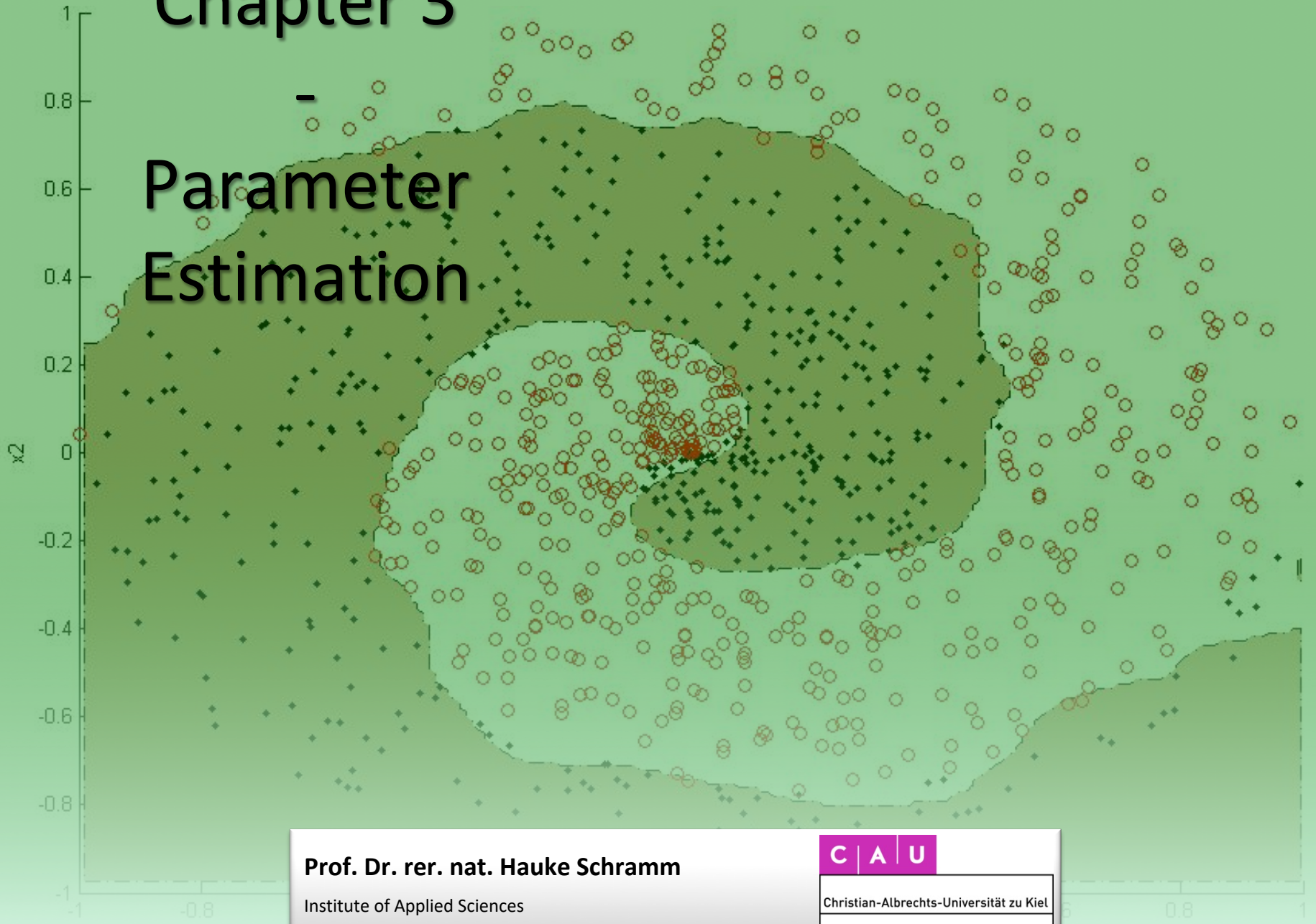
Chapter 3
-
Parameter Estimation



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

Outline of Chapter 3

3. Parameter estimation
    1. Introduction
    2. Maximum-Likelihood Estimation
    3. Bayes Learning
    4. Distribution of Distances
    5. Evaluation Criteria

3. Parameter estimation

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

3.1 Introduction

Chapter 2. has shown:

→ We could design an optimal classifier if we knew:

- $P(\omega_i)$ (the prior probabilities)
- $p(x \mid \omega_i)$ (the class-conditional probabilities)

Unfortunately, we rarely have this complete information!

→ We do have:

- some vague knowledge about the situation and
- a number of training samples

3.1 Introduction

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

One approach to this problem:

- Use samples to estimate the unknown probabilities and prob. densities
- Use the resulting estimates as if they were the true values

Estimation of $P(\omega_i)$ is typically no problem:

For typical pattern classification problems we use *relative class frequencies*.

Estimation of $p(x \mid \omega_i)$ is difficult:

- availability of "sufficient" training data
- unknown functional form of this distribution

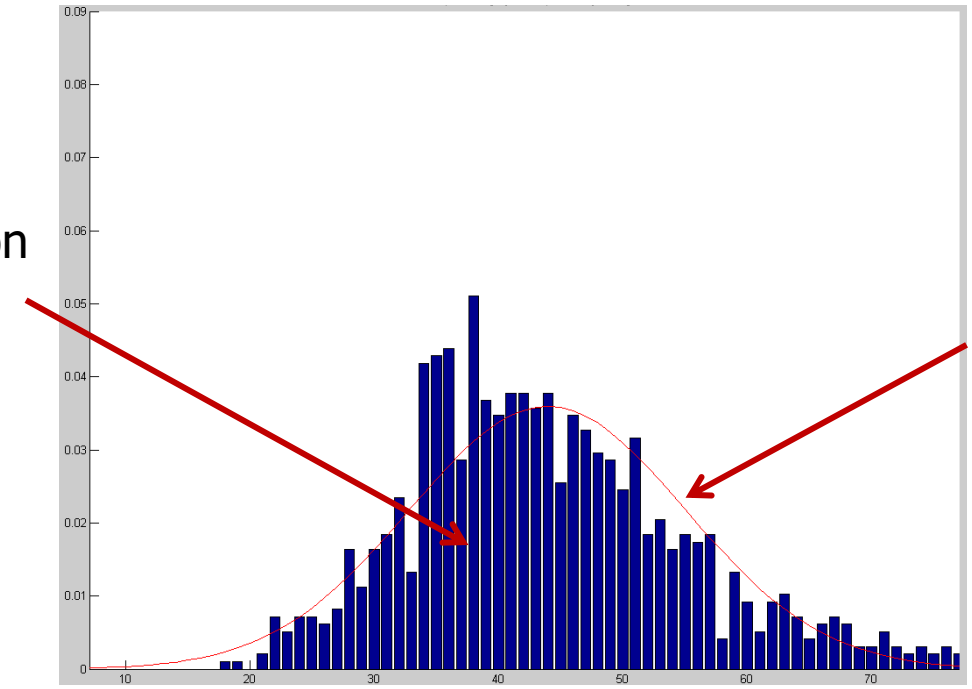
3.1 Introduction

Substantial reduction of complexity if we can **parameterize** $p(x \mid \omega_i)$

What does parameterization mean?

→ make the unknown *function* $p(x \mid \omega_i)$ depend on (a few) parameters

Measured distribution of a 1D variable



Parameterization by

- assuming a normal distribution and
- estimating the parameters (how?) from the training samples.

3.1 Introduction

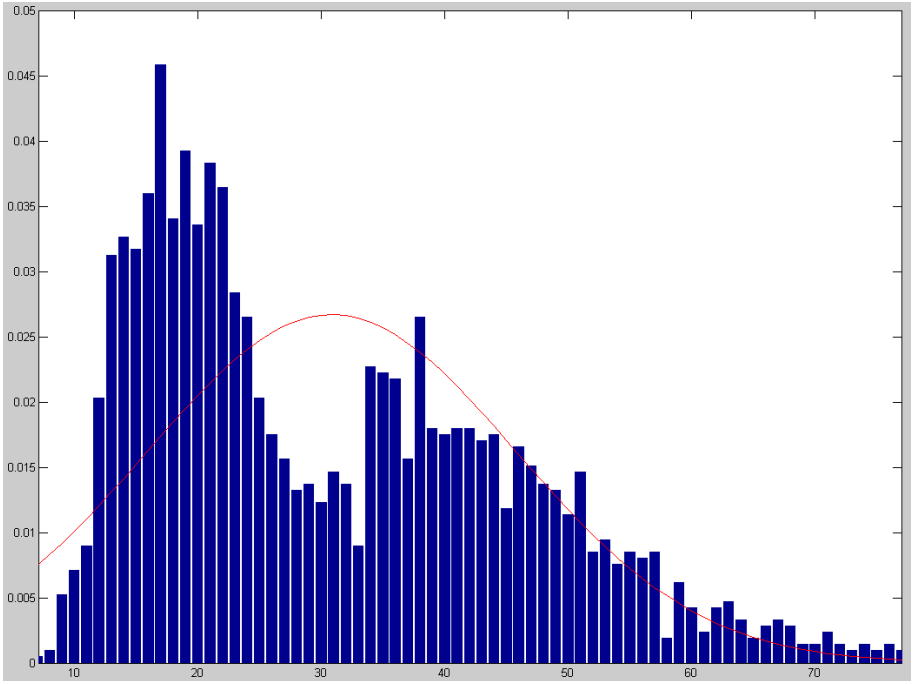
Substantial reduction of complexity if we can **parameterize** $p(x \mid \omega_i)$

What does parameterization mean?

→ make the unknown *function* $p(x \mid \omega_i)$ depend on (a few) parameters

Attention:

Assumption about the functional form must be valid!



3.1 Introduction

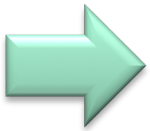
Substantial reduction of complexity if we can **parameterize** $p(x \mid \omega_i)$

What does parameterization mean?

→ make the unknown *function* $p(x \mid \omega_i)$ depend on (a few) parameters

Substantial reduction of complexity

Estimating an un-known function



Estimating the parameters of a known functional form.

e.g. Gaussian density → μ, Σ

3.1 Introduction

Situation:

- Known functional form of the distribution, e.g. Gaussian densitiy
- Unknown but fixed parameters, e.g. μ, Σ

Question:

How can we estimate the value of the unknown parameters from training data?

Answer of the **Maximum-Likelihood technique**:

Maximize the probability of obtaining the observed training samples

3.2 Maximum-likelihood estimation

Properties:

- Has good convergence properties as the sample size increases
- Simpler than any other alternative technique

3.2 Maximum-likelihood estimation

General principle:

Assume we have c classes and

- likelihoods have a **known parametric form**

  Example: normal distribution → $p(x \mid \omega_j) \sim N(\mu_j, \Sigma_j)$

3.2 Maximum-likelihood estimation

General principle:

Assume we have c classes and

- likelihoods have a **known parametric form**

  Example: normal distribution → $p(x \mid \omega_j) \sim N(\mu_j, \Sigma_j)$

- Introduce a parameter vector θ_j for each class j

$$\theta_j = (\mu_j, \Sigma_j) = (\mu_j^1, \mu_j^2, \ldots, \sigma_j^{11}, \sigma_j^{12}, \ldots, \sigma_j^{21}, \sigma_j^{22}, \ldots)$$

  → contains all elements of mean vector and covariance matrix of class j

3.2 Maximum-likelihood estimation

General principle:

Assume we have c classes and

- likelihoods have a **known parametric form.**

  Example: normal distribution → $p(x \mid \omega_j) \sim N(\mu_j, \Sigma_j)$

- Introduce a parameter vector θ_j for each class j

$$\theta_j = (\mu_j, \Sigma_j) = (\mu_j^1, \mu_j^2, \ldots, \sigma_j^{11}, \sigma_j^{12}, \ldots, \sigma_j^{21}, \sigma_j^{22}, \ldots)$$

  → contains all elements of mean vector and covariance matrix of class j

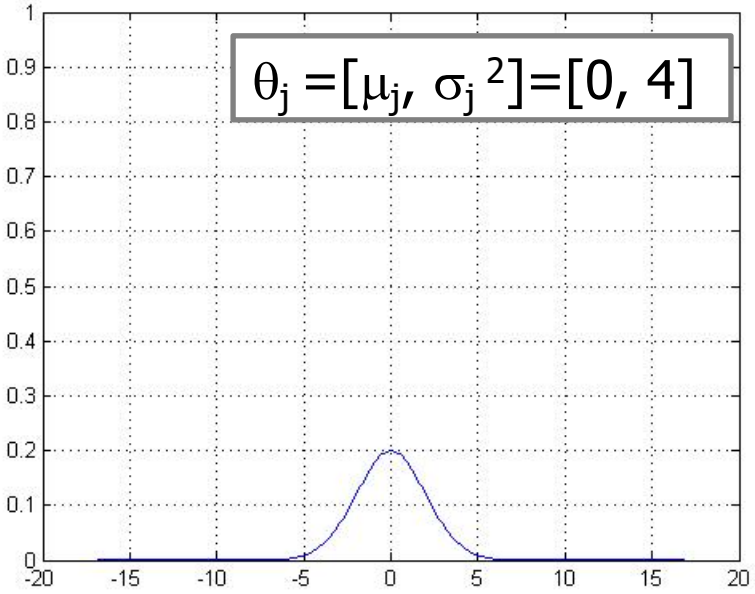
- Dependency of $p(x \mid \omega_j)$ on θ_j is shown by writing $p(x \mid \omega_j) \equiv p(x \mid \omega_j, \theta_j)$

3.2 Maximum-likelihood estimation

Interpretation of $p(x \mid \omega_j, \theta_j)$:

Probability of observation x depends upon

- the class ω_j
- the parameters θ_j of the Gaussian distribution

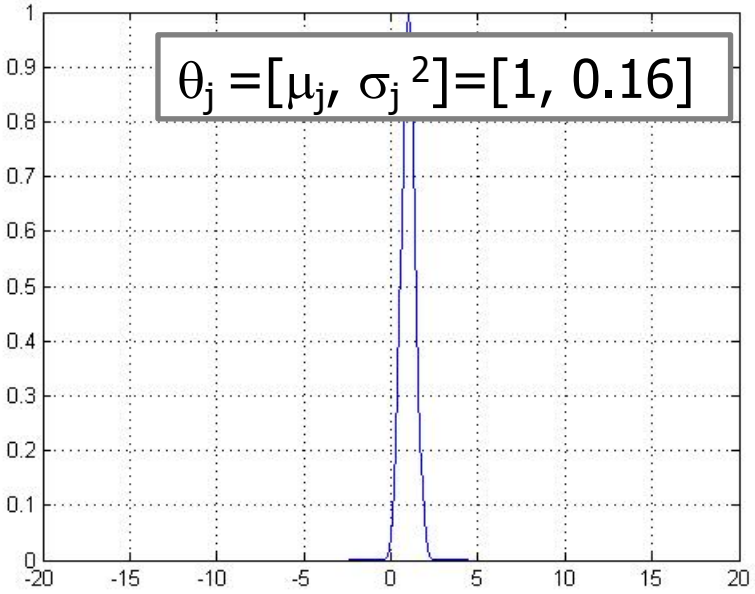


3.2 Maximum-likelihood estimation

Interpretation of $p(x \mid \omega_j, \theta_j)$:

Probability of observation x depends upon

- the class ω_j
- the parameters θ_j of the Gaussian distribution

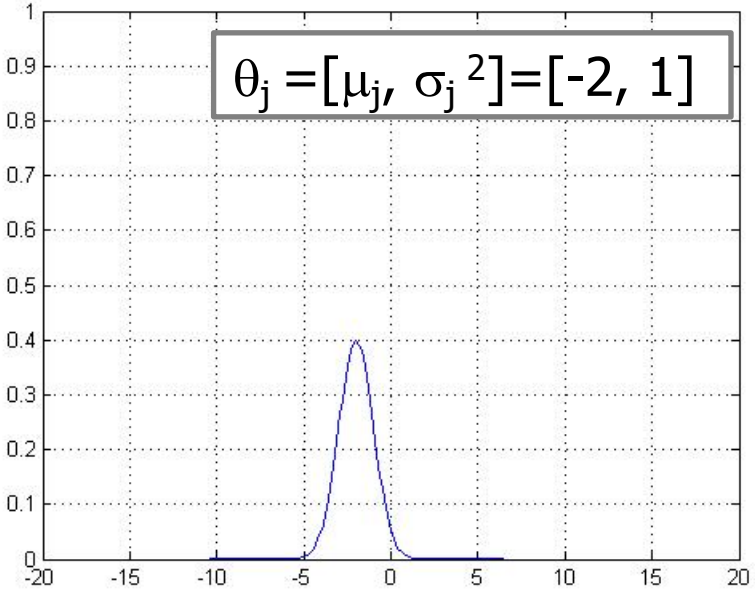


3.2 Maximum-likelihood estimation

Interpretation of $p(x \mid \omega_j, \theta_j)$:

Probability of observation x depends upon

- the class ω_j
- the parameters θ_j of the Gaussian distribution



3.2 Maximum-likelihood estimation

Assumption:

We are given some training data, which contains samples of each class:

$D_1, D_2, \ldots, D_c$

D_2: samples of class ω_2

Goal is to use the training data information to estimate the parameters

$\theta_1, \theta_2, \ldots, \theta_c$

each θ_j (j = 1, 2, …, c) contains the parameters of the corresponding class ω_j

In case of a 1D Gaussian distribution: $\theta_j = [\mu_j, \sigma_j^2]$

3.2 Maximum-likelihood estimation

If samples in D_i give no information about θ_j ($i \neq j$) we may consider

c **separate problems** of the following form:

Given: Training samples $D = x_1, x_2, \ldots, x_n$

independently drawn from the same probab. dens. $p(x \mid \theta)$

identically distributed

Goal: Find an optimal estimation of the class parameters θ.

Maximize the probability of obtaining the observed training samples

3.2 Maximum-likelihood estimation

If samples in D_i give no information about θ_j ($i \neq j$) we may consider

c **separate problems** of the following form:

Given: Training samples $D = x_1, x_2, \ldots, x_n$

independently drawn from the probability density $p(x \mid \theta)$

The **probability of the training data** for this class is:

→ likelihood

$$p(D \mid \theta) = \prod_{k=1}^{n} p(x_k \mid \theta)$$

Likelihood function

3.2 Maximum-likelihood estimation

If samples in D_i give no information about θ_j ($i \neq j$) we may consider

c **separate problems** of the following form:

Given: Training samples $D = x_1, x_2, \ldots, x_n$

independently drawn from the probability density $p(x \mid \theta)$

The **log-likelihood of the training data** for this class is:

$$\ln p(D \mid \theta) = \sum_{k=1}^{n} \ln p(x_k \mid \theta)$$

Log-likelihood function

3.2 Maximum-likelihood estimation

The **maximum-likelihood estimate** of θ is:

$$\hat{\theta} = \arg\max_{\theta} p(D \mid \theta) = \arg\min_{\theta} \{-\ln p(D \mid \theta)\}$$

Maximum likelihood corresponds to minimum negative log-likelihood due to functional form of log-function and probability values between 0 and 1.

3.2 Maximum-likelihood estimation

The **maximum-likelihood estimate** of θ is:

$$\hat{\theta} = \arg\max_{\theta} p(D \mid \theta) = \arg\min_{\theta} \{-\ln p(D \mid \theta)\}$$

or

$$\hat{\theta} = \arg\min_{\theta} \left\{ -\sum_{k=1}^{n} \ln p(x_k \mid \theta) \right\}$$

Informal interpretation:

It is the value of θ that best agrees with the actually observed training sample.

3.2 Maximum-likelihood estimation

Necessary condition for an optimum:

$$\sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) = 0$$

with $$\nabla_\theta = \left[\frac{\partial}{\partial \theta_1}, \frac{\partial}{\partial \theta_2}, \ldots, \frac{\partial}{\partial \theta_p} \right]^t$$

Gradient operator

English: Del
German: Nabla-Operator

Interpretation: vector of partial derivative operators

3.2 Maximum-likelihood estimation

Example:

- Estimation of unknown mean vector μ of a Gaussian distribution
- Known covariance matrix Σ

3.2 Maximum-likelihood estimation

Example:

- Estimation of unknown mean vector μ of a Gaussian distribution
- Known covariance matrix Σ

Likelihood of a single training vector x_k given μ is:

$$p(x_k \mid \mu) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma)}} \cdot e^{-\frac{1}{2} \cdot (x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)}$$

3.2 Maximum-likelihood estimation

Example:

- Estimation of unknown mean vector μ of a Gaussian distribution
- Known covariance matrix Σ

Log-likelihood of a single training vector x_k given μ is:

$$\ln p(x_k \mid \mu) = -\frac{1}{2}\ln\left((2\pi)^D \cdot \det(\Sigma)\right) - \frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)$$

3.2 Maximum-likelihood estimation

Example:

Maximization by $\nabla_\mu \ln p(x_k \mid \mu) = 0$

$$\ln p(x_k \mid \mu) = -\frac{1}{2}\ln\left((2\pi)^D \cdot \det(\Sigma)\right) - \frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)$$

3.2 Maximum-likelihood estimation

Example:

Maximization by $\nabla_\mu \ln p(x_k \mid \mu) = 0$

$$\ln p(x_k \mid \mu) = -\frac{1}{2}\ln\left((2\pi)^D \cdot \det(\Sigma)\right) - \frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)$$

independent of μ

3.2 Maximum-likelihood estimation

Example:

Maximization by $\nabla_\mu \ln p(x_k \mid \mu) = 0$

$$\ln p(x_k \mid \mu) = \underbrace{-\frac{1}{2}\ln\left((2\pi)^D \cdot \det(\Sigma)\right)}_{\text{independent of } \mu} - \frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)$$

$$\nabla_\mu \ln p(x_k \mid \mu) = -\nabla_\mu \left[\frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu)\right]$$

3.2 Maximum-likelihood estimation

Example:

Maximization by

$$\nabla_\mu \ln p(x_k \mid \mu) = -\nabla_\mu \left[\frac{1}{2} (x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

3.2 Maximum-likelihood estimation

Example:

Maximization by

$$\nabla_\mu \ln p(x_k \mid \mu) = -\nabla_\mu \left[\frac{1}{2} (x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$\Rightarrow \quad \nabla_\mu \ln p(x_k \mid \mu) = \Sigma^{-1} \cdot (x_k - \mu)$$

chain rule (see next pages for a 2D example)

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

elements of inverse covariance matrix

Note: $\sigma_{12} = \sigma_{21}$ → $i_{12} = i_{21}$

Inverse of a symmetric matrix is also symmetric (if inverse exists).

$$\nabla_\mu \left[-\frac{1}{2} (x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$= \nabla_\mu \left[-\frac{1}{2} \begin{bmatrix} x_{k1} - \mu_1 & x_{k2} - \mu_2 \end{bmatrix} \cdot \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{k1} - \mu_1 \\ x_{k2} - \mu_2 \end{bmatrix} \right]$$

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

elements of inverse covariance matrix

Note: $\sigma_{12} = \sigma_{21}$ → $i_{12} = i_{21}$

$$\nabla_\mu \left[-\frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$= \nabla_\mu \left[-\frac{1}{2} \begin{bmatrix} x_{k1} - \mu_1 & x_{k2} - \mu_2 \end{bmatrix} \cdot \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{k1} - \mu_1 \\ x_{k2} - \mu_2 \end{bmatrix} \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1} - \mu_1)^2 + i_{21} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{12} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{22} \cdot (x_{k2} - \mu_2)^2 \right] \right]$$

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

$$\nabla_\mu \left[-\frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$= \nabla_\mu \left[-\frac{1}{2} \begin{bmatrix} x_{k1} - \mu_1 & x_{k2} - \mu_2 \end{bmatrix} \cdot \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{k1} - \mu_1 \\ x_{k2} - \mu_2 \end{bmatrix} \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1} - \mu_1)^2 + i_{21} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{12} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{22} \cdot (x_{k2} - \mu_2)^2 \right] \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1} - \mu_1)^2 + 2 \cdot i_{12} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{22} \cdot (x_{k2} - \mu_2)^2 \right] \right]$$

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

$$\nabla_\mu \left[-\frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1} - \mu_1)^2 + 2 \cdot i_{12} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{22} \cdot (x_{k2} - \mu_2)^2 \right] \right]$$

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

$$\nabla_\mu \left[-\frac{1}{2}(x_k - \mu)^t \cdot \Sigma^{-1} \cdot (x_k - \mu) \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1} - \mu_1)^2 + 2 \cdot i_{12} \cdot (x_{k1} - \mu_1) \cdot (x_{k2} - \mu_2) + i_{22} \cdot (x_{k2} - \mu_2)^2 \right] \right]$$

$$= -\frac{1}{2} \cdot \begin{bmatrix} -2 \cdot i_{11} \cdot (x_{k1} - \mu_1) - 2 \cdot i_{12} \cdot (x_{k2} - \mu_2) \\ -2 \cdot i_{12} \cdot (x_{k1} - \mu_1) - 2 \cdot i_{22} \cdot (x_{k2} - \mu_2) \end{bmatrix} = \begin{bmatrix} i_{11} \cdot (x_{k1} - \mu_1) + i_{12} \cdot (x_{k2} - \mu_2) \\ i_{12} \cdot (x_{k1} - \mu_1) + i_{22} \cdot (x_{k2} - \mu_2) \end{bmatrix}$$

3.2 Maximum-likelihood estimation

Example:

Simple illustration with 2D features

$$\nabla_\mu \left[-\frac{1}{2}(x_k-\mu)^t \cdot \Sigma^{-1} \cdot (x_k-\mu) \right]$$

$$= \begin{bmatrix} \frac{\partial}{\partial \mu_1} \\ \frac{\partial}{\partial \mu_2} \end{bmatrix} \left[-\frac{1}{2} \left[i_{11} \cdot (x_{k1}-\mu_1)^2 + 2 \cdot i_{12} \cdot (x_{k1}-\mu_1) \cdot (x_{k2}-\mu_2) + i_{22} \cdot (x_{k2}-\mu_2)^2 \right] \right]$$

$$= -\frac{1}{2} \cdot \begin{bmatrix} -2 \cdot i_{11} \cdot (x_{k1}-\mu_1) - 2 \cdot i_{12} \cdot (x_{k2}-\mu_2) \\ -2 \cdot i_{12} \cdot (x_{k1}-\mu_1) - 2 \cdot i_{22} \cdot (x_{k2}-\mu_2) \end{bmatrix} = \begin{bmatrix} i_{11} \cdot (x_{k1}-\mu_1) + i_{12} \cdot (x_{k2}-\mu_2) \\ i_{12} \cdot (x_{k1}-\mu_1) + i_{22} \cdot (x_{k2}-\mu_2) \end{bmatrix}$$

$$= \begin{bmatrix} i_{11} & i_{12} \\ i_{12} & i_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{k1}-\mu_1 \\ x_{k2}-\mu_2 \end{bmatrix} = \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{k1}-\mu_1 \\ x_{k2}-\mu_2 \end{bmatrix} = \Sigma^{-1} \cdot [x_k - \mu]$$

3.2 Maximum-likelihood estimation

Example:

Maximum likelihood condition (see above):

$$\sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) = 0$$

3.2 Maximum-likelihood estimation

Example:

Maximum likelihood condition (see above):

$$\sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) = 0$$

For our special example:

$$\nabla_\theta \ln p(x_k \mid \theta) = \nabla_\mu \ln p(x_k \mid \mu) = \Sigma^{-1} \cdot (x_k - \mu)$$

3.2 Maximum-likelihood estimation

Example:

Maximum likelihood condition (see above):

$$\sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) = 0$$

For our special example:

$$\nabla_\theta \ln p(x_k \mid \theta) = \nabla_\mu \ln p(x_k \mid \mu) = \Sigma^{-1} \cdot (x_k - \mu)$$

and thus, the maximum-likelihood estimate of μ must satisfy

inverse covariance matrix

$$\sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) = \sum_{k=1}^{n} \nabla_\mu \ln p(x_k \mid \mu) = \sum_{k=1}^{n} \Sigma^{-1} \cdot (x_k - \mu) = 0$$

3.2 Maximum-likelihood estimation

Example:

Maximum-likelihood estimate of μ

$$\sum_{k=1}^{n} \Sigma^{-1} \cdot \left(x_k - \hat{\mu} \right) = 0$$

← Multiplication with Σ

3.2 Maximum-likelihood estimation

Example:

$$\sum_{k=1}^{n} \Sigma^{-1} \cdot \left(x_k - \hat{\mu}\right) = 0$$

← Multiplication with Σ

$$\sum_{k=1}^{n} \left(x_k - \hat{\mu}\right) = 0$$

3.2 Maximum-likelihood estimation

Example:

$$\sum_{k=1}^{n} \Sigma^{-1} \cdot \left(x_k - \hat{\mu}\right) = 0$$

← Multiplication with Σ

$$\sum_{k=1}^{n} \left(x_k - \hat{\mu}\right) = 0$$

$$\sum_{k=1}^{n} x_k - \sum_{k=1}^{n} \hat{\mu} = \sum_{k=1}^{n} x_k - \hat{\mu} \sum_{k=1}^{n} 1 = \sum_{k=1}^{n} x_k - \hat{\mu} \cdot n = 0$$

3.2 Maximum-likelihood estimation

Example:

$$\sum_{k=1}^{n} \Sigma^{-1} \cdot \left(x_k - \hat{\mu}\right) = 0$$

← Multiplication with Σ

$$\sum_{k=1}^{n} \left(x_k - \hat{\mu}\right) = 0$$

$$\sum_{k=1}^{n} x_k - \sum_{k=1}^{n} \hat{\mu} = \sum_{k=1}^{n} x_k - \hat{\mu} \sum_{k=1}^{n} 1 = \sum_{k=1}^{n} x_k - \hat{\mu} \cdot n = 0$$

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k$$

3.2 Maximum-likelihood estimation

Example:

$$\sum_{k=1}^{n} \Sigma^{-1} \cdot \left(x_k - \hat{\mu}\right) = 0 \qquad \leftarrow \text{Multiplication with } \Sigma$$

$$\sum_{k=1}^{n} \left(x_k - \hat{\mu}\right) = 0$$

$$\sum_{k=1}^{n} x_k - \sum_{k=1}^{n} \hat{\mu} = \sum_{k=1}^{n} x_k - \hat{\mu} \sum_{k=1}^{n} 1 = \sum_{k=1}^{n} x_k - \hat{\mu} \cdot n = 0$$

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k \qquad \text{2D case :} \qquad \begin{bmatrix} \hat{\mu}_1 \\ \hat{\mu}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{n} \cdot \sum_{k=1}^{n} x_{k1} \\ \frac{1}{n} \cdot \sum_{k=1}^{n} x_{k2} \end{bmatrix}$$

3.2 Maximum-likelihood estimation

Example:

Very satisfying result:

Maximum-likelihood estimate of the unknown parameter is the

arithmetic average of the training samples

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

3.2 Maximum-likelihood estimation

Another Example:

- Gaussian distribution
- Estimation of unknown mean vector μ and unknown covariance matrix Σ

3.2 Maximum-likelihood estimation

Another Example:

- Gaussian distribution
- Estimation of unknown mean vector μ and unknown covariance matrix Σ

With the same method as described above we obtain:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n} (x_k - \hat{\mu}) \cdot (x_k - \hat{\mu})^t$$

x_k and μ are vectors

→ one matrix for each of the n training samples.

