{\mu}_{id}} \;&=\; \sum_n p(i|x_n,\lambda) \cdot \sum_{d'} \Sigma^{-1}_{idd'}\,[\tilde{\mu}_{id'} - x_{nd'}] \\
&=\; \sum_{d'} \Sigma^{-1}_{idd'} \sum_n p(i|x_n,\lambda) \cdot [\tilde{\mu}_{id'} - x_{nd'}] \overset{!}{=} 0 \qquad \forall \quad d \\
\Leftrightarrow \quad \tilde{\mu}_{id} \;&=\; \frac{\sum\limits_n p(i|x_n,\lambda) \cdot x_{nd}}{\sum\limits_n p(i|x_n,\lambda)} \\
&=\; \sum_n \frac{p(i|x_n,\lambda)}{\sum\limits_{n'} p(i|x_{n'},\lambda)} \cdot x_{nd}
\end{aligned}$$

Rewriting into a vector equation with the weights $\gamma_i(n)$ results in

$$\tilde{\mu}_i = \sum_{n=1}^{N} \gamma_i(n) \cdot x_n \quad \text{where} \quad \gamma_i(n) = \frac{p(i|x_n,\lambda)}{\sum\limits_{n'} p(i|x_{n'},\lambda)}.$$

The weights $\gamma_i(n)$ determine the contribution of the observation x_n to the estimated value $\tilde{\mu}_i$.

Note: $\sum\limits_{n=1}^{N} \gamma_i(n) = 1$

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:
This is a *weighted* maximum likelihood estimation where the weights depend on $p(i|x_n, \lambda)$ and change during the iterations:

- μ_i : weighted empirical mean 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 $\mu_i, \Sigma_i, c_i$

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: $c_i = \tilde{c}_i$    $\mu_i = \tilde{\mu}_i$    $\Sigma_i = \tilde{\Sigma}_i$

        GOTO Loop
```

Exercises:

1. Write a C program for the example above.
2. Calculate the formulae of the EM algorithm for the Gaussian distribution with a *pooled covariance matrix*, i.e. $p(x_n|i, \mu_i, \Sigma)$ with Σ independent of i.

6.1.3 Mixture Densities for K Classes

case distinction:

a) no tying:

for each class k: weight $c_{ik} = p(i|k)$, $\sum_{i=1}^{I_k} p(i|k) = 1$

parameters of conditional distributions ϑ_{ik}

Model: $p(x|k) = \sum_{i=1}^{I_k} c_{ik} \cdot p(x|i, k, \vartheta_{ik})$

The training can be done separately for each class k.

b) *tied mixture densities*

Tying between models $p(x|k)$ by class-independent parameters ϑ_i.

Model: $p(x|k) = \sum_{i=1}^{I} c_{ik} \cdot p(x|i, \vartheta_i)$

i.e., the single densities $p(x|i, \vartheta_i)$ act as *basis distributions.*

Exercise:

Calculate the EM iteration formula for the above model, i.e. *tied mixture densities* for Gaussian distributions.

Approach:

$$
\begin{aligned}
Q(\lambda, \tilde{\lambda}) &= \sum_{k=1}^{K} \sum_{n=1}^{N_k} \sum_{i=1}^{I} p(i|x_{nk}, k, \lambda) \cdot \log \left[\tilde{c}_{ik} \cdot p(x_{nk}|i, \tilde{\vartheta}_i) \right] \\
\text{with} \quad p(i|x_{nk}, k, \lambda) &= \frac{c_{ik} \cdot p(x_{nk}|i, \vartheta_i)}{\sum_{i'=1}^{I} c_{i'k} \cdot p(x_{nk}|i', \vartheta_{i'})}
\end{aligned}
$$

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 by the 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. Therefore, we expect that one sum element dominates the others.

Training method:

```
        choose suitable initial values $\mu_i, \Sigma_i, c_i$

Loop:   determine the *best* single density for each observation $x_n$

                    $\operatorname{argmax}_i [c_i \cdot p(x_n|i,\mu_i,\Sigma_i)]$
        The 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 the best single density
              $\mu_i$ = mean of the observations $x_n$ assigned to each single density $i$
              $\Sigma_i$ = covariance matrix of the observations $x_n$ assigned to each single density $i$
          GOTO Loop
```

Interpretation: the maximum approximation results from a "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 (subgroups) 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 disjoint).

  Vectors $x_1, \ldots, x_N$ with label 0 or 1, i.e. 2 clusters

  $$\Longrightarrow \frac{2^N - 2}{2} = 2^{N-1} - 1 \qquad \text{partitions}$$
- 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 disjoint) vectors.

$$S(N, I) \;=\; \frac{1}{I!} \cdot \sum_{i=1}^{I} \binom{I}{i} (-1)^{I-i} \cdot i^N \cong \frac{1}{I!} I^N \,.$$

Notation: Stirling numbers of the second kind

Recursion:

$$\begin{array}{lll} S(N+1, I) & = S(N, I-1) & x_{N+1} \text{ defines the new } I^{\text{th}} \text{ cluster} \\ & \quad + \; I \cdot S(N, I) & x_{N+1} \text{ will be inserted in each} \\ & & \quad \text{of the } I \text{ old clusters} \end{array}$$

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:	$\{a, b, c\} \cup \{d\} \cup \{e\}$	$\binom{5}{3} = 10$
	$\{a, b\} \cup \{c, d\} \cup \{e\}$	$5 \cdot 3 = 15$

In practice:

- heuristic methods;
- iterative methods: improving an initial partition step-by-step until a local optimum is achieved (cf. EM algorithm).

6.2.2 Squared Error Criterion and Exchange Method

(often called variance criterion; minimum distance method)

Iteration methods for optimizing the squared error criterion:

nearest mean, K-means, ISODATA
(here: $K = I \hat{=}$ number of clusters)

Assumption: everything is reduced to one (Euclidean) distance measure (after suitably scaling the axes).

Let A_i be clusters with $i = 1, \ldots, I$, 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 the scores G_i and G_j changed after an exchange operation? (calculate for yourself, cf. [Duda & Hart 73], 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, online)

choose initial partition $A_1, \dots, A_I$ and calculate $G_i$ and $\mu_i$

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: $\mu_j, G_j, \mu_i, G_i$

      if ($G_{TOT}$ does not change in $N$ passes) STOP
      otherwise GOTO Loop
```

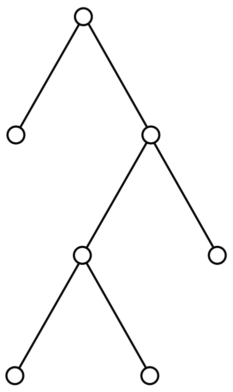
```
NEAREST MEAN (batch, offline)

choose initial partition $A_1, \dots, A_I$ and calculate $G_i$ and $\mu_i$

Loop: for each observation $x_n$ where $n = 1, \dots, N$ determine the
      nearest cluster $A_i$ and memorize the cluster index $i$

      for each cluster $A_i$ where $i = 1, \dots, I$ calculate the new
      mean $\mu_i$ from the observations $x_n$ assigned to this cluster

      if (no change) STOP
      otherwise GOTO Loop
```

Remarks:

1. Both algorithms can be transferred to other models than "mean" models.
2. Empirical experience: The batch variant is less sensitive regarding bad local minima.
3. Still open: how do we choose the initial partition?
   $\longrightarrow$ hierarchical clustering

6.2.3 Hierarchical Cluster Analysis

The hierarchy is modeled by means of a 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*)

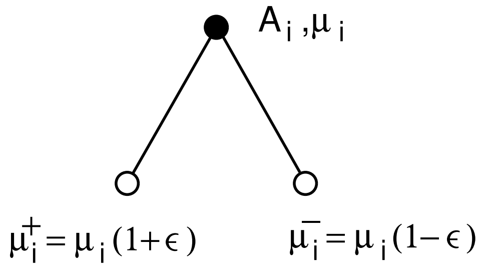
Top-down method: basic concept

```
Start: all data $x_1, \ldots, x_N$ form 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) until stop criterion is achieved

       check the quality of the total partitioning

       if "good" STOP,

       otherwise GOTO Loop
```

Problem:
There are $2^{m-1} - 1$ possibilities to split a cluster containing m elements into 2 new clusters. Due to the computing time, usually, it is not possible to evaluate all of them.

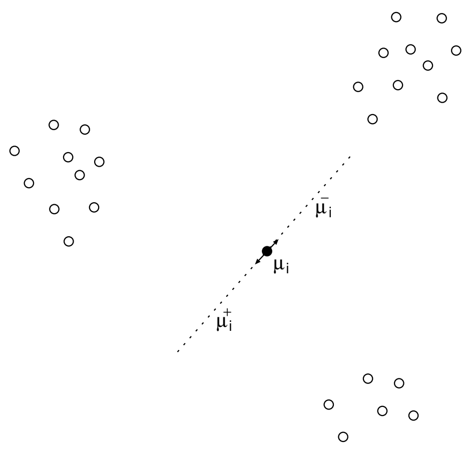
Heuristic for the "distance method":

1. *infinitesimal* perturbation of the mean μ_i



2. iterative training to improve μ_i^- and μ_i^+

   Example:



   Normally, the perturbation is performed into a standard direction because the direction is not critical.

   Speech processing: Linde, Buzo, Gray [Linde & Buzo+ 80] "LBG algorithm"

Note:

- The extension to mixture densities as an extension of distance models is obvious.
- It is suitable for vector spaces. An 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, $\dfrac{m(m-1)}{2}$ 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 $\Longrightarrow$ for $N \geq 10000$, the computing time becomes critical.

improved variant $(*)$:

Loop: find the cluster pair whose merging produces the minimum deterioration of the global criterion.

