Detection & Pattern Recognition

José Heinz

Electrical Engineering & Information Technology

heinzmollersantos@gmail.com

Resume

Here is the exam preparation for the course Detection & Pattern Recognition.

1. Mathematical notations and symbols

- true class: ω
- estimated class: $\hat{\omega}$
- probability of an event A: $P(A)$
- a priori of class ω_i: $P(\omega = \omega_i) = P(\omega_i)$
- PDF of a RV $\underline{x}$: $p(\underline{x})$
- likelihood of $\underline{x}$: $p(\underline{x} \mid \omega_i)$
- a posteriori prob. of class ω_i: $P(\omega_i \mid \underline{x})$
- expectation of a RV $\underline{x}$: $\underline{\mu} = \mathsf{E}(\underline{x}) = \int \underline{x} p(\underline{x}) d\underline{x}$
- covariance matrix of $\underline{x}$: $\mathbf{C} = \mathsf{E}[(\underline{x} - \underline{\mu})(\underline{x} - \underline{\mu})^T]$
- discriminant function for class ω_i: $f_i(\underline{x})$

There are for binary detectrion four cases. Let H_0 and H_1 be the null hypothesis and the alternatice hypothesis for binary detection. Let $H \in \{H_0, H_1\}$ be the unknown true state. Let $\hat{H}(\underline{x}) \in \{H_0, H_1\}$ be the estimated state of H.

$\hat{H}\|H$	$H = H_0$	$H = H_1$
$\hat{H} = H_0$	correct rejection **true negative** (TN)	miss **false negative** (FN) type II error
$\hat{H} = H_1$	false alarm **false positive** (FP) type I error	detection, hit **true positive** (TP)

Figure 1: *Four possible combinations for $\hat{H}$*

One can define the performance measures of this example for $\hat{H}(\underline{x})$ as for the
TN-Rate: $P_{TN} = P(\hat{H} = H_0 \mid H = H_0)$ with the cond. probability.
Analog for the FP-Rate $P_{FP} = P(\hat{H} = H_1 \mid H = H_0)$.
For the FN-Rate $P(\hat{H} = H_0 \mid H = H_1)$
for the TP-Rate $P(\hat{H} = H_1 \mid H = H_1)$
Magically this happens:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$
$$P(A \cap B) = 0 \iff A \cap B = \emptyset$$
$$\Rightarrow P_{TN} + P_{FP} = 1$$
$$\Rightarrow P_{FN} + P_{TP} = 1$$

Now we can define the confusion matrix as:

$$n_{ij} = \#(\hat{\omega} = \omega_i, \omega = \omega_j) \in \mathbb{N}$$
$$[n_{ij}]_{ij} \in (c \times c),$$

where c is the number of classes. The properties of this crazy ass matrix are

- column sum: $n_{:j} = \sum_{i=1}^{c} n_{ij} = \#(\omega = \omega_j)$
- row sum: $n_{i:} = \sum_{j=1}^{c} n_{ij} = \#(\hat{\omega} = \omega_i)$
- Matrix sum: $N = \sum_{i=1}^{c} \sum_{j=1}^{c} n_{ij} = total$

Normalization of n_{ij} to make a probability:

- marginal probability: $P_j = P(\omega = \omega_j) = \frac{n_{:j}}{N}$ and $P(\hat{\omega} = \omega_i) = \frac{n i:}{N}$ with $\sum_{j=1}^{c} P_j = 1$
- joint probability: $P(\hat{\omega} = \omega_i, \omega = \omega_j) = \frac{n_{ij}}{N}$ with $\sum_{i=1}^{c} \sum_{j=1}^{c} P_{ij} = 1$
- conditional probability (column sum normalization): $\hat{P_{ij}} = P(\hat{\omega} = \omega_i \mid \omega = \omega_j) = \frac{n_{ij}}{n_{:j}}$ which can be interpreted as the TN, FP, FN, TP- Rate.
- there is also row sum normalization.

Example:

1.4 Confusion matrix

E1.10: Confusion matrix of fish sorting (2)

$$[n_{ij}] = \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} 50 & 5 \\ 10 & 35 \end{bmatrix}, \qquad \sum_i \sum_j n_{ij} = 100$$

Marginal probabilities

column: 60% salmon and 40% sea bass
row: 55% classified salmon and 45% classified sea bass

Joint probabilities

$$[P_{ij}] = [P(\hat{\omega} = \omega_i, \omega = \omega_j)] = \begin{bmatrix} 50\% & 5\% \\ 10\% & 35\% \end{bmatrix},$$

Conditional probabilities

$$[\bar{P}_{ij}] = [P(\hat{\omega} = \omega_i | \omega = \omega_j)] = \begin{bmatrix} 50/60 & 5/40 \\ 10/60 & 35/40 \end{bmatrix} = \begin{bmatrix} P_{\text{TN}} & P_{\text{FN}} \\ P_{\text{FP}} & P_{\text{TP}} \end{bmatrix},$$
$$[P(\omega = \omega_j | \hat{\omega} = \omega_i)] = \begin{bmatrix} 50/55 & 5/55 \\ 10/45 & 35/45 \end{bmatrix}.$$

Figure 2: *Confusion matrix example*

2. Bayesian decision theory

Given two events, for the general case; the Bayesian theorem/rule/formula is:

$$P(A \mid B) = \frac{P(A, B)}{P(B)} = \frac{P(A \mid B) P(B)}{P(A)}.$$

This nice formula allows the conversion between $P(A \mid B)$ and $P(B \mid A)$.
Now we may consider a RV $\underline{x}$ and classes $\omega_i, i = 1, 2, ..., c$.

$$P(\omega_j \mid \underline{x}) = \frac{p(\underline{x}, \omega_j)}{p(\underline{x})} = \frac{p(\underline{x} \mid \omega_j) P(\omega_j)}{p(\underline{x})} = \frac{p(\underline{x}, \omega_j) P(\omega_j)}{\sum_{k=1}^{c} p(\underline{x} \mid \omega_k) P(\omega_k)}$$

i.e.,

$$posterior = \frac{likelihood \cdot prior}{evidence}$$

2.1 Minimum bayesian risk decision

The joint prob. matrix $[P_{ij}]$ can be derived from the confusion matrix $[n_{ij}]_{ij}$ with the following formula as stated: $P_{ij} = P(\hat{\omega} = \omega_i, \omega = \omega_j) = \frac{n_{ij}}{N}$. These are $c^2 - 1$ measures. With a few formulations we can derive the recognition rate and error rate:

$$1 = \sum_{i=1}^{c} \sum_{j=1}^{c} P_{ij} = \sum_{i=1}^{c} P_{ii} + \sum_{i \neq j}^{c} P_{ij}$$
$$= \underbrace{P(\hat{\omega} = \omega)}_{Recognition\ Rate} + \underbrace{P(\hat{\omega} \neq \omega)}_{Error\ Rate (ER)}.$$