3.2 Maximum-likelihood estimation

Another Example:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}\left(x_k - \hat{\mu}\right)\cdot\left(x_k - \hat{\mu}\right)^t$$

arithmetic average over the n matrixes

3.2 Maximum-likelihood estimation

Another Example:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}\left(x_k - \hat{\mu}\right)\cdot\left(x_k - \hat{\mu}\right)^t$$

arithmetic average over the n matrixes

Again, a very satisfying result since the true covariance matrix is by definition the expected value of the matrix $\left(x - \hat{\mu}\right)\cdot\left(x - \hat{\mu}\right)^t$

$$E[f(x)] = \sum_{x \in X} f(x)P(x)$$

equally distributed P(x) → 1/n

3.2 Maximum-likelihood estimation

Another Example:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}\left(x_k - \hat{\mu}\right)\cdot\left(x_k - \hat{\mu}\right)^t$$

Practical remarks:

Octave functions `mean` and `cov` provide these measurements

→ When using these functions for computing the parameters of the normal distribution, **we follow the maximum-likelihood principle**

→ Late justification for our approach in Chapter 2.

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

$$p(x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

with $x \in \mathbb{N}_0$ and $\lambda \in \mathbb{R}_+$

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

$$p(x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

with $x \in \mathbb{N}_0$ and $\lambda \in \mathbb{R}_+$

Likelihood function:

$$L(\lambda) = \prod_{i=1}^{n} p(x_i \mid \lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$$

only for $x_i \in \mathbb{N}_0$

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

$$p(x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

with $x \in \mathbb{N}_0$ and $\lambda \in \mathbb{R}_+$

Likelihood function:

$$L(\lambda) = \prod_{i=1}^{n} p(x_i \mid \lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$$

only for $x_i \in \mathbb{N}_0$

Log-likelihood function:

$$\ln L(\lambda) = \sum_{i=1}^{n} \ln\left(\frac{\lambda^{x_i}}{x_i!} e^{-\lambda}\right) = \sum_{i=1}^{n} \left(x_i \ln(\lambda) - \ln(x_i!) - \lambda\right)$$

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

Likelihood function:

$$\ln L(\lambda) = \sum_{i=1}^{n} (x_i \ln(\lambda) - \ln(x_i!) - \lambda) = \ln(\lambda) \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \ln(x_i!) - n\lambda$$

Derive and set to zero:

$$\frac{d}{d\lambda} \ln L(\lambda) = \frac{1}{\lambda} \sum_{i=1}^{n} x_i - n = 0$$

Result:

$$\lambda = \frac{1}{n} \sum_{i=1}^{n} x_i = \mu_x$$

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

Check second order derivative to ensure that this is a maximum:

$$\frac{d^2}{(d\lambda)^2} \ln L(\lambda) = -\frac{1}{\lambda^2} \sum_{i=1}^{n} x_i < 0$$

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

Practical validation of result:

For all values of parameter λ

Compute Log-likelihood function for

training samples $D = x_1, x_2, \ldots, x_n$ and given λ

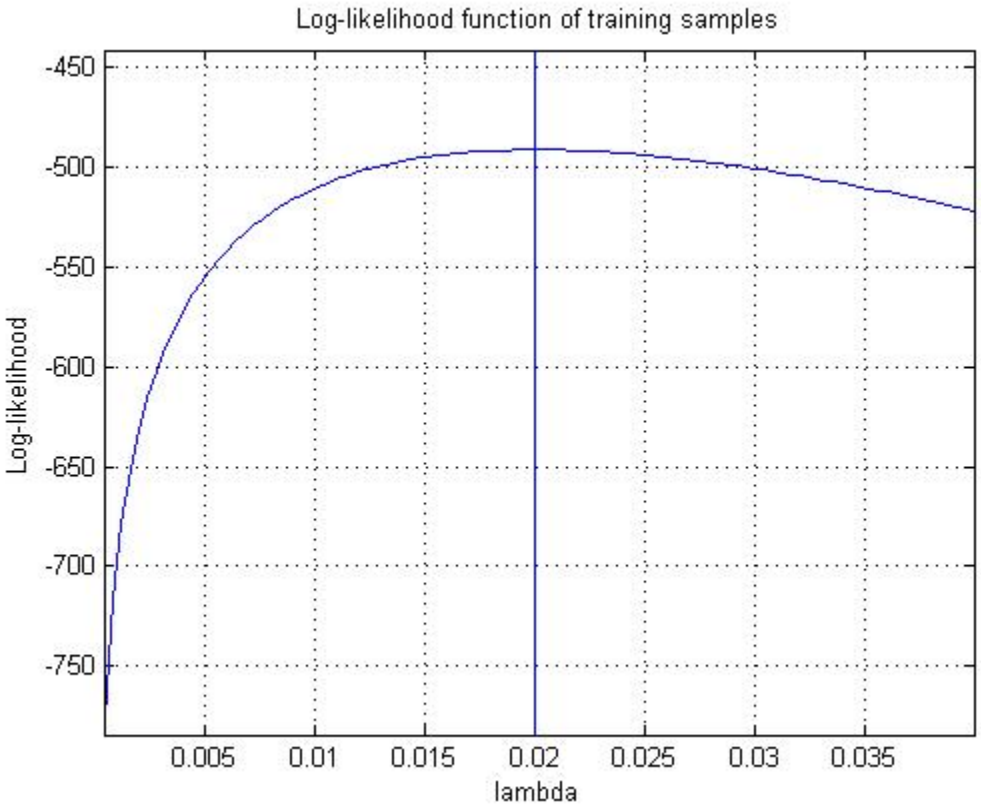
$$\ln L(\lambda) = \sum_{i=1}^{n} \ln\left(\frac{\lambda^{x_i}}{x_i!} e^{-\lambda}\right)$$

Decide for λ_{opt} with highest Log-likelihood.

3.2 Maximum-likelihood estimation

Another Example: ***Poisson distribution***

Practical validation:



3.2 Maximum-likelihood estimation

Another Example: ***Equal distribution***

$$p(x \mid \lambda) = \begin{cases} 1/\lambda & \text{for } x \in \{1,\ldots,\lambda\} \\ 0 & \text{for } x \notin \{1,\ldots,\lambda\} \end{cases}$$

with $x \in \mathbb{N}_+$ and $\lambda \in \mathbb{N}_+$

3.2 Maximum-likelihood estimation

Another Example: ***Equal distribution***

$$p(x \mid \lambda) = \begin{cases} 1/\lambda & \text{for } x \in \{1,\ldots,\lambda\} \\ 0 & \text{for } x \notin \{1,\ldots,\lambda\} \end{cases}$$

with $x \in \mathbb{N}_+$ and $\lambda \in \mathbb{N}_+$

Likelihood function:

$$L(\lambda) = \prod_{i=1}^{n} p(x_i \mid \lambda) = \begin{cases} 1/\lambda^n & \text{for } x_i \in \{1,\ldots,\lambda\} \text{ for all i} \\ 0 & \text{for } x_i \notin \{1,\ldots,\lambda\} \text{ for at least one i} \end{cases}$$

3.2 Maximum-likelihood estimation

Another Example: ***Equal distribution***

$$p(x \mid \lambda) = \begin{cases} 1/\lambda & \text{for } x \in \{1,\ldots,\lambda\} \\ 0 & \text{for } x \notin \{1,\ldots,\lambda\} \end{cases}$$

with $x \in \mathbb{N}_+$ and $\lambda \in \mathbb{N}_+$

Likelihood function:

$$L(\lambda) = \prod_{i=1}^{n} p(x_i \mid \lambda) = \begin{cases} 1/\lambda^n & \text{for } x_i \in \{1,\ldots,\lambda\} \text{ for all } i \\ 0 & \text{for } x_i \notin \{1,\ldots,\lambda\} \text{ for at least one } i \end{cases}$$

$$L(\lambda) = \begin{cases} 1/\lambda^n & \text{if } \max\{x_1, x_2, \ldots, x_n\} \le \lambda \\ 0 & \text{if } \max\{x_1, x_2, \ldots, x_n\} > \lambda \end{cases}$$

3.2 Maximum-likelihood estimation

Another Example: ***Equal distribution***

Maximization of

$$L(\lambda) = \begin{cases} 1/\lambda^n & \text{if } \max\{x_1, x_2, \ldots, x_n\} \le \lambda \\ 0 & \text{if } \max\{x_1, x_2, \ldots, x_n\} > \lambda \end{cases}$$

Apparently, L() is a strictly decreasing function in λ

(for $\max\{x_1, x_2, \ldots, x_n\} \le \lambda$).

Thus, λ must be chosen as small as possible **under the condition**

$$\max\{x_1, x_2, \ldots, x_n\} \le \lambda$$

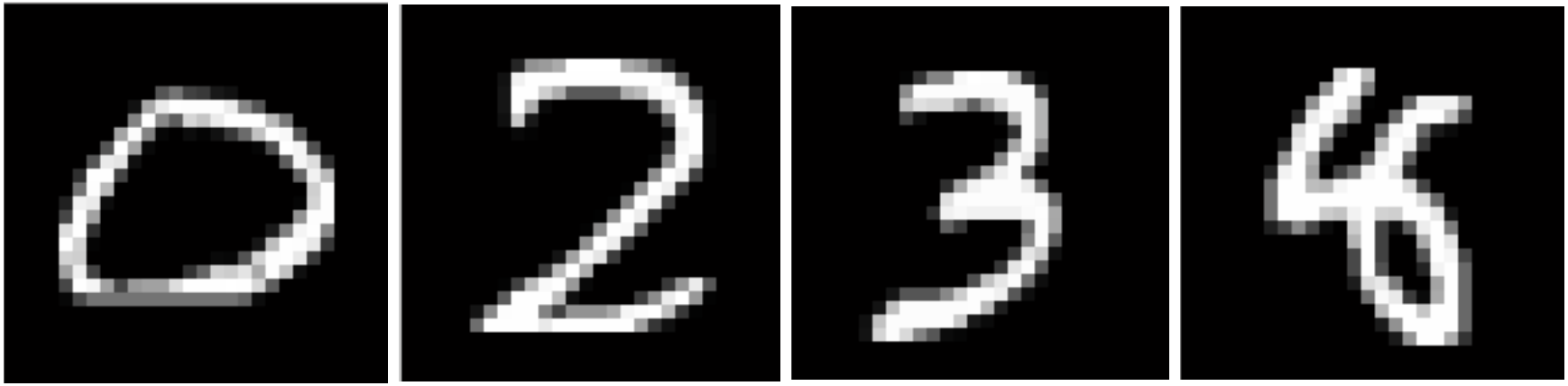
→ **ML estimation is given by** $\lambda = \max\{x_1, x_2, \ldots, x_n\}$

3.3 Practical example

Digit classfication with ML parameter estimation

Minimum Distance Classifier

Bayes Classifier
(full covariance matrix)



3.3 Practical example

Digit classfication with ML parameter estimation

Minimum Distance Classifier

$$\hat{\omega} = \arg\min_{\omega_i} \left[\sum_{d=1}^{D} \left(x_d - \hat{\mu}_{id} \right)^2 \right]$$

Bayes Classifier
(full covariance matrix)

$$\hat{\omega} = \arg\max_{\omega_i} \left[p(\mathrm{x} \mid \omega_i) \cdot P(\omega_i) \right]$$

What is a feature component?

→ gray value of one pixel

→ feature vector x contains up to 28 x 28 = 784 components

3.3 Practical example

Digit classfication with ML parameter estimation

Minimum Distance Classifier

$$\hat{\omega} = \arg\min_{\omega_i} \left[\sum_{d=1}^{D} \left(x_d - \hat{\mu}_{id} \right)^2 \right]$$

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k$$

Bayes Classifier
(full covariance matrix)

$$\hat{\omega} = \arg\max_{\omega_i} \left[p(\mathrm{x} \mid \omega_i) \cdot P(\omega_i) \right]$$

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k \qquad \hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{n} \left(x_k - \hat{\mu} \right) \cdot \left(x_k - \hat{\mu} \right)^t$$

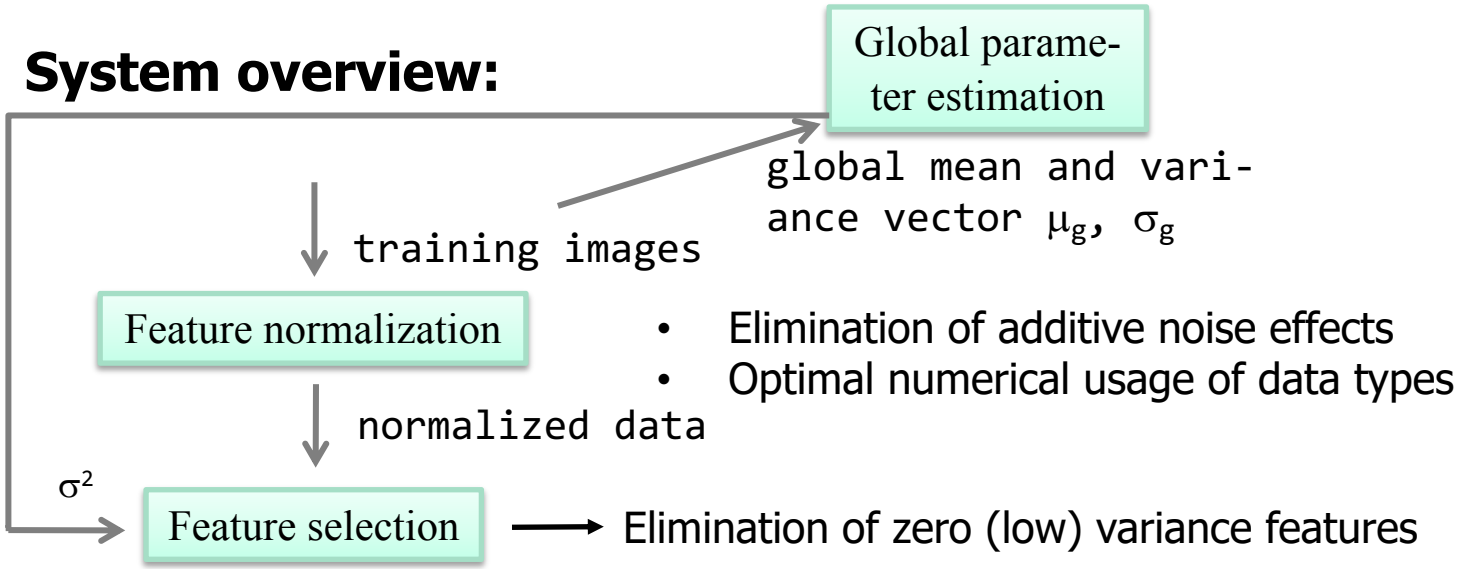
Matlab realization of digit classfication

Available in the OLAT material folder:

- Full script `classifier_0_to_9.m`
- Training and evaluation data

Matlab realization of digit classfication

System overview:

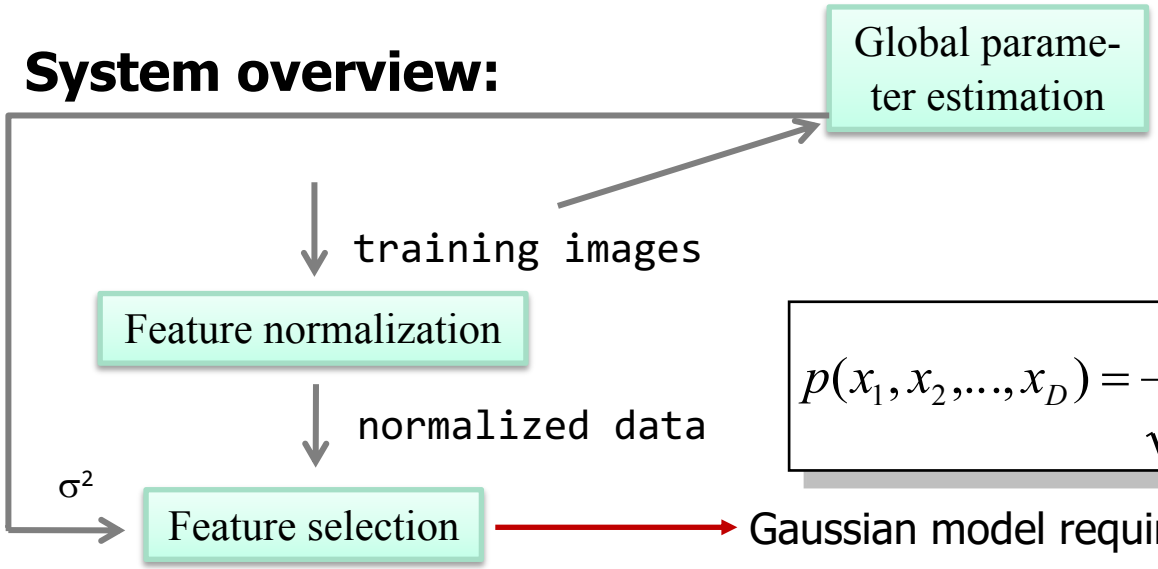


- Elimination of additive noise effects
- Optimal numerical usage of data types

→ Elimination of zero (low) variance features

Matlab realization of digit classfication

System overview:



$$p(x_1, x_2, \ldots, x_D) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma)}} \cdot e^{-\frac{1}{2}\cdot(x-\mu)^t \cdot \Sigma^{-1} \cdot (x-\mu)}$$

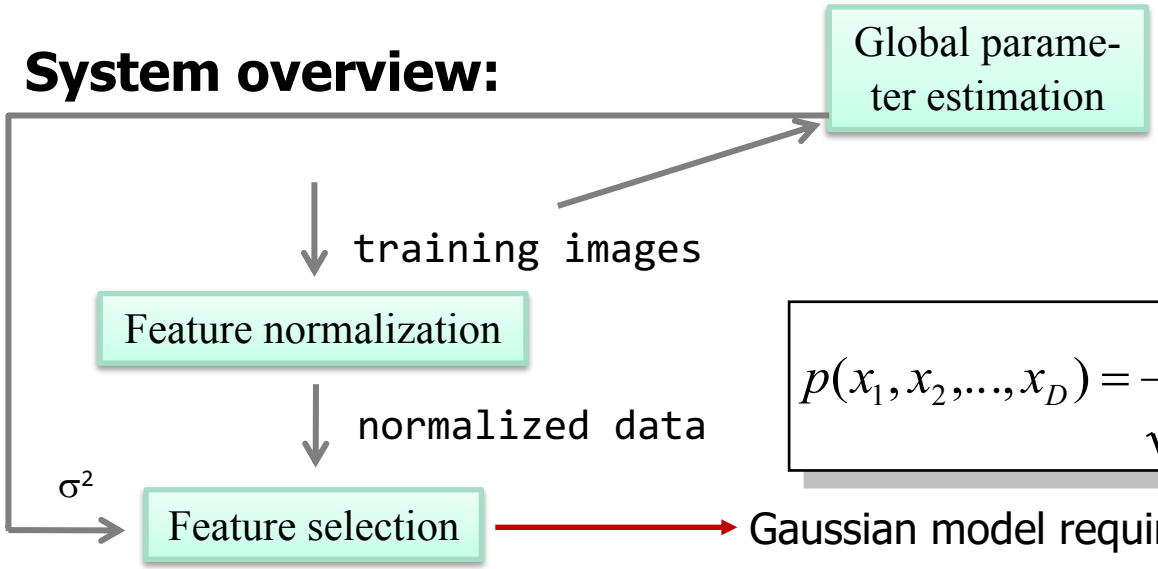
Gaussian model requires inversion of covariance matrix Σ. This inversion is only possible if matrix Σ is not singular (i.e. its determinant is not zero). Therefore we must eliminate features (pixels) with zero variance and covariances.

$$\sigma_{xy} = E[(x - \mu_x)(y - \mu_y)]$$

A pixel which is for example always black has zero variance and covariance.

Matlab realization of digit classfication

System overview:



$$p(x_1, x_2, \ldots, x_D) = \frac{1}{\sqrt{(2\pi)^D \cdot \det(\Sigma)}} \cdot e^{-\frac{1}{2}\cdot(x-\mu)^t \cdot \Sigma^{-1} \cdot (x-\mu)}$$

Gaussian model requires inversion of covariance matrix Σ. This inversion is only possible if matrix Σ is not singular (i.e. its determinant is not zero). Therefore we must eliminate features (pixels) with zero variance and covariances.

Example:

pixel variance

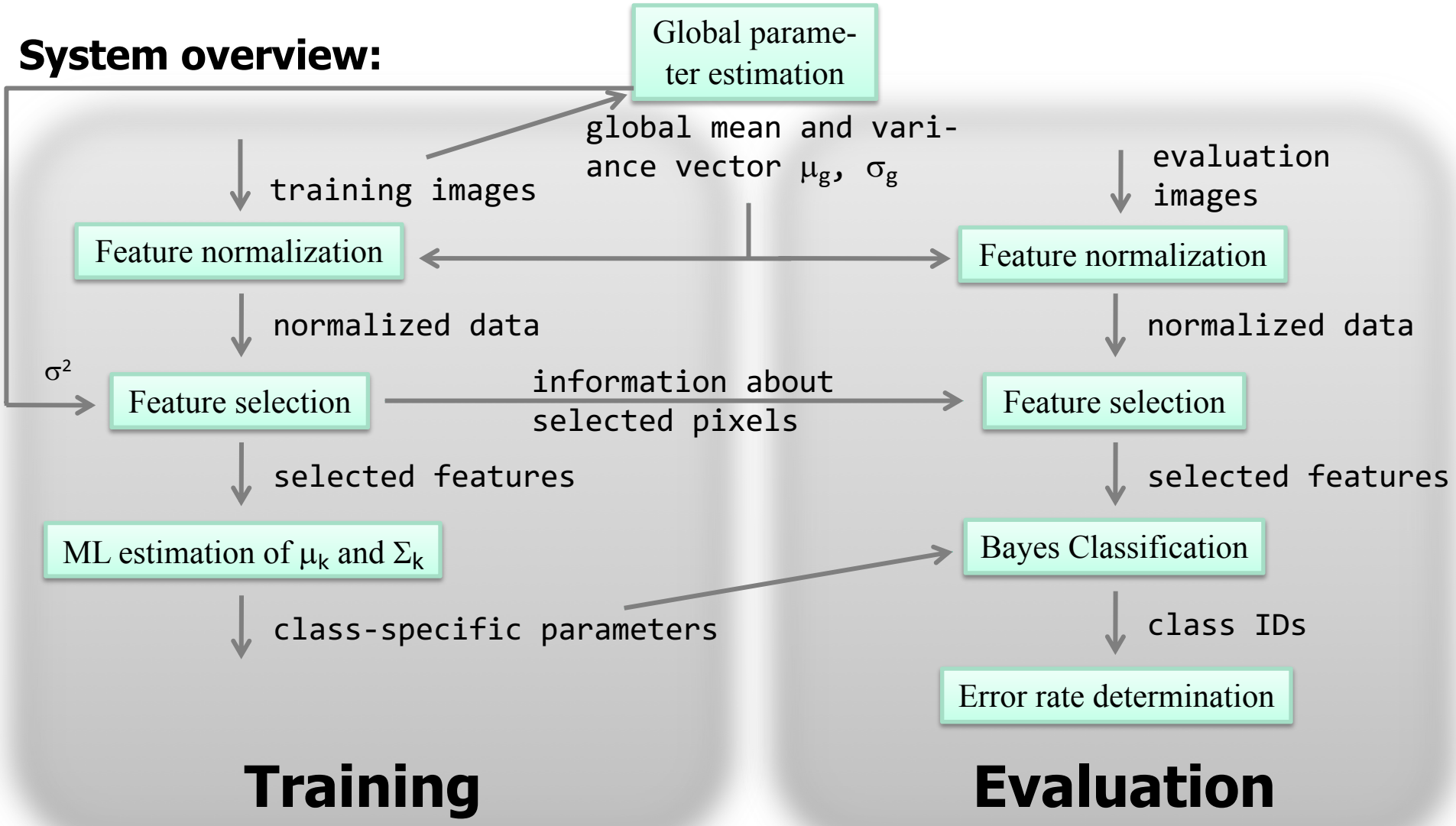
pixel covariance

```
A =
     0     0     0
     0     3     4
     0     4     4
```

det(A) = 0

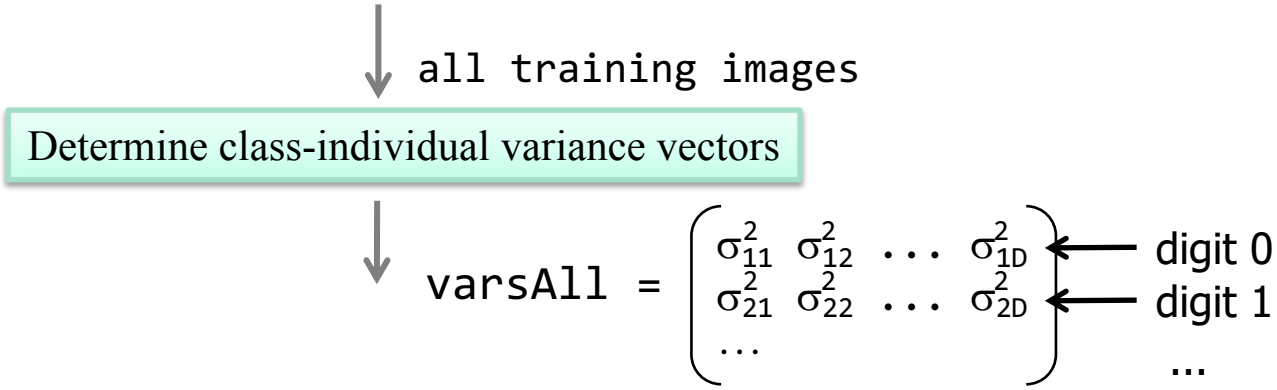
Matlab realization of digit classfication

