Pattern Recognition

Part 7: Gaussian Mixture Models (GMMs)

Gerhard Schmidt

Christian-Albrechts-Universität zu Kiel
Faculty of Engineering
Institute of Electrical and Information Engineering
Digital Signal Processing and System Theory

Gaussian Mixture Models (GMMs)

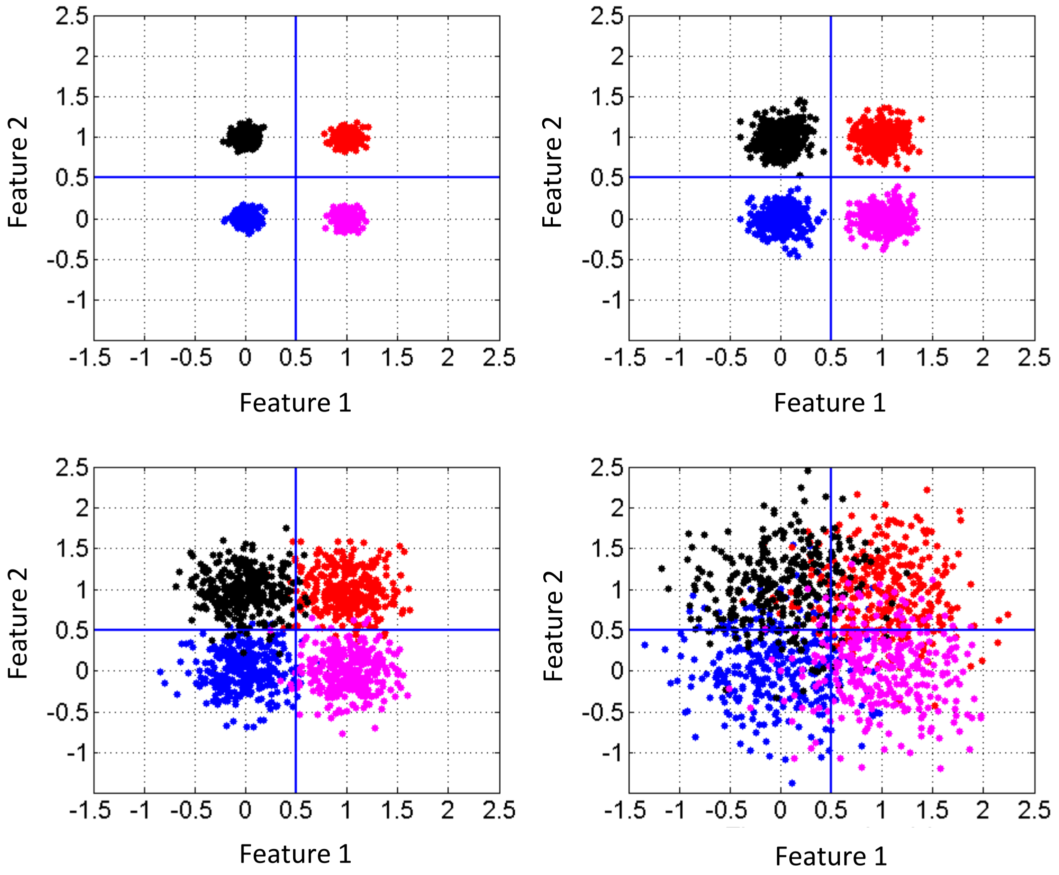
Contents

- Motivation
- Uncertainties in Machine Learning
- Fundamentals
  - Gaussian Mixture Models in practice
  - Generation of Gaussian Mixture Models
- Applications in speech and audio processing
  - Signal separation
  - Speaker recognition

Motivation

Codebook approaches and their limitations

- On one hand, ***codebooks*** are generally able to separate „classes“, but on the other hand, they do not reveal the probabilities of the observed data.
- If one wants to take advantage of the Bayesian probability, one has to model and estimate the ***probability density of the data***.



Literature

Gaussian Mixture Models:

- ❑ C. M. Bishop: ***Pattern Recognition and Machine Learning***, Springer, *2006*
- ❑ L. Rabiner, B. H. Juang: ***Fundamentals of Speech Recognition***, Prentice Hall, 1993
- ❑ B. Gold, N. Morgan: ***Speech and Audio Signal Processing***, Wiley, 2000

Uncertainties:

- ❑ E. Hüllermeier, W. Waegeman: ***Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods***, Machine Learning, Springer Science and Business Media LLC, 2021

Speaker recognition:

- ❑ G. Kolano: ***Lernverfahren zur Sprecherverifikation***, Shaker, 2000 (in German)
- ❑ J. Benesty, et al.: ***Handbook on Speech Processing***, Chapters 37 and 38 on „*Speaker Recognition*“, Springer, 2008

Multivariate Gaussian Distributed Probability Densities – Part 1

Definitions:

- For a ***feature vector***

$$\boldsymbol{x} = \big[x_0,\, x_1,\, \dots,\, x_{D-1}\big]^{\mathrm{T}}$$

we first define single ***Gaussian densities*** as

$$\mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu},\, \boldsymbol{\Sigma}\big) = \frac{1}{(2\pi)^{D/2}\,\det\{\boldsymbol{\Sigma}\}^{1/2}}\,\exp\Big\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big\}\,.$$

- In order to approximate arbitrary densities, we use a weighted sum of ***multiple Gaussian curves***:

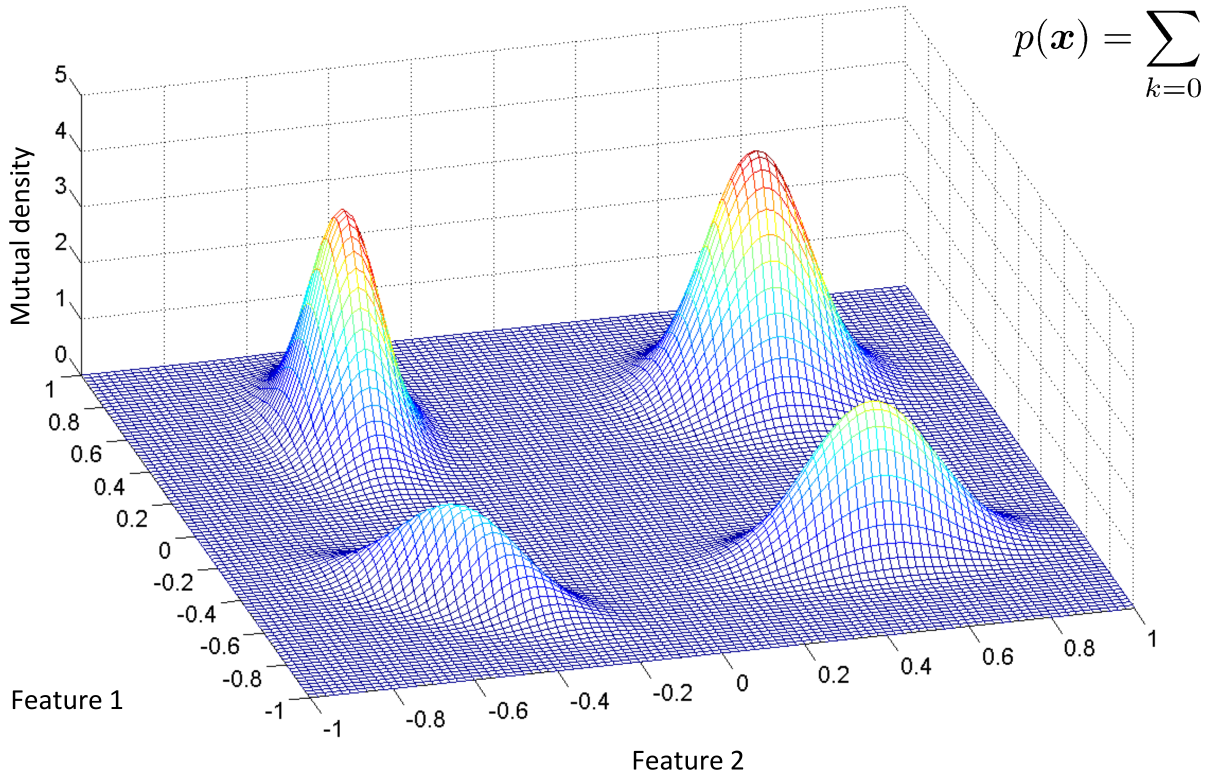
$$f_{\boldsymbol{x}}(\boldsymbol{x}) = p(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_k\, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big).$$

- Constraint for the ***weights***:

$$\sum_{k=0}^{K-1} g_k = 1.$$

Multivariate Gaussian Distributed Probability Densities – Part 2

Example:



$$p(\boldsymbol{x}) = \sum_{k=0}^{3} g_k \, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \, \boldsymbol{\Sigma}_k\big)$$

Contents

- ❑ Motivation
- ❑ ***Uncertainties in Machine Learning***
- ❑ Fundamentals
  - ❑ Gaussian Mixture Models in practice
  - ❑ Generation of Gaussian Mixture Models
- ❑ Applications in speech and audio processing
  - ❑ Signal separation
  - ❑ Speaker recognition

Uncertainties in Machine Learning – Motivation

Uncertainties in machine learning:

- When do you trust a human being (e.g. a medical doctor)?

AI-generated picture for the words “trust and doubt”

Uncertainties in Machine Learning – Motivation

Uncertainties in machine learning:

- ❑ When do you trust a human being (e.g. a medical doctor)?
- ❑ Two types of uncertainty:
  - ❑ The measured features do not allow a clear (secure) decision (***aleatoric*** uncertainty).

    This can not be changed, only minimized …

    Well covered by most machine learning approaches.

AI-generated picture for the words “trust and doubt”

Uncertainties in Machine Learning – Motivation

Uncertainties in machine learning:

- ❑ When do you trust a human being (e.g. a medical doctor)?
- ❑ Two types of uncertainty:
  - ❑ The measured features do not allow a clear (secure) decision (***aleatoric*** uncertainty).

    This can not be changed, only minimized …

    Well covered by most machine learning approaches.
  - ❑ New features that have not been seen in the training data should be used for a decision (***epistemic*** uncertainty).

    This can be measured in addition to standard training procedures.

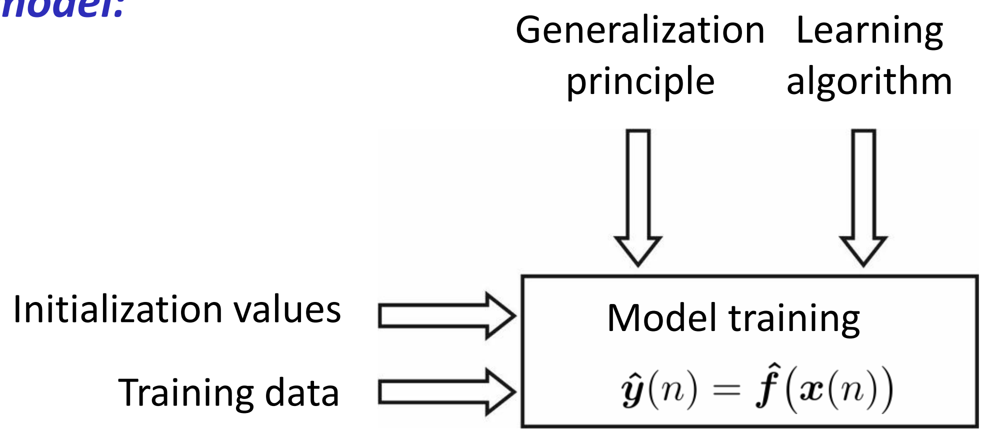
    Currently not often covered by machine learning approaches.

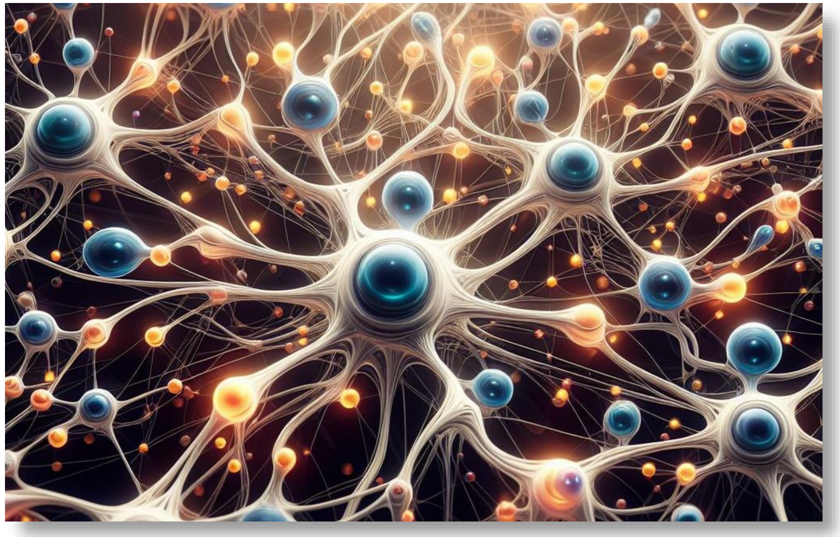
AI-generated picture for the words “trust and doubt”

Gaussian Mixture Models (GMMs)

Uncertainties in Machine Learning – Model of Supervised Learning

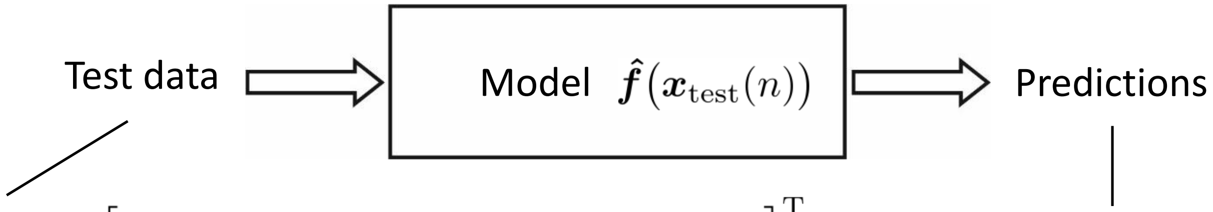
Train the model:





AI-generated picture using „Draw an artificial neural network“

Test the model:


Test data
Model $\hat{\boldsymbol{f}}\big(\boldsymbol{x}_{\text{test}}(n)\big)$
Predictions

$$\boldsymbol{x}_{\text{test}}(n) = \Big[x_{\text{test},0}(n),\, x_{\text{test},1}(n),\, \dots,\, x_{\text{test},N-1}(n)\Big]^{\mathrm{T}}$$

$$\begin{aligned}\hat{\boldsymbol{y}}_{\text{test}}(n) &= \hat{\boldsymbol{f}}\big(\boldsymbol{x}_{\text{test}}(n)\big) \\ &= \Big[\hat{y}_{\text{test},0}(n),\, \hat{y}_{\text{test},1}(n),\, \dots,\, \hat{y}_{\text{test},M-1}(n)\Big]^{\mathrm{T}}\end{aligned}$$

Uncertainties in Machine Learning – Mathematical Description Part 1

Some notation:

- For a given ***set of training data***

$$\boldsymbol{S} = \Big\{\{\boldsymbol{x}(0),\, \boldsymbol{y}(0)\},\, \{\boldsymbol{x}(1),\, \boldsymbol{y}(1)\},\, ...,\, \{\boldsymbol{x}(T-1),\, \boldsymbol{y}(T-1)\}\Big\}$$

  the learning algorithm of supervised learning defines the ***risk (expected loss)***

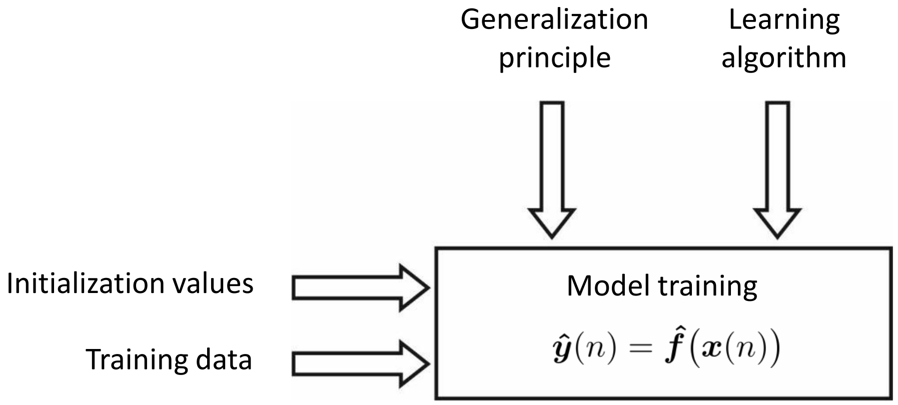
$$R(\hat{\boldsymbol{f}}) = \int\limits_{x,y} L\big(\hat{\boldsymbol{f}}(\boldsymbol{x}),\, \boldsymbol{y}\big)\, df_{xy}(\boldsymbol{x}, \boldsymbol{y})$$

  for the possible space of data defined by $\boldsymbol{x}$ and $\boldsymbol{y}$, using the ***probability density function*** $f_{xy}(\boldsymbol{x}, \boldsymbol{y})$ and the ***loss function*** $L(...)$.

- The risk is minimized

$$\hat{\boldsymbol{f}}_{\text{opt}} = \text{argmin}\Big\{R(\hat{\boldsymbol{f}})\Big\}$$

  and thus results in the ***hypothesis (true risk minimizer, optimal estimator)***.



