

# PSR 07/08 - Summary

Max Jakob <max.jakob@web.de>

Linlin Li <ridingwind@gmail.com>

Torsten Marek <torsten@diotavelli.net>

Raveesh Meena <raveesh.mena@gmail.com>

Will Roberts <wroberts@coli.uni-sb.de>

Benjamin Roth <beroth@coli.uni-sb.de>

March 13, 2008



# Contents

<b>1</b>	<b>Feature Extraction from Sound</b>	<b>5</b>
1.1	Digital Sound . . . . .	5
1.1.1	Sampling . . . . .	5
1.1.2	Quantization . . . . .	5
1.2	MFCC Feature Vectors . . . . .	5
1.2.1	Preemphasis . . . . .	6
1.2.2	Mel-scaled Filterbank . . . . .	7
1.2.3	Impulse Response Filtering . . . . .	7
1.2.4	Discrete Cosine Transformation . . . . .	8
1.2.5	Dynamic Features and LDA . . . . .	8
1.3	Other Features . . . . .	9
1.3.1	Short Time Energy . . . . .	9
<b>2</b>	<b>Bayesian Decision Theory</b>	<b>11</b>
2.1	The Bayes Classifier . . . . .	11
2.1.1	Without Knowledge . . . . .	11
2.1.2	With Knowledge . . . . .	11
2.2	Loss Functions . . . . .	12
2.2.1	Two-class Problem . . . . .	12
<b>3</b>	<b>Maximum Likelihood Estimation</b>	<b>15</b>
3.1	Estimation with the Bernoulli Experiment . . . . .	15
3.1.1	Problem Formulation . . . . .	15
3.1.2	Maximizing the Probability . . . . .	15
3.2	Example MLEs . . . . .	16
3.2.1	Unknown $\sigma$ in a Univariate ND . . . . .	16
3.2.2	Unknown $\mu$ in a Univariate ND . . . . .	17
3.2.3	Exponential Distributions . . . . .	17
<b>4</b>	<b>Nonparametric Techniques</b>	<b>19</b>
4.1	KNN Density Estimation . . . . .	19
4.2	Parzen Windows . . . . .	19
4.2.1	Unit Cube Window Function . . . . .	19
4.2.2	Other Window Functions . . . . .	20
4.2.3	Classification using Parzen Windows . . . . .	20
4.3	$K_n$ Nearest Neighbor Estimation . . . . .	21
4.3.1	Distance Functions . . . . .	21
<b>5</b>	<b>Gaussian Mixture Models</b>	<b>23</b>
5.1	Overview . . . . .	23
5.2	Training . . . . .	23

<b>6</b>	<b>Decision Trees</b>	<b>27</b>
6.1	Structure . . . . .	27
6.2	Types of questions . . . . .	27
6.2.1	Nominal data . . . . .	27
6.2.2	Continuous data . . . . .	27
6.3	Splitting . . . . .	28
6.3.1	Impurity of a node . . . . .	28
6.3.2	Training algorithms (Greedy) . . . . .	29
6.4	Pruning . . . . .	30

# 1 Feature Extraction from Sound

## 1.1 Digital Sound

### 1.1.1 Sampling

An incoming analog speech signal is measured  $f_s$  times per second (5000 Hz is sufficient to understand speech, CD quality is 44,100 Hz). The highest tone that can be represented is  $\frac{f_s}{2}$ . All higher frequencies have to be removed prior to sampling to avoid spectral aliasing.

### 1.1.2 Quantization

Each input sample needs to be quantized in order to be stored as a digital value.

#### Uniform Quantization

$$x_q(n) = \frac{x(n) - \min x(n)}{\Delta r} \quad (1.1)$$

with the quantization step  $\Delta r$

$$\Delta r = \frac{\max x(n) - \min x(n)}{2^s - 1} \quad (1.2)$$

#### Symbols

$s$  the sampling width in bits

#### $\mu$ -Law

The  $\mu$ -law algorithm can be used to reduce the quantization error.

$$f_n^{(\mu)} = f_{max} \operatorname{sign}(f_n) \frac{\log(1 + \mu \frac{|f_n|}{f_{max}})}{\log(1 + \mu)}, \mu \approx 200 \quad (1.3)$$

## 1.2 MFCC Feature Vectors

The annotated pipeline for computing MFCC (Mel-Frequency Cepstral Coefficients) feature vectors.