System overview:



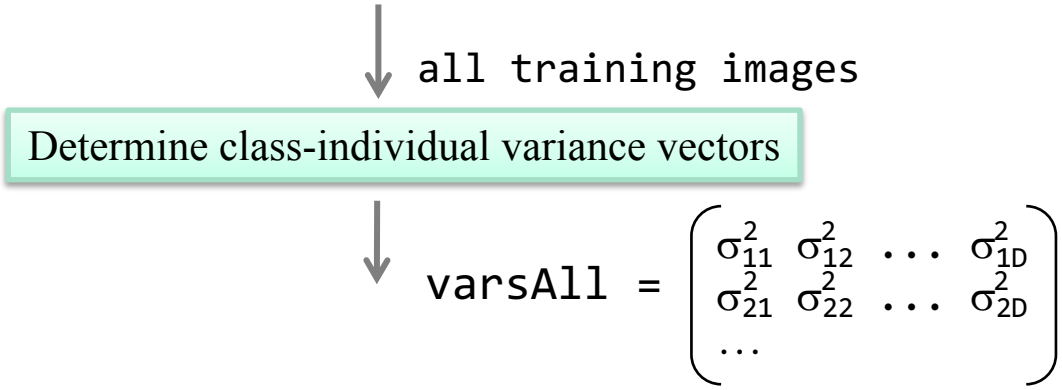
Matlab realization of digit classfication

Feature selection



Matlab realization of digit classfication

Feature selection – Matlab realization



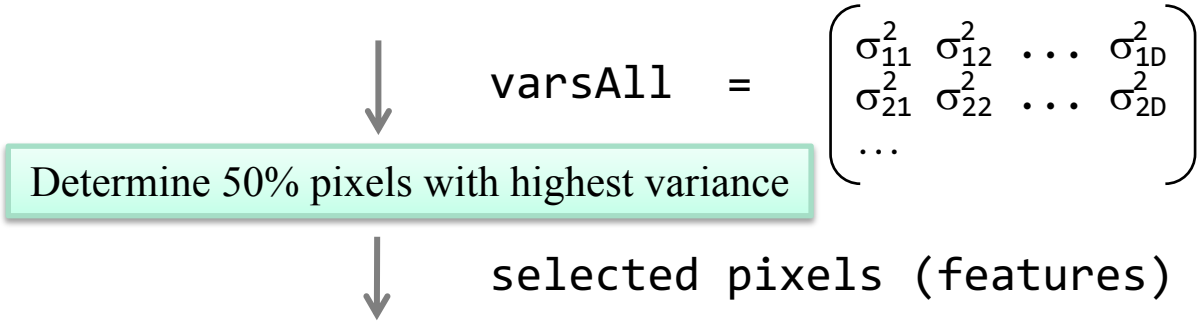
Determine class-individual variance vectors

```
varsAll = [];
for i = 0:9
    train_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (train_file);
    D = train_images; % renaming: training data is stored in matrix
    varsAll = [varsAll; var(D)];
end
```

row vector

Matlab realization of digit classfication

Feature selection



Matlab realization of digit classfication

Feature selection – Matlab realization

varsAll = $\begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1D}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2D}^2 \\ \dots \end{pmatrix}$

↓

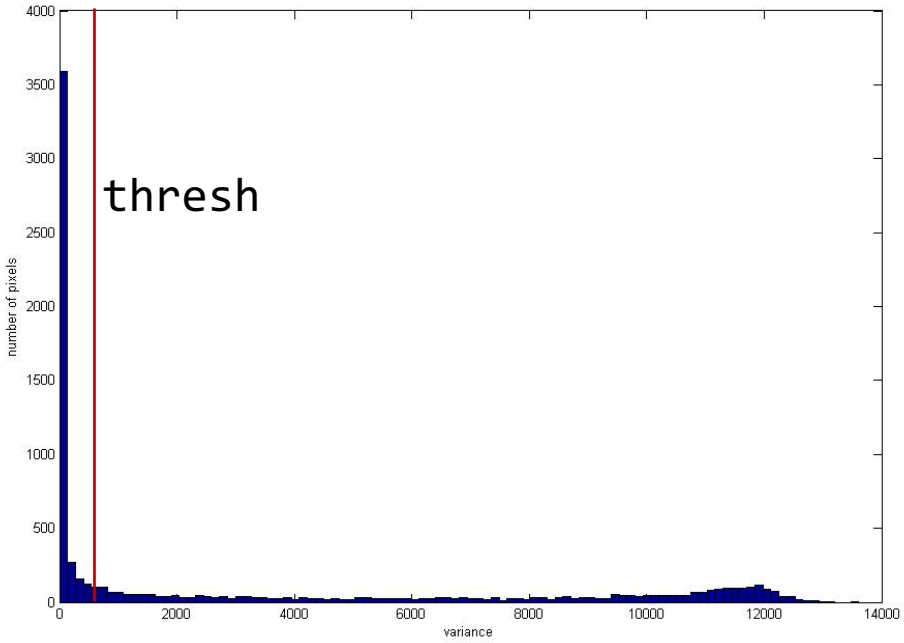
Determine 50% pixels with highest variance

↓

selected pixels (features)

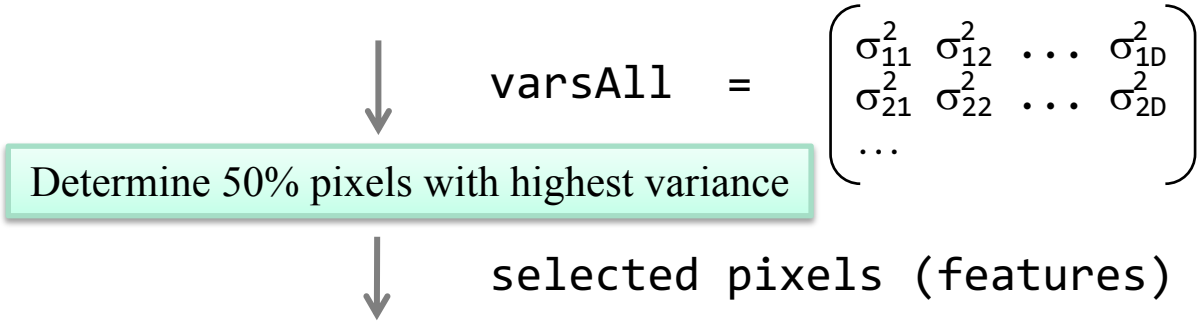
Determine threshold for the selection

```
thresh = quantile(varsAll(:), 0.5);
```



Matlab realization of digit classfication

Feature selection – Matlab realization



Variance of an active pixel must be larger than `thresh` in all 10 digits

```
non_zero_var_label = sum(varsAll > thresh) == 10;
```

Why?

Because we estimate individual Gaussians for each class (digit). Thus, we must be able to invert the class-specific Σ.

Matlab realization of digit classfication

Feature selection – Matlab realization

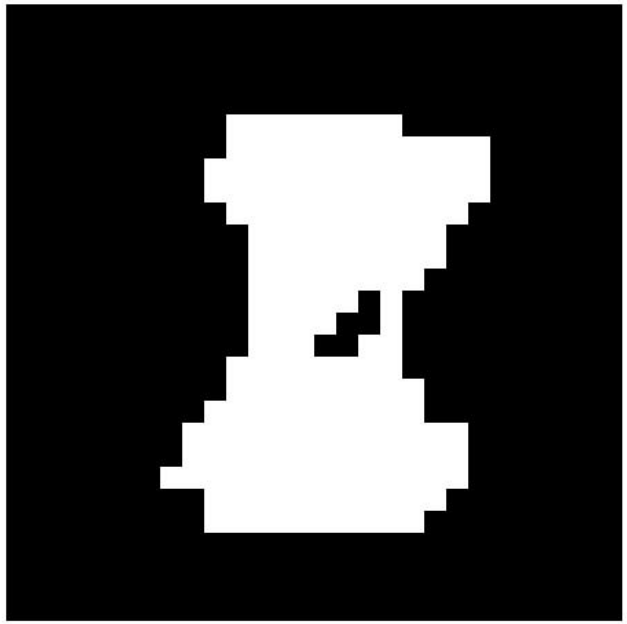
varsAll = $\begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1D}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2D}^2 \\ \dots & & & \end{pmatrix}$

Determine 50% pixels with highest variance

selected pixels (features)

Study the active pixels

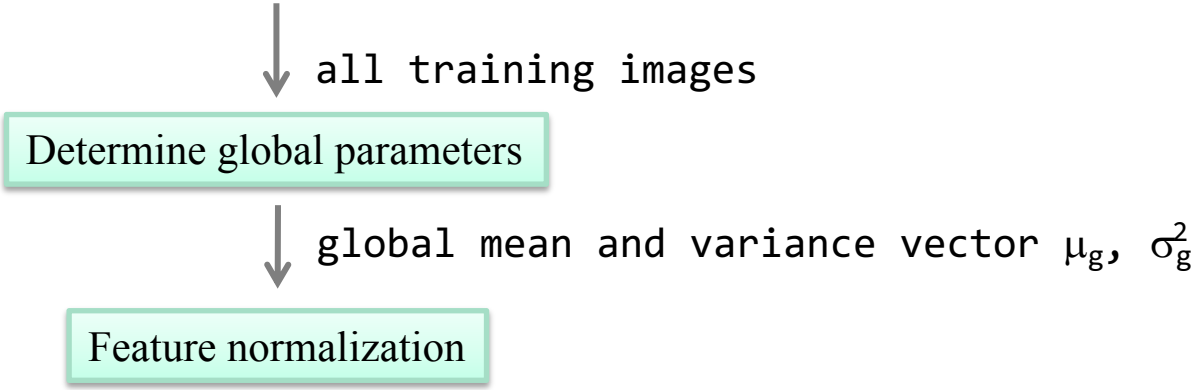
```
figure;
imshow(reshape(non_zero_var_label,28,28)',[]);
```



Only these pixels are used for classification

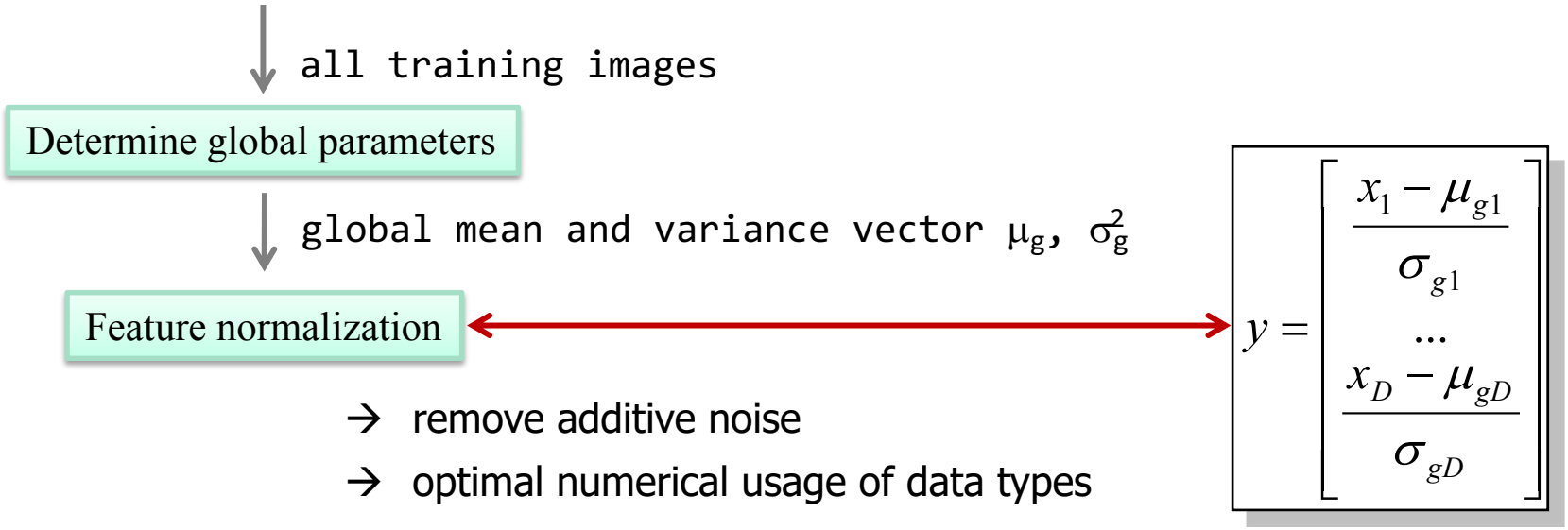
Matlab realization of digit classfication

Feature normalization



Matlab realization of digit classfication

Feature normalization



→ remove additive noise

→ optimal numerical usage of data types

→ target of each feature component:
mean 0, variance 1

Matlab realization of digit classfication

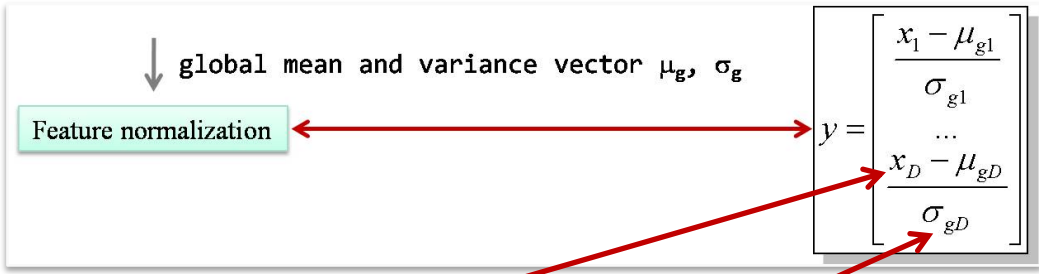
Feature normalization – Matlab realization

```
% Load all images into a common array
AllImages = [];
for i = 0:9
    train_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (train_file);
    D = train_images;                % just a renaming
    AllImages = [AllImages; D];      % store current images
end

% Compute global mean and variance
means = mean(AllImages);
vars  = var(AllImages);
```

Matlab realization of digit classfication

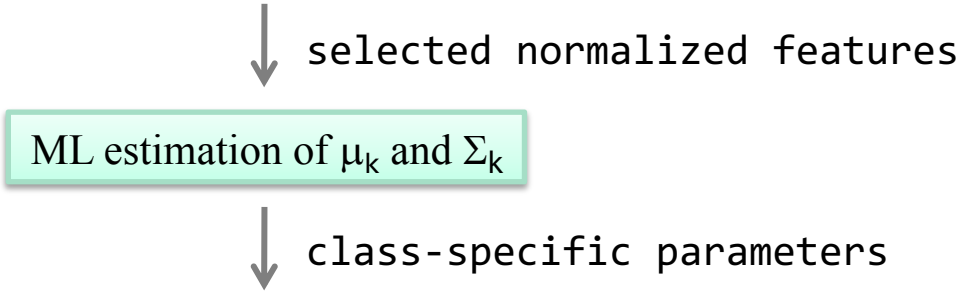
Feature normalization – Matlab realization

```
% Select the means and variances only for the active pixels.
means = means(non_zero_var_label);
vars  = vars(non_zero_var_label);

% Normalize the training data
for i = 0:9
    train_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (train_file);
    D = train_images;
    D = D(:,non_zero_var_label);


    % Normalization - Step 1: subtract mean
    D = D - repmat(means,[size(D,1) 1]);
    % Normalization - Step 2: divide by standard deviation
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]);
end
```



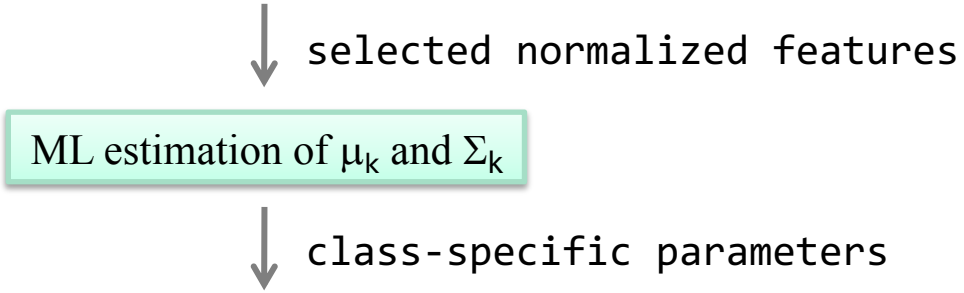
Matlab realization of digit classfication

Parameter estimation



Matlab realization of digit classfication

Parameter estimation – Matlab realization




```
for i = 0:9
    train_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (train_file), D = train_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    M{i+1} = mean(D);     % Store class-individual mean in a cell array
    C{i+1} = cov(D);      % Store      - " -     covariance matrix
end
```

Matlab realization of digit classfication

Evaluation on test images

```
err = [];                          % Error matrix
for i = 0:9                        % Process current test data set for digit i
    log_lik = [];
    eval_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (eval_file), D = eval_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    % Compute the negative log-likelihood P(x | class) for each digit.
    for j = 0:9
        new_ll = -log(mvnpdf(double(D), M{j+1}, C{j+1}));
        log_lik = [log_lik; new_ll'];
    end
```

We work with negative log-likelihood to prevent numerical errors due to handling of very small probabilities.

Matlab realization of digit classfication

Evaluation on test images

```
err = [];                          % Error matrix
for i = 0:9                        % Process current test data set for digit i
    log_lik = [];
    eval_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (eval_file), D = eval_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    % Compute the negative log-likelihood P(x | class) for each digit.
    for j = 0:9
        new_ll = -log(mvnpdf(double(D), M{j+1}, C{j+1}));
        log_lik = [log_lik; new_ll'];
    end
```

Column vector with negative log-likelihood of each test image.
size(new_ll) : [336 1] → 336 test images for this specific digit.

Matlab realization of digit classfication

Evaluation on test images

```
err = [];                          % Error matrix
for i = 0:9                        % Process current test data set for digit i
    log_lik = [];
    eval_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (eval_file), D = eval_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    % Compute the negative log-likelihood P(x | class) for each digit.
    for j = 0:9
        new_ll = -log(mvnpdf(double(D), M{j+1}, C{j+1}));
        log_lik = [log_lik; new_ll'];
    end

    % Decide for digit with lowest negative log-likelihood
    [min_ll, ind] = min(log_lik);
```

M x N matrix with likelihoods of the M=10 classes for each of the N test images.

Matlab realization of digit classfication

Evaluation on test images

```
err = [];                          % Error matrix
for i = 0:9                        % Process current test data set for digit i
    log_lik = [];
    eval_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (eval_file), D = eval_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    % Compute the negative log-likelihood P(x | class) for each digit.
    for j = 0:9
        new_ll = -log(mvnpdf(double(D), M{j+1}, C{j+1}));
        log_lik = [log_lik; new_ll'];
    end

    % Decide for digit with lowest negative log-likelihood
    [min_ll, ind] = min(log_lik);
```

1 x N row vector with indices of the best classes for each of the N test images

Matlab realization of digit classfication

Evaluation on test images

```
err = [];                          % Error matrix
for i = 0:9                        % Process current test data set for digit i
    log_lik = [];
    eval_file = ['D:\...\digit_' int2str(i) '.mat'];
    load (eval_file), D = eval_images;
    D = D(:,non_zero_var_label);                % Feature selection
    D = D - repmat(means,[size(D,1) 1]);        % Feature normalization
    D = D ./ repmat(sqrt(vars),[size(D,1) 1]); % Feature normalization

    % Compute the negative log-likelihood P(x | class) for each digit.
    for j = 0:9
        new_ll = -log(mvnpdf(double(D), M{j+1}, C{j+1}));
        log_lik = [log_lik; new_ll'];
    end

    % Decide for digit with lowest
    [min_ll, ind] = min(log_lik);
    err = [err; hist(ind,1:10)];
end
```

Add errors of digit i (number of confusions with other digits) as a new line to the error matrix

3.3 Practical example

Digit classfication with ML parameter estimation

Result with Bayes classifier with full covariance matrix:

Bayes Classifier with ML Parameter Estimation:

Error matrix

correct classifications

276	0	9	6	0	15	0	0	21	0
0	229	70	0	0	0	0	0	79	0
0	0	321	2	0	4	1	2	11	3
2	0	10	294	0	9	0	0	21	1
0	0	17	2	241	0	0	6	58	3
2	0	1	6	0	276	3	0	8	1
1	0	7	1	0	3	296	0	11	0
1	0	12	0	0	1	0	311	16	2
0	0	2	1	0	6	0	0	316	0
0	0	2	2	1	4	0	32	34	261

Overall digit error rate: 15.36

Remark: Only 'active' pixels are used for classification

3.3 Practical example

Digit classfication with ML parameter estimation

For comparison: Result with distance classifier (see Chapter 2)

Distance Classifier with ML parameter estimation

Error matrix:

302	0	4	1	3	9	5	0	3	0
0	313	59	0	1	0	0	0	5	0
12	5	266	1	25	0	9	7	15	4
1	2	29	251	0	13	0	6	34	1
0	7	1	0	283	0	9	0	13	14
4	47	4	5	11	193	11	14	6	2
5	2	4	1	4	4	298	0	1	0
0	1	8	0	3	0	0	322	8	1
5	11	4	8	2	13	6	5	268	3
2	11	0	0	38	3	0	47	16	219

Overall digit error rate: 18.54

Remark: All pixels are used for classification

3.4 Bayes Learning

Goal:

Estimation of parameter vector θ_k of class k

given training data $D_k = x_{1k}, x_{2k}, \ldots, x_{nk}$

Assumption:
samples are independent and identically distributed (i.i.d.)

Assumption: Separation of training data into K class specific subsets $D_1, D_2 \ldots, D_K$ whereas data in D_i has no influence on estimation of parameters of class j, $i \neq j$.

→ Omit the class dependency for simplification and just write

$D = x_1, x_2, \ldots, x_n$ and θ

3.4 Bayes Learning

Reminder: **Maximum-likelihood estimate** of θ given $D = x_1, x_2, \ldots, x_n$ is:

$$\hat{\Theta} = \arg\max_{\Theta} p(D|\Theta)$$

$$\hat{\Theta} = \arg\max_{\Theta} \prod_{k=1}^{n} p(x_k|\Theta)$$

Assumption here: ***True*** parameter vector is unknown but ***fixed.***

3.4 Bayes Learning

ML estimate: $\hat{\Theta} = \arg\max_{\Theta} p(D|\Theta)$

Concept of **Bayes Learning:**

- Parameter vector θ is a ***random variable*** with a-priori distribution $p(\theta)$
- Distribution $p(\theta)$ is known to us due to previous measurements or knowledge about the task (usually also acquired from experiments)

  → example: see below

Step 1: Determination of a-posteriori distribution of parameter θ

$$p(\Theta|x_1, x_2, ..., x_n) = p(\Theta|D)$$

Step 2: Maximization

$$\hat{\Theta} = \arg\max_{\Theta} p(\Theta|D)$$

3.4 Bayes Learning

Calculation of the a-posteriori distribution

$$p(\Theta|D) = \frac{p(D|\Theta) \cdot p(\Theta)}{p(D)} = \frac{p(D|\Theta) \cdot p(\Theta)}{\int_{\Theta'} p(D, \Theta') d\Theta'}$$

$$p(\Theta|D) = \frac{p(D|\Theta) \cdot p(\Theta)}{\int_{\Theta'} p(D|\Theta') p(\Theta') d\Theta'}$$

Constant with respect to θ

$$p(\Theta|D) = const(\Theta) \cdot p(\Theta) \cdot p(D|\Theta)$$

$$p(\Theta|D) = const(\Theta) \cdot p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta)$$

3.4 Bayes Learning

Optimization over parameter θ

$$\hat{\Theta} = \arg\max_{\Theta} \left\{ const(\Theta) \cdot p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

$$\hat{\Theta} = \arg\max_{\Theta} \left\{ p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

MAP parameter estimation

3.4 Bayes Learning

Comparison with ML estimation

MAP
$$\hat{\Theta} = \arg\max_{\Theta} \left\{ p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

ML
$$\hat{\Theta} = \arg\max_{\Theta} \left\{ \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

Bayesian parameter estimation

- makes use of additional knowledge source: prior parameter distribution
- takes number of training samples into account:

  more samples → higher influence of training data

3.4 Bayes Learning

Bayesian parameter estimation

- makes use of additional knowledge source: prior parameter distribution
- takes number of training samples into account:

  more samples → higher influence of training data

→ Use Bayes Learning if, in addition to the training data, something is known about the parameter (vector) θ which can be used to formulate an a-priori distribution $p(\theta)$.

3.4 Bayes Learning

Practical application

MAP speaker adaptation in automatic speech recognition

- Speech recognition: pattern classification problem

  classes: phonemes or words

  features: spectral power distribution, e.g. of 10ms frames

- Inventory of class-conditioned distributions → acoustic model
- Acoustic model is usually trained speaker-independently
- Much better results if acoustic model is ***adapted*** to specific user

  → Use Bayes parameter estimation

3.4 Bayes Learning

Example

Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(x|\mu) = N(x; \mu, \sigma^2)$$

Assumptions:

- Variance σ^2 is known
- A-priori distribution $p(\Theta) = p(\mu)$ is also a Gaussian distribution:

$$p(\mu|\mu_0, \sigma_0^2)$$

Searched-for mean of the Gaussian distribution is also normally distributed.

The parameters of this second distribution μ_0, σ_0^2 are called ***Hyperparameters***. They must be known, e.g. from experiments.

3.4 Bayes Learning

Example

Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

Assumptions:

- Variance σ^2 is known
- A-priori distribution $p(\Theta) = p(\mu)$ is also a Gaussian distribution:

$$p(\mu|\mu_0, \sigma_0^2)$$

Describes our uncertainty

Describes our best knowledge about the unknown parameter μ ***before*** any training data has been observed.

3.4 Bayes Learning

Example

Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

Assumptions:

- Variance σ^2 is known
- A-priori distribution $p(\Theta) = p(\mu)$ is also a Gaussian distribution:

$$p(\mu|\mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2\right\}$$

This equation describes our best knowledge before we have observed the (new) training data.

