ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Pattern Recognition & Machine Learning

Bayes Decision Theory - Training

Panagiotis C. Petrantonakis

Assistant Professor

Dept. of Electrical and Computer Engineering

ppetrant@ece.auth.gr

Introduction

- We saw that we can design an optimal classifier if we know the priors $P(\omega_i)$ and the class-conditional densities $p(\boldsymbol{x}|\omega_i)$.

- In practice, we rarely have this probabilistic structure of a problem.

- But we have…

Introduction

- We saw that we can design an optimal classifier fi we know the priors $P(\omega_i)$ and the class-conditional densities $p(\boldsymbol{x}|\omega_i)$.

- In practice, we rarely have this probabilistic structure of a problem.

- But we have…training data!!

Introduction

- We saw that we can design an optimal classifier fi we know the priors $P(\omega_i)$ and the class-conditional densities $p(\boldsymbol{x}|\omega_i)$.

- In practice, we rarely have this probabilistic structure of a problem.

- But we have…training data!!

- Thus, how can we use them to train our classifier?
  - Maximum likelihood estimation
  - Bayesian Estimation
  - Non-Parametric Techniques

- First two: parameterize conditional density. Then estimate parameters.
- Last one: end up with arbitrary distributions.

Independent Identically Distributed (i.i.d.) samples

- Suppose we have c datasets $D_1, \dots, D_c$.

- We say that: samples D_j are i.i.d. if they have been drawn independently from the same distribution $p(\boldsymbol{x}|\omega_j)$.

- Assuming that $p(\boldsymbol{x}|\omega_j)$ has a known parametric form, e.g. $p(\boldsymbol{x}|\omega_j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$:

$$p\left(\boldsymbol{x}\middle|\omega_j\right) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_j\right|^{1/2}} exp\left[-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)^t \boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)\right]$$

- We can write $p(\boldsymbol{x}|\omega_j)$ as $p(\boldsymbol{x}|\omega_j, \boldsymbol{\theta}_j)$ to show the dependence of $p(\boldsymbol{x}|\omega_j)$ on $\boldsymbol{\theta}_j$ where $\boldsymbol{\theta}_j$ consists of the components of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$.

Maximum Likelihood (ML) Estimation

- Training: Our problem is to use the information provided by the training samples to obtain estimates for the unknown parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \ldots, c$ associated with each class.

- ML is such a method with some attractive attributes
  - good convergence: especially with large training sets
  - simple (compared to Bayesian approach and non-parametric techniques)

- To further simplify things we assume:
  - D_i gives no information about $\boldsymbol{\theta}_j$ if $i \neq j$. Thus the parameters for different classes are independent.

- With the above assumption we have c separate problems of the form:
  - Use a set D of training samples, drawn i.i.d from $p(\boldsymbol{x}|\boldsymbol{\theta})$, to estimate $\boldsymbol{\theta}$.

Maximum Likelihood (ML) Estimation

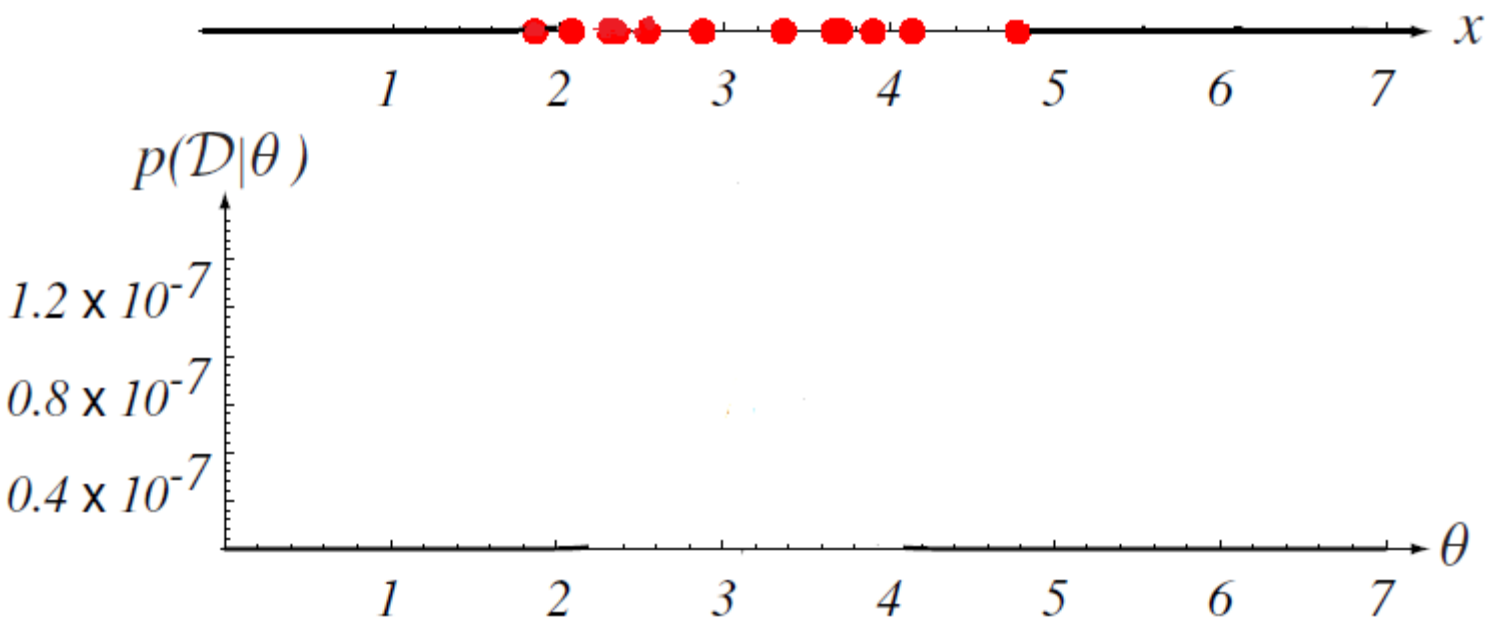
- Suppose D contains N samples, *i.e.*, $\boldsymbol{x}_n, n = 1, \ldots, N$.
- Because the samples are i.i.d. we have:

$$p(D|\boldsymbol{\theta}) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- Viewed as a function of $\boldsymbol{\theta}$, $p(D|\boldsymbol{\theta})$ is called the likelihood of $\boldsymbol{\theta}$ with respect to the samples of the dataset D (training set).
- The ML estimation of $\boldsymbol{\theta}$, i.e., $\widehat{\boldsymbol{\theta}}_{ML}$, is the value of $\boldsymbol{\theta}$ that maximizes $p(D|\boldsymbol{\theta})$
- Intuitively, $\widehat{\boldsymbol{\theta}}_{ML}$ best supports the observed data.