## 1 Feature Extraction from Sound

### 1.2.1 Preemphasis

$$f'_n = f_n - \alpha f_{n-1} \quad (1.4)$$

$\alpha = 0.95$ , from experiments.

Preemphasis boosts higher frequencies and filters out distortions introduced by the lips.

#### Output

A sequence  $f'_n$  of numerical values.

### Discrete Fourier Transformation

The discrete Fourier transformation is calculated using the Fast Fourier Transform algorithm (FFT). The formula for the DFT is given by

$$F_\nu^{(m)} = \sum_{n=0}^{N-1} f_{m-n} w_n e^{-i2\pi\nu \frac{n}{N}} \quad (1.5)$$

#### Symbols

$\nu$  frequency in Hz

$m$  frame index

$N$  the frame width (needs to be a power of 2 for the FFT)

$w$  windowing function

### Windowing

For the Fourier Transformation, the input signal is split into frames (usually 25ms long, speech can be assumed to be “static” for this interval), with a new frame every 10ms (i.e. there is an overlap between consecutive frames).

### Transformation

For each frame, the Fourier transformation is computed for all frequencies that can be detected in this time interval.<sup>1</sup>

### Imaginary Parts

For real inputs (as opposed to imaginary inputs), the upper half of the Fourier transform is redundant and just a mirror image of the lower half.  $N/2$  (for even  $N$ ) is the Nyquist element (or Nyquist frequency).

---

<sup>1</sup>The frame needs to contain at least one complete waveform.

## Output

A sequence of feature vectors with  $N$  dimensions in the frequency domain (where  $N$  is a power of 2, usually 256), the “spectrum”. The result of the Fourier transform consists of complex numbers, so they are converted to absolute values first (i.e. “vector length”, if an imaginary number is understood as a 2-dimensional vector).

### 1.2.2 Mel-scaled Filterbank

The Mel-scaled filterbank is used to smooth over neighboring frequencies (similar to the ear), which leads to smaller feature vectors.

The center frequencies (i.e. peak position of one filter) is given by the Mel scale:

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \quad (1.6)$$

### Speaker Adaptation

The average position of formants depends on the length of the vocal tract, thus the center frequencies of the Mel scale should vary with regard to the speaker.

## Output

A sequence of feature vectors with usually 24 dimensions in the frequency domain.

### 1.2.3 Impulse Response Filtering

A measured speech signal  $f$  consists of the original signal  $e$  and the impulse response  $h$ , which contains, among others, the reverb added by the speaker’s environment.

$$f_m = \sum_{n=-\infty}^{\infty} h_{m-n} e_n \quad (1.7)$$

which is the convolution

$$f(t) = h(t) \star e(t) \quad (1.8)$$

In the frequency domain, a convolution is simply a point-wise multiplication. With the Fourier coefficient vectors ( $m$  is the frame index), this becomes

$$F^{(m)}(\omega) = H^{(m)}(\omega)E^{(m)}(\omega) \quad (1.9)$$

$$\log F^{(m)}(\omega) = \log H^{(m)}(\omega) + \log E^{(m)}(\omega) \quad (1.10)$$

Since  $H$  is stationary and  $E^{(m)}$  dynamic, an average over  $F^{(m)}$  will be dominated by  $H$ :

$$H \approx \frac{1}{N} \sum_{i=0}^N \log F^{(m)}(\omega) \quad (1.11)$$

## 1 Feature Extraction from Sound

After several seconds of speech,  $H$  can be reliably computed and removed from the input signal.

### Output

The input vectors, possibly with the impulse response removed.

### 1.2.4 Discrete Cosine Transformation

The feature vectors (the spectrum) is analyzed for spectral components using the inverse DFT, which converts the vectors from the frequency domain into a time-like domain (called *quefreny*). Now it is possible to separate the high harmonic frequencies from the formant frequencies, which reside in the lower coefficients.

In practice, the DCT is used instead of the IDFT because the input only contains real data and because it can be computed faster.

$$c_n^{(m)} = \sqrt{\frac{2}{N}} \sum_{l=1}^N \log(F_l^{(m)}) \cos\left(\frac{\pi n(l + \frac{1}{2})}{N}\right), \quad n = 1, 2, \dots \quad (1.12)$$

Please keep in mind that the log does not belong to the DCT, but is special to the MFCC features, and due to the impulse response from the previous step. It is also motivated by the fact that loudness is perceived on a logarithmic scale by the human ear.