Standard methods:

- [Gordon 81], p.46
- [Spath 77], p.170
- [Duda & Hart 73], 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 (μ_i, μ_j: empirical means of A_i, A_j, respectively)

$$d(A_i, A_j) := ||\mu_i - \mu_j||$$

- 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, $N_i := |A_i|, N_j := |A_j|$)

$$d(A_i, A_j) := \frac{N_i \cdot N_j}{N_i + N_j} ||\mu_i - \mu_j||^2$$

(consistent with Gaussian model)

Exercise:

Assume Gaussian model
(note: 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"
- define preferably 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 the minimum
       deterioration of global criterion

       merge these clusters

       if (number of clusters > 1) GOTO Loop
```

Remarks:

1. Computational effort hardly increases in comparison to standard methods (at most by factor 2).
2. The "optimum" is achieved with each merging.
3. The Ward algorithm ("sum of squares") already has this optimality property (see [Duda & Hart 73], p.260, exercise 19).

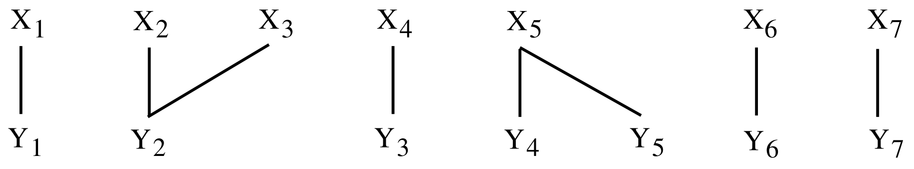
7 Stochastic Finite Automata

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:



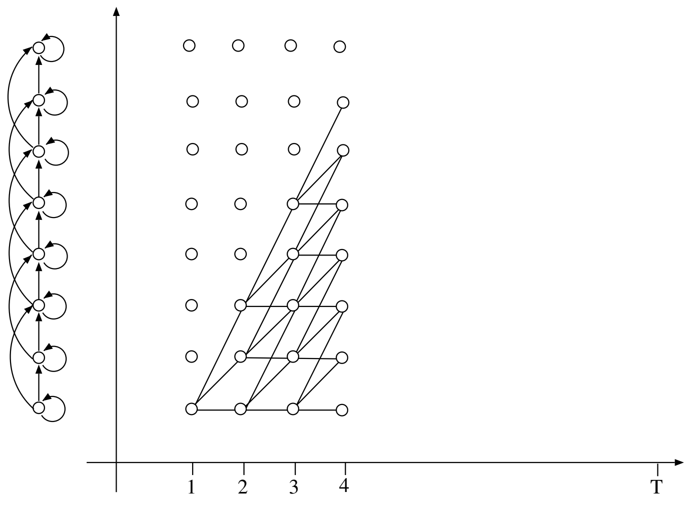
In other words:
The sequence as a whole cannot be described as an element of a suitable vector space, because a mapping of observations x_t to corresponding points on the coordinate axes is difficult.

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 sequences of continuous vectors (or discrete *noisy* symbols).
- *syntactic pattern recognition* (probability scores)

stochastic finite automaton
= stochastic regular grammar
= hidden Markov model (HMM)



- Also: *lattice / trellis*
- The path corresponds to the derivation of a regular grammar.
- Recombination of the different paths in one point

Estimating the number of possible paths for the standard $(0, 1, 2)$-model that is used in speech recognition (cf. the above figure):

Considering 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 finite states are allowed. If we allow only one finite state and assume the simplification that all finite 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 finite 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 the 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 a transition $\sigma \longrightarrow s$:

  $p(x_t|\sigma, s) =$ "usual model" (e.g. Gaussian distribution, mixture densities,...).

  Note the normalization: $\int p(x_t|\sigma, s) \;=\; 1$

- Specifying the initial and finite states.

Examples:

1. Speech:

   $x_t =$ energy distribution (over frequency axis), calculated every 10 ms by means of Fourier analysis.

   HMMs are extremely important:

   - extreme nonlinear variations of the speaking rate;
   - avoids local decisions (e.g. about the identity of one phoneme).

   similar: signal flow over time-axis
   (e.g. ECG, EEG, ...)

2. Character recognition:

   Approach:

   (a) time axis $\hat{=}$ horizontal axis
   (b) decomposition of the image by means of a window function
   (c) feature vector x_t for each window

3. Processing of “noisy” symbol chains

   - *Levenshtein* distance: deletions, insertions, substitutions
   - in particular: written natural language

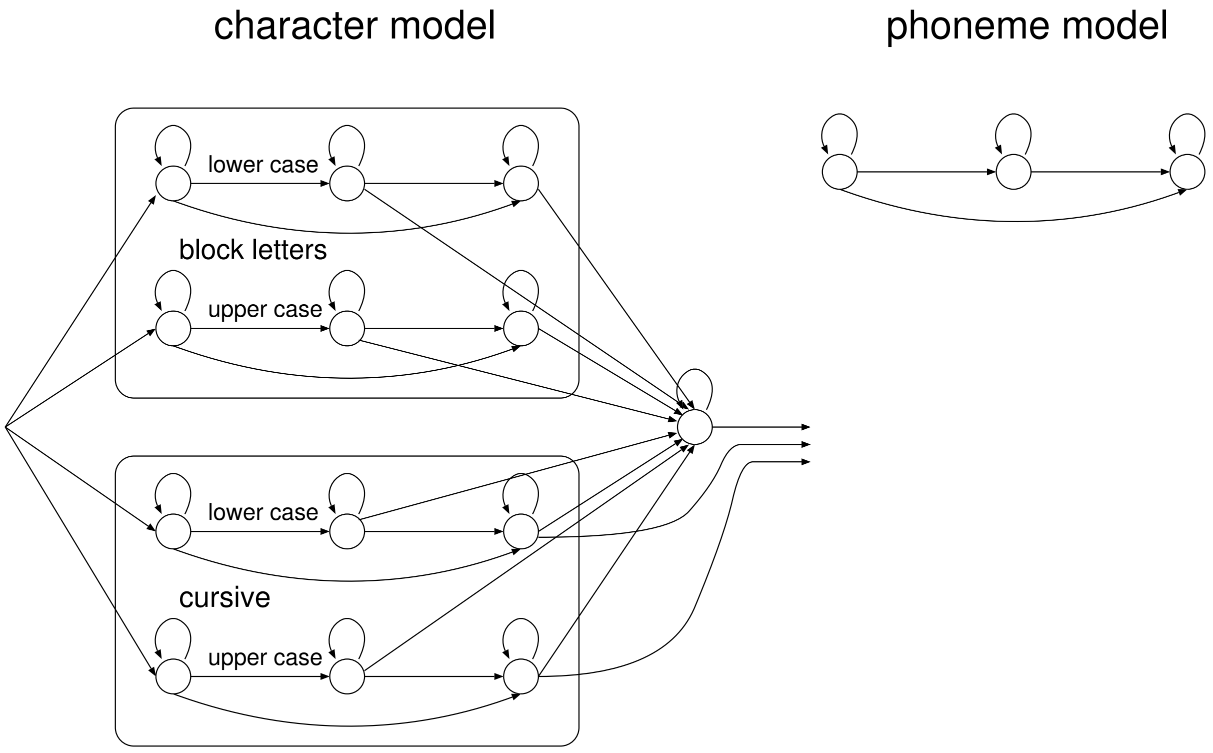


Figure: Hidden Markov models for character and speech recognition.

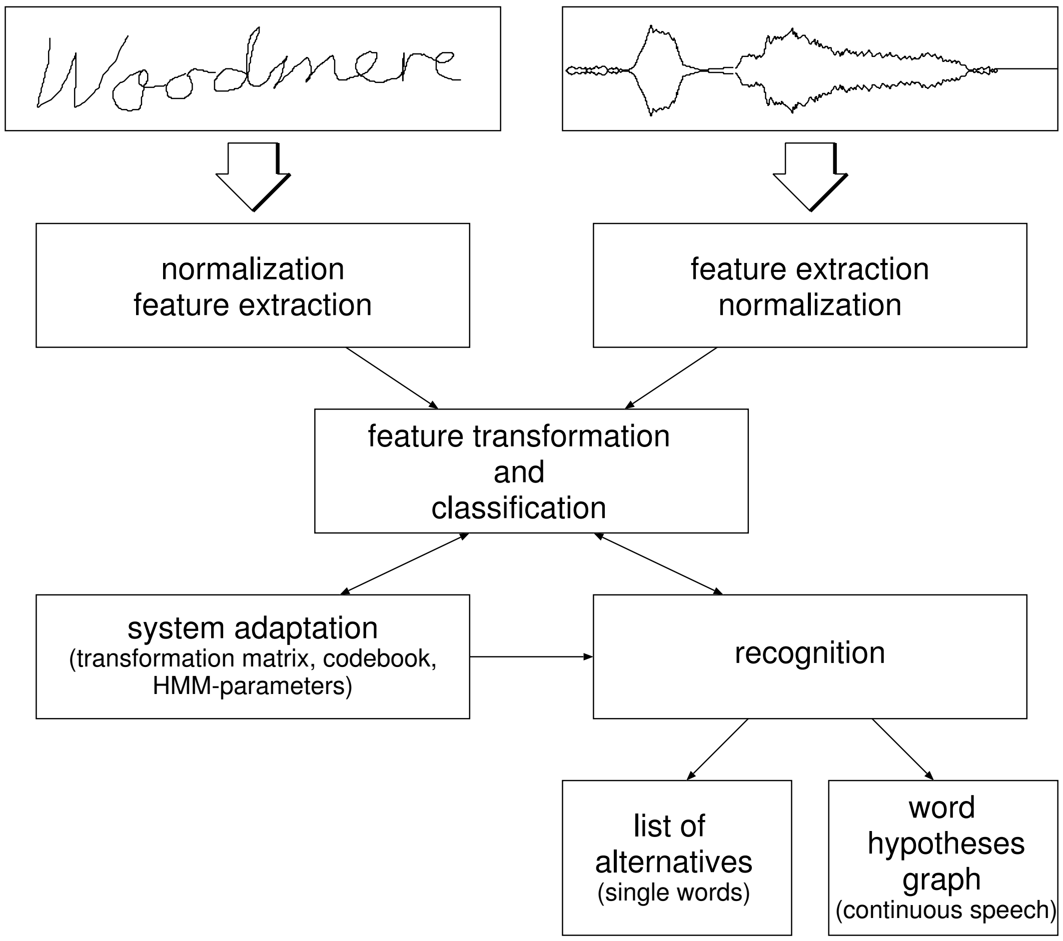


Figure: Block diagram of a recognition system with feature extraction, feature transformation and classification, training of HMM parameters and Viterbi recognition.

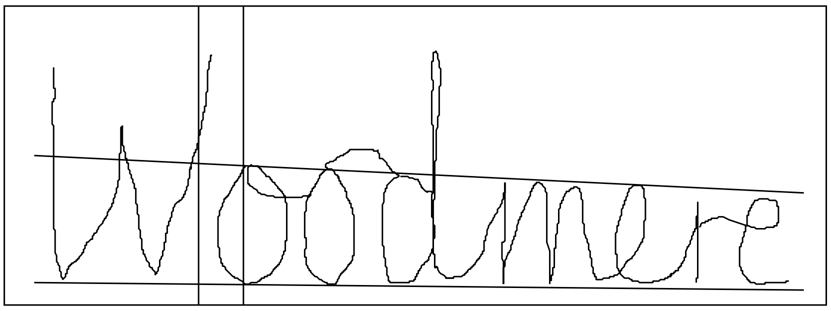


Figure: Address word after preprocessing and normalization.

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 the whole sequence. Although they actually have to be treated separately for each class $k = 1, \ldots, K$ (i.e., p has the form $p(x_1^T \mid k)$), in the following, to simplify matters, we want to use only the short form.

$$\begin{aligned} p(x_1^T) &= \sum_{[s_1^T]} p(x_1^T, s_1^T) \qquad \text{(introducing states)} \quad (*) \\ p(x_1^T, s_1^T) &= \prod_{t=1}^{T} p(x_t, s_t \mid x_1^{t-1}, s_1^{t-1}) \end{aligned}$$

Model assumptions:

- Dependency only on states (*hidden*):

$$\boxed{p(x_t, s_t \mid x_1^{t-1}, s_1^{t-1}) = p(x_t, s_t \mid s_1^{t-1})}$$

- Dependency only on the previous state (*first-order Markov*):

$$\boxed{p(x_t, s_t \mid s_1^{t-1}) = p(x_t, s_t \mid s_{t-1}) = p(s_t \mid s_{t-1}) \cdot p(x_t \mid s_{t-1}, s_t)}$$

($\hat{=}$ transition probability $\cdot$ emission probability (density))

Inserting in $(*)$ results in:

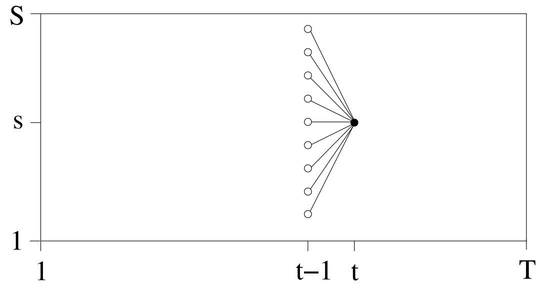
$$\begin{aligned} p(x_1^T) &= \sum_{[s_1^T]} \prod_{t=1}^{T} \left[p(s_t \mid s_{t-1}) \cdot p(x_t \mid s_{t-1}, s_t) \right] \\ &= \sum_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t \mid s_{t-1}) \ . \end{aligned}$$

The 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 a simple and efficient method which we will derive in the following (cf. [Ney 99a]):

7.2.1 Baum Recursion

We define the auxiliary function $Q(t,s)$ as "the 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: $[\cdots \longrightarrow (s,t)] = [\cdots \longrightarrow (\sigma, t-1)]\;[(\sigma,t-1) \longrightarrow (s,t)]$

$$Q(t,s) \;=\; \sum_{\sigma} \Big[\underbrace{\Big[\sum_{s_1^{t-1} : s_{t-1} = \sigma} \prod_{\tau=1}^{t-1} p(x_\tau, s_\tau | s_{\tau-1}) \Big]}_{=Q(t-1,\sigma)} \cdot p(x_t, s|\sigma) \Big]$$

$$\boxed{\begin{aligned} Q(t,s) &= \sum_{\sigma} p(x_t, s|\sigma) \cdot Q(t-1,\sigma) \\ p(x_1^T) &= Q(T,S) \end{aligned}}$$

Recursion equation of type *divide and conquer* (*dynamic programming*)

The computing time is proportional to $T \cdot S^2$ – instead of $3^{T-1}/(2 \cdot (T-1))$ for the standard $(0,1,2)$-model or even S^{T-1} in general.