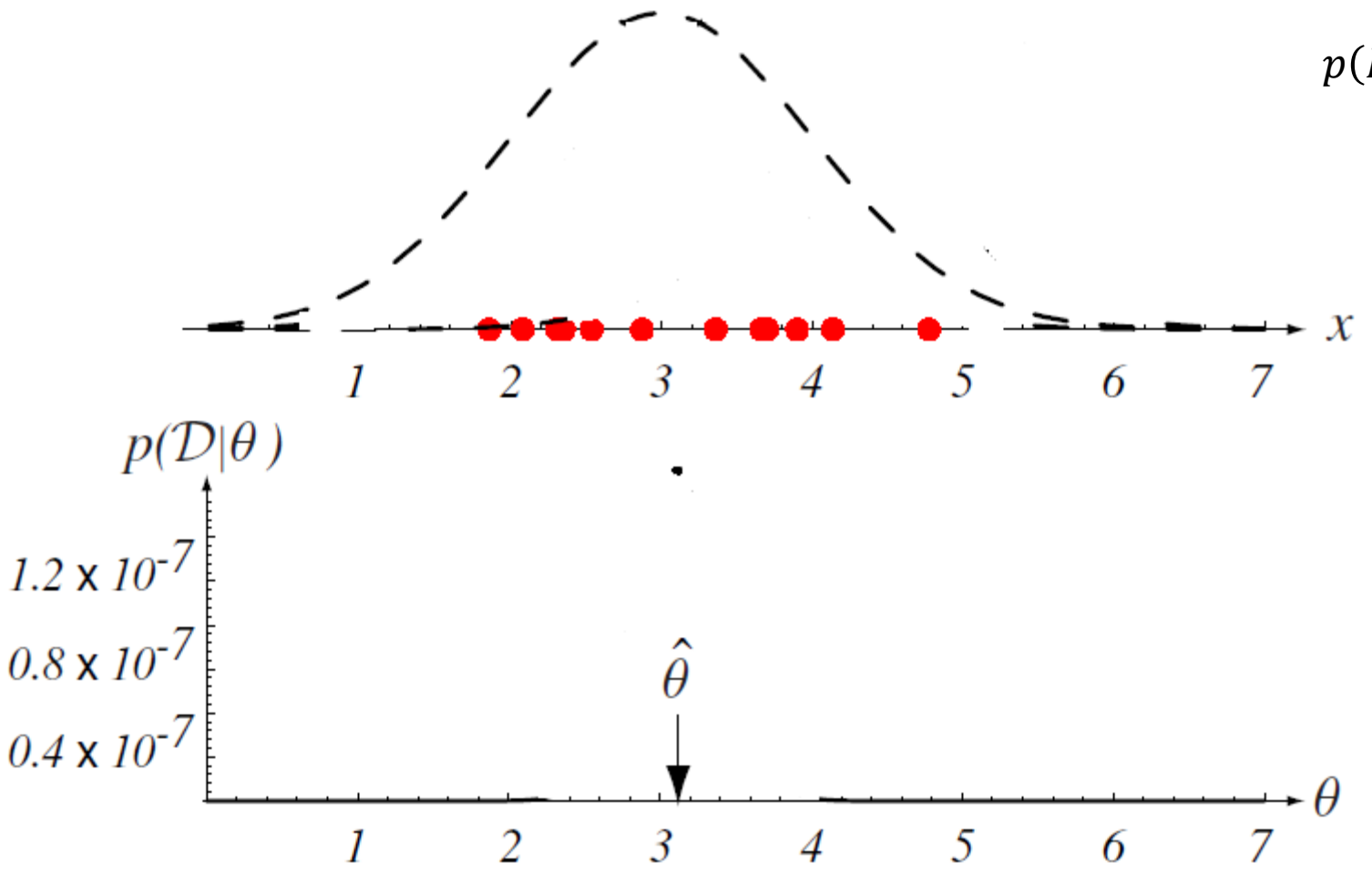
Maximum Likelihood (ML) Estimation -Example

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$



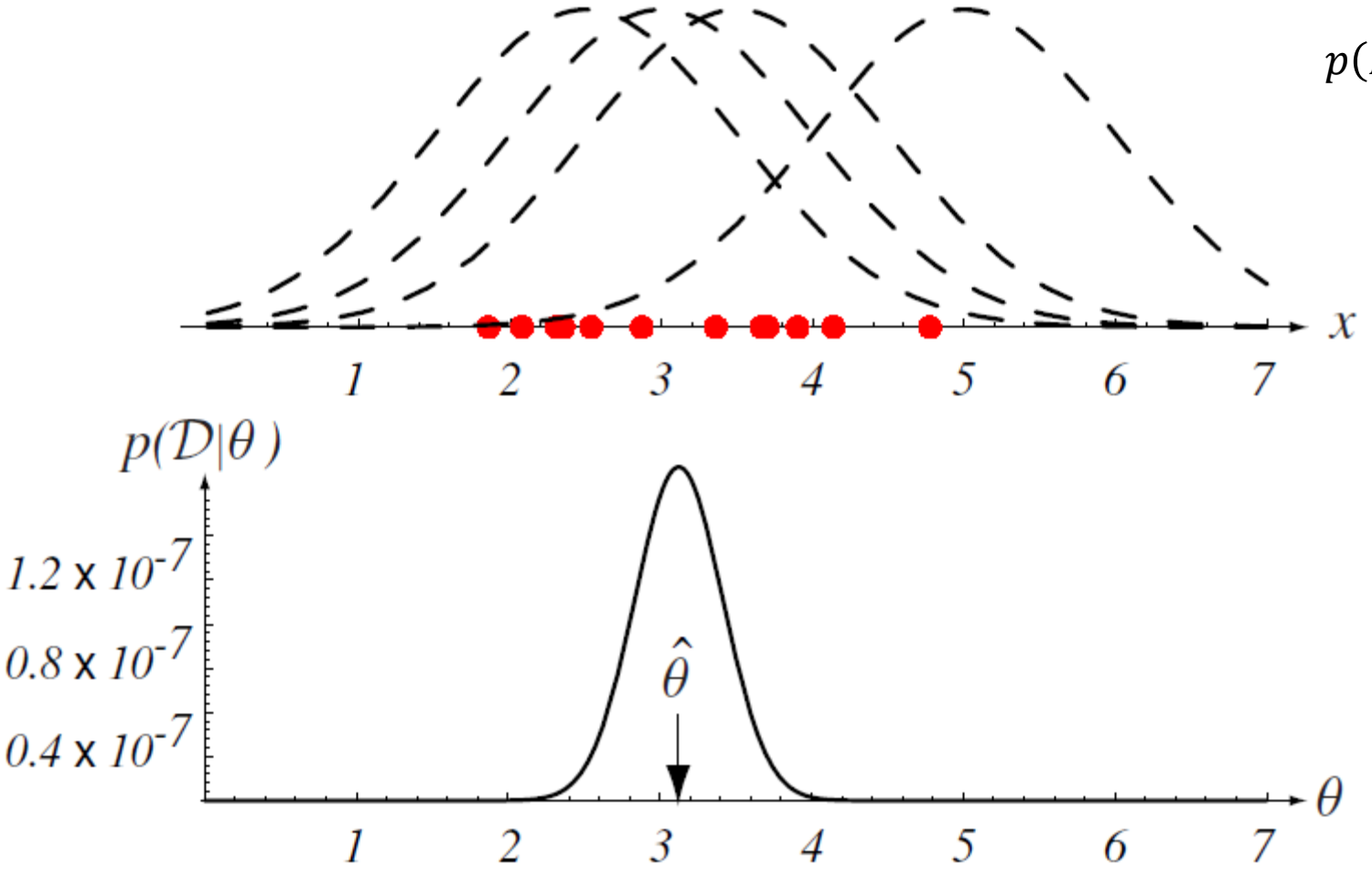
Maximum Likelihood (ML) Estimation -Example

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$



Maximum Likelihood (ML) Estimation -Example

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

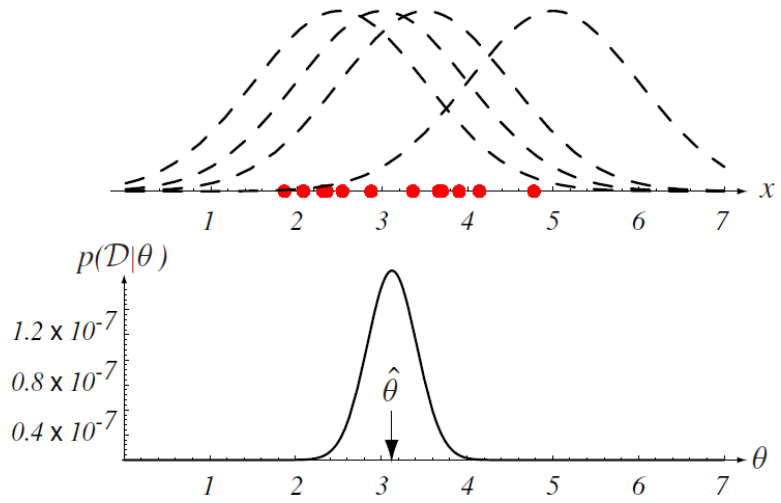


Duda, Hart, Stork

Maximum Likelihood (ML) Estimation -Example

- If we had a very large number of training points, what would happen to the likelihood?