We can now define the loss of a decision when the class is not correctly chosen:

$$l_{ij} = l(\hat{\omega} = \omega_i, \omega = \omega_j) \geq 0, \ 1 \leq i, j \leq c,$$

where tipically $l_{ii} = 0$ and $l_{ij} > 0 \ \forall i \neq j$.
With the loss we can penalize the decisions like this and call it Bayesian Risk BR:

$$BR = \sum_i \sum_j l_{ij} P_{ij} = \mathsf{E}_{\hat{\omega}, \omega}[l(\hat{\omega}, \omega)] \geq 0$$

and if the loss function is the $0/1 - loss$: $l_{ij} = 1 \ \forall i \neq j$, meaning that all errors are equally bad, then the BR simplifies to the Error Rate ER:

$$BR = \sum \sum_{i \neq j} P_{ij} = ER.$$

as expectation of the loss over $\hat{\omega}$ and ω.
Now we get to define if the loss is dependent of the input vector $\underline{x}$. Knowing also that the estimated class $\hat{\omega}$ is also dependent of the RV $\underline{x}$. Then it follows that the loss is also dependent of the RV such that $l = l(\hat{\omega}(x), \omega)$.
Then it can be derived that the expectation function over the loss is defined over $(\underline{x}, \omega)$, meaning; computing the average loss for all $(\underline{x}, \omega)$. In the following definition we define d as the number of dimensions or characteristics of $\underline{x} \in \mathbb{R}^d$:

$$BR = \mathsf{E}[l(\hat{\omega}(\underline{x}), \omega)] = \int_{\mathbb{R}^d} \sum_{j=1}^{c} l(\hat{\omega}(\underline{x}), \omega = \omega_j) p(\underline{x}, \omega_j) d\underline{x}$$
$$= \int_{\mathbb{R}^d} \underbrace{\sum_{j=1}^{c} l(\hat{\omega}(\underline{x}), \omega = \omega_j) P(\omega_j, \underline{x})}_{conditional\ Risk:\ R(\hat{\omega} | \underline{x})} p(\underline{x}) d\underline{x}$$
$$\int_{\mathbb{R}^d} R(\hat{\omega} \mid \underline{x}) p(\underline{x}) d\underline{x}.$$

Meaning that in order to minimize the Bayesian Risk, we have to minimize the Coditional Risk, and therefore minimizing the loss of the system:

$$min\ BR \Rightarrow min_{\hat{\omega}}\ R(\hat{\omega} \mid \underline{x})$$
$$min_{\hat{\omega}} \sum_{j=1}^{c} l(\hat{\omega}(\underline{x}), \omega = \omega_j) P(\omega_j, \underline{x}).$$

However, we are not done yet, bro... We have the following dependencies:

- $l(\hat{\omega}, \omega)$ which is chosen by YOU
- $P(\omega_j \mid \underline{x})$ calculated with BAYESIAN THEOREM
- $\hat{\omega}(\underline{x})$ TO BE DESIGNED!

2.2 MBR: First design of $\hat{\omega}(\underline{x})$

Let $\hat{\omega}(\underline{x}) = \hat{\omega}_{MBR}(\underline{x})$ be the MBR decision/detector or classifier be defined as

$$\hat{\omega}_{MBR}(\underline{x}) = argmin_{\hat{\omega}} R(\hat{\omega} \mid \underline{x}) = argmin_{\omega_i} R(\hat{\omega}_i = \omega_i \mid \underline{x}).$$

For this calculation one can do the following algorithm:

i.e. for any given $\underline{x}$:

- calculate c values $R(\hat{\omega}(\underline{x}) \mid \underline{x}) = \sum_{j=1}^{c} l_{ij} P(\omega_j \mid \underline{x})$ for $1 \leq i \leq c$
- look for minimum

or mathematically easier:

$$\underbrace{\begin{bmatrix} l_{11} & \dots & l_{1j} \\ \vdots & \dots & \vdots \\ l_{i1} & \dots & l_{ij} \end{bmatrix}}_{c \times c} \underbrace{\begin{bmatrix} P(\omega_1 \mid \underline{x}) \\ \vdots \\ P(\omega_c \mid \underline{x}) \end{bmatrix}}_{c \times 1} = \underbrace{\begin{bmatrix} R(\hat{\omega} = \omega_1 \mid \underline{x}) \\ \vdots \\ R(\hat{\omega} = \omega_c \mid \underline{x}) \end{bmatrix}}_{\Rightarrow \hat{\omega}(\underline{x}) = \text{Row of min}}$$

For the Bayesian Rule vector, ignore the denominator $p(\underline{x})$:
$P(\omega_j \mid \underline{x}) = \frac{p(\underline{x}, \omega_j) P(\omega_j)}{p(\underline{x})} \propto p(\underline{x}, \omega_j) P(\omega_j)$
because of independency of ω_j

2.3 Two-class MBR: Likelihood Ratio Test

For two classes $c = 2$, the algorithm can be simplified as the likelihood ratio test (LRT):

$$LR = \frac{p(\underline{x} \mid \omega_2)}{p(\underline{x}, \omega_1)} \underset{\omega_1}{\overset{\omega_2}{\lessgtr}} \frac{l_{21} - l_{11}}{l_{12} - l_{22}} \cdot \frac{P(\omega_1)}{P(\omega_2)} = \underbrace{\gamma}_{Ratio}$$

and applying the $0/1 - loss$:

$$LR = \frac{p(\underline{x} \mid \omega_2)}{p(\underline{x}, \omega_1)} \underset{\omega_1}{\overset{\omega_2}{\lessgtr}} \frac{P(\omega_1)}{P(\omega_2)} = \underbrace{\gamma}_{Ratio}$$

and also the conditional Risk changes when applied the $0/1 - loss$ to

$$R(\hat{\omega} = \omega_i \mid \underline{x}) = \sum_{j=1}^{c} l_{ij} P(\omega_j \mid \underline{x})$$
$$\overset{0/1-loss}{=} \underbrace{\sum_{j=1, j \neq i}^{c} P(\omega_j \mid \underline{x}) + P(\omega_i \mid \underline{x})}_{=1} - P(\omega_i \mid \underline{x})$$
$$= 1 - P(\omega_i \mid \underline{x})$$

Hence $\underset{\omega_i}{min} ER$ is equivalent to $\underset{\omega_i}{max} P(\omega_i \mid \underline{x})$ which is the Max. Aposterior prob. **(MAP)**.

2.4 Maximum Likelihood; Special case for MAP

Let the probability of c classes be equal, meaning $P(\omega_i) = \frac{1}{c}, \ \forall i$. The for the computation of the posterior:

$$P(\omega_i \mid \underline{x}) = p(\underline{x} \mid \omega_i) \cdot \frac{P(\omega_i)}{p(\underline{x})}$$
$$\Rightarrow \underset{\omega_i}{max}\ P(\omega_i \mid \underline{x}) \Leftrightarrow \underbrace{\underset{\omega_i}{max}\ p(\underline{x} \mid \omega_i)}_{Maximum\ Likelihood\ (ML)}$$

which is possible because $p(\underline{x})$ is independent of ω_i

2.5 Discriminant Functions using MBR, MAP, ML Decision

The main goal of this section is to create a unified framework with c discriminant functions $f_i(\underline{x})$ for $\omega_i \ 1 \leq i \leq c$. The main objective is also to find the best estimated class $\hat{\omega}(\underline{x}) = \underset{i}{argmax}\ f_i(\underline{x})$ with $\hat{\omega}(\underline{x}) \in \{\omega_1, ..., \omega_c\}$.

