

## Naive Bayes classification

## Introduction

- So far, we have seen how we can make decisions (classify) in the Bayesian framework

- Compute

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \quad \forall i = 1 \dots C$$

where  $p(\mathbf{x})$  might be ignored (as long as a probability value is not needed).

- Assign  $\mathbf{x}$  to the class that maximizes  $g_i(\mathbf{x})$ .
- We have supposed that all probability densities are known, which is *never* the case in practice.
  - We may *estimate* their parameters from *training sets*, when their analytic expression is known or estimate them in a nonparametric framework (see next chapters)
  - Now, let us introduce a simplified framework called “naive Bayes” classification

## Naive Bayes assumption

- One of the most practical learning methods
- Applies to moderate as well as large training sets
  - Discrete or continuous features
- Relies on a class-conditional independence of features

$\mathbf{x} = \{x_1, x_2, \dots, x_D\}$ , where the index denotes dimension

$$p(\mathbf{x}|\omega_i) = p(x_1|\omega_i) \times p(x_2|\omega_i) \times \dots \times p(x_D|\omega_i) = \prod_{d=1}^D p(x_d|\omega_i)$$

- The MAP discriminant function becomes

$$g_i(\mathbf{x}) = P(\omega_i) \prod_{d=1}^D p(x_d|\omega_i)$$

## Naive Bayes classification

- Training

- We can look-up all the probabilities with a single scan of the database
  - ✓ We suppose that we have  $N$  training samples
- For each class  $\omega_i$ ,  $i = 1 \dots C$ 
  - ✓ Estimate  $P(\omega_i) = N_i/N$  where  $N_i$  is the number of occurrences of class  $\omega_i$
  - ✓ For each feature  $x_d$ ,  $d = 1 \dots D$  estimate PDF,  $p(x_d|\omega_i)$

- Decision (classification)

- Assign  $\mathbf{x}$  to the class that maximizes

$$g_i(\mathbf{x}) = P(\omega_i) \prod_{d=1}^D p(x_d|\omega_i)$$

(computed using the learned probabilities)

- NB: may be better to take logs to avoid numerical problems
- NB: if  $p(\mathbf{x})$  is needed, it may be computed as  $p(\mathbf{x}) = \sum_{i=1}^C P(\omega_i) \prod_{d=1}^D p(x_d|\omega_i)$

## Remarks

- **Only 1D probabilities are involved**
  - Easier to model using frequencies (for symbolic or discrete attributes), or parametric/non-parametric density estimation (for continuous ones).
  - Fast and space efficient!
- **Conditional independence assumption is often violated**
  - Hence the name “Naïve Bayes” or “Idiot Bayes”!
  - But it works surprisingly well anyway...
- **Successful applications:**
  - Diagnosis
  - Text classification (e.g. spam detection,...)

## Some words about feature independence

### Feature independence

$$p(x_1, x_2) = p(x_1|x_2)p(x_2) = p(x_2|x_1)p(x_1) = p(x_1)p(x_2)$$

- The knowledge of  $x_1$  (resp.  $x_2$ ) brings no extra information about  $x_2$  (resp.  $x_1$ )

### Class-conditional feature independence

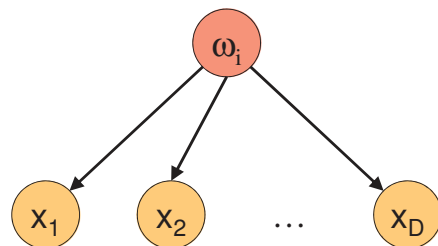
$$p(x_1, x_2|\omega_i) = p(x_1|x_2, \omega_i)p(x_2|\omega_i) = p(x_2|x_1, \omega_i)p(x_1|\omega_i) = p(x_1|\omega_i)p(x_2|\omega_i)$$