### Output

A feature vector in the “quefreny” domain, the “cepstrum”. Only the 20 lowest cepstral coefficients are kept (assuming the input had 24 dimensions).

### 1.2.5 Dynamic Features and LDA

The feature vectors are enriched with their 1st and 2nd derivatives, tripling its size. After that, Linear Discriminant Analysis is used to reduce the size of the feature vector again.

### Output

MFCC feature vectors.

## 1.3 Other Features

### 1.3.1 Short Time Energy

Short time energy is useful for distinguishing between voiced and voiceless sounds. It also shows the positions of actual speech or silence and allows for a coarse detection of syllable borders and cores. The short time energy is calculated on windows of 25ms, every 10ms.

It is defined as

$$E^{(m)} = \sum_{n=0}^{N-1} f_{n+m}^2 \quad (1.13)$$

#### Symbols

$N$  window width

$m$  frame offset

## *1 Feature Extraction from Sound*

## 2 Bayesian Decision Theory

### 2.1 The Bayes Classifier

#### 2.1.1 Without Knowledge

To minimize the number of wrong classifications, the Bayes Classifier always picks the class with the highest probability.

$$\bar{\omega}_i = \arg \max_{\omega_k} P(\omega_k) \quad (2.1)$$

#### 2.1.2 With Knowledge

Given a measurement  $x$ , the probability of  $P(\omega_k|x)$  depends on  $x$ , thus

$$\bar{\omega}_i = \arg \max_{\omega_k} P(\omega_k|x) \quad (2.2)$$

As a reminder, the definition of a conditional probability

$$P(a|b) = \frac{P(a,b)}{P(b)} \quad (2.3)$$

Expanding  $P(\omega_k|x)$  in 2.2 yields

$$\bar{\omega}_i = \arg \max_{\omega_k} \frac{P(\omega_k, x)}{P(x)} \quad (2.4)$$

Now we expand the joint probability  $P(\omega_k, x) = P(x, \omega_k)$

$$\bar{\omega}_i = \arg \max_{\omega_k} \frac{P(x|\omega_k)P(\omega_k)}{P(x)} \quad (2.5)$$

$P(x)$  is just a scaling factor and can be dropped WLOG:

$$\bar{\omega}_i = \arg \max_{\omega_k} P(x|\omega_k)P(\omega_k) \quad (2.6)$$

yielding the Bayes Decision Rule with prior  $P(\omega_k)$  and posterior  $P(x|\omega_k)$ .

## 2.2 Loss Functions

Let  $\omega_1 \dots \omega_c$  be the true classes,  $\alpha_1 \dots \alpha_c$  the actions taken on classification, then  $\lambda(\alpha_i|\omega_j)$  is the loss incurred by choosing action  $\alpha_i$  for an element with the true class  $\omega_j$ . The conditional risk  $R$  for an element  $x$  and an action  $\alpha_i$  is defined as

$$R(\alpha_i|x) = \sum_{j=1}^c \lambda(\alpha_i|\omega_j)P(\omega_j|x) \quad (2.7)$$

To minimize the overall risk, we need a decision rule  $f := x \mapsto \alpha$  that minimizes the overall risk for a specific  $x$ .

$$R = \int R(f(x)|x)p(x)dx \quad (2.8)$$

### 2.2.1 Two-class Problem

In a two-class problem with classes  $\omega_1, \omega_2$  and corresponding actions  $\alpha_1, \alpha_2$ , we have

$$R(\alpha_1|x) = \lambda_{11}P(\omega_1|x) + \lambda_{12}P(\omega_2|x) \quad (2.9)$$

$$R(\alpha_2|x) = \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x) \quad (2.10)$$

where  $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$ .

$\alpha_1$  classifies  $x$  as  $\omega_1$ , thus we decide for  $\omega_1$  if

$$R(\alpha_1|x) < R(\alpha_2|x) \quad (2.11)$$

$$\Leftrightarrow \lambda_{11}P(\omega_1|x) + \lambda_{12}P(\omega_2|x) < \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x) \quad (2.12)$$

$$\Leftrightarrow (\lambda_{12} - \lambda_{22})P(\omega_2|x) < (\lambda_{21} - \lambda_{11})P(\omega_1|x) \quad (2.13)$$

which replaces the Bayes Decision Rule.