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

After observing the training data $D = x_1, x_2, \ldots, x_n$ we have more knowledge:

$$p(\mu) \rightarrow p(\mu|D)$$

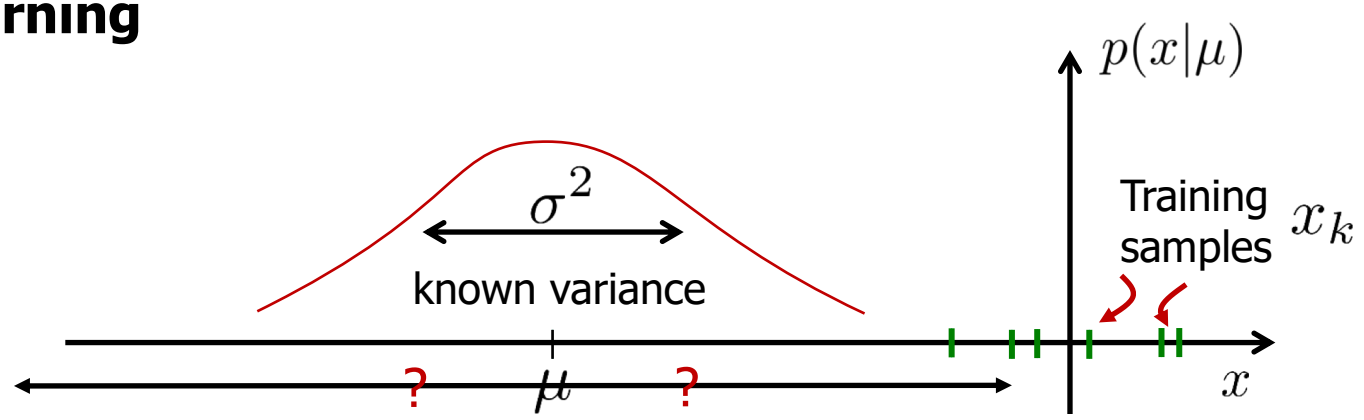
Therefore, we may update the distribution using Bayes rule:

$$\begin{aligned} p(\mu|D) &= \frac{p(D|\mu)p(\mu)}{p(D)} \\ &= \frac{p(D|\mu)p(\mu)}{\int_{\mu'} p(D|\mu')p(\mu'))d\mu'} \\ &= const(\mu) \cdot p(\mu) \cdot p(D|\mu) \end{aligned}$$

Independent of μ
(but dependent on D)

3.4 Bayes Learning

searched-for distribution



3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const(\mu) \cdot p(\mu) \cdot p(D|\mu)$$

i.i.d.

$$p(D|\mu) = \prod_{k=1}^{n} p(x_k|\mu)$$

$$\rightarrow \quad p(\mu|D) = const(\mu) \cdot p(\mu) \cdot \prod_{k=1}^{n} p(x_k|\mu)$$

$$p(x|\mu) = N(x; \mu, \sigma^2)$$

$$p(x_k|\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

$$\rightarrow \quad p(\mu|D) = const(\mu) \cdot p(\mu) \cdot \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const(\mu) \cdot p(\mu) \cdot \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

Independent of k and μ

$$\rightarrow \quad p(\mu|D) = const'(\mu) \cdot p(\mu) \cdot \prod_{k=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const'(\mu) \cdot p(\mu) \cdot \prod_{k=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

$$p(\mu|\mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2\right\}$$

Independent of μ

$$= const'(\mu) \cdot \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2\right\} \cdot \prod_{k=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right\}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const''(\mu) \cdot \exp\left\{-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right\} \cdot \prod_{k=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_k-\mu}{\sigma}\right)^2\right\}$$

$$\prod_{k=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_k-\mu}{\sigma}\right)^2\right\} = \exp\left\{-\frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k-\mu}{\sigma}\right)^2\right\}$$

$$p(\mu|D) = const''(\mu) \cdot \exp\left\{-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right\} \cdot \exp\left\{-\frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k-\mu}{\sigma}\right)^2\right\}$$

$$p(\mu|D) = const''(\mu) \cdot \exp\left\{-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2 - \frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k-\mu}{\sigma}\right)^2\right\}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const''(\mu) \cdot \exp\left\{ -\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2 - \frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k - \mu}{\sigma}\right)^2 \right\}$$

The hyperparameters μ_0, σ_0^2 influence the resulting distribution similar to the pairs x_k, σ^2

→ Mixture of prior knowledge and new observations

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const''(\mu) \cdot \exp\left\{ -\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2 - \frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k - \mu}{\sigma}\right)^2 \right\}$$

$$p(\mu|D) = const''(\mu)\cdot\exp\left\{ -\frac{1}{2}\left[\frac{\mu^2}{\sigma_0^2} - \frac{2\mu\mu_0}{\sigma_0^2} + \frac{\mu_0^2}{\sigma_0^2} + \frac{n\mu^2}{\sigma^2} - \frac{2\mu\sum_{k=1}^{n} x_k}{\sigma^2} + \frac{\sum_{k=1}^{n} x_k^2}{\sigma^2}\right]\right\}$$

$$p(\mu|D) = const''(\mu)\cdot\exp\left\{ -\frac{1}{2}\left[\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)\mu^2 - 2\left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=1}^{n} x_k\right)\mu + \frac{\mu_0^2}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=}^{n} x_k^2\right]\right\}$$

Independent of μ

$$p(\mu|D) = const'''(\mu) \cdot \exp\left\{ -\frac{1}{2}\left[\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)\mu^2 - 2\mu\left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=1}^{n} x_k\right)\right]\right\}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const'''(\mu) \cdot \exp\left\{-\frac{1}{2}\left[\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)\mu^2 - 2\left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=1}^{n} x_k\right)\mu\right]\right\}$$

In the document ProductOfGaussians.pdf (OLAT Material Folder -> Additonal Reading) it is shown that the product of n Gaussians is again a Gaussian.

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

$$p(\mu|D) = const'''(\mu) \cdot \exp\left\{-\frac{1}{2}\left[\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)\mu^2 - 2\left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=1}^{n} x_k\right)\mu\right]\right\}$$

In the document ProductOfGaussians.pdf (OLAT Material Folder -> Additonal Reading) it is shown that the product of n Gaussians is again a Gaussian.

$$\begin{aligned} p(\mu|D) &:= \frac{1}{\sqrt{2\pi\sigma_N^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu - \mu_N}{\sigma_N}\right)^2\right\} \\ &= \frac{1}{\sqrt{2\pi\sigma_N^2}} \exp\left\{-\frac{1}{2}\left[\frac{\mu^2}{\sigma_N^2} - 2\frac{\mu_N}{\sigma_N^2}\mu + \frac{\mu_N^2}{\sigma_N^2}\right]\right\} \\ &= const''''(\mu) \exp\left\{-\frac{1}{2}\left[\frac{1}{\sigma_N^2}\mu^2 - 2\frac{\mu_N}{\sigma_N^2}\mu\right]\right\} \end{aligned}$$

3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

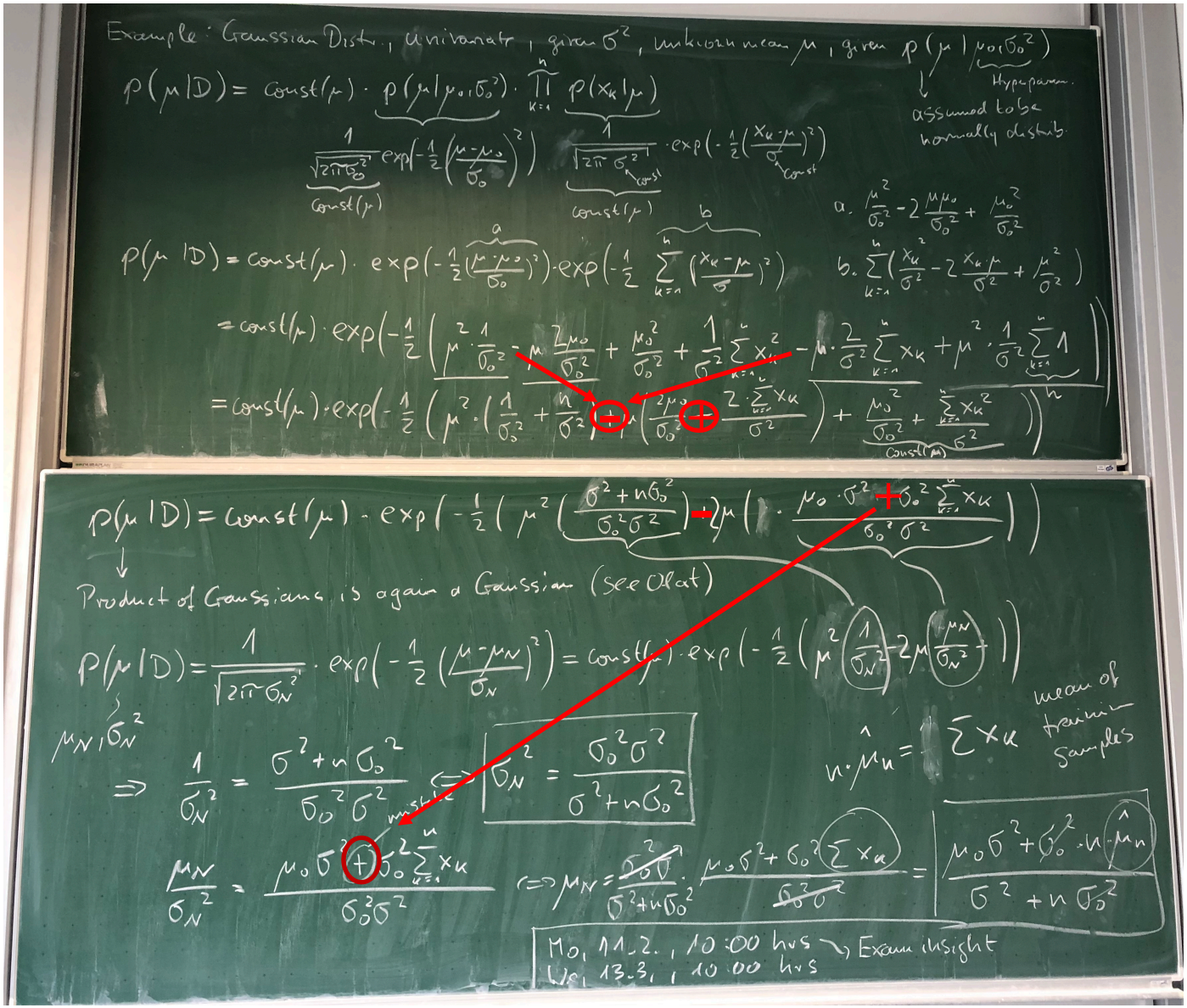
$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}$$

$$\frac{\mu_N}{\sigma_N^2} = \frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{k=1}^{n} x_k$$

$$\frac{\mu_N}{\sigma_N^2} = \frac{\mu_0}{\sigma_0^2} + \frac{n}{\sigma^2}\hat{\mu}_n \qquad \text{with} \quad \hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} x_k$$

***empirical* mean**

Mistake in last lecture:



3.4 Bayes Learning

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

Solving for μ_N and σ_N^2 gives

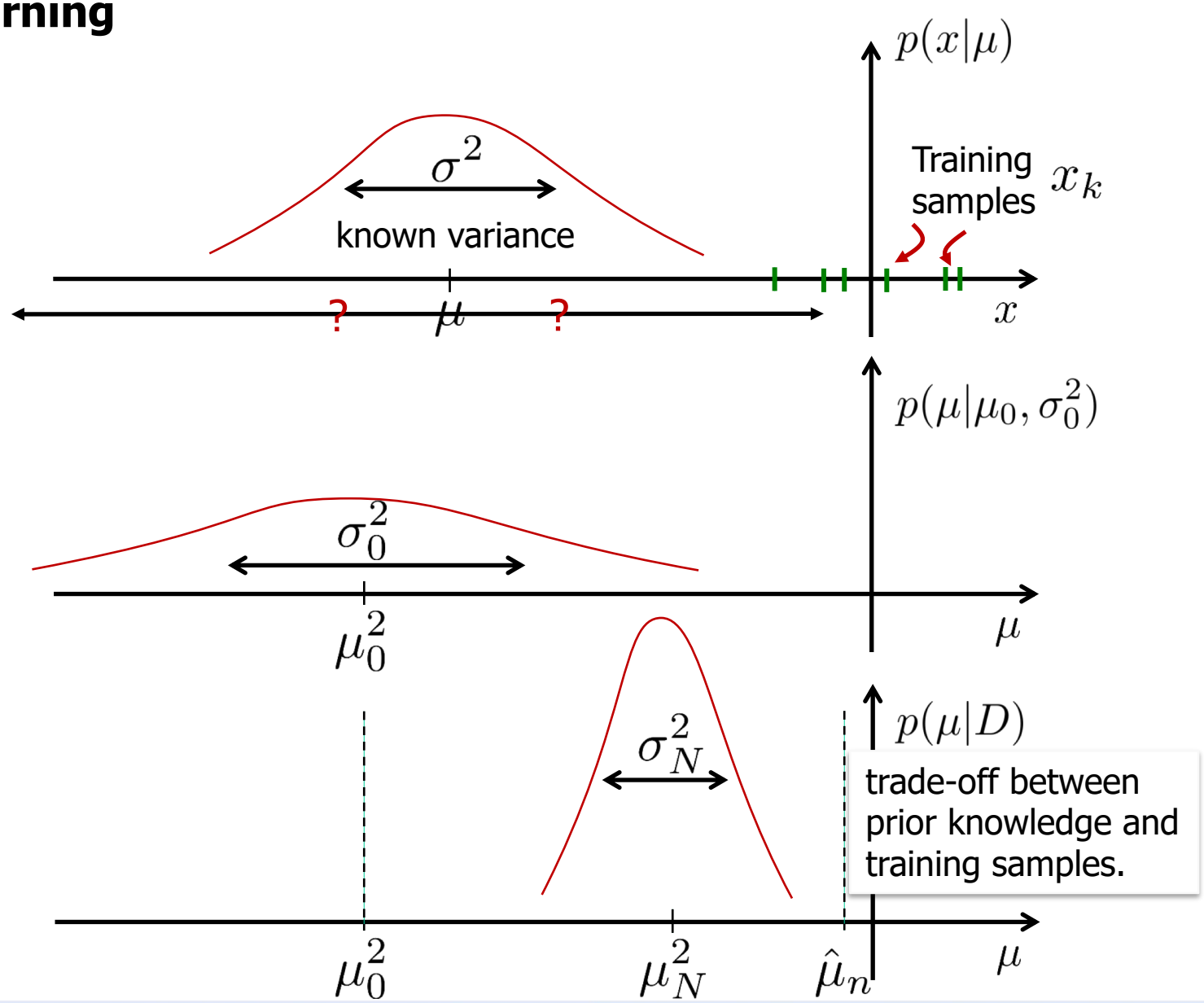
$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

Observations:

- Larger n → smaller influence of a-priori information
- n → infinity → $\mu_N \to \hat{\mu}_n$ (ML estimation) and $\sigma_N^2 \to 0$

3.4 Bayes Learning



3.4 Bayes Learning

Practical application

MAP speaker adaptation in automatic speech recognition

- Speech recognition: pattern classification problem

  classes: phonemes or words

  features: spectral power distribution, e.g. of 10ms frames

- Inventory of class-conditioned distributions → acoustic model
- Acoustic model is usually trained speaker-independently
- Much better results if acoustic model is ***adapted*** to specific user

  → Use Bayes parameter estimation

3.4 Bayes Learning

Practical application

- Hyperparameter μ_0, σ_0^2 are given by the means and variances of the speaker-independent model. They define the a-priori Gaussian distribution.
- The new data, coming from the target speaker, constitute the training corpus D.
- The new adapted parameters result from reestimation equations, similar to

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2 + \sigma^2}$$

The general case (multivariate Gaussian, unknown mean vector and full covariance matrix) is more difficult.

For details:

K. Fukunaga. Introduction to Statistical Pattern Recognition. Computer Science and Scientific Computing, Academic Press, 1990.

3.4 Bayes Learning

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

Wrap-up:

Optimal decision based on MAP decision rule

Required: Prior probability and likelihood

Estimation of likelihood:

Parameterization and estimation of parameters based on

- Maximum-likelihood parameter estimation
- Maximum A-Posteriori (MAP) parameter estimation

3.4 Bayes Learning

MAP

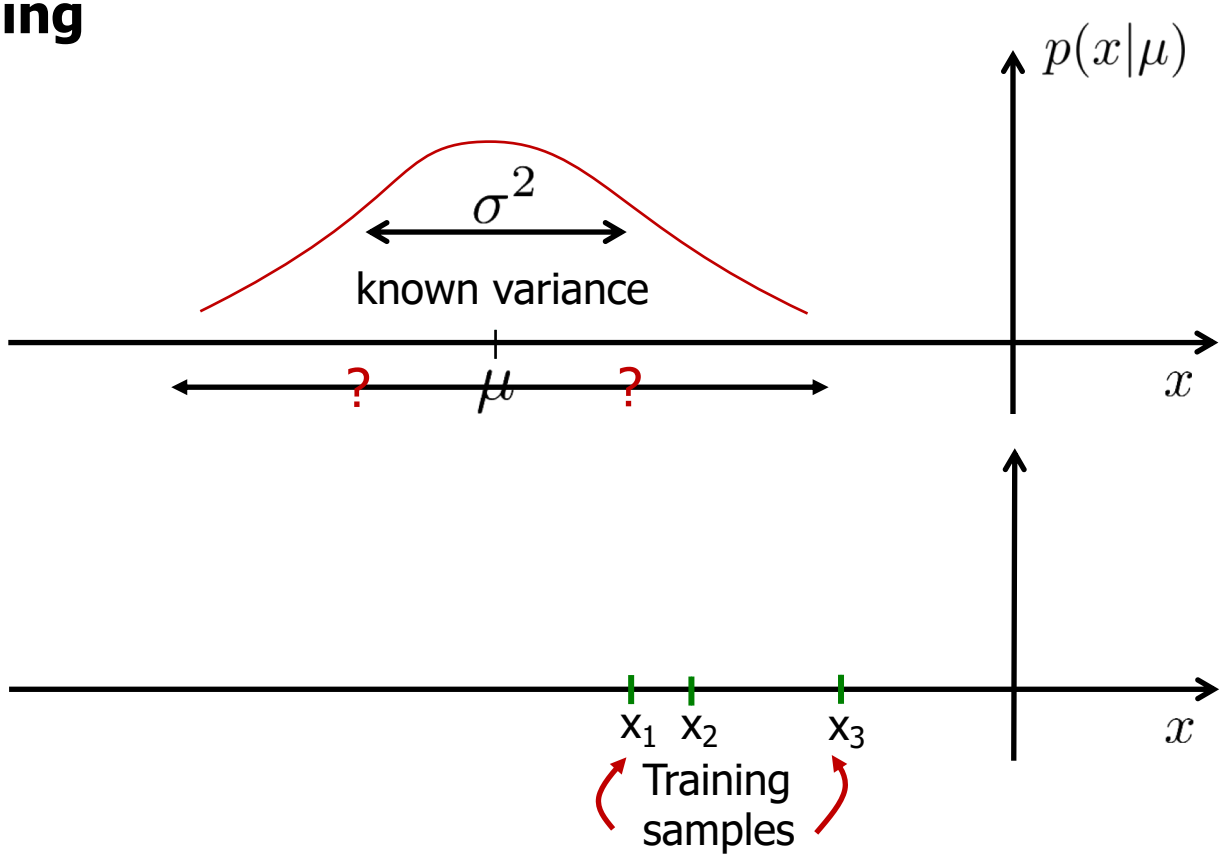
$$\hat{\Theta} = \arg\max_{\Theta} \left\{ p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

ML

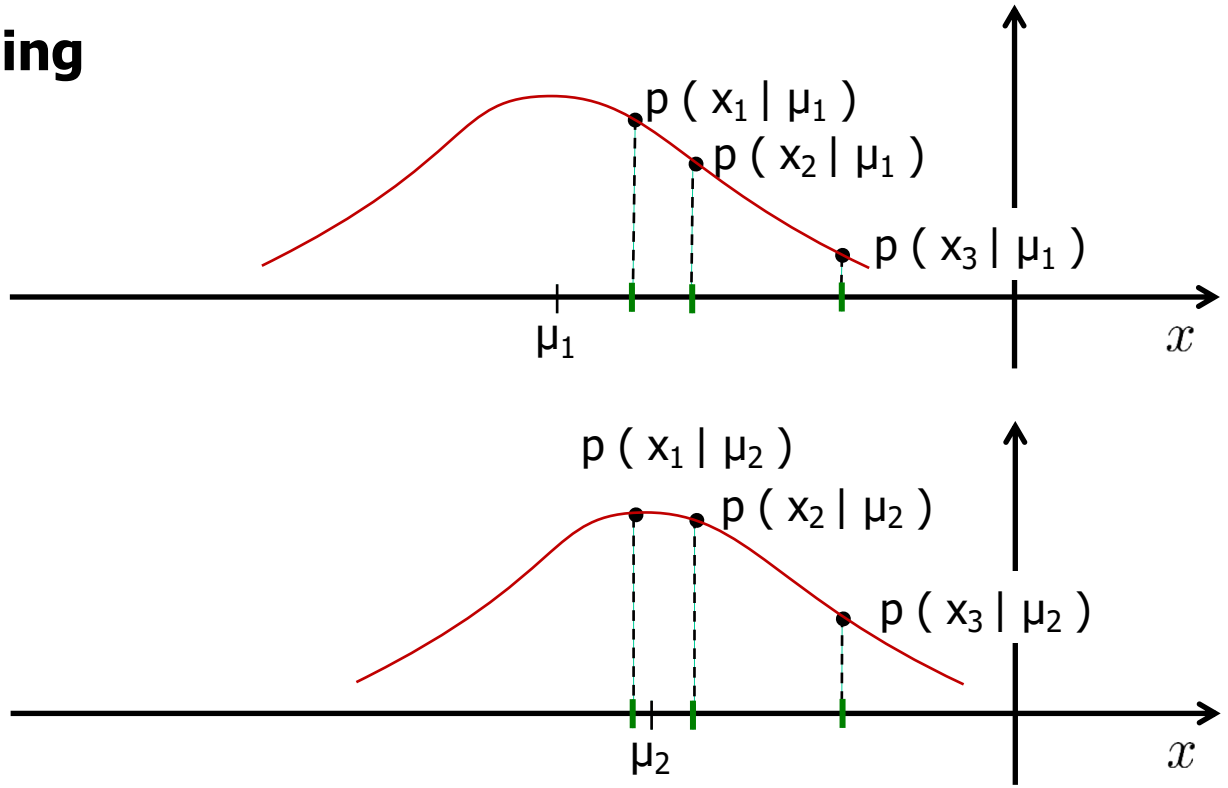
$$\hat{\Theta} = \arg\max_{\Theta} \left\{ \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

3.4 Bayes Learning

searched-for distribution



3.4 Bayes Learning

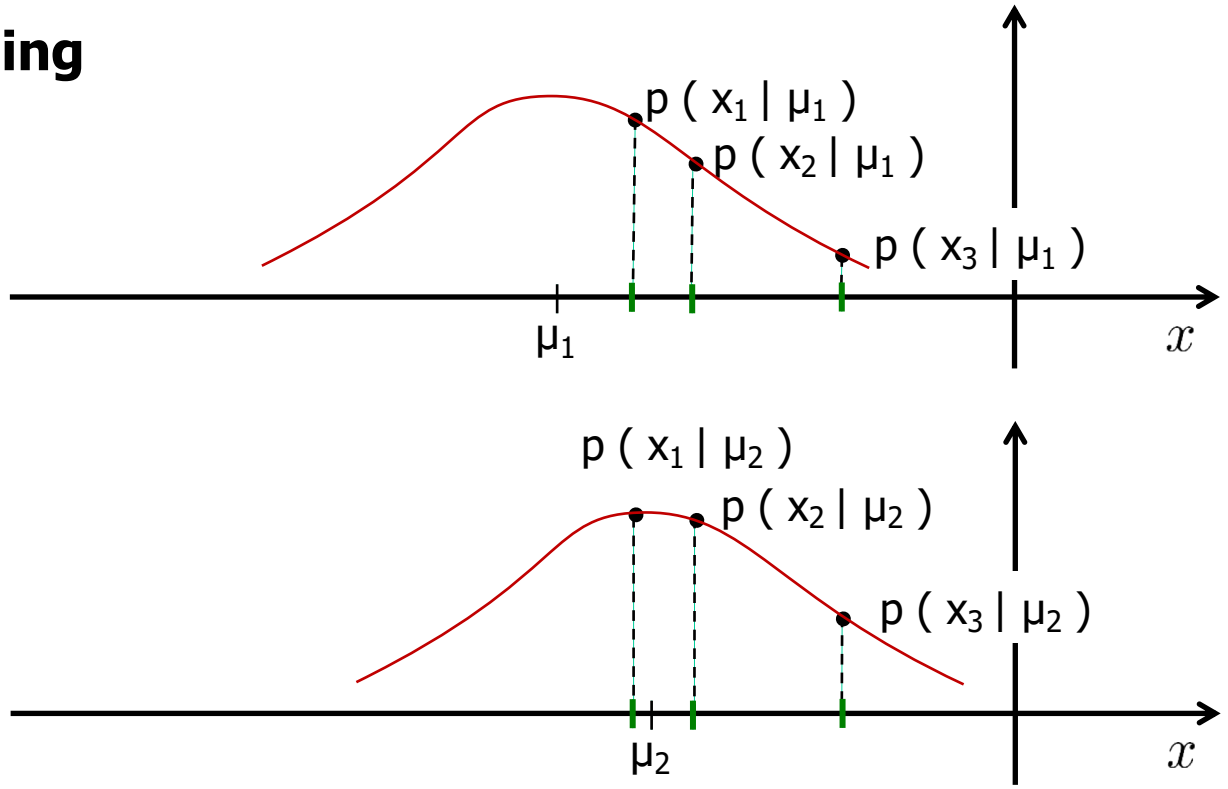


Decide for parameter μ which provides the largest product of individual training data likelihoods.

ML

$$\hat{\Theta} = \arg\max_{\Theta} \left\{ \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

3.4 Bayes Learning

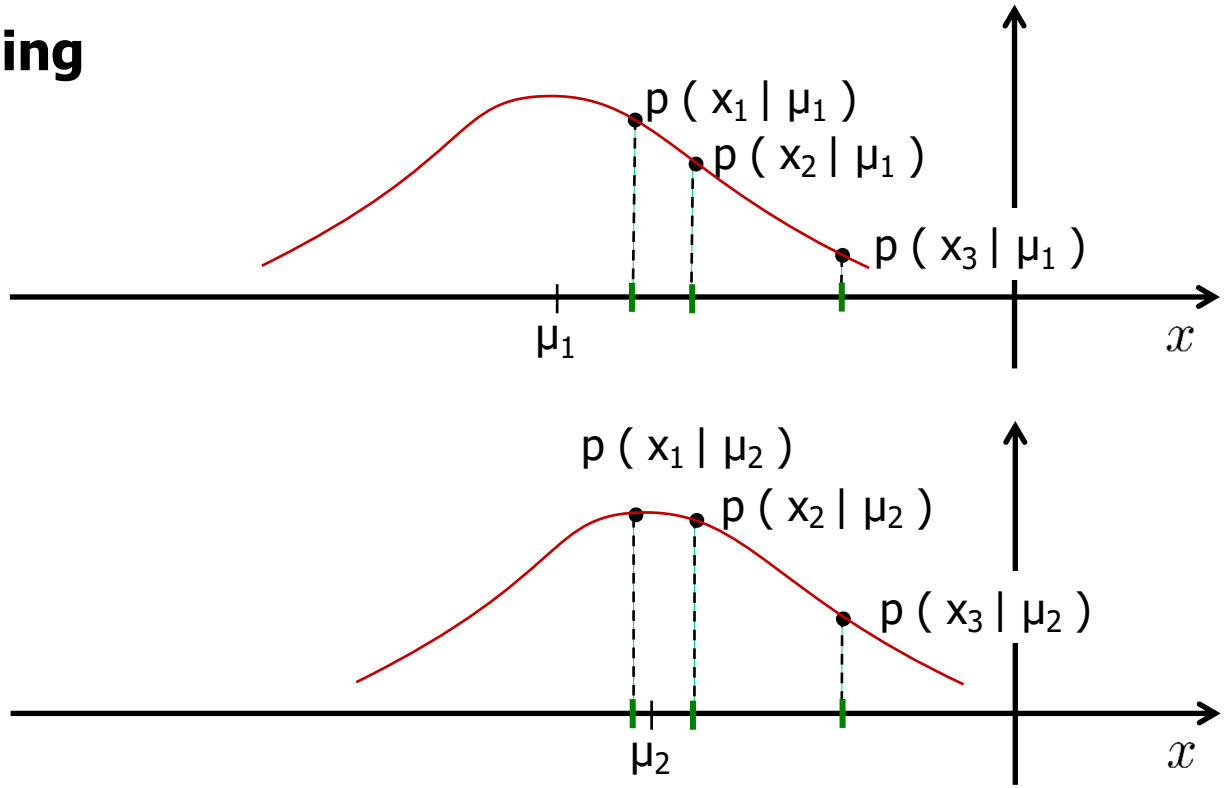


Decide for parameter μ which provides the largest product of individual training data likelihoods.