For historical reasons, this equation is 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 a 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".

As for the Baum(-Welch) recursion, we define $Q(t, s)$:

$$Q(t,s) := \max_{s_1^t : s_t = s} \prod_{\tau=1}^{t} p(x_\tau, s_\tau | s_{\tau-1}) \; .$$

The decomposition results in:

$$Q(t,s) = \max_{\sigma} \, [p(x_t, s|\sigma) \cdot \underbrace{\max_{s_1^{t-1} : s_{t-1} = \sigma} \prod_{\tau=1}^{t-1} p(x_\tau, s_\tau | s_{\tau-1})]}_{=Q(\tau-1,\sigma)} \; .$$

$$\boxed{Q(t,s) = \max_{\sigma} \left[p(x_t, s|\sigma) \cdot Q(t-1, \sigma)\right]}$$

Recursive equation of dynamic programming

As in the previous case, the computing 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 ($t = T$), 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 Incorporation into the Bayes Decision Rule

Each class k has its own automaton with the finite 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, we have:

$$p(x_1^T|k) \;=\; Q_k(T, S(k)),$$

and the decision rule is:

$$\begin{aligned} x_1^T \longrightarrow r(x_1^T) &= \underset{k}{\operatorname{argmax}} \, \{p(k) \cdot p(x_1^T|k)\} \\ &= \underset{k}{\operatorname{argmax}} \, \{p(k) \cdot Q_k(T, S(k))\} \,. \end{aligned}$$

7.4 Maximum Likelihood Training

The same approach as for mixture densities, i.e. Baum algorithm, EM algorithm

Vector sequence:	x_1^T	(only one observation; for several: 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: $\quad p(x_1^T, s_1^T | \tilde{\lambda}) = \prod_{t=1}^{T} \left[p(s_t | s_{t-1}, \tilde{\lambda}) \cdot p(x_t | s_{t-1}, s_t, \tilde{\lambda}) \right]$
- Parameter λ: $\quad \alpha(s|\sigma), \quad \sum_s \alpha(s|\sigma) = 1, \quad \forall \sigma$ instead of $p(s|\sigma)$.

$p(x|\sigma, s; \vartheta) = p(x|s; \vartheta_s)$ e.g. Gaussian distribution or mixture density with the parameters ϑ_s

Parameter set: $\quad \lambda = \{\alpha(s|\sigma), \vartheta_s\}$

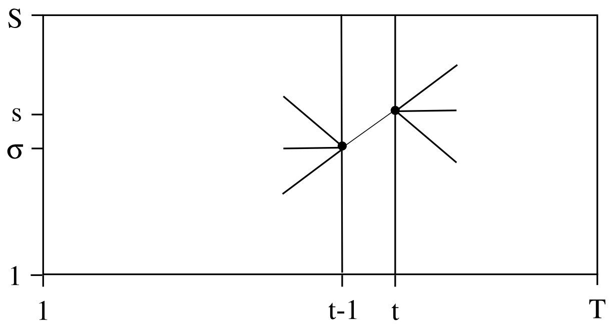
Regrouping of the terms results in:

$$\begin{aligned} Q(\lambda, \tilde{\lambda}) = \frac{1}{p(x_1^T|\lambda)} \quad \cdot \quad & \left[\sum_{s,\sigma} \sum_t p(x_1^T, s_{t-1} = \sigma, s_t = s | \lambda) \cdot \log \tilde{\alpha}(s|\sigma) \right. \\ & \left. + \quad \sum_s \sum_t p(x_1^T, s_t = s | \lambda) \cdot \log p(x_t | s, \tilde{\vartheta}_s) \right] \end{aligned}$$

Remark:
An initial distribution $\pi(s)$ which is often found in literature is omitted here; it can be easily modeled by inserting a fictitious state $s = 0$ so that $\pi(s) := p(s|0)$.

$$\begin{aligned} p(x_1^T|\lambda) &= \sum_{[s_1^T]} p(x_1^T, s_1^T|\lambda) \\ p(x_1^T, s_{t-1}=\sigma, s_t=s|\lambda) &:= \sum_{\substack{[s_1^T];\\ s_{t-1}=\sigma; s_t=s}} p(x_1^T, s_1^T|\lambda) \end{aligned}$$

"Paths go through $(t-1, \sigma)$ and (t, s)"



Analogous definition for $p(x_1^T, s_t - 1 = s|\lambda)$.
We define the "posterior" auxiliary functions:

$$\begin{aligned} \gamma_t(\sigma, s) &= p(s_{t-1}=\sigma, s_t=s|x_1^T, \lambda) = \frac{p(x_1^T, s_{t-1}=\sigma, s_t=s|\lambda)}{p(x_1^T|\lambda)} \\ &= \frac{\sum\limits_{\substack{[s_1^T];\\ s_{t-1}=\sigma; s_t=s}} p(x_1^T, s_1^T|\lambda)}{\sum\limits_{s_1^T} p(x_1^T, s_1^T|\lambda)} \\ \gamma_t(s) &= p(s_t=s|x_1^T, \lambda) = \frac{p(x_1^T, s_t=s|\lambda)}{p(x_1^T|\lambda)} \end{aligned}$$

Model: Gaussian distribution $\vartheta_s = [\mu_s, \Sigma_s]$ for each state s

Iteration formulae (reestimation):

$$\tilde{\alpha}(s|\sigma) = \frac{\sum\limits_t \gamma_t(\sigma,s)}{\sum\limits_t \sum\limits_{s'} \gamma_t(\sigma,s')} = \frac{\sum\limits_t \gamma_t(\sigma,s)}{\sum\limits_t \gamma_t(\sigma)}$$

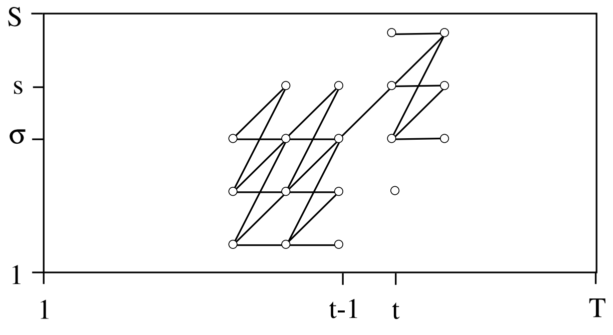
$$\begin{aligned}
\Rightarrow \quad Q(\lambda,\tilde{\lambda}) &= \sum_{s,\sigma}\sum_t \gamma_t(\sigma,s)\cdot\log\tilde{\alpha}(s|\sigma) \\
&\quad + \sum_{s,\sigma}\sum_t \gamma_t(s)\cdot\log p(x_t|s,\tilde{\theta}_s) \\
p(x_t|s,\tilde{\theta}_s) &\overset{!}{=} \mathcal{N}(x_t|\tilde{\mu}_s,\tilde{\Sigma}_s)
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad \tilde{\mu}_s &= \frac{\sum\limits_t \gamma_t(s)\cdot x_t}{\sum\limits_t \gamma_t(s)} \\
\tilde{\Sigma}_s &= \frac{\sum\limits_t \gamma_t(s)\cdot[x_t-\tilde{\mu}_s]\cdot[x_t-\tilde{\mu}_s]^T}{\sum\limits_t \gamma_t(s)}
\end{aligned}$$

Remarks:

1. Mixture densities: analogous result, but more complicated proof of the EM method (exercise)
2. Necessary: extension to several 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).

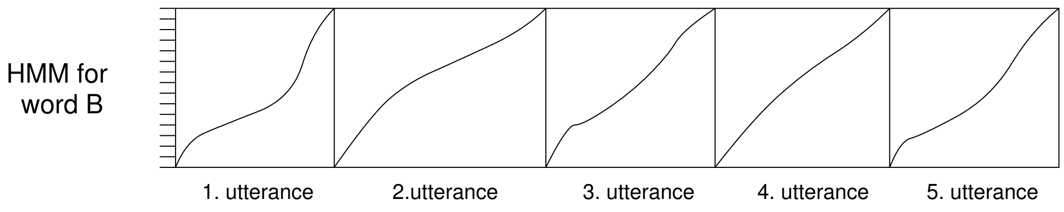
$\gamma_t(\sigma, s)$ and $\gamma_t(s)$ can be efficiently computed by combining the forward and backward variants of the Baum recursion, i.e. the *forward-backward algorithm.*
The principle is simple, but the implementation has to be done using tricks concerning the CPU and memory requirements.

Instead, we will deal with 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 by means of the maximum approximation: (cf. [Ney 99a])
training speech 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 mean
   $\hat{\Sigma}$ = 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 (Outlook)

Goal: transferring the HMM concept to general context-free grammars.

Notation:	$A, B, C, \dots, Z$	non-terminals
	$x_1, \dots, x_t, \dots, x_T$	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

  $$\begin{tikzcd}\sigma \arrow[r, "x"] & s\end{tikzcd}$$

  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 the interpretation).

  The whole regular grammar can be built from this rule type (except for the initial states).

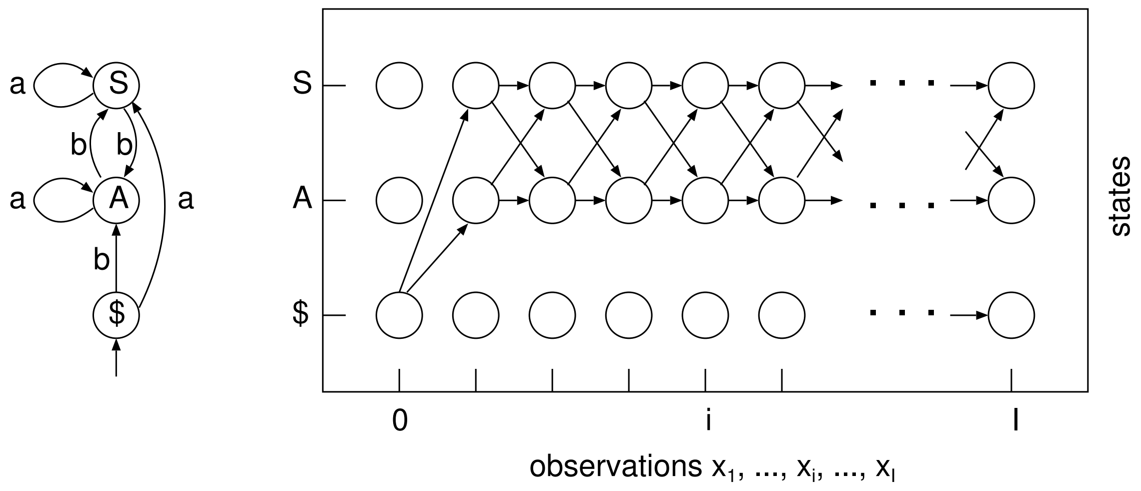
- *Stochastic context-free grammar* (Chomsky normal form):