Defining first a few elements:

- Decision Region: $Ri = \{\underline{x} \mid f_i(\underline{x}) > f_j(\underline{x}) \ \forall j \neq i\}$
- — Boundary: $\beta_{ij} = \{\underline{x} \mid f_i(\underline{x}) = f_j(\underline{x}) > f_k(\underline{x}) \ \forall k \neq i, j\}$

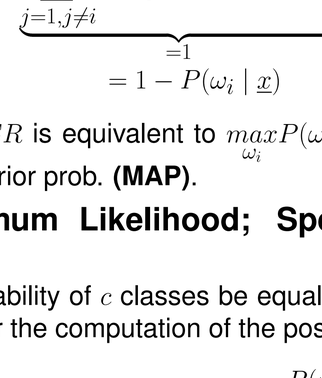


Figure 3: *Example of the representative space for $c = 10$ (10 classes) with $d = 2$ (2 dimensions)*

Discriminant functions apply another function $\phi()$ to the MAP and ML. The different types of discriminant functions are the following:

- **MBR**: $f_i(\underline{x}) = R(\hat{\omega} = \omega_j \mid \underline{x}) = \sum_{j=1}^{c} l_{ij} P(\omega_j \mid \underline{x})$ (no log function)
- **MAP**: $f_i(\underline{x}) = P(\omega_i \mid \underline{x}) \sim p(\underline{x} \mid \omega_i) \cdot P(\omega_i)$ or with the log fct: $f_i(\underline{x}) = ln(p(\underline{x} \mid \omega_i)) + ln(P(\omega_i))$
- **ML**: $f_i(\underline{x}) = p(\underline{x} \mid \omega_i)$ or $ln(p(\underline{x} \mid \omega_i))$

2.6 MAP decision for Gaußian likelihoods

Assumptions:

- Classes $c \geq 2$
- Features $\underline{x} \in \mathbb{R}^d, d \geq 1$
- MAP: $0/1 - loss$
- Known priors $P(\omega_i) = P_i$
- Known Gaußian Likeligood $(\underline{x}, \omega_i) \sim \mathbf{N}(\underline{\mu}_i \mid \mathbf{C}_i)$

The MAP discriminant function for class ω_i is then described by using the log function:

$$f_i(\underline{x}) = ln\ p(\underline{x} \mid \omega_i) + ln\ P_i$$
$$= \underbrace{-\frac{d}{2} ln(2\pi)}_{const} - \frac{1}{2} ln \mid \mathbf{C}_i \mid -\frac{1}{2}(\underline{x} - \underline{\mu}_i)^T \mathbf{C}_i^{-1} (\underline{x} - \underline{\mu}_i) + ln\ P_i$$

Case A: $\mathbf{C}_i = \sigma^2 \mathbf{I}$
For decision region:

$$f_i(\underline{x}) = -\frac{1}{2\sigma^2} \mid\mid \underline{x} - \underline{\mu}_i \mid\mid^2 + ln\ P_i$$
$$\text{with } \mid\mid \underline{x} \mid\mid = const$$
$$\Rightarrow f_i(\underline{x}) = \underbrace{(\frac{\underline{\mu}_i}{2})^T}_{\underline{w}_i} \underline{x} + \underbrace{ln\ P_i - \frac{1}{2\sigma^2} \mid\mid \underline{\mu}_i \mid\mid^2}_{w_{i0} \text{ (offset)}}$$
$$f_i(\underline{x}) = \underline{w}_i^T \underline{x} + w_{i0}$$

as an affine function for every class.
For decision boundary β_{ij} between ω_i and ω_j:

$$f_i(\underline{x}) - f_j(\underline{x}) = 0$$
$$(\underline{w}_i - \underline{w}_j)^T \underline{x} + (w_{i0} - w_{j0}) = 0.$$

Properties:

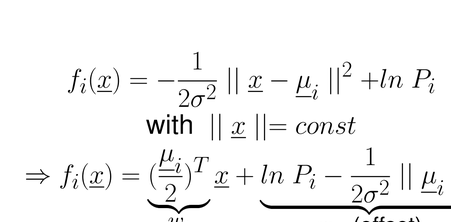
- a linear flat decision boundary
- $\underline{\mu}_i - \underline{\mu}_j = \sigma^2(\underline{w}_i - \underline{w}_j) \perp \beta_{ij}$
- if $P_i = P_j$: $\max_i f_i(\underline{x}) \triangleq \min_i \mid\mid \underline{x} - \underline{\mu}_i \mid\mid$ as nearest mean

E2.5: MAP decision for Gaussian likelihood – Case A

- $d = 2$ features $\underline{x} = [x_1, x_2]^T$ with Gaussian likelihood $\underline{x} \mid \omega_i \sim N(\underline{\mu}_i, \mathbf{C}_i)$
- equal covariance matrices $\mathbf{C}_i = \sigma^2 \mathbf{I}$,
- equal priors $P(\omega_i) = \text{const}$

In this case, the MAP decision rule simplifies to the **nearest mean classifier** (see section 4.2). Each point in the following **Voronoi diagram** represents a class center $\underline{\mu}_i$ and the lines represent the decision boundaries.

a) $c = 4$ classes b) $c = 10$ classes



MATLAB: `voronoi`

Figure 4: *Example of MAP decision*

Case B: $\mathbf{C}_i = \mathbf{C} \neq \sigma^2 \mathbf{I}$

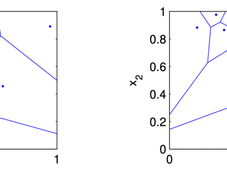
$$f_i(\underline{x}) = -\frac{1}{2} \underbrace{(\underline{x} - \underline{\mu}_i)^T \mathbf{C}^{-1} (\underline{x} - \underline{\mu}_i)}_{\text{Mahalanobis Distance}} + ln\ P_i$$
$$= \underline{w}_i^T \underline{x} + w_{i0}$$
$$\text{with } \underline{w}_i = \mathbf{C}^{-1} \underline{\mu}_i$$
$$w_{i0} = ln\ P_i - \frac{1}{2} \underline{\mu}_i^T \mathbf{C}^{-1} \underline{\mu}_i$$

The **Euclidean distance** between $\underline{x}$ and $\underline{\mu}$ is defined by

$$D_{\text{Eucl}}(\underline{x}, \underline{\mu}) = \sqrt{(\underline{x} - \underline{\mu})^T (\underline{x} - \underline{\mu})}.$$

All $\underline{x}$ with the same Euclidean distance to $\underline{\mu}$ lie on a ball centered at $\underline{\mu}$. Given two centers $\underline{\mu}_i$ and $\underline{\mu}_j$, all $\underline{x}$ having the same Euclidean distance to $\underline{\mu}_i$ and $\underline{\mu}_j$ lie on the red solid line.