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

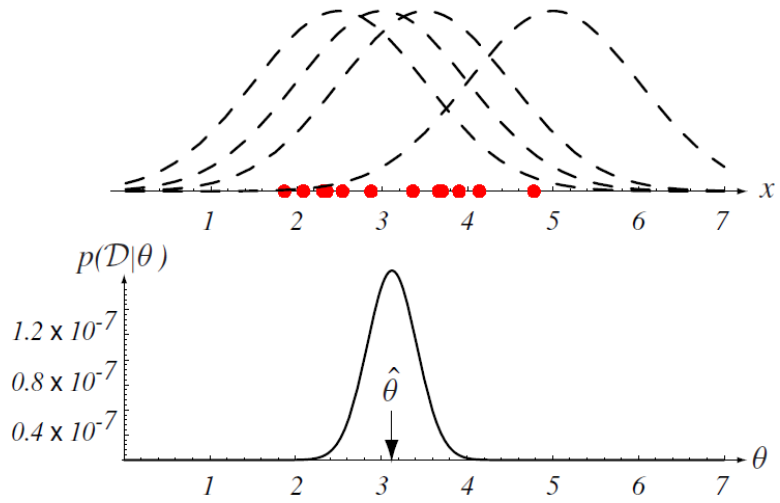


Duda, Hart, Stork

Maximum Likelihood (ML) Estimation -Example

- If we had a very large number of training points, what would happen to the likelihood?
  - It would be very narrow!

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$



Duda, Hart, Stork

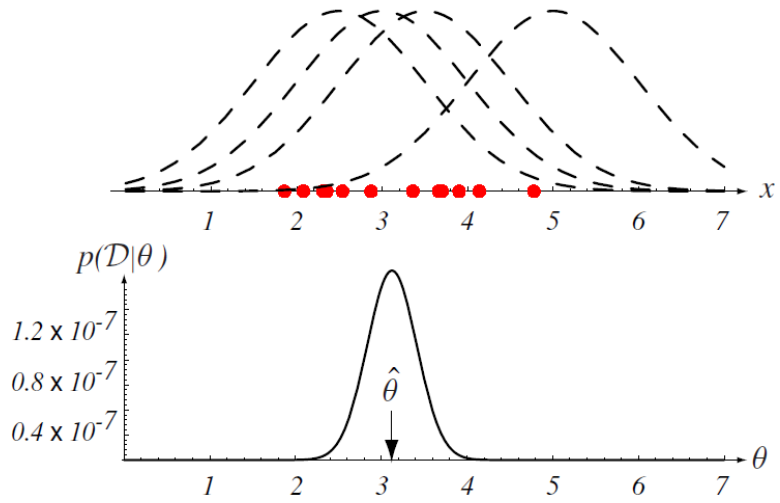
Maximum Likelihood (ML) Estimation -Example

- If we had a very large number of training points, what would happen to the likelihood?
  - It would be very narrow!

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- Notice that:
  - the likelihood is shown as a function of θ (in this case the mean of the gaussian distribution) whereas the conditional density is shown as a function of x.

- As a likelihood, i.e., a function of θ, it is not a probability density function and thus its area has no significance.



Duda, Hart, Stork

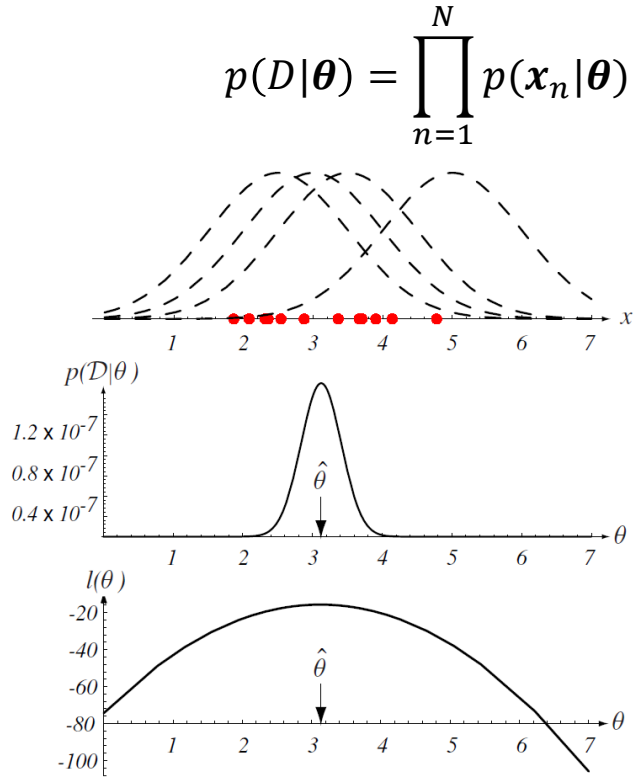
Maximum Likelihood (ML) Estimation

- By definition, $\widehat{\boldsymbol{\theta}}_{ML}$, is the vector that maximizes likelihood:

$$p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- Thus it needs to:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) = 0$$



Duda, Hart, Stork

- It is usually easier to work with logarithms.

- As the logarithm is a monotonically increasing function, if $\widehat{\boldsymbol{\theta}}_{ML}$ maximizes the likelihood function it also maximizes the log-likelihood, i.e., $l(\boldsymbol{\theta}) = \ln p(D|\boldsymbol{\theta})$.

Log-likelihood

- We have the gradient operator: $\boldsymbol{\nabla_\theta} = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ \vdots \\ \frac{\partial}{\partial \theta_d} \end{bmatrix}$
- Using the log-likelihood

$$l(\boldsymbol{\theta}) = \ln \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- We have:

$$\boldsymbol{\nabla_\theta} l(\boldsymbol{\theta}) = \boldsymbol{\nabla_\theta} \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \sum_{n=1}^{N} \boldsymbol{\nabla_\theta} \ln p(\boldsymbol{x}_n|\boldsymbol{\theta}) = 0$$

- Thus, we have a set of ? equations: $\boldsymbol{\nabla_\theta} l(\boldsymbol{\theta}) = \mathbf{0}$, from which we can find $\widehat{\boldsymbol{\theta}}_{ML}$

Log-likelihood

- We have the gradient operator: $\boldsymbol{\nabla_\theta} = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ \vdots \\ \frac{\partial}{\partial \theta_d} \end{bmatrix}$
- Using the log-likelihood

$$l(\boldsymbol{\theta}) = \ln \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- We have:

$$\boldsymbol{\nabla_\theta} l(\boldsymbol{\theta}) = \boldsymbol{\nabla_\theta} \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \sum_{n=1}^{N} \boldsymbol{\nabla_\theta} \ln p(\boldsymbol{x}_n|\boldsymbol{\theta}) = 0$$

- Thus, we have a set of d equations: $\boldsymbol{\nabla_\theta} l(\boldsymbol{\theta}) = \mathbf{0}$, from which we can find $\widehat{\boldsymbol{\theta}}_{ML}$

Maximum Likelihood (ML) Estimation

- Recap:
  - Assuming a probabilistic model for our data, e.g., $p(\boldsymbol{x}|\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, we can use the corresponding training datasets to find the optimum parameters for this model.
  - We do so by solving the system $\boldsymbol{\nabla}_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \mathbf{0}$

- Given that we have selected a reliable model, the ML classifier can give excellent results!

- What if the model is wrong, i.e., it does not adequately represent the underlying distributions?
  - Then the results will be probably poor!