Uncertainties in Machine Learning – Mathematical Description Part 2

Some notation – continued:

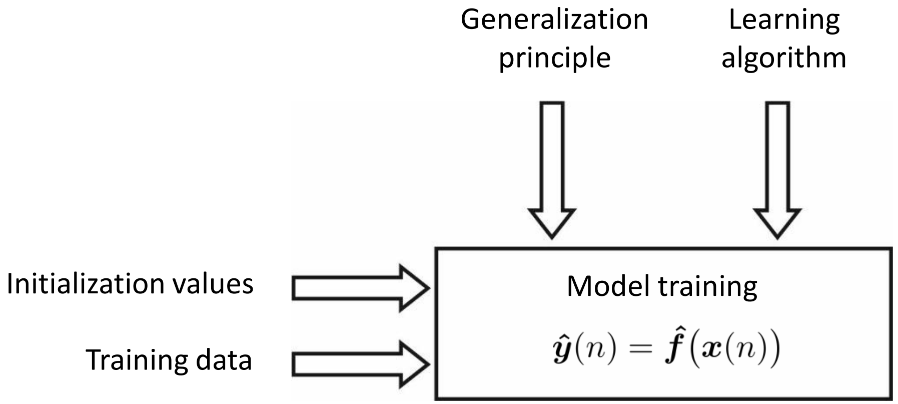
- For an associated pair $\{\boldsymbol{x}(n),\, \boldsymbol{y}(n)\}$ of training data, the ***empirical risk*** is defined as

$$\hat{R}(\hat{\boldsymbol{f}}) \;=\; \frac{1}{T}\sum_{n=0}^{T-1} L\big(\hat{\boldsymbol{f}}(\boldsymbol{x}(n)), \boldsymbol{y}(n)\big).$$

- For this the hypothesis

$$\hat{\boldsymbol{f}} \;=\; \mathrm{argmin}\Big\{\hat{R}(\hat{\boldsymbol{f}})\Big\}$$

is calculated.

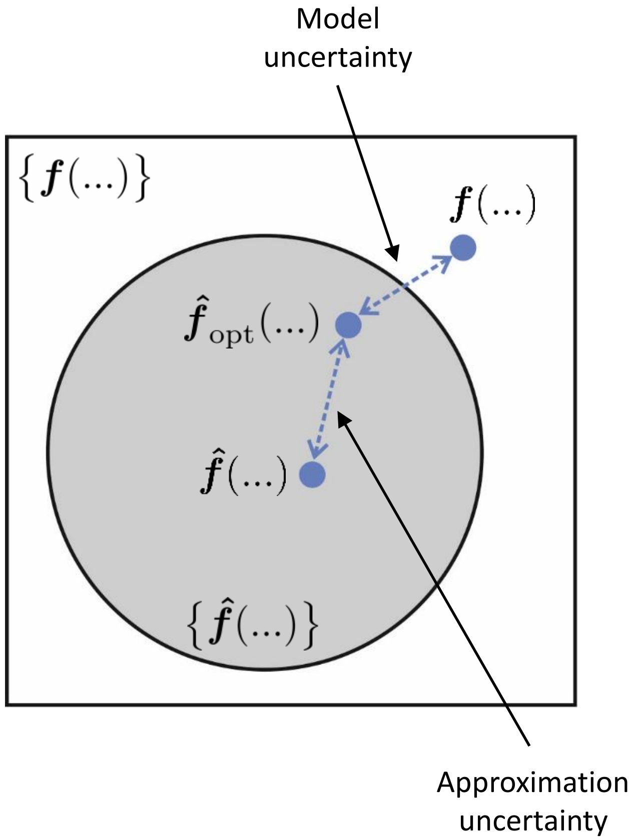


Uncertainties in Machine Learning – Different Types of Uncertainties

Some notation – continued:

- The posterior distribution of the model is updated using the training data to reflect the uncertainty about the model. The difference between the true underlying relationship ***(ground truth)*** $\boldsymbol{f}(\ldots)$ and the posterior distribution ***(best possible)*** $\hat{\boldsymbol{f}}_{\text{opt}}(\ldots)$ of the model is the ***model uncertainty***.

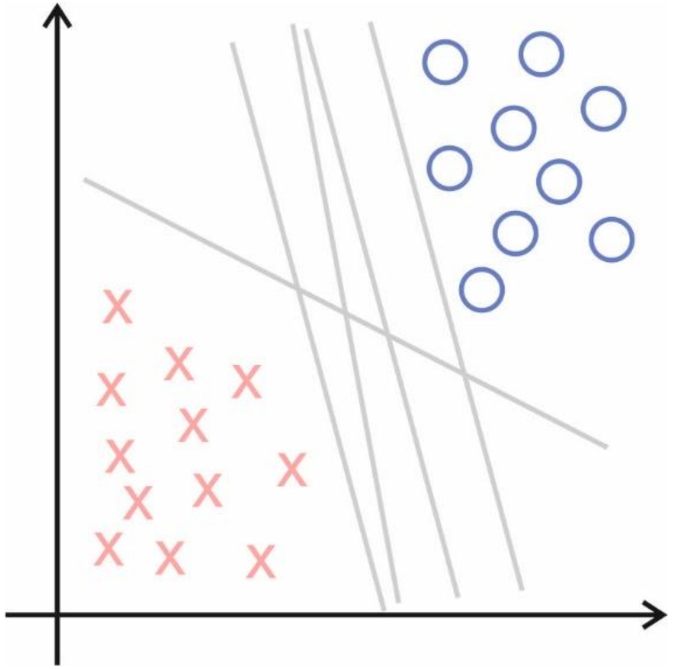
- The risk function is an estimate of the error of the model on new data. The empirical risk funktion is an estimate of the risk function that is calculated using the training data. The difference between the empirical risk minimizer $\hat{\boldsymbol{f}}(\ldots)$ ***(trained predictor)*** and the true risk minimizer $\hat{\boldsymbol{f}}_{\text{opt}}(\ldots)$ ***(best possible)*** is the ***approximation uncertainty***.


Model uncertainty
$\{\boldsymbol{f}(\ldots)\}$
$\boldsymbol{f}(\ldots)$
$\hat{\boldsymbol{f}}_{\text{opt}}(\ldots)$
$\hat{\boldsymbol{f}}(\ldots)$
$\{\hat{\boldsymbol{f}}(\ldots)\}$
Approximation uncertainty

Uncertainties in Machine Learning – Types of Uncertainty

Epistemic uncertainty:

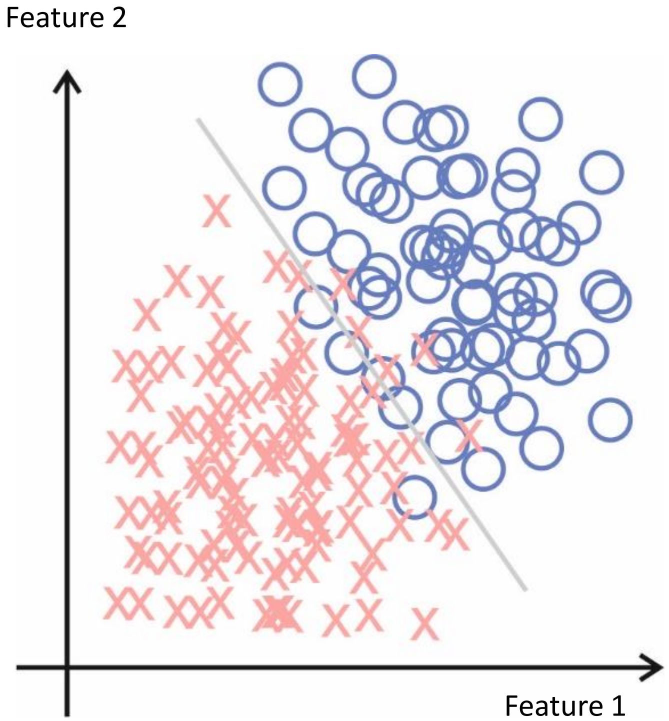
- ❑ ***Epistemic uncertainty*** (systematic uncertainty) is the uncertainty that arises from things that we could know in principle but do not in practice.
- ❑ Epistemic uncertainty arises from a ***lack of knowledge*** and can be reduced by acquiring more data.



Uncertainties in Machine Learning – Types of Uncertainty

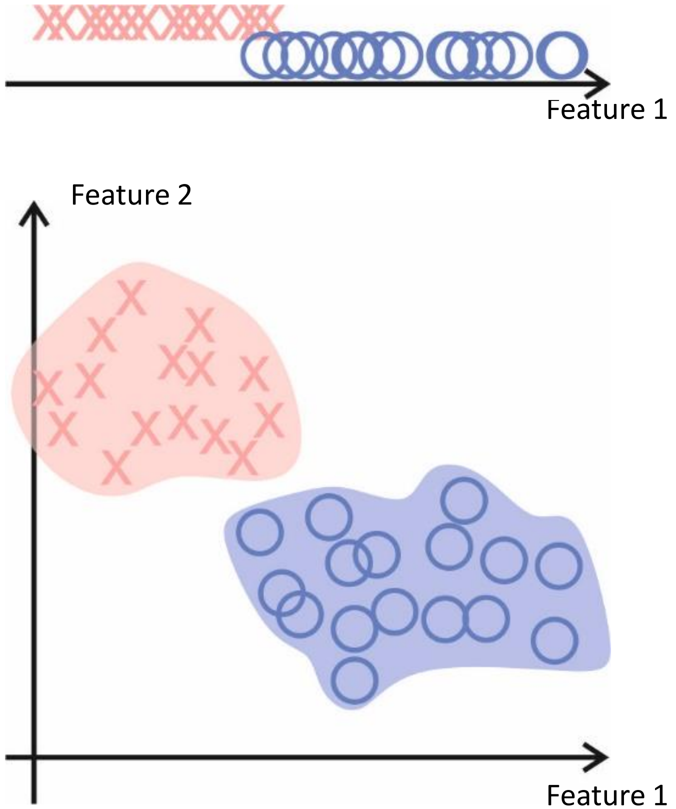
Aleatoric uncertainty:

- ❑ ***Aleatoric uncertainty*** (statistical uncertainty) is the uncertainty that arises from the inherent ***randomness*** of the system being modeled.
- ❑ This uncertainty can be modeled using ***probability distributions***.
- ❑ Aleatoric uncertainty cannot be reduced by collecting more data.



Uncertainties in Machine Learning – Types of Uncertainty

Aleatoric uncertainty:



- ❑ Increasing the dimension of the features can be an effective way to reduce aleatoric uncertainty.

- ❑ It is important to be aware of the ***limitations*** of this approach.
  - ❑ Increasing the dimension of features also requires more data to train the model.
  - ❑ The amount of data required to reduce aleatoric uncertainty by increasing dimension of features depends on complexity of the system being modeled.
  - ❑ Reduction of aleatoric uncertainty can lead to higher epistemic uncertainty.

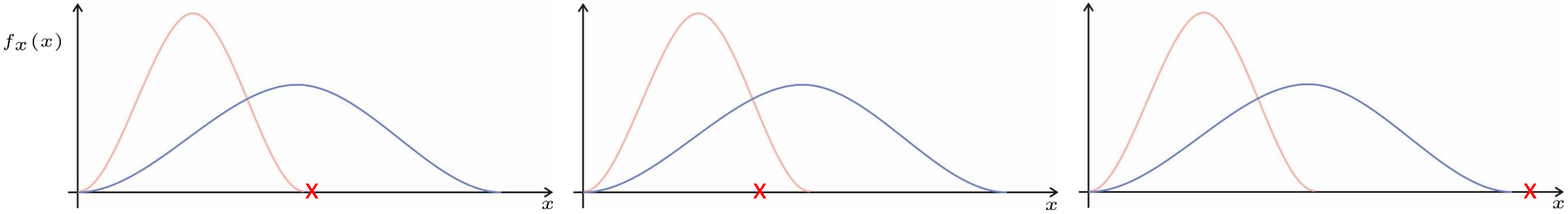
Uncertainties in Machine Learning – Types of Uncertainty

Estimation of Uncertainties – Part 1

- One way to quantify the uncertainties is to estimate probability densities of the individual data points per class.
- The probability densities can be used to predict the class of a new data point. For new data, there are three possible cases:
  1. One probability is high and the others are low: The classification is unambiguous.
  2. All probabilities are high: ***Aleatoric uncertainty*** is high.
  3. All probabilities are low: ***Epistemic uncertainty*** is high.

x: new data point

Classification is unambiguous ***High aleatoric uncertainty*** ***High epistemic uncertainty***

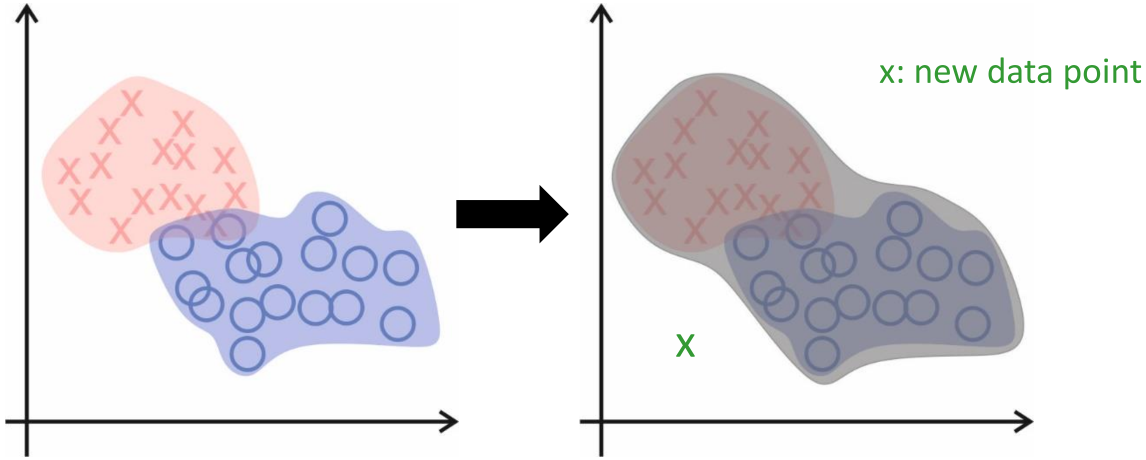


Estimation of Uncertainties – Part 2

Estimation of Uncertainties – Part 1

- ❑ Second possibility is to train a common model for all data points in the space of features → ***Gaussian Mixture Model***.
- ❑ If the probability is low that new data lie within this model, the ***epistemic uncertainty is high***.
- ❑ For aleatoric uncertainty, a model is trained that classifies the different classes.
- ❑ If the probability that new data belong to a class is approximately the same for all classes, ***aleatoric uncertainty is high***.

High epistemic uncertainty



Contents

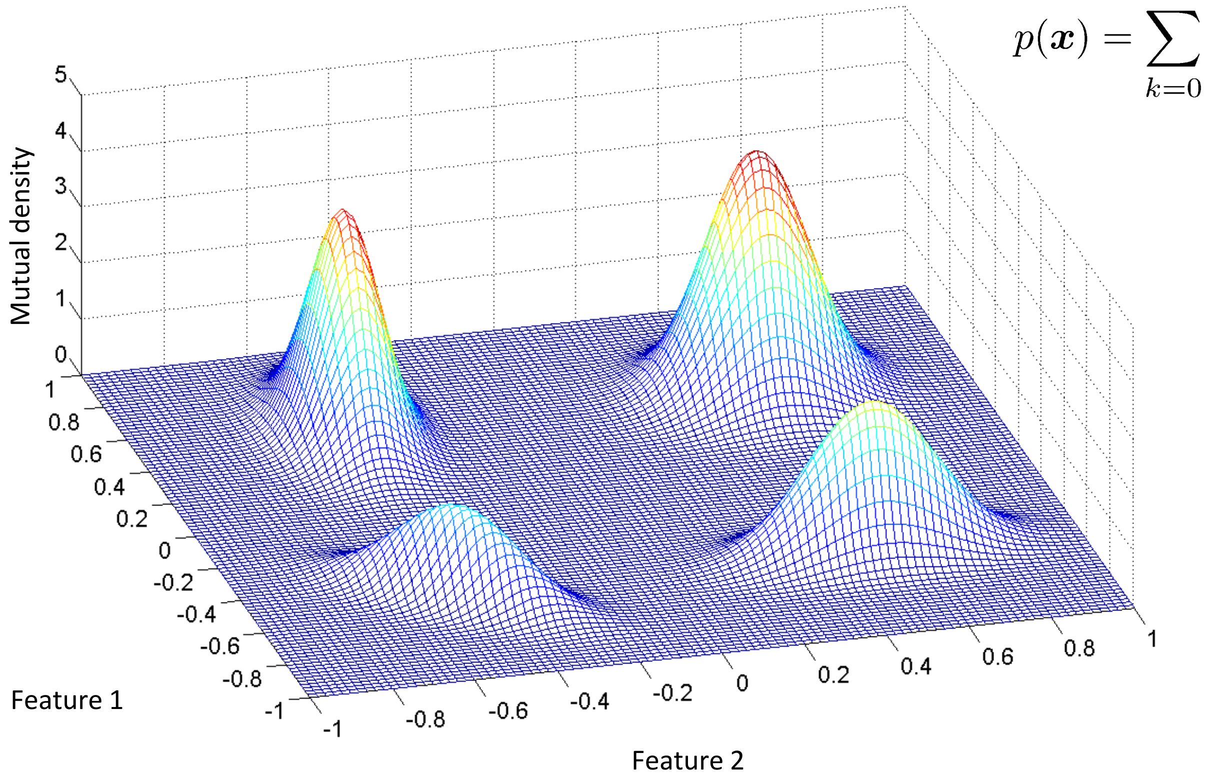
- ❑ Motivation
- ❑ Uncertainties in Machine Learning
- ❑ ***Fundamentals***
  - ❑ ***Gaussian Mixture Models in practice***
  - ❑ ***Generation of Gaussian Mixture Models***
- ❑ Applications in speech and audio processing
  - ❑ Signal separation
  - ❑ Speaker recognition

Gaussian Mixture Models (GMMs)

Christian-Albrechts-Universität zu Kiel

Multivariate Gaussian Distributed Probability Densities – Part 2

Example:

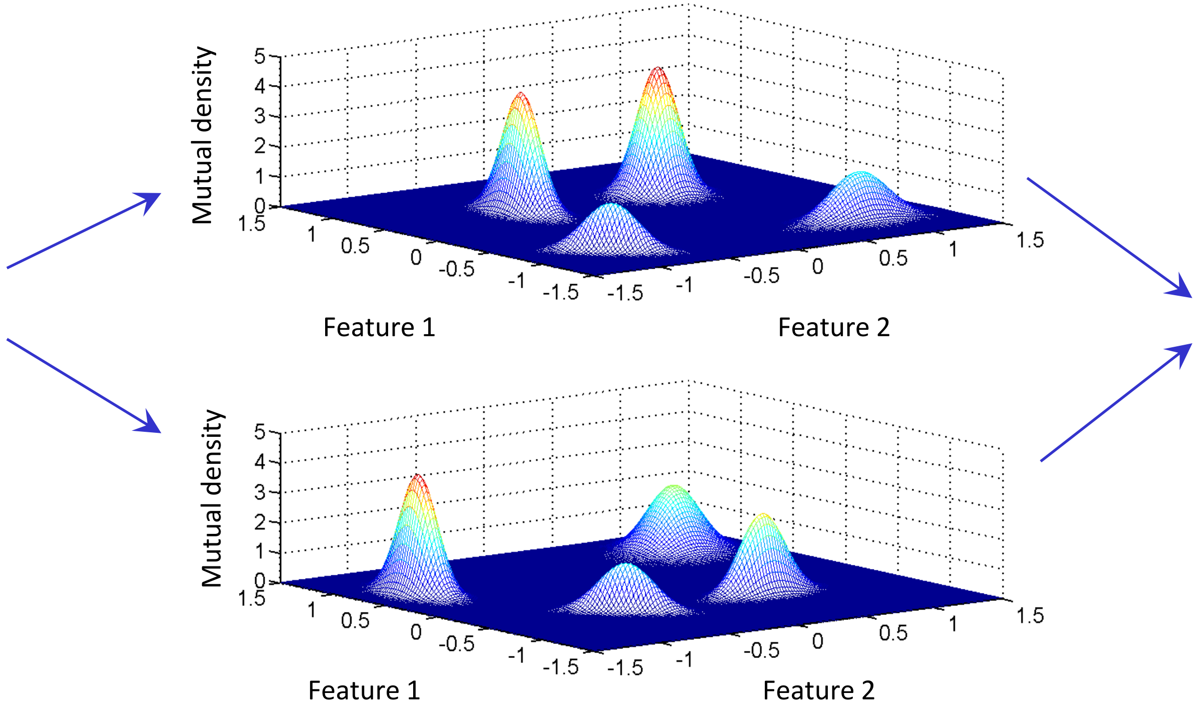


$$p(\boldsymbol{x}) = \sum_{k=0}^{3} g_k \, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \, \boldsymbol{\Sigma}_k\big)$$

Gaussian Mixture Models (GMMs)

Recognition using Statistical Models

Decision (1 out of N):



Model of probability densities
(trained based on data of the first hypothesis)

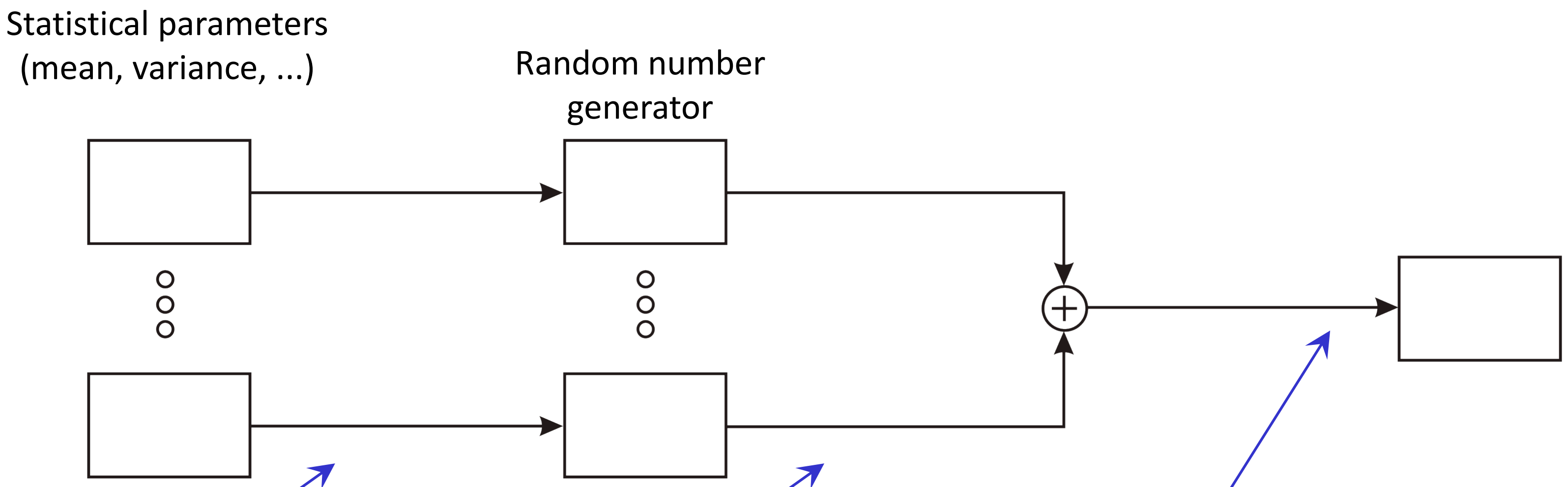
Observed data

Decision

Model of probability densities
(trained based on data of the second hypothesis)

Gaussian Mixture Models (GMMs)

Fundamentals – Generation of Feature Data



All statistical properties are known

For each class, the observed data is known. The statistical properties can be estimated.

The data is known, but a unique classification is no longer possible. The estimation of statistical properties is more difficult.

The Core of the EM Algorithm – Part 1

Cost function:

- For a given ***set of feature vectors***

$$\boldsymbol{X} = \big[\boldsymbol{x}(0),\, \boldsymbol{x}(1),\, ...,\, \boldsymbol{x}(N-1)\big]$$

a ***multivariate density model*** is to be parameterized in such a way that the ***observation probability of the feature vectors*** defined by this model is being ***maximized***:

$$p\big(\boldsymbol{X}|\boldsymbol{g},\boldsymbol{\mu},\boldsymbol{\Sigma}\big) = \prod_{n=0}^{N-1}\left[\sum_{k=0}^{K-1} g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)\right] \longrightarrow \max.$$

Alternatively, the ***logarithmic probability*** (which is a monotonically increasing function in the range (0; 1]) can be maximized:

$$\ln p\big(\boldsymbol{X}|\boldsymbol{g},\boldsymbol{\mu},\boldsymbol{\Sigma}\big) = \sum_{n=0}^{N-1}\ln\left\{\sum_{k=0}^{K-1} g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)\right\} \longrightarrow \max.$$

The Core of the EM Algorithm – Part 2

Model adaption – the E step (E = expectation):

- ❑ Assumption: An existing ***model is to be improved***. To do so, for each feature vector the ***classification probability*** to each Gaussian curve is being calculated:

$$\gamma\big(z_k(n)\big) = \frac{g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{j=0}^{K-1} g_j\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_j,\,\boldsymbol{\Sigma}_j\big)}.$$

Model adaption – the M step (M = maximization):

- ❑ Adaption of the ***mean values***:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\,\boldsymbol{x}(n)}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}.$$

The Core of the EM Algorithm – Part 3

Model adaption – the M step (M = maximization):

- Adaption of the covariance matrices:

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big) \left(\boldsymbol{x}(n) - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\boldsymbol{x}(n) - \boldsymbol{\mu}_k^{\text{new}}\right)^{\text{T}}}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}.$$

- Adaption of the weights:

$$g_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}{N}.$$

The Core of the EM Algorithm – Part 4

Abort condition:

- If the ***new overall probability*** of the feature space ***is only slightly increased,*** then ***break***:

$$\sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{\text{new}}\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{\text{new}},\,\boldsymbol{\Sigma}_k^{\text{new}}\big)\right\} \;<\; \sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{\text{old}}\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{\text{old}},\,\boldsymbol{\Sigma}_k^{\text{old}}\big)\right\} + \epsilon.$$

A Difficulty of the EM Algorithm – Part 1

„Pitfalls" of the cost function:

- When trying to maximize the cost function

$$\ln p\big(\boldsymbol{X}|\boldsymbol{g},\boldsymbol{\mu},\boldsymbol{\Sigma}\big) \;=\; \sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)\right\} \longrightarrow \max$$

  some ***problems*** are likely to appear, if only ***a few feature vectors*** are assigned to one specific class.

- An ***example***:

  For simplicity, we assume we assume ***diagonal covariance matrices*** with a fixed value on the diagonal:

$$\boldsymbol{\Sigma}_k \;=\; \sigma_k^2\,\boldsymbol{I}.$$

  Additionally assume that ***one of the feature vectors is exactly at the mean value*** of one Gaussian curve:

$$\boldsymbol{x}(n_0) \;=\; \boldsymbol{\mu}_{k_0}.$$

A Difficulty of the EM Algorithm – Part 2

„Pitfalls“ of the cost function:

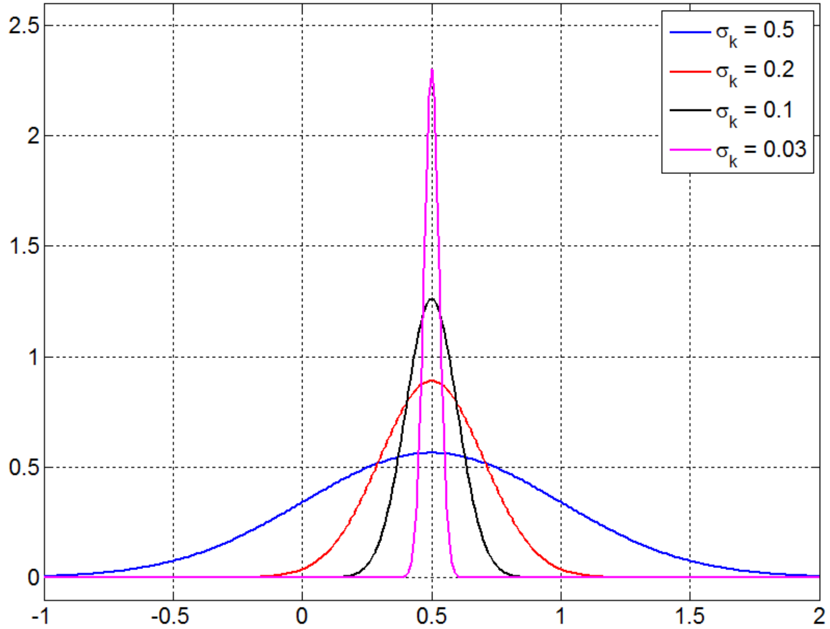
- An ***example*** (continued):

  If the ***variances*** of these Gaussian curves ***tended to zero***,

  $$\sigma_k^2 \longrightarrow 0,$$

  the ***contribution of this Gaussian curve*** at the model evaluation at the end of the adaption would ***tend to infinity*** – and the optimization goal would be reached.

- In order to avoid this case, the variances (i.e., the diagonal entries of the covariance matrices) usually are ***limited to lower bounds***. In doing so, the ***minimal „widths“ of the Gaussian curves*** can be defined.



Codebooks versus GMMs – Part 1

Model adaption – the E step:

- ❑ Assumption: An existing model is to be improved. To do so, for each feature vector the classification probability to each Gaussian curve is being calculated:

$$\gamma\big(z_k(n)\big) = \frac{g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{j=0}^{K-1} g_j\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_j,\,\boldsymbol{\Sigma}_j\big)}.$$

Instead of the „soft“ classification of the feature vectors in the case of GMMs, the codebook training uses a „hard“ classification (i.e., each feature vector is allocated to exactly one class).

The distance function is similar, if the same covariance matrices and weights are used for all classes.

Model adaption – the M step:

- ❑ Adaption of the mean values:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\,\boldsymbol{x}(n)}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}.$$

The adaption of the mean value is similar in codebook training (but using binary class allocations).

Codebooks versus GMMs – Part 2

Model adaption – the M step:

- Adaption of the covariance matrices:

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\, \big(\boldsymbol{x}(n) - \boldsymbol{\mu}_k^{\text{new}}\big)\, \big(\boldsymbol{x}(n) - \boldsymbol{\mu}_k^{\text{new}}\big)^{\mathrm{T}}}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}.$$

The sum of the diagonal elements (trace) of the covariance matrices is used for the evaluation in codebook training. Especially, the sum of the traces of all classes is used (again assuming a binary class allocation).

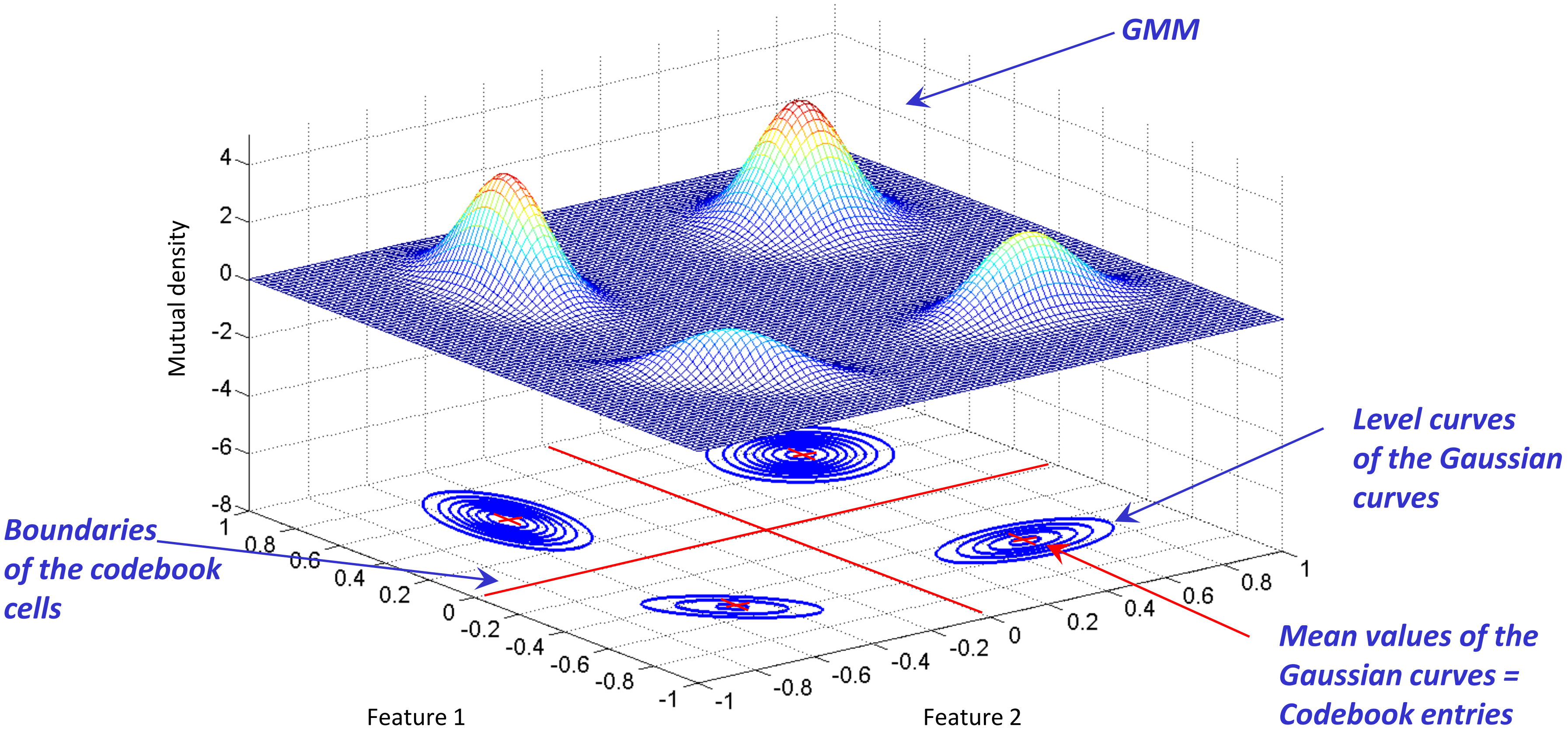
- Adaption of the weights:

$$g_k^{\text{new}} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}{N}.$$

If a MAP-based (MAP means "maximum a posteriori") cost function is chosen, then the same weighting can be used for the codebook (again assuming a binary class allocation). Otherwise, for codebooks all weights are chosen to be the same.