?

$$p(x_1 \mid \mu_1) \cdot p(x_2 \mid \mu_1) \cdot p(x_3 \mid \mu_1) \;\lesseqgtr\; p(x_1 \mid \mu_2) \cdot p(x_2 \mid \mu_2) \cdot p(x_3 \mid \mu_2)$$

3.4 Bayes Learning

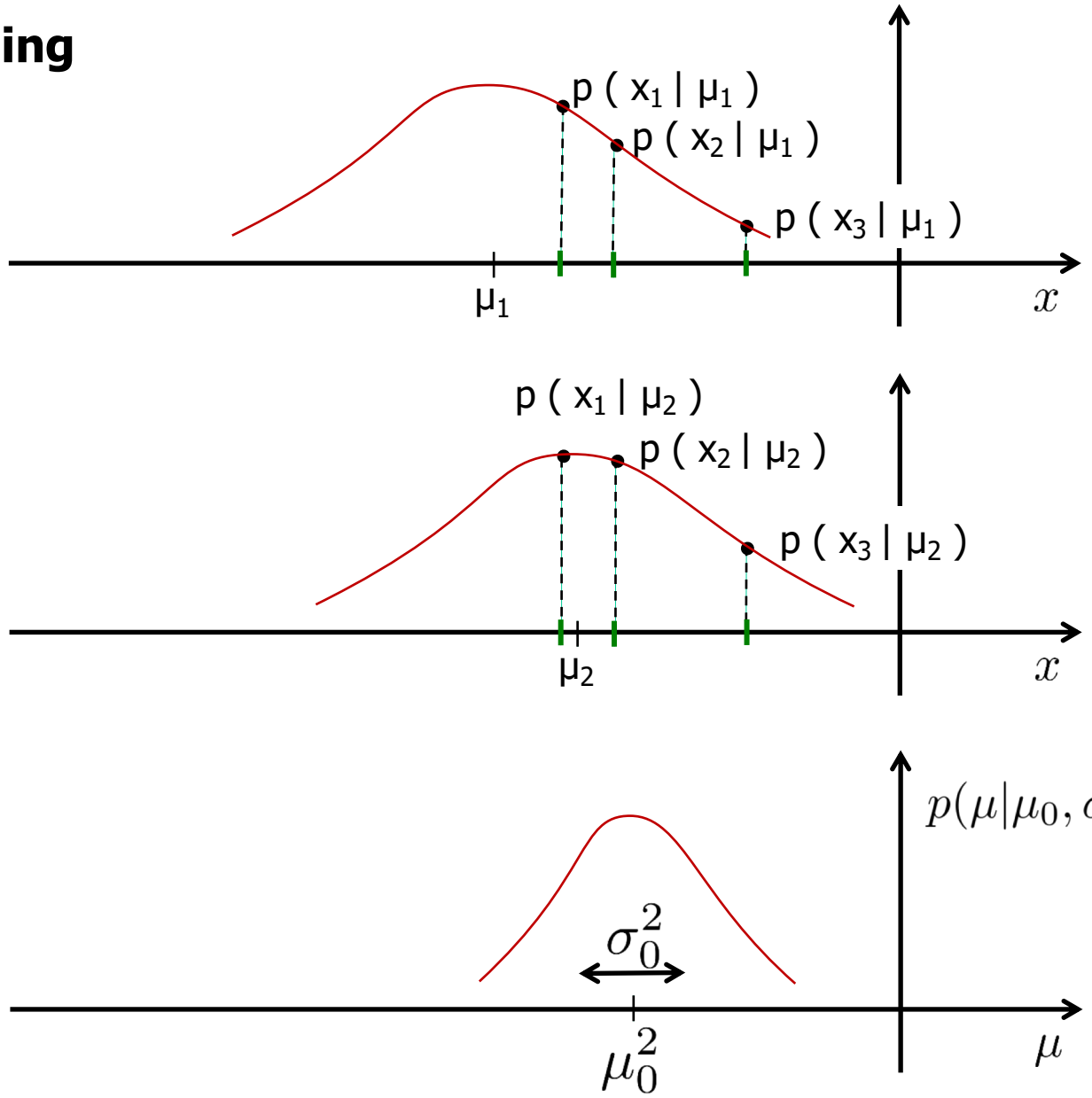


Do the same for MAP training but multiply each likelihood product with probability that given μ is correct.

MAP

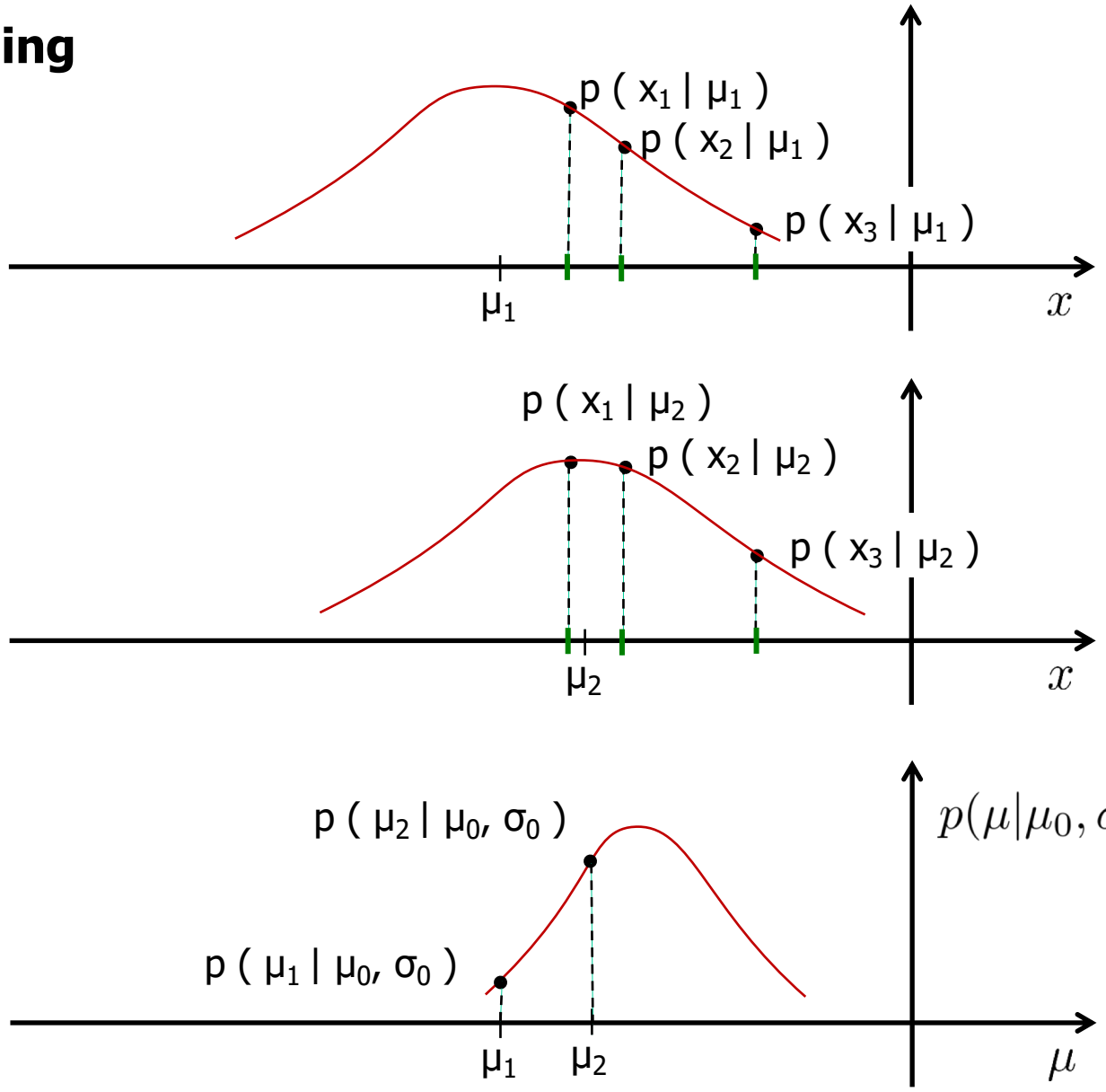
$$\hat{\Theta} = \arg\max_{\Theta} \left\{ p(\Theta) \cdot \prod_{k=1}^{n} p(x_k|\Theta) \right\}$$

3.4 Bayes Learning



known
prior
distribution

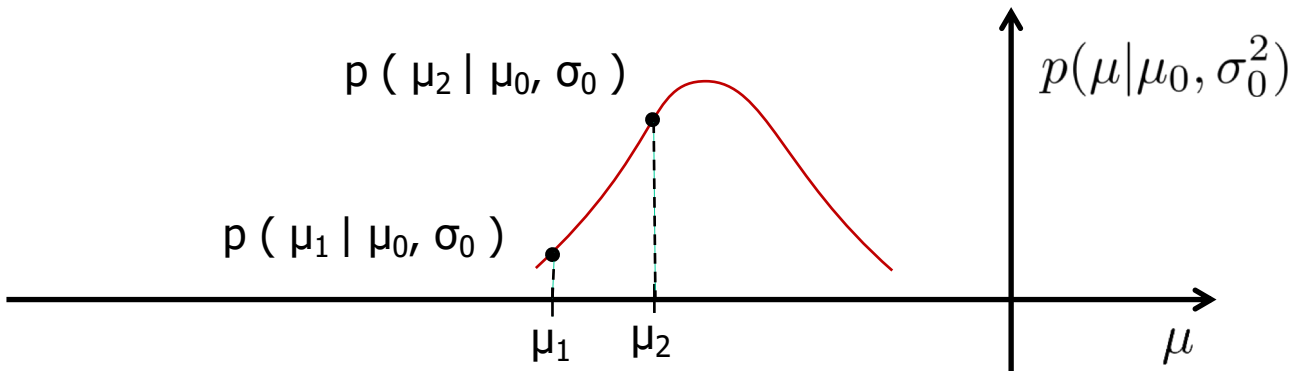
3.4 Bayes Learning



known prior distribution

3.4 Bayes Learning

known prior distribution



$$p(\mu_1 \mid \mu_0, \sigma_0) \cdot p(x_1 \mid \mu_1) \cdot p(x_2 \mid \mu_1) \cdot p(x_3 \mid \mu_1)$$

$$\lesseqgtr \; ?$$

$$p(\mu_2 \mid \mu_0, \sigma_0) \cdot p(x_1 \mid \mu_2) \cdot p(x_2 \mid \mu_2) \cdot p(x_3 \mid \mu_2)$$

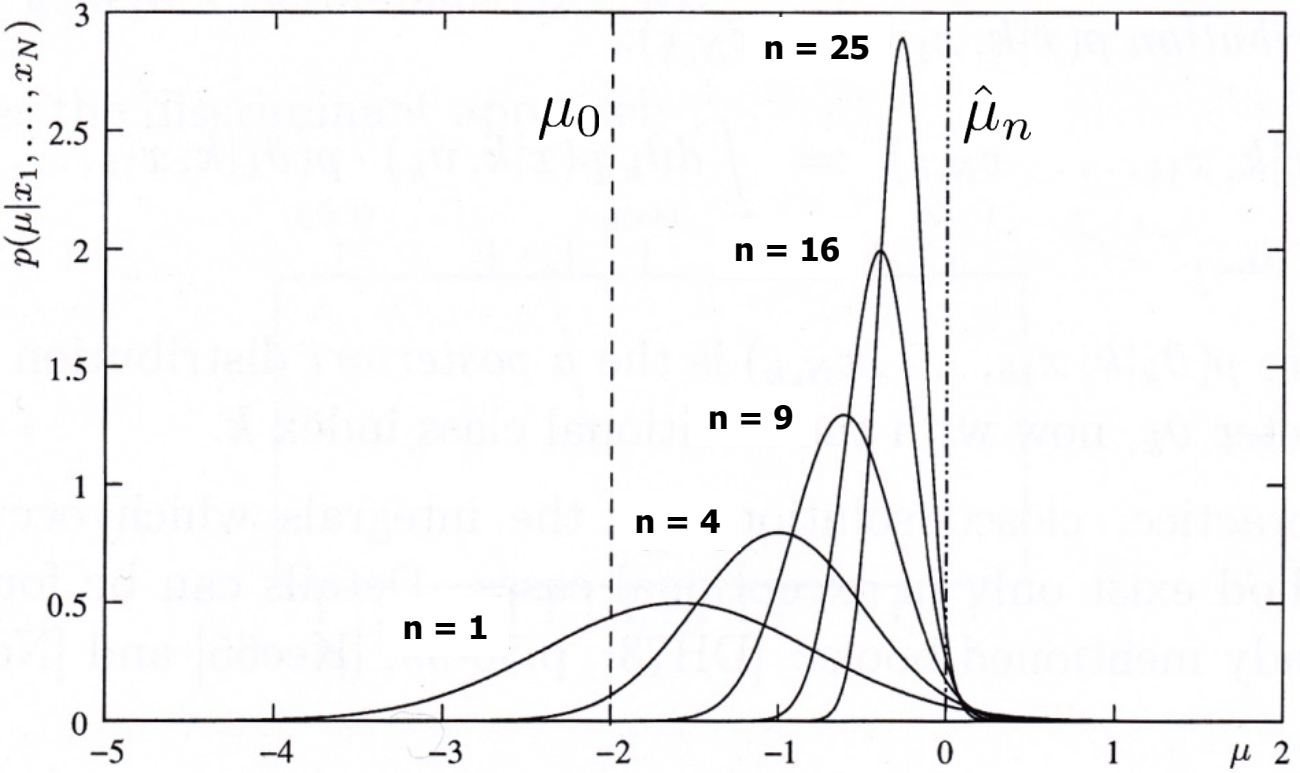
3.4 Bayes Learning

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

Example: Bayes learning of mean ($\Theta = \mu$) of univariate Gaussian distribution

Illustration with $\mu_0 = -2, \sigma_0 = 1, \sigma = 2, \hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} x_k = 0$



3.4 Bayes Learning – Octave Experiment

Task:

Estimation of mean of Gaussian density $p(x|\mu) = N(x; \mu, \sigma^2)$

Assumptions:

Unknown mean is normally distributed random variable with $p(\mu|\mu_0, \sigma_0^2)$

Given

- Variance $\sigma^2 = 2$
- Hyperparameters: $\mu_0 = -2, \sigma_0^2 = 1$
- n observations of x_k → training data

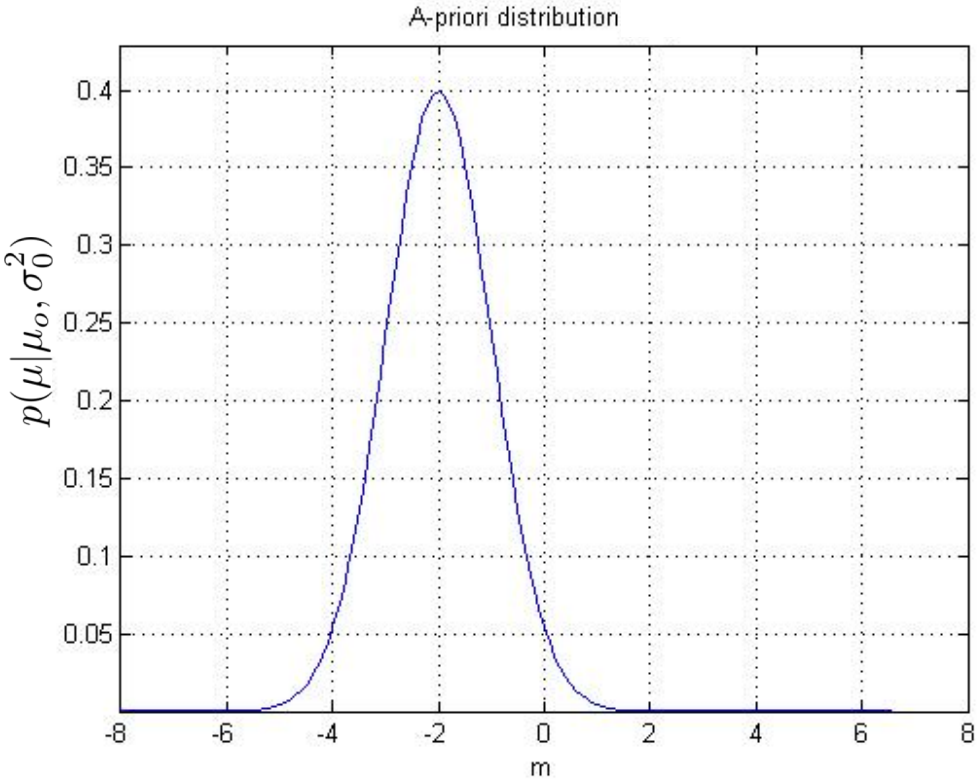
Goal:

"Good" estimation of μ, taking into account the available knowledge sources.

3.4 Bayes Learning – Octave Experiment

Given

- Hyperparameters: $\mu_o = -2, \sigma_0^2 = 1$



3.4 Bayes Learning – Octave Experiment

Given

- Hyperparameters: $\mu_0 = -2, \sigma_0^2 = 1$
- n observations of x_k → training data

Generation of training data

- Normally distributed $N(x; \mu, \sigma^2)$
- Mean of training data is assumed to be: $\hat{\mu}_n = 3$

  --> prior knowledge is quite bad
- Variance is known: $\sigma^2 = 2$

3.4 Bayes Learning – Octave Experiment

Gradually increase the training corpus (n = 1, 5, 14, 30, 55, 91, 140, 204, 285)

For each training set with n samples:

Estimate and plot

- $log(p(D|\mu))$
- $log(p(D|\mu) \cdot p(\mu))$

Maximized for MAP parameter estimation

3.4 Bayes Learning – Octave Experiment

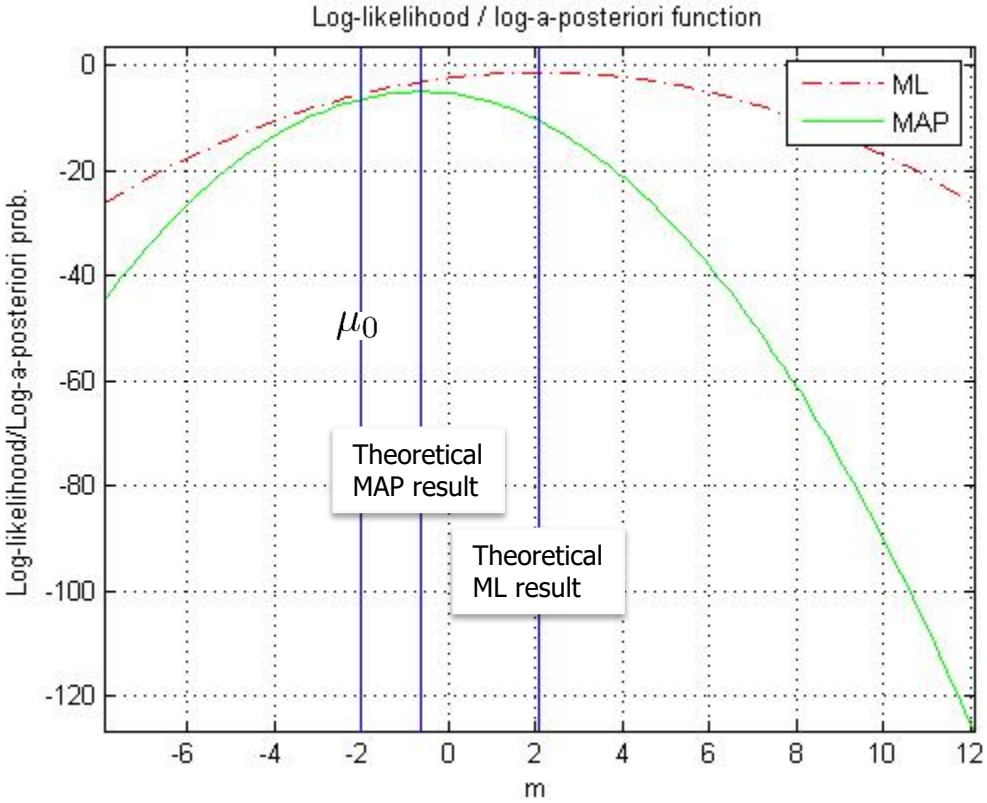
For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 1

o =

2.0918



Theoretical ML estimate: 2.09, Theoretical MAP estimate: -0.64

3.4 Bayes Learning – Octave Experiment

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

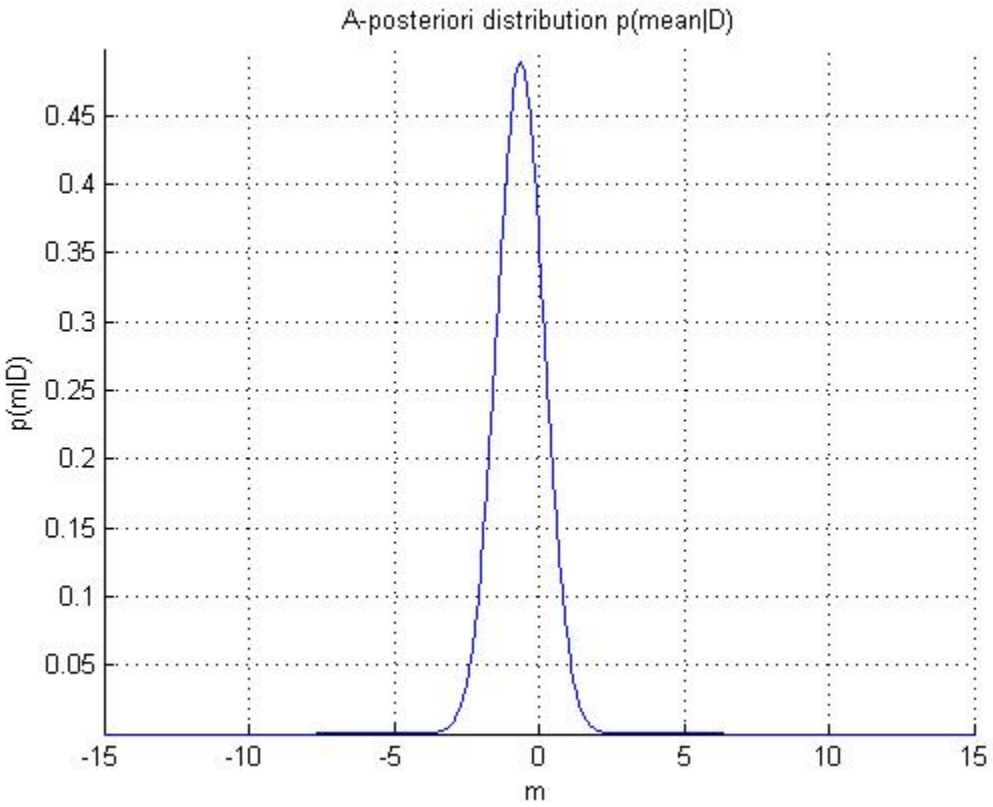
For each training set with n samples:

Estimate a-posteriori distribution

1. n = 1

o =

2.0918



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 5

o =

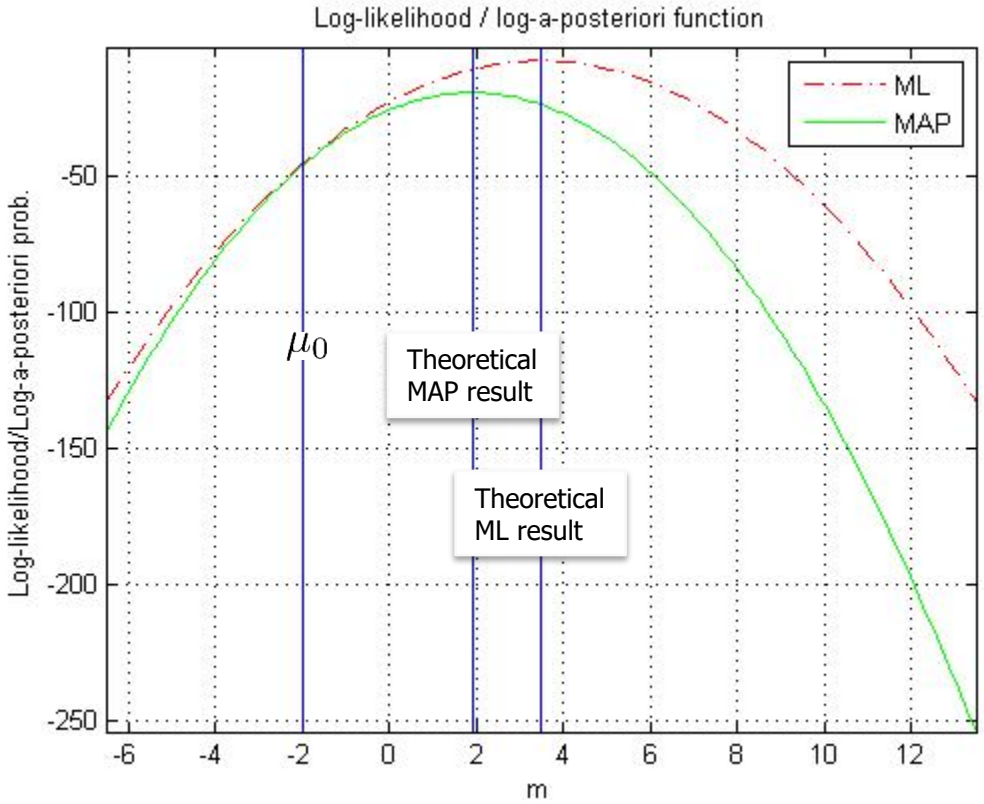
2.0918

2.7448

4.0153

3.4262

5.1905



Theoretical ML estimate: 3.49, Theoretical MAP estimate: 1.92

3.4 Bayes Learning – Octave Experiment

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

For each training set with n samples:

Estimate a-posteriori distribution

1. n = 5

o =

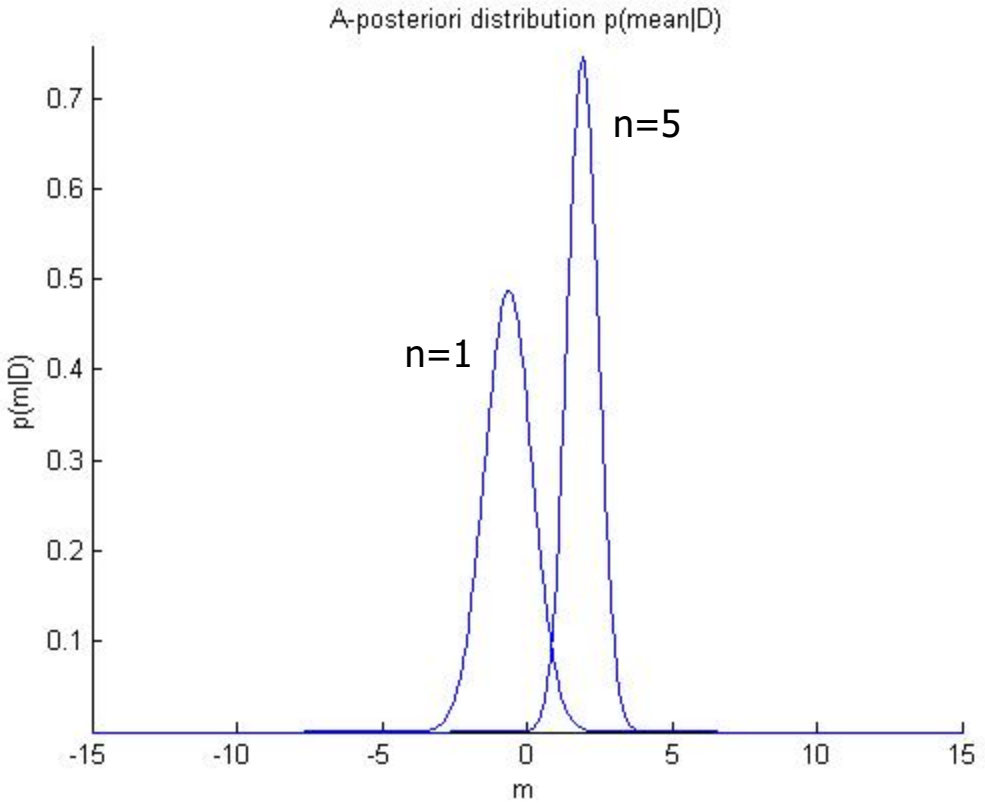
2.0918

2.7448

4.0153

3.4262

5.1905



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 14

o =

2.0918

2.7448

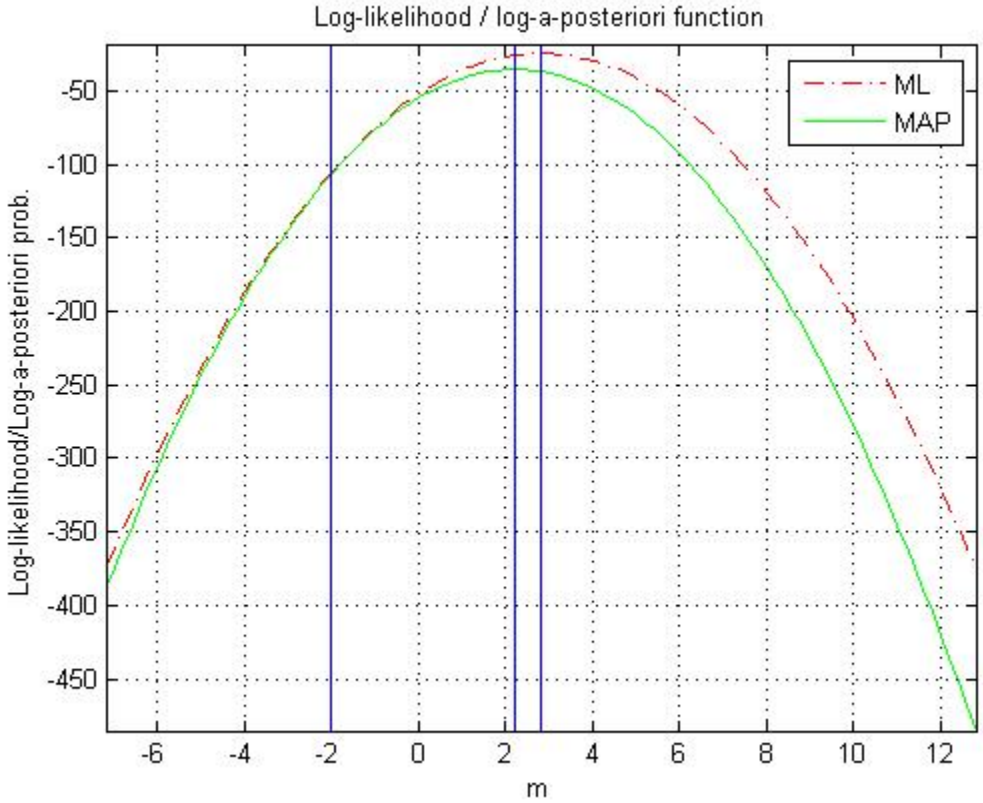
4.0153

3.4262

5.1905

2.9374

...



