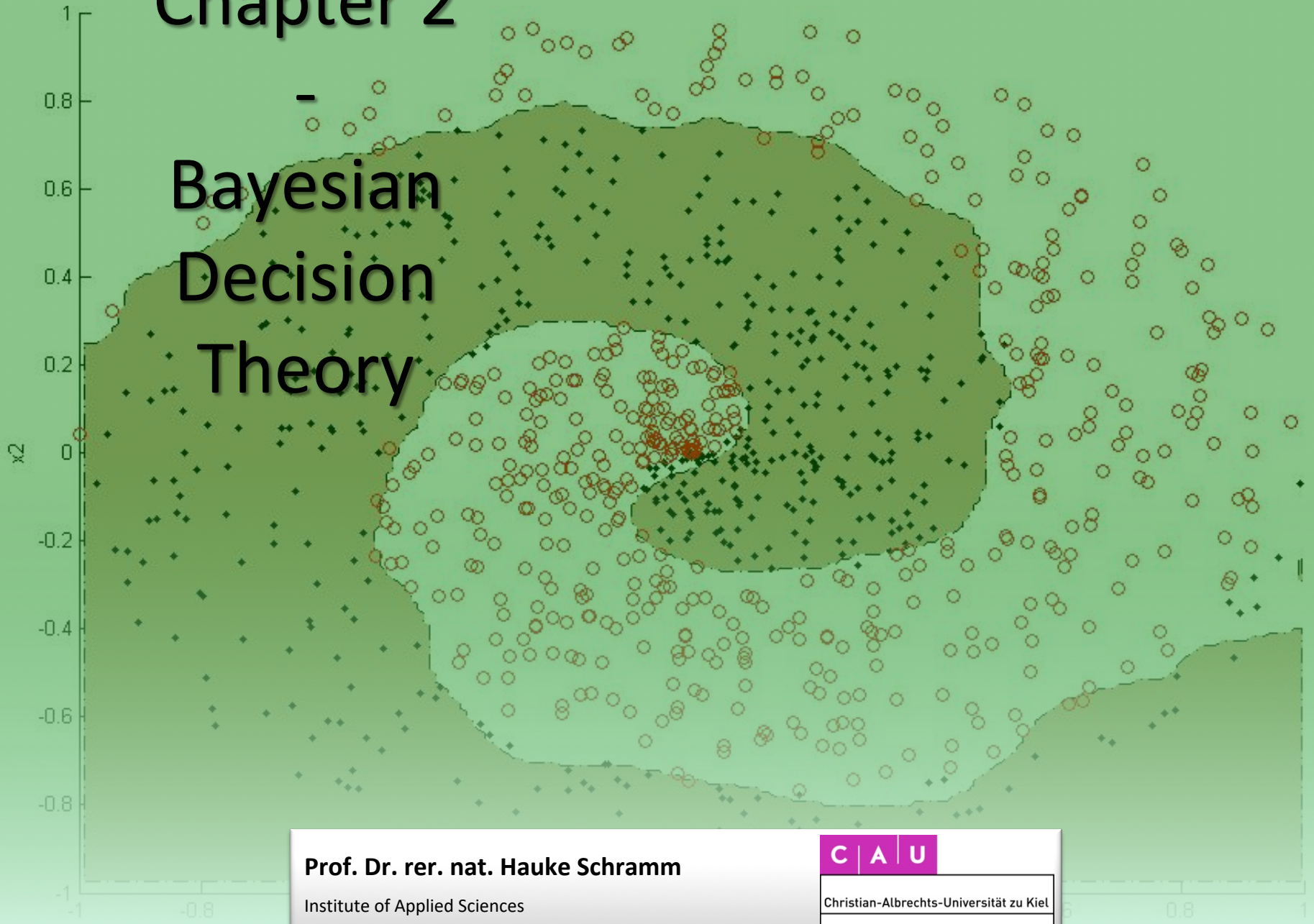
Chapter 2
-
Bayesian Decision Theory



Prof. Dr. rer. nat. Hauke Schramm

Institute of Applied Sciences

University of Applied Sciences Kiel

C A U

Christian-Albrechts-Universität zu Kiel

Institut für Informatik

Topics of the lecture

Introduction

1. Fundamentals of probability theory

   Probability distribution, expectation, covariance matrix, marginal, multivariate Gaussian, ...

2. Bayesian decision theory

   Bayes theorem, prior-/a-priori-/a-posteriori-probabilities, Bayes risk, decision boundaries, ...

3. Maximum-likelihood parameter estimation

   Theory and practical applications

4. Non-parametric techniques

   Density estimation, Parzen windows, nearest neighbor classification

Outline of Chapter 2

2. Bayesian decision theory
    1. Introduction
    2. Generalization and formalization
    3. Discriminant functions
    4. Decision boundaries
    5. Minimum distance classifier
    6. Application example

2. Bayesian Decision Theory

2.1 Introduction

Let us again consider the simple fish classification example:

- Two classes ω_1 = 'salmon' and ω_2 = 'sea bass' are considered
- Both classes have the same prior probability $P(\omega_1) = P(\omega_2)$
- No other types of fish are relevant → $P(\omega_1) + P(\omega_2) = 1$

2. Bayesian Decision Theory

2.1 Introduction

Assumption:

We are forced to make a decision without being allowed to take a look

Optimal decision rule?

→ Decide $\begin{cases} \omega_1 & \text{if } P(\omega_1) > P(\omega_2) \\ \omega_2 & \text{if } P(\omega_2) \geq P(\omega_1) \end{cases}$

→ Always make the same decision

→ Expected error rate = the smaller of $P(\omega_1)$ and $P(\omega_2)$

2. Bayesian Decision Theory

2.1 Introduction

Normally, we are not asked to make a decision with so little information

→ Measurements (size, lightness, ...) can be used to improve the classifier

→ Combined into a ***feature*** vector **x**

→ Distribution of these random variables for a class ω is expressed in

2. Bayesian Decision Theory

2.1 Introduction

Normally, we are not asked to make a decision with so little information

→ Measurements (size, lightness, ...) can be used to improve the classifier

→ Combined into a ***feature*** vector **x**

→ Distribution of these random variables for a class ω is expressed in

Class-conditional probability distribution $p(\mathbf{x} \mid \omega)$

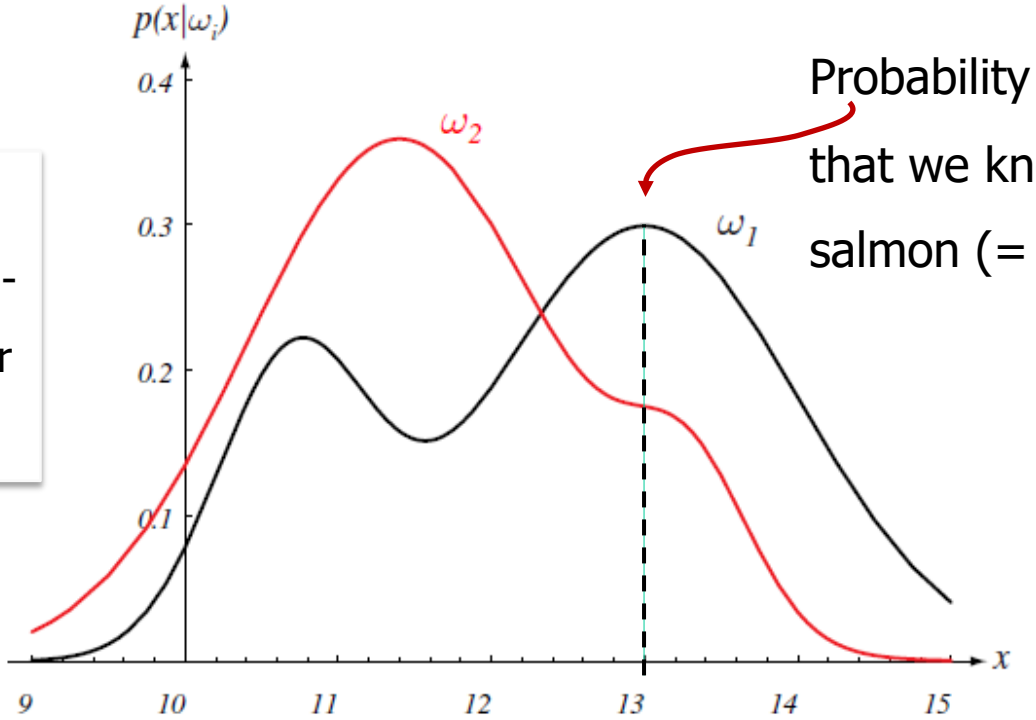
2. Bayesian Decision Theory

2.1 Introduction

Example:

Hypothetical class-conditional probability density function of a single feature x:

Note: Since density functions are normalized, the area under each curve is 1.

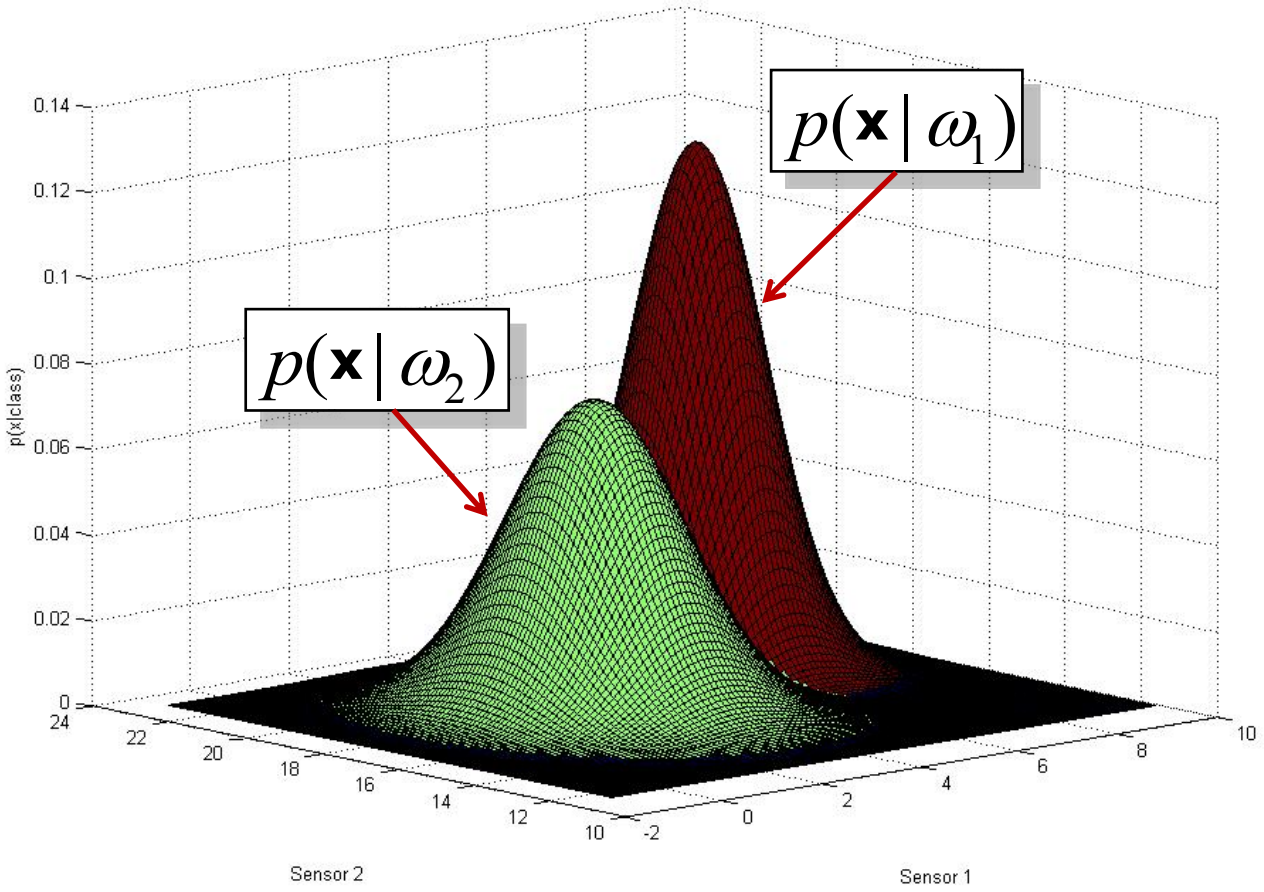


Probability of x = 13 given that we know the fish is a salmon (= ω_1).

2.1 Introduction

Example:

Hypothetical class-conditional probability density function of a **2D feature x**:



2.1 Introduction

Back to 1-D example

Assumption:

We know the prior probability $P(\omega_j)$ and class-conditional probability $p(x|\omega_j)$

for both classes j = 1, 2.

Optimal decision rule if we observe x for an unknown fish?

Reasonable: Decide for the class with highest **posterior probability** $P(\omega_j|x)$

Interpretation / Application :

- Unknown object (fish) is observed → get the measurement x
- Decide for the class with the highest probability when knowing x.

2.1 Introduction

Assumption:

We know the prior probability $P(\omega_j)$ and class-conditional probability $p(x|\omega_j)$

for both classes j = 1, 2.

How do we get the posterior probability $P(\omega_j|x)$?

→ Bayes Rule

2.1 Introduction

Assumption:

We know the prior probability $P(\omega_j)$ and class-conditional probability $p(x|\omega_j)$ for both classes j = 1, 2.

How do we get the posterior probability $P(\omega_j|x)$?

→ Bayes Rule

$$P(y \mid x) = \frac{P(x|y) \cdot P(y)}{\sum_{y \in Y} P(x \mid y) \cdot P(y)}$$

"Inverts" statistical connections

2.1 Introduction

How do we get the posterior probability $P(\omega_j \mid x)$?

Derivation of posterior probability from prior and class-conditional probability

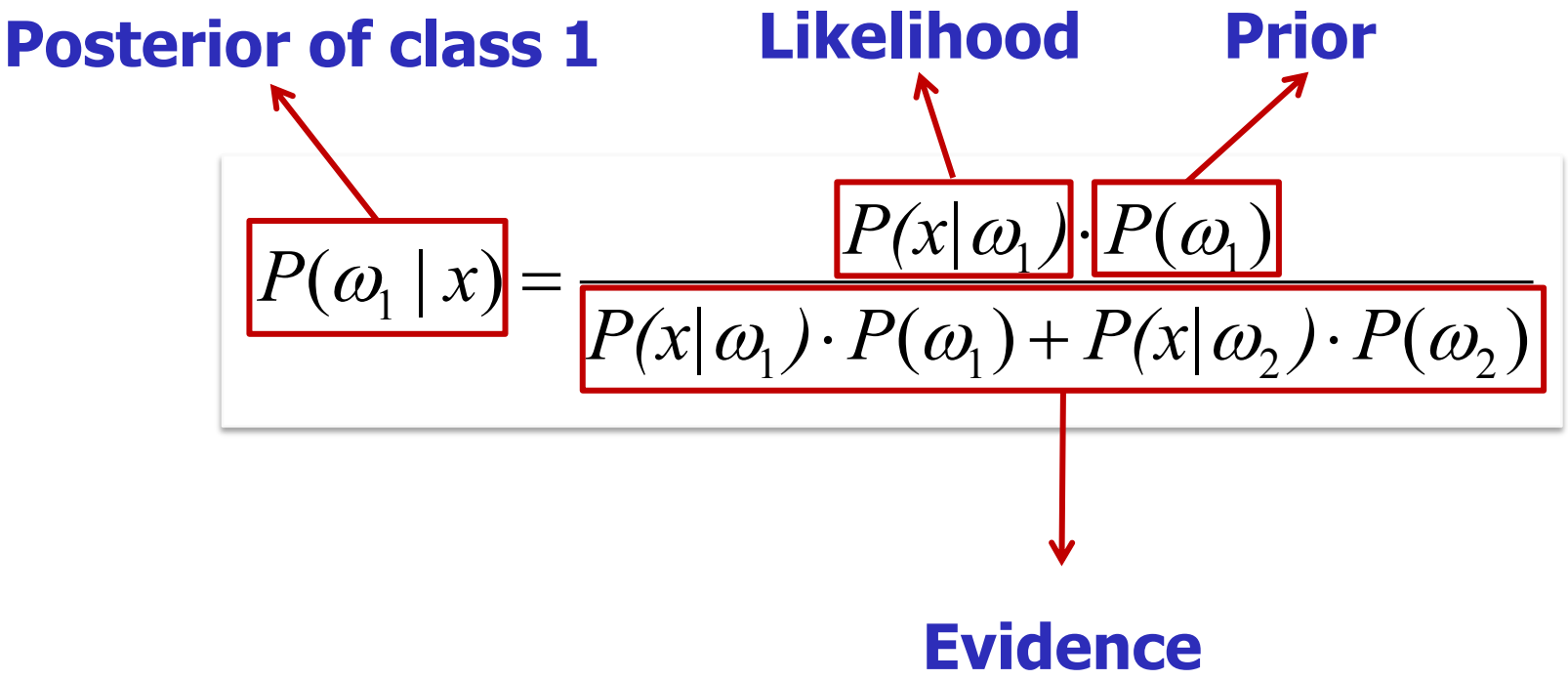
$$P(\omega_1 \mid x) = \frac{P(x \mid \omega_1) \cdot P(\omega_1)}{\sum_{\omega_j \in [\omega_1, \omega_2]} P(x \mid \omega_j) \cdot P(\omega_j)}$$

$$= \frac{P(x \mid \omega_1) \cdot P(\omega_1)}{P(x \mid \omega_1) \cdot P(\omega_1) + P(x \mid \omega_2) \cdot P(\omega_2)}$$

2.1 Introduction

How do we get the posterior probability $P(\omega_j \mid x)$?

Derivation of posterior probability from prior and class-conditional probability



2.1 Introduction

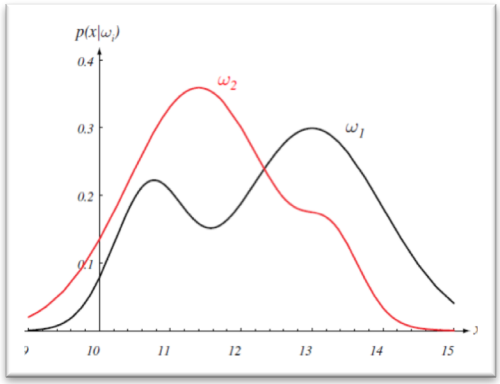
How do we get the posterior probability $P(\omega_j \mid x)$?