Codebooks versus GMMs – Part 3

Combined diagram:



Latent (Hidden) Random Variables – Part 1

Gaussian mixture models:

- Assume the following probability density function:

$$p(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_k \, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \, \boldsymbol{\Sigma}_k\big).$$

- The generation of data based on this model can be interpreted as a ***two-stage process***:
- In a first step, a Gaussian curve is randomly chosen to generate a random vector. The Gaussians are chosen using the probabilities g_k. In the following, the ***latent („hidden“) random variable***

$$\boldsymbol{z} = \big[z_0, \, \ldots, \, z_{K-1}\big]^{\mathrm{T}}$$

  will be introduced.
- In a second step, a random vector is generated based on one single Gaussian:

$$\mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \, \boldsymbol{\Sigma}_k\big) \, .$$

Latent (Hidden) Random Variables – Part 2

Properties of the latent random variables:

- The elements of the latent random vector may be either 0 or 1,

$$z_k \in \{0, 1\}.$$

- Only one element of the random vector can have the value 1,

$$\sum_{k=0}^{K-1} z_k = 1.$$

- The probabilities for the random vector entries are

$$p(z_k = 1) = g_k.$$

- Hereby, the „vector probability“ can be described as

$$p(\boldsymbol{z}) = \prod_{k=0}^{K-1} (g_k)^{z_k}.$$

Latent (Hidden) Random Variables – Part 3

Properties of the latent random variables:

- This indirect approach using the additional latent random process leads to the following conditional probabilities:

$$\begin{aligned} p(\boldsymbol{x}|z_k = 1) &= \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big), \\ p(\boldsymbol{x}|\boldsymbol{z}) &= \prod_{k=0}^{K-1} \big[\mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big)\big]^{z_k}. \end{aligned}$$

- Using this definition, the Gaussian mixture model can be described as follows:

$$p(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_k\, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big) = \sum_{\boldsymbol{z}} p(\boldsymbol{z})\, p(\boldsymbol{x}|\boldsymbol{z}).$$

Latent (Hidden) Random Variables – Part 4

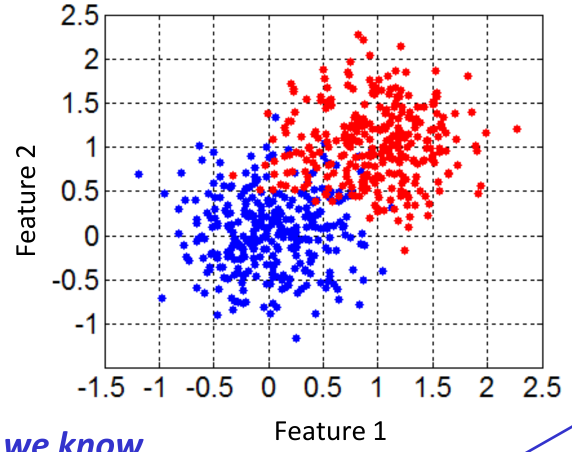
Properties of the latent random variables:

- In order to understand the EM algorithm, we also consider the ***conditional probability*** that a certain class k was active if a certain feature vector was observed:

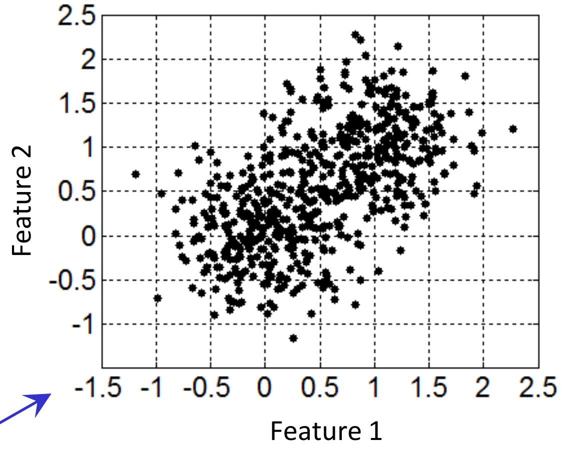
$$\begin{aligned} p(z_k = 1|\boldsymbol{x}) &= \frac{p(z_k = 1)\, p(\boldsymbol{x}|z_k = 1)}{p(\boldsymbol{x})} \\ &= \frac{g_k\, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big)}{\sum\limits_{j=0}^{K-1} g_j\, \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_j,\, \boldsymbol{\Sigma}_j\big)} \\ &= \gamma(z_k). \end{aligned}$$

Latent (Hidden) Random Variables – Part 5

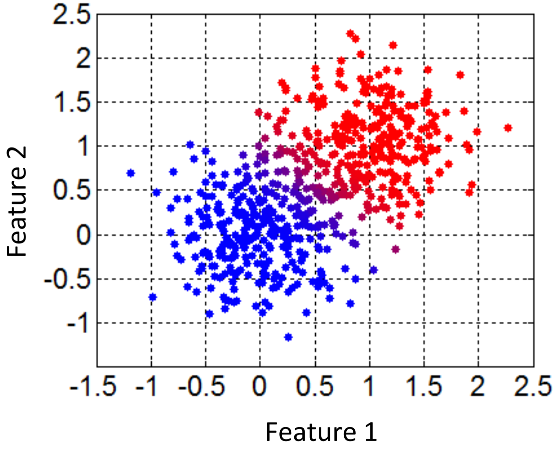
Example:



Measured feature space (assumed we know by which Gaussian the respective feature vectors were generated)



Feature space that can be observed (without knowledge of the generating Gaussians)



Subsequent (soft) assignment of the feature vectors to the (estimated) original distributions (colors according to the conditional probabilities)

The EM Algorithm … Once Again – Part 1

Derivation of the cost function:

$$\frac{d}{d\boldsymbol{\mu}_k} \ln p\big(\boldsymbol{X}|\boldsymbol{g},\boldsymbol{\mu},\boldsymbol{\Sigma}\big) \;=\; \frac{d}{d\boldsymbol{\mu}_k}\sum_{n=0}^{N-1}\ln\left\{\sum_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)\right\}$$

Set the derivation to zero:

$$\begin{aligned} 0 \;&=\; \frac{d}{d\boldsymbol{\mu}_k}\sum_{n=0}^{N-1}\ln\left\{\sum_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)\right\} \\ &=\; \sum_{n=0}^{N-1}\frac{g_k\,\dfrac{d}{d\boldsymbol{\mu}_k}\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)} \\ &=\; \sum_{n=0}^{N-1}\frac{g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)}\boldsymbol{\Sigma}_k^{-1}\big(\boldsymbol{x}(n)-\boldsymbol{\mu}_k\big) \end{aligned}$$

The EM Algorithm … Once Again – Part 2

Result up to now:

$$\mathbf{0} = \sum_{n=0}^{N-1} \frac{g_k \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)}{\sum\limits_{l=0}^{K-1} g_l \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\big)} \boldsymbol{\Sigma}_k^{-1} \big(\boldsymbol{x}(n) - \boldsymbol{\mu}_k\big)$$

… insert the conditional probabilities for the latent variables …

$$\frac{g_k \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)}{\sum\limits_{j=0}^{K-1} g_j \mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big)} = \gamma(z_k)$$

$$\mathbf{0} = \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big) \boldsymbol{\Sigma}_k^{-1} \big(\boldsymbol{x}(n) - \boldsymbol{\mu}_k\big)$$

… multiply with the covariance matrix and separate the terms in brackets …

$$\boldsymbol{\mu}_k \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big) = \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big) \boldsymbol{x}(n)$$

Gaussian Mixture Models (GMMs)

The EM Algorithm … Once Again – Part 3

Result up to now:

$$\boldsymbol{\mu}_k \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big) \;=\; \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big)\, \boldsymbol{x}(n)$$

$$\boxed{\boldsymbol{\mu}_k \;=\; \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\, \boldsymbol{x}(n)}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}}$$

A similar approach leads to the new covariance matrix:

$$\mathbf{0} \;=\; \frac{d}{d\boldsymbol{\Sigma}_k} \sum_{n=0}^{N-1} \ln\left\{ \sum_{l=0}^{K-1} g_l\, \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\, \boldsymbol{\Sigma}_l\big) \right\}$$

The EM Algorithm … Once Again – Part 4

A similar approach leads to the new covariance matrix:

$$\mathbf{0} \;=\; \frac{d}{d\boldsymbol{\Sigma}_k} \sum_{n=0}^{N-1} \ln\left\{ \sum_{l=0}^{K-1} g_l\, \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\, \boldsymbol{\Sigma}_l\big) \right\}$$

Resolved:

$$\boxed{\boldsymbol{\Sigma}_k = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\,\big(\boldsymbol{x}(n)-\boldsymbol{\mu}_k\big)\,\big(\boldsymbol{x}(n)-\boldsymbol{\mu}_k\big)^{\mathrm{T}}}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)}}$$

Gaussian Mixture Models (GMMs)

The EM Algorithm … Once Again – Part 5

For the calculation of the weights we choose the Lagrange method:

$$0 = \frac{d}{dg_k}\left[\sum_{n=0}^{N-1}\ln\left\{\sum_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)\right\} + \lambda\left(\sum_{l=0}^{K-1} g_l - 1\right)\right]$$

The dervation leads to:

$$0 = \sum_{n=0}^{N-1}\frac{\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)} + \lambda$$

multiply both sides with g_k …

$$\begin{aligned}0 &= \sum_{n=0}^{N-1}\frac{g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{l=0}^{K-1} g_l\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_l,\,\boldsymbol{\Sigma}_l\big)} + \lambda\,g_k\\ &= \sum_{n=0}^{N-1}\gamma\big(z_k(n)\big) + \lambda\,g_k\end{aligned}$$

The EM Algorithm … Once Again – Part 6

Result up to now:

$$0 = \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big) + \lambda\, g_k$$

sum over all k …

$$\begin{aligned} 0 &= \sum_{n=0}^{N-1} \underbrace{\sum_{k=0}^{K-1} \gamma\big(z_k(n)\big)}_{=1} + \lambda \underbrace{\sum_{k=0}^{K-1} g_k}_{=1} \\ 0 &= N + \lambda \\ \lambda &= -N \end{aligned}$$

Insert in equation above:

$$\boxed{g_k = \frac{1}{N} \sum_{n=0}^{N-1} \gamma\big(z_k(n)\big)}$$

Initialize the EM Algorithm

Initialization using the k-means or the LBG algorithm

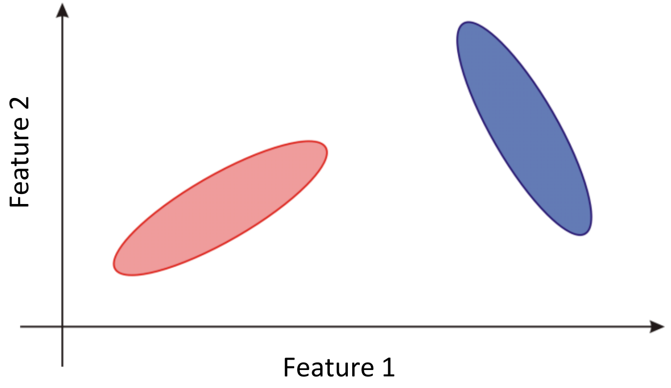
- ❑ In literature, it is often said that the initialization has only low influence on the final converged EM solution. But nevertheless it is supposed to do a „***reasonable***“ initialization.
- ❑ To do so, a codebook is trained on the basis of the feature data. The resulting codebook vectors are used as ***mean vectors*** when the GMM is initialized.
- ❑ Based on all feature vectors that are assigned to a certain codebook vector, an initial value for the ***covariance matrices*** is generated by „averaging“ over all products of the training vectors times their transposed counterparts.
- ❑ Finally, the ratio of the number of feature vectors that are assigned to one codebook entry to the overall number of feature vectors give an initial value of the ***weights***.

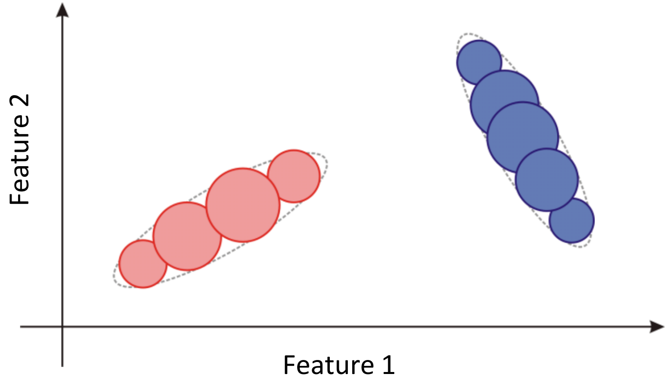
Gaussian Mixture Models (GMMs)

Fully Populated versus Diagonal Covariance Matrices

Complexity reduction:

- ❑ If the covariance matrices are fully populated, the ***computational complexity*** is relatively high (approx. D^2 operations).
- ❑ If the matrix is ***diagonal***, the computational complexity can be ***reduced*** considerably (to approx. $2D$ operations).
- ❑ To achieve the same quality using diagonal matrices, ***more distributions*** are necessary.



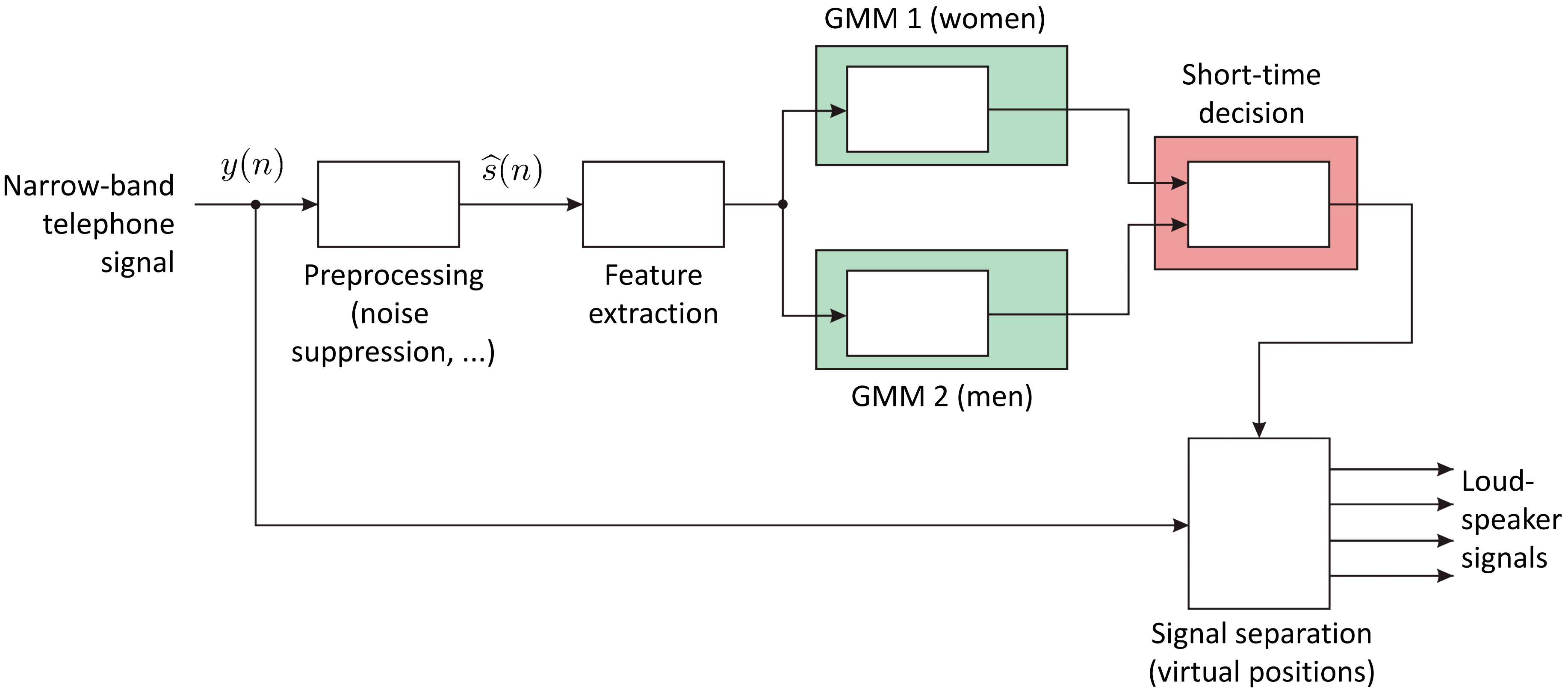


Contents

- ❑ Motivation
- ❑ Uncertainties in Machine Learning
- ❑ Fundamentals
  - ❑ Gaussian Mixture Models in practice
  - ❑ Generation of Gaussian Mixture Models
- ❑ ***Applications in speech and audio processing***
  - ❑ ***Signal separation***
  - ❑ ***Speaker recognition***

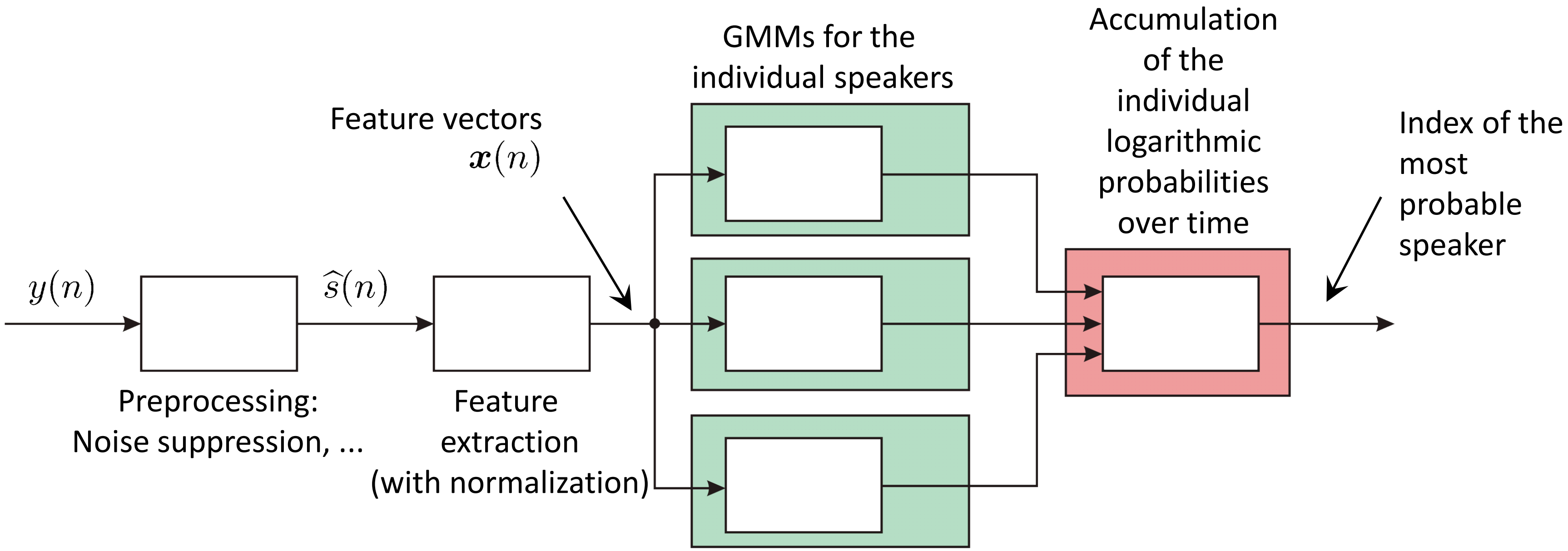
Applications in Speech and Audio Signal Processing: Signal Separation