Case A: $\mathbf{C}_i = \mathbf{C}_j = \sigma^2 \mathbf{I}$



The **Mahalanobis distance** between $\underline{x}$ and $\underline{\mu}$ for a given covariance matrix $\mathbf{C}$ is defined by

$$D_{\text{Maha}}(\underline{x}, \underline{\mu}, \mathbf{C}) = \sqrt{(\underline{x} - \underline{\mu})^T \mathbf{C}^{-1} (\underline{x} - \underline{\mu})}.$$

All $\underline{x}$ with the same Mahalanobis distance to $\underline{\mu}$ lie on an ellipsoid centered at $\underline{\mu}$, see slide 2-10. Given two centers $\underline{\mu}_i, \underline{\mu}_j$ and two identical covariance matrices $\mathbf{C}_i = \mathbf{C}_j \neq \sigma^2 \mathbf{I}$, all $\underline{x}$ having the same Mahalanobis distance $D_{\text{Maha}}(\underline{x}, \underline{\mu}_i, \mathbf{C}_i) = D_{\text{Maha}}(\underline{x}, \underline{\mu}_j, \mathbf{C}_j)$ lie on the red solid line.

Case B: $\mathbf{C}_i = \mathbf{C}_j \neq \sigma^2 \mathbf{I}$

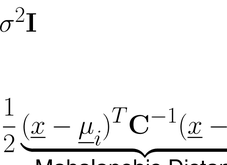


Figure 5: *Mahalanobis vs Euclidean distance*

Case C: $\mathbf{C}_i \neq \mathbf{C}_j$

E2.6: MAP decision for Gaussian likelihood – Case C

Assume $P_1 = P_2 = 1/2$ and

$$\underline{\mu}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \underline{\mu}_2 = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, \quad \mathbf{C}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{C}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

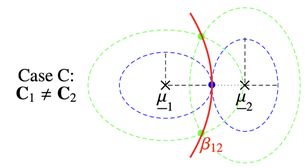
The MAP decision boundary β_{12} is then given by

$$\ln |\mathbf{C}_1| + (\underline{x} - \underline{\mu}_1)^T \mathbf{C}_1^{-1} (\underline{x} - \underline{\mu}_1) = \ln |\mathbf{C}_2| + (\underline{x} - \underline{\mu}_2)^T \mathbf{C}_2^{-1} (\underline{x} - \underline{\mu}_2)$$

or after some simplifications

$$x_1^2 - x_2^2 - 40x_1 + 200 = 0.$$

Case C: $\mathbf{C}_1 \neq \mathbf{C}_2$



Summary of MAP decision boundaries for Gaussian likelihood

	distance of $\underline{x}$ on β_{ij} to $\underline{\mu}_i$ and $\underline{\mu}_j$	decision boundary β_{ij}
Case A: $\mathbf{C}_i = \sigma^2 \mathbf{I}$	same Euclidean distance	affine in $\underline{x}$, orthogonal to $\underline{\mu}_i - \underline{\mu}_j$
Case B: $\mathbf{C}_i = \mathbf{C}_j \neq \sigma^2 \mathbf{I}$	same Mahalanobis distance	affine in $\underline{x}$, not orthogonal to $\underline{\mu}_i - \underline{\mu}_j$
Case C: $\mathbf{C}_i \neq \mathbf{C}_j$	different Mahalanobis distances	quadratic in $\underline{x}$

Figure 6: *Summary*

For minimax theory and Neyman-Pearson lemma, the lecture notes.

3. Detection

Given:

- data $\underline{x}$
- likelihood $p(\underline{x} \mid \omega)$
- prior (optional)
- loss (optional)

Desired is a detector with good detector performance for **binary** detection. We use now the notation H_0 and H_1 instead of ω. Defining also the estimated hypothesis $\hat{H}(\underline{x}) \in \{H_0, H_1\}$
We define a received signal as a vector with a time dependency n. The vector $\underline{x} = [x(0), x(1), ...x(d-1)]^T \in \mathbb{R}^d$.
We can define two hypotheses. $H_0 : x(n) = z(n)$ and $H1 : x(x) = A + z(n)$ and $z(n)$ as gaußan noise $\mathbf{N}(0, \sigma^2)$. and the losses l_{ij} for $\hat{H} \in \{H_0, H_1\}$.

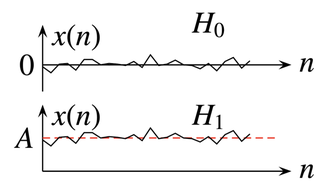


Figure 7: *Representation of the signals to be detected*

We got also the known priors $P_0 = P(H_0)$ and $P_1 = P(H_1) = 1 - P_0$. Then the likelihoods are

$$p(\underline{x} \mid H_0) = \prod_{n=0}^{d-1} p(x(n) \mid H_0) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{(-\frac{1}{2\sigma^2} \sum_{n=0}^{d-1} x^2(n))}$$
$$p(\underline{x} \mid H_1) = \prod_{n=0}^{d-1} p(x(n) \mid H_0) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{(-\frac{1}{2\sigma^2} \sum_{n=0}^{d-1} (x(n)-A)^2)}$$

One can use a general LRT detector:

$$ln \frac{p(\underline{x} \mid H_1)}{p(\underline{x} \mid H_0)} = -\frac{1}{2} [\sum_{n=0}^{d-1} (x(n) - A)^2 - \sum_{n=0}^{d-1} x^2(n)]$$
$$= \underbrace{\frac{dA^2}{\sigma^2}}_{\rho > 0} [\underbrace{\frac{1}{A\,d} \sum_{n=0}^{d-1} x(n)}_{t(\underline{x})} - \frac{1}{2}]$$
$$= \rho[t(\underline{x}) - \frac{1}{2}] \underset{H_0}{\overset{H_1}{\lessgtr}} ln(\underbrace{\tilde{\gamma}}_{\text{LR}})$$
$$\Rightarrow t(\underline{x}) \underset{H_0}{\overset{H_1}{\lessgtr}} \frac{1}{\rho} ln\gamma + \frac{1}{2} = \gamma$$

where γ is the new threshold for $t(\underline{x})$. For this notation is:

- Test: $t(\underline{x}) = \frac{1}{Ad} \sum_{n=0}^{d-1} x(n) = \frac{1}{Ad}$
- SNR (signal to noise ratio) $\frac{A^2}{\sigma^2}$ of $x(n) \mid H_1 = x(n) = Az(n) \sim \mathbf{N}(A, \sigma^2)$
- SNR $\rho = \frac{dA^2}{\sigma^2}$ of $\bar{x} \mid H_1 = A + \bar{z} \sim \mathbf{N}(A, \frac{\sigma^2}{d})$

Distributions of the function $t(\underline{x})$:

$$H_0 : t(\underline{x}) = \frac{1}{A} \cdot \bar{z} \sim \mathbf{N}(0, \frac{\sigma^2}{A^2 d}) = \mathbf{N}(0, \frac{1}{\rho})$$
$$H_1 : t(\underline{x}) = \frac{1}{A}(A + \bar{z}) \sim \mathbf{N}(1, \frac{1}{\rho})$$