Theoretical ML estimate: 2.82, Theoretical MAP estimate: 2.22

3.4 Bayes Learning – Octave Experiment

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

For each training set with n samples:

Estimate a-posteriori distribution

1. n = 14

o =

3.1936

2.9779

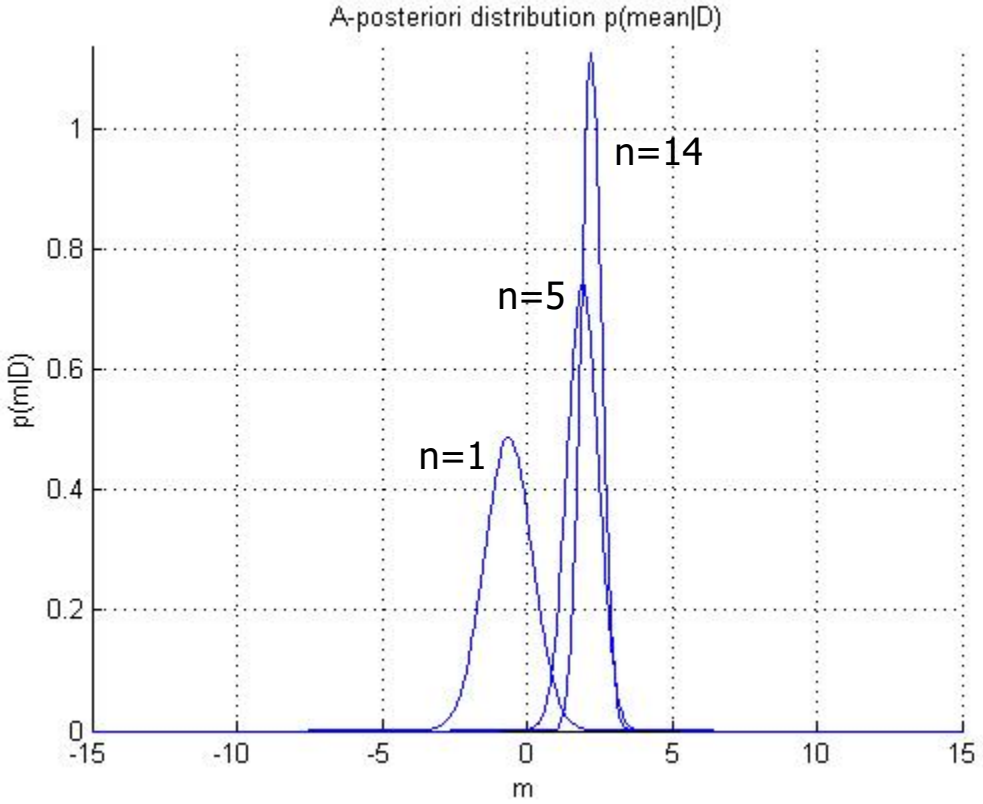
1.6728

0.9097

3.5118

3.6757

...



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 30

o =

2.0918

2.7448

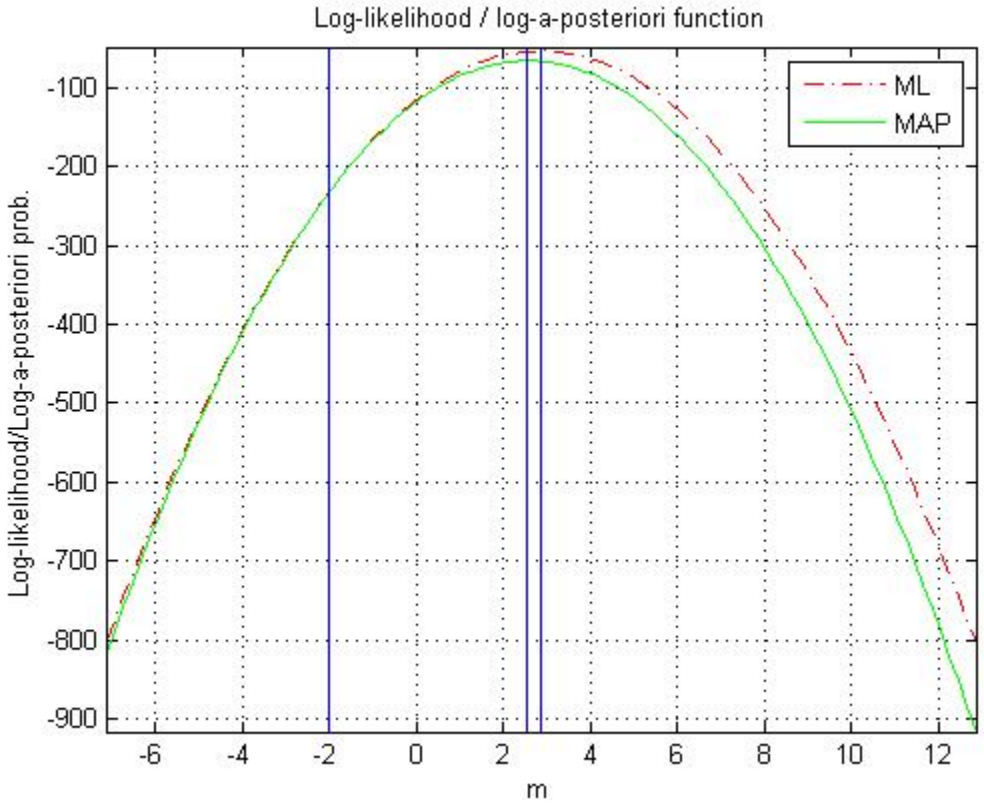
4.0153

3.4262

5.1905

2.9374

...



Theoretical ML estimate: 2.87, Theoretical MAP estimate: 2.57

3.4 Bayes Learning – Octave Experiment

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

For each training set with n samples:

Estimate a-posteriori distribution

1. n = 30

o =

3.1936

2.9779

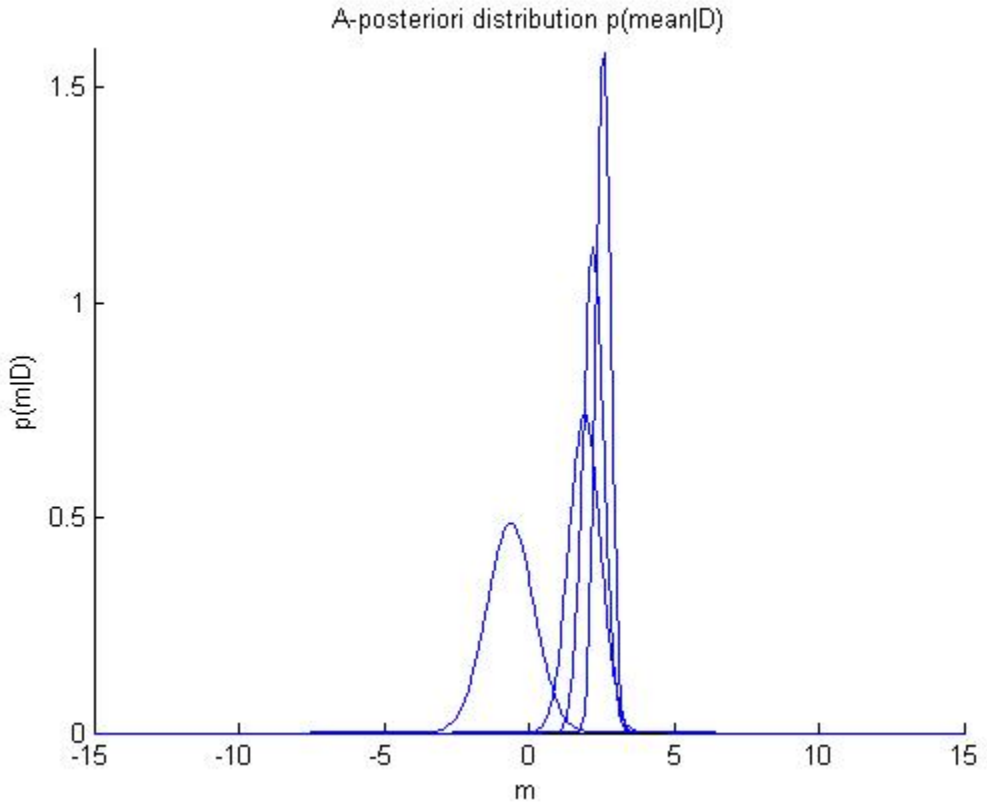
1.6728

0.9097

3.5118

3.6757

...



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 55

o =

2.0918

2.7448

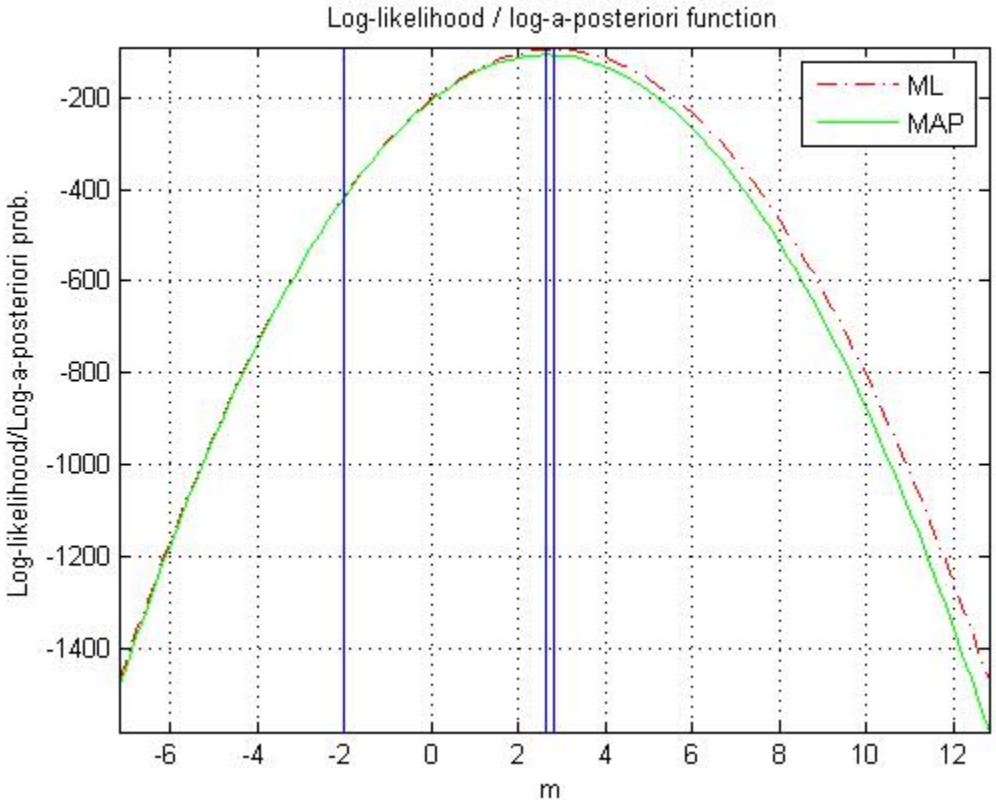
4.0153

3.4262

5.1905

2.9374

...



Theoretical ML estimate: 2.82, Theoretical MAP estimate: 2.65

3.4 Bayes Learning – Octave Experiment

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

For each training set with n samples:

Estimate a-posteriori distribution

1. n = 55

o =

3.1936

2.9779

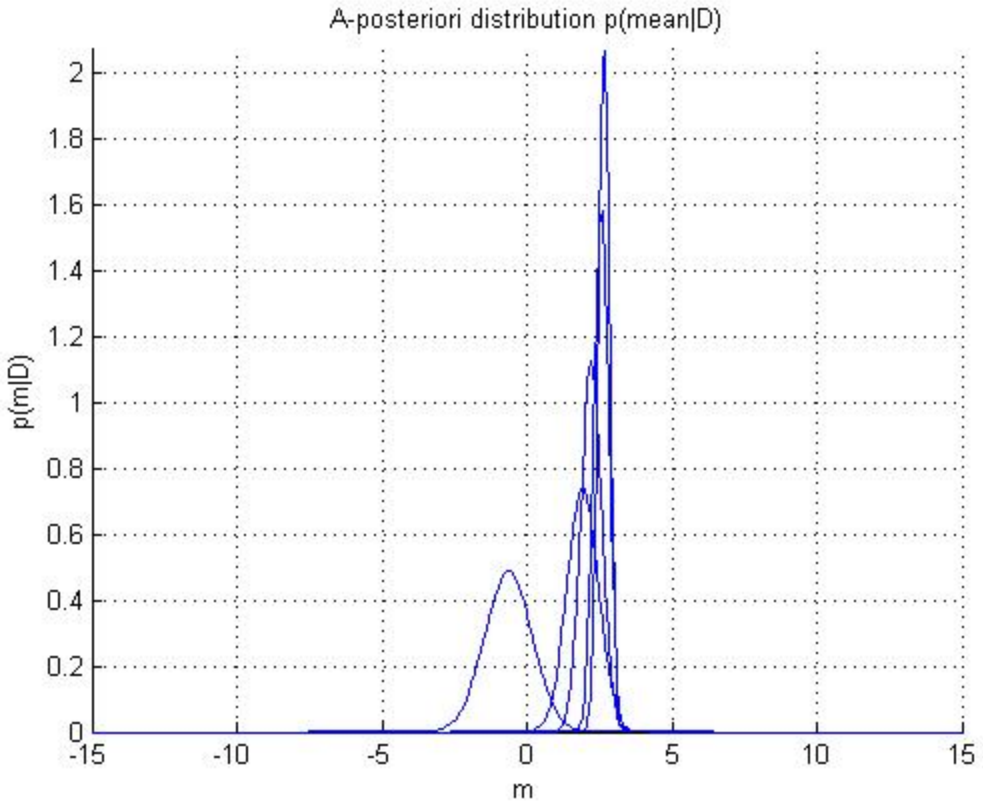
1.6728

0.9097

3.5118

3.6757

...



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 91

o =

2.0918

2.7448

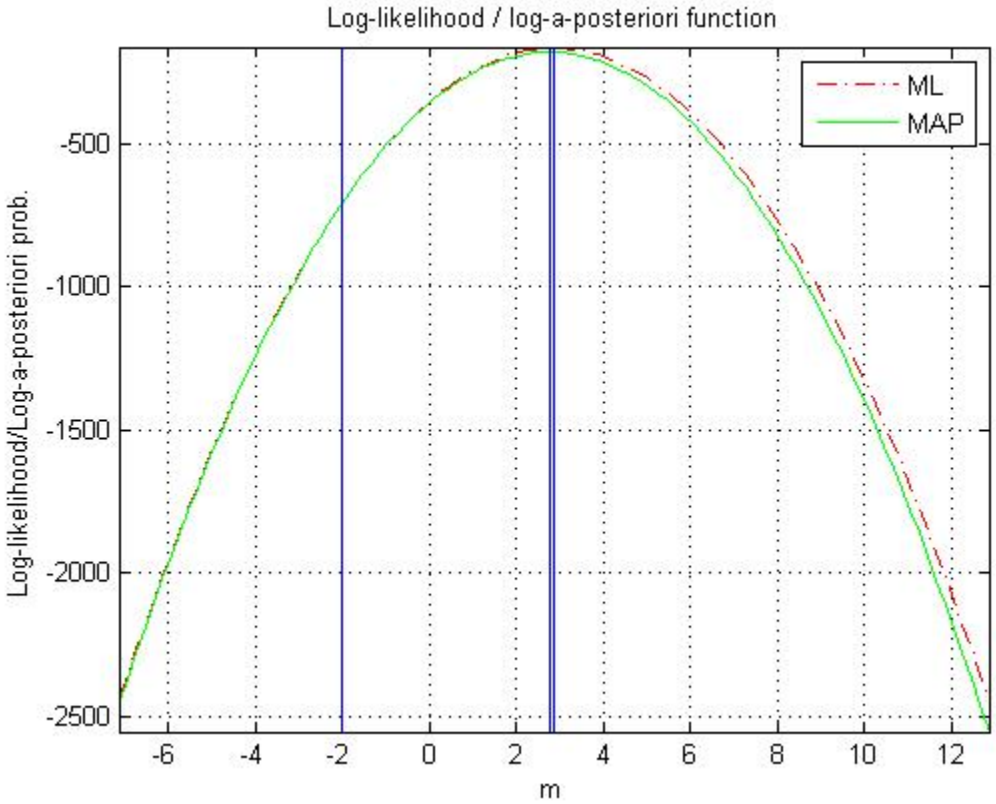
4.0153

3.4262

5.1905

2.9374

...



Theoretical ML estimate: 2.89, Theoretical MAP estimate: 2.76

3.4 Bayes Learning – Octave Experiment

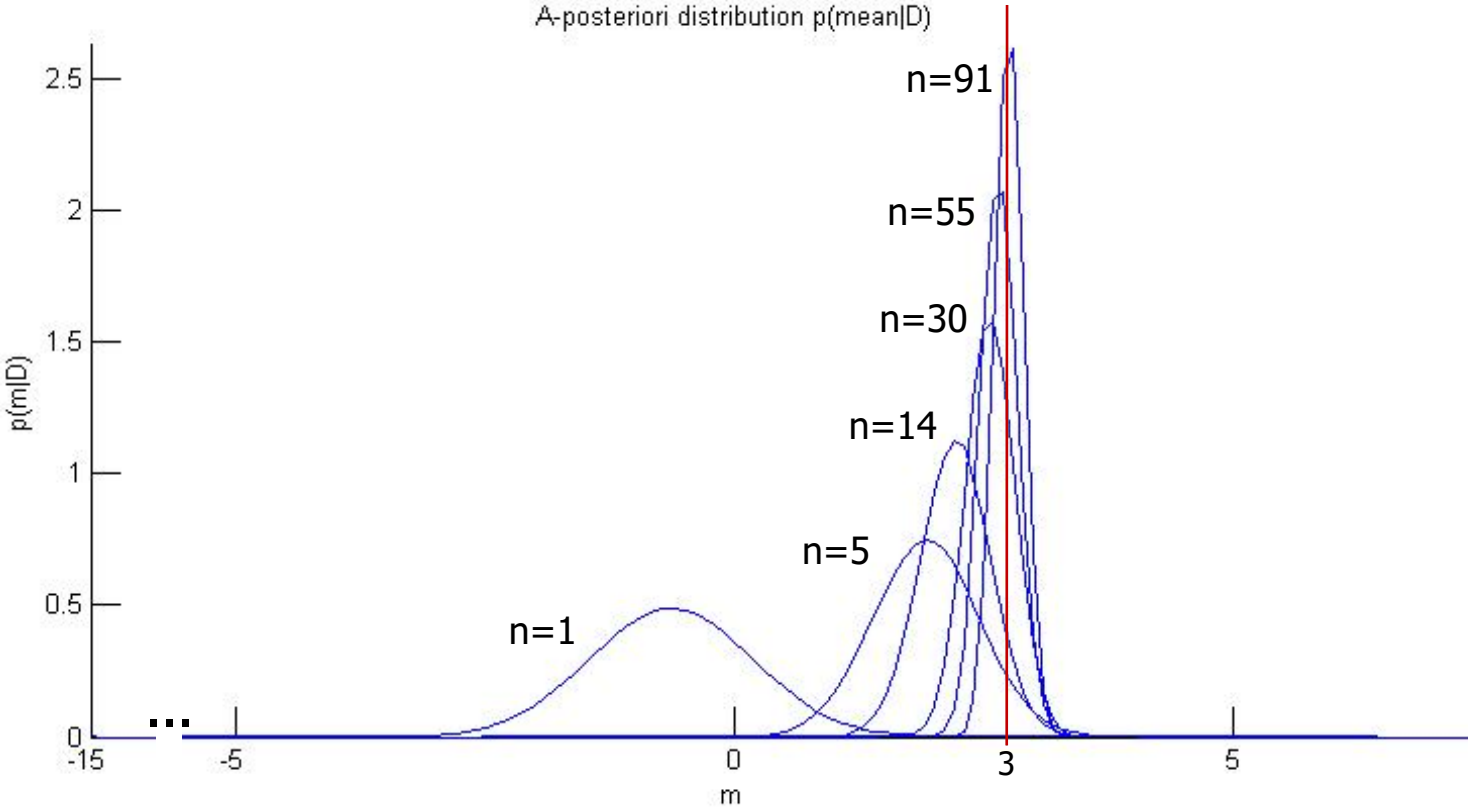
$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_N^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

For each training set with n samples:

Estimate a-posteriori distribution

1. n = 91



3.4 Bayes Learning – Octave Experiment

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 285

o =

2.0918

2.7448

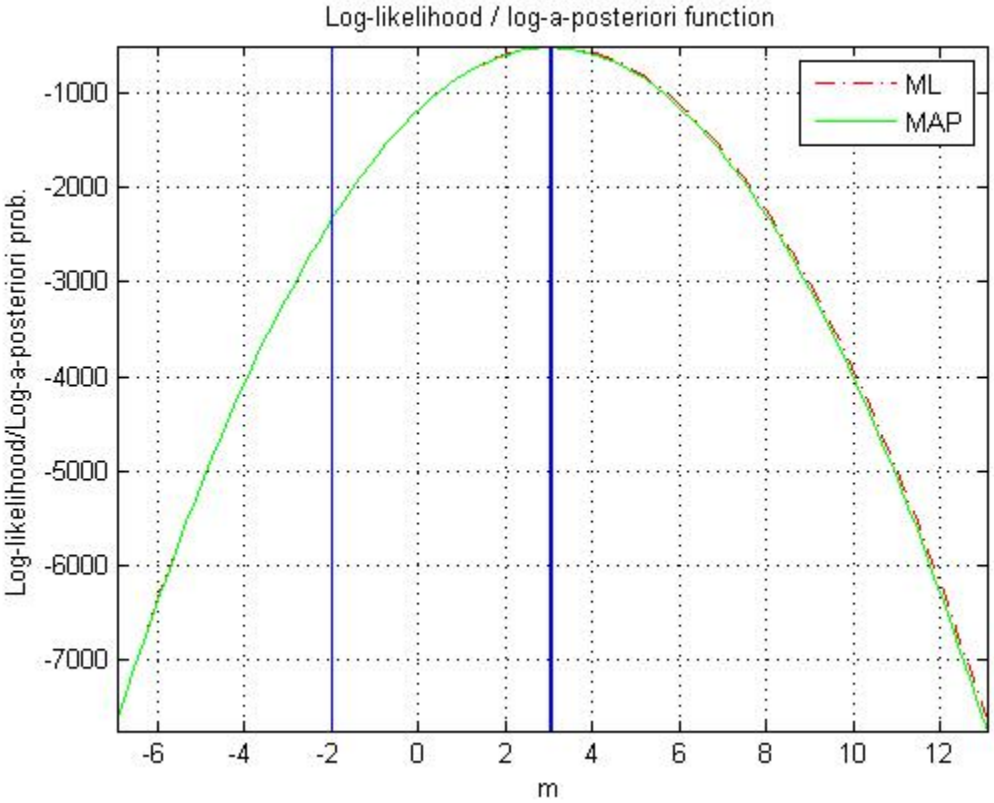
4.0153

3.4262

5.1905

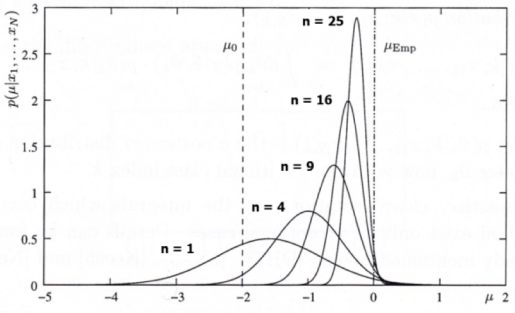
2.9374

...



Theoretical ML estimate: 3.08, Theoretical MAP estimate: 3.04

3.4 Bayes Learning



Remarks (1):

In contrast to the ML solution Bayes learning provides **a full distribution**

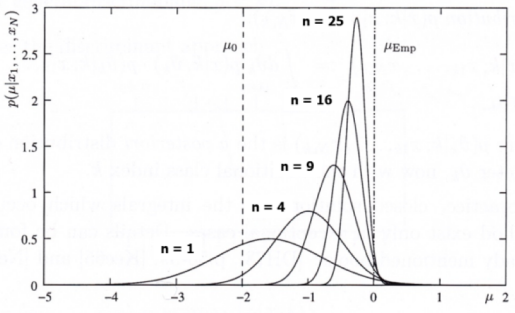
$$p(\mu|D)$$

If only a single parameter estimation is of interest use the

Maximum-A-Posteriori estimation

$$\hat{\mu}_{MAP} = \arg\max_{\mu} p(\mu|D)$$

3.4 Bayes Learning



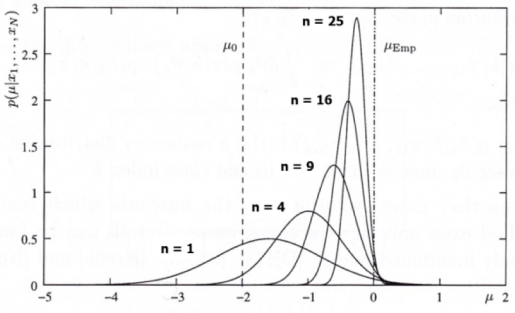
Remarks (2):

$$\mu_N = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

Describes an averaging between

- the empirical mean $\hat{\mu}_n$ with weight $\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}$ and
- the a-priori mean μ_0 with weight $\frac{\sigma^2}{n\sigma_0^2+\sigma^2}$

3.4 Bayes Learning



Remarks (3):

For n → infinity, the influence of the prior distribution tends to zero and the a-posteriori distribution $p(\mu|\mu_N, \sigma_N^2)$ results in the delta function centered at the empirical mean (ML solution).