Spectral envelope estimation using Gaussian mixture models:


GMM 1 (women)
Short-time decision
Narrow-band telephone signal
$y(n)$
$\widehat{s}(n)$
Preprocessing (noise suppression, ...)
Feature extraction
GMM 2 (men)
Loud-speaker signals
Signal separation (virtual positions)

Applications in Speech and Audio Signal Processing: Speaker Recognition

Speaker recognition using Gaussian mixture models:


$y(n)$
$\widehat{s}(n)$
Feature vectors
$\boldsymbol{x}(n)$
GMMs for the individual speakers
Accumulation of the individual logarithmic probabilities over time
Index of the most probable speaker
Preprocessing: Noise suppression, ...
Feature extraction (with normalization)

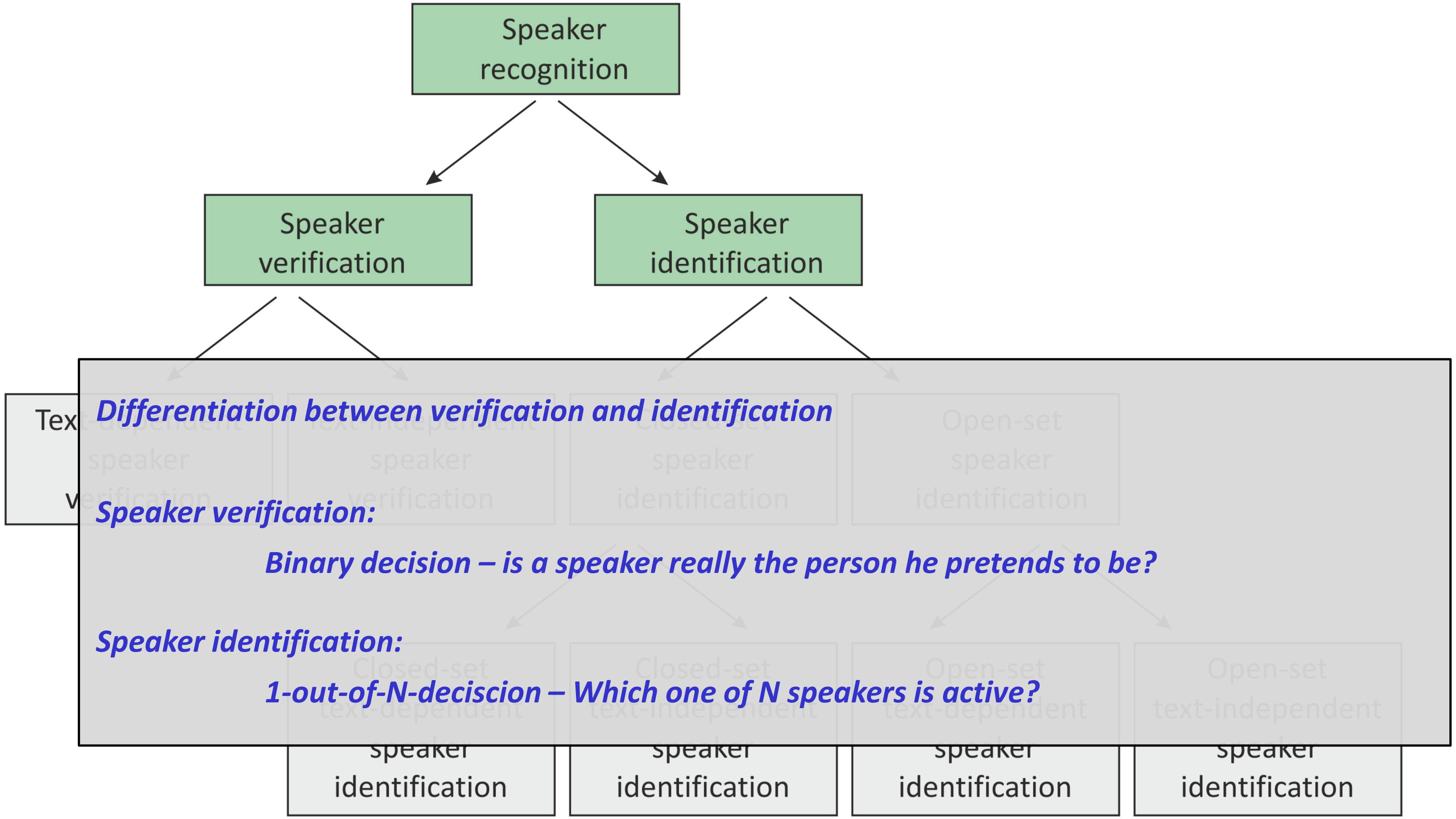
Speaker Recognition – Motivation

Applications for speaker recognition

- ❑ ***Admission control*** (for supplementation of immobilizer systems in cars or admission to protected areas or rooms).
- ❑ ***Personalization*** of speech services (systems recognize the user/caller again and can access preference data bases).
- ❑ ***Improvement of speech signal enhancement schemes*** (e.g., speaker specific signal reconstruction).
- ❑ The post-training (***optimization***) of a ***speech recognition system*** can be done speaker dependent. In the case that a speech dialog system is used randomly by multiple users, the post-training/adaptation of the recognizer can be speaker-dependent

Gaussian Mixture Models (GMMs)

Variants of Speaker Recognition – Part 1


Speaker recognition
Speaker verification
Speaker identification
Text-dependent speaker verification
Text-independent speaker verification
Closed-set speaker identification
Open-set speaker identification
Closed-set text-dependent speaker identification
Closed-set text-independent speaker identification
Open-set text-dependent speaker identification
Open-set text-independent speaker identification

Differentiation between verification and identification

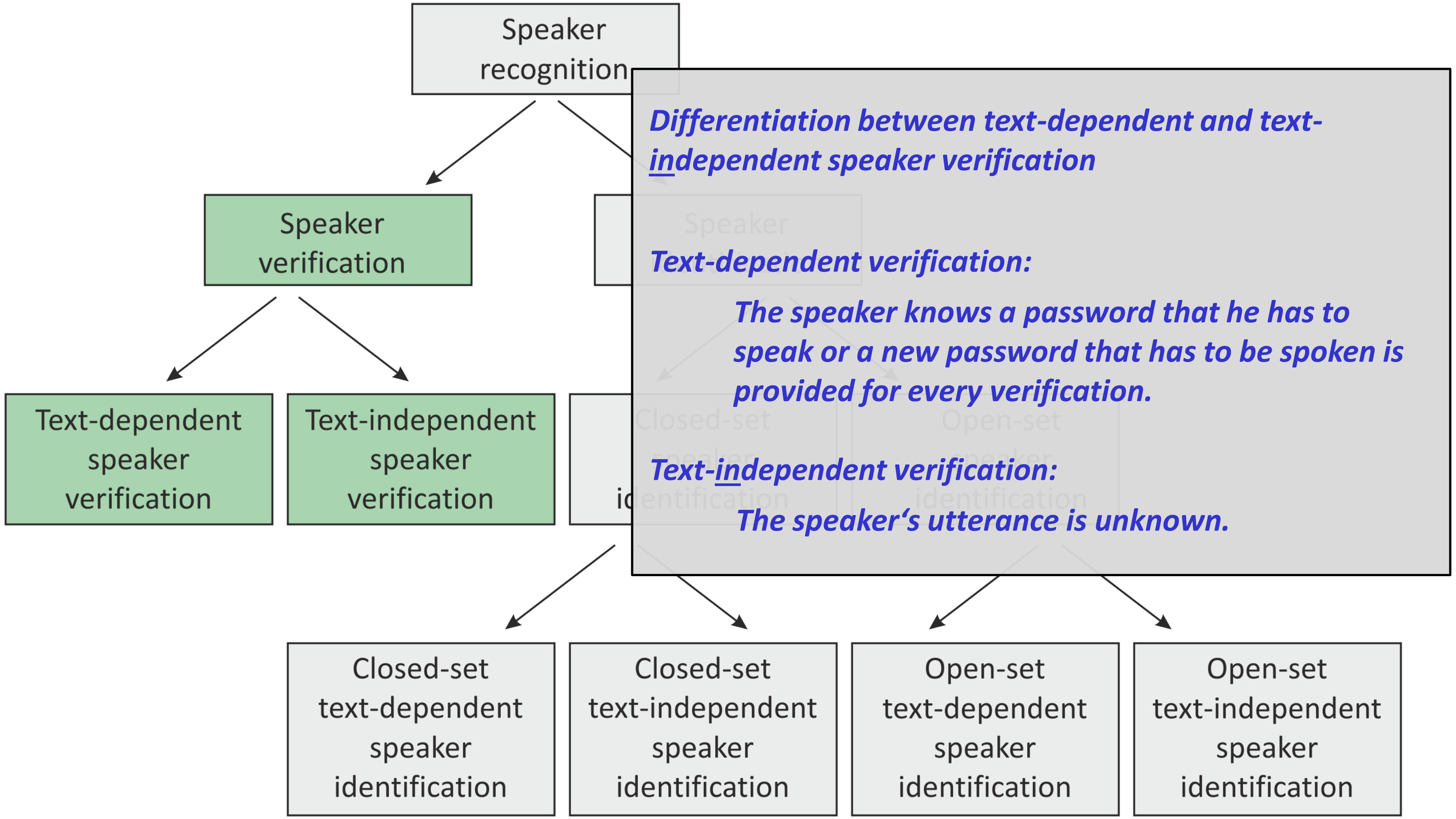
Speaker verification:

Binary decision – is a speaker really the person he pretends to be?

Speaker identification:

1-out-of-N-deciscion – Which one of N speakers is active?

Variants of Speaker Recognition – Part 2


Speaker recognition
Speaker verification
Speaker
Text-dependent speaker verification
Text-independent speaker verification
identification
Closed-set
Open-set
identification
Closed-set text-dependent speaker identification
Closed-set text-independent speaker identification
Open-set text-dependent speaker identification
Open-set text-independent speaker identification

Differentiation between text-dependent and text-independent speaker verification

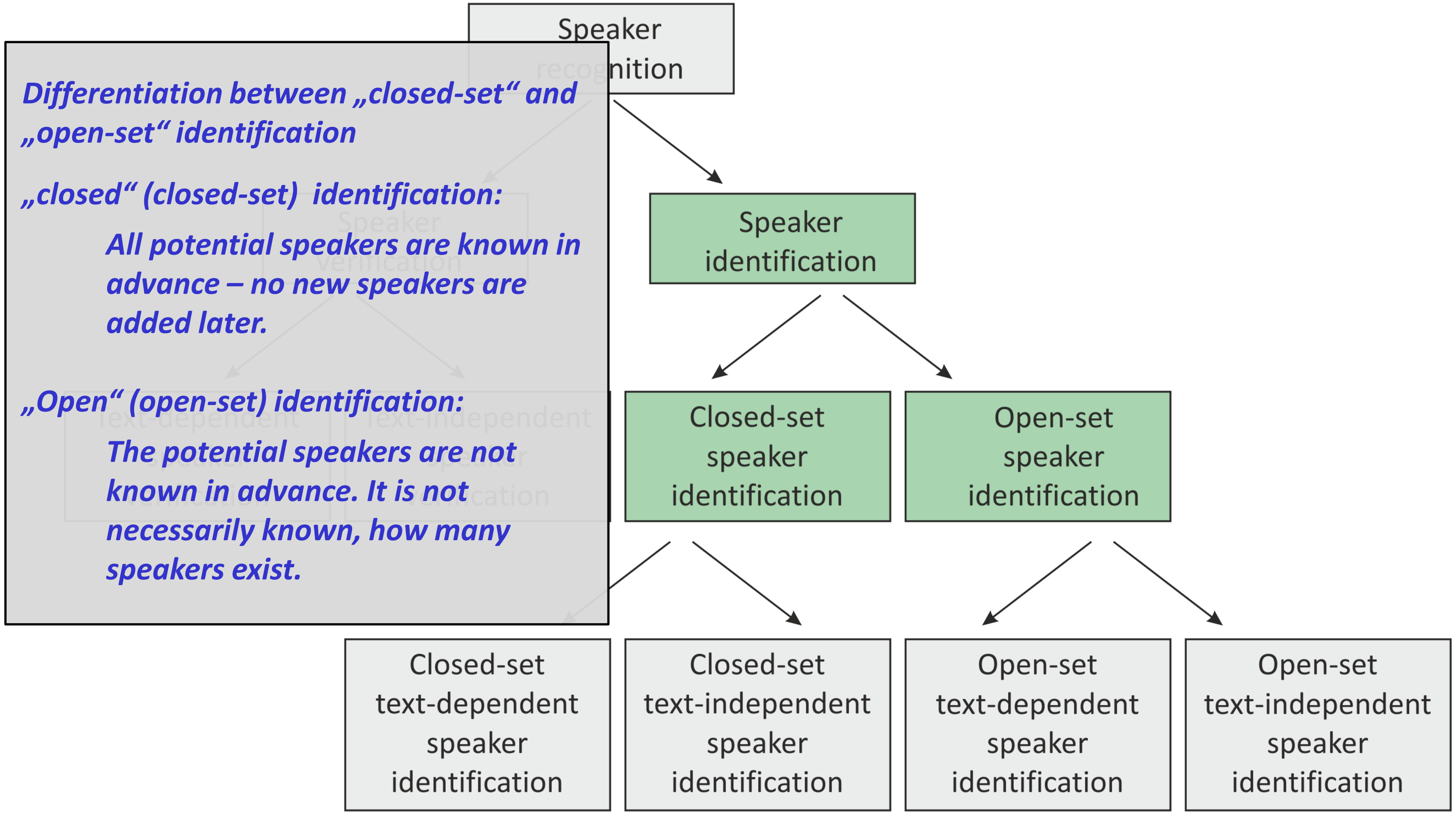
Text-dependent verification:

> ***The speaker knows a password that he has to speak or a new password that has to be spoken is provided for every verification.***

Text-independent verification:

> ***The speaker‘s utterance is unknown.***

Variants of Speaker Recognition – Part 3



Differentiation between „closed-set" and „open-set" identification

„closed" (closed-set) identification:

All potential speakers are known in advance – no new speakers are added later.

„Open" (open-set) identification:

The potential speakers are not known in advance. It is not necessarily known, how many speakers exist.

Speaker recognition

Speaker verification

Text-dependent speaker verification

Text-independent speaker verification

Speaker identification

Closed-set speaker identification

Open-set speaker identification

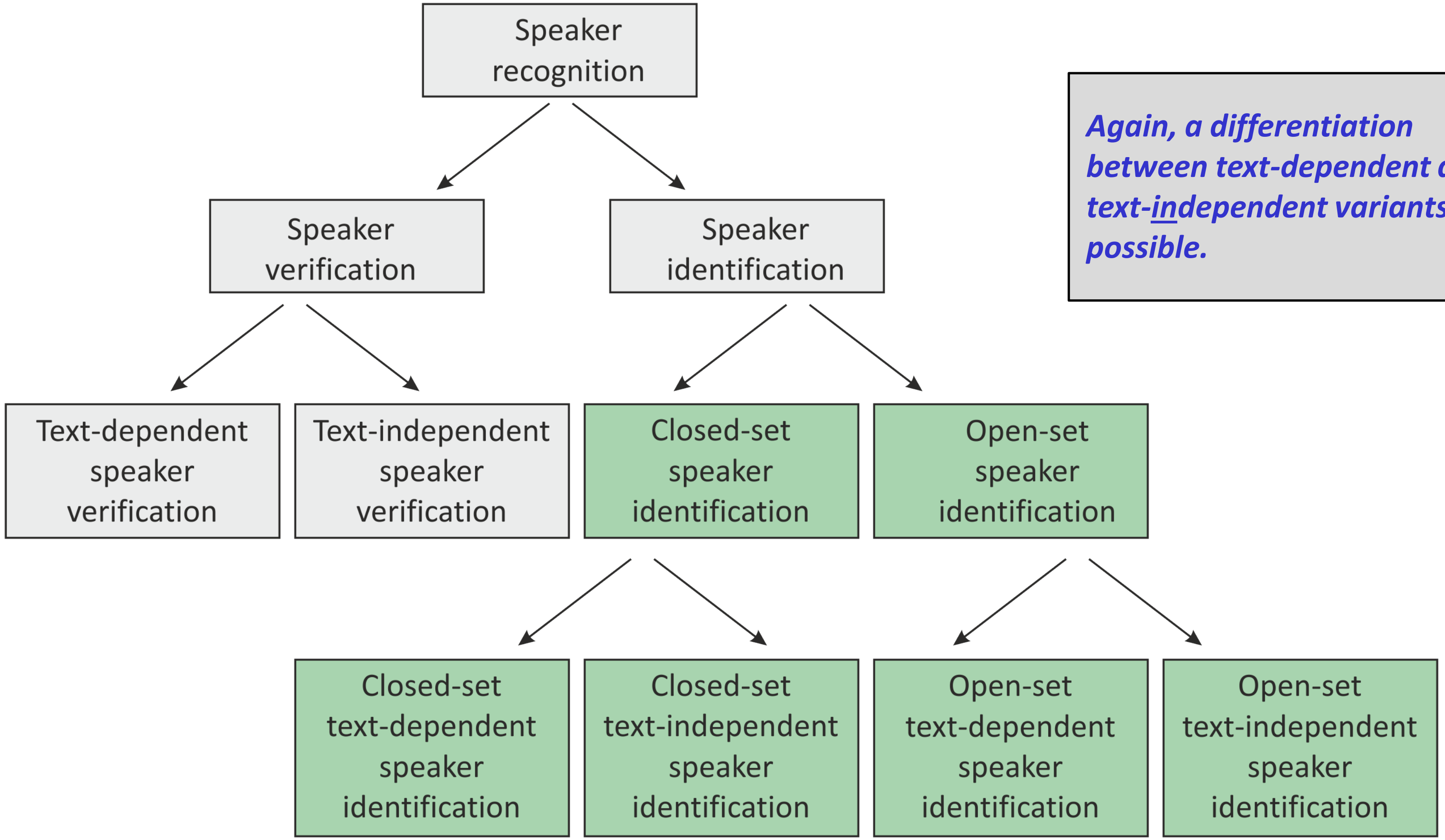
Closed-set text-dependent speaker identification

Closed-set text-independent speaker identification

Open-set text-dependent speaker identification

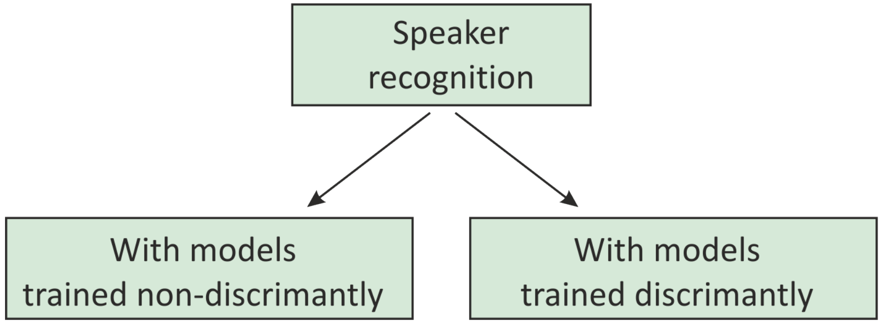
Open-set text-independent speaker identification

Variants of Speaker Recognition – Part 4


Speaker recognition
Speaker verification
Speaker identification
Text-dependent speaker verification
Text-independent speaker verification
Closed-set speaker identification
Open-set speaker identification
Closed-set text-dependent speaker identification
Closed-set text-independent speaker identification
Open-set text-dependent speaker identification
Open-set text-independent speaker identification

Again, a differentiation between text-dependent and text-independent variants is possible.