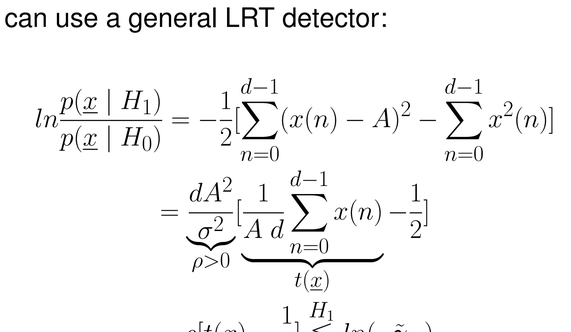


Figure 8: *False Alarm rate in red*

The next step is to minimize the P_{FA}, which is the false alarm rate. The line on figure 8 is the γ which is described in the last equations. For this we remember from Mathematics the Standard Gaußian distribution and Q function:

Standard Gaussian distribution and Q-function

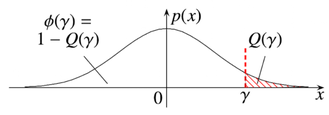
A random variable x is **standard Gaussian** or **standard normal** distributed if it has the PDF

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

It has zero mean and unit variance. The short notation is $x \sim N(0, 1)$. The **cumulative distribution function (CDF)** of x is $\phi(x) = \int_{-\infty}^{x} p(z)\mathrm{d}z$ with $p(x) = \mathrm{d}\phi(x)/\mathrm{d}x$. The **Q-function** is defined as

$$Q(x) = 1 - \phi(x) = \int_x^\infty p(z)\mathrm{d}z = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-z^2/2}\mathrm{d}z.$$

$Q(\gamma)$ describes the probability $P(x > \gamma)$ that x obtains a value larger than γ. It corresponds to the area below $p(x)$ from γ to ∞ and decreases from 1 to 0 as γ increases.



γ	0	1.28	1.64	2.33	3.09
$Q(\gamma)$	50%	10%	5%	1%	0.1%

Figure 9: *Q-Function*

We can then define the false alarm rate as:

$$P_{FA} = P(\hat{H} = H_1 \mid H = H_0) = P(t > \gamma \mid H_0), \text{ with } (t \mid H_0) \sim \mathbf{N}(0, \frac{1}{\rho})$$
$$= P(\underbrace{\sqrt{\rho} t}_{\mathbf{N}(0,1)} > \sqrt{\rho}\gamma \mid H_0) = Q(\sqrt{\rho}\gamma)$$

and we also define the Detection Rate:

$$P_D(\gamma) = P(\hat{H} = H_1 \mid H = H_1) = P(t > \gamma \mid H_1), \text{ with } (t \mid H_1) \sim \mathbf{N}(1, \frac{1}{\rho})$$
$$= P(\underbrace{\sqrt{\rho}(t-1)}_{\mathbf{N}(0,1)} > \sqrt{\rho}(\gamma - 1) \mid H_1) = Q(\sqrt{\rho}(\gamma - 1))$$

E3.1: Detection of DC in AWGN (cont.)

In E3.1 we assumed to know the priors P_0, P_1 and Gaussian likelihoods $P(\underline{x}|H_0)$, $P(\underline{x}|H_1)$ including the DC value A and noise variance σ^2. We studied different detectors (MBR, MAP, ML, NP). Are they still possible if A and/or σ^2 are unknown? Remember that the detector is

$$t(\underline{x}) = \frac{\bar{x}}{A} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \qquad \text{or} \qquad \bar{x} = \frac{1}{d} \sum_{n=0}^{d-1} x(n) \underset{H_0}{\overset{H_1}{\gtrless}} A\gamma,$$

where the threshold $A\gamma$ for $\bar{x}$ depends on the type of detector as well as on A and σ^2.

Detector	γ	$A\gamma$	A unknown	σ^2 unknown
MAP	$\frac{1}{2} + \frac{1}{\rho} \ln\left(\frac{P_0}{P_1}\right)$	$\frac{A}{2} + \frac{\sigma^2}{dA} \ln\left(\frac{P_0}{P_1}\right)$	No	No
ML	$\frac{1}{2}$	$\frac{A}{2}$	No	Yes
NP	$\frac{1}{\sqrt{\rho}} Q^{-1}(\alpha)$	$\frac{\sigma}{\sqrt{d}} Q^{-1}(\alpha)$	Yes	No

$\rho = d\frac{A^2}{\sigma^2}$

Hence, different detectors require to know different parameters.

Figure 10: *From lecture table*

4. Supervised Learning

For this approach we have to define the following:

- Bayesian decision theory: Assumed complete class-coditional likelihoods $p(\underline{x} \mid \omega_i)$, see chapter 2.
- Machine Learning: the likelihoods $p(\underline{x} \mid \omega_i)$ are not completely known. Hence the bayesian decision theory cannot be applied.
- Supervised Learning: Here we have the correct class label for each training sample. $x_n \rightarrow y_n$

we also got the following:

Trainig set: $S = \{(\underline{x}_1, y_1), ..., (\underline{x}_N, y_N)\}$
$\underline{x}_N \in \mathbb{R}^d$: as the feature vector
$y_n \in \{\omega_1, ..., \omega_c\}$: as correct class label for $\underline{x}_n$
N_j training samples for class ω_j
$N = \sum_{j=1}^{c} N_j$ as the training set.

The objective is to learn a classifier: $\hat{y}(\underline{x}) \in \{\omega_1, \omega_2, ..., \omega_c\}$ from S.

4.1 Nearest Mean

This is old, simple and poor. Estimate $\hat{\mu}_j$ from S. Then the nearest class center wins.
Euclidean distance: $D_{Eucl}(\underline{x}, \underline{\mu}) = \sqrt{(\underline{x} - \underline{\mu})^T (\underline{x} - \underline{\mu})}$
+simple
+linear decision boundary
-sensitive to feature scaling

Euclidean distance vs Mahalanobis distance

Two popular distance measures in the feature space $\mathbb{R}^d$ are the **Euclidean distance** $D_{\text{Eucl}}(\underline{x}_1, \underline{x}_2) = \sqrt{(\underline{x}_1 - \underline{x}_2)^T (\underline{x}_1 - \underline{x}_2)}$ and the **Mahalanobis distance** $D_{\text{Maha}}(\underline{x}_1, \underline{x}_2, \mathbf{C}) = \sqrt{(\underline{x}_1 - \underline{x}_2)^T \mathbf{C}^{-1} (\underline{x}_1 - \underline{x}_2)}$, where $\mathbf{C}$ is the covariance matrix of $\underline{x}_i$.

One advantage of the Mahalanobis distance is its invariance with respect to a linear transformation $\tilde{\underline{x}} = \mathbf{A}\underline{x}$ of the features space (including scaling, rotation etc.), where $\mathbf{A}$ is any square invertible matrix. The new covariance matrix of $\tilde{\underline{x}}$ is

$$\tilde{\mathbf{C}} = \mathrm{E}\left((\tilde{\underline{x}} - \mathrm{E}\tilde{\underline{x}})((\tilde{\underline{x}} - \mathrm{E}\tilde{\underline{x}})^T\right) = \mathbf{A}\mathbf{C}\mathbf{A}^T.$$

Therefore,

$$D_{\text{Maha}}(\tilde{\underline{x}}_1, \tilde{\underline{x}}_2, \tilde{\mathbf{C}}) = \sqrt{(\tilde{\underline{x}}_1 - \tilde{\underline{x}}_2)^T \tilde{\mathbf{C}}^{-1} (\tilde{\underline{x}}_1 - \tilde{\underline{x}}_2)} = \sqrt{(\underline{x}_1 - \underline{x}_2)^T \mathbf{A}^T \cdot (\mathbf{A}\mathbf{C}\mathbf{A}^T)^{-1} \cdot \mathbf{A}(\underline{x}_1 - \underline{x}_2)}$$
$$= \sqrt{(\underline{x}_1 - \underline{x}_2)^T \mathbf{C}^{-1} (\underline{x}_1 - \underline{x}_2)} = D_{\text{Maha}}(\underline{x}_1, \underline{x}_2, \mathbf{C}).$$

This is not the case for the Euclidean distance. However, the Mahalanobis distance needs to estimate $\mathbf{C}$.