Derivation of posterior probability from prior and class-conditional probability

Posterior of class 2

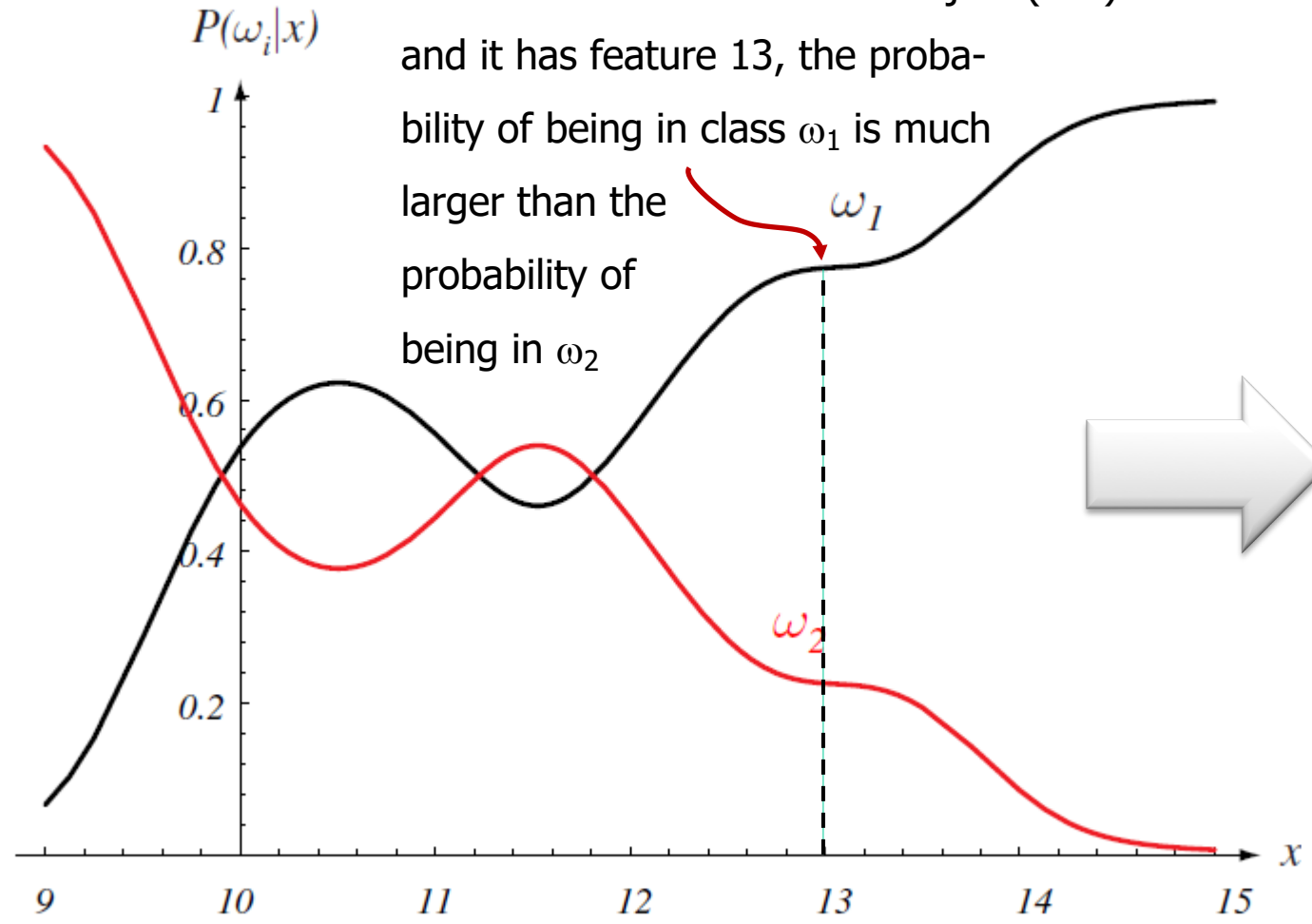
$$P(\omega_2 \mid x) = \frac{P(x \mid \omega_2) \cdot P(\omega_2)}{P(x \mid \omega_1) \cdot P(\omega_1) + P(x \mid \omega_2) \cdot P(\omega_2)}$$

2.1 Introduction



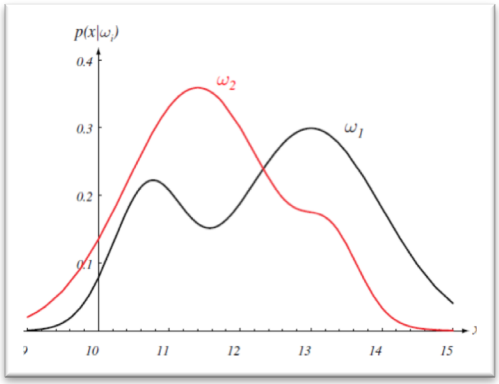
corresponding class-conditionals

If we measure an unknown object (fish) and it has feature 13, the probability of being in class ω_1 is much larger than the probability of being in ω_2

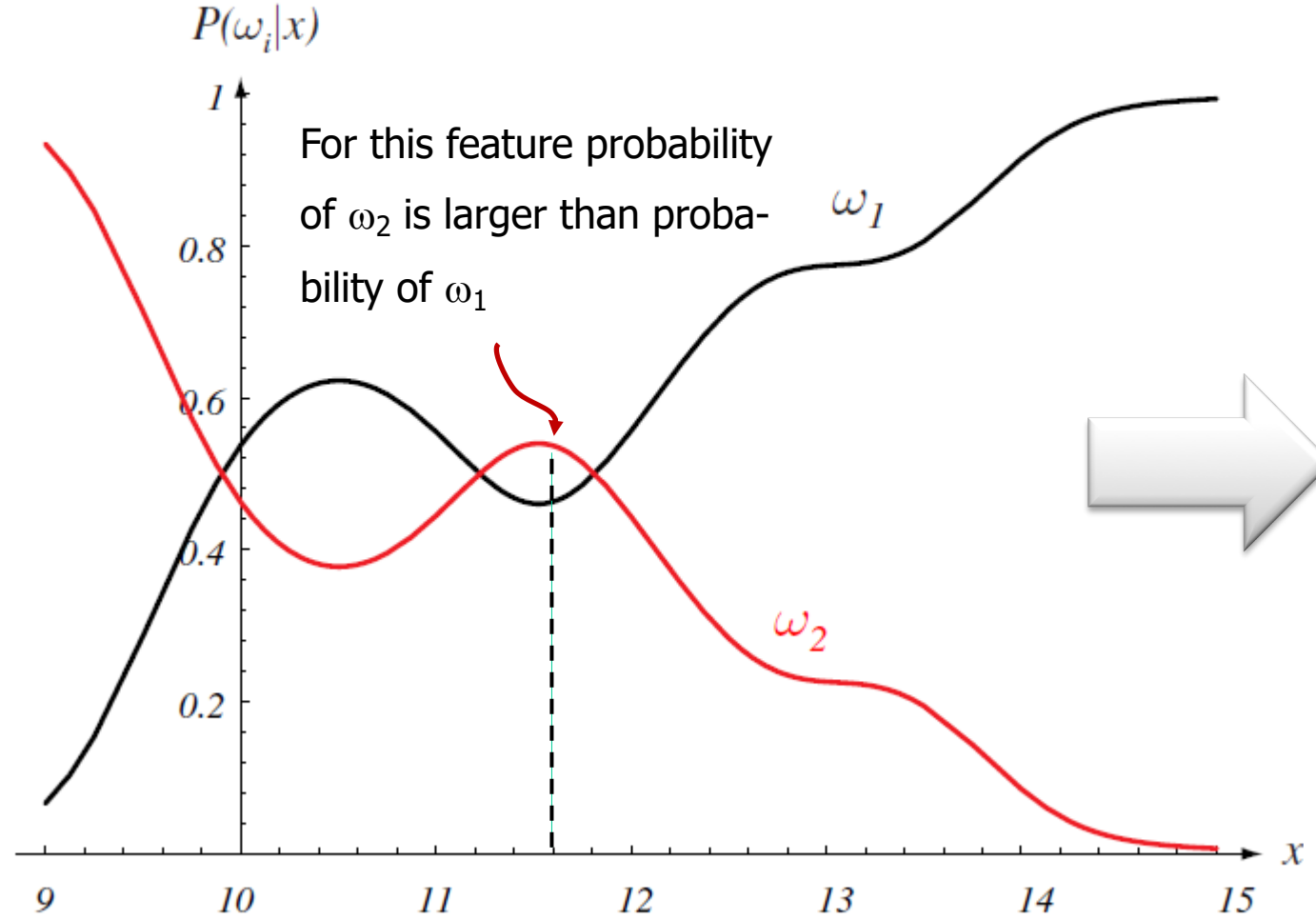


Decision for class 1

2.1 Introduction



corresponding class-conditionals



Decision for class 2

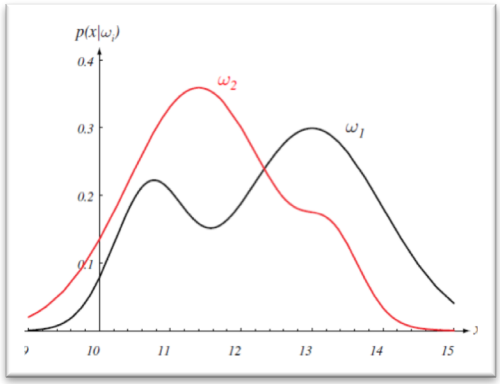
2.1 Introduction

Bayes Decision Rule for Two-Category Classification

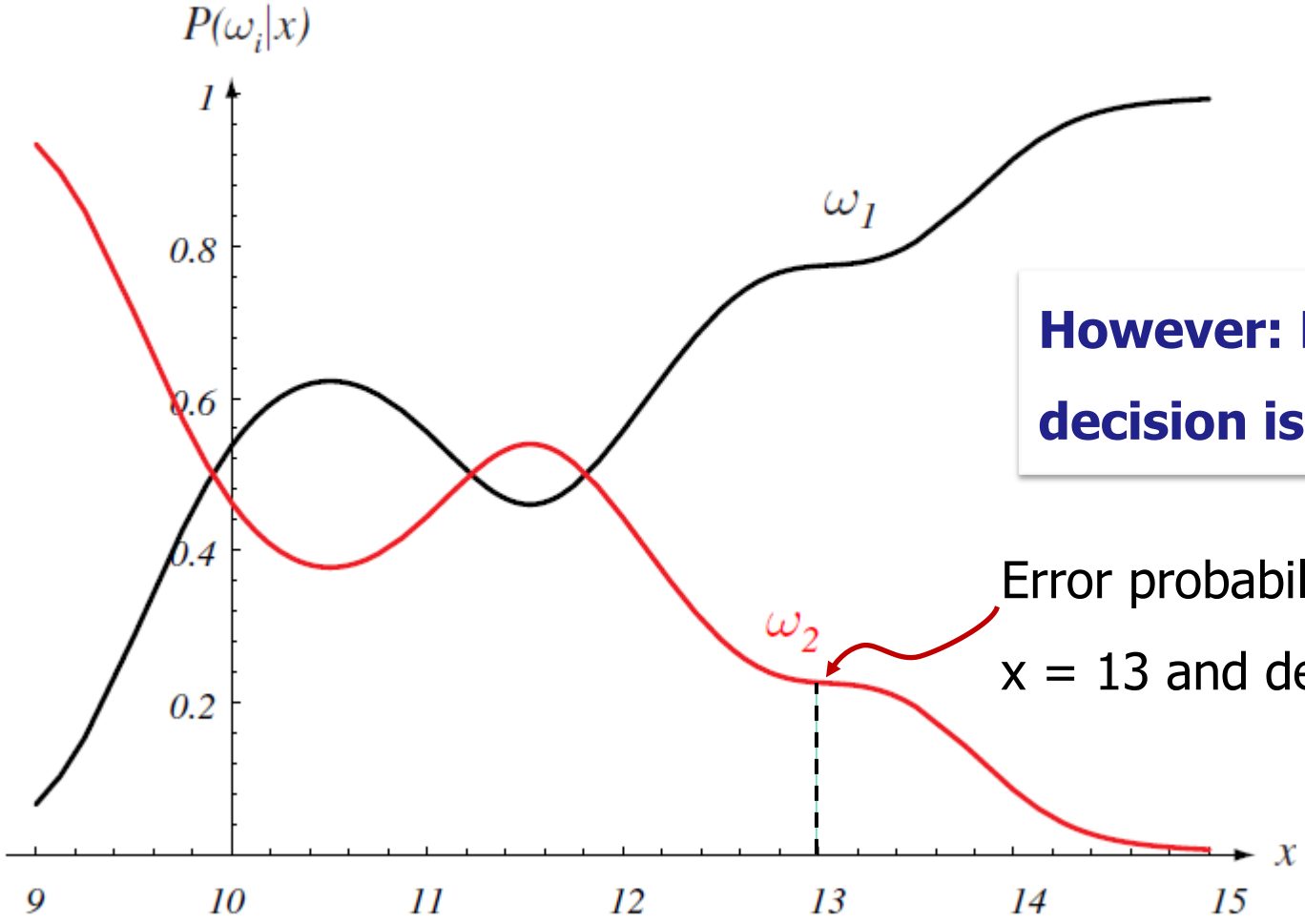
If $P(\omega_1 \mid x) > P(\omega_2 \mid x)$ → Decide ω_1

If $P(\omega_1 \mid x) \leq P(\omega_2 \mid x)$ → Decide ω_2

2.1 Introduction



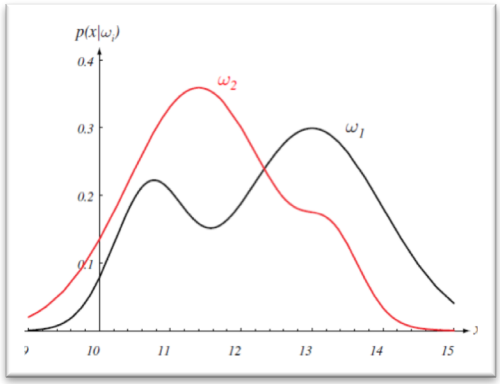
corresponding class-conditionals



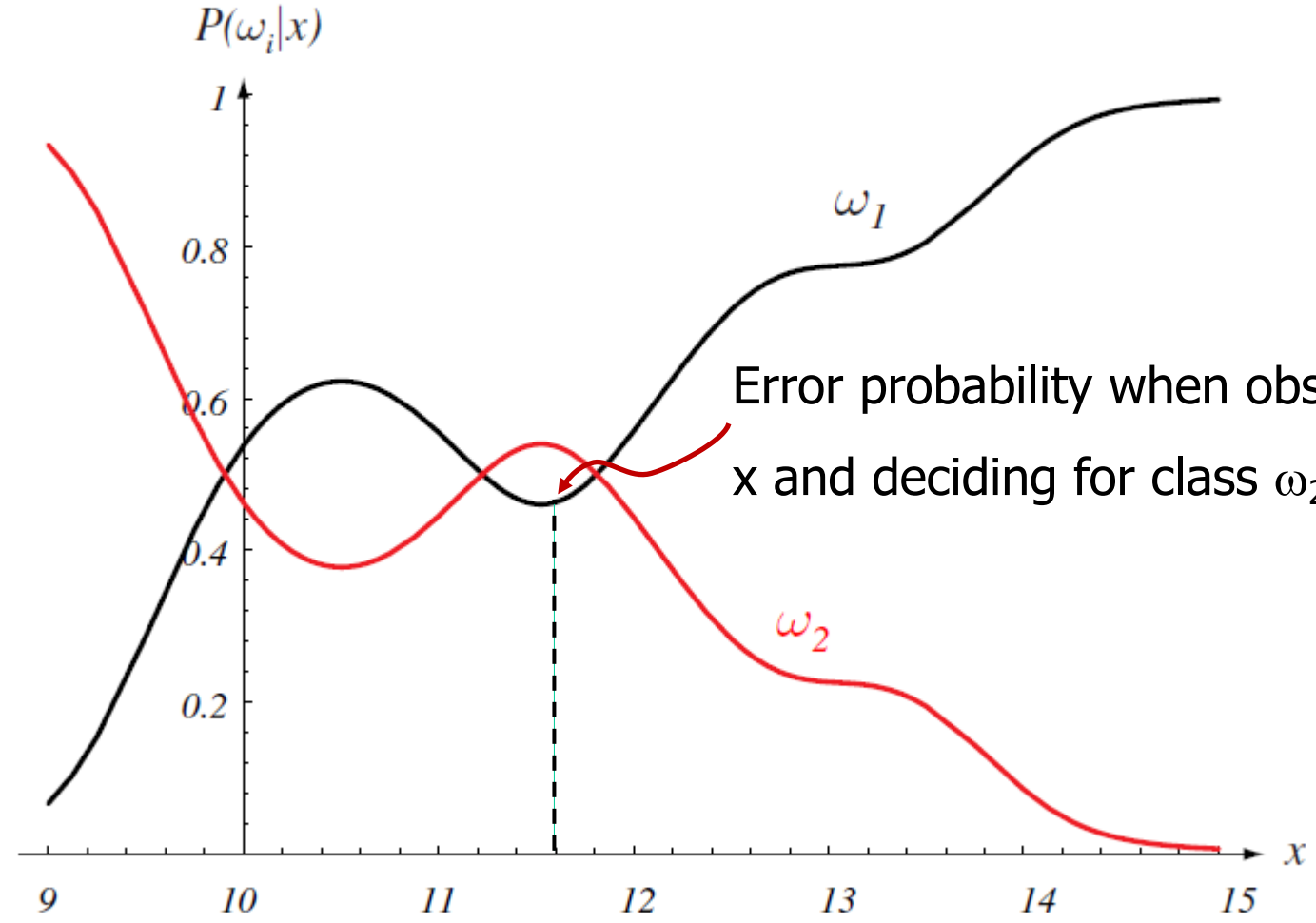
However: Note, that this decision is not perfect!