Variants of Speaker Recognition – Part 5


Speaker recognition
With models trained non-discrimantly
With models trained discrimantly

Differentiation between non-discriminant and discriminant training methods

Non-discriminant training:

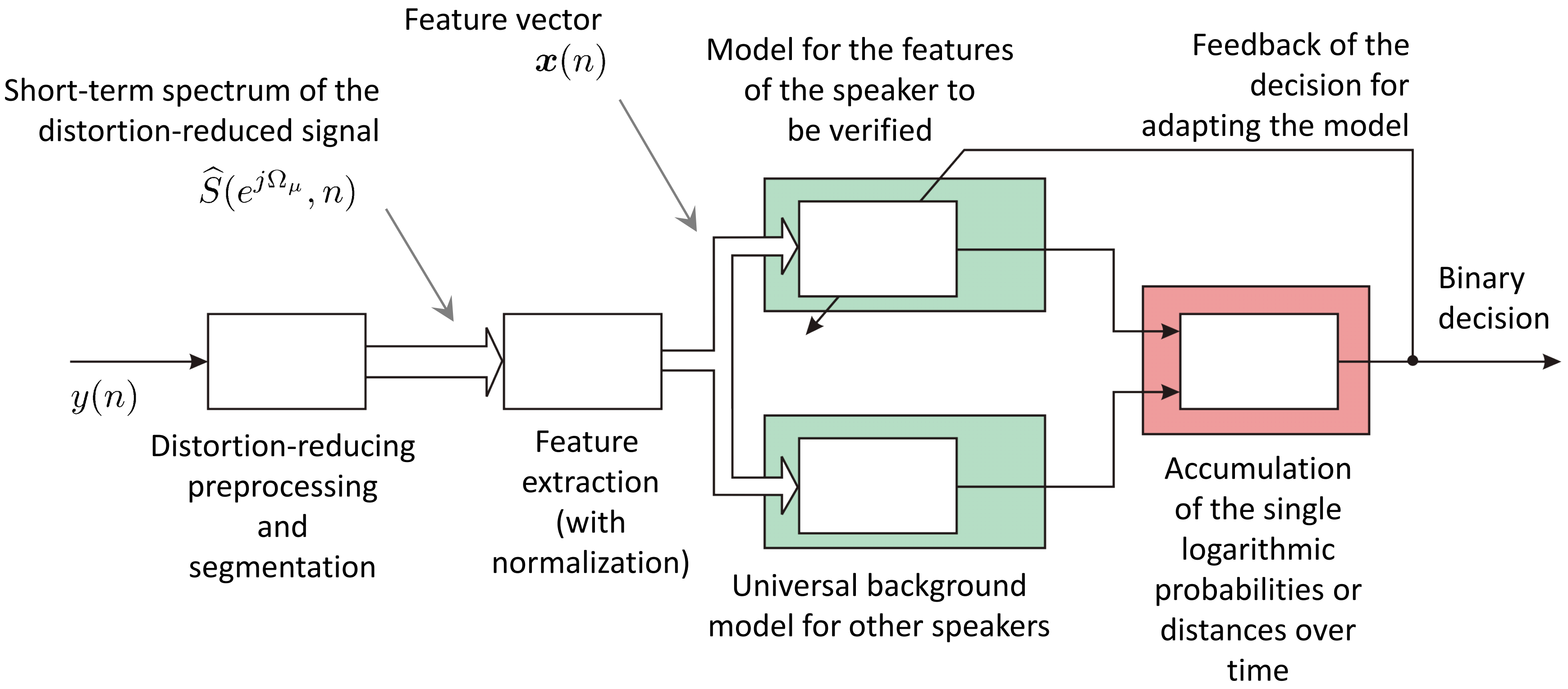
> ***The models are trained for each speaker independently, i.e., the model has to fit to the extracted training data as good as possible – however, a good discrimination of other speakers is not considered.***

Discriminant training:

> ***All speakers are considered during the training of the models to fit the individual models not only to one speaker, but also to learn the differences between the speaker features.***

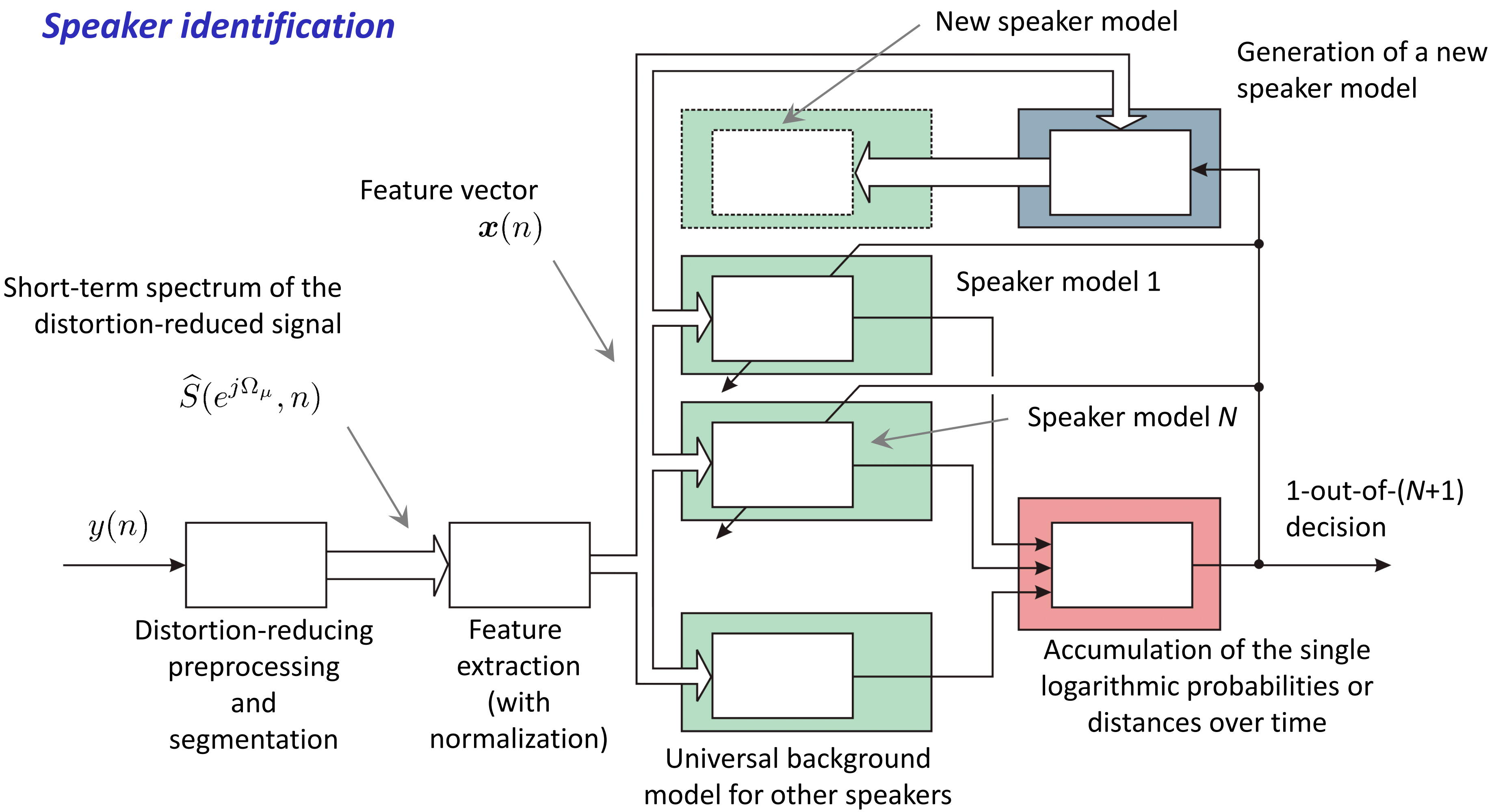
Basics of Speaker Recognition – Part 1

Speaker verification


Feature vector
$\boldsymbol{x}(n)$
Short-term spectrum of the distortion-reduced signal
$\widehat{S}(e^{j\Omega_\mu},n)$
Model for the features of the speaker to be verified
Feedback of the decision for adapting the model
Binary decision
$y(n)$
Distortion-reducing preprocessing and segmentation
Feature extraction (with normalization)
Universal background model for other speakers
Accumulation of the single logarithmic probabilities or distances over time

Basics of Speaker Recognition – Part 2

Speaker identification


New speaker model
Generation of a new speaker model
Feature vector
$\boldsymbol{x}(n)$
Short-term spectrum of the distortion-reduced signal
$\widehat{S}(e^{j\Omega_\mu}, n)$
Speaker model 1
Speaker model *N*
1-out-of-(*N*+1) decision
$y(n)$
Distortion-reducing preprocessing and segmentation
Feature extraction (with normalization)
Universal background model for other speakers
Accumulation of the single logarithmic probabilities or distances over time

Difficulties in Speaker Recognition

Some typical problems…

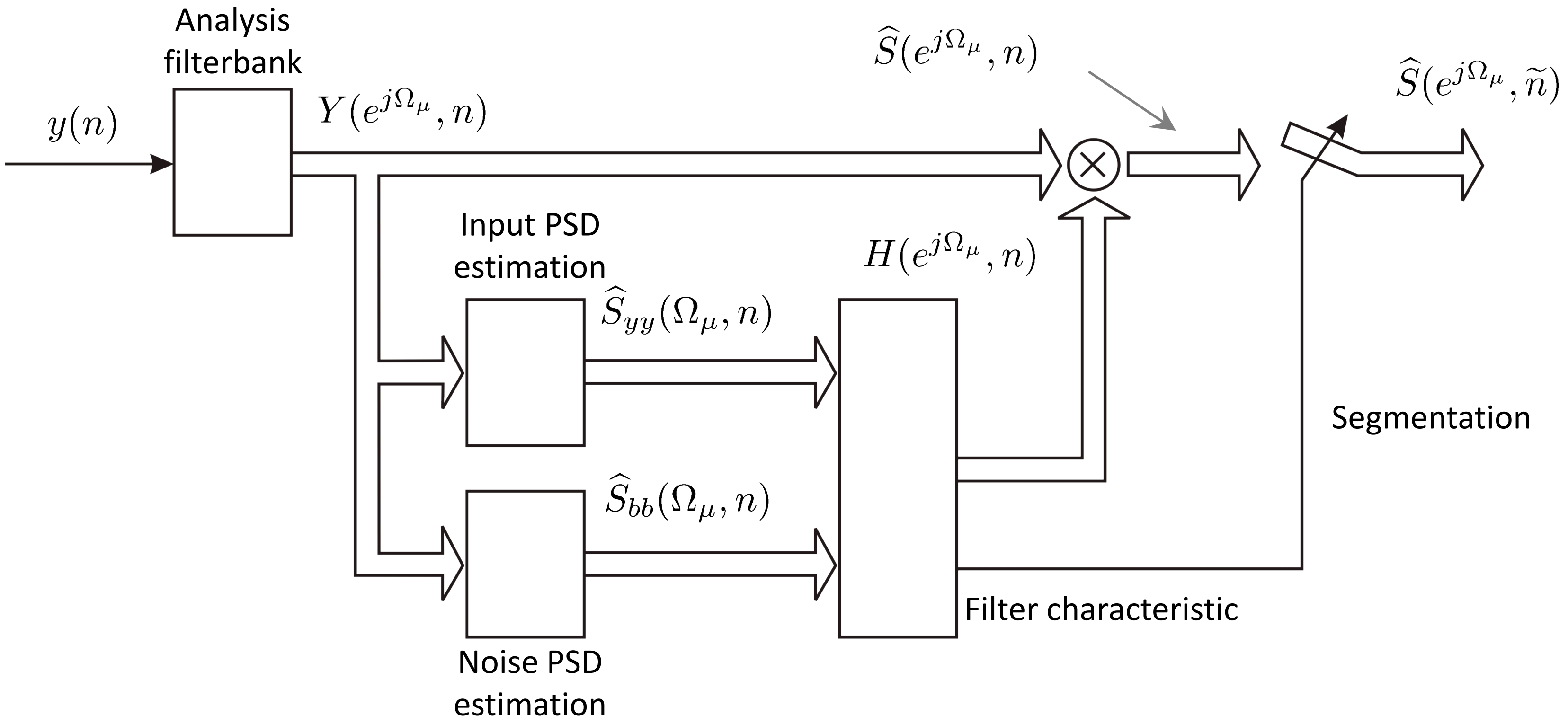
- In many practical applications only a relatively ***small amount of training data*** for the individual speakers is available. Additionally, this training data is often not phonetically ***„balanced“***. During the recognition itself, a ***decision*** should be made as ***fast*** as possible.
- As a consequence, text-independent systems become a strong ***text-dependency***: Speaker A speaks words that are contained in the small training set of speaker B, but not in his own. That probability to identify speaker B is rather high for a small amount of training data.
- It is often reported in literature that preprocessing or ***normalization*** have a negative influence on the recognition rate. This is true if the ***recording conditions*** during training and test match well. However, such a ***match between training and test*** conditions is not always given in practice.
- ***Speech pauses should be removed*** before the recognition task itself. Otherwise, the background noise will have a strong influence on the decision: speakers with similar background noise during recording will be preferred.

Gaussian Mixture Models (GMMs)

Christian-Albrechts-Universität zu Kiel

Speaker Recognition – Preprocessing and Segmentation – Part 1

Subband structure:



PSD abbreviates power spectral density.