Figure 11: *From the lecture, Mahalanobis and Pros, Cons.*

Given
training samples $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
Parameter
a suitable distance metric $D(\underline{x}, \underline{\mu})$
Training
- estimate all class centers
  $\hat{\underline{\mu}}_j = \frac{1}{N_j} \sum_{y_n=\omega_j} \underline{x}_n \quad (1 \le j \le c)$
- estimate all class covariance matrices when using the Mahalanobis distance
  $\hat{\mathbf{C}}_j = \frac{1}{N_j} \sum_{y_n=\omega_j} (\underline{x}_n - \hat{\underline{\mu}}_j)(\underline{x}_n - \hat{\underline{\mu}}_j)^T \quad (1 \le j \le c)$

Classification
for each new feature vector $\underline{x}$,
- calculate the distance of $\underline{x}$ to all class centers
  $d_j = D_{\text{Eucl}}(\underline{x}, \hat{\underline{\mu}}_j)$ or $D_{\text{Maha}}(\underline{x}, \hat{\underline{\mu}}_j, \hat{\mathbf{C}}_j) \quad (1 \le j \le c)$
- choose the class with the smallest distance to $\underline{x}$
  $\hat{y}(\underline{x}) = \arg\min_{\omega_j} d_j$

Comments
- multiclass classifier
- computational complexity
  – training: $O(Nd)$ for $\hat{\underline{\mu}}_j$ and $O(Nd^2)$ for $\hat{\underline{\mathbf{C}}}_j$
  – classification: c distance calculations

4.2 k-Nearest Neighbor (KNN)

Look at the lecture notes.

Table 4.2: k-Nearest neighbor (kNN) (1)

Given
training samples $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
Parameter
- $k \in \mathbb{N}$

Training
no
Classification
for each new feature vector $\underline{x}$,
- calculate the Euclidean distance of $\underline{x}$ to all training samples
  $d_n = D_{\text{Eucl}}(\underline{x}, \underline{x}_n) \quad (1 \le n \le N)$
- determine the k nearest neighbors of $\underline{x}$
- choose the most common class among the k nearest neighbors

Figure 13: *k-Nearest Neighbor algorithm.*

4.3 Bayes-Plug-In

In practice are $p(\underline{x} \mid \omega_j), P(\omega_j)$ unknown. The idea is to make it parametric through a parameter $\underline{\theta}_j \to p(\underline{x} \mid \omega_j; \underline{\theta}_j)$.
e.g. $(\underline{x} \mid \omega_j) \sim \mathbf{N}(\underline{\mu}_l, \mathbf{C}_j)$ and $\underline{\theta}_j = \{\underline{\mu}_j, \mathbf{C}_j\}$

ML estimation for Gaußian distribution:
Let $\underline{x}_n \in \mathbb{R}^d \quad (1 \le n \le N)$ i.i.d. $\mathbf{N}(\underline{\mu}, \mathbf{C})$. The probability density function is:
$p(x_n \mid \underline{\theta}) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}|^{1/2}} \exp\left(-\frac{1}{2}(\underline{x}_n - \underline{\mu})^T \mathbf{C}^{-1} (\underline{x}_n - \underline{\mu})\right)$ then for all vectors N, due to i.i.d.:

$$p(\underline{x}_1, ..., \underline{x}_N; \underline{\theta}) \overset{i.i.d}{=} \prod_{n=1}^{N} p(\underline{x}_n, \underline{\theta})$$

and using the log-likelihood, we have:

$$L(\underline{\theta}) = ln\ p(\underline{x}_1, ..., \underline{x}_N; \underline{\theta}) = \sum_{n=1}^{N} ln\ p(\underline{x}_n; \underline{\theta})$$
$$= const - \frac{N}{2} ln \mid \mathbf{C} \mid -\frac{1}{2} \sum_{n=1}^{N} \left(-\frac{1}{2}(\underline{x}_n - \underline{\mu})^T \mathbf{C}^{-1} (\underline{x}_n - \underline{\mu})\right)$$

and we can use gradient descent:

$$\nabla L(\underline{\theta}) = \underline{0} \tag{1}$$

and as solutions for the mean and covariance matrix $\hat{\underline{\mu}}$ and $\hat{\mathbf{C}}$:

$$\hat{\underline{\mu}} = \frac{1}{N_j} \sum_{n=1}^{N_j} x_n$$
$$\hat{\mathbf{C}} = \frac{1}{N_j} \sum_{n=1}^{N_j} (\underline{x}_n - \hat{\underline{\mu}}_j)(\underline{x}_n - \hat{\underline{\mu}}_j)^T$$

Table 4.3: Gaussian classifier

Given
training samples $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
Parameter
no
Training
We assume a Gaussian likelihood $N(\underline{\mu}_j, \mathbf{C}_j)$ for each class ω_j.
- estimate the class means by
  $\hat{\underline{\mu}}_j = \frac{1}{N_j} \sum_{y_n=\omega_j} \underline{x}_n$
- estimate the class covariance matrices by
  $\hat{\mathbf{C}}_j = \frac{1}{N_j} \sum_{y_n=\omega_j} (\underline{x}_n - \hat{\underline{\mu}}_j)(\underline{x}_n - \hat{\underline{\mu}}_j)^T$
- estimate the priors by
  $\hat{P}(\omega_j) = \frac{N_j}{N} \qquad (1 \le j \le c)$

Classification
use $N(\hat{\underline{\mu}}_j, \hat{\mathbf{C}}_j)$ and $\hat{P}(\omega_j)$ in the Bayesian decision rules (MBR, MAP, ML etc.) from ch. 2

Comments
- multiclass classifier
- computational complexity
  – training: $O(Nd^2)$
  – classification: depending on the decision rule

Figure 14: *Gaußian classifier alg.*

Problem is that it has to be guaranteed that the covariance matrix is invertible, so we can specify the "naive Gaussian classifier"as just the diagonal elements, making the matrix invertible at all feasible points.