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 the non-terminal A is given.

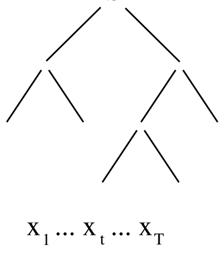
Remark:

For significantly distinctive structures only a small number of probabilities $p(BC|A) > 0$.



Consider the chain of rules $r_1 \ldots r_M$ (in Chomsky normal form: $M = 2T - 1$) :

$$S \xrightarrow{r_1} \cdots \xrightarrow{r_m} \cdots \xrightarrow{r_M} [x_1 \ldots x_t \ldots x_T]$$



r: A $\longrightarrow$ BC
p(BC|A) := q(r)
r: A $\longrightarrow$ x
p(x|A) := q(r)

Model: $q(r_m|r_{m-1}) = q(r_m)$ "arbitrary order"

$$\boxed{p(x_1^T, r_1^M) = \prod_{m=1}^{M} q(r_m)}$$

$$\begin{aligned} p(x_1^T) &= \sum_{[r_1^M]} p(x_1^T, r_1^M) && \text{"all derivations" (recognizer)} \\ &\cong \max_{[r_1^M]} p(x_1^T, r_1^M) && \text{"best derivations" (parser)} \end{aligned}$$

Remarks:

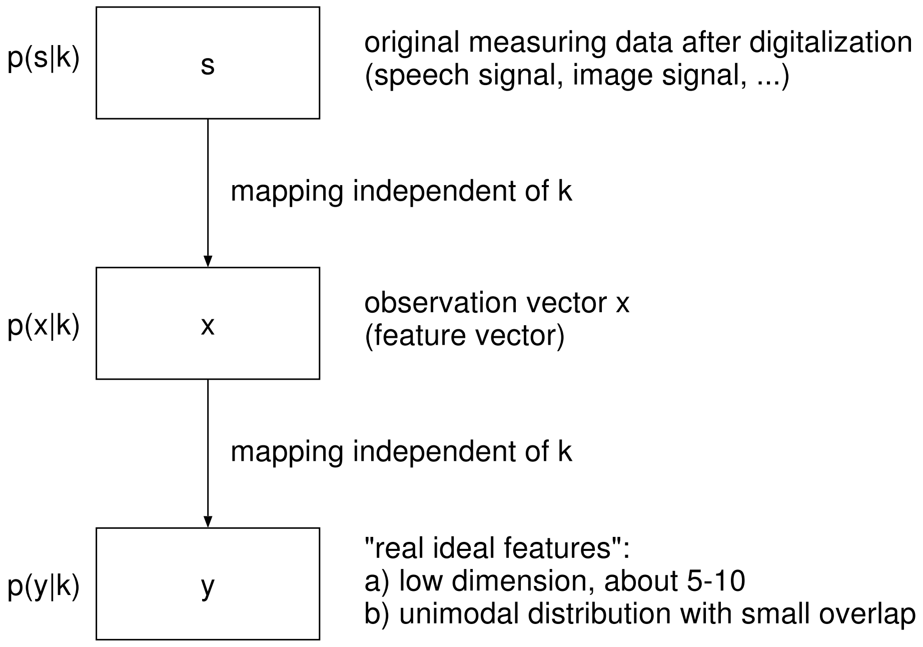
1. Efficient algorithms based on dynamic programming (or divide and conquer) exist for both variants ($\sum$, max).
2. In this way, we obtain a stochastic variant of the Cocke-Younger-Kasami algorithm or the Earley algorithm (bottom-up or top-down parser for context-free grammars).
3. Training:
   - extensions to the reestimation equations exist
   - So far, we do not have much experience with real problems.
   - the current problem: learning the rules (*grammatical inference*)
4. Hierarchical structure is possible:
   - word level: combining word observations
   - sentence level: combining words into sentences

   (Example: introduce a parity check in digit sequence recognition)

8 Feature Extraction and Linear Mapping

8.1 Ideal Feature Extraction

Different data types (k = class index):


p(s|k)
s
original measuring data after digitalization
(speech signal, image signal, ...)
mapping independent of k
p(x|k)
x
observation vector x
(feature vector)
mapping independent of k
p(y|k)
y
"real ideal features":
a) low dimension, about 5-10
b) unimodal distribution with small overlap

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 of the error rate when the Bayes decision rule is used. Only the computing time may change (the original data s are always extremely extensive).

   Usually, the mappings $s \longrightarrow x \longrightarrow y$ are connected with a data reduction, therefore there is no gain in reducing the error rate for given distributions.

2. The distributions are not known in practice, thus the feature extraction has the following advantages:

   (a) less amount of data;

   (b) fixed dimension of the observation vector $x, y \in \mathbb{R}^D$;

   (c) Dependencies between components of x or y can be captured easier (cf. problem: estimation of covariance matrix).

3. (Possible) strategy:

   (a) Fully automatic determination of the 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) The choice of the 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 viewed in the following.

8.2 Linear Mappings

8.2.1 Effect for a Given Gaussian Distribution

Given μ and Σ:

(example for illustration)
$x \in \mathbb{R}^D : \qquad p(x|\mu_x, \Sigma_x)$
If $V \in \mathbb{R}^{D\times D}$ is invertible, the "distance ratios" remain unchanged.

Proof:

$$\begin{aligned} y = V \cdot x \implies \mu_y &= E\{y\} = V \cdot \mu_x \\ \Sigma_y &= \cdots = V \cdot \Sigma_x \cdot V^T \end{aligned}$$

$$\begin{aligned} d_x^2 &\equiv (x-\mu_x)^T \Sigma_x^{-1} (x-\mu_x) \quad \text{(exponent in } p(x|\mu_x,\Sigma_x)) \\ d_y^2 &\equiv (y-\mu_y)^T \Sigma_y^{-1} (y-\mu_y) \\ &= (x-\mu_x)^T \cdot \underbrace{V^T \Sigma_y^{-1} V}_{\substack{V^T (V^T)^{-1} \Sigma_x^{-1} V^{-1} V \\ = \Sigma_x^{-1}}} \cdot (x-\mu_x) \end{aligned}$$

$$\implies d_x^2 \equiv d_y^2$$

8.2.2 Minimization of the Representation Error: Karhunen-Loève Transformation

([Fukunaga 90], p.417)

other names: KL expansion, principal component analysis, Hotelling transformation [also principal axis transformation, diagonalization]

$x \in \mathrm{I\!R}^D$ [the class membership is not given]
Basis vectors, orthogonal and normalized:
$V = [v_1, \ldots, v_i, \ldots, v_D], \quad v_i \in \mathrm{I\!R}^D, \quad i = 1, \ldots, D$
with

$$v_i^T v_j = \begin{cases} 1, & \text{i=j} \\ 0, & \text{otherwise} \end{cases}$$

Decomposition:

$$x = \sum_{i=1}^{D} \alpha_i(x) \cdot v_i, \qquad \text{with } \boxed{\alpha_i(x) = v_i^T \cdot x}$$

Approach:
Starting from $i > d$, replace the coefficients $\alpha_i(x)$ with a 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 $\quad$ ($v_i =$ mean 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 $v_i \in \mathbb{R}^D$ with $\|v_i\|^2 = 1$ yields the eigenvalue equation:

$$\Sigma \cdot v_i \;=\; \lambda_i \cdot v_i, \qquad\qquad \lambda_i > 0.$$

(Derivation: either clever estimation using linear algebra or by means of the Lagrange formalism)

Sort the eigenvalues in descending order: $\lambda_1 > \lambda_2 > \ldots > \lambda_D$. Then set

$$F(d) := \sum_{i>d} \lambda_i$$

Then, $[v_1, \ldots, v_{d^*}]$ is the matrix for dimension reduction (with suitably chosen d^*).

Remark:
The same derivation is possible when the representation error is averaged over one sample (difference: $\lambda_i \geq 0$).

Karhunen-Loève transformation:

1. Calculate the (empirical) covariance matrix Σ;
2. Diagonalize Σ;
3. Arrange the eigenvectors v_i according to the eigenvalues λ_i;
4. Choose the new dimension d (*trial-and-error*); sometimes, the relative error is a helpful criterion:

$$E_d = \frac{F(d)}{F(0)} = \frac{\sum\limits_{i>d} \lambda_i}{\sum\limits_{i=1}^{D} \lambda_i} \; .$$

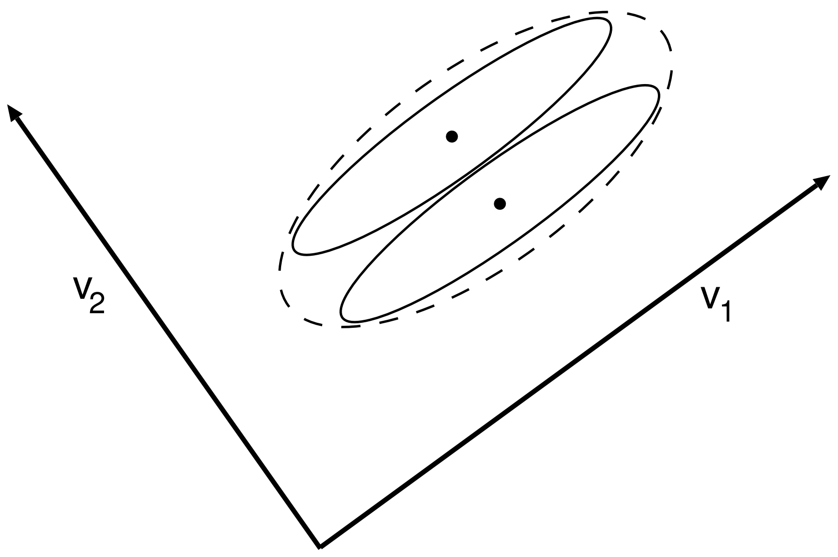
Remarks:

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. The derivation does not make any assumption about the distributions.
4. The new features $\alpha_i(x) = v_i^T \cdot x$ have a diagonal covariance matrix, i.e., they are “decorrelated”.