Error probability when observing x = 13 and deciding for class ω_1

2.1 Introduction



corresponding class-conditionals

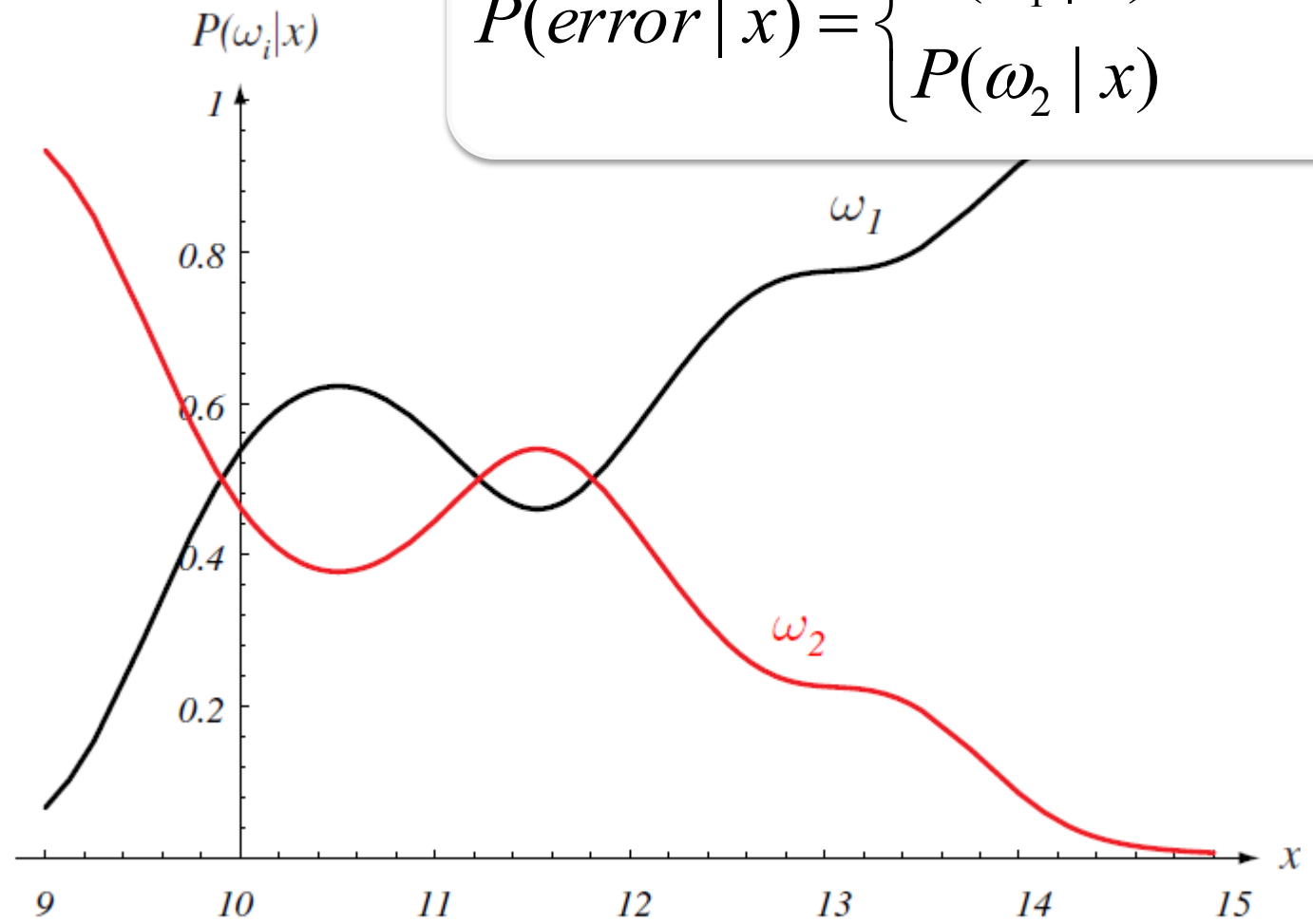


Error probability when observing x and deciding for class ω_2

2.1 Introduction

Error probability for a single decision

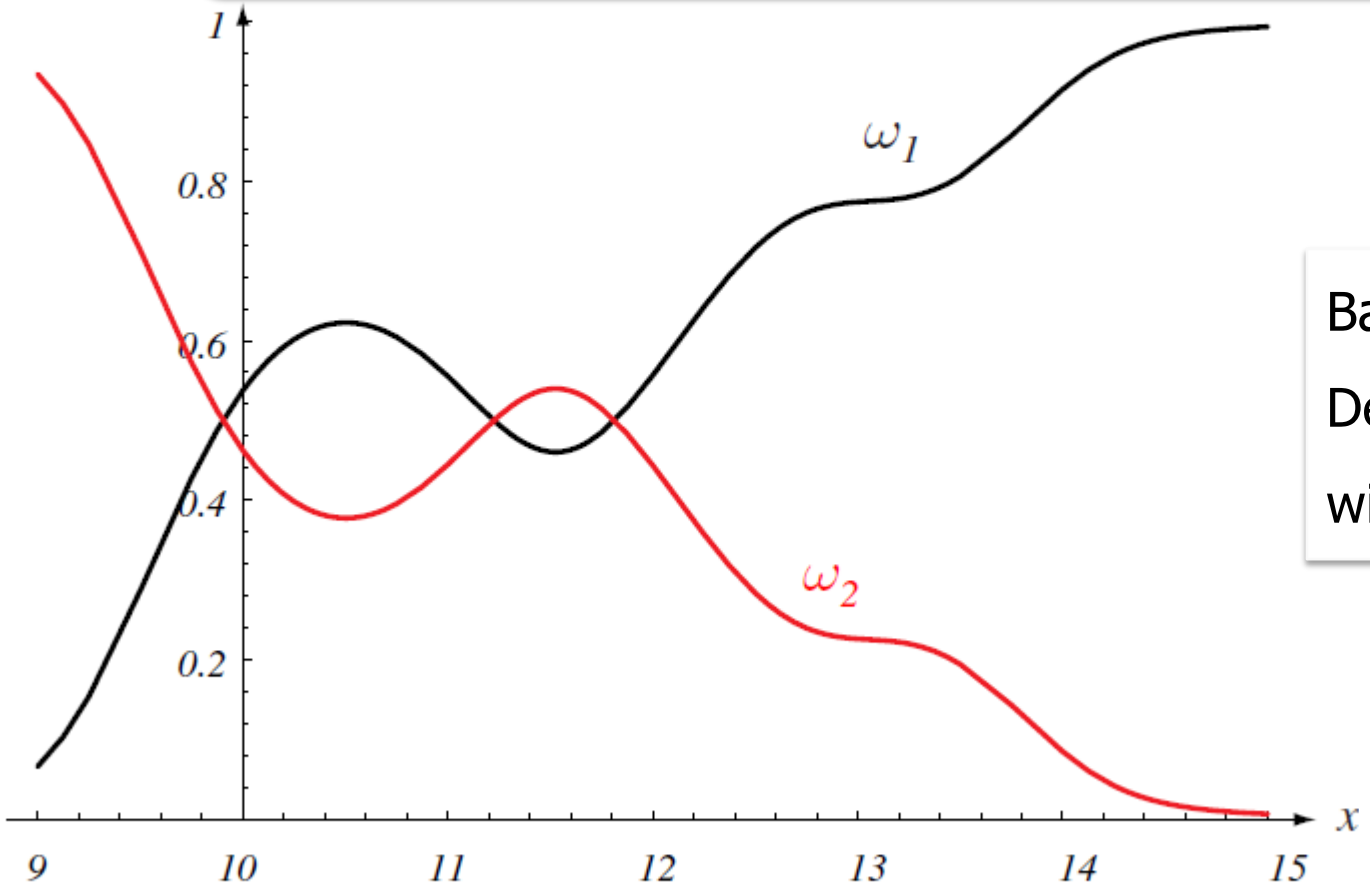
$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) & \text{if we decide } \omega_2 \\ P(\omega_2 \mid x) & \text{if we decide } \omega_1 \end{cases}$$



2.1 Introduction

Mean error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$



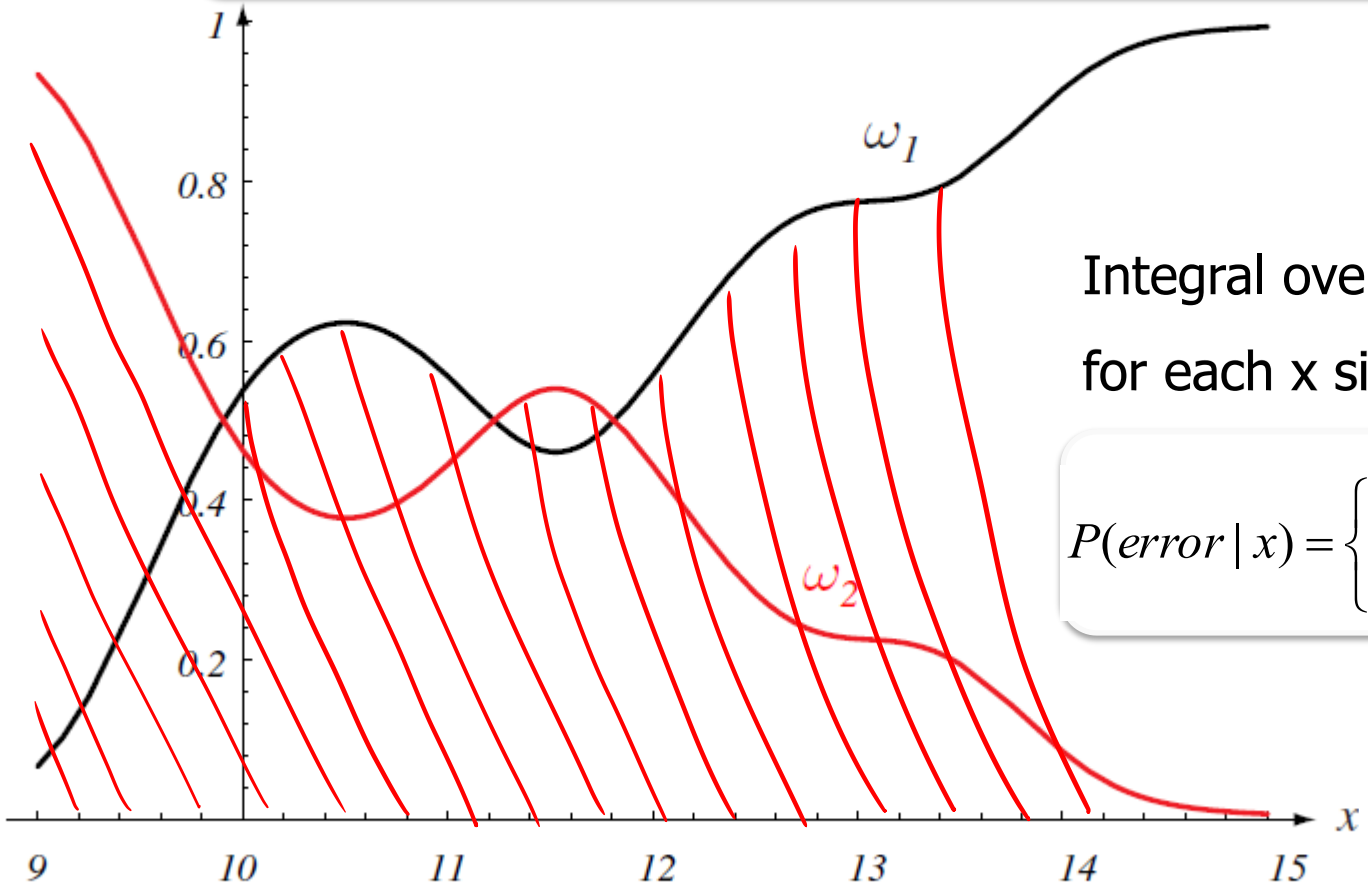
Bad decision rule:

Decide for class ω_j with smaller $P(\omega_j|x)$

2.1 Introduction

Mean error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\,p(x)dx$$



Integral over the larger $P(\omega_j|x)$ for each x since:

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) & \text{if decision } \omega_2 \\ P(\omega_2 \mid x) & \text{if decision } \omega_1 \end{cases}$$

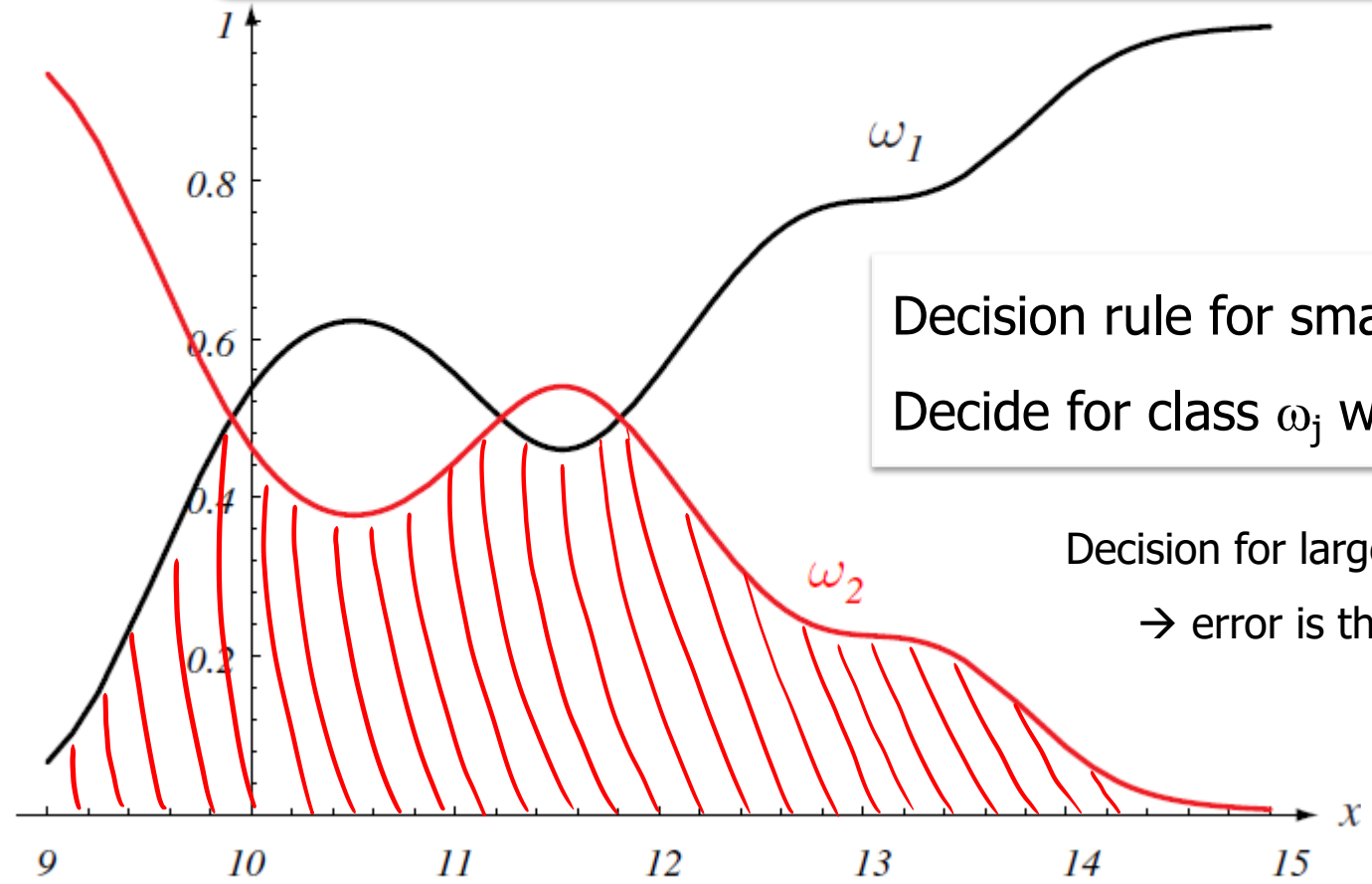
Decision for smaller prob.

→ error is the larger prob.

2.1 Introduction

Mean error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$



Decision rule for smallest possible error:

Decide for class ω_j with largest $P(\omega_j|x)$

Decision for larger probability

→ error is the smaller probability

2. Bayesian Decision Theory

2.2 Generalization and formalization

Classification: Assignment of a vector **x** to class $\omega \in \{\omega_1, ..., \omega_K\}$

Assignment is given by a **Decision Rule** r(**x**):

$r(\mathbf{x}) : \mathbf{x} \rightarrow \omega$

$r(\mathbf{x}) = \omega_k$

Goal: Find an **optimal Decision Rule**

(Note: Lower case k, i.e. just any of the classes in $\{\omega_1, ..., \omega_K\}$)

*"The decision rule r(x) assigns a vector **x** to the class ω_k."*

2.2 Generalization and formalization

Let ω_i be the correct class

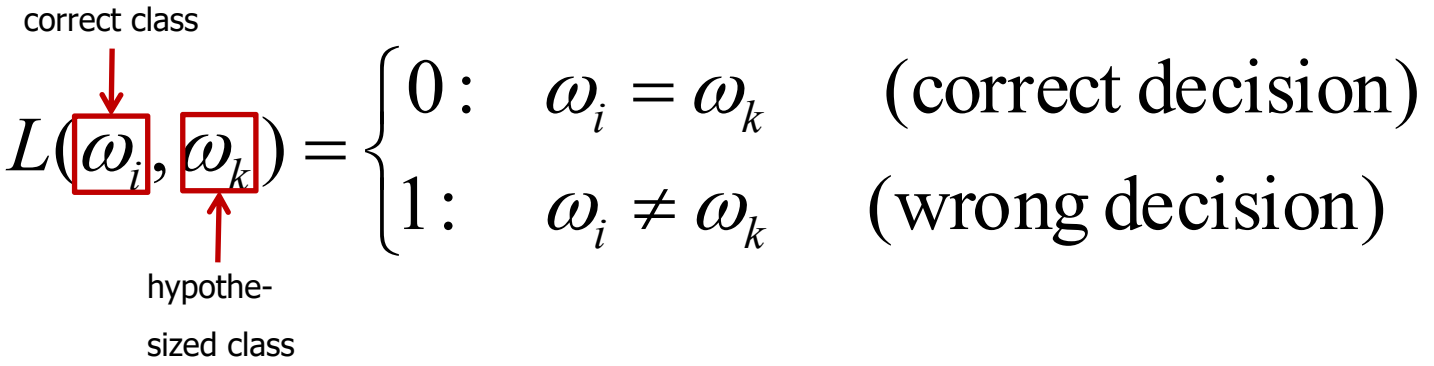
→ Each decision r(**x**) = ω_k of our classifier is associated a particular cost:

2.2 Generalization and formalization

Let ω_i be the correct class

→ Each decision r(**x**) = ω_k of our classifier is associated a particular cost:

Cost function:



Correct decision of the classifier → cost 0

Wrong decision of the classifier → cost 1

2.2 Generalization and formalization

Let ω_i be the correct class

→ Each decision r(**x**) = ω_k of our classifier is associated a particular cost:

Cost function:

$$L(\omega_i, \omega_k) = \begin{cases} 0: & \omega_i = \omega_k \quad \text{(correct decision)} \\ 1: & \omega_i \neq \omega_k \quad \text{(wrong decision)} \end{cases}$$

Which costs are expected for a given decision rule r(**x**) = ω_k and fixed **x**?

In decision theoretic terminology the expected cost is called a **risk**.

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k in case of a fixed **x**?

Assumption: We know the correct class for the given fixed **x** is ω_i

→ Expected cost is $L(\omega_i, r(x))$

Problem: We do not know the correct class for **x**!

Solution:

Consider correct class as a **random variable** and compute expected value of costs over all possible values of this random variable.

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k in case of a fixed **x**?

Review: Expected value of a function over a discrete random variable ω

$$E\{g(\omega)\} = \sum_{\omega_k \in \Omega} g(\omega_k) \cdot P(\omega_k)$$

discrete since class labels are discrete

Cost or risk function

All possible classes

Probability that ω_k is correct

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k in case of a fixed **x**?

Review: Expected value of a function over a discrete random variable ω

$$E\{g(\omega)\} = \sum_{\omega_k \in \Omega} g(\omega_k) \cdot P(\omega_k)$$

Insert cost function

$$E\{L(\omega, r(x))\} = \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

Note: Only this term is variable since this is what we search for.
Everything else is fixed!

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k in case of a fixed **x**?

$$E\{L(\omega, r(x))\} = \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

New name: **Risk function** (depends on classifier r(x))

2.2 Generalization and formalization

let us now consider all possible x

What is the **risk** of a decision r(**x**) = ω_k in case of a fixed **x**?

$$E\{L(\omega, r(x))\} = \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

New name: **Risk function** (depends on classifier r(x))

$$R(r(x) \mid x) = \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

What would be the cost in case of ω_j being the correct class?

What is the probability of ω_j being the correct class for given x?

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k for all possible values of **x**?

Given: Risk function for a given fixed **x** → R(r(**x**) | **x**)

Required: Risk function for all possible values of **x**

Solution:

Similar to method above

→ Consider **x** as random variable and compute expected value

→ However, this time random variable **x** is *continuous*

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k for all possible values of **x**?

Expected function value of a continuous random variable

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

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k for all possible values of **x**?

Expected function value of a continuous random variable

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

Expected value of risk function $R(r(x) \mid x)$

$$E\{R(r(x) \mid x)\} = \int_{-\infty}^{\infty} [R(r(x) \mid x) \cdot p(x)]\, dx = \mathrm{R}$$

insert function from above

2.2 Generalization and formalization

What is the **risk** of a decision r(**x**) = ω_k for all possible values of **x**?

$$E\{R(r(x)\mid x)\} = \int_{-\infty}^{\infty} [R(r(x)\mid x)\cdot p(x)]\,dx = \mathrm{R}$$

$$\mathrm{R} = \int_{-\infty}^{\infty} \left[\sum_{j=1}^{K} [L(\omega_j, r(x))\cdot P(\omega_j \mid x)]\right]\cdot p(x)\,dx$$

2.2 Generalization and formalization

Overall risk when using decision rule r(**x**)

$$R = \int_{-\infty}^{\infty} \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right] \cdot p(x)\, dx$$

An **optimal classifier** (or: optimal decision rule) minimizes this risk.

→ Choice of r(**x**)?

2.2 Generalization and formalization

Overall risk when using decision rule r(**x**)

$$R = \int_{-\infty}^{\infty} \left[\sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right] \right] \cdot p(x)\, dx$$

Find a decision rule r(**x**) which minimizes this term for each possible **x.**

→ This leads to a minimal R since p(**x**) is not influenced by the choice of r(**x**).