- If the assumed model is poor we cannot assure that the classifier that we derive is the best!

Bayesian Estimation (or Bayesian Learning)

- The answers that we get from BE are nearly identical with those of ML. Nevertheless, there is a fundamental difference between the two approaches:
  - In ML the parameter's value is fixed, whereas in BE we consider θ to be a random variable.

Bayesian Estimation (or Bayesian Learning)

- The answers that we get from BE are nearly identical with those of ML. Nevertheless, there is a fundamental difference between the two approaches:
  - In ML the parameter's value is fixed, whereas in BE we consider θ to be a random variable.
- Training data allow us to convert the distribution of the parameter under consideration, i.e., $p(\theta)$, into a posterior probability.
- Bayes: $P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$

Bayesian Estimation (or Bayesian Learning)

- The answers that we get from BE are nearly identical with those of ML. Nevertheless, there is a fundamental difference between the two approaches:
  - In ML the parameter's value is fixed, whereas in BE we consider θ to be a random variable.
- Training data allow us to convert the distribution of the parameter under consideration, i.e., $p(\theta)$, into a posterior probability.
- Bayes: $P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$
- Thus, assuming that the conditional density and the a priori probabilities are unknown, how can we estimate a posteriori probabilities using training data?
- In other words, again, if we have c datasets $D_1, \dots, D_c$, one for each class, we want to find:

$$P(\omega_i|\boldsymbol{x}, D_i) = \frac{p(\boldsymbol{x}|\omega_i, D_i)P(\omega_i, D_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j, D_j)P(\omega_j, D_j)}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D_i) = \frac{p(\boldsymbol{x}|\omega_i, D_i)P(\omega_i|D_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j, D_j)P(\omega_j|D_j)}$$

- We need to determine both the class-conditional densities along with the prior probabilities.

- We can assume that the priors are known or easily computed by training data. Thus the above equation becomes:

$$P(\omega_i|\boldsymbol{x}, D_i) = \frac{p(\boldsymbol{x}|\omega_i, D_i)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j, D_j)P(\omega_j)}$$

As we did in ML case, the samples of D_i have no influence on the estimation of the parameter set of $p(\boldsymbol{x}|\omega_j, D_j)$ if $i \neq j$.

Bayesian Estimation

- For simplicity we rewrite the equation:

$$P(\omega_i|\boldsymbol{x}, D_i) = \frac{p(\boldsymbol{x}|\omega_i, D_i)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j, D_j)P(\omega_j)}$$

as:

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

where D is the training set for a particular class i and $p(\boldsymbol{x})$ is the probability density of that class.

- Although the desired probability density $p(\boldsymbol{x})$ is unknown we assume (again, as in ML) that it has a known parametric form, i.e., $p(\boldsymbol{x}|\boldsymbol{\theta})$.

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- Any information that we may have about $\boldsymbol{\theta}$, prior to observing our data, is contained in a known prior density $p(\boldsymbol{\theta})$.

- Once we have observed our samples (data) we can get the posterior density $p(\boldsymbol{\theta}|D)$ which we hope that is going to be sharply shaped around the true value of $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int p(D|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- Any information that we may have about $\boldsymbol{\theta}$, prior to observing our data, is contained in a known prior density $p(\boldsymbol{\theta})$.
- Once we have observed our samples (data) we can get the posterior density $p(\boldsymbol{\theta}|D)$ which we hope that is going to be sharply shaped around the true value of $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int p(D|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

- Our goal now is to compute $p(\boldsymbol{x}|D)$, which is as close as we can come to knowing the real $p(\boldsymbol{x})$.
- How can we accomplish that?

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- We take $p(\boldsymbol{x}|D)$ by marginalizing the joint density $p(\boldsymbol{x}, \boldsymbol{\theta}|D)$ over $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}, \boldsymbol{\theta}\ |D)d\boldsymbol{\theta}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- We take $p(\boldsymbol{x}|D)$ by marginalizing the joint density $p(\boldsymbol{x}, \boldsymbol{\theta}|D)$ over $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}, \boldsymbol{\theta}\,|D)d\boldsymbol{\theta}$$

- If we use the product rule we can rewrite the above equation as:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta}, D)p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- We take $p(\boldsymbol{x}|D)$ by marginalizing the joint density $p(\boldsymbol{x}, \boldsymbol{\theta}|D)$ over $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}, \boldsymbol{\theta}\,|D)d\boldsymbol{\theta}$$

- If we use the product rule we can rewrite the above equation as:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta}, D)p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

and because the selection of $\boldsymbol{x}$ and D are done independently, we get:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- We take $p(\boldsymbol{x}|D)$ by marginalizing the joint density $p(\boldsymbol{x}, \boldsymbol{\theta}|D)$ over $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}, \boldsymbol{\theta}\,|D)d\boldsymbol{\theta}$$

- If we use the product rule we can rewrite the above equation as:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta}, D)p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

and because the selection of $\boldsymbol{x}$ and D are done independently, we get:

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- Thus we have the class conditional density

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

linked to the posterior density:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

where $p(D|\boldsymbol{\theta})$ is again the…

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x}, D) = \frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|D)P(\omega_j)}$$

- Thus we have the class conditional density

$$p(\boldsymbol{x}|D) = \int p(\boldsymbol{x}\,|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

linked to the posterior density:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

where $p(D|\boldsymbol{\theta})$ is again the… likelihood $p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$

Bayesian Estimation

$$P(\omega_i|\boldsymbol{x},D)=\frac{p(\boldsymbol{x}|D)P(\omega_i)}{\sum_{j=1}^{c}p(\boldsymbol{x}|D)P(\omega_j)}$$

- These equations constitute the formal solution to the BE problem.

$$p(\boldsymbol{x}|D)=\int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

- They illustrate the relation to the ML estimation approach.
  - Suppose $p(D|\boldsymbol{\theta})$ reaches a sharp peak at $\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}$.
  - if $p(\boldsymbol{\theta})\neq 0$ at $\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}$ and it is smooth, then $p(\boldsymbol{\theta}|D)$ also peaks at that particular point
  - thus $p(\boldsymbol{x}|D)$ will be approximately equal to $p(\boldsymbol{x}|\widehat{\boldsymbol{\theta}})$
  - This is the result that one would take when following the ML approach.

$$p(\boldsymbol{\theta}|D)=\frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

$$p(D|\boldsymbol{\theta})=\prod_{n=1}^{N}p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

- Thus, if $p(D|\boldsymbol{\theta})$ is very sharp the influence of the prior information on the uncertainty of the value of $\boldsymbol{\theta}$ can be ignored.

- Nevertheless, in the general case, the Bayesian solution tells us how to use all the available information to compute $p(\boldsymbol{x}|D)$!

ML vs. BE

- Computational Complexity:
  - ML: merely differential calculus or gradient search
  - BE: complex multidimensional integration
- Interpretability:
  - ML: solution would be easier to interpret as it returns a single model from the set a designer provided
  - BE: the solution is a weighted average of models (may end up with a model that was not initially in the designer's set, i.e., assume uniform distribution and end up with another one)
- Performance:
  - ML: does not take into consideration information about the prior distribution of the parameters
  - BE: takes into account this information and leads to better solutions
- In summary, there are strong theoretical arguments supporting BE, though, in practice ML is simpler and can also lead to classifiers nearly as accurate.