5. If (in the context of time sequence analysis/signal analysis, cf. [Ney 99a])

   $$\Sigma_{ij} \;=\; R(i-j),$$

   i.e., if Σ_{ij} has a *Töplitz structure*, the diagonalization is achieved using Fourier/cosine transformation.

Counter-example (to point 2):
"The 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)

([Duda & Hart 73], p.118-120)

Given: training data $x_n \in \mathbb{R}^D$ with $n = 1, \ldots, N$:

$$\mu \;:=\; \frac{1}{N}\sum_x x$$

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) the distances within each class remain constant;

ii) the distances between the class centers μ_k are maximized.

Let $\operatorname{tr}(X) := \sum\limits_{d=1}^{D} X_{dd}$ be the trace of a matrix $X \in \mathbb{R}^{D\times D}$
such that $u^T u = \|u\|^2 = \sum\limits_d u_d^2 = \operatorname{tr}(uu^T)$ for $u \in \mathbb{R}^D$.
Then,

for i)

$$\begin{aligned} D_W(V) &:= \sum_k \sum_{x\in A_k} \|V^T(x-\mu_k)\|^2 \\ &= \operatorname{tr}\left[\sum_k \sum_{x\in A_k} V^T(x-\mu_k)(x-\mu_k)^T V\right] \\ &= \operatorname{tr}\left[V^T \underbrace{\sum_k \sum_{x\in A_k} (x-\mu_k)(x-\mu_k)^T}_{=:W} V\right] \end{aligned}$$

W : *within-class scatter matrix*

for ii)

$$\begin{aligned} D_B(V) &:= \sum_k n_k \|V^T(\mu_k-\mu)\|^2 \\ &\vdots \\ &= \operatorname{tr}\left[\sum_k n_k\, V^T(\mu_k-\mu)(\mu_k-\mu)^T V\right] \\ &= \operatorname{tr}\left[V^T \underbrace{\sum_k n_k\, (\mu_k-\mu)(\mu_k-\mu)^T}_{=:B} V\right] \end{aligned}$$

B : *between-class scatter matrix*

The result is the *total scatter matrix*

$$T \ := \ \sum_x (x-\mu)(x-\mu)^T$$

which is independent of class memberships.

Then, we have:

$$\boxed{T \ = \ W + B}$$

Transition: trace → determinant:
The within-class scatter matrix is given by the volume of a hyperellipsoid; 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

$$F(V) \ = \ \frac{\det(V^T B V)}{\det(V^T W V)} \ \overset{!}{=} \ \text{maximum}$$

$$\text{resp.} \quad F(V) \ = \ \frac{\text{tr}(V^T B V)}{\text{tr}(V^T W V)} \ \overset{!}{=} \ \text{maximum}$$

or alternatively

$$a^T W a \ \overset{!}{=} \ \text{minimum} \qquad \text{and} \qquad a^T B a \ \overset{!}{=} \ \text{maximum}$$

resp.

$$a^T W a \ \overset{!}{=} \ \text{minimum} \quad \text{with constraint} \quad a^T B a \ = \ 1$$

or

$$a^T B a \ \overset{!}{=} \ \text{maximum} \quad \text{with constraint} \quad a^T W a \ = \ 1$$

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 the eigenvectors v_d
b) sort them according to the eigenvalues λ_d

Direct derivation:

Mapping $V = [v_1, \ldots, v_d, \ldots, v_D]$ with $v_d \in \mathbb{R}^D$, choose each vector v_d so that:

- $v_d^T \cdot B \cdot v_d =$ maximum
- with the constraint: $v_d^T \cdot W \cdot v_d =$ *const*

$$\text{(equivalent to: } \frac{v_d^T \cdot B \cdot v_d}{v_d^T \cdot W \cdot v_d} = \mathit{maximum}\text{)}$$

Lagrange: $\qquad f(v_d, \lambda_d) = v_d^T \cdot B \cdot v_d - \lambda_d (v_d^T \cdot W \cdot v_d - 1)$

After several calculation steps (linear algebra, differentiation), we obtain:

$$B \cdot v_d = \lambda_d \cdot W \cdot v_d.$$

Remarks:

1. Due to the properties of the functions tr and det, the solution for V is not unique :

   an invertible mapping leaves the optimum invariant.

2. Without dimension reduction, i.e. $V \in \mathbb{R}^{D \times D}$, we have:

$$F(I) = F(V)$$

   I.e., the LDA does not make sense without dimension reduction. A regular mapping in the new feature space leaves the criterion $F(V)$ invariant.

3. The eigenvalues λ_d are invariant with respect to invertible linear transformations (cf. [Duda & Hart 73], p.223):

$$\begin{aligned} \text{tr}(W^{-1}B) &= \sum_d \lambda_d & \qquad tr(W^{-1}T) &= \sum_d (1+\lambda_d) \\ \det(W^{-1}B) &= \prod_d \lambda_d & \qquad det(W^{-1}T) &= \prod_d (1+\lambda_d) \end{aligned}$$

   $(= 0$ because of 4., if $D \geq K)$

4. The between-class scatter matrix B has the rank $(K-1)$ according to the definition (cf. covariance matrix for Gaussian model), i.e., only $(K-1)$ eigenvalues λ_d are not equal to zero.

5. Special case: $W = I$ (*Isotropy*)

$$\Downarrow$$

$$\{\mu_k - \mu;\ k = 1, \ldots, K\}$$

$$\Downarrow$$

   Gram-Schmidt orthogonalization

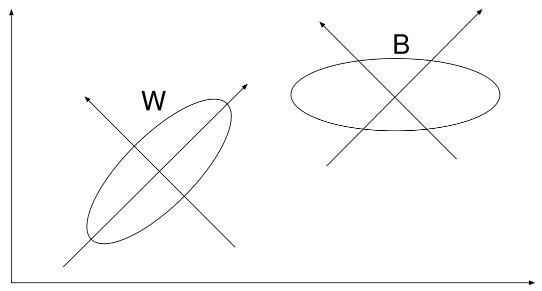
$$\Downarrow$$

   eigenvectors v_k

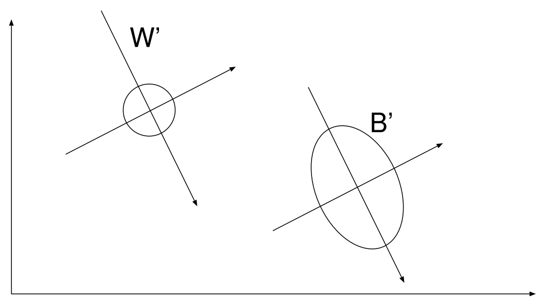
6. The derivation of the LDA uses only distances, but no Gaussian model.

Illustration ("simultaneous diagonalization") ([Fukunaga 90],p.31)

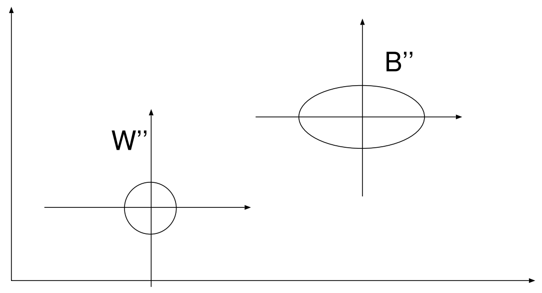
1. Determine the principal axes of W and assign them to B.



2. Scale the axes so that W is a hypersphere. Determine the principal axes of B and assign them to W.



3. Rotate the axes appropriately.



4. Reduce the dimension according to the new axes.

8.2.4 Motivation for Determinant Criterion

- Basics
  Gaussian model: $\mathcal{N}(x|\mu,\Sigma),\ x \in I\!R^D$
  Data: $x_1,\ldots,x_N$

$$\begin{aligned}
F(\mu,\Sigma) &:= \sum_n \log \mathcal{N}(x_n|\mu,\Sigma) \\
&= -\frac{1}{2}\sum_n [\log\det(2\pi\Sigma) + (x_n-\mu)^T\Sigma^{-1}(x_n-\mu)] \\
&\quad (\text{with } a^Tb = tr(ab^T) = tr(ba^T) \\
&\quad\ \text{and } tr(A+B) = tr(A)+tr(B)\ ) \\
&= -\frac{N}{2}\log\det(2\pi\Sigma) - \frac{1}{2}\sum_n tr(\Sigma^{-1}(x_n-\mu)(x_n-\mu)^T)] \\
&\quad (\text{insert ML estimate } \hat{\mu} \text{ and } \hat{\Sigma} = \frac{1}{N}\sum_n (x_n-\hat{\mu})(x_n-\hat{\mu})^T) \\
&= -\frac{N}{2}[\log\det(2\pi\hat{\Sigma}) + tr(I_D)] \\
&= -\frac{N}{2}[\log\det(2\pi\hat{\Sigma}) + D] \\
&= -\frac{N}{2}\log\det(2\pi e\hat{\Sigma})
\end{aligned}$$

- Model score
  class-dependent models: $p(x|k) = \mathcal{N}(x|\mu_k, W)$

  Score: N samples
  $$-\frac{N}{2}\log\det(2\pi e W)\ (\star)$$