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

optimal decision rule

all possible decision rules

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

Cost function:

$$L(\underbrace{\omega_i}_{\text{correct class}}, \omega_k) = \begin{cases} 0: & \omega_i = \omega_k \quad \text{(correct decision)} \\ 1: & \omega_i \neq \omega_k \quad \text{(wrong decision)} \end{cases}$$

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \sum_{j=1}^{K} \left[L(\omega_j, r(x)) \cdot P(\omega_j \mid x) \right]$$

All terms are 1 but $L(\omega_k, r(x)=\omega_k)$

$$r_{opt}(x) = \arg\min_{r(x)} \left[L(\omega_1, r(x)) \cdot P(\omega_1 \mid x) + \ldots \right.$$

$$\left. \ldots + L(\omega_K, r(x)) \cdot P(\omega_K \mid x) \right]$$

Cost function:

$$L(\omega_i, \omega_k) = \begin{cases} 0: & \omega_i = \omega_k \quad \text{(correct decision)} \\ 1: & \omega_i \neq \omega_k \quad \text{(wrong decision)} \end{cases}$$

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \Big[1 \cdot P(\omega_1 \mid x) + \ldots + 1 \cdot P(\omega_{k-1} \mid x) + 0 \cdot P(r(x) = \omega_k \mid x) + 1 \cdot P(\omega_{k+1} \mid x) + \ldots + 1 \cdot P(\omega_K \mid x) \Big]$$

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \Big[1 \cdot P(\omega_1 \mid x) + \ldots + 1 \cdot P(\omega_{k-1} \mid x) + 0 \cdot P(r(x) = \omega_k \mid x) + 1 \cdot P(\omega_{k+1} \mid x) + \ldots + 1 \cdot P(\omega_K \mid x) \Big]$$

$$r_{opt}(x) = \arg\min_{r(x)} \sum_{j \neq k}^{K} P(\omega_j \mid x)$$

using $\sum_{j=1}^{K} P(\omega_j \mid x) = 1$

leads to

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \Big[1 \cdot P(\omega_1 \mid x) + \ldots + 1 \cdot P(\omega_{k-1} \mid x) + 0 \cdot P(r(x) = \omega_k \mid x) + 1 \cdot P(\omega_{k+1} \mid x) + \ldots + 1 \cdot P(\omega_K \mid x) \Big]$$

$$r_{opt}(x) = \arg\min_{r(x)} \sum_{j \neq k}^{K} P(\omega_j \mid x)$$

$$r_{opt}(x) = \arg\min_{r(x)} \Big[1 - P(r(x) = \omega_k \mid x) \Big]$$

2.2 Generalization and formalization

Minimizing the overall risk with an 'optimal' decision rule r(**x**)

$$r_{opt}(x) = \arg\min_{r(x)} \left[1 - P(r(x) = \omega_k \mid x)\right]$$

Which choice of r(x) minimizes this term?

Choosing the class ω_k which has the highest posterior probability

$P(\omega_k|x)$ minimizes the given term and consequently the overall risk.

2.2 Generalization and formalization

Bayes Decision Rule

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

MAP Classifier

MAP: Maximum A-Posteriori

Practical usage of this rule:

- Given: observation (vector) **x** from an unknown object
- Compute posterior probability $P(\omega_k|\mathbf{x})$ of all possible classes
- Decide for the class with the highest posterior probability

This is the best we can do in case of having correct posterior probability distributions!

2.2 Generalization and formalization

Bayes Decision Rule

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

MAP Classifier

MAP: Maximum A-Posteriori

Interpretation:

The Bayes Decision Rule transforms the classification problem into a problem of estimating the correct posterior probability distributions.

Basis of the statistical pattern recognition approach!

2.2 Generalization and formalization

Bayes Decision Rule

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

MAP Classifier

MAP: Maximum A-Posteriori

Remaining problem: How to estimate the correct posteriors?

2. Bayesian Decision Theory

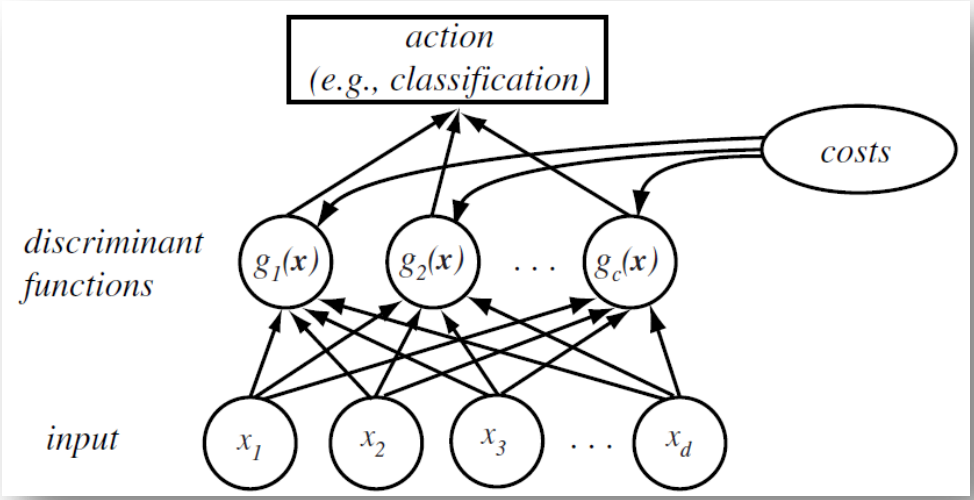
2.3 Discriminant functions

Useful and most prominent representation of a pattern classifier:

→ Set of **Discriminant Functions** $g_i(x), \quad i = 1, 2, \ldots, c$

Decision rule: Decide class ω_i if $g_i(x) > g_j(x), \quad \forall j \neq i$

Illustration of this concept:



2.3 Discriminant functions

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

Discriminant function for Bayes classifier?

$$g_i(x) = P(\omega_i|x)$$

→ Decision for the class with the **largest discriminant function**

is decision for the class with the **largest posterior probability**.

Other choices of discriminant function with identical classification result:

$$g_i(x) = p(x|\omega_i)P(\omega_i)$$

$$g_i(x) = \log p(x|\omega_i) + \log P(\omega_i)$$

2.3 Discriminant functions

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

Special case: two categories ω_1 and ω_2

→ Classifier has two discriminant functions g_1 and g_2

Let $g(x) \equiv g_1(x) - g_2(x)$

Decide ω_1 *if* $g(x) > 0$ otherwise decide ω_2

$$g(x) = P(\omega_1 \mid x) - P(\omega_2 \mid x)$$

2.3 Discriminant functions

$$r_{opt}(x) = \arg\max_{\omega_k} \left[P(\omega_k \mid x) \right]$$

Special case: two categories ω_1 and ω_2

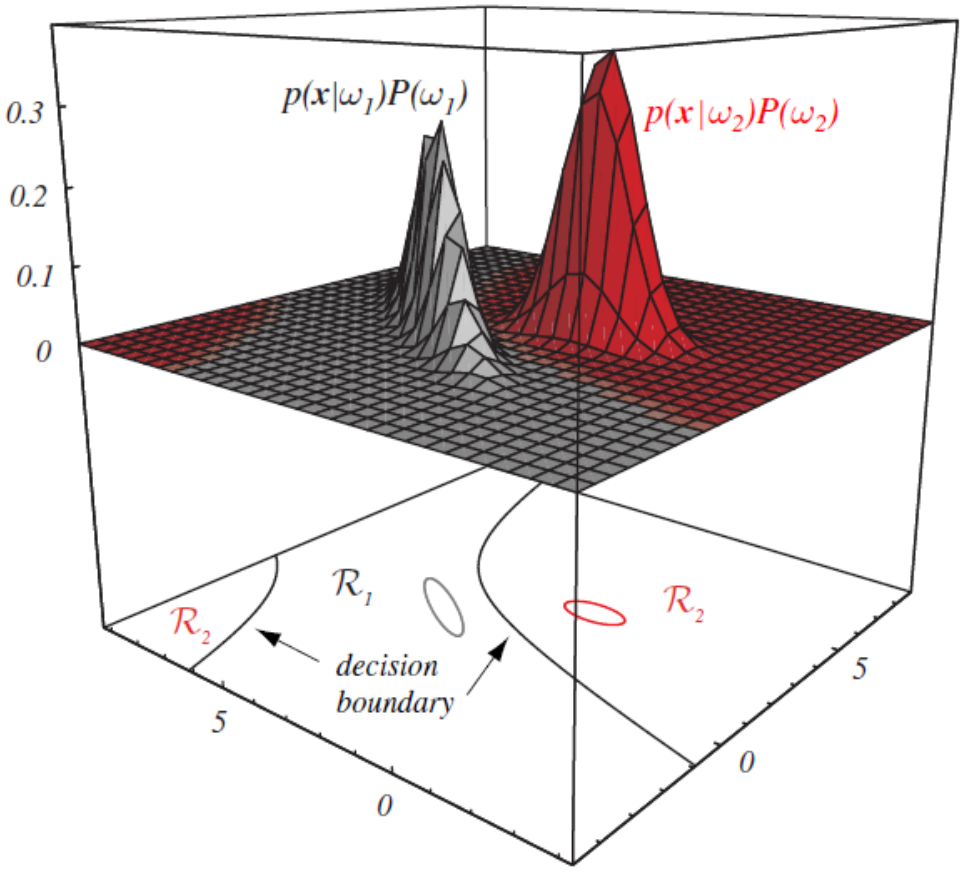
$$\begin{aligned} g(x) &= P(\omega_1 \mid x) - P(\omega_2 \mid x) \\ &= \ln P(\omega_1 \mid x) - \ln P(\omega_2 \mid x) \\ &= \ln \frac{p(x \mid \omega_1) P(\omega_1)}{p(x)} - \ln \frac{p(x \mid \omega_2) P(\omega_2)}{p(x)} \\ &= \dots \\ &= \ln \frac{P(x \mid \omega_1)}{P(x \mid \omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)} \end{aligned}$$

2. Bayesian Decision Theory

2.4 Decision boundaries

Effect of any decision rule:

Division of the feature space into **decision regions** separated by **decision boundaries** where the largest discriminant functions have the same value.



Example: Consider classes ω_k and ω_c

Decision boundary = $\left\{ \boldsymbol{x} \in \Re^D : g_k(\boldsymbol{x}) = g_c(\boldsymbol{x}) \right\}$

2.4 Decision boundaries

Example: Iris classification (see laboratory)

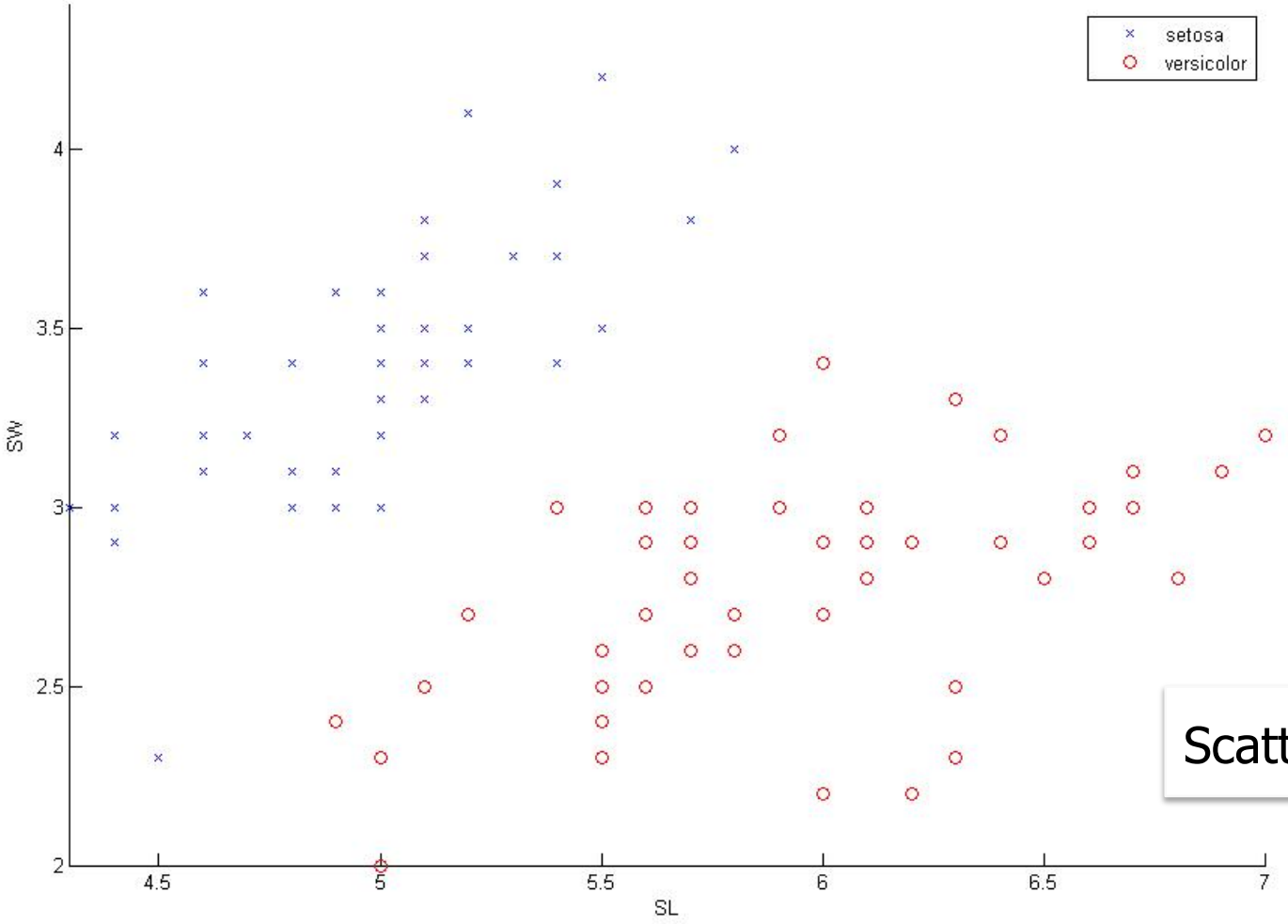
- Iris Versicolor and Iris Setosa
- 2D feature is used: Sepal length (SL) and width (SW)

Goal:

- Estimate one 2D-Gaussian (dependent components) for each class
- Visualize the decision boundary

2.4 Decision boundaries

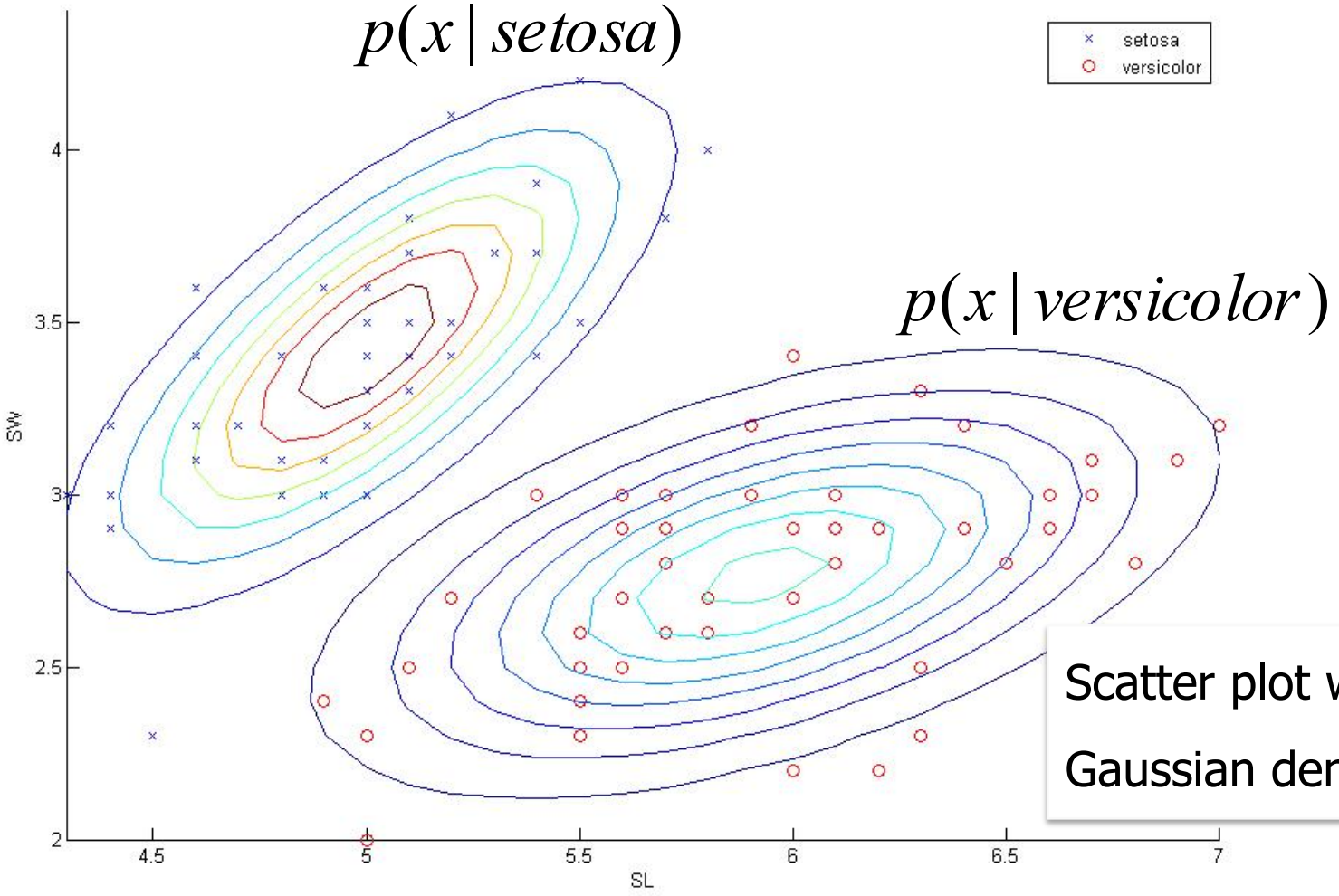
Example: Iris classification



Scatter plot

2.4 Decision boundaries

Example: Iris classification

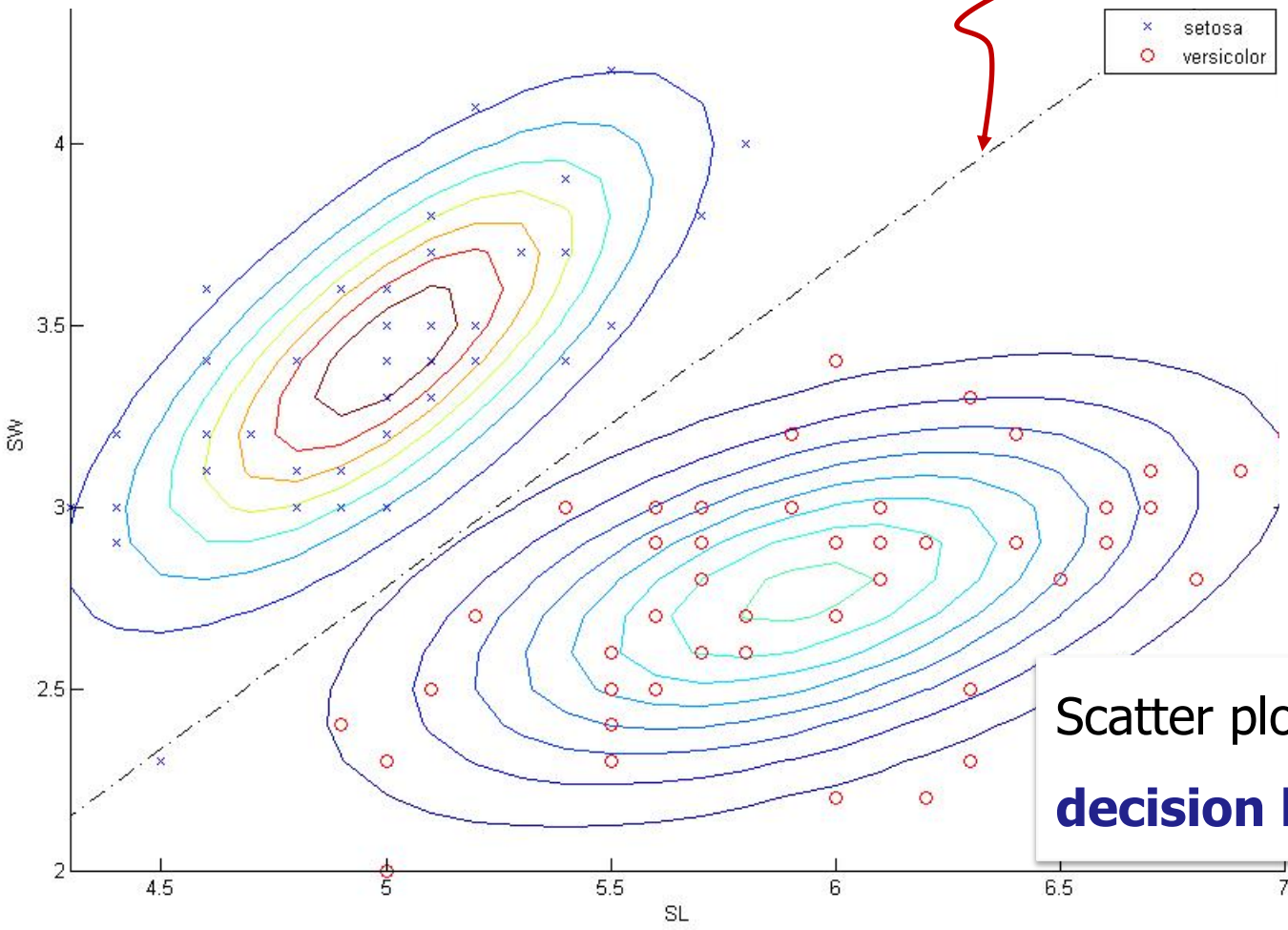


Scatter plot with Gaussian densities

2.4 Decision boundaries

Example: Iris classification

$$p(x \mid setosa) \cdot P(setosa) =$$
$$p(x \mid versicolor) \cdot P(versicolor)$$