Exercise 6 - Bayes Learning

Task:

Estimation of constant c, observed under the influence of noise e_k

$$x_k = c + e_k$$

Assumptions:

- Noise is normally distributed with mean 0 and variance σ_e^2
- Constant c is a random variable with zero mean and variance σ_c^2

Given

- Prior knowledge of c (μ_c ,σ_c^2) and noise e(μ_e,σ_e^2)
- n observations of x_k → training data

Goal:

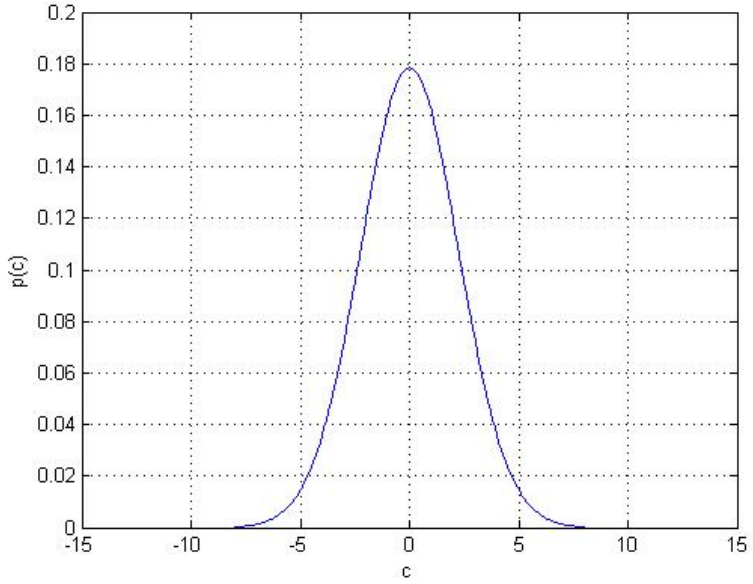
"Good" estimation of c, taking into account the available knowledge sources.

Exercise 6 - Bayes Learning

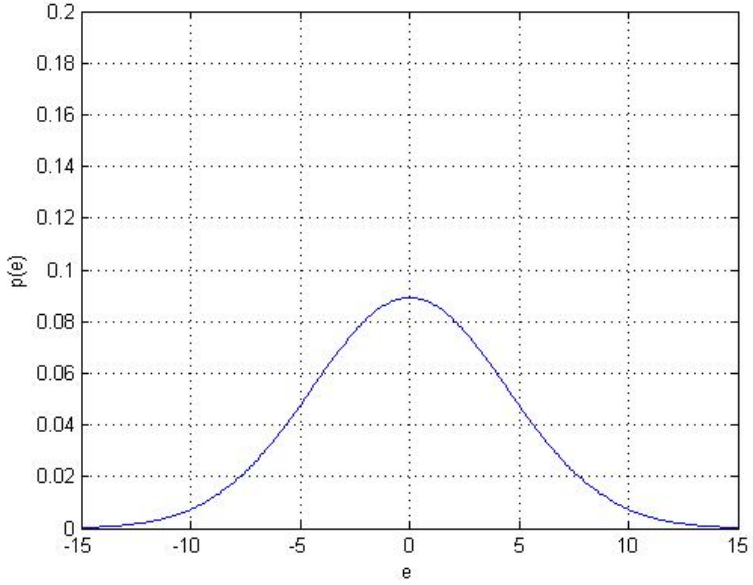
Given

- Prior knowledge of c → $p(c|\mu_c, \sigma_c^2)$
- Prior knowledge of e → $p(e|\mu_e, \sigma_e^2)$

$$p(c|\mu_c = 0, \sigma_c^2 = 5)$$



$$p(e|\mu_e = 0, \sigma_e^2 = 20)$$



Exercise 6 - Bayes Learning

Task:

Estimation of constant c, observed under the influence of noise e_k

$$x_k = c + e_k$$

Assumptions:

- Noise is normally distributed with mean 0 and variance σ_e^2
- Constant c is a random variable with zero mean and variance σ_c^2

Given

- Prior knowledge of c (μ_c ,σ_c) and noise e(μ_e, σ_e)
- n observations of x_k → training data

Goal:

"Good" estimation of c, taking into account the available knowledge sources.

Exercise 6 - Bayes Learning

Generation of training data

- Constant is assumed to have a fixed value: $c = 30$

  → prior knowledge is quite wrong!

- For different n ({1 2 4 8 16 32 64 128 256 512 1024})

  Generate n training samples using $x_k = c + e_k$

- For each training set with n samples:

  Estimate and plot

  – $log(p(D|\mu))$

  – $log(p(D|\mu) \cdot p(\mu))$

Maximized for MAP parameter estimation

Exercise 6 - Bayes Learning

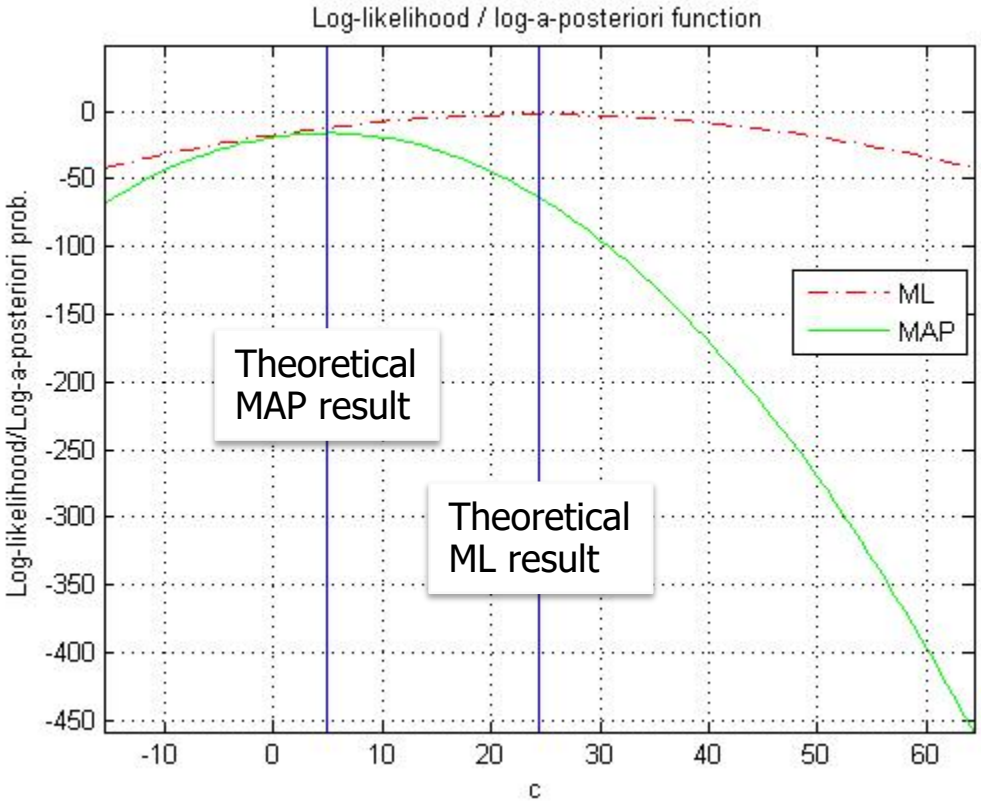
For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

1. n = 1

e =

24.4132



Exercise 6 - Bayes Learning

For each training set with n samples:

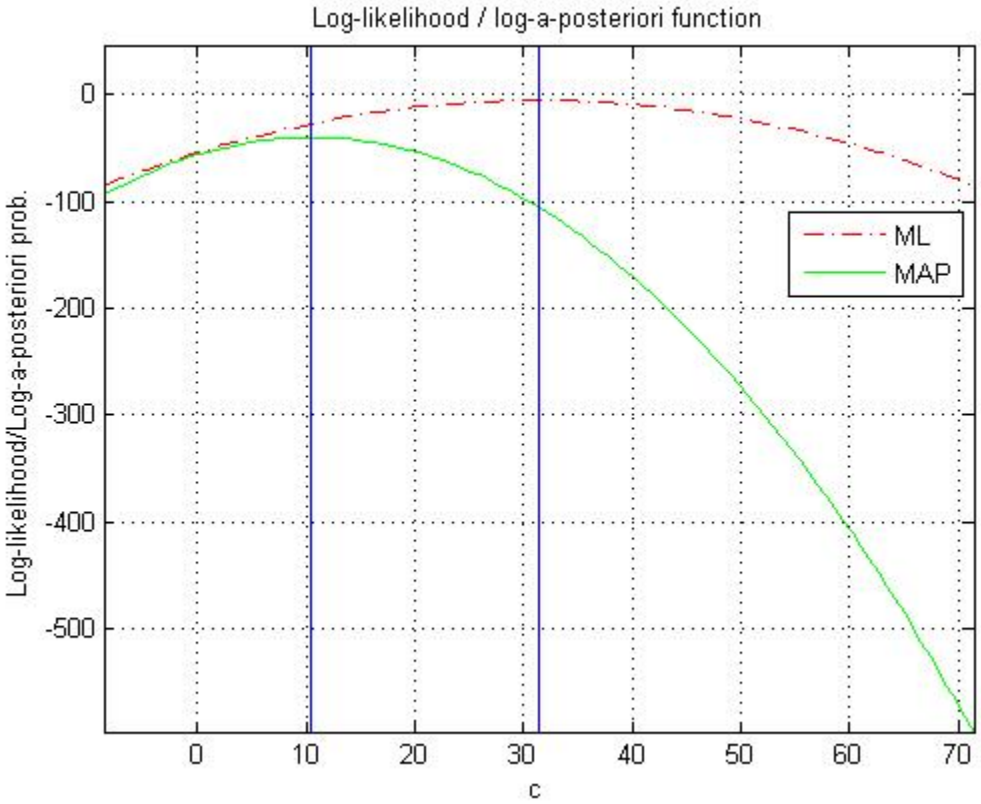
Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 2

e =

34.0240

28.8097



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 4

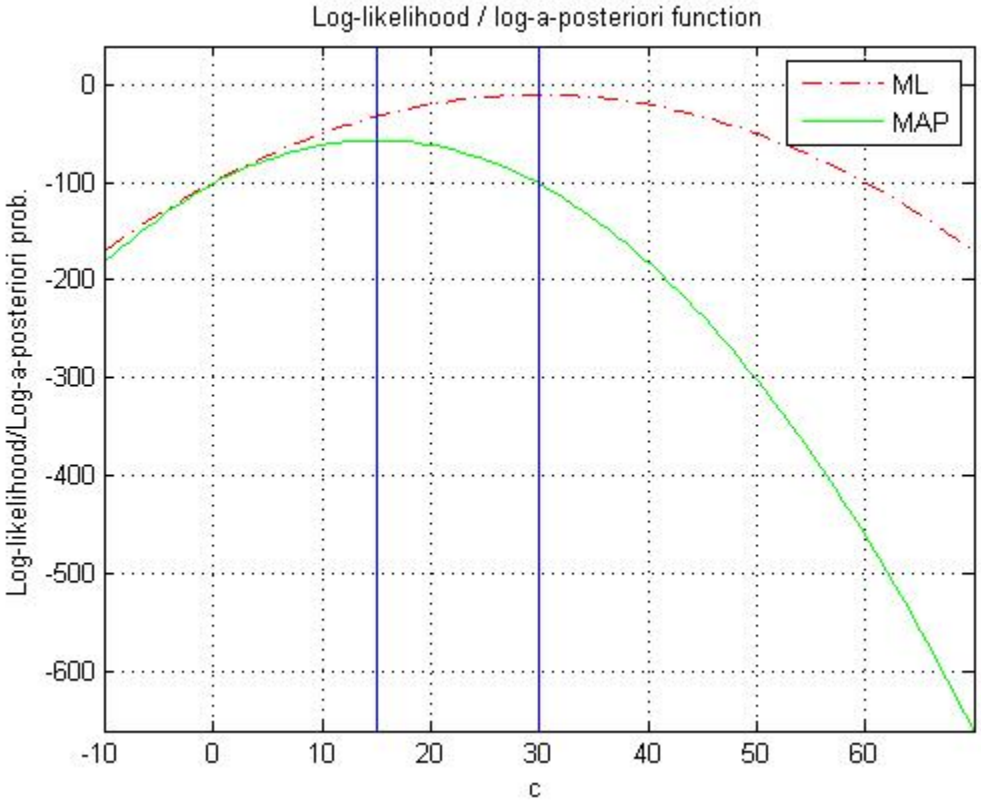
e =

34.5851

30.2318

26.8810

28.0725



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 8

e =

28.8391

24.3278

20.7094

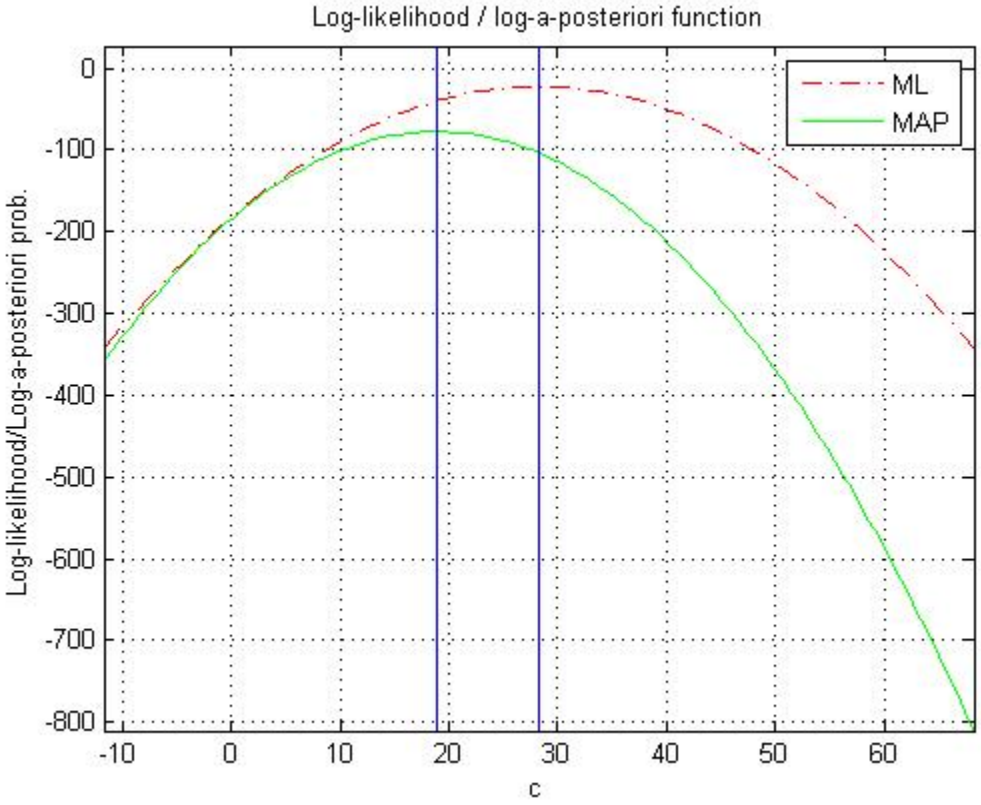
28.5715

31.0383

27.6976

29.5379

35.3295



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 16

e =

27.6209

32.2582

28.6064

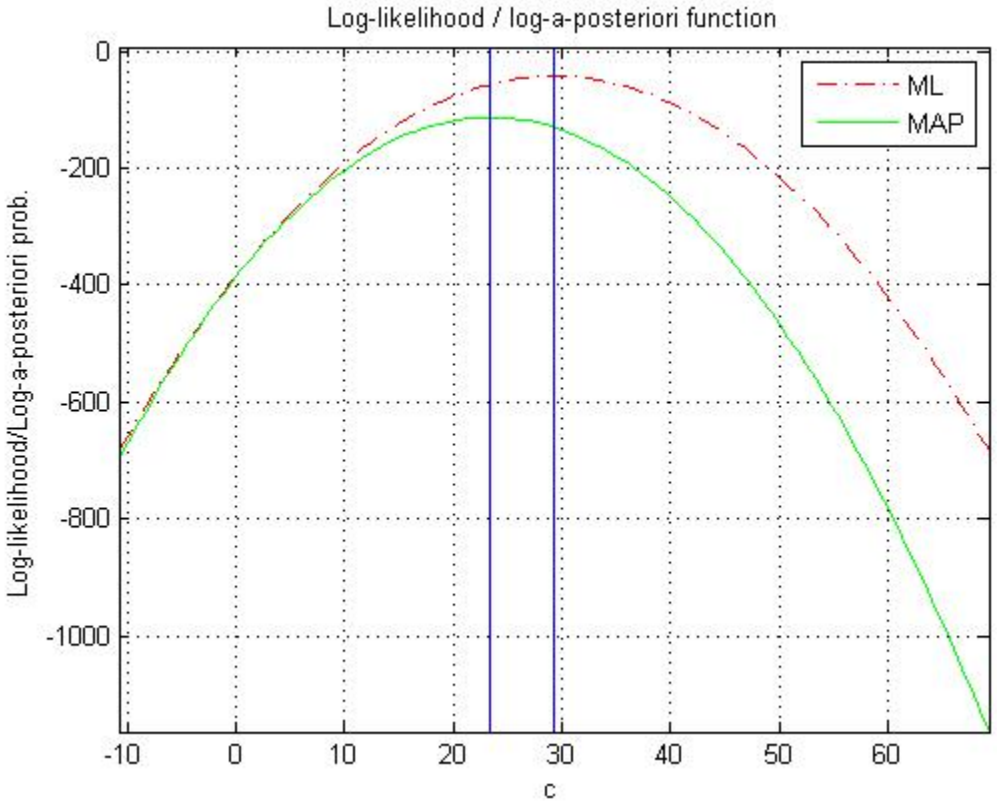
24.0197

36.4282

27.6559

26.6136

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 32

e =

30.9436

26.6712

33.6657

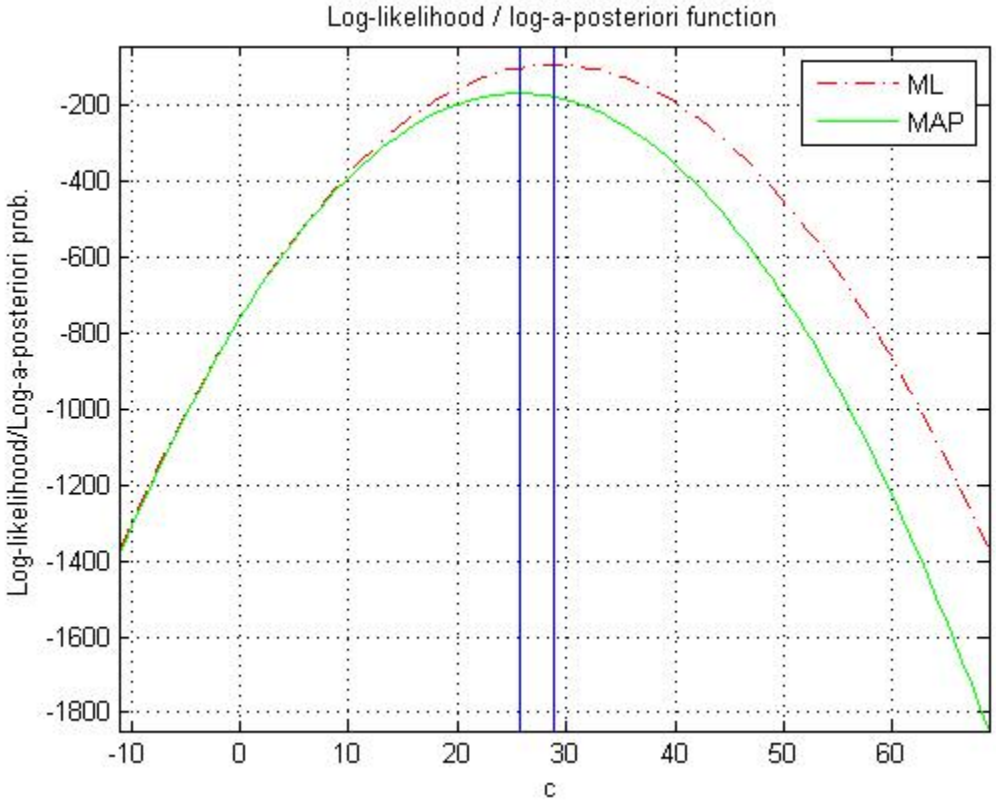
23.0147

26.6674

30.3635

24.2568

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 64

e =

39.0078

37.1491

26.2075

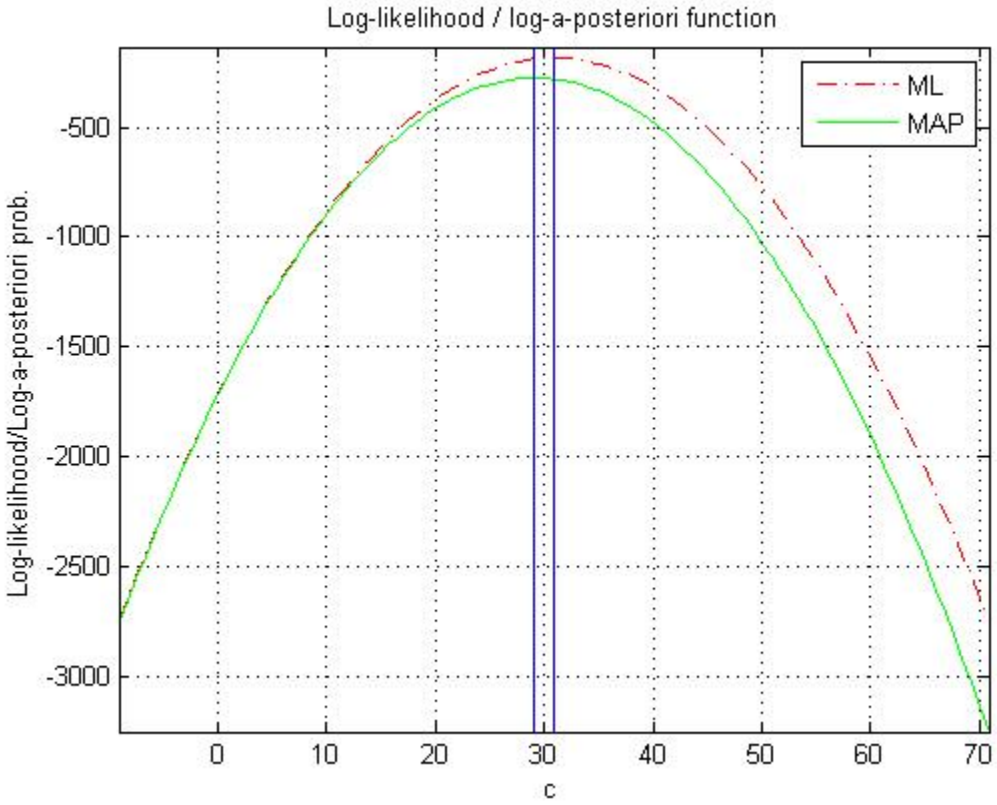
25.4153

28.8009

27.8261

35.7681

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 128

e =

32.3061

41.2414

21.6919

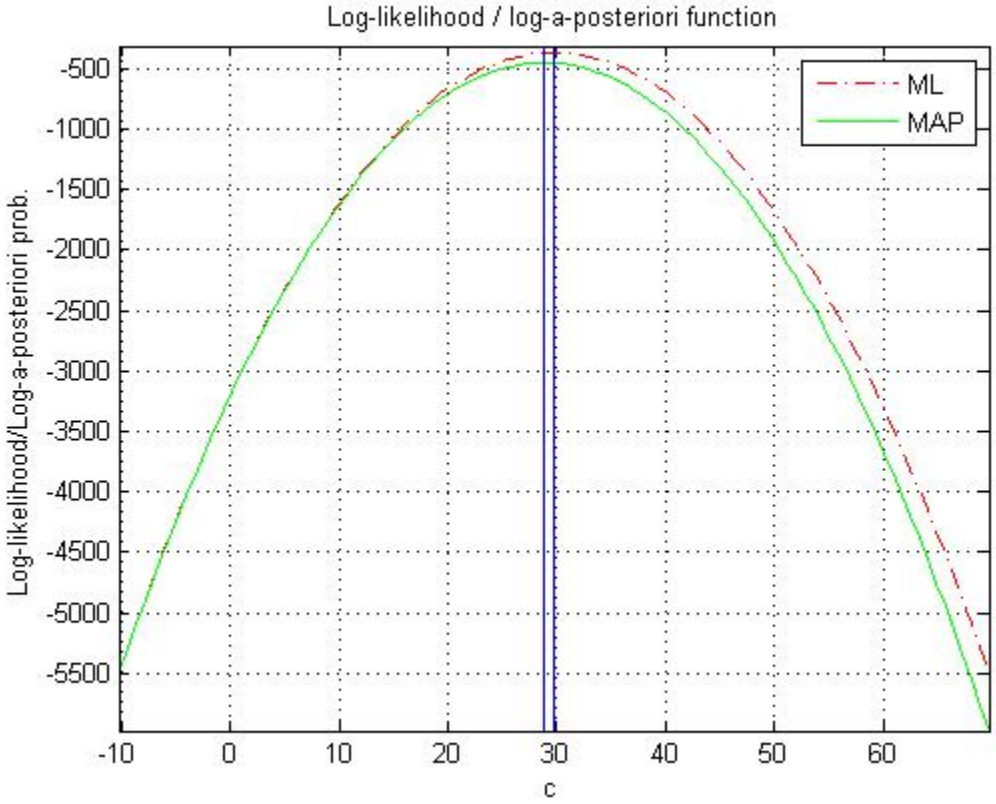
31.6026

31.1737

32.7232

27.2424

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 256

e =

35.2116

30.8420

23.6114

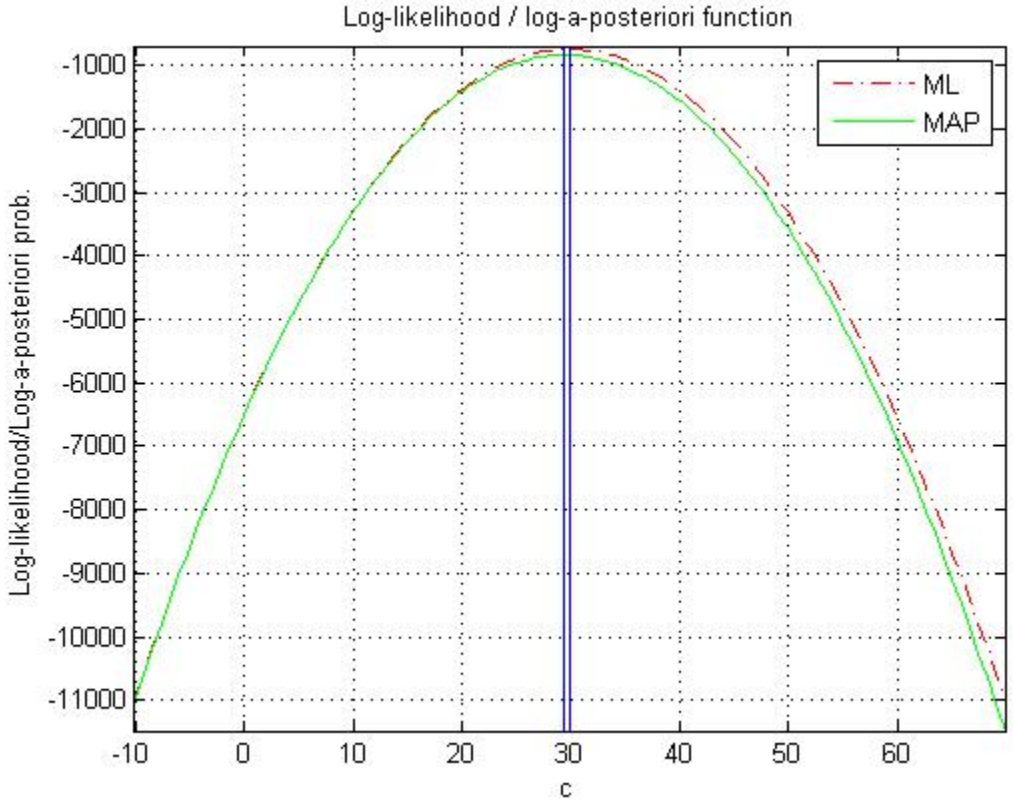
31.1572

23.1048

39.2989

33.3336

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 512

e =

27.0422

36.5392

18.6778

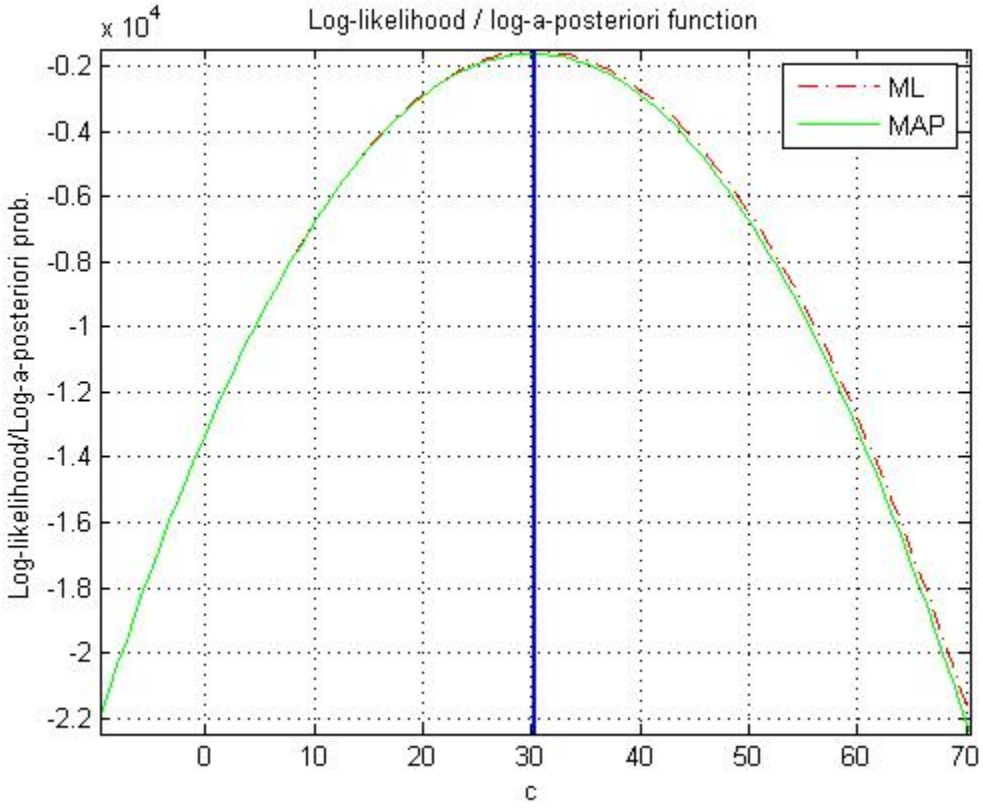
34.4218

37.4214

29.6516

29.5996

...