### An example (symbolic features)

$$\begin{aligned} P(\text{rainy, windy}|\text{storm}) &= P(\text{rainy}|\text{windy, storm}).P(\text{windy}|\text{storm}) \\ &= P(\text{rainy}|\text{storm}).P(\text{windy}|\text{storm}) \end{aligned}$$

## Graphical representation

- The naive Bayes model is often represented as a directed graph



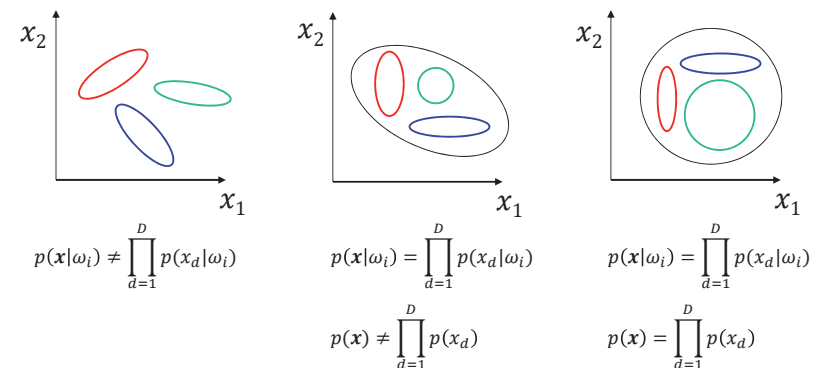
- The arrows state that each class causes certain features, with a certain probability

Following [Bishop]

## Class-conditionally independent features

- Feature independence vs. class-conditional feature independ.

- Iso-probability lines (Gaussian PDF's),  $C = 3$ ,  $D = 2$



Following [Gutierrez]

## A famous example (from Machine Learning, Mitchell)

Day	Outlook	Temperature	Humidity	Wind	Play tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

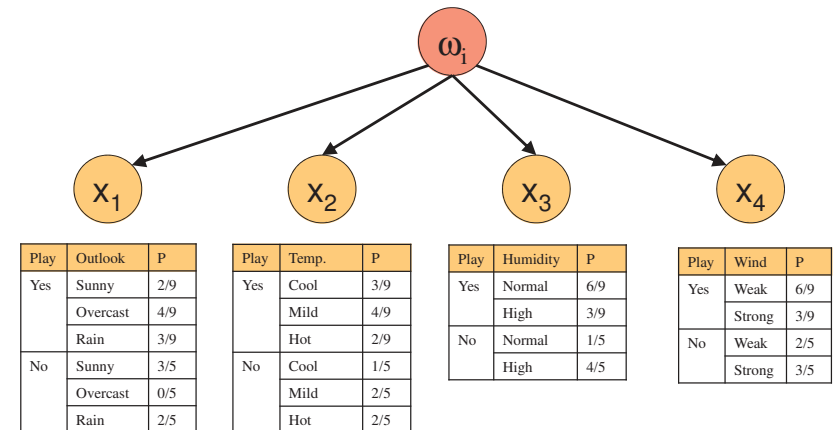
- Predict the target value (play=yes or no) for observation:

Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong

## A famous example (from Machine Learning, Mitchell)

- Probability estimation

- $P(\text{Play=yes})=9/14$ ,  $P(\text{Play=no})=5/14$



## A famous example (from Machine Learning, Mitchell)

- $P(\text{Play=yes})=9/14$ ,  $P(\text{Play=no})=5/14$

Play	Outlook	P	Play	Temp.	P	Play	Humidity	P	Play	Wind	P			
Yes	Sunny	2/9	Yes	Cool	3/9	Yes	Normal	6/9	Yes	Weak	6/9			
	Overcast	4/9			Mild		4/9			High	3/9		Strong	3/9
	Rain	3/9			Hot		2/9							
No	Sunny	3/5	No	Cool	1/5	No	Normal	1/5	No	Weak	2/5			
	Overcast	0/5			High		4/5			Strong	3/5			
	Rain	2/5			Mild		2/5							
				Hot	2/5									

$$x = \{\text{sunny, cool, high, strong}\}$$

$$g_{\text{yes}}(x) = P(\text{yes}) \times P(\text{sunny}|\text{yes}) \times P(\text{cool}|\text{yes}) \times P(\text{high}|\text{yes}) \times P(\text{strong}|\text{yes})$$

$$g_{\text{yes}}(x) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = \frac{2}{14 \times 27} \approx 0.0053$$

$$g_{\text{no}}(x) = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = \frac{36}{14 \times 75} \approx 0.0206$$