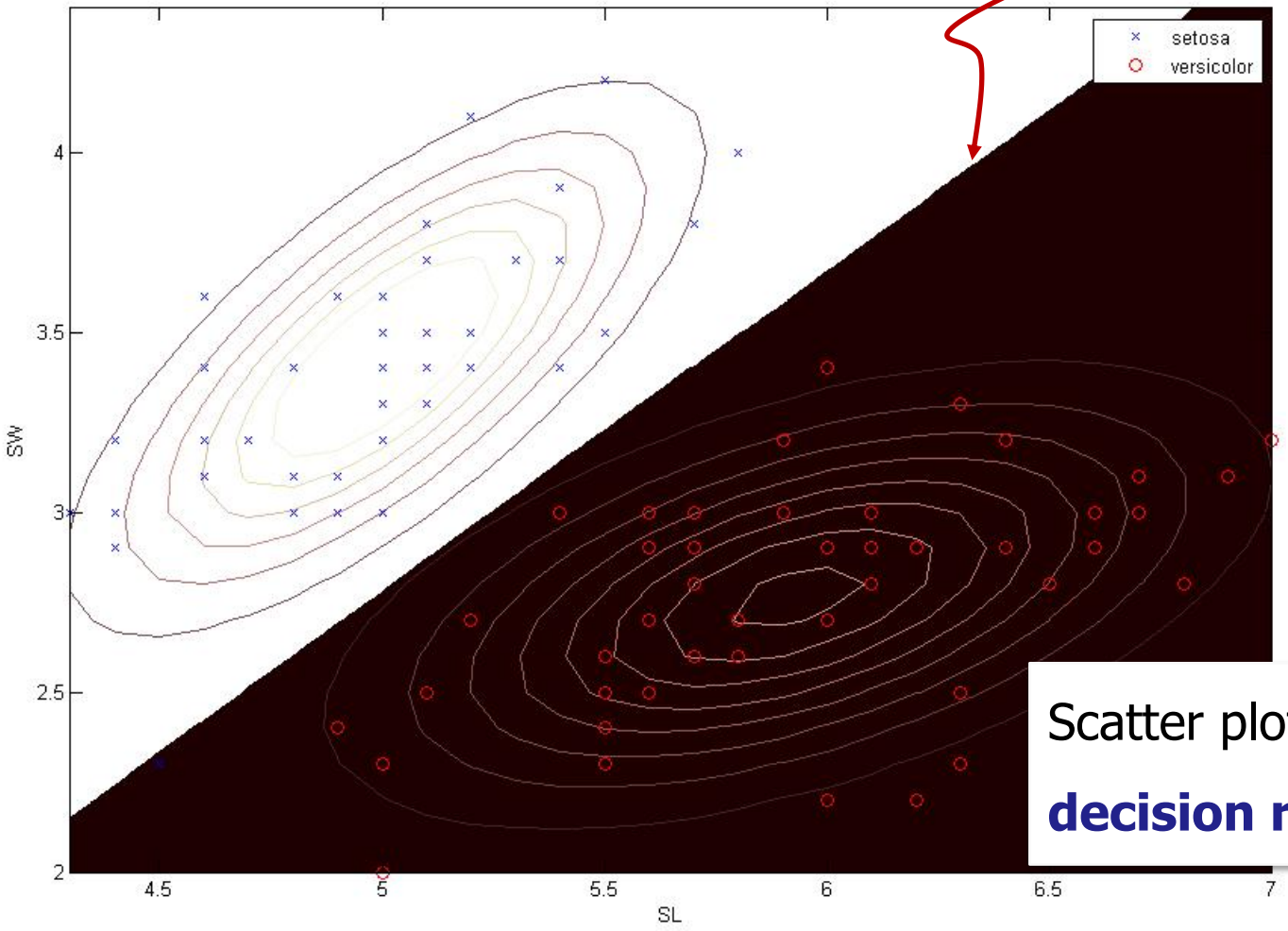
Scatter plot with

decision boundary

2.4 Decision boundaries

Example: Iris classification

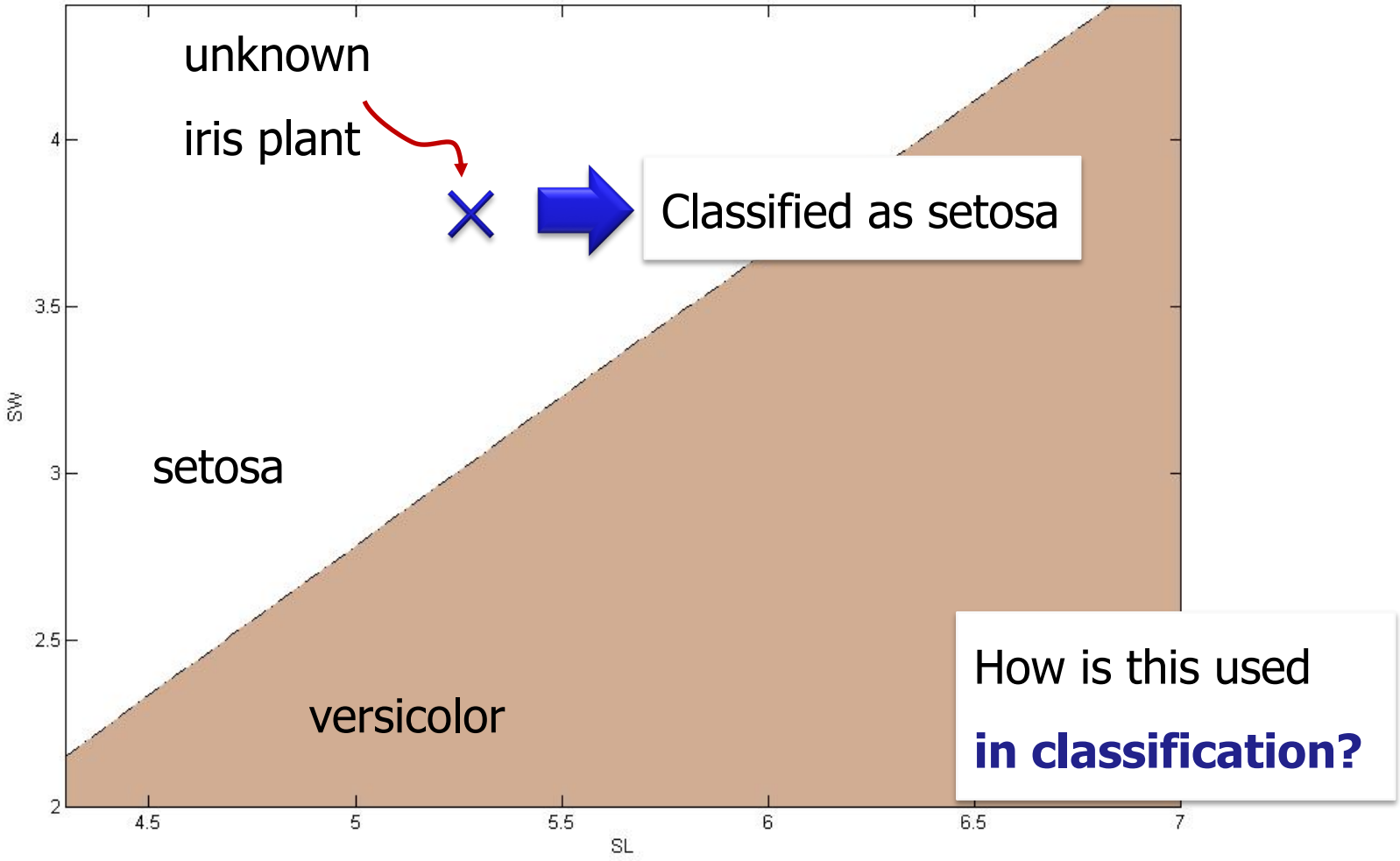
$$p(x \mid setosa) \cdot P(setosa) =$$
$$p(x \mid versicolor) \cdot P(versicolor)$$



Scatter plot with **decision regions**

2.4 Decision boundaries

Example: Iris classification



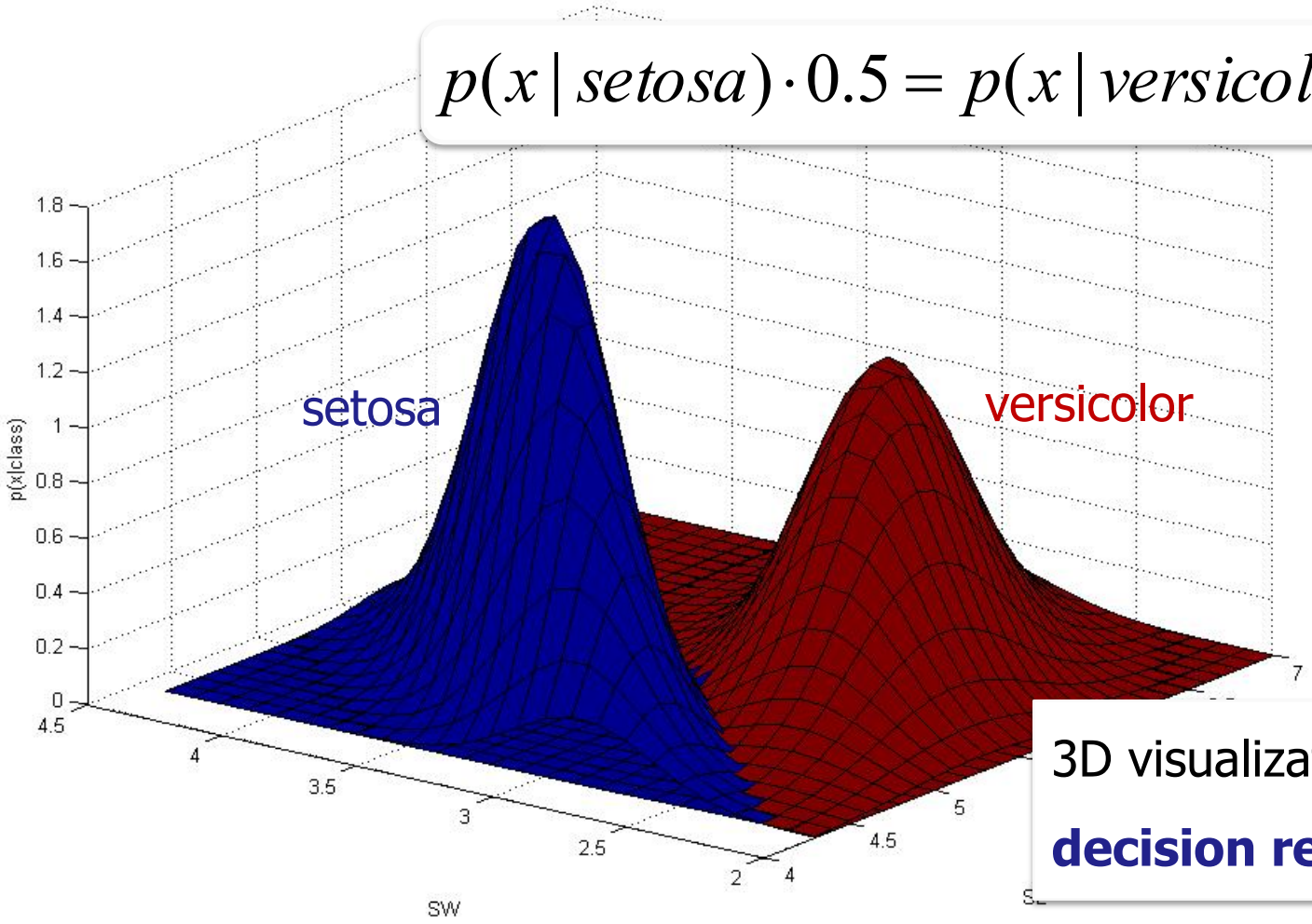
How is this used **in classification?**

2.4 Decision boundaries

Example: Iris classification

Decision regions with different prior probabilities

$$p(x \mid setosa) \cdot 0.5 = p(x \mid versicolor) \cdot 0.5$$



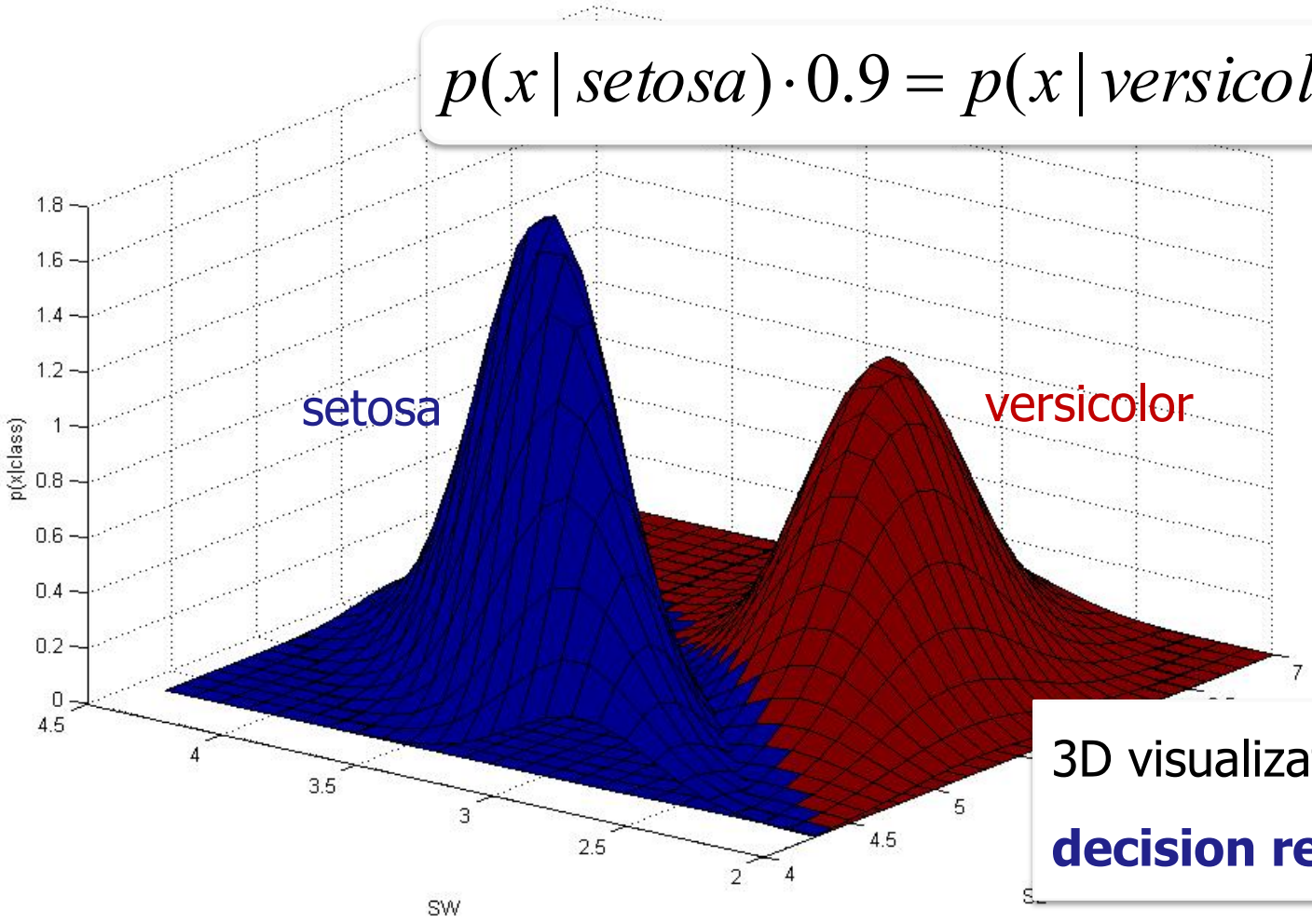
3D visualization of **decision regions**

2.4 Decision boundaries

Example: Iris classification

Decision regions with different prior probabilities

$$p(x \mid setosa) \cdot 0.9 = p(x \mid versicolor) \cdot 0.1$$



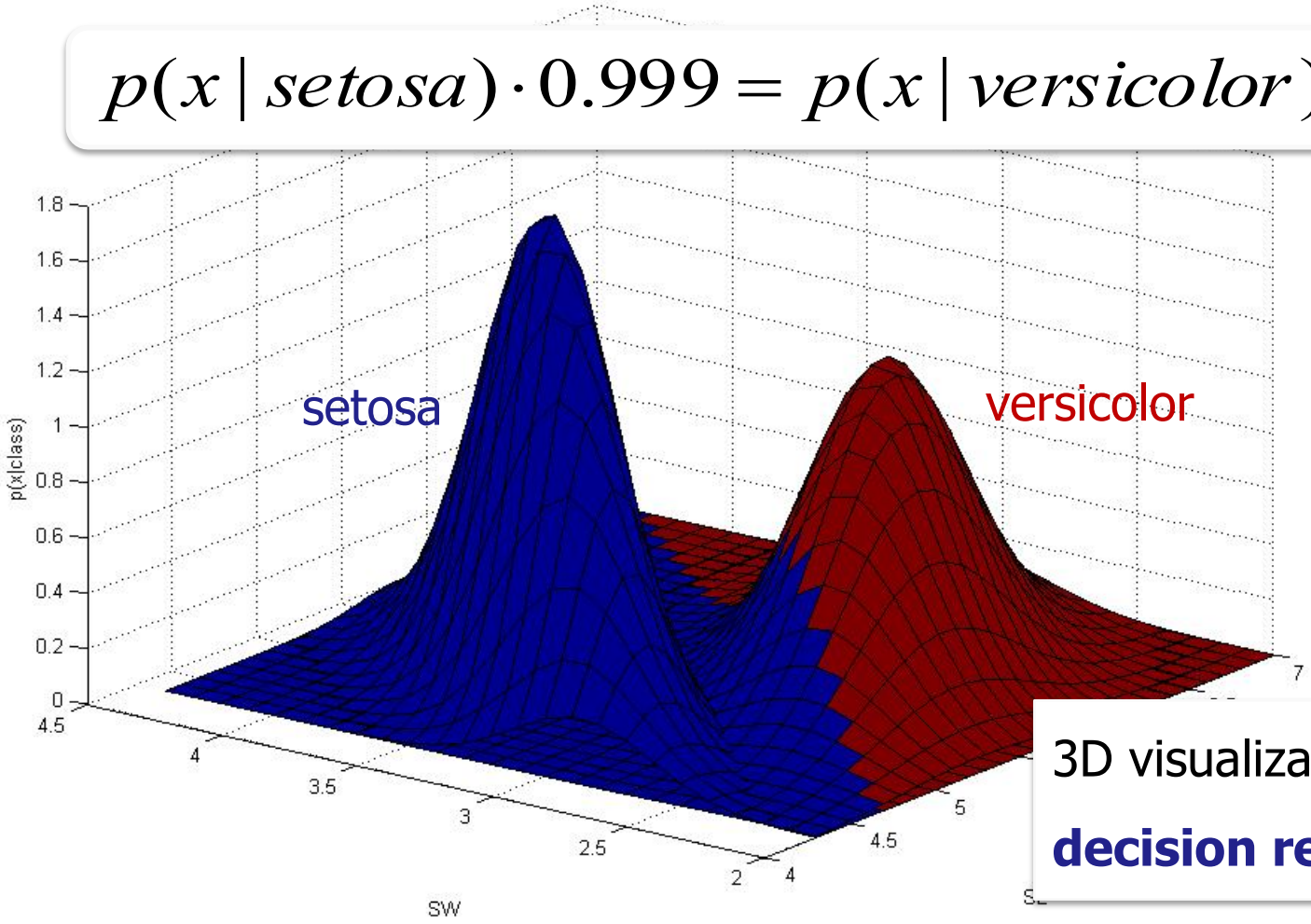
3D visualization of **decision regions**