Until now…

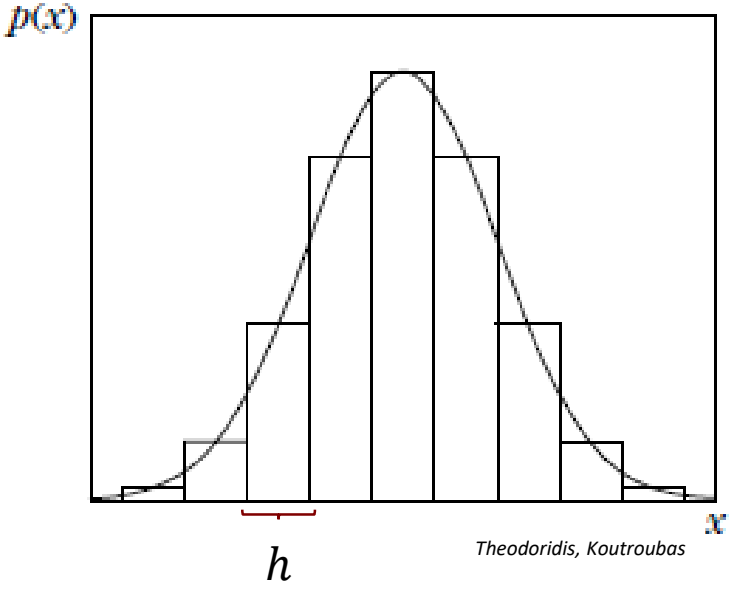
- We talked about supervised learning of Bayes classifiers assuming that the forms of the underlying density functions are known (in some sort of parametric form)!
- Nevertheless, the common parametric forms rarely fit real data. In particular, most pdfs are unimodal (they have single local maximum).
- Many practical applications require multimodal density functions.
- We will examine now *non-parametric* techniques that can lead to arbitrary shapes of distributions.
- With non-parametric techniques we make no assumptions about the forms of the underlying distributions.

Density estimation

- There is a variety of density estimation methods via non-parametric approaches.
- Here we will talk about two of them:
  - Parzen windows
  - k-Nearest Neighbor
- Both approaches are variations of the *histogram* approach to density estimation
- We will start with a simple example on density approximation of a single continuous variable x using histogram approach.

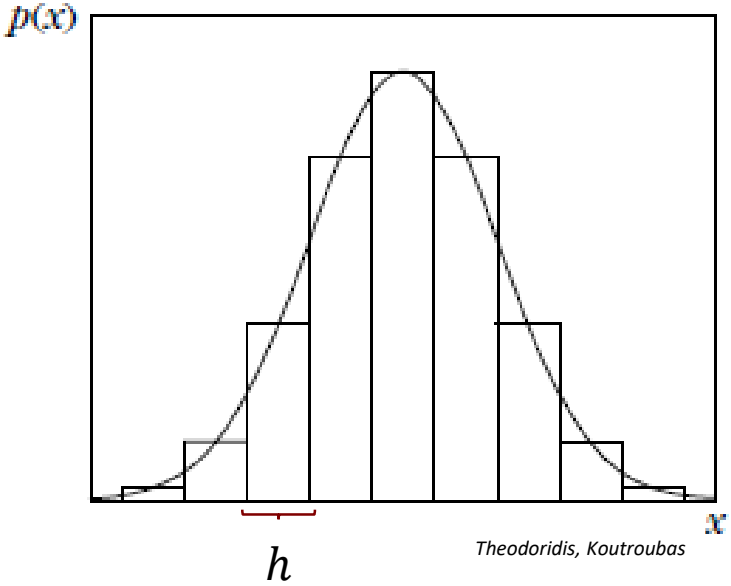
Density Estimation – Histogram approach

- The x-axis (one feature) is first divided into successive bins of length h.
- The probability of a sample x being located in a bin is estimated for each of the bins.
- If N is the total number of samples and k_N of these are located inside a bin, the corresponding probability is the *frequency ratio:* $P \approx \frac{k_N}{N}$
- This approximation converges to true P as $N \to \infty$.
- The pdf value is assumed constant within a bin and is approximately $\hat{p}(x) = \hat{p}(\bar{x}) \approx \frac{1}{h}\frac{k_N}{N}, |x - \bar{x}| \leq \frac{h}{2}$ where $\bar{x}$ is the midpoint of the bin.



Density Estimation – Histogram approach

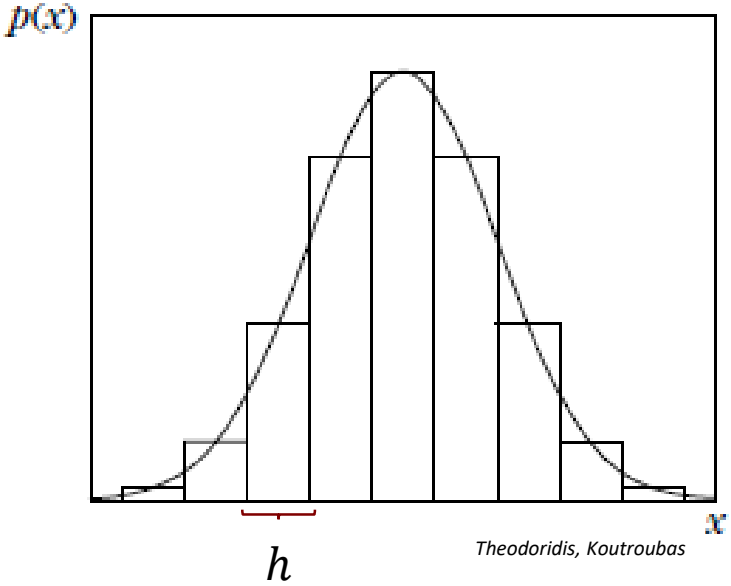
$$\hat{p}(x) \approx \frac{1}{h}\frac{k_N}{N}, |x - \bar{x}| \leq \frac{h}{2}$$



- Thus, this value determines the amplitude of the histogram curve (approximation of $p(x)$)
- We need relatively small h so as the assumption that $p(x)$ is constant within the bin holds.
- $\hat{p}(x)$ converges to $p(x)$ as $N \to \infty$ and:
  - $h_N \to 0$
  - $k_N \to \infty$
  - $\frac{k_N}{N} \to 0$
- By h_N we highlight the dependency of h on N

Density Estimation – Histogram approach

- $\hat{p}(x)$ converges to $p(x)$ as $N \to \infty$ and:
  - $h_N \to 0$ (we already justified this one)
  - $k_N \to \infty$
  - $\frac{k_N}{N} \to 0$



- $k_N \approx PN$ and k_N tends to infinity as $N \to \infty$
- As $h_N \to 0$ also $P \to 0$, thus $\frac{k_N}{N} \to 0$
- Thus, N must be "large enough," h_N "small enough," and k_N "large enough" too.
- How small and how large depend on the type of the pdf function and the degree of approximation one is satisfied with.

Parzen windows

- Multidimensional case:
  - Instead of bins we have hypercubes with length of side h

- Lets define the function:

$$\varphi(\boldsymbol{x}_i) = \begin{cases} 1\,, & \forall j, \left|x_{ij}\right| \leq \frac{1}{2} \\ \\ 0, & otherwise \end{cases}$$

where x_{ij} are the components of the feature vector $\boldsymbol{x}_i, i = 1, \ldots, N$.

- In words: $\varphi(\boldsymbol{x}_i)$ is 1 only if the feature vector is within a hypercube with side $h =$

Parzen windows

- Multidimensional case:
  - Instead of bins we have hypercubes with length of side h

- Lets define the function:

$$\varphi(\boldsymbol{x}_i) = \begin{cases} 1\,, & \forall j, \left|x_{ij}\right| \leq \frac{1}{2} \\ \\ 0, & otherwise \end{cases}$$

where x_{ij} are the components of the feature vector $\boldsymbol{x}_i, i = 1, \ldots, N$.