Speaker Recognition – Preprocessing and Segmentation – Part 2

Noise reduction:

$$\widetilde{H}(e^{j\Omega_\mu},n) = \max\left\{0,\, 1-\frac{\widehat{S}_{bb}(\Omega_\mu,n)}{\widehat{S}_{yy}(\Omega_\mu,n)}\right\}$$

Noise reduction without limitation of the attenuation (needed for the segmentation)

$$H(e^{j\Omega_\mu},n) = \max\left\{H_{\min},\, \widetilde{H}(e^{j\Omega_\mu},n)\right\}$$

Noise reduction with limitation of the attenuation (needed for the signal enhancement)

Segmentation:

$$a(n) = \begin{cases} 1, & \text{if } \frac{1}{N/2+1}\sum\limits_{\mu=0}^{N/2}\widetilde{H}(e^{j\Omega_\mu},n) \;>\; 0.1\ldots0.3, \\ 0, & \text{else.} \end{cases}$$

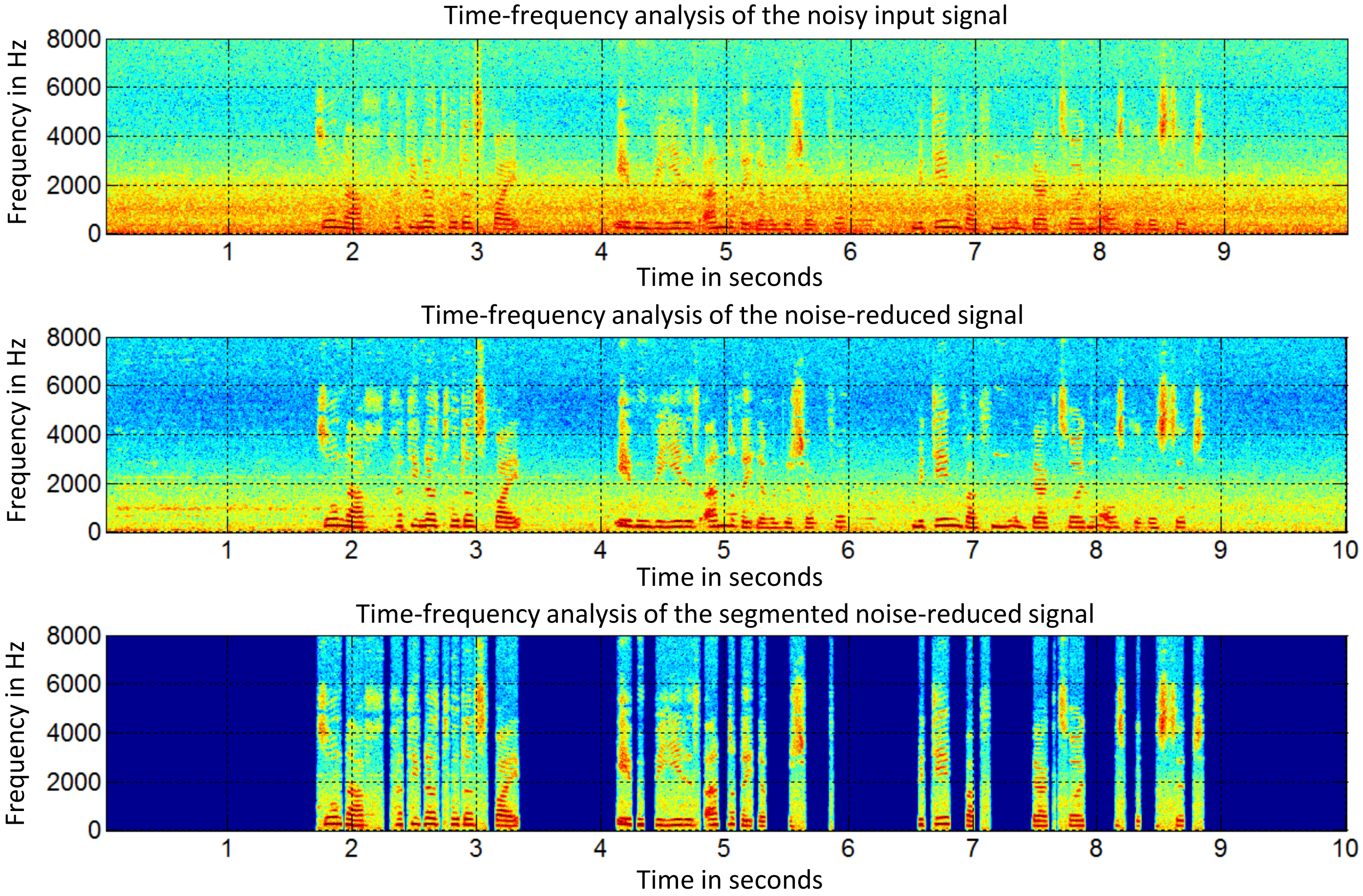
If the noise reduction filter is open in 10…30 percent of all subbands, the current frame is classified to contain speech.

Gaussian Mixture Models (GMMs)

Christian-Albrechts-Universität zu Kiel

Speaker Recognition – Preprocessing and Segmentation – Part 3

Example:


Time-frequency analysis of the noisy input signal

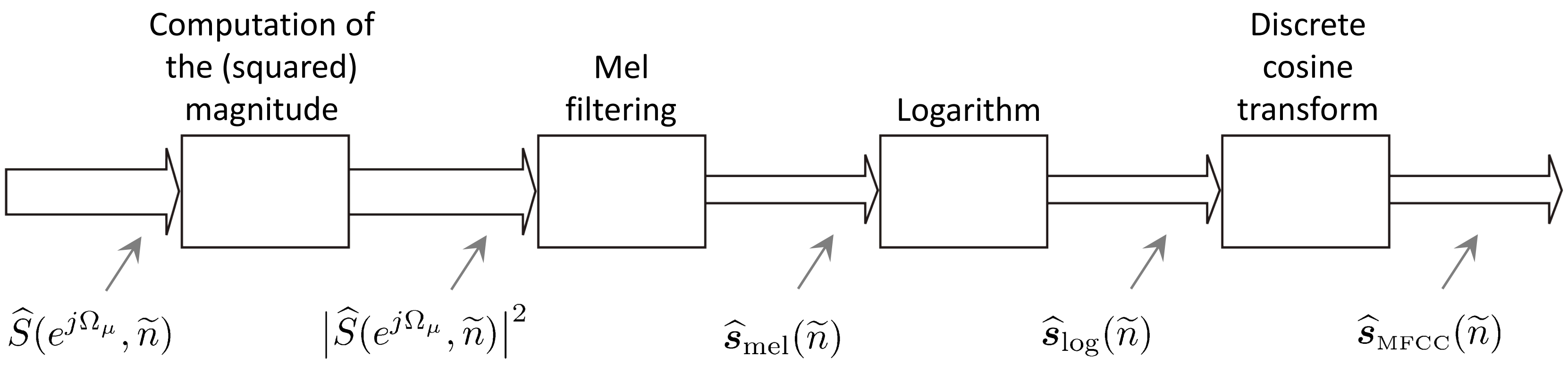
Time-frequency analysis of the noise-reduced signal

Time-frequency analysis of the segmented noise-reduced signal

- ❑ Input signal
- ❑ Signal after noise reduction
- ❑ Signal after segmentation

Speaker Recognition – Feature Extraction – Part 1

Mel-filtered cepstral coefficients (MFCCs):


Computation of the (squared) magnitude
Mel filtering
Logarithm
Discrete cosine transform
$\widehat{S}(e^{j\Omega_\mu},\widetilde{n})$
$\big|\widehat{S}(e^{j\Omega_\mu},\widetilde{n})\big|^2$
$\widehat{\boldsymbol{s}}_{\mathrm{mel}}(\widetilde{n})$
$\widehat{\boldsymbol{s}}_{\log}(\widetilde{n})$
$\widehat{\boldsymbol{s}}_{\mathrm{MFCC}}(\widetilde{n})$

- ❑ The ***first*** (zeroth) ***coefficient*** of the feature vectors is often replaced by the ***normalized short-term power of the current signal frame***.
- ❑ The normalization is done such that the ***maximum short-term power*** of an utterance is mapped to a ***defined value***.

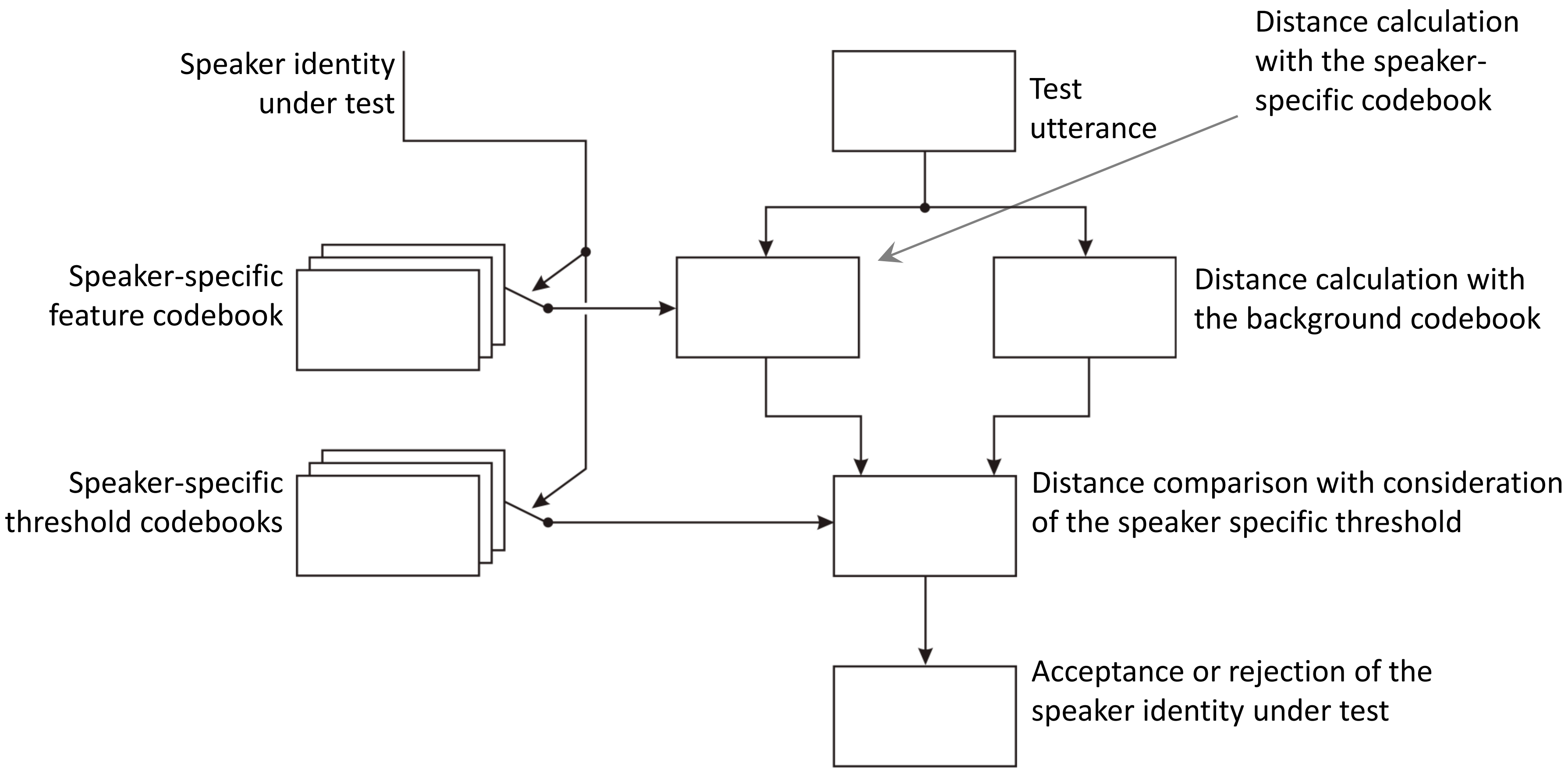
Speaker Recognition – Feature Extraction – Part 2

Some remarks:

- ❑ Many publications deal with the ***selection of features***. The most common conclusion is that a compact representation of the short-term spectral envelope should be used.
- ❑ ***MFCCs*** and ***cepstral coefficients*** (with slight modification) have proven to be useful.
- ❑ It is astonishing that these are the same features that are used for speech recognition. In the application of speech recognition, the interest is to remove differences between speakers to obtain only information about the words that have been spoken.
- ❑ However, it should be mentioned that ***different preprocessing*** is used for ***speaker*** and ***speech recognition***.
- ❑ As a consequence, it can be concluded that a ***speaker-specific speech recognition*** yields better results compared to a non speaker-specific one – this can also be observed in practice. For this reason, it is often desired to ***adapt the models of a speech recognition system*** to the current speaker.

Speaker Recognition With Codebooks – Recognition Phase

Flow chart – speaker verification:

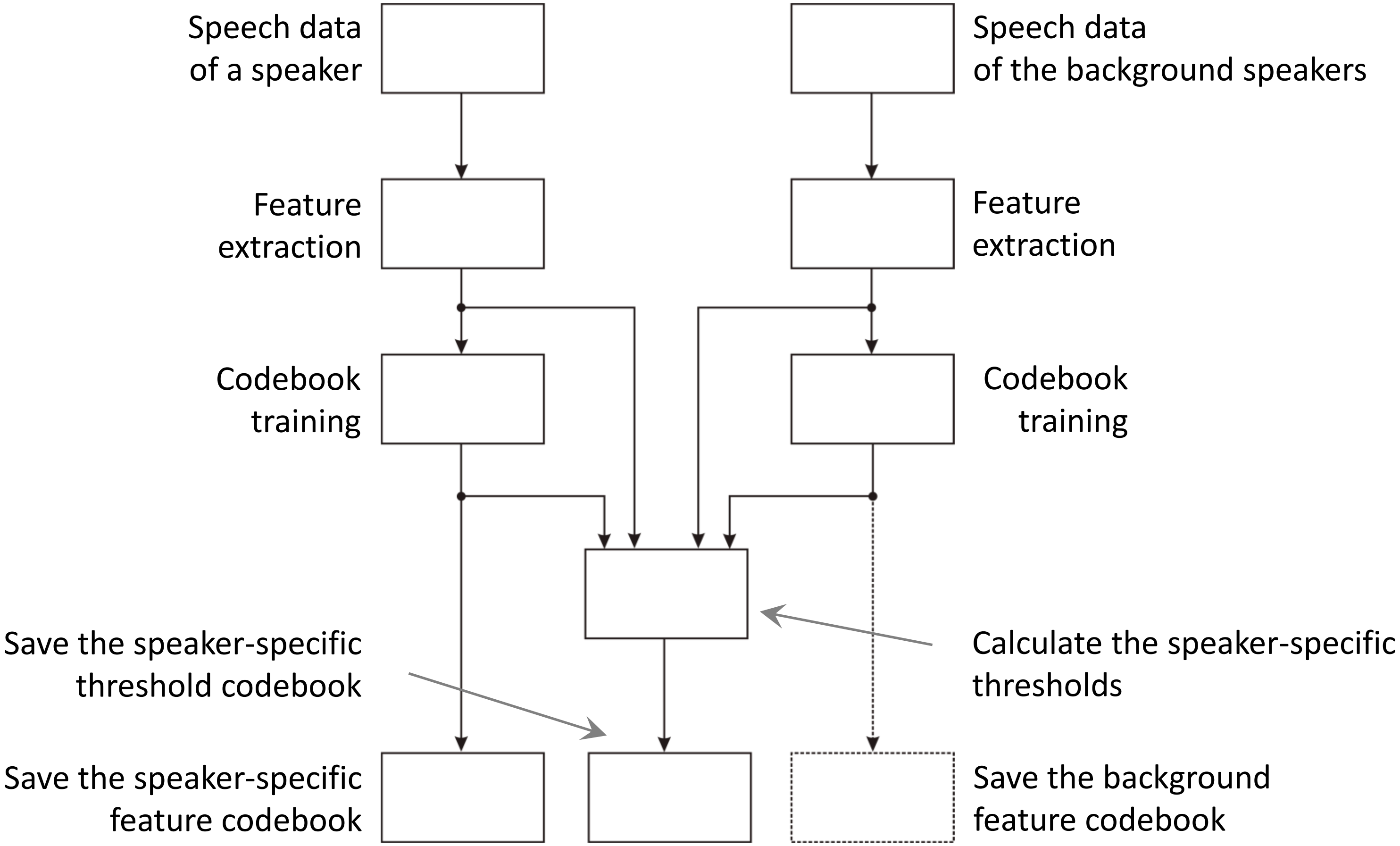

Speaker identity under test
Test utterance
Distance calculation with the speaker-specific codebook
Speaker-specific feature codebook
Distance calculation with the background codebook
Speaker-specific threshold codebooks
Distance comparison with consideration of the speaker specific threshold
Acceptance or rejection of the speaker identity under test

Gaussian Mixture Models (GMMs)

Christian-Albrechts-Universität zu Kiel

Speaker Recognition With Codebooks– Training Phase

Flow chart – speaker verification:



Speaker Recognition With Gaussian Mixture Models – Recognition Phase (Part 1)

Approach of the speaker verification:

- ❑ Pose two hypothesis:

$$
\begin{aligned}
H_0 : &\quad \text{The speaker under test is the same as the desired speaker.} \\
H_1 : &\quad \text{The speaker under test is not the same as the desired speaker.}
\end{aligned}
$$

- ❑ If the same „costs“ for different kinds of errors are assumed, the target and the test speaker are decided to be same person if

$$p(H_0|\boldsymbol{X}) \quad > \quad p(H_1|\boldsymbol{X}).$$

The matrix $\boldsymbol{X}$ contains the feature vectors of the utterance (after noise and speech pauses have been removed).

Speaker Recognition With Gaussian Mixture Models – Recognition Phase (Part 2)

Approach of the speaker verification :

- The conditional probabilities can be re-written as follows:

$$\begin{aligned} p(H_0|\boldsymbol{X}) &= \frac{p(\boldsymbol{X}|H_0)\,p(H_0)}{p(\boldsymbol{X})}, \\ p(H_1|\boldsymbol{X}) &= \frac{p(\boldsymbol{X}|H_1)\,p(H_1)}{p(\boldsymbol{X})}. \end{aligned}$$

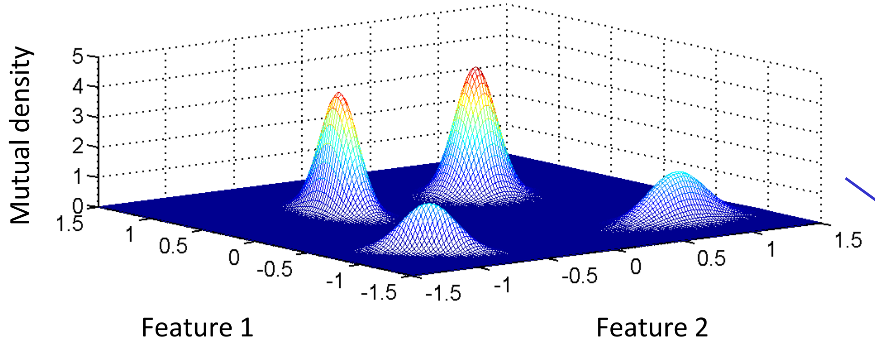
- This yields for our condition:

$$\begin{aligned} p(H_0|\boldsymbol{X}) &> p(H_1|\boldsymbol{X}) \\ p(\boldsymbol{X}|H_0) &> p(\boldsymbol{X}|H_1)\,\frac{p(H_1)}{p(H_0)}. \end{aligned}$$

- Different speaker probabilities can be modeled by the ratio of $p(H_0)$ and $p(H_1)$.