Using eq. 2.6, we can rephrase this as

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} > \frac{(\lambda_{12} - \lambda_{22})P(\omega_2)}{(\lambda_{21} - \lambda_{11})P(\omega_1)} \quad (2.14)$$

or

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} > \theta \quad (2.15)$$

where  $\theta$  can be tuned to minimize the overall risk, which is difficult solve in the general case. In the special case of minimizing the number of classification errors, we take a “Zero-One-Loss” function

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases} \quad i, j = 1, \dots, c \quad (2.16)$$

Putting eq. 2.16 into eq. 2.7, we get

$$\begin{aligned} R(\alpha_i|x) &= \sum_{j=1, i \neq j}^c P(\omega_j|x) \\ &= 1 - P(\omega_i|x) \end{aligned} \tag{2.17}$$

To minimize  $R(\alpha_i|x)$ , we have to pick an  $i$  such that  $P(\omega_i|x)$  is maximal, in which case we have reconstructed the Bayes decision rule (eq. 2.6).



# 3 Maximum Likelihood Estimation

A popular statistical method used to estimate parameters of the underlying probability distribution of given samples. It finds the most likely value of the parameter (the likelihood of the sample is the highest when the estimate parameter is used).

## 3.1 Estimation with the Bernoulli Experiment

### 3.1.1 Problem Formulation

The result  $x$  of drawing  $N$  balls is known. What is the probability  $p$  of observing such a sequence?



Figure 3.1: The Bernoulli Experiment

### 3.1.2 Maximizing the Probability

Find a  $p$  that maximizes the observation sequence

$$\hat{p} = \arg \max_p P_{Sequence} = \arg \max_p \binom{N}{x} (1-p)^{N-x} p^x \quad (3.1)$$

We get the likelihood  $L$

$$L(p) = \binom{N}{x} (1-p)^{N-x} p^x \quad (3.2)$$

take the derivative

### 3 Maximum Likelihood Estimation

$$\frac{dL(p)}{dp} = \binom{N}{x} \frac{x}{p} - \binom{N}{x} \frac{N-x}{1-p} \quad (3.3)$$

and find the  $p$  that maximizes  $L$  by finding the roots of  $\frac{dL(p)}{dp}$

$$\hat{p} = \frac{x}{N} \quad (3.4)$$

## 3.2 Example MLEs

In all MLEs, we assume that all  $x_i$  are independent and identically-distributed.

### 3.2.1 Unknown $\sigma$ in a Univariate ND

$$P(x|\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (3.5)$$

The probability for a given sequence of samples  $x_1 \dots x_N$  is

$$P(\sigma) = \prod_{i=1}^N P(x_i|\sigma) \quad (3.6)$$

We take the log-likelihood  $L(\sigma)$

$$\begin{aligned} L(\sigma) &= \log \left[ \prod_{i=1}^N P(x_i|\sigma) \right] \\ &= \sum_{i=1}^N \log(P(x_i|\sigma)) \\ &= \sum_{i=1}^N -\frac{1}{2} \left( \frac{x_i - \mu}{\sigma} \right)^2 - \frac{1}{2} \log 2\pi - \log \sigma \end{aligned} \quad (3.7)$$

We derive  $L(\sigma)$

$$\begin{aligned} \frac{dL(\sigma)}{d\sigma} &= \sum_{i=1}^N \left[ \frac{(x_i - \mu)^2}{\sigma^3} - \frac{1}{\sigma} \right] \\ &= \frac{1}{\sigma^3} \sum_{i=1}^N \left[ (x_i - \mu)^2 \right] - \frac{N}{\sigma} \end{aligned} \quad (3.8)$$

Now we find the roots of the function and get

$$0 = \frac{1}{\sigma^3} \sum_{i=1}^N \left[ (x_i - \mu)^2 \right] - \frac{N}{\sigma} \quad (3.9)$$

$$\Leftrightarrow \sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2 \quad (3.10)$$

### 3.2.2 Unknown $\mu$ in a Univariate ND

The conditional probability  $P(x|\mu)$  is given by

$$P(x|\mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (3.11)$$

Similar to eqs. 3.6 and 3.7, we calculate the log-likelihood  $L(\mu)$  and derive with regard to  $\mu$

$$\begin{aligned} \frac{dL(\mu)}{d\mu} &= \sum_{i=1}^N -\frac{(x_i - \mu)}{\sigma^2} \\ &= -\frac{1}{\sigma^2} \sum_{i=1}^N (x_i - \mu) \end{aligned} \quad (3.12)$$