It's a "no play" day!  
...with probability:

$$P(\text{no}|x) = \frac{0.0206}{0.0053 + 0.0206} \approx 0.795$$

## Dealing with zero probabilities

- $P(\text{play=yes})=9/14$ ,  $P(\text{play=no})=5/14$

Play	Outlook	P
Yes	Sunny	2/9
	Overcast	4/9
	Rain	3/9
No	Sunny	3/5
	Overcast	0/5
	Rain	2/5

Play	Temp.	P
Yes	Cool	3/9
	Mild	4/9
	Hot	2/9
No	Cool	1/5
	Mild	2/5
	Hot	2/5

Play	Humidity	P
Yes	Normal	6/9
	High	3/9
No	Normal	1/5
	High	4/5

Play	Wind	P
Yes	Weak	6/9
	Strong	3/9
No	Weak	2/5
	Strong	3/5

- Problem: artificial zero probabilities

$$x = \{\text{overcast, cool, high, strong}\}$$

$$g_{\text{no}}(x) = \frac{5}{14} \times \frac{0}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0 !!!$$

## Dealing with zero probabilities

- Solution: use *Laplace smoothing* (a.k.a. *additive smoothing*)

$$p(x_d | \omega_i) = \frac{\text{count}(x_d, \omega_i) + 1}{\sum_{\text{possible } x_d} [\text{count}(x_d, \omega_i) + 1]} = \frac{\text{count}(x_d, \omega_i) + 1}{N_i + k_d}$$

where

$N_i$  is the number of samples that belong to class  $\omega_i$

$k_d$  is the number of possible values for  $x_d$

- Sometimes, an  $\alpha$  weight is used instead of 1, leading to  $\alpha k_d$  (denominator)

- In our example, we obtain

No	Sunny	3/5
	Overcast	0/5
	Rain	2/5




No	Sunny	4/8
	Overcast	1/8
	Rain	3/8

## A specific example: text classification

- Applications of text classification are many
  - Spam detection
  - Language identification
  - Authorship identification
  - Assigning subject categories, topic, genres
  - ...
- Building and maintaining hand-coded rules may be expensive
  - Many Pattern Recognition algorithms may be suited
  - Naïve Bayes is a good baseline
- Naïve Bayes text classification relies on the *bag-of-words* representation

## The bag-of-words (BoW) representation

- Simplified representation
- Example (using a subset of words) [Jurafsky2001]

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.	x love xxxxxxxxxxxxxxxx sweet xxxxxx satirical xxxxxxxxxxxx xxxxxxxxxxxx great xxxxxxxx xxxxxxxxxxxxxxxxxxxxxx fun xxxx xxxxxxxxxxxxxxxxxxxxxx whimsical xxxx Romantic xxxx laughing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxx recommend xxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xx several xxxxxxxxxxxxxxxxx xxxxx happy xxxxxxxxx again xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxx		<table><tr><td>great</td><td>1</td></tr><tr><td>love</td><td>1</td></tr><tr><td>recommend</td><td>1</td></tr><tr><td>laugh</td><td>1</td></tr><tr><td>happy</td><td>1</td></tr><tr><td>...</td><td>...</td></tr></table>	great	1	love	1	recommend	1	laugh	1	happy	1	...	...
great	1														
love	1														
recommend	1														
laugh	1														
happy	1														
...	...														