2.4 Decision boundaries

Example: Iris classification

Decision regions with different prior probabilities

$$p(x \mid setosa) \cdot 0.999 = p(x \mid versicolor) \cdot 0.001$$



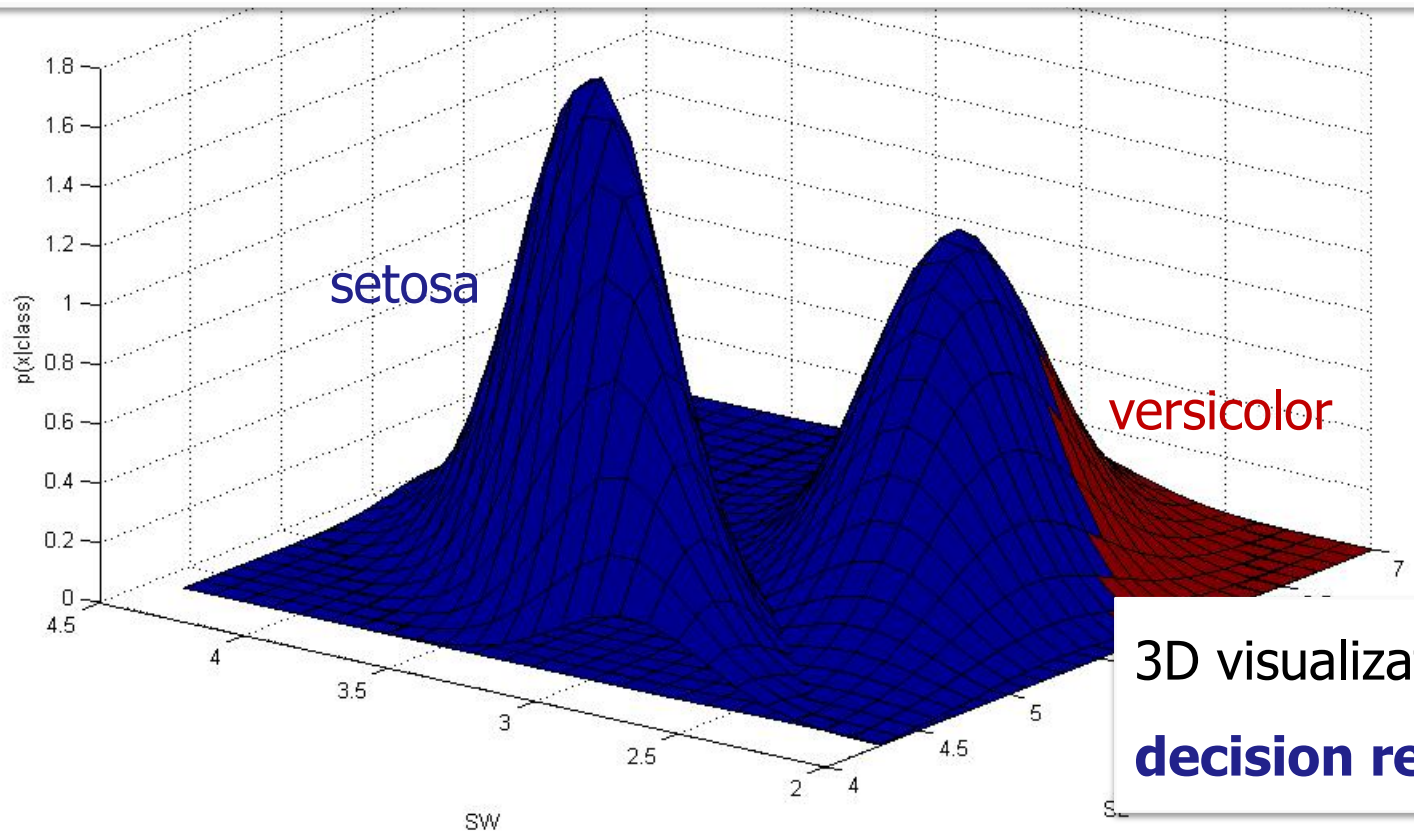
3D visualization of **decision regions**

2.4 Decision boundaries

Example: Iris classification

Decision regions with different prior probabilities

$$p(x \mid setosa) \cdot 0.9...9 = p(x \mid versicolor) \cdot 0.0...01$$



3D visualization of **decision regions**

Practical realization with Octave

```
% ...
% get the class-conditional probabilities
mx = [mean(setosaSL) mean(setosaSW)];
Cx = cov([setosaSL setosaSW]);
p = mvnpdf(X, mx, Cx);
p_x_1 = reshape(p,size(X1,1),size(X2,2));

mx = [mean(versicolorSL) mean(versicolorSW)];
Cx = cov([versicolorSL versicolorSW]);
p = mvnpdf(X, mx, Cx);
p_x_2 = reshape(p,size(X1,1),size(X2,2));

% compute prior probabilities
p_1 = obsPerClass(1) / (obsPerClass(1) + obsPerClass(2));
p_2 = obsPerClass(2) / (obsPerClass(1) + obsPerClass(2));
% p_1 = 0.9;  % manual setting of priors
% p_2 = 0.1;
```

3D visualization of

decision regions

Practical realization with Matlab

```
% ...
% get the class-conditional probabilities
mx = [mean(setosa.SL) mean(setosa.SW)];
Cx = cov([setosa.SL setosa.SW]);
p = mvnpdf(X, mx, Cx);
p_x_1 = reshape(p,size(X1,1),size(X2,2));
```

$p(x \mid setosa)$

```
mx = [mean(versicolor.SL) mean(versicolor.SW)];
Cx = cov([versicolor.SL versicolor.SW]);
p = mvnpdf(X, mx, Cx);
p_x_2 = reshape(p,size(X1,1),size(X2,2));
```

$p(x \mid versicolor)$

```
% compute prior probabilities
p_1 = obsPerClass(1) / (obsPerClass(1) + obsPerClass(2));
p_2 = obsPerClass(2) / (obsPerClass(1) + obsPerClass(2));
% p_1 = 0.9;  % manual setting of priors
% p_2 = 0.1;
```

$P(\text{setosa})$

$P(versicolor)$

Practical realization with Matlab

```
% compute the evidence p(x)
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

Practical realization with Matlab

```
% compute the evidence p(x)
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

$$p(x) = p(x \mid setosa) \cdot P(\ setosa) + p(x \mid versicolor) \cdot P(\ versicolor)$$

Practical realization with Matlab

```
% compute the evidence p(x)
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

$$p(x) = p(x \mid setosa) \cdot P(\ setosa) + p(x \mid versicolor) \cdot P(\ versicolor)$$

Practical realization with Matlab

```
% compute the evidence
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

```
% compute the posteriors
p_1_x = p_x_1_p_1 ./ p_x;
p_2_x = p_x_2_p_2 ./ p_x;
```

Practical realization with Matlab

```
% compute the evidence
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

$$p(setosa \mid x) = \frac{p(x \mid setosa) \cdot P(\ setosa)}{p(x)}$$

```
% compute the posteriors
p_1_x = p_x_1_p_1 ./ p_x;
p_2_x = p_x_2_p_2 ./ p_x;
```

Practical realization with Matlab

```
% compute the evidence
p_x_1_p_1 = p_x_1 * p_1;
p_x_2_p_2 = p_x_2 * p_2;
p_x = p_x_1_p_1 + p_x_2_p_2;
```

$$p(\ versicolor\,|\,x) = \frac{p(x|\ \ versicolor)\cdot P(\ versicolor)}{p(x)}$$

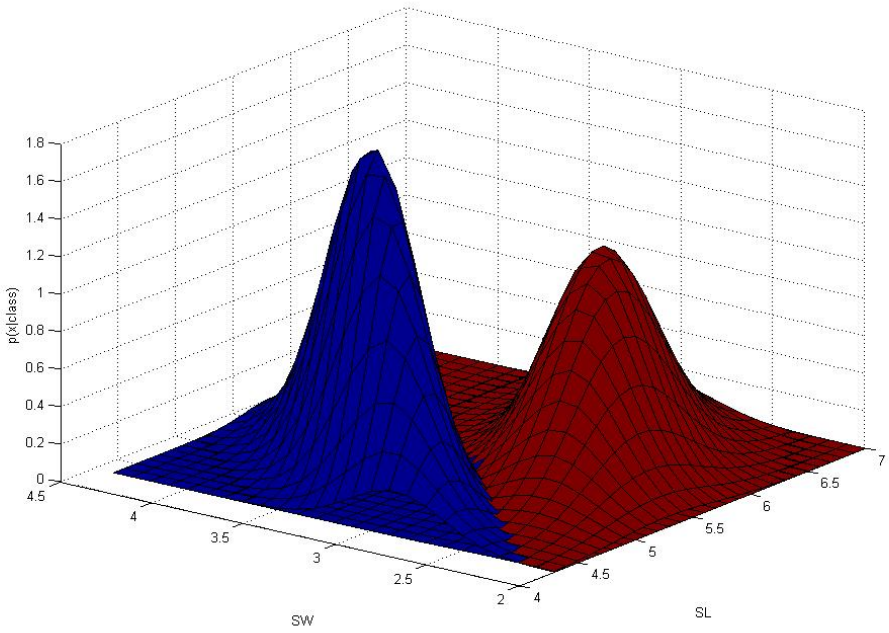
```
% compute the posteriors
p_1_x = p_x_1_p_1 ./ p_x;
p_2_x = p_x_2_p_2 ./ p_x;
```

Practical realization with Matlab

```
figure;
xlabel('SL');
ylabel('SW');
zlabel('p(x|class)');
hold on;

% show decision boundary
C1 = 0.4*double(p_1_x > p_2_x);
C2 = 0.6*double(p_2_x > p_1_x);
C = C1+C2;
surf(pts_x, pts_y, p_x_1, C);
surf(pts_x, pts_y, p_x_2, C);
view(-53,27);
grid;
```

Generate color matrix for usage in function `surf`.

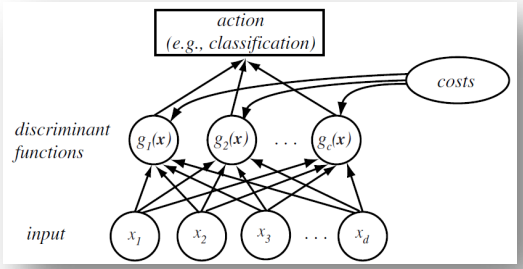
Choose a viewpoint for image plot.



2.5 Minimum distance classifier

Chapter 2.3: Minimum error-rate achieved by discriminant function

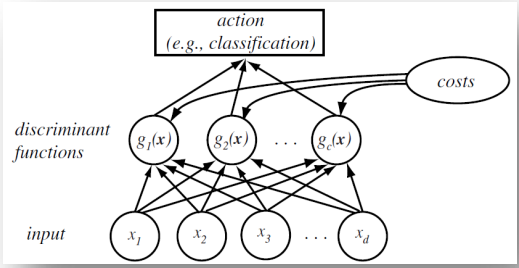
$$g_i(x) = ln\ p(x \mid \omega_i) + ln\ P(\omega_i)$$



2.5 Minimum distance classifier

Chapter 2.3: Minimum error-rate achieved by discriminant function

$$g_i(x) = ln\, p(x \mid \omega_i) + ln\, P(\omega_i)$$



Assumption 1:

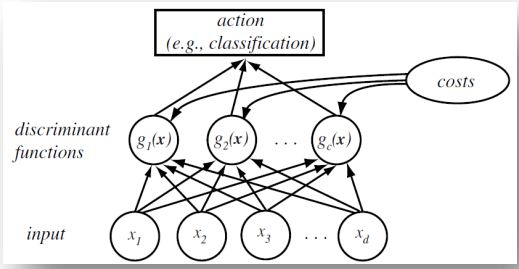
We use a multivariate normal density for modeling $p(x \mid \omega_i)$:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma_i)}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_i)^t \cdot \Sigma_i^{-1} \cdot (x-\mu_i)}$$

2.5 Minimum distance classifier

Chapter 2.3: Minimum error-rate achieved by discriminant function

$$g_i(x) = ln\ p(x \mid \omega_i) + ln\ P(\omega_i)$$



Assumption 1:

We use a multivariate normal density for modeling $p(x \mid \omega_i)$:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma_i)}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_i)^t \cdot \Sigma_i^{-1} \cdot (x-\mu_i)}$$

Assumption 2:

We consider the case $\Sigma_i = \sigma^2 . I$ (I stands for the identity matrix)

- Feature vector components are statistically independent → diagonal covariance matrix
- Each feature vector component has the same variance σ^2

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma_i)}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_i)^t \cdot \Sigma_i^{-1} \cdot (x-\mu_i)}$$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma_i)}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_i)^t \cdot \Sigma_i^{-1} \cdot (x-\mu_i)}$$

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\sigma^2 I)}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_i)^t \cdot (\sigma^2 I)^{-1} \cdot (x-\mu_i)}$$

σ^{2D}

$1/\sigma^2 \cdot I$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\sigma^2 I)}} \cdot e^{-\frac{1}{2}\cdot(x-\mu_i)^t \cdot (\sigma^2 I)^{-1} (x-\mu_i)}$$

σ^{2D}

$1/\sigma^2 \cdot I$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\sigma^2 I)}} \cdot e^{-\frac{1}{2}\cdot(x-\mu_i)^t \cdot (\sigma^2 I)^{-1} (x-\mu_i)}$$

σ^{2D}

$1/\sigma^2 \cdot I$

$$p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D \cdot \sigma^{2D}}} \cdot e^{-\frac{1}{2\sigma^2}\cdot(x-\mu_i)^t (x-\mu_i)}$$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$\ln p(x \mid \omega_i) = \ln \frac{1}{\sqrt{(2\pi)^D \cdot \sigma^{2D}}} \cdot e^{-\frac{1}{2\sigma^2} \cdot (x-\mu_i)^t (x-\mu_i)}$$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$\ln p(x \mid \omega_i) = \ln \frac{1}{\sqrt{(2\pi)^D \cdot \sigma^{2D}}} \cdot e^{-\frac{1}{2\sigma^2} \cdot (x-\mu_i)^t (x-\mu_i)}$$

$$\ln p(x \mid \omega_i) = \ln \frac{1}{\sqrt{(2\pi)^D \cdot \sigma^{2D}}} + \ln e^{-\frac{1}{2\sigma^2} \cdot (x-\mu_i)^t (x-\mu_i)}$$

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$\ln p(x \mid \omega_i) = \ln\left[(2\pi)^D \cdot \sigma^{2D}\right]^{-\frac{1}{2}} - \frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i) \cdot \ln e$$

1

independent of class ω_i

→ unimportant additive constant in discriminant function

→ can be ignored when introducing the term into discriminant function

2.5 Minimum distance classifier

Influence of assumption 2 ($\Sigma_i = \sigma^2 . I$) on normal density:

$$\ln p(x \mid \omega_i) = \ln\left[(2\pi)^D \cdot \sigma^{2D}\right]^{-\frac{1}{2}} - \frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i) \cdot \ln e$$

$$\ln p(x \mid \omega_i) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = \ln p(x \mid \omega_i) + \ln P(\omega_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = \ln p(x \mid \omega_i) + \ln P(\omega_i)$$

Insert $$\ln p(x \mid \omega_i) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = \ln p(x \mid \omega_i) + \ln P(\omega_i)$$

Insert $$\ln p(x \mid \omega_i) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i)$$

$$g_i(x) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i) + \ln P(\omega_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i) + \ln P(\omega_i)$$

class independent → ignore

If priors are the same for all classes, this term becomes another unimportant additive constant → ignore

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -\frac{1}{2\sigma^2} \cdot (x - \mu_i)^t (x - \mu_i) + \ln P(\omega_i)$$

$$g_i(x) = -(x - \mu_i)^t (x - \mu_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -(x - \mu_i)^t (x - \mu_i)$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -\left(x - \mu_i\right)^t \left(x - \mu_i\right)$$

$$\sum_{d=1}^{D} \left(x_d - \mu_{id}\right)^2$$

squared Euclidean Distance

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -(x-\mu_i)^t(x-\mu_i)$$

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

squared Euclidean Distance

Euclidean Distance

$$d_2(x,\mu_k) = \sqrt{\sum_{d=1}^{D}(x_d-\mu_{kd})^2}$$

2.5 Minimum distance classifier

Discriminant function:

$$g_i(x) = -\sum_{d=1}^{D} (x_d - \mu_{id})^2$$

or

$$g_i(x) = -[d_2(x, \mu_i)]^2$$

with $d_2(x, \mu_k) = \sqrt{\sum_{d=1}^{D} (x_d - \mu_{kd})^2}$

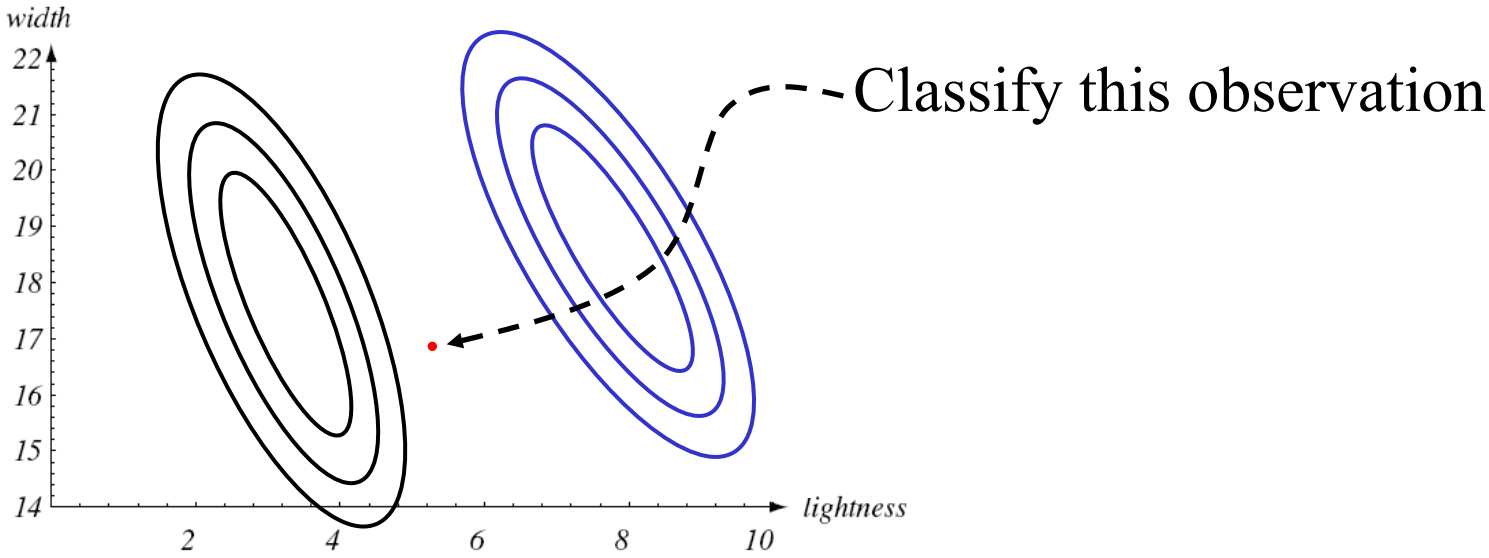
Decide for the class with largest $g_i(x)$

→ Assign observation x to the class which **minimizes distance** between observation vector x and mean vector μ.

2.5 Minimum distance classifier

Decision rule: $$\hat{\omega} = \arg\min_{\omega}\left[d(x, \mu_{\omega})\right]$$

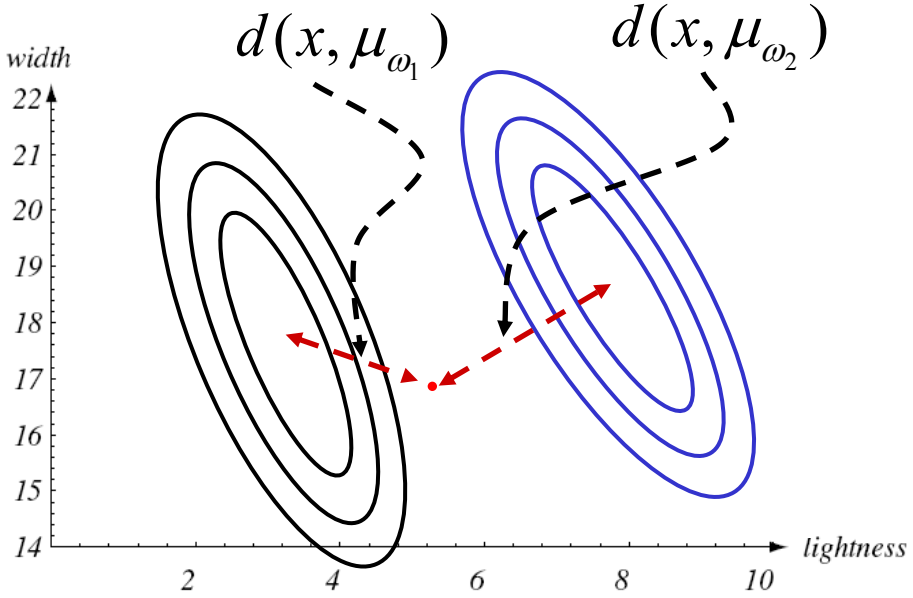
Example:



2.5 Minimum distance classifier

$$\hat{\omega} = \arg\min_{\omega}\left[\, d(x, \mu_{\omega})\right]$$

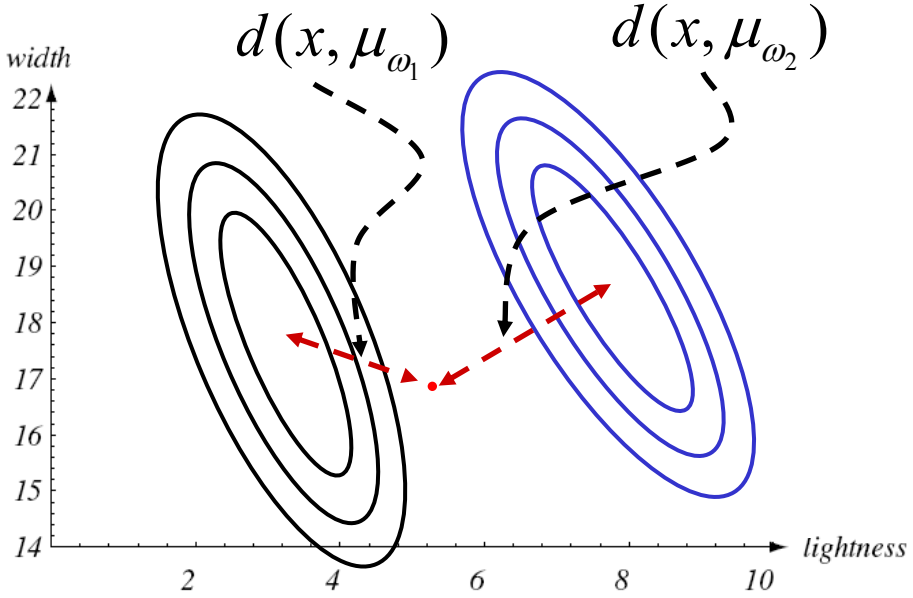
Example:



2.5 Minimum distance classifier

$$\hat{\omega} = \arg\min_{\omega}\left[d(x, \mu_{\omega})\right]$$

Example:



$$d(x, \mu_{\omega_1}) < d(x, \mu_{\omega_2})$$

→ Assign to class 1

2.5 Minimum distance classifier

$$\hat{\omega} = \arg\min_{\omega} \left[d(x, \mu_{\omega}) \right]$$

Other important distance norms:

$$d_1(x, \mu_k) = \sum_{d=1}^{D} \left| x_d - \mu_{kd} \right|$$ L1-Norm

$$d_{\infty}(x, \mu_k) = \max_{d} \left| x_d - \mu_{kd} \right|$$ Chebyshev-Distance

2.6 Application example

Face pixel classification in 2D color images using two classifiers:

- Minimum distance classifier
- Bayes classifier using Gaussian densities with full covariance matrix
- Full script has been uploaded to the material folder in OLAT

2.6 Application example

Step 1: Determine training and test data

training image

images have been taken from the internet

test image

2.6 Application example

Step 2: Get ground truth information for the training image

training image

Region growing

See lecture on image processing

label image

2.6 Application example

Step 2: Get ground truth information for the training image