4.4 Gaußian Mixture Model

Is a Mixture of Gaußian PDFs. Here we introduce the **Unimodal probability Function.** This means using GMM for one class with m Gaußian Modes (sub-classes).

$$p_m(\underline{x}; \underline{\theta}) \sim \mathbf{N}(\underline{\mu}, \mathbf{C}_m)$$
$$\underline{\theta}\{\underline{\mu}, \mathbf{C}_m\},\ 1 \le m \le M$$

Introducing a new random mode variable $Z \in \{1, 2, ..., M\}$ as the label for each mode in a PDF.
We can use the law of total probability, adding the new random mode variable:

$$p(\underline{x}, \underline{\theta}) = \sum_{m=1}^{M} \underbrace{P(z=m)}_{\alpha_m} p(\underline{x} \mid z=m) = \sum_{m=1}^{M} \alpha_m p_m(\underline{x}; \underline{\theta}_m)$$

also

$$\alpha_m = P(z=m) \ge 0,\ \sum_{m=1}^{M} \alpha_m = 1$$

as prior of z, **mode weight** with $m-1$ parameters
$\underline{\theta} = \{\underline{\theta}_1, ..., \underline{\theta}_m; \alpha_1, ..., \alpha_{m-1}\}$ contains $M\frac{d(d+3)}{2} + m - 1$ parameters if covariance matrix non diagonal,
$2Md + M - 1$ if the covariance matrix diagonal.

Approach A:

Assuming $d = 1$ and $M = 2$, i.e.,

$$p(x; \underline{\vartheta}) = \alpha_1 p_1(x) + \alpha_2 p_2(x), \qquad \alpha_1 = \alpha,\ \alpha_2 = 1 - \alpha,$$
$$p_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2}(x - \mu_i)^2\right), \qquad i = 1, 2$$

with $\underline{\vartheta} = [\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \alpha]^T$. Given N i.i.d. measurements $x_n \sim p(x; \underline{\vartheta}), 1 \le n \le N$, the log-likelihood is

$$L(\underline{\vartheta}) = \ln \prod_{n=1}^{N} p(x_n; \underline{\vartheta}) = \sum_{n=1}^{N} \ln p(x_n\ \underline{\vartheta}).$$

The gradient vector ∇L of $L(\underline{\vartheta})$ contains the following elements:

$$\frac{\partial L}{\partial \mu_i} = \sum_{n=1}^{N} \frac{\alpha_i p_i(x_n)}{p(x_n; \underline{\vartheta})} \cdot \frac{x_n - \mu_i}{\sigma_i^2}, \qquad i = 1, 2$$
$$\frac{\partial L}{\partial \sigma_i^2} = \sum_{n=1}^{N} \frac{\alpha_i p_i(x_n)}{p(x_n; \underline{\vartheta})} \cdot \frac{1}{2\sigma_i^2}\left(\frac{(x_n - \mu_i)^2}{\sigma_i^2} - 1\right), \qquad i = 1, 2$$
$$\frac{\partial L}{\partial \alpha} = \sum_{n=1}^{N} \frac{p_1(x_n) - p_2(x_n)}{p(x_n; \underline{\vartheta})}.$$

Obviously, the nonlinear equation system $\nabla L = \underline{0}$ has no closed-form solution.

Figure 15: *this approach fails often*

Approach B: A probability model of N i.i.d. $\underline{x}_n \in \mathbb{R}^d$ and $z_n, 1 \le n \le N$. With z_n as additional mode/sub-class measurement. Let

$$\underline{x} = \begin{bmatrix} \underline{x}_1 \\ \vdots \\ \underline{x}_N \end{bmatrix}\ Nd \times 1,\ \underline{z} = \begin{bmatrix} z1 \\ \vdots \\ z_N \end{bmatrix}\ N \times 1$$

and we build a joint probe mode of $\underline{x}$ and $\underline{z}$:

$$p(\underline{x}, \underline{z}; \underline{\theta}) \overset{i.i.d.}{=} p(\underline{x}_n, \underline{z}_n; \underline{\theta})$$
$$\Rightarrow \max_{\underline{\theta}} p(\underline{x}_n, \underline{z}_n; \underline{\theta}),\ \text{as GMM}$$

But for this, we have to know what $\underline{x}$ and $\underline{z}$ exactly are. Therefore this only works if the mode labels are known!

EM algorithm (1)

The **Expectation-Maximization** (**EM**) algorithm from Dempster (1977) is a popular iterative method for finding ML parameter estimates when we have missing (bad) data.

Let $p(\underline{x}, \underline{z}; \underline{\vartheta})$ be the likelihood where $\underline{\vartheta}$ is the unknown parameter to be estimated, $\underline{x}$ is the observed (good) data, and $\underline{z}$ is the missing (bad) data. If we assume to know both $\underline{x}$ *and* $\underline{z}$, we could easily find the ML estimate of $\underline{\vartheta}$ by $\max_{\underline{\vartheta}} p(\underline{x}, \underline{z}; \underline{\vartheta})$. But $\underline{z}$ is unknown. Can we still estimate $\underline{\vartheta}$ from $\underline{x}$? Yes, it is possible by using the EM algorithm. It is an iterative method which starts with an initial guess $\hat{\underline{\vartheta}}(0)$ about $\underline{\vartheta}$ and produces a sequence of improved parameter estimates $\hat{\underline{\vartheta}}(1), \hat{\underline{\vartheta}}(2), \ldots$.

Let $L(\underline{x}, \underline{z}; \underline{\vartheta}) = \ln p(\underline{x}, \underline{z}; \underline{\vartheta})$ be the log-likelihood. At each iteration $k = 1, 2, \ldots$, the EM algorithm consists of two steps. The first **expectation step (E-step)** computes the expectation of the log-likelihood over the missing data $\underline{z}$ conditioned on the current parameter estimate $\hat{\underline{\vartheta}}(k)$ in order to "remove" $\underline{z}$.

EM algorithm (2)

For this purpose, we first calculate the marginal PDF of $\underline{z}$ conditioned on $\underline{x}$ and $\hat{\underline{\vartheta}}(k)$[1]

$$p(\underline{z} | \underline{x}, \hat{\underline{\vartheta}}(k)) = \frac{p(\underline{x}, \underline{z}; \hat{\underline{\vartheta}}(k))}{p(\underline{x}; \hat{\underline{\vartheta}}(k))} = \frac{p(\underline{x}, \underline{z}; \hat{\underline{\vartheta}}(k))}{\int p(\underline{x}, \underline{z}; \hat{\underline{\vartheta}}(k)) \mathrm{d}\underline{z}}$$

Then an expectation of $L(\underline{x}, \underline{z}; \underline{\vartheta})$ over $\underline{z}|\underline{x}, \hat{\underline{\vartheta}}(k)$ returns an averaged log-likelihood

$$Q(\underline{\vartheta}) = \mathrm{E}_{\underline{z}|\underline{x}, \hat{\underline{\vartheta}}(k)} L(\underline{x}, \underline{z}; \underline{\vartheta}) = \int \left(\ln p(\underline{x}, \underline{z}; \underline{\vartheta})\right) p(\underline{z}|\underline{x}, \hat{\underline{\vartheta}}(k)) \mathrm{d}\underline{z}$$

which only depends on $\underline{\vartheta}$.
The second step, the **maximization step (M-step)**, maximizes $Q(\underline{\vartheta})$ over $\underline{\vartheta}$ and returns an improved parameter estimate

$$\hat{\underline{\vartheta}}(k+1) = \arg\max_{\underline{\vartheta}} Q(\underline{\vartheta})$$

The iteration stops if a given number of iterations is reached or if $\|\hat{\underline{\vartheta}}(k+1) - \hat{\underline{\vartheta}}(k)\|$ is below a certain threshold.