  $p(x) = \mathcal{N}(x|\mu,T)$ 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 Classifiers/Discriminants

(cf. Chapter 4.2)

The number of classes K is arbitrary (i.e. not only 2 classes).

Reminder: Gaussian model with $\Sigma_k = \Sigma$ leads to linear Interfaces.

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 ([Duda & Hart 73], p.151,177).

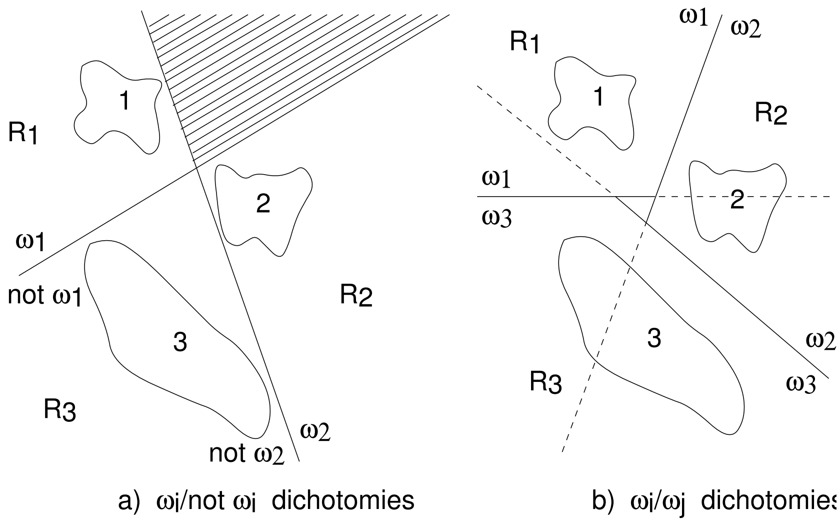
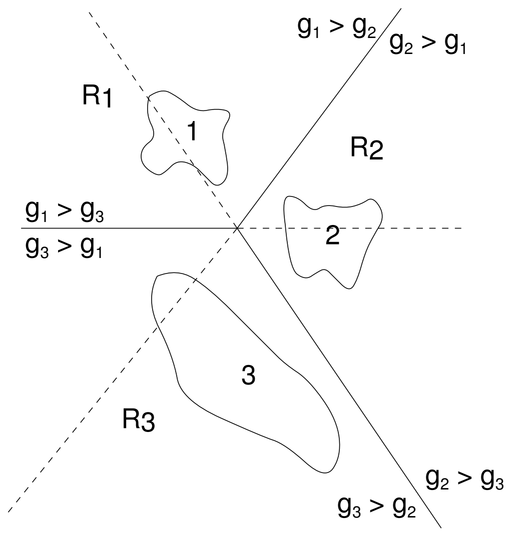
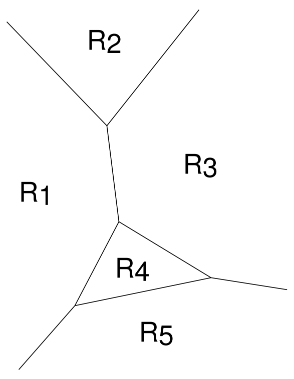


Figure: linear classification boundaries.



a) three-class problem



b) five-class problem

Model:
Decision rule:

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

Discriminant:

$$g(x,k) \; = \; v_k^T \cdot x, \qquad x, v_k \in \mathbb{R}^D.$$

Training: (ideal values)

$$\begin{array}{ll} v_k^T x \overset{!}{=} 1 \;, & \text{if } x \text{ belongs to class } k, \\ v_k^T x \overset{!}{=} 0 \;, & \text{if } x \text{ does not belong to class } k \;. \end{array}$$

In general, ideal demands can be realized only approximatively. Therefore, in general, the squared error criterion is 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 we obtain a linear equation system for v_k with $k = 1, \ldots, K$. 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. $g(x, k) = v_k^T f(x), \qquad f : \mathbb{R}^D \rightarrow \mathbb{R}^{D'},\ v_k \in \mathbb{R}^{D'}$

Notation: *polynomial discriminant function*, *regression classifier*

9 Classification of Strings

Problem: Given a sequence of class labels/decisions

$$x_1^N \rightarrow \hat{c}_1^N(x_1^N) = [\hat{c}_1(x_1^N), \ldots, \hat{c}_n(x_1^N), \ldots, \hat{c}_N(x_1^N)]$$

For a sequence of feature vectors $x_n \in \mathrm{I\!R}^D$ with $n = 1, \ldots, N$, we have a separate decision for every single point n (the c_i). Each class depends on the whole sequence (i.e. context matters). With N being a fixed length, we have:

$$p(c_1^N, x_1^N) = p(c_1^N) \prod_{n=1}^{N} p_n(x_n|c_n)$$

Decision rule:

$$x_1^N \rightarrow \ \hat{c}_1^N(x_1^N) = \underset{c_1^N}{\operatorname{argmin}}\, R(c_1^N|x_1^N)$$
$$R(c_1^N|x_1^N) = \sum_{\tilde{c}_1^N} p(\tilde{c}_1^N) \cdot L\left[\tilde{c}_1^N, c_1^N\right]$$

Cost function for the string error (E_S):

$$L_{\text{str}}\left[\tilde{c}_1^N, c_1^N\right] = 1 - \prod_{n=1}^{N} \delta(c_n, \tilde{c}_n) = \begin{cases} 1 & c_1^N \neq \tilde{c}_1^N \\ 0 & c_1^N = \tilde{c}_1^N \end{cases}$$

For the symbol error we have:

$$L_{\text{sym}}\left[\tilde{c}_1^N, c_1^N\right] = \frac{1}{N} \sum_{n=1}^{N} \left[1 - \delta(\tilde{c}_n, c_n)\right]$$

($\hat{=}$ count how many symbols are wrong in the sequence, can take on values from 0 to 1)

Of course the rule depends on the cost function. For the string error, we have:

$$R_{L_{\text{str}}}(c_1^N|x_1^N) = \sum_{\tilde{c}_1^N} p(\tilde{c}_1^N|x_1^N) \cdot \left[1 - \delta(c_1^N, \tilde{c}_1^N)\right] = 1 - p(c_1^N|x_1^N)$$
$$x_1^N \rightarrow \hat{c}_1^N(x_1^N) = \underset{c_1^N}{\operatorname{argmax}} \left\{p(c_1^N|x_1^N)\right\}$$
$$= \underset{c_1^N}{\operatorname{argmax}} \left\{p(c_1^N, x_1^N)\right\}$$

For the symbol error, we get:

$$\begin{aligned}
R_{L_{\text{sym}}}(c_1^N|x_1^N) &= \sum_{\tilde{c}_1^N} p_n(\tilde{c}_1^N|x_1^N) \cdot \frac{1}{N}\sum_{n=1}^{N} [1-\delta(\tilde{c}_n, c_n)] \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{\tilde{c}_1^N} p_n(\tilde{c}_1^N|x_1^N)\,[1-\delta(\tilde{c}_n, c_n)] \\
&= \frac{1}{N}\sum_{n=1}^{N}\left[1-p_n(c_n|x_1^N)\right]
\end{aligned}$$

$$\begin{aligned}
\min_{c_1^N} R_{L_{\text{sym}}}(c_1^N|x_1^N) &= \min_{c_1^N} \frac{1}{N}\sum_{n=1}^{N}\left[1-p_n(c_n|x_1^N)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}\min_{c_n}\left[1-p_n(c_n|x_1^N)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}\left[1-\max_{c_n} p_n(c_n|x_1^N)\right]
\end{aligned}$$

Decision rule:

$$\begin{aligned}
x_1^N \to \hat{c}_1^N(x_1^N) &= \left[\hat{c}_1(x_1^N), \ldots, \hat{c}_N(x_1^N)\right] \\
\text{where } \hat{c}_n(x_1^N) &= \operatorname*{argmax}_c \left\{p_n(c|x_1^N)\right\} \\
&= \operatorname*{argmax}_c \left\{p_n(c, x_1^N)\right\}
\end{aligned}$$

What about a decision with no context? Given a position n, let's look at the decision rule $x_n \to \hat{c}_n(x_n)$ in contrast to $x_1^N \to \hat{c}_n(x_1^N)$. The result for the symbol error will be

$$\begin{aligned}
x_n \to \hat{c}_n(x_n) &= \operatorname*{argmax}_c \left\{p_n(c|x_n)\right\} \\
&= \operatorname*{argmax}_c \left\{p_n(c, x_n)\right\} \\
&= \operatorname*{argmax}_c \left\{p_n(c) \cdot p(x_n|c)\right\}
\end{aligned}$$

9.1 Error rate for strings

We look at the accuracies $A = 1 - E$. In case of the String error, we have:
String accuracy:

$$\begin{aligned} A_{\text{string}} &= \sum_{x_1^N} p(\tilde{c}_1^N(x_1^N), x_1^N) \\ &= \sum_{x_1^N} \max_{c_1^N} \left\{ p(c_1^N, x_1^N) \right\} \end{aligned}$$

Symbol accuracy with context:

$$\begin{aligned} A_i &= \sum_{x_1^N} p_i(\hat{c}_i(x_1^N), x_1^N) \\ &= \sum_{x_1^N} \max_c p_i(c, x_1^N) \\ &= \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} p(c_1^N, x_1^N) \right\} \end{aligned}$$

Symbol accuracy without context:

$$\begin{aligned} \tilde{A}_i &= \sum_{x_1^N} p_i(\hat{c}_i(x_i), x_1^N) \\ &= \sum_{x_i} p_i(\hat{c}_i(x_i), x_i) \\ &= \sum_{x_i} \max_c \left\{ p_i(c) \cdot p(x|c_i) \right\} \end{aligned}$$

Now, how to compute $p_i(c)$?

$$\begin{aligned} p_i(c) &= \sum_{c_1^N : c_i = c} p(c_1^N), \qquad \text{with } p(c_1^N) = \prod_{n=1}^{N} p(c_n|c_{n-1}) \\ &= \sum_{c_1^i : c_i = c} p(c_1^i) \cdot \sum_{c_{i+1}^N} p(c_{i+1}^N | c_1^i) \end{aligned}$$

We define an auxiliary function

$$Q(i, c) = \sum_{c'} Q(i-1, c') \cdot p(c|c')$$

which leads to

$$p_i(c) = Q(i, c) \qquad (\hat{=} \text{ position dependent unigram prob.})$$