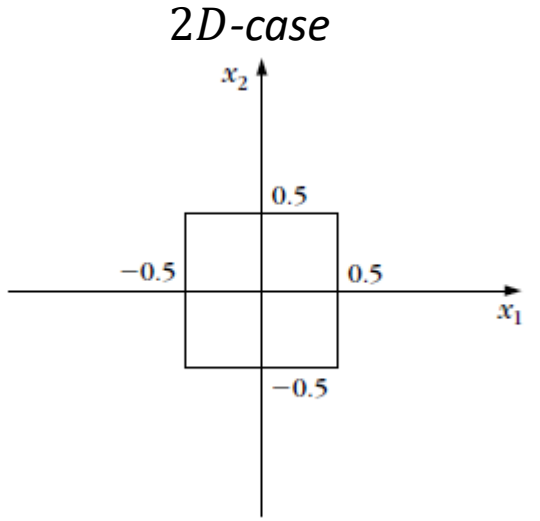
- In words: $\varphi(\boldsymbol{x}_i)$ is 1 only if the feature vector is within a hypercube with side $h = 1$ and center at the …

Parzen windows

- Multidimensional case:
  - Instead of bins we have hypercubes with length of side h

- Lets define the function:

$$\varphi(\boldsymbol{x}_i) = \begin{cases} 1\,, & \forall j, \left|x_{ij}\right| \leq \frac{1}{2} \\ \\ 0, & otherwise \end{cases}$$



Theodoridis, Koutroubas

where x_{ij} are the components of the feature vector $\boldsymbol{x}_i, i = 1, \ldots, N$.

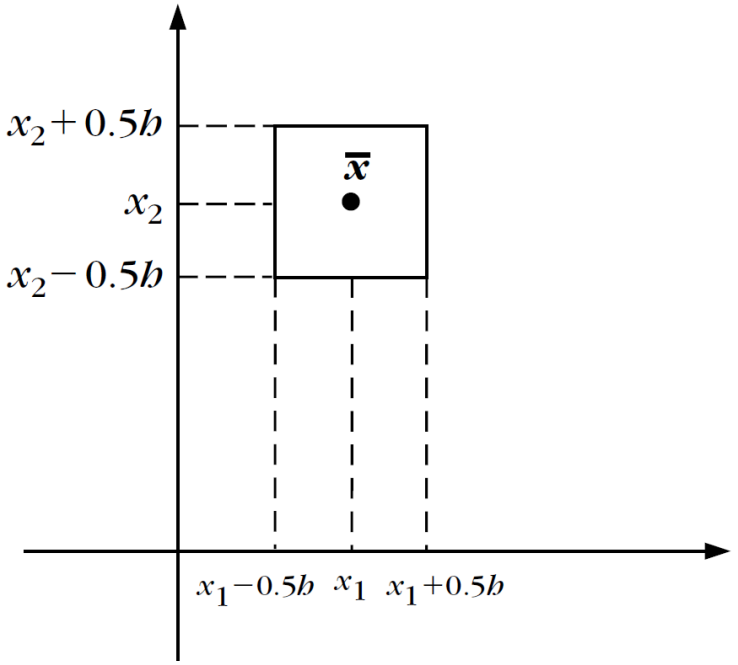
- In words: $\varphi(\boldsymbol{x}_i)$ is 1 only if the feature vector is within a hypercube with side $h = 1$ and center at the … origin and zero outside of the hypercube.

Parzen windows

- In the general case:

$$\varphi\left(\frac{\boldsymbol{x_i}-\bar{\boldsymbol{x}}}{h}\right) = \begin{cases} 1\,, & \forall j, \left|x_{ij} - \bar{x}_j\right| \leq \frac{h}{2} \\ 0, & otherwise \end{cases}$$

- Thus, how $\hat{p}(\bar{x}) \approx \frac{1}{h}\frac{k_N}{N}$, changes in order to account for the general case?

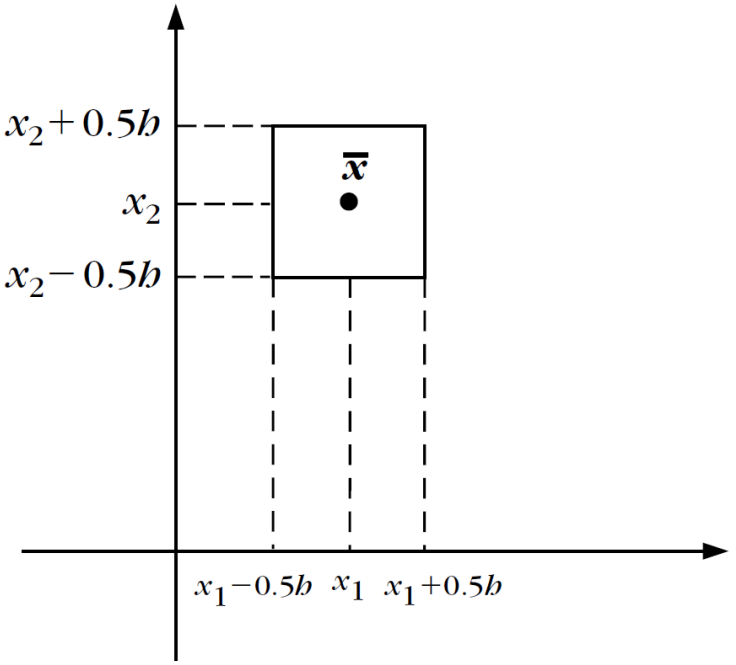


Theodoridis, Koutroubas

Parzen windows

- In the general case:

$$\varphi\left(\frac{\boldsymbol{x_i}-\bar{\boldsymbol{x}}}{h}\right) = \begin{cases} 1\,, & \forall j, \left|x_{ij} - \bar{x}_j\right| \leq \frac{h}{2} \\ 0, & otherwise \end{cases}$$



Theodoridis, Koutroubas

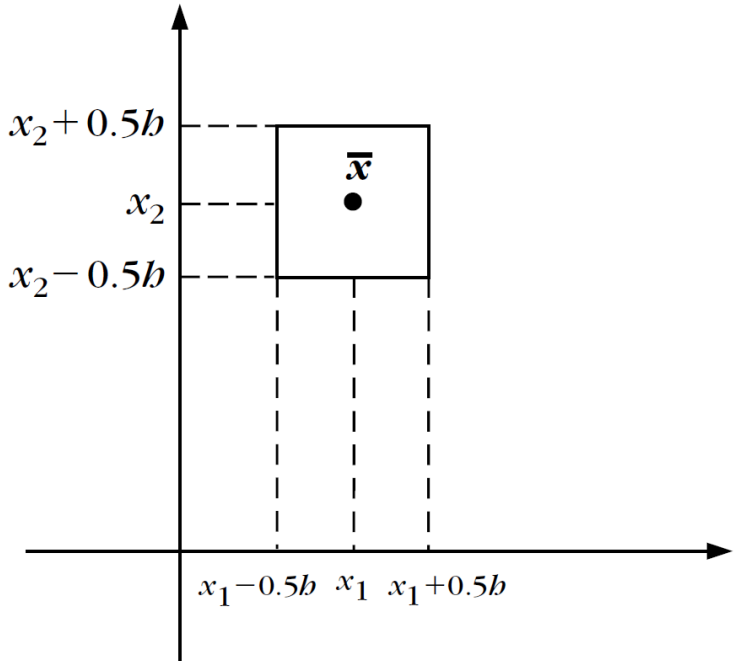
- Thus, how $\hat{p}(\bar{x}) \approx \frac{1}{h}\frac{k_N}{N}$, changes in order to account for the general case?

$$\hat{p}(x) \approx \frac{1}{h^d}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi\left(\frac{\boldsymbol{x_i}-\bar{\boldsymbol{x}}}{h}\right)\right)$$

- Interpretation: we estimate the pdf at point $\bar{\boldsymbol{x}}$ and within a hypercube with side h. The summation equals k_N

Parzen windows

$$\hat{p}(x) \approx \frac{1}{h^d}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi\left(\frac{\boldsymbol{x_i}-\bar{\boldsymbol{x}}}{h}\right)\right)$$



Theodoridis, Koutroubas

- There is an issue:
  - we try to estimate a continuous function $p(\boldsymbol{x})$ using an expansion of discontinuous terms, i.e., step function $\varphi(.)$
- This led Parzen to generalize the above approximation using smooth functions instead of discontinuous $\varphi(.)$
- Such smooth functions are known as *kernels* or *Parzen windows* and provided that $\varphi(\boldsymbol{x}) \geq 0$ and $\int_{\boldsymbol{x}} \varphi(\boldsymbol{x})d\boldsymbol{x} = 1$ this approach leads to good approximations.