To find the roots, we set the derivative to 0.

$$0 = -\frac{1}{\sigma^2} \sum_{i=1}^N (x_i - \mu) \quad (3.13)$$

$-\frac{1}{\sigma^2}$  can be dropped from the equation, because it will never be 0, leaving

$$0 = \sum_{i=1}^N (x_i - \mu) \quad (3.14)$$

$$\Leftrightarrow \mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (3.15)$$

### 3.2.3 Exponential Distributions

$$P(x|\lambda) = \lambda e^{-\lambda x} \quad (3.16)$$

Under the independence assumption, this can be simplified to

$$P(\lambda) = \prod_{i=1}^N P(x_i|\lambda) \quad (3.17)$$

The log-likelihood  $L(\lambda)$  is

$$\begin{aligned} L(\lambda) &= \sum_{i=1}^N \log(\lambda e^{-\lambda x_i}) \\ &= \sum_{i=1}^N \log \lambda - \lambda x_i \\ &= N \log \lambda - \lambda \sum_{i=1}^N x_i \end{aligned} \quad (3.18)$$

### 3 Maximum Likelihood Estimation

To find the maxima of  $L(\lambda)$ , we take the first derivative

$$\frac{dL(\lambda)}{d\lambda} = \frac{N}{\lambda} - \sum_{i=1}^N x_i \quad (3.19)$$

and find the roots

$$0 = \frac{N}{\lambda} - \sum_{i=1}^N x_i \quad (3.20)$$

$$\begin{aligned} \Leftrightarrow \lambda &= \frac{1}{\frac{1}{N} \sum_{i=1}^N x_i} \\ &= \frac{1}{\bar{x}} \end{aligned} \quad (3.21)$$

## 4 Nonparametric Techniques

### 4.1 KNN Density Estimation

Suppose there are  $k$  training vectors inside a region  $R$  from a total of  $n$  training vectors. The probability  $P$  of being in  $R$  can be estimated as:

$$P \approx \frac{k}{n} \quad (4.1)$$

With the number of training vectors approaching infinity, we can make the Volume of the region smaller and smaller, to ultimately arrive at the density of a point.

For  $n \rightarrow \infty$ ,  $V_n$  the volume of  $R$  and  $k_n$  the number of points in  $R$ , we define a Sequence of probabilities:

$$P_n(x) = \frac{k_n}{nV_n} \quad (4.2)$$

In order to converge to the true probability  $P(x)$  it is important

1. that  $P(x)$  has the property:  $V_n \rightarrow 0$  for  $n \rightarrow \infty$  ( $P(x)$  is a local property)
2. that  $k_n \rightarrow \infty$  for  $n \rightarrow \infty$  (estimate is reliable)
3. that  $\frac{k_n}{n} \rightarrow 0$  for  $n \rightarrow \infty$  (otherwise volume cannot shrink to zero)

A possible choice for  $k_n$  is

$$k_n = \sqrt{n} \quad (4.3)$$

and pick  $V_n$  such that it contains  $k_n$  samples.

### 4.2 Parzen Windows

#### 4.2.1 Unit Cube Window Function

The idea of Parzen windows is that every data point represents a piece of probability. This probability is expressed by a *window function*. A possible choice for such a window function is the *unit cube*:

$$\phi(x) = \begin{cases} 1 & \text{if } |x^j| \leq \frac{1}{2} \text{ for all } j = 1 \dots d \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

With  $x^j$  the  $j$ 'th component of  $\vec{x}$ .

## 4 Nonparametric Techniques

The window function can be shifted and squeezed. A window function centered at a data point  $\vec{x}_i$  with length of edge  $h_n$  is

$$\phi\left(\frac{\vec{x} - \vec{x}_i}{h_n}\right) \quad (4.5)$$

Instead of counting how many data points lie within a volume  $V_n$  (as with the *KNN*-estimation), we can count how many window functions fire at a certain point  $\vec{x}$ :

$$k_n(\vec{x}) = \sum_{i=1}^n \phi\left(\frac{\vec{x} - \vec{x}_i}{h_n}\right) \quad (4.6)$$

It is then:

$$P_n(\vec{x}) = \frac{k_n(\vec{x})}{nV_n} \quad (4.7)$$

the volume is  $(h_n)^d$ , with  $d$  the number of dimensions of  $\vec{x}$ . A possible value for  $h_n$  is  $h_1/\sqrt{n}$ . This *non-parametric* technique has one parameter to set, namely  $h_1$ .