- Orderless document representation: frequencies of words from a dictionary [Salton & McGill, 1983]
  - Disregards grammar, and even word order

## Visualization: tag clouds

- The larger the font, the more frequent the word
  - Many available software
- See e.g. <http://chir.ag/projects/preztags/>



## Naive Bayes for text classification

---

- **Classes  $\omega_i$  = text categories**
- **Extract *vocabulary* ( $V$  words) from training corpus ( $N$  texts)**
  - Texts previously labeled by an expert
- **Calculate  $P(\omega_i)$  terms**
  - For each class  $\omega_i$ 
    - ✓ Count texts with class  $\omega_i$  :  $N_i$
    - ✓  $P(\omega_i) = N_i / N$
- **Calculate  $p(x_k / \omega_i)$  terms**
  - Concatenate all docs with class  $\omega_i$  in a single text,  $t_i$  ( $n$  words).
  - For each word  $x_k$  in vocabulary
    - ✓  $n_k$  = number of occurrences of  $x_k$  in  $t_i$
    - ✓  $p(x_k / \omega_i) = (n_k + \alpha) / (n + \alpha V)$
- **Classification: use naïve Bayes rule.**

## Summary

---

- **Naive Bayes is not ...so naïve!**
  - Very fast, low complexity =  $O(V \log(V))$  where  $V$ =size(vocabulary)
  - Low storage requirements
  - Robust to irrelevant features
  - Optimal if the independence assumption holds
- **Other classifiers might give better accuracy, but at the price of higher complexity...**
  - Naïve Bayes is a good trade-off
  - It is a popular algorithm (text classification, natural language processing, ...)
  - Can be applied with continuous variables (see computer exercise 4)
- **Variants of BoW have been proposed for image classification**
  - Bags of *visual* words, e.g. (Csurka,2004)



---

## Probability Density Estimation

Parametric methods:  
Maximum Likelihood  
Bayesian Learning

## Summary of episodes 1-2

---

- It is easy to find decision regions (or discriminant functions) when the underlying probabilistic structure is known
  - General case: use Bayesian decision theory
  - Gaussian data → quadratic classifiers (linear, in some particular cases)
- The involved probabilities:
  - $P(\omega_i)$ , prior probability of class  $\omega_i$
  - $p(\mathbf{x}|\omega_i)$ , class-conditional probability density function
- In most cases, we do not know these probabilities!  
...What can we do ?

## Introduction

---

- Let us first suppose that we have in hands a learning data set (set of *training samples*) for each class,  $\mathcal{D}_i$ 
  - Both feature vectors,  $\mathbf{x}$ , and class labels,  $\omega_i$ , are known
  - This is called **supervised** learning (as opposed to non-supervised methods, or clustering, that we will study latter)
- There are two main approaches of density estimation
- Parametric methods
  - Suppose a particular, parametric, shape of the probability density function (e.g. Gaussian) → simplifies the problem
- Non-parametric methods
  - Do not assume anything about the form of the underlying density (see next chapter)

## Parametric, supervised learning (1/2)

---

- Learning prior probabilities presents no serious difficulties
  - We may use frequencies  $N_i/N$  (see exercise 1 in the present chapter)
- There are 2 approaches of the (class-conditional) PDF estimation problem
- The *estimative* (or plugin) approach:  $p(\mathbf{x}|\omega_i)$  depends on a parameter  $\theta_i$ 
  - The latter is unknown
  - Estimate  $\hat{\theta}_i$  (in the sense of ML for example)
  - Then, take  $p(\mathbf{x}|\omega_i) = p(\mathbf{x}|\hat{\theta}_i)$
- The result may depend on the training data set...