which can be computed recursively (dynamic programming).

9.1.1 Effect of Language model: Full vs. Unigram model

Accuracy with full context:

$$A_i^* = \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} p(c_1^N) \prod_n p_n(x_n|c_n) \right\}$$

Accuracy with unigram language model ($\tilde{p}_n(c_n)$):

$$\tilde{A}_i = \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} \prod_n \tilde{p}_n(c_n) \prod_n p_n(x_n|c_n) \right\}$$

Consider the difference:

$$\begin{aligned}
A_i^* - \tilde{A}_i &= \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} p(c_1^N) \prod_n p_n(x_n|c_n) \right\} \\
&\quad - \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} \prod_n \tilde{p}_n(c_n) \prod_n p_n(x_n|c_n) \right\} \\
&\le \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} \left[p(c_1^N) - \tilde{p}(c_1^N) \right] \prod_n p_n(x_n|c_n) \right\} \\
&= \sum_{x_1^N} \max_c \left\{ p_i(x_i|c_i) \sum_{c_1^N : c_i = c} \left[p(c_1^N) - \tilde{p}(c_1^N) \right] \prod_{n \neq i} p_n(x_n|c_n) \right\}
\end{aligned}$$

Note that $\sum_{c_1^N : c_i = c} \left[p(c_1^N) - \tilde{p}(c_1^N) \right] \tilde{p}(x_1^N) = 0$ for any $\tilde{p}(x_1^N)$ independent of $\{c_1^N : c_i = c\}$. Using this we get

$$\begin{aligned}
A_i^* - \tilde{A}_i &\le \ldots \\
&= \sum_{x_1^N} \max_c \left\{ p_i(x_i|c) \sum_{c_1^N : c_i = c} \left[p(c_1^N) - \tilde{p}(c_1^N) \right] \cdot \left[\prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right] \right\} \\
&\le \sum_{x_1^N} \max_c \left\{ p_i(x_i|c) \sum_{c_1^N : c_i = c} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right\} \\
&\le \sum_{x_1^N} \sum_c \left(p_i(x_i|c) \sum_{c_1^N : c_i = c} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right)
\end{aligned}$$

Now we can rearrange the sums

$$\begin{aligned}
A_i^* - \tilde{A}_i \;&\leq\; \ldots \\
&= \sum_c \sum_{c_1^N : c_i = c} \left|p(c_1^N) - \tilde{p}(c_1^N)\right| \sum_{x_1^N} \left(p_i(x_i|c) \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
&= \sum_{c_1^N} \left|p(c_1^N) - \tilde{p}(c_1^N)\right| \sum_{x_i} p_i(x_i|c) \sum_{x_1^N \setminus x_i} \left(\left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
&= \sum_{c_1^N} \left(\left|p(c_1^N) - \tilde{p}(c_1^N)\right| \cdot \sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
&\leq \sum_{c_1^N} \left|p(c_1^N) - \tilde{p}(c_1^N)\right| \cdot \underbrace{\max_{c_1^N \setminus c_i} \left\{ \sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right\}}_{R}
\end{aligned}$$

Finally, we can apply the Pinsker inequality

$$\left(\sum_n |p_n - q_n| \right)^2 \;\leq\; \frac{1}{2} \sum_n p_n \log \frac{p_n}{q_n}$$

and we get

$$\begin{aligned}
A_i^* - \tilde{A}_i \;&\leq\; \ldots \\
&\leq \left(\frac{1}{2} \sum_{c_1^N} p(c_1^N) \;\log\; \frac{p(c_1^N)}{\tilde{p}(c_1^N)} \right)^{\frac{1}{2}} \cdot R \\
&= \left(\underbrace{\frac{1}{2} \sum_{c_1^N} p(c_1^N) \;\log\; p(c_1^N)}_{\text{log PP: full LM}} - \underbrace{\frac{1}{2} \sum_{c_1^N} p(c_1^N) \;\log\; \tilde{p}(c_1^N)}_{\text{log PP: unigram LM}} \right)^{\frac{1}{2}} \cdot R
\end{aligned}$$

9.1.2 Effect of one more observation

$$\begin{aligned}
A_i(N) \;&=\; \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} p(c_1^N) \prod_{n=1}^{N} p_n(x_n|c_n) \right\} \\
A_i(N-1) \;&=\; \sum_{x_1^{N-1}} \max_c \left\{ \sum_{c_1^{N-1} : c_i = c} p(c_1^{N-1}) \prod_{n=1}^{N-1} p_n(x_n|c_n) \right\}
\end{aligned}$$

We know: $A_i(N) \geq A_i(N-1)$

$$\begin{aligned}
0 &\leq A_i(N) - A_i(N-1) \\
&= \sum_{x_1^N} \max_c \left\{ \sum_{c_1^N : c_i = c} p(c_1^N) \prod_{n=1}^{N} p_n(x_n|c_n) \right\} \\
&\quad - \sum_{x_1^N} \max_c \left\{ \sum_{c_1^{N-1} : c_i = c} p(c_1^{N-1})\, p_N(c_N) \prod_{n=1}^{N} p_n(x_n|c_n) \right\} \\
&\leq \sum_{x_1^N} \max_c \left\{ \sum_{c_1^{N-1} : c_i = c} \underbrace{\left[p(c_1^N) - p(c_1^{N-1})\, p_N(c_N) \right]}_{\left[p(c_1^N) - \tilde{p}(c_1^N) \right]} \prod_{n=1}^{N} p_n(x_n|c_n) \right\} \\
&\leq \sum_{c_1^N} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \underbrace{\sum_{c_1^N \backslash c_i} \sum_{x_1^N \backslash x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right|}_{R}
\end{aligned}$$

It follows

$$\begin{aligned}
[A_i(N) - A_i(N-1)]^2 &\leq \left(\sum_{c_1^N} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \right)^2 \cdot R^2 \\
&\leq \frac{1}{2} \sum_{c_1^N} p(c_1^N) \log \frac{p(c_1^N)}{\tilde{p}(c_1^N)} \cdot R^2
\end{aligned}$$

This corresponds to the difference of the perplexity.

9.2 Language model: Perplexity, Markov chains and stationarity

Consider the class prior in Bayes decision rule for strings:

$$x_1^N \;\rightarrow\; \hat{c}_1^N(x_1^N) = \underset{c_1^N}{\operatorname{argmax}} \left\{ p(c_1^N) \prod_n p(x_n|c_n) \right\}$$

The log perplexity is defined as the normalized average of the logarithmic prior:

$$-\log\ PP \;:=\; \frac{1}{N} \sum_{c_1^N} p(c_1^N) \log p(c_1^N)$$

Remarks:

- Experimental correlation of PP with error rate
- No theory for correlation of PP with error rate (no results from information theory)
- Additional problem: Suboptimal decision rule (string error vs. symbol error)

Re-write the perplexity using chain rule of conditional probabilities:

$$p(c_1^N) \;=\; \prod_n p_n(c_n|c_0^{n-1})$$

$$\begin{aligned}
-\log\ PP &= \frac{1}{N}\sum_{n=1}^{N}\sum_{c_1^N} p(c_1^N)\ \log\ p_n(c_n|c_0^{n-1}) \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{c_1^n} p(c_1^n)\ \log\ p_n(c_n|c_0^{n-1}) \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{c_1^{n-1}}\sum_{c_n} p_n(c_0^{n-1},c_n)\ \log\ p_n(c_n|c_0^{n-1})
\end{aligned}$$

Using the short-hand notation $(c_0^{n-1}, c_n) \to (h, c)$ we get:

$$\begin{aligned}