Speaker Recognition With Gaussian Mixture Models – Recognition Phase (Part 3)

Probability density model
(trained on data of hypothesis H_0, i.e. on training data of the target speaker)

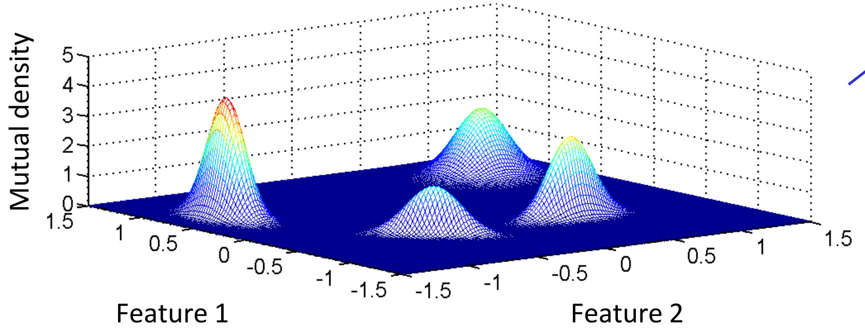


Observed data

Multiplication with the speaker probability

Decision

Multiplication with the complementary speaker probability



Probability density model
(trained on data of hypothesis H_1 , i.e. on training data of non-target speaker(s))

Speaker Recognition With Gaussian Mixture Models – Recognition Phase (Part 4)

Approach of the speaker verification:

- If ***Gaussian mixture models*** are used, the (logarithmic) probability density functions are:

$$\begin{aligned}
\ln p(\boldsymbol{X}|H_0) &= \ln p\big(\boldsymbol{X}|\boldsymbol{g}^{(s)}, \boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}\big) \\
&= \sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{(s)}\, \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{(s)},\, \boldsymbol{\Sigma}_k^{(s)}\big)\right\}, \\
\ln p(\boldsymbol{X}|H_1) &= \ln p\big(\boldsymbol{X}|\boldsymbol{g}^{(b)}, \boldsymbol{\mu}^{(b)}, \boldsymbol{\Sigma}^{(b)}\big) \\
&= \sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{(b)}\, \mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{(b)},\, \boldsymbol{\Sigma}_k^{(b)}\big)\right\}.
\end{aligned}$$

The superscripts $^{(s)}$ and $^{(b)}$ denote the individual speaker and background model, respectively.

Speaker Recognition With Gaussian Mixture Models – Recognition Phase (Part 5)

Approach of the speaker verification :

- The ***decision rule***

$$p(\boldsymbol{X}|H_0) \quad > \quad p(\boldsymbol{X}|H_1)\,\frac{p(H_1)}{p(H_0)}$$

can be re-written as follows:

$$\sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{(s)}\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{(s)},\,\boldsymbol{\Sigma}_k^{(s)}\big)\right\}$$
$$> \sum_{n=0}^{N-1} \ln\left\{\sum_{k=0}^{K-1} g_k^{(b)}\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k^{(b)},\,\boldsymbol{\Sigma}_k^{(b)}\big)\right\} + \ln\,p(H_1) - \ln\,p(H_0).$$

Results of a Speaker Verification – Part 1

Boundary conditions:

- ❑ The ***results*** are taken from the ***dissertation of G. Kolano*** (work done at the Daimler Research Center in Ulm, see literature section for details).
- ❑ A ***data base with 106 speakers*** (only male speakers) has been used. The data based consists of English double-digits (i.e., the vocabulary is limited).
- ❑ All data has been transmitted over ***telephone channels***. Thus, the bandwidth of the data is approximately 3.8 kHz (8 kHz sample rate). Especially for speaker recognition, these are rather bad boundary conditions.
- ❑ Out of the 106 speakers, ***33 have been used for training the background models***, the remaining ***73 have been used for the evaluation of the speaker identification***.
- ❑ ***MFCCs*** have been used as features. They were only computed if the current signal frame has been classified as voiced speech.

Results of a Speaker Verification – Part 2

Comparison between codebooks and GMMs:

- ❑ The background model has the same size as the speaker model for the cases.
- ❑ Results in terms of error rates:

Model order (Number of codebook entries or number Gaussian distributions)	Codebuch approach	Gaussian mixture model
4	11.5 %	4.2 %
8	9.6 %	3.0 %
16	8.2 %	2.3 %
32	6.8 %	2.0 %

Conclusion:

GMMs are – at least in this test – clearly superior to codebook approaches, but …

Results of a Speaker Verification – Part 3

Comparison between codebooks and GMMs:

- The ***covariance matrices*** of the GMM approach were ***fully populated***. Thus, clearly a larger amount of model parameters have been used in this approach and the computational complexity is clearly higher.
- Number of model parameters:

Model order (Number of codebook entries or number Gaussian distributions)	Codebook approach	Gaussian mixture model
4	68	683
8	136	1367
16	272	2735
32	544	5471

Conclusion:

… GMMs require clearly more memory and computational power, compared to codebook approaches.

Results of a Speaker Verification – Part 4

Comparison between global and individual thresholds:

- ❑ So far, ***individual thresholds*** and a priory-probabilities have been trained for each speaker.
- ❑ ***Comparison*** between global and individual thresholds:

Model order (Number of codebook entries or number Gaussian distributions)	Codebook approach	Gaussian mixture model
4	12.9 % / 11.5 %	5.3 % / 4.2 %
8	11.1 % / 9.6 %	4.1 % / 3.0 %
16	9.6 % / 8.2 %	3.4 % / 2.3 %
32	8.2 % / 6.8 %	3.0 % / 2.0 %

Global threshold

Individual Threshold

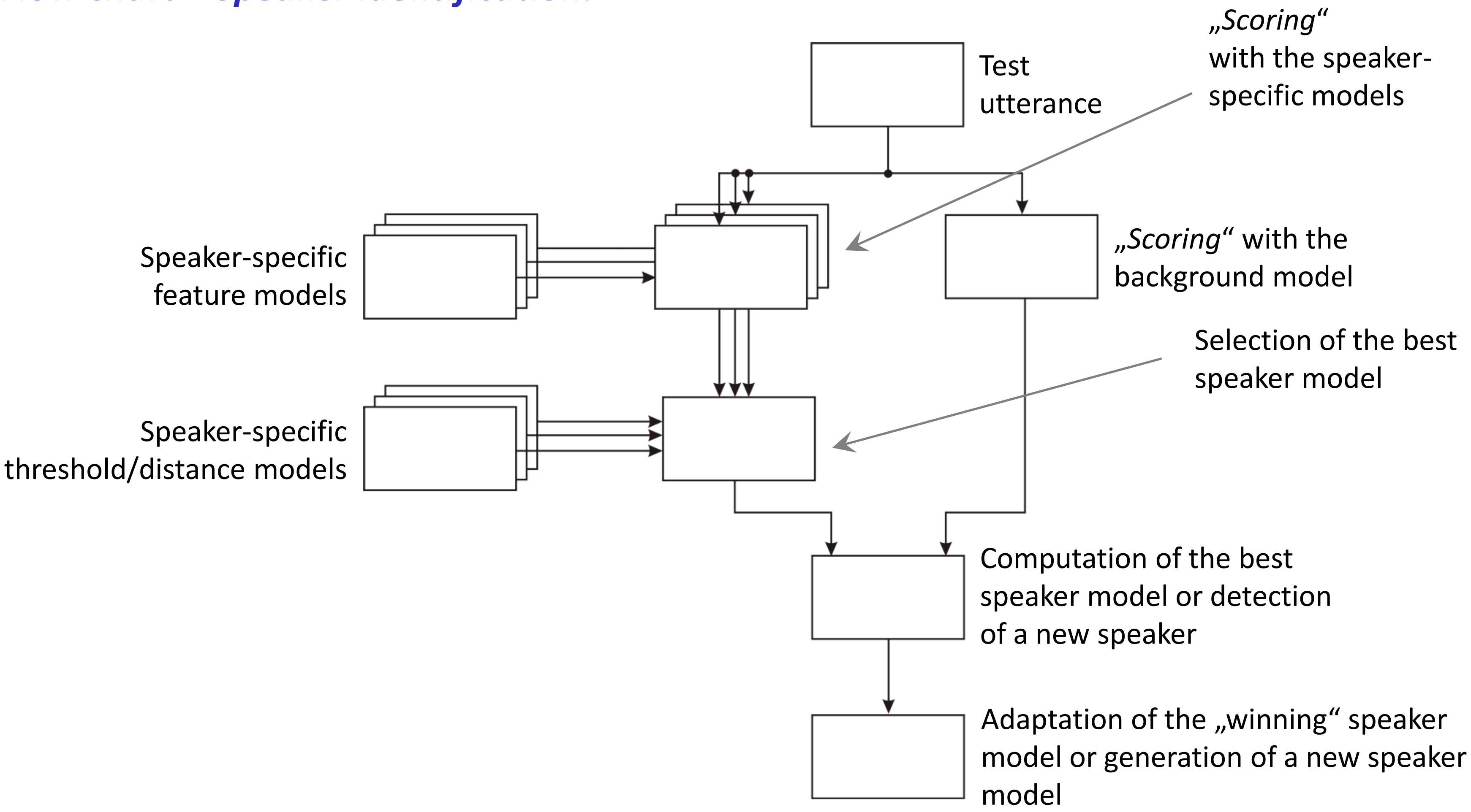
Conclusion:

By training the thresholds, the recognition rate can be improved or the number of parameters can be decreased.

Gaussian Mixture Models (GMMs)

From Speaker Verification to Speaker Identification

Flow chart – Speaker identification:



Test utterance
„*Scoring*“ with the speaker-specific models
Speaker-specific feature models
„*Scoring*“ with the background model
Selection of the best speaker model
Speaker-specific threshold/distance models
Computation of the best speaker model or detection of a new speaker
Adaptation of the „winning“ speaker model or generation of a new speaker model

Results of a Speaker Identification – Part 1

Boundary conditions:

- ❑ The results are taken from a publication of ***D. Reynolds*** (work done at the MIT, see literature section for details).
- ❑ A data base with ***51 speakers*** (only male speakers) has been used. The data base consists of ***English conversations*** (approximately 10 utterances with a duration of 45 seconds each).
- ❑ All data has been transmitted over ***telephone channels***. Thus, the bandwidth of the data is approximately 3.8 kHz (8 kHz sample rate).
- ❑ ***MFCCs*** have been used as features. Modeling has been done with GMMs, where ***only diagonal covariance matrices*** have been used.

Results of a Speaker Identification – Part 2

Results:

- Length of test and training data vs. recognition rate:

Length of training data	Model order (number of Gaussian distributions)	Length of test data		
		1 sec	5 sec	10 sec
30 sec	8 16 32	54.6 % 63.7 % 64.6 %	79.8 % 87.3 % 85.3 %	86.6 % 90.5 % 88.4 %
60 sec	8 16 32	66.1 % 74.9 % 78.6 %	91.5 % 95.7 % 95.6 %	97.3 % 98.8 % 98.3 %
90 sec	8 16 32	71.5 % 79.0 % 84.7 %	95.5 % 98.0 % 98.8 %	98.8 % 99.7 % 99.6 %

Adaption of the Models During Run-Time – Part 1

General:

- After a ***speaker recognition*** has been ***successful*** (this should be validated e.g. by using a dialog system), the ***speaker model of the active speaker*** can be ***adapted***.
- Generally, all model parameters can be adapted. However, ***updating only the mean values of GMMs*** proved to provide a ***good cost-value ratio***. For codebooks, the mean values can be seen as the individual codebook entries, i.e., all parameters are adapted.
- Both, the amount of ***training data*** and the ***number of new feature vectors*** should be considered. The codebook adaption can be done according to

$$\boldsymbol{c}_i^{(\mathrm{new})} = \frac{N_i}{N_i + N_i^{(x)}}\,\boldsymbol{c}_i^{(\mathrm{old})} + \frac{1}{N_i + N_i^{(x)}} \sum_{n=0}^{N_i^{(x)}-1} \boldsymbol{x}_i(n)$$

 where $\boldsymbol{c}_i^{(\mathrm{new})}$ denotes the new codebook entry and $\boldsymbol{c}_i^{(\mathrm{old})}$ the old one. N_i is the number of vectors that have been used to form the entry during training and $N_i^{(x)}$s the number of those feature vectors which have been assigned to the corresponding codebook vector.

Adaption of the Models During Run-Time – Part 2

General:

- The ***mean values of GMMs*** can be updated similar to the codebooks by a modified iteration step of the EM algorithm (see last lecture). First, a ***„soft“ assignment*** to the individual classes is done (E-step):

$$\gamma\big(z_k(n)\big) = \frac{g_k\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_k,\,\boldsymbol{\Sigma}_k\big)}{\sum\limits_{j=0}^{K-1} g_j\,\mathcal{N}\big(\boldsymbol{x}(n)|\boldsymbol{\mu}_j,\,\boldsymbol{\Sigma}_j\big)}.$$

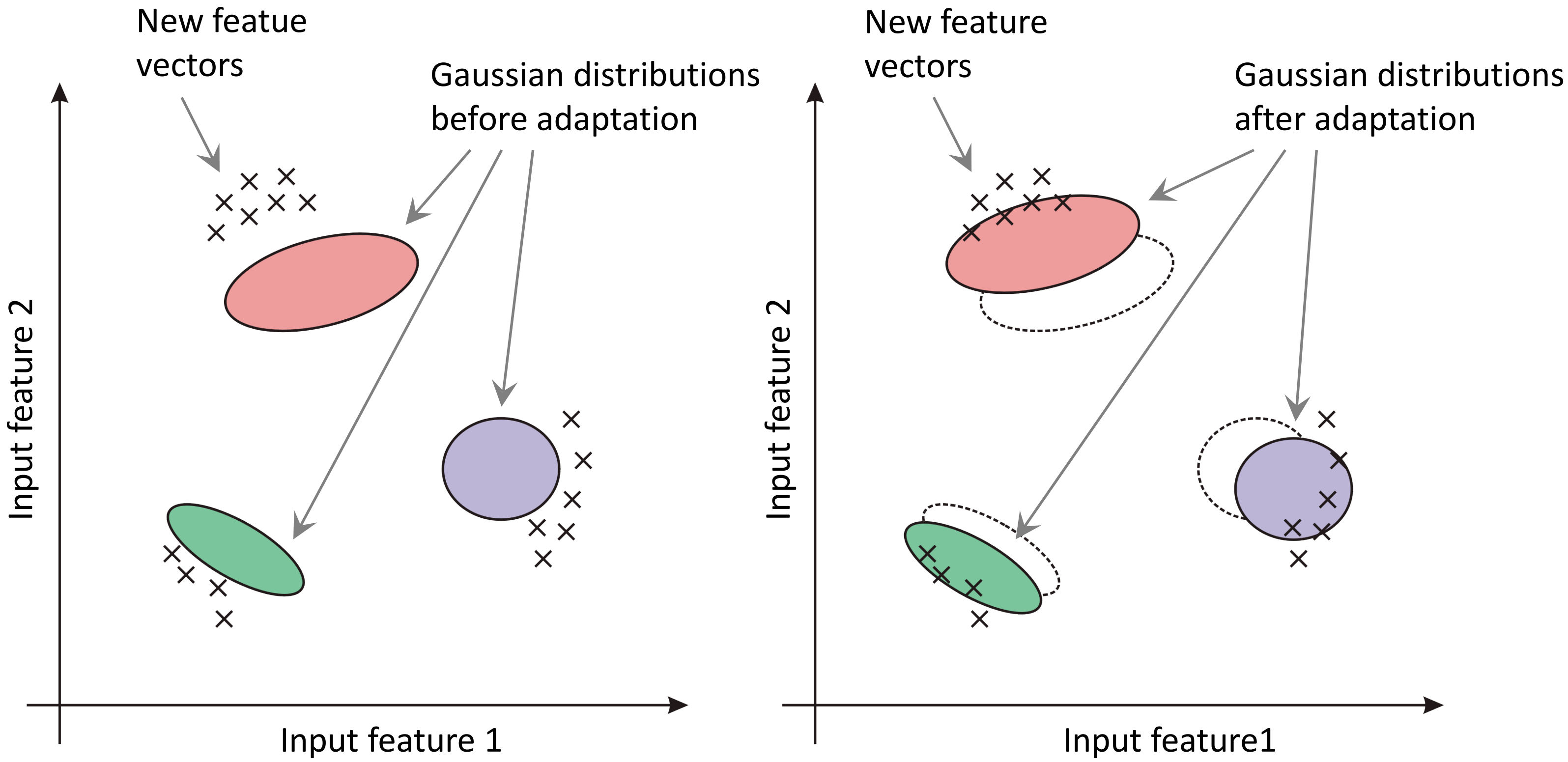
Next, the mean values are corrected (M-step) by

$$\boldsymbol{\mu}_k^{(\text{new})} = \frac{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big)\,\boldsymbol{x}(n) + N_k\,\boldsymbol{\mu}_k^{(\text{old})}}{\sum\limits_{n=0}^{N-1} \gamma\big(z_k(n)\big) + N_k}.$$

The variable N_k denotes the sum of the „soft“ assignments of the *k*th class in the last iteration during the training.

Adaption of the Models During Run-Time – Part 3

Example:



Gaussian Mixture Models (GMMs)

Christian-Albrechts-Universität zu Kiel

Summary and Outlook

Summary:

- ❑ Motivation
- ❑ Uncertainties in Machine Learning
- ❑ Basics
  - ❑ Training of GMMs
  - ❑ Initialization of GMMs
- ❑ Applications examples taken from speech and audio processing
  - ❑ Signal separation
  - ❑ Speaker recognition

Next part:

- ❑ Neural networks