### 4.2.2 Other Window Functions

Other window functions are

- Gaussian distribution with unit covariance matrix and mean 0:

$$\phi(\vec{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\vec{x}^t \cdot \vec{x}} \quad (4.8)$$

- Sphere

$$\phi(\vec{x}) = \begin{cases} 1 & \text{if } |\vec{x}| \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

### 4.2.3 Classification using Parzen Windows

Classification is done by estimating the densities for different classes and using Bayes Decision Rule.

### 4.3 $K_n$ Nearest Neighbor Estimation

Above it was stated that the density can be estimated by looking at the neighbors in a certain window. How can we make use of this for classification, i.e. estimate  $P(\omega_i|x)$  for a given class  $\omega_i$ ?

let

- $V$  be the volume (picked such that  $k$  samples lie in it)
- $k_i$  be the number of samples belonging to class  $\omega_j$  in  $V$
- $k$  be the total number of samples in  $V$  ( $k = \sum_j k_j$ )

then

$$P_n(x, \omega_i) = \frac{k_i}{nV} \quad (4.10)$$

$$P_n(x) = \sum_j P_n(x, \omega_j) = \frac{k}{nV} \quad (4.11)$$

$$P_n(\omega_i|x) = \frac{P_n(x, \omega_i)}{P_n(x)} = \frac{k_i}{k} \quad (4.12)$$

A  $1nn$ -classifier is in the worst case only twice as bad as Bayes (having perfect information).

#### 4.3.1 Distance Functions

In order to determine the nearest neighbor, different distance functions can be used. Usually they are instantiations of the  $L_\alpha$ -Norm, which expresses the distance between two points as:

$$D(\vec{x}, \vec{y})^\alpha = \sum_{i=1}^d |x_i - y_i|^\alpha \quad (4.13)$$

Important  $L_\alpha$ -Norms are:

- $\alpha = 1$  : Manhattan distance
- $\alpha = 2$ : Euclidean distance
- $\alpha \rightarrow \infty$ : Maximum norm, length of biggest component

## 4 *Nonparametric Techniques*

# 5 Gaussian Mixture Models

## 5.1 Overview

**Idea:** composite distribution made up of many weighted Gaussians (actually a weighted average). Each Gaussian has its own mean and covariance matrix which have to be estimated separately.

**Applications:**

- speaker identification
- image segmentation

Gaussian distribution (multivariate):

$$P(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^t \Sigma^{-1} (x-\mu)} \quad (5.1)$$

Mixture model using mixture weights  $p_k$  and Gaussians  $p(x|\theta_k) = p(x|\mu_k, \Sigma_k)$ :

$$P(x|\theta_1 \dots \theta_K, p_1 \dots p_K) = \sum_{k=1}^K p_k p(x|\theta_k) \quad (5.2)$$

## 5.2 Training

Training is done using the EM-Algorithm, an iterative optimization of the likelihood.

Define the auxiliary objective function:

$$Q(\theta^j, \theta^{j+1}) = \sum_{i=1}^N \sum_y P(y|x_i, \theta^j) \log P(y, x_i|\theta^{j+1}) \quad (5.3)$$

where

$y$  is the assignment of a given  $x_i$  to a given Gaussian  $k$  (hidden property), and  $x_i$  are the training samples.

Then

1. Initialize  $\theta^0, j = 0$
2. Calculate  $Q(\theta^j, \theta^{j+1})$
3. Update parameters:  $\hat{\theta}^{j+1} = \arg \max_{\theta^{j+1}} Q(\theta^j, \theta^{j+1})$
4. Repeat (2) and (3) until convergence.