Exercise 6 - Bayes Learning

For each training set with n samples:

Estimate and plot log (likelihood), log (likelihood * prior):

2. n = 1024

e =

28.3212

30.5653

33.1773

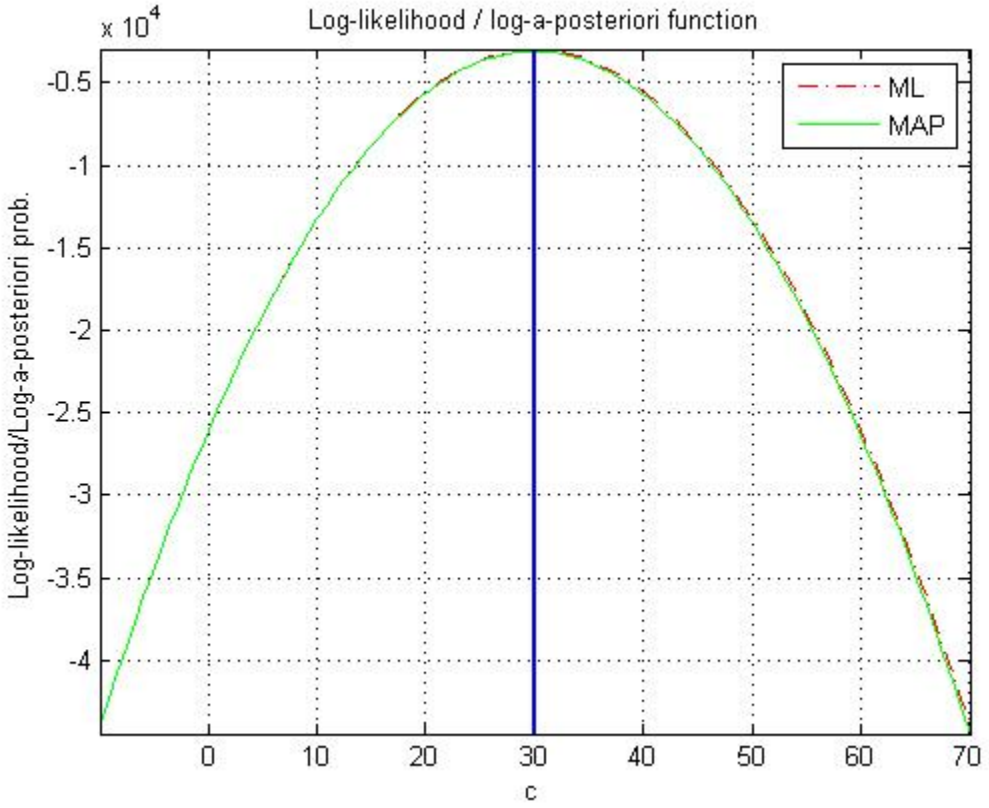
27.8407

21.0703

26.5033

26.0548

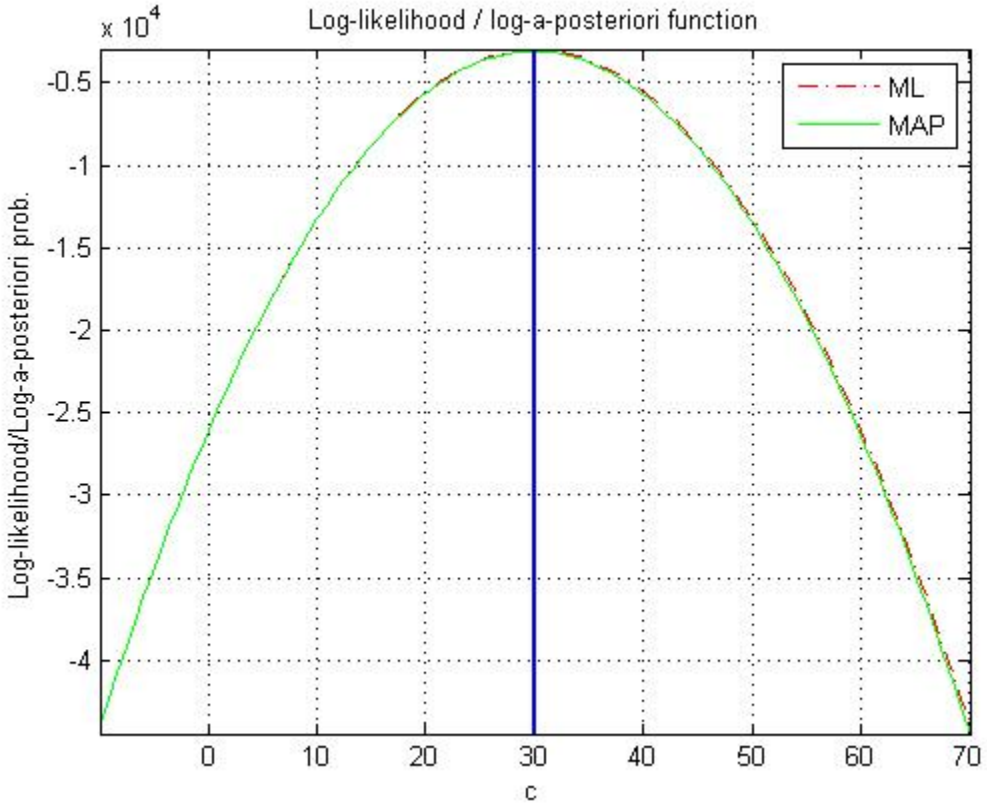
...



Exercise 6 - Bayes Learning

Result:

Both techniques have successfully estimated the "unknown" parameter c (which has been set to 30 for generation of the training data - see above).



3.4 Bayes Learning

Recursive Bayes Approach (or: incremental / online learning)

Using the i.i.d. (independently and identically distributed) assumption for the training data, we may derive a recursion:

$$p(D^n|\Theta) = p(x_n, ..., x_1|\Theta) = p(x_n|\Theta)p(D^{n-1}|\Theta)$$

with

$$\begin{aligned} D^n &= \{x_1, ..., x_n\} \\ D^{n-1} &= \{x_1, ..., x_{n-1}\} \end{aligned}$$

Inserting this into Bayes rule

$$p(\Theta|D) = \frac{p(D|\Theta) \cdot p(\Theta)}{\int_{\Theta'} p(D|\Theta')p(\Theta')d\Theta'}$$

3.4 Bayes Learning

Recursive Bayes Approach

leads to

$$p(\Theta|D^n) = \frac{p(D^n|\Theta)p(\Theta)}{\int_{\Theta'} p(D^n|\Theta')p(\Theta')d\Theta'}$$

and

$$p(\Theta|D^n) = \frac{p(x_n|\Theta)p(D^{n-1}|\Theta)p(\Theta)}{\int_{\Theta'} p(x_n|\Theta')p(D^{n-1}|\Theta')p(\Theta')d\Theta'}$$

With

$$p(D^{n-1}|\Theta)p(\Theta) = p(\Theta|D^{n-1})p(D^{n-1})$$

3.4 Bayes Learning

Incremental / Online learning

we obtain

$$p(\Theta|D^n) = \frac{p(x_n|\Theta)p(\Theta|D^{n-1})p(D^{n-1})}{\int_{\Theta'} p(x_n|\Theta')p(\Theta'|D^{n-1})p(D^{n-1})d\Theta'}$$

and thus

$$p(\Theta|D^n) = \frac{p(x_n|\Theta)p(\Theta|D^{n-1})}{\int_{\Theta'} p(x_n|\Theta')p(\Theta'|D^{n-1})d\Theta'}$$

3.4 Bayes Learning

$$p(\Theta|D^n) = \frac{p(x_n|\Theta)p(\Theta|D^{n-1})}{\int_{\Theta'} p(x_n|\Theta')p(\Theta'|D^{n-1})d\Theta'}$$

Incremental / Online learning

Start with $p(\Theta|D^0) = p(\Theta)$

and incrementally increase the amount of training data.

This leads to a sequence of densities:

$$p(\Theta), p(\Theta|x_1), p(\Theta|x_1, x_2), \ldots$$

Incremental or Online Learning

3.5 Distribution of Distances in $\mathbb{R}^D$

Problem

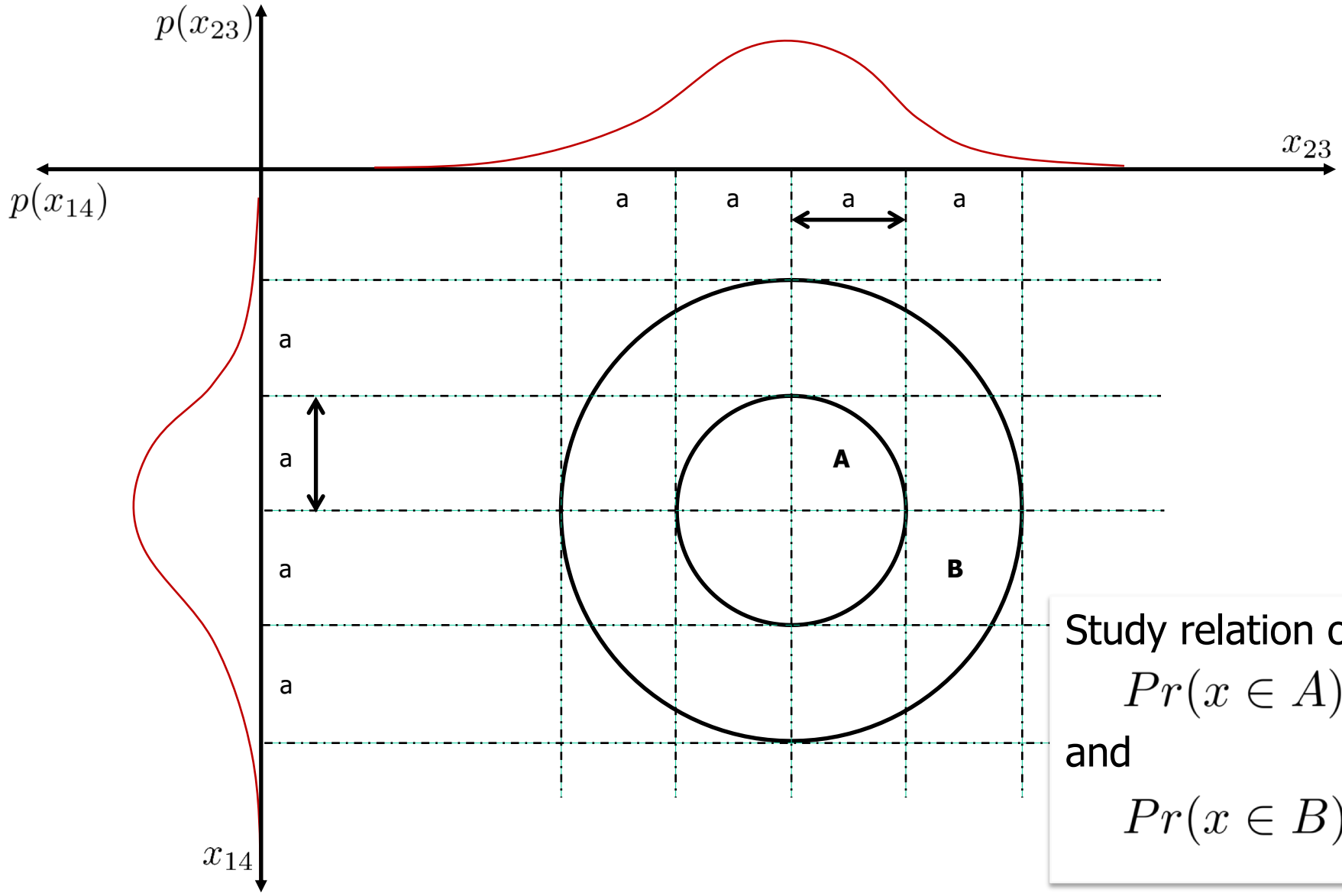
Space is very sparsely filled with training data if $D >> 1$

(in contrast to two or three dimensional case)

For demonstration let us consider the following example:

- Fixed class k
- Given distribution $p(x|k)$ (e.g. Gaussian) for $x \in \mathbb{R}^D$

3.5 Distribution of Distances in $\mathbb{R}^D$



Study relation of

$$Pr(x \in A)$$

and

$$Pr(x \in B)$$

3.5 Distribution of Distances in $\mathbb{R}^D$

Approximation of $Pr(x \in A)$ and $Pr(x \in B)$

$$Pr(x \in A) = c_D \cdot a^D \cdot \langle p(x|k) \rangle_A$$

Volume content of an n-sphere (sphere generalized to n dimensions) with c_D being a constant depending on D

Average distribution density in region A

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

3.5 Distribution of Distances in $\mathbb{R}^D$

Probability ratio of observing a sample in A, respectively B:

$$\frac{Pr(x \in B)}{Pr(x \in A)} = \frac{c_D \cdot a^D \cdot (2^D - 1) \cdot \langle p(x|k) \rangle_B}{c_D \cdot a^D \cdot \langle p(x|k) \rangle_A}$$

Example: $D = 16$ $\quad \frac{\langle p(x|k) \rangle_B}{\langle p(x|k) \rangle_A} = \frac{1}{10}$

$$\frac{Pr(x \in B)}{Pr(x \in A)} = 6554$$

3.5 Distribution of Distances in $\mathbb{R}^D$

$$\frac{Pr(x \in B)}{Pr(x \in A)} = 6554$$

Despite the 10-times higher probability density in A it is much more likely to observe a training sample falling into region B than into A.

→ Much more training samples will occur
far away from the center of the distribution.

3.5 Distribution of Distances in $\mathbb{R}^D$

Let us consider the distribution of distances between

- Random variable $x \in \mathbb{R}^D$ (normally distributed)
- Class center μ_k of class k

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

How does this distribution change in case of an increasing dimensionality?

Can be exactly calculated if $p(x|k)$ is normally distributed with $\Sigma_k = I$

General case with arbitrary covariance matrix can be reduced to this special case by using the **Whitening transformation**.

3.5 Distribution of Distances in $\mathbb{R}^D$

Result is the Chi-square distribution:

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

$$p(\chi^2) = \frac{1}{\Gamma(\frac{D}{2}) 2^{\frac{D}{2}}} \cdot (\chi^2)^{\frac{D}{2}-1} \cdot \exp -\frac{1}{2}\chi^2$$

with

$$E\{\chi^2\} = D \quad \text{mean}$$

$$Var\{\chi^2\} = 2D \quad \text{variance}$$

(without proof)

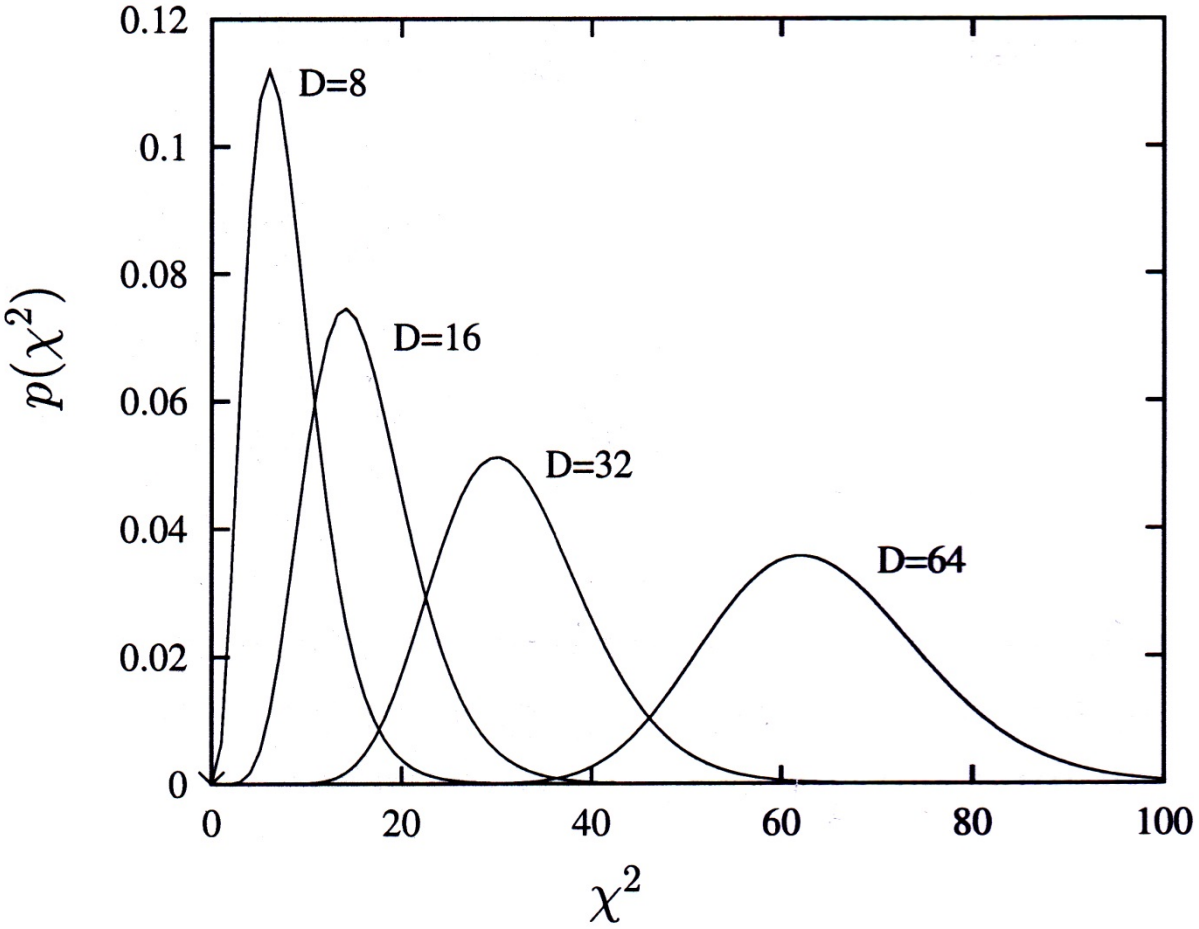
and $\Gamma\left(\frac{D}{2}\right) = \left(\frac{D}{2} - 1\right)!$ Gamma function

3.5 Distribution of Distances in $\mathbb{R}^D$

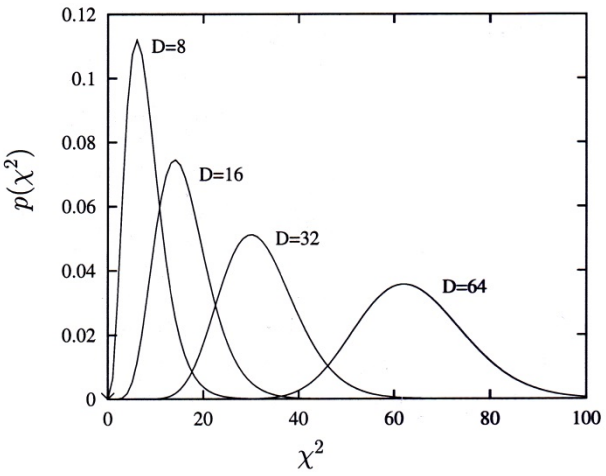
$$\chi^2 = \sum_{d=1}^{D} (x_d - \mu_{kd})^2$$

$$p(\chi^2) = \frac{1}{\Gamma(\frac{D}{2}) 2^{\frac{D}{2}}} \cdot (\chi^2)^{\frac{D}{2}-1} \cdot \exp -\frac{1}{2}\chi^2$$

Illustration



3.5 Distribution of Distances in $\mathbb{R}^D$



Consequences:

Distance of observations to 'their' mean increases with growing number of dimensions (increased variance and mean!)

→ **Concentration around the mean decreases if D increases.**

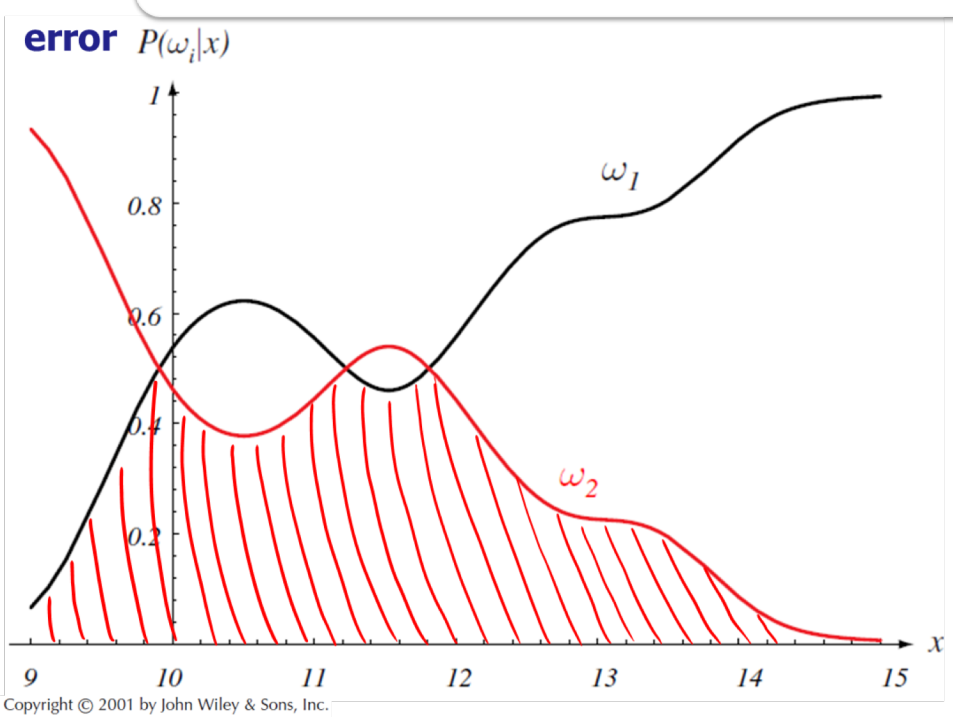
$$E\{\chi^2\} = D$$
$$Var\{\chi^2\} = 2D$$

3.6 Evaluation criteria

Theoretical error rate determination:

Mean error

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



→ Difficult since, in general, the form of the distributions is unknown

3.6 Evaluation criteria

Most important evaluation criterion: **Empirical error rate**

Error rate of the system when it is practically applied.

$$\text{(empirical) error rate} = \frac{\text{number of recognition errors}}{\text{number of recognition tests}}$$

3.6 Evaluation criteria

Most important evaluation criterion: **Empirical error rate**

We distinguish two sample types:

- **Training sample:** used for the design of the classifier
  - Choice of model $p(x|k, \Theta_k)$
  - Estimation of parameters Θ_k
- **Test sample:** used to measure the reliability of the classifier by means of the empirical error rate

3.6 Evaluation criteria

Hints:

- Strict separation of training and test data is mandatory for objective determination of the error rate

  Example: Multi-modal distribution with #modes = # training samples

  → zero error rate on training data (but not on test data)

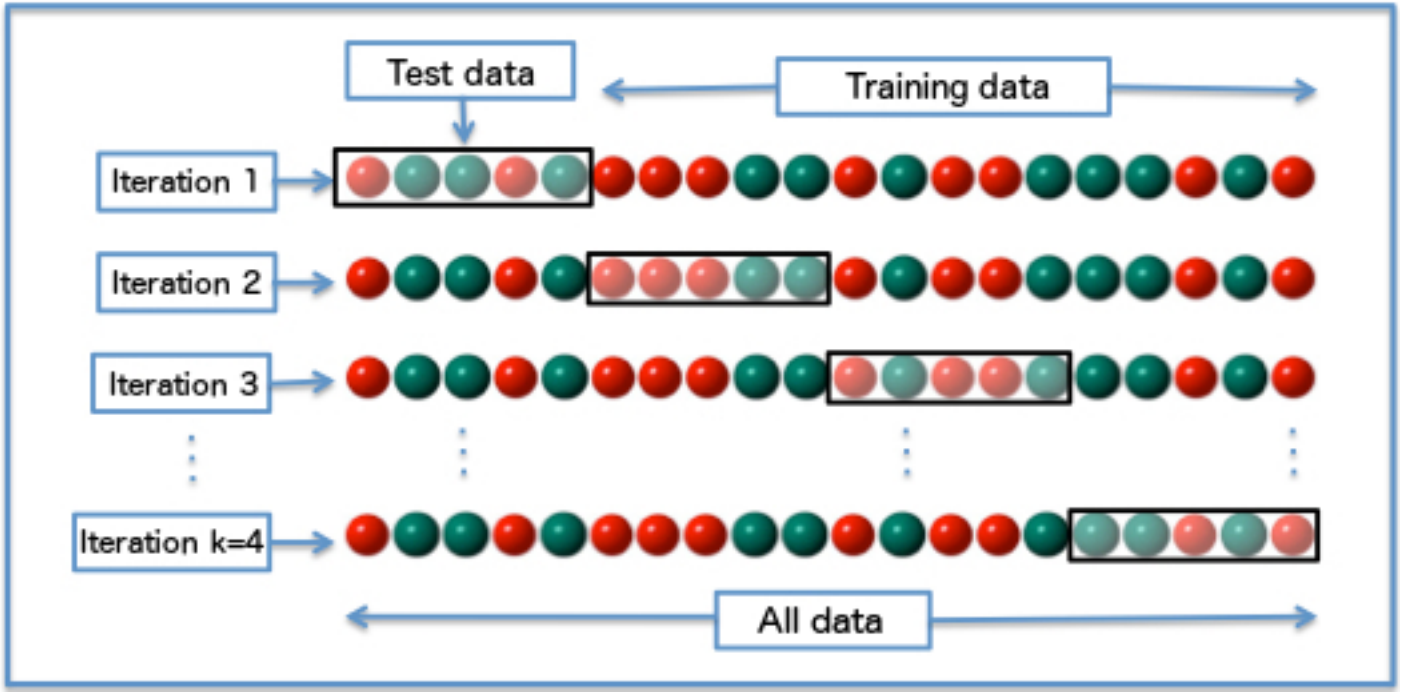
- Repeated tests on same test data may be delusive since system is optimized on this special test data. → **Training on test data**
- Scientific objectivity requires a test on unseen test set.

  → **Cross-Validation**

3.6 Evaluation criteria

Cross-Validation:

- Goal: Predict quality of model when explicit validation set is not available



Cross-Validation Error Rate = Average error rate across all k trials

Leaving-One-Out: Special case with k = N (number of data points)