## Parametric, supervised learning (2/2)

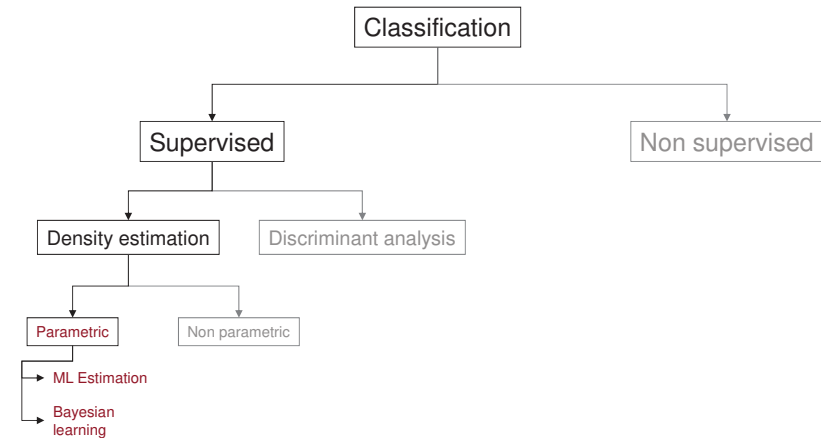
### ▪ The *predictive* approach or Bayesian learning

- The parameter vector,  $\theta_i$ , is viewed as a random variable
- The basic idea is to compute  $p(\mathbf{x}, \theta_i | \mathcal{D}_i)$  and to integrate it w.r.t.  $\theta_i$  (theorem of total probabilities)

$$p(\mathbf{x} | \omega_i) = p(\mathbf{x} | \mathcal{D}_i) = \int p(\mathbf{x}, \theta_i | \mathcal{D}_i) d\theta_i$$

- The parameters disappear thanks to integration (marginalization)
- This expression may be written as an expectation (mean) of  $p(\mathbf{x} | \theta_i)$ : contrary to the estimative approach, Bayesian estimation takes into account the variability of parameter estimation with respect to the set of samples.
- To calculate  $p(\mathbf{x}, \theta_i | \mathcal{D}_i)$ , we use Bayesian theory, hence the name of the approach

## A hierarchy of methods



## Maximum Likelihood (ML) estimation

### ▪ Attractive attributes

- Often simpler than alternative methods
- Good convergence properties as the number of training samples increases

### ▪ Using simplified notations: $p(\mathbf{x} | \omega_i) \rightarrow p(\mathbf{x})$

- The problem is to estimate the class-conditional PDF,  $p(\mathbf{x})$ , which is a function of a parameter vector  $\theta$ . More explicitly, we write it as  $p(\mathbf{x} | \theta)$ .
- Note that it should be written as  $p(\mathbf{x}; \theta)$  because  $\theta$  is not a random variable...
- We have in hands a collection of  $N$  samples:  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$   
These values are drawn independently according to the probability law  $p(\mathbf{x} | \theta)$ .  
Thus,

$$p(\mathcal{D} | \theta) = \prod_{k=1}^N p(\mathbf{x}_k | \theta)$$

## Maximum Likelihood (ML) estimation

### ▪ The estimate of $\theta$ in the sense of Maximum Likelihood is

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

- In other words, we take the value that best agrees with observed training samples

### ▪ Let us define the log-likelihood function

- Since **ln** is monotonically increasing, this does not change the estimate
- This changes products into sums, which may be helpful !

$$\hat{\theta} = \arg \max_{\theta} \left[ \sum_{k=1}^N \ln(p(\mathbf{x}_k | \theta)) \right]$$

- Then, we take  $p(\mathbf{x} | \omega_i) = p(\mathbf{x} | \hat{\theta})$



## Likelihood function, example (following [Duda])

### Training points in 1D (in red)

- Drawn from a Gaussian distribution with variance 1 and mean 3 (shown in dot-dashed line)