```
% Read image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait1.jpg');

% Resize image by factor of 10 for faster processing
J = imresize(I,[size(I,1)/10 size(I,2)/10]);

% Convert color image to gray levels (required by regiongrow).
grayJ = rgb2gray(J);

% Region growing requires a starting point which is determined here.
S = zeros(size(grayJ,1), size(grayJ,2));
S(floor(size(S,1)/2), floor(size(S,2)/2)) = 1;  % set center pixel to 1

% regiongrow labels (face) pixels with 1 in the result matrix G.
% Everything else is labeled as 0. DIPUM TOOLBOX
G = regiongrow(grayJ, S, 20);
```

2.6 Application example

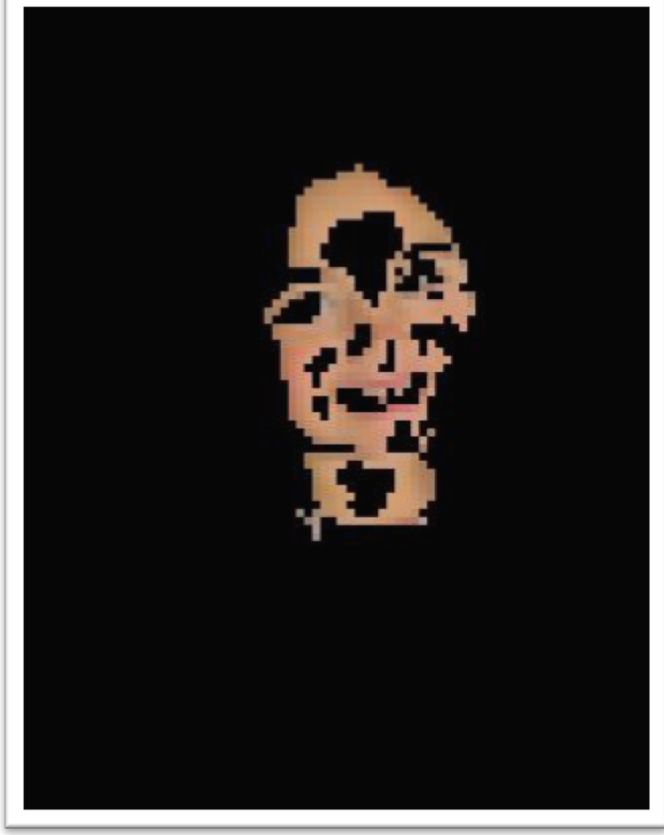
Step 2: Get ground truth information for the training image

```
% Build 3D matrix from G. Required for selecting the face pixels from J
G3D = repmat(G, [1 1 3]); % repeat G-labels in all 3 color dimensions

% Show the face pixels only
K = J .* uint8(G3D);
figure, imshow(K);
```

2.6 Application example

Step 2: Get ground truth information for the training image

```
% Build 3D matrix from G. Required for select
G3D = repmat(G, [1 1 3]); % repeat G-labels i

% Show the face pixels only
K = J .* uint8(G3D);
figure, imshow(K);
```



2.6 Application example

Step 2: Get ground truth information for the training image

```
% Build 3D matrix from G. Required for selecting the face pixels from J
G3D = repmat(G, [1 1 3]); % repeat G-labels in all 3 color dimensions

% Show the face pixels only
K = J .* uint8(G3D);
figure, imshow(K);

% Grep the face pixels from the color image to form feature vectors.
faceFeats = J(logical(G3D));              % column vector
nonFaceFeats = J(~logical(G3D));          % column vector

% Reshape the feature vectors
% first third of (non)faceFeats --> first column
% second third of (non)faceFeats --> second column
% last third of (non)faceFeats --> third column
faceFeats = reshape(faceFeats, [size(faceFeats,1)/3 3]);
nonFaceFeats = reshape(nonFaceFeats, [size(nonFaceFeats,1)/3 3]);
```

2.6 Application example

Step 2: Get ground truth information for the training image

Face feature vectors:

Attention!

```
>> faceFeats(1:5,:)'

ans =

   164   164   175   168   163
   120   119   128   122   116
    87    85    95    89    83
```

Color code: www.colorschemer.com

3 dimensional color vector of one face pixel

R(ed): 168, G(reen): 122, B(lue):89

2.6 Application example

Step 3: Estimate parameters

Get mean vectors and covariance matrices of both classes (face / non-face):

```
% class 1 (faces)
MF = mean(faceFeats);
CF = cov(double(faceFeats));

% class 2 (non-faces)
MN = mean(nonFaceFeats);
CN = cov(double(nonFaceFeats));
```

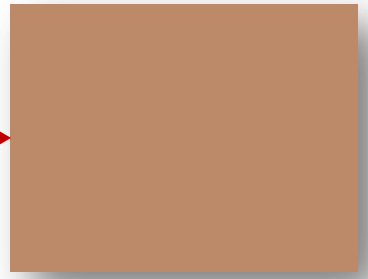
variables: columns
observations: rows

2.6 Application example

Step 3: Estimate parameters

Mean vectors

```
>> MF

MF =

   188.8650   138.1624   104.9708
```



mean face color

```
>> MN

MN =

   146.9406   139.4386   135.0395
```

mean non-face color

2.6 Application example

Step 3: Estimate parameters

Covariance matrices:

```
>> CF
CF =
  188.6837  123.6362  114.9723
  123.6362  145.9900  128.9170
  114.9723  128.9170  174.5110
```

Red (first variable in RGB) varies the most.

```
>> CN
CN =
1.0e+003 *
   4.6452    4.2355    4.1702
   4.2355    4.3569    4.5059
   4.1702    4.5059    4.8697
```

Much stronger variation of non-face pixels.

2.6 Application example

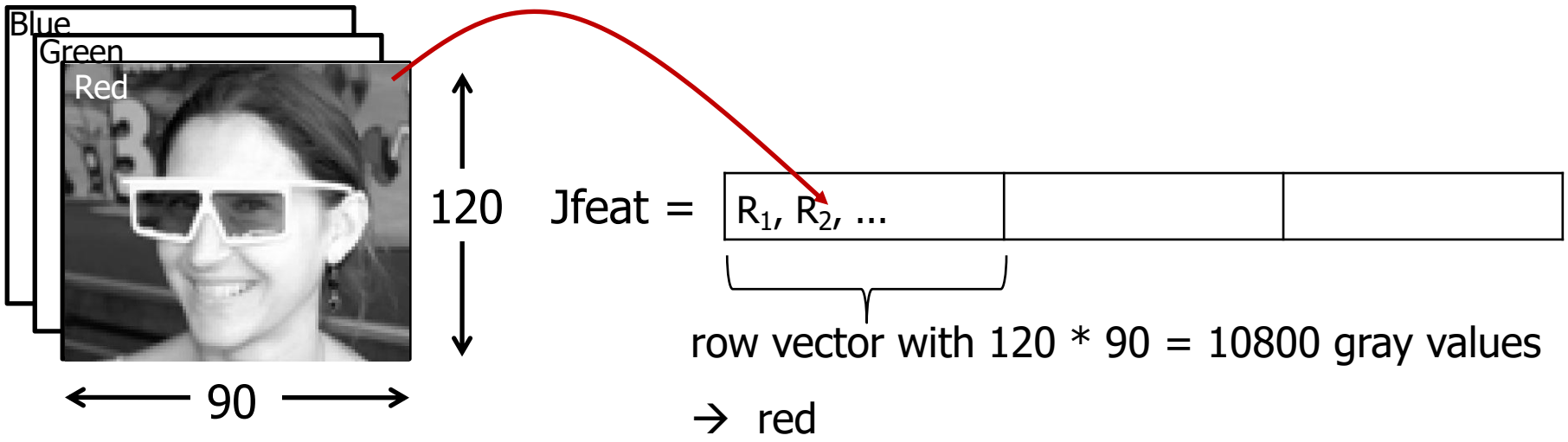
Step 4a: Apply minimum distance classifier

```
% Load a new image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait2.jpg');
J = imresize(I,[size(I,1)/5 size(I,2)/5]);   % down-sampling
```

2.6 Application example

Step 4a: Apply minimum distance classifier

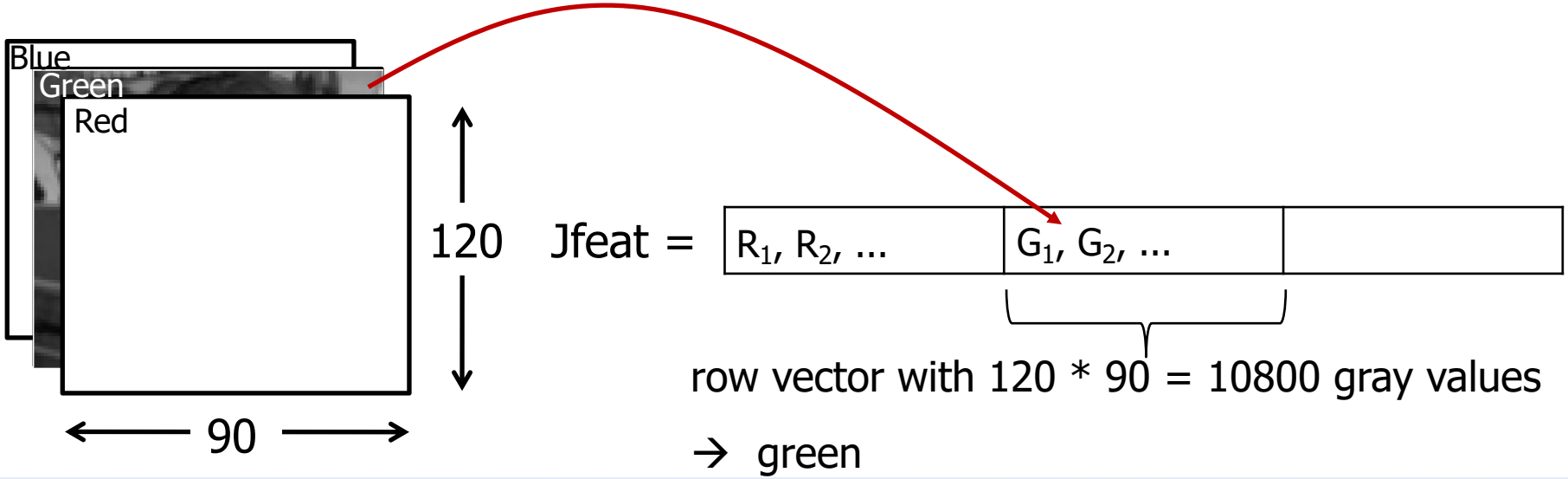
```
% Load a new image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait2.jpg');
J = imresize(I,[size(I,1)/5 size(I,2)/5]);   % down-sampling

% Convert image into feature vectors using im2col
Jfeat = im2col(J, [1 1 3]);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

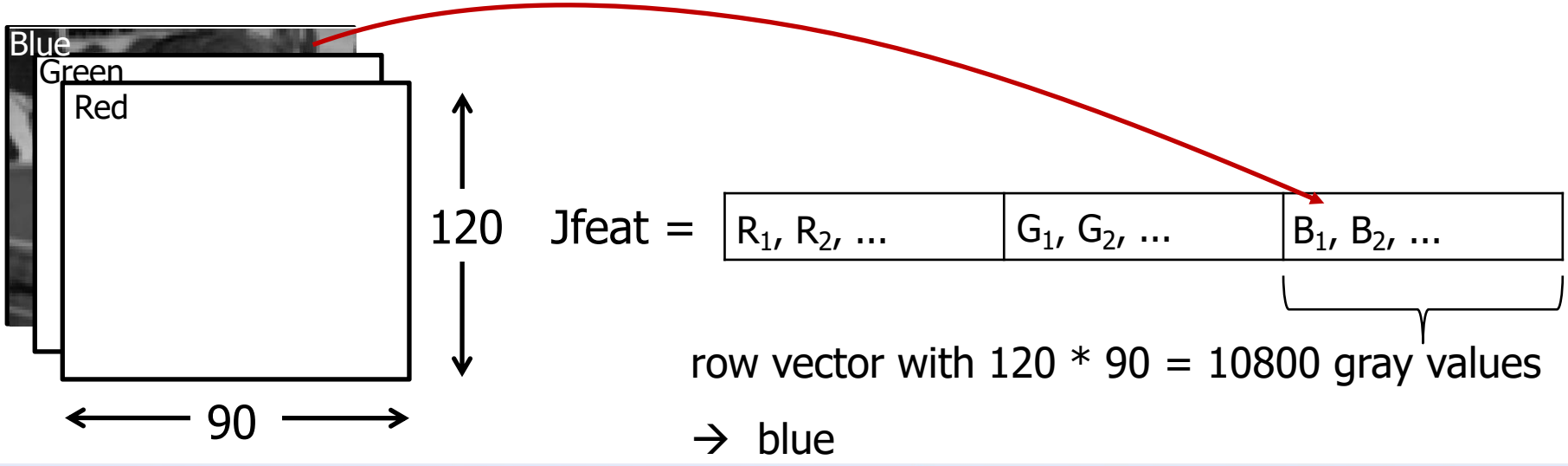
```
% Load a new image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait2.jpg');
J = imresize(I,[size(I,1)/5 size(I,2)/5]);   % down-sampling

% Convert image into feature vectors using im2col
Jfeat = im2col(J, [1 1 3]);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

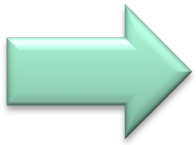
```
% Load a new image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait2.jpg');
J = imresize(I,[size(I,1)/5 size(I,2)/5]);   % down-sampling