## 5 Gaussian Mixture Models

Notes:

- EM algorithm derived from maximum likelihood principle
- formula for  $Q$  depends on specific problem to be solved
- $Q$  function must satisfy some properties (*e.g.*, being bounded)

How to use  $Q$ :

$$P(y|x_i, \theta^j) = \frac{P(y, x_i|\theta^j)}{\sum_y P(y, x_i|\theta^j)} = \frac{P(y|\theta^j)P(x_i|y, \theta^j)}{\sum_y P(y|\theta^j)P(x_i|y, \theta^j)}$$

Rewriting using  $y = k$ :

$$P(y|\theta) = p_k \tag{5.4}$$

$$P(x_i|y, \theta) = P(x_i|k, \theta) = P(x_i|\theta_k) \tag{5.5}$$

$$P(y|x_i, \theta) = \frac{p_k P(x_i|\theta_k)}{\sum_k p_k P(x_i|\theta_k)} \tag{5.6}$$

Second part of  $Q$ :

$$\begin{aligned} P(y, x_i|\theta^{j+1}) &= P(y|\theta^{j+1})P(x_i|y, \theta^{j+1}) \\ &= p_k^{j+1} P(x_i|\theta_k^{j+1}) \end{aligned} \tag{5.7}$$

$Q$  for mixture model:

$$Q(\theta^j, \theta^{j+1}) = \sum_{i=1}^N \sum_k \frac{p_k^j P(x_i|\theta_k^j)}{\sum_{k'} p_{k'}^j P(x_i|\theta_{k'}^j)} \log(p_k^{j+1} P(x_i|\theta_k^{j+1})) \tag{5.8}$$

Weights must be normalized:

$$\sum_k p_k^{j+1} = 1$$

Use a Lagrange multiplier:  $+ \mu(1 - \sum_k p_k^{j+1})$ .

$$\frac{\partial \hat{Q}(\theta^j, \theta^{j+1})}{\partial p_k^{j+1}} = \sum_{i=1}^N \frac{p_k^j P(x_i|\theta_k^j)}{\sum_{k'} p_{k'}^j P(x_i|\theta_{k'}^j)} \frac{1}{p_k^{j+1}} - \mu = 0$$

Solution:

$$p_k^{j+1} = \frac{1}{\mu} \sum_{i=1}^N \frac{p_k^j P(x_i|\theta_k^j)}{\sum_{k'} p_{k'}^j P(x_i|\theta_{k'}^j)}$$

Lagrange multiplier:

$$\begin{aligned} 1 &= \frac{1}{\mu} \sum_{i=1}^N \frac{\sum_k p_k^j P(x_i|\theta_k^j)}{\sum_{k'} p_{k'}^j P(x_i|\theta_{k'}^j)} \\ &= \frac{1}{\mu} \sum_{i=1}^N 1 \\ &= \frac{N}{\mu} \\ &\Leftrightarrow \mu = N \end{aligned}$$

Before updating, auxiliary values  $\gamma_{i,k}$  are computed for all Gaussians  $k$  and data points  $i$

$$\gamma_{i,k}^j = \frac{p_k^j P(x_i | \theta_k^j)}{\sum_{k'} p_{k'}^j P(x_i | \theta_{k'}^j)} \quad (5.9)$$

With the auxiliary values, weights, mean and covariances can be updated. The update equation for the weights  $p_k$ :

$$p_k^{j+1} = \frac{1}{N} \sum_{i=1}^N \gamma_{i,k}^j \quad (5.10)$$

the means  $\mu_k$ :

$$\mu_k^{j+1} = \frac{1}{\sum_{i=1}^N \gamma_{i,k}^j} \sum_{i=1}^N \gamma_{i,k}^j x_i \quad (5.11)$$

and the covariances  $\Sigma_k$ :

$$\Sigma_k^{j+1} = \frac{1}{\sum_{i=1}^N \gamma_{i,k}^j} \sum_{i=1}^N \gamma_{i,k}^j (x_i - \mu_k)(x_i - \mu_k)^t \quad (5.12)$$

Or, in the univariate case, the standard deviations  $\sigma_k$ :

$$(\sigma_k^{j+1})^2 = \frac{1}{\sum_{i=1}^N \gamma_{i,k}^j} \sum_{i=1}^N \gamma_{i,k}^j (x_i - \mu_k)^2 \quad (5.13)$$



# 6 Decision Trees

- Easy to *interpret*
- Can classify both *categorical* and *numerical* data (output must be categorical)
- *Prior knowledge* can be included
- Trees created from *numeric data* sets can be quite *complex*
- Decision tree algorithms are *unstable*: slight alterations of even a single training point can lead to radically different decisions:

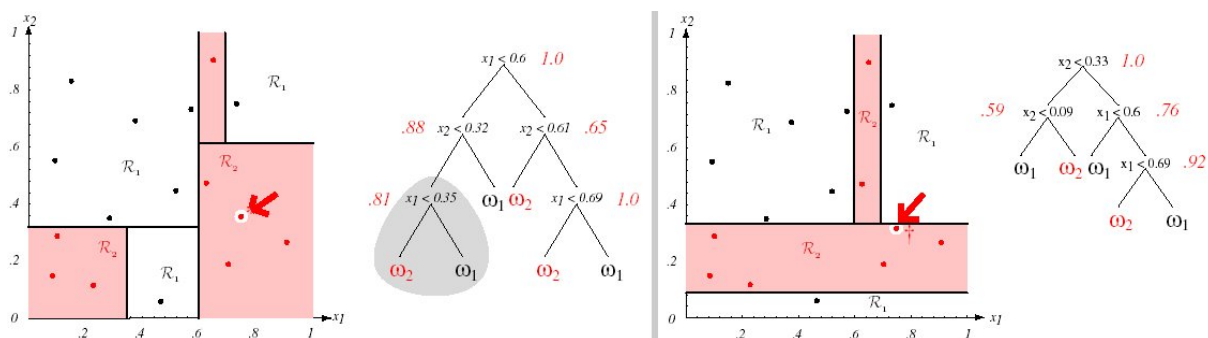


Figure 6.1: Change of decision boundaries and decision trees with slight alteration of one training sample.

## 6.1 Structure

- Binary or not?

## 6.2 Types of questions

### 6.2.1 Nominal data

- Depend on developers intuition about the task
- Usually asking for presence/absence of words is a good baseline

### 6.2.2 Continuous data

Feature vector  $(x_1, \dots, x_n)$

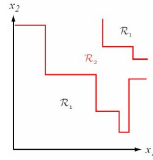
## 6 Decision Trees

### Simple partitioning

Creates a binary tree with all decision boundaries parallel to the axis.

Question type:  $x_i < a$

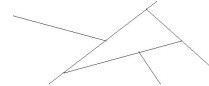
Feature space partitioning:



### Binary space partition trees

Question type:  $\sum_i a_i x_i < \alpha$

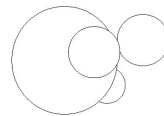
Feature space partitioning:



### Sphere trees

Question type:  $\sum_i (z_i - x_i)^2 < \alpha$

Feature space partitioning:



### Nonlinear generalization

Question type:  $\Psi(x, a) > 0$

$\Psi$  is an arbitrary nonlinear function; may be different at each node.

Issue: lack of simplicity.

## 6.3 Splitting

### 6.3.1 Impurity of a node

$P_N(\omega_i)$ : Probability of class  $\omega_i$  at node N

**Misclassification impurity**

$$i(N) = 1 - \max P_N(\omega_i) \quad (6.1)$$

Measures fraction of data that does not belong to the dominant class.  
Issue: too greedy for multiple splits.

**Entropy impurity**

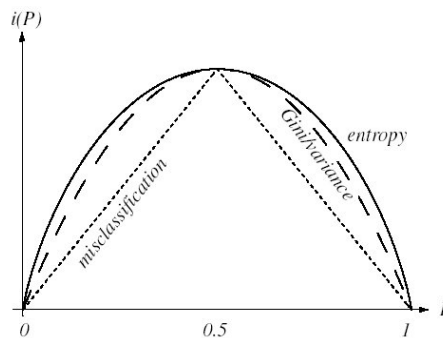
$$i(N) = - \sum_i P_N(\omega_i) \log_2 P_N(\omega_i) \quad (6.2)$$

Very popular; no practical difference to Gini impurity.

**Gini impurity**

$$i(N) = \frac{1}{2} \sum_i (1 - P_N(\omega_i))^2 \quad (6.3)$$

Very popular; no practical difference to entropy impurity.

**Impurity functions in 2-class problem****6.3.2 Training algorithms (Greedy)**

Until the tree is large enough:

    Test all nodes N:

        Test all questions Q:

            Calculate change in impurity

        Add node/question to achieve largest drop in impurity

This is one example of a more generic tree-growing methodology known as CART (Classification and Regression Trees).

**Influence of amount of training data**

- High variability for small amounts of training data
- Saturation for large amounts of data

**6.4 Pruning**

Inverse method to Splitting: merge nodes for avoiding overfitting your training data. It's usually possible to prune a lot of nodes from the decision tree without sacrificing performance on the test set.