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{h}\sum_{c} p_n(h,c)\ \log\ p_n(c|h) \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{h} p_n(h)\sum_{c} p_n(c|h)\ \log\ p_n(c|h)
\end{aligned}$$

Additional assumptions:

- Markov chain of order k:

  $$p_n(c_n|c_0^{n-1}) = p_n(c_n|c_{n-k}^{n-1})$$

  The dependence is limited to k predecessor symbols. The history is re-defined: $h_n := c_{n-k}^{n-1}$ (k-gram of symbols)

  $$p_n(c_n|c_{n-k}^{n-1}) = p_n(c_n|h_n)$$

  Special attention is required for initial positions $n = 1, ..., k$ of the symbol string.

- Stationarity (position independence): The probability distributions are independent of position n.
  - Conditional probability:

    $$p_n(c|h) = p(c|h)$$

  - History probability:

    $$p_n(h) = p(h)$$

Resulting perplexity:

$$\begin{aligned} -\log\ PP &= \frac{1}{N}\sum_{n=1}^{N}\sum_{h} p_n(h)\sum_{c} p_n(c|h)\ \log\ p_n(c|h) \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{h} p(h)\sum_{c} p(c|h)\ \log\ p(c|h) \\ &= \sum_{h} p(h)\sum_{c} p(c|h)\ \log\ p(c|h) \\ &= \sum_{h}\sum_{c} p(h,c)\ \log\ p(c|h) \end{aligned}$$

For a model $q(c|h)$ we consider the difference in log-perplexity:

$$\begin{aligned}\Delta \log\ PP_q &= \sum_h \sum_c p(h,c)\ [\log\ p(c|h) - \log\ q(c|h)] \\ &= \sum_h \sum_c p(h,c)\ \log\ \frac{p(c|h)}{q(c|h)}\end{aligned}$$

With the cross-log-perplexity (or cross-entropy) PP_q for the model $q(c|h)$:

$$-\log\ PP_q = \sum_h \sum_c p(h,c)\ \log\ q(c|h)$$

9.2.1 From the Expectation to the Empirical Average

Given empirical data: A long string of symbols c_1^N (training text). We convert the symbol string c_1^N into pairs $(h_n, c_n), n = 1, ..., N$ and re-write the cross-log-entropy PP_q:

$$\begin{aligned}-\log\ PP_q &= \sum_h \sum_c p(h,c)\ \log\ q(c|h) \\ &= \frac{1}{N} \sum_n\ \log\ q(c_n|h_n)\end{aligned}$$

This is the formula of perplexity that is used in practice to optimize and to measure the test set perplexity.
Remarks:

- Long text = superstring: Concatenate all symbol strings!
- 'Events' to be predicted: True symbols and string end (= white space or sentence end).
- History: Suitable definition at the beginning of the string.

9.3 Bounds

9.3.1 L1 Norm and Maximum

As a first step we upper-bound the maximum by a sum:

$$\begin{aligned}
A_i \; & -\tilde{A}_i \le \\
\le \; & \sum_{x_1^N} \max_c \left\{ p_i(x_i|c) \sum_{c_1^N : c_i = c} \left[p(c_1^N) - \tilde{p}(c_1^N) \right] \cdot \left[\prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right] \right\} \\
\le \; & \sum_{x_1^N} \max_c \left\{ p_i(x_i|c) \sum_{c_1^N : c_i = c} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right\} \\
\le \; & \sum_{x_1^N} \sum_c \left(p_i(x_i|c) \sum_{c_1^N : c_i = c} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
= \; & \sum_{c_1^N} \left(\left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
\le \; & \sum_{c_1^N} \left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \max_{c_1^N \setminus c_i} \left\{ \sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right\}
\end{aligned}$$

9.3.2 Cauchy-Schwartz

We use the Cauchy-Schwartz inequality for the last step:

$$\begin{aligned}
A_i \; & -\tilde{A}_i \le \\
\le \; & \sum_{c_1^N} \left(\left| p(c_1^N) - \tilde{p}(c_1^N) \right| \cdot \sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right) \\
\le \; & \sqrt{\sum_{c_1^N} \left| p(c_1^N) - \tilde{p}(c_1^N) \right|^2} \cdot \sqrt{\sum_{c_1^N} \left(\sum_{x_1^N \setminus x_i} \left| \prod_{n \neq i} p_n(x_n|c_n) - \prod_{n \neq i} \tilde{p}_n(x_n) \right| \right)^2}
\end{aligned}$$

Note: The sum over c_1^N in the second term includes the index c_i.

Index

References

[Amari 67] S. I. Amari. A Theory of Adaptive Pattern Classifiers. *IEEE Trans. on Elec. Com.*, Vol. EC 16, pp. 279–307, 1967.

[Bentley 75] J. Bentley. Multidimensional Binary search Trees used for Associative Searching. *Communications of ACM*, Vol. 18, No. 9, pp. 509–517, September 1975.

[Cover & Thomas 91] T. Cover, J. Thomas. *Elements of Information Theory.* Wiley series in telecommunications. John Wiley & Sons, New York, NY, 1991.

[Duda & Hart 73] R. Duda, P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, NY, 1973.

[Friedman & Bentley$^+$ 77] J. Friedman, J. Bentley, R. Finkel. Finding Best Matches in Logarithmic Expected Time. *ACM Trans. on Mathematical Software*, Vol. 3, No. 3, pp. 209–226, September 1977.

[Fukunaga 90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Computer Science and Scientific Computing. Academic Press, Inc., San Diego, CA, 2nd edition, 1990.

[Gordon 81] A. D. Gordon. *Classification.* Monographs on Applied Probability and Statistics. Chapman and Hall, London, 1981.

[Keehn 65] D. G. Keehn. A note of learning for Gaussian properties. *IEEE Trans. on Information Theory*, Vol. IT-11, pp. 126–132, 1965.

[Krengel 91] U. Krengel. *Einf"uhrung in die Wahrscheinlichkeitstheorie und Statistik.* Vieweg Studium, Aufbaukurs Mathematik. Vieweg, Braunschweig, 3rd edition, 1991.

[Linde & Buzo$^+$ 80] Y. Linde, A. Buzo, R. M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Trans. Communications*, Vol. 28, pp. 84–95, January 1980.

[Lippmann 87] R. P. Lippmann. An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, Vol. , pp. 4 – 22, April 1987.

[Ney 99a] H. Ney. Vorlesungsmitschrift Spracherkennung und Suchverfahren, WS 1999/2000, 1999.

[Ney 99b] H. Ney. Vorlesungsmitschrift stochastische modellierung in der mustererkennung ss 1999, 1999.

[Parker 82] D. Parker. Learning Logic. Invention Report, S81-64, File 1, Office of Technology Licensing, Stanford University, 1982.

[Spath 77] H. Spath. *Clusteranalyse-Algorithmen.* R. Oldenbourg Verlag, M"unchen, 2nd edition, 1977.

[Werbos 74] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. thesis, Harvard University, 1974.