Table 4.5: GMM classifier (1)

Given
training samples $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
Parameter
model order M_j for all classes ω_j $(1 \le j \le c)$
Training
Estimate for each class ω_j the GMM parameters from the N_j training samples of class ω_j by using the EM algorithm from Table 4.4.
Classification
use the estimated GMM models in the Bayesian decision rules (MBR, MAP, ML etc.) from ch. 2

Figure 16: *Modes as missing data*

4.5 Parzen Window Method (Kernel Function)

to estimate $p(\underline{x})$ with $\hat{p(\underline{x})}$

Given
training samples $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
Parameter
- a suitable kernel function $\phi(\underline{x})$ with $\int \phi(\underline{x}) \mathrm{d}\underline{x} = 1$
- bandwidth $h > 0$ with $\phi_h(\underline{x}) = \frac{1}{h^d}\phi(\underline{x}/h)$

Training
no or
estimate the kernel function $\phi_h(\underline{x})$ or the bandwidth h
ML classification
for each new feature vector $\underline{x}$,
- estimate the likelihood $p(\underline{x}|\omega_j)$ for each class ω_j by using the Parzen window method
  $\hat{p}(\underline{x}|\omega_j) = \frac{1}{N_j} \sum_{y_n=\omega_j} \phi_h(\underline{x} - \underline{x}_n) \quad (1 \le j \le c)$
- choose the class with the largest likelihood
  $\hat{y}(\underline{x}) = \arg\max_{\omega_j} \hat{p}(\underline{x}|\omega_j)$

Comments
- multiclass classifier
- computational complexity
  – classification: N kernel calculations
–) high classification complexity
–) difficult choice of h
–) needs a huge amount of training data for a large dimension d

Figure 17: *Parzen Window alg.*

4.6 Discriminant Functions

The main idea is to create c discriminant functions $f_j(\underline{x})$ for c classes ω_j. For each $\underline{x}$: $\hat{y}(\underline{x}) = \underset{\omega_j}{arg\ max} f_j(\underline{x})$.

Training	Bayes Plug-In	Discriminant
need models for	$p(\underline{x} \mid \omega_j; \underline{\theta})$	$f_j(\underline{x}; \underline{\theta})$
estimate from	$\underline{\theta}_j, P(\omega_j)$	$\underline{\theta}_j$
derive $f_j(\underline{x})$ from	$p(\underline{x}_j, \omega_j; \hat{\underline{\theta}})$	-

Tabela 1: *Table of Prof. Yang to describe this section comparing the older sections.*

Advantage of discriminant functions is that we do not need the $\underline{x} \mid \omega_j$. We can define our space $\underline{x} \in \mathbb{R}^d$ and our decision Region $R_i = \{\underline{x} \mid f_i(\underline{x}) > f_j(\underline{x}), i \ne j\}$ and our decision boundary $\beta_{ij} = \{\underline{x} \mid f_i(\underline{x}) = f_i(\underline{x}) > f_k(\underline{x})\}$ where the decision Regions are define to be convex. (For the convex definition, see lecture.)

4.7 Linear (affine) discriminant functions

We define our discriminant function with the learnable parameters $\underline{w}_j$ and w_{j0} as $f_j(\underline{x}) = \underline{w}_j^T \underline{x} + w_{j0}$ for the class ω_j with $\underline{w}_j$ as the weight vector and w_{j0} as the offset.
Example for two classes with classification capibility:

$$f(\underline{x}) = f_1(\underline{x}) - f_2(\underline{x}) = (\underline{w}_1 - \underline{w}_2)^T \underline{x} + (w_{10} - w_{20})$$
$$= \underline{w}^T \underline{x} + w_0 \underset{\omega_2}{\overset{\omega_1}{\lessgtr}} 0$$

and the decision boundary $f(\underline{x}) = 0$ as an hyperplane. The vector $\underline{w}$ is the normal vector to the hyperplane.

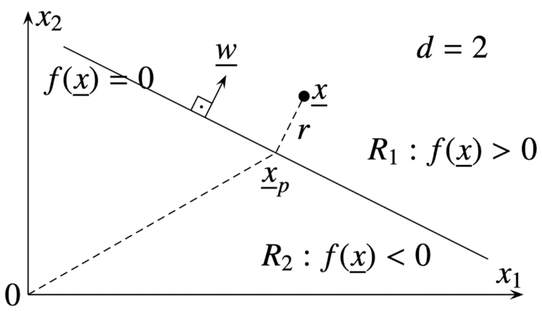


Figure 18: *Hyperplane and the two decision regions.*

However this approach with a linear affine function as discriminant function only works well for linearly separable data and is also not feasible in most cases because of the $N(c-1)$ equalities and infinite many solutions for linearly separable data. For non linearly separable data, there is no solution. To find a solution for the set S of infinite many solutions with the linear discriminant function: Training by optimization.

$$\underline{\theta} = \{(\underline{w}_j, w_{j0}) \text{ from } S\}$$
$$\hat{\underline{\theta}_j} = \underset{\underline{\theta}}{arg\ min}\ J(\underline{\theta})$$

with the cost function defined as $J(\underline{\theta})$.
For the cost function there are different options.
a) l_2**-norm**:
For instance for two classes. Use for the classes ω_1 and ω_2 a numerical value to represent them:

$$y_n = \begin{cases} 1 \text{ for } \omega_1 \\ -1 \text{ for } \omega_2 \end{cases}$$

and choose the discriminant function such that $f(\underline{x}_n) = \underline{w}^T \underline{x} + w_0 \approx y_n$.
With this definition, create an Error vector $\underline{e}_n = f(\underline{x}_n) - y_n,\ \underline{e} = [e_1, ..., e_N]^T$ so we can then use the cost function with the norm:

$$J(\underline{\theta}) = || \underline{e} ||_2^2 = \sum_{n=1}^{N} (\underline{w}^T \underline{x} + w_0 - y_n)^2$$
$$\underset{\underline{w}, w_0}{min}\ J(\underline{\theta}) \text{ as Least Squares.}$$

b) l_1**-norm**:

$$J(\underline{\theta}) = \sum_{n=1}^{N} \mid \underline{w}^T \underline{x} + w_0 - y_n \mid$$

c) Maximum Margin: see Support vector machine.
d) Cross entropy: see Deep Learning.

4.8 Feature Mapping

Is an approach to make non linearl separable data linear and non convex regions convex. See the examples on the lecture.
It is powerful but, d features grow and become unmanageable. Huge computational complexity. Might lead to overfitting.

4.9 Support vector machine

5. 5

Referências

[1] J. Flusser and T. Suk. Pattern recognition by affine moment invariants. *Pattern Recognition*, 26:167–174, 1993.

[2] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, 1962.

[3] P. Maragos. Pattern spectrum and multiscale shape representation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 11:701–715, 1989.

[4] A. Meijster and M. H. F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Trans. Patt. Anal. Mach. Intell.*, 24(4):484–494, 2002.

[5] P. F. M. Nacken. *Image Analysis Methods Based on Hierarchies of Graphs and Multi-Scale Mathematical Morphology*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1994.