Parzen windows – gaussian kernel

- If we use the gaussian kernel the approximation for the density $p(\boldsymbol{x})$ would be:

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{n} \frac{1}{(2\pi)^{\frac{d}{2}} h^d} exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i)}{2h^2}\right)$$

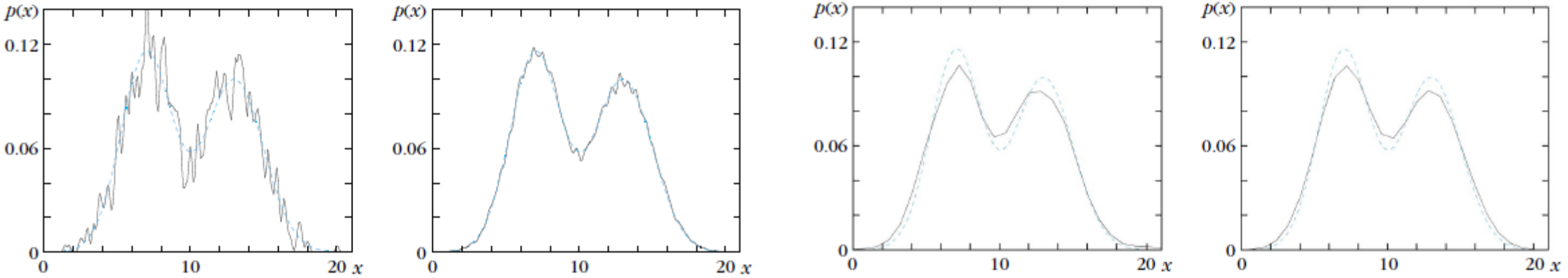
- In words: the unknown pdf is approximated as an *average* of N gaussians each one centered at a different point of the training set.

- The smaller the h the narrower the kernel will be and thus each individual gaussian is more "spiky" and localized around its mean value (here the feature vector under consideration).

- The larger the h the broader the kernel and the influence in the neighbor around its mean is more important.

Parzen windows – gaussian kernel - example

- In general we need to find the appropriate h value in accordance with the number of samples.



$h = 0.1, N = 1000$ $h = 0.1, N = 20000$ $h = 0.8, N = 1000$ $h = 0.8, N = 20000$

Theodoridis, Koutroubas

For fixed h the number of samples influence the smoothness of the resulting approximation.

For the same example, the number of samples has little influence on the approximation accuracy for a larger h

Classification using Parzen windows

- We have already seen that for the two-class classification we can minimize the overall risk by classifying ω_1 if:

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

With the Parzen approach we can estimate the corresponding density function of the first part and the above rule becomes:

$$\frac{\frac{1}{h^d}\left(\frac{1}{N_1}\sum_{n=1}^{N_1}\varphi\left(\frac{\boldsymbol{x}_n-\bar{\boldsymbol{x}}}{h}\right)\right)}{\frac{1}{h^d}\left(\frac{1}{N_2}\sum_{n=1}^{N_2}\varphi\left(\frac{\boldsymbol{x}_n-\bar{\boldsymbol{x}}}{h}\right)\right)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)}$$

- What is $\bar{\boldsymbol{x}}$, N_1 and N_2 in terms of the classification task?

Classification using Parzen windows

- We have already seen that for the two-class classification we can minimize the overall risk by classifying ω_1 if:

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

With the Parzen approach we can estimate the corresponding density function of the first part and the above rule becomes:

$$\frac{\frac{1}{h^d}\left(\frac{1}{N_1}\sum_{n=1}^{N_1}\varphi\left(\frac{\boldsymbol{x}_n-\overline{\boldsymbol{x}}}{h}\right)\right)}{\frac{1}{h^d}\left(\frac{1}{N_2}\sum_{n=1}^{N_2}\varphi\left(\frac{\boldsymbol{x}_n-\overline{\boldsymbol{x}}}{h}\right)\right)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)}$$

- $\overline{\boldsymbol{x}}$: sample to classify, N_1: number of samples in the training set for class ω_1 and N_2 : number of samples in the training set for class ω_2

k-Nearest Neighbor estimation

- In Parzen approach the volume around the sample $\boldsymbol{x}$ is considered fixed, whereas what changes is the number of samples that fall within this space.

- In k-NN approach the roles are reversed, i.e., we consider the number of the samples fixed, i.e., k, and the volume changes so as to determine the minimum volume needed to include k samples.
  - low density areas: big volume
  - high density areas: small volume
  - again we can thing of more complex regions than just the hypercube case, e.g. hypersphere.

- For the k-NN case the estimation of the density function is know:

$$\hat{p}(\boldsymbol{x}) = \frac{1}{V(\boldsymbol{x})}\frac{k}{N}$$

Classification using k-NN estimation

- During a classification of a test sample $\boldsymbol{x}$ we:
  - compute its distance (e.g., Euclidian) from all the training vectors (all classes, e.g., ω_1, ω_2).
  - determine r_1: radius of the hypersphere, centered at $\boldsymbol{x}$, that contains k samples of class ω_1
  - determine r_2: radius of the hypersphere, centered at $\boldsymbol{x}$, that contains k samples of class ω_2
  - estimate respective volumes, V_1 and V_2
  - assign $\boldsymbol{x}$ to class ω_1 if:

$$\frac{\frac{1}{V_1}\frac{k}{N_1}}{\frac{1}{V_2}\frac{k}{N_2}} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)} \quad \rightarrow \quad \frac{V_2}{V_1} > \frac{N_1}{N_2}\frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)}$$

- If we use Mahalanobis distance we should replace hyperspheres with…

Classification using k-NN estimation

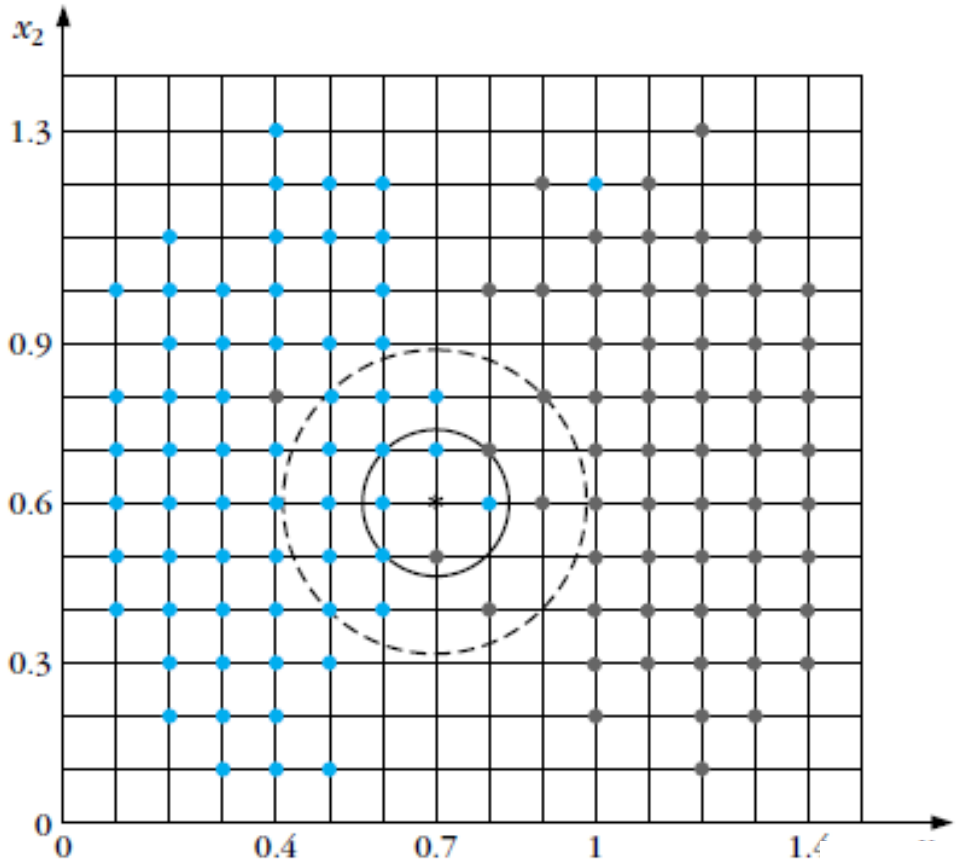
- During a classification of a test sample $\boldsymbol{x}$ we:
  - compute its distance (e.g., Euclidian) from all the training vectors (all classes, e.g., ω_1, ω_2).
  - determine r_1: radius of the hypersphere, centered at $\boldsymbol{x}$, that contains k samples of class ω_1
  - determine r_2: radius of the hypersphere, centered at $\boldsymbol{x}$, that contains k samples of class ω_2
  - estimate respective volumes, V_1 and V_2
  - assign $\boldsymbol{x}$ to class ω_1 if:

$$\frac{\frac{1}{V_1}\frac{k}{N_1}}{\frac{1}{V_2}\frac{k}{N_2}} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)} \quad \rightarrow \quad \frac{V_2}{V_1} > \frac{N_1}{N_2}\frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)}$$

- If we use Mahalanobis distance we should replace hyperspheres with…hyperellipsoids.

Classification using 5-NN estimation - example

- In which class would you assign the sample denoted by a star in the following figure with classes ω_1 (black dots) and ω_2 (blue dots) being equiprobable (assume simple costs)?



$$\frac{V_2}{V_1} > \frac{N_1}{N_2}\frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(\omega_2)}{P(\omega_1)}$$

Theodoridis, Koutroubas

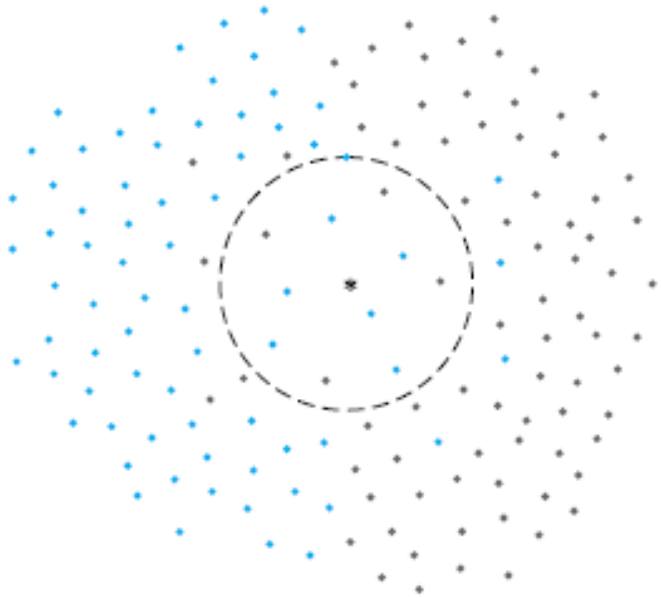
k-Nearest Neighbor Classifier

- A variation of the previous approach of k-NN based density estimation can lead to a suboptimal but useful, in practice, non-linear classification algorithm: the *k*-NN classifier

- The *k*-NN algorithm is summarized as follows:
  - From the N training samples identify k nearest neighbors (irrespectively of the class they belong to)
  - From these k samples, find the number of samples k_i that belong to class $\omega_i, i = 1, \dots, c$.
  - Assign $\boldsymbol{x}$ to the class with the maximum k_i.

- Various distance measures can be used

- Simplest form: $k = 1$, i.e., 1-NN classifier.



Theodoridis, Koutroubas

Error using k-NN classifier

- It can be shown that as $N \to \infty$ the classification error probability for the 1-NN classifier is bounded:

$$P_B \leq P_{1-NN} \leq \cdots \leq 2P_B$$

- Thus, the error committed by the 1-NN classifier is at most twice the optimal Bayes classifier.

- Moreover, the asymptotic performance of the k-NN classifier is better that 1-NN. For the two class case it can be shown that:

$$P_B \leq P_{k-NN} \leq P_B + \sqrt{\frac{2P_{1-NN}}{k}}$$

suggesting that for $k \to \infty$ the k-NN classifier tends to be optimal.

- Finally, for small Bayesian error it can be proven that: $P_{3-NN} \approx P_B + 3P_B^2$
  - Thus for $P_B = 0.01$ I get $P_{3-NN} \approx 0.0103$

Naïve Bayes Classifier

- In this lecture we presented various techniques for the estimation of the probability density function $p(\boldsymbol{x}|\omega_i), i = 1, \ldots, c$ based on the available training set.

- It may be obvious until now that the number of training samples plays crucial role for the abovementioned estimation.

- Due to *curse of dimensionality*, the bigger the number of dimensions the bigger the number of training samples needed for effective estimation.
  - Crudely speaking, if we need N samples for good estimation in one dimension, in d dimensions I would need N^d.

- Thus, big d makes good estimation of $p(\boldsymbol{x}|\omega_i)$ unrealistic.

- We could make some assumptions to alleviate this problem.

Naïve Bayes Classifier

- One widely used approach is to assume that individual features are independent.
- Under this assumption we can write:

$$p(\boldsymbol{x}|\omega_i) = \prod_{j=1}^{d} p\left(x_j|\omega_i\right), i = 1, \ldots, c$$

- Now, we only need to estimate d one-dimensional densities for each one of the c classes.
- This leads to the so called *Naïve Bayes* classifier which can be very robust to the violation of the *independence* assumption and deliver good results in many applications in practice.

Sum up…

- We saw various training techniques for the Bayes classifier
- In essence, training is just the estimation of the probability density that generate my data using the available training data
- Training approaches
  - Maximum likelihood
  - Bayesian Estimation (Bayesian Learning)
  - Non-Parametric Techniques
    - Parzen windows
    - k-NN estimation
- We also described an alteration of *k*-NN estimation that leads to the celebrated *k*-NN classifier
- Finally, we presented the Naïve Bayes approach that assumes different features are statistically independent.

Questions!

- Which aspect of the Bayes classifier is responsible for its inductive bias?

Questions!

- Which aspect of the Bayes classifier is responsible for its inductive bias?
  - probability density functions!
  - You, as a designer, can choose, these pdfs and as a result develop classifiers that have inductive biases based on what you believe/know about the problem under consideration!

Questions!

- Which aspect of the Bayes classifier is responsible for its inductive bias?
  - probability density functions!
  - You, as a designer, can choose, these pdfs and as a result develop classifiers that have inductive biases based on what you believe/know about the problem under consideration!
- Which aspect of the *k*-NN classifier is responsible for its inductive bias?

Questions!

- Which aspect of the Bayes classifier is responsible for its inductive bias?
  - probability density functions!
  - You, as a designer, can choose, these pdfs and as a result develop classifiers that have inductive biases based on what you believe/know about the problem under consideration!

- Which aspect of the *k*-NN classifier is responsible for its inductive bias?
  - label for an example should be similar to the label of nearby points.
  - all features are equally important (remember the difference with the DT)
  - if you have data with only a few relevant features and lots of irrelevant features, *k*-NN is likely to do poorly!

ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Questions?

Pattern Recognition & Machine Learning

Bayes Decision Theory - Training