% Convert image into feature vectors using im2col
Jfeat = im2col(J, [1 1 3]);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

```
% Load a new image
I = imread('D:\Uebungen\PAT\Exercise3\color_faces\portrait2.jpg');
J = imresize(I,[size(I,1)/5 size(I,2)/5]);   % down-sampling

% Convert image into feature vectors using im2col
Jfeat = im2col(J, [1 1 3]);

% Convert to other format (1 feature vector per column)
Jfeat = reshape(Jfeat, [size(Jfeat,2)/3 3])';
```



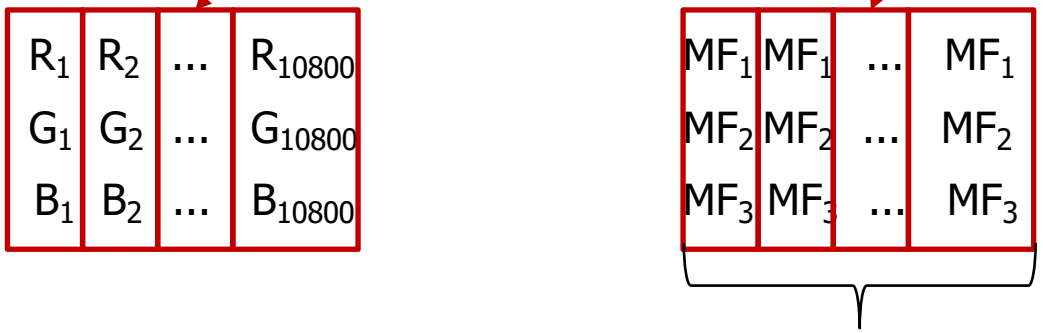
Jfeat =

R_1	R_2	...	R_{10800}
G_1	G_2	...	G_{10800}
B_1	B_2	...	B_{10800}

2.6 Application example

Step 4a: Apply minimum distance classifier

```
% For each feature vector: Compute its distance to both classes
dist_1 = sum((double(Jfeat) - repmat(MF',[1 size(Jfeat,2)])).^2);
dist_2 = sum((double(Jfeat) - repmat(MN',[1 size(Jfeat,2)])).^2);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

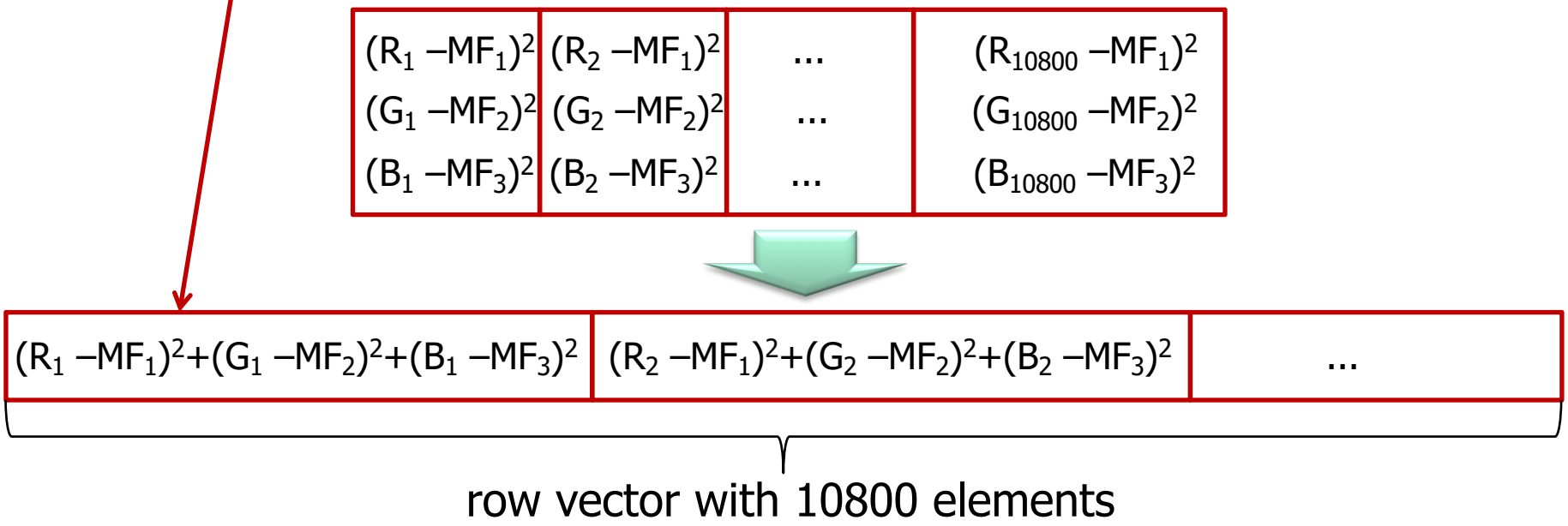
```
% For each feature vector: Compute its distance to both classes
dist_1 = sum((double(Jfeat) - repmat(MF',[1 size(Jfeat,2)])).^2);
dist_2 = sum((double(Jfeat) - repmat(MN',[1 size(Jfeat,2)])).^2);
```

$(R_1 - MF_1)^2$	$(R_2 - MF_1)^2$	...	$(R_{10800} - MF_1)^2$
$(G_1 - MF_2)^2$	$(G_2 - MF_2)^2$	...	$(G_{10800} - MF_2)^2$
$(B_1 - MF_3)^2$	$(B_2 - MF_3)^2$	...	$(B_{10800} - MF_3)^2$

2.6 Application example

Step 4a: Apply minimum distance classifier

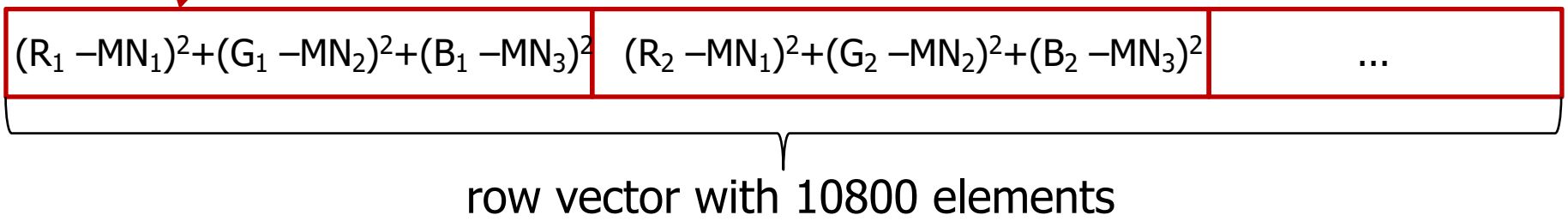
```
% For each feature vector: Compute its distance to both classes
dist_1 = sum((double(Jfeat) - repmat(MF',[1 size(Jfeat,2)])).^2);
dist_2 = sum((double(Jfeat) - repmat(MN',[1 size(Jfeat,2)])).^2);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

```
% For each feature vector: Compute its distance to both classes
dist_1 = sum((double(Jfeat) - repmat(MF',[1 size(Jfeat,2)])).^2);
dist_2 = sum((double(Jfeat) - repmat(MN',[1 size(Jfeat,2)])).^2);
```



2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_i(x) = -\sum_{d=1}^{D} \left(x_d - \mu_{id}\right)^2$$

discriminant function

2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_1(x) = -\left((x_1 - \mu_{11})^2 + (x_2 - \mu_{12})^2 + (x_3 - \mu_{13})^2\right)$$

$$g_i(x) = -\sum_{d=1}^{D} (x_d - \mu_{id})^2$$

discriminant function

$$g_2(x) = -\left((x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2 + (x_3 - \mu_{23})^2\right)$$

2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_1(x) = -\left((x_1 - \mu_{11})^2 + (x_2 - \mu_{12})^2 + (x_3 - \mu_{13})^2\right)$$

value of distance for **class 1** (face), and **pixel 1**

$(R_1 - MF_1)^2+(G_1 - MF_2)^2+(B_1 - MF_3)^2$	$(R_2 - MF_1)^2+(G_2 - MF_2)^2+(B_2 - MF_3)^2$	...

$$g_i(x) = -\sum_{d=1}^{D}(x_d - \mu_{id})^2$$

discriminant function

$$g_2(x) = -\left((x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2 + (x_3 - \mu_{23})^2\right)$$

2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_1(x) = -\left((x_1 - \mu_{11})^2 + (x_2 - \mu_{12})^2 + (x_3 - \mu_{13})^2\right)$$

value of distance for **class 1** (face) and **pixel 2**

$(R_1 - MF_1)^2+(G_1 - MF_2)^2+(B_1 - MF_3)^2$	$(R_2 - MF_1)^2+(G_2 - MF_2)^2+(B_2 - MF_3)^2$	...

$$g_i(x) = -\sum_{d=1}^{D}(x_d - \mu_{id})^2$$

discriminant function

$$g_2(x) = -\left((x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2 + (x_3 - \mu_{23})^2\right)$$

2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_1(x) = -\left((x_1 - \mu_{11})^2 + (x_2 - \mu_{12})^2 + (x_3 - \mu_{13})^2\right)$$

$(R_1 - MF_1)^2+(G_1 - MF_2)^2+(B_1 - MF_3)^2$	$(R_2 - MF_1)^2+(G_2 - MF_2)^2+(B_2 - MF_3)^2$	...

$$g_i(x) = -\sum_{d=1}^{D}(x_d - \mu_{id})^2$$

discriminant function

$(R_1 - MN_1)^2+(G_1 - MN_2)^2+(B_1 - MN_3)^2$	$(R_2 - MN_1)^2+(G_2 - MN_2)^2+(B_2 - MN_3)^2$	...

value of distance for **class 2** (non-face), and **pixel 1**

$$g_2(x) = -\left((x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2 + (x_3 - \mu_{23})^2\right)$$

2.6 Application example

Step 4a: Apply minimum distance classifier

$$g_1(x) = -\left((x_1 - \mu_{11})^2 + (x_2 - \mu_{12})^2 + (x_3 - \mu_{13})^2\right)$$

$(R_1 - MF_1)^2 + (G_1 - MF_2)^2 + (B_1 - MF_3)^2$	$(R_2 - MF_1)^2 + (G_2 - MF_2)^2 + (B_2 - MF_3)^2$	...

$$g_i(x) = -\sum_{d=1}^{D} (x_d - \mu_{id})^2$$

discriminant function

$(R_1 - MN_1)^2 + (G_1 - MN_2)^2 + (B_1 - MN_3)^2$	$(R_2 - MN_1)^2 + (G_2 - MN_2)^2 + (B_2 - MN_3)^2$	...

$$g_2(x) = -\left((x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2 + (x_3 - \mu_{23})^2\right)$$

value of distance for **class 2** (non-face), and **pixel 2**

2.6 Application example

Step 4a: Apply minimum distance classifier

```
% Assign each feature vector to the class to which it has the smallest
% distance. Face pixels will be set to 1 in result.
result = dist_1 < dist_2;
```

Decision rule:

Decide for class with largest discriminant function

→ Decide for class with smallest distance $\sum_{d=1}^{D}(x_d - \mu_{id})^2$

2.6 Application example

Step 4a: Apply minimum distance classifier

```
% Assign each feature vector to the class to which it has the smallest
% distance. Face pixels will be set to 1 in result.
result = dist_1 < dist_2;
```

dist_1

$(R_1 - MF_1)^2+(G_1 - MF_2)^2+(B_1 - MF_3)^2$	$(R_2 - MF_1)^2+(G_2 - MF_2)^2+(B_2 - MF_3)^2$	...

<

dist_2

$(R_1 - MN_1)^2+(G_1 - MN_2)^2+(B_1 - MN_3)^2$	$(R_2 - MN_1)^2+(G_2 - MN_2)^2+(B_2 - MN_3)^2$	...

=

result

0	1	1	0	0	0	1	0	...	1

row vector with 10800 elements: face pixels are set to 1, non-face to 0

2.6 Application example

Step 4a: Apply minimum distance classifier

```
% Assign each feature vector to the class to which it has the smallest
% distance. Face pixels will be set to 1 in result.
result = dist_1 < dist_2;

% Reshape result row vector into a 2D image
classified = reshape(result, size(J, 1), size(J, 2));

% Plot image
figure, imshow(J);
figure, imshow(classified);
```

2.6 Application example

Result with Minimum Distance Classifier:

original image

classification result

face: 1 (white)
non-face: 0 (black)

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Features are required in transposed format ( M x 3 → 3 x M)
faceFeats = faceFeats';
nonFaceFeats = nonFaceFeats';
```

```
% Alternative 1: Compute prior probabilities from training data
p1 = size(faceFeats,2)/(size(faceFeats,2)+size(nonFaceFeats,2));
p2 = size(nonFaceFeats,2)/(size(faceFeats,2)+size(nonFaceFeats,2));
```

```
% Alternative 2: Use the same prior for class 1 and 2
p1 = 0.5;
p2 = 0.5;
```

2 Alternatives for computing the prior probabilities for both classes.

- Interpretation of the priors?
- Which alternative would you prefer in this specific situation?

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

2 Alternatives for computing the prior probabilities for both classes.

- Interpretation of the priors?
- Which alternative would you prefer in this specific situation?

Interpretation of the priors: Percentage the face covers in the image.

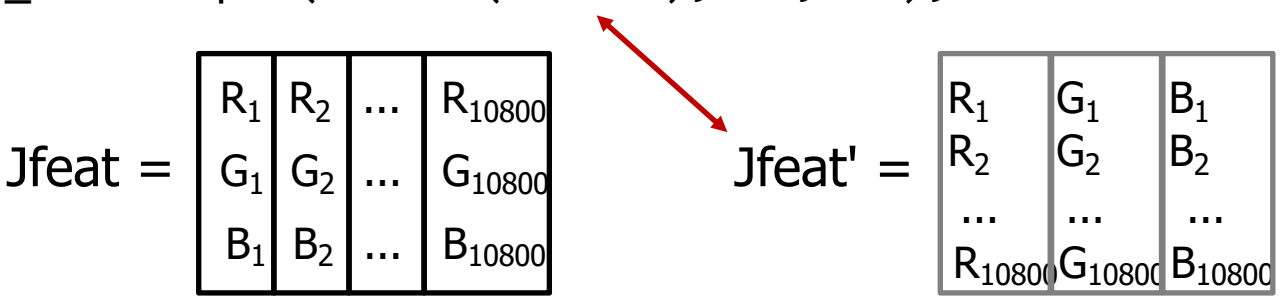
Alternative 1: Should only be used if the training data is representative!

Alternative 2: Same chance for both classes. Reasonable since we do not know which image fraction the test-face covers.

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Features are required in transposed format ( M x 3 → 3 x M)
faceFeats = faceFeats';
nonFaceFeats = nonFaceFeats';

% Use the same prior for class 1 and 2
p1 = 0.5;
p2 = 0.5;

% Compute the likelihoods (class-conditional prob.) for each image
% pixel and each class
p_x_1 = mvnpdf(double(Jfeat'), MF, CF);
p_x_2 = mvnpdf(double(Jfeat'), MN, CN);
```



2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Features are required in transposed format ( M x 3 → 3 x M)
faceFeats = faceFeats';
nonFaceFeats = nonFaceFeats';

% Use the same prior for class 1 and 2
p1 = 0.5;
p2 = 0.5;

% Compute the likelihoods (class-conditional prob.) for each image
% pixel and each class
p_x_1 = mvnpdf(double(Jfeat'), MF, CF);
p_x_2 = mvnpdf(double(Jfeat'), MN, CN);
```

$$p\_x\_1 = \begin{bmatrix} p(R_1, G_1, B_1 \mid \text{class 1}) \\ p(R_2, G_2, B_2 \mid \text{class 1}) \\ \dots \\ p(R_{10800}, G_{10800}, B_{10800} \mid \text{class 1}) \end{bmatrix}$$

column vector with 10800 elements

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Features are required in transposed format ( M x 3 → 3 x M)
faceFeats = faceFeats';
nonFaceFeats = nonFaceFeats';

% Use the same prior for class 1 and 2
p1 = 0.5;
p2 = 0.5;

% Compute the likelihoods (class-conditional prob.) for each image
% pixel and each class
p_x_1 = mvnpdf(double(Jfeat'), MF, CF);
p_x_2 = mvnpdf(double(Jfeat'), MN, CN);
```

p_x_1 = $p(R_1, G_1, B_1 \mid \text{class 1})$
$p(R_2, G_2, B_2 \mid \text{class 1})$
...
$p(R_{10800}, G_{10800}, B_{10800} \mid \text{class 1})$

p_x_2 = $p(R_1, G_1, B_1 \mid \text{class 2})$
$p(R_2, G_2, B_2 \mid \text{class 2})$
...
$p(R_{10800}, G_{10800}, B_{10800} \mid \text{class 2})$

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Compute p_x_1_p_1 = p(x|class 1)*P(class 1)
%     and p_x_2_p_2 = p(x|class 2)*P(class 2)
p_x_1_p_1 = p_x_1 * p1;
p_x_2_p_2 = p_x_2 * p2;
```

$$g_i(x) = P(\omega_i|x)$$

Discriminant function MAP Classifier

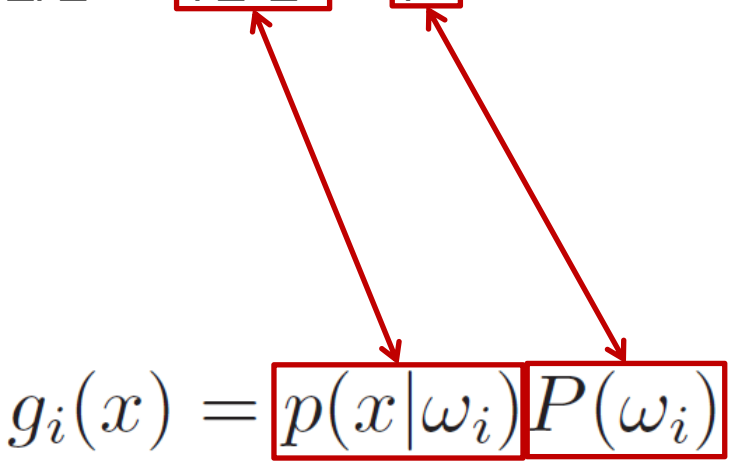
$$g_i(x) = p(x|\omega_i)P(\omega_i)$$

Equivalent discriminant function
(see Chapter 2.3)

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Compute p_x_1_p_1 = p(x|class 1)*P(class 1)
%     and p_x_2_p_2 = p(x|class 2)*P(class 2)
p_x_1_p_1 = p_x_1 * p1;
p_x_2_p_2 = p_x_2 * p2;
```



$$g_i(x) = p(x|\omega_i)P(\omega_i)$$

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Compute p_x_1_p_1 = p(x|class 1)*P(class 1)
%     and p_x_2_p_2 = p(x|class 2)*P(class 2)
p_x_1_p_1 = p_x_1 * p1;
p_x_2_p_2 = p_x_2 * p2;

% Compare according to Bayes decision rule
% Decision for class with largest discriminant
%    -> face pixels will be set to 1
result = p_x_1_p_1 > p_x_2_p_2;
```

2.6 Application example

Step 4b: Bayes classifier using Gaussian densities with full covariance matrix

```
% Compute p_x_1_p_1 = p(x|class 1)*P(class 1)
%     and p_x_2_p_2 = p(x|class 2)*P(class 2)
p_x_1_p_1 = p_x_1 * p1;
p_x_2_p_2 = p_x_2 * p2;

% Compare according to Bayes decision rule
% Decision for class with largest discriminant
%    -> face pixels will be set to 1
result = p_x_1_p_1 > p_x_2_p_2;

% Reshape classification result to image size
classified = reshape(result, size(J, 1), size(J, 2));

% Plot original image and classification result
figure, imshow(J);
figure, imshow(classified);
```

2.6 Application example

Result with Bayes Classifier (full covariance):

original image

classification result

face: 1 (white)
non-face: 0 (black)

2.6 Application example

Result with Minimum Distance Classifier:

original image

classification result

face: 1 (white)
non-face: 0 (black)