### $\theta = \text{mean}$

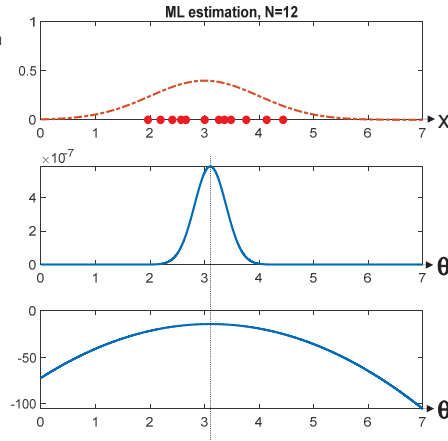
### Likelihood function

$$p(\mathcal{D}|\theta) = \prod_{k=1}^N p(x_k|\theta)$$

- Becomes narrower (and centered on 3) as  $N \nearrow$

### Log-likelihood function

$$\ln[p(\mathcal{D}|\theta)]$$



## ML for 1D Gaussian distributions

### Two parameters must be estimated

- The mean  $\theta_1 = \mu$  and variance  $\theta_2 = \sigma^2$ , since  $p(x|\theta) = \mathcal{N}(\mu, \sigma^2)$

$$\hat{\theta} = \arg \max_{\theta} \left[ \sum_{k=1}^N \ln \left[ \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(x_k - \mu)^2}{2\sigma^2} \right) \right] \right] = \arg \max_{\theta} \left[ -N \ln(\sqrt{2\pi\theta_2}) - \sum_{k=1}^N \frac{(x_k - \theta_1)^2}{2\theta_2} \right]$$

- Canceling the derivative of the log-likelihood w.r.t.  $\mu$  and  $\sigma^2$  (respectively)

$$\sum_{k=1}^N (x_k - \theta_1) = 0 \quad \text{and} \quad -\frac{N}{\theta_2} + \sum_{k=1}^N \frac{(x_k - \theta_1)^2}{\theta_2^2} = 0$$

- We obtain the usual sample mean and variance

$$\hat{\theta}_1 = \frac{1}{N} \sum_{k=1}^N x_k \quad \text{and} \quad \hat{\theta}_2 = \frac{1}{N} \sum_{k=1}^N (x_k - \hat{\theta}_1)^2$$

## ML for multivariate Gaussian distributions

### The log-likelihood is (where $d = \dim(\mathbf{x})$ )

$$\ln[p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma})] = -\frac{N}{2} \ln|\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{k=1}^N (\mathbf{x}_k - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_k - \boldsymbol{\mu}) + cst$$

### The maximum likelihood estimates of $\theta_1 = \boldsymbol{\mu}$ and $\theta_2 = \boldsymbol{\Sigma}$ are

$$\hat{\theta}_1 = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k \quad \text{and} \quad \hat{\theta}_2 = \hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{k=1}^N (\mathbf{x}_k - \hat{\theta}_1)(\mathbf{x}_k - \hat{\theta}_1)^T$$

### Specific case: $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}_d \rightarrow 2^{\text{nd}}$ parameter is scalar $\theta_2 = \sigma^2$

$$\hat{\theta}_1 = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k \quad \text{and} \quad \hat{\theta}_2 = \frac{1}{Nd} \sum_{k=1}^N \|\mathbf{x}_k - \hat{\theta}_1\|^2$$

## The question of bias

### The ML estimator of the mean is unbiased

### The ML estimator of the variance is biased

$$E[\hat{\theta}_1] = \frac{1}{N} \sum_{k=1}^N E[x_k] = \mu \quad \text{and} \quad E[\hat{\theta}_2] = \frac{1}{N} \sum_{k=1}^N E[(x_k - \hat{\theta}_1)^2] = \frac{N-1}{N} \sigma^2$$

- This is because the variance estimator uses the *estimated* mean and not its *true* value...
- However, when  $N$  becomes arbitrarily large, the bias cancels.

### In general, the ML estimator has attractive *asymptotic* properties:

- Asymptotically (i.e. when  $N \rightarrow \infty$ ), the ML estimator is consistent i.e. it converges in probability to the true value of parameters
- In many situations, the ML estimate is asymptotically normal, with variance decaying as  $1/N$ . e.g. the mean  $\hat{\theta}_1 \sim \mathcal{N}(\mu, \frac{\sigma^2}{N})$
- It is asymptotically efficient, i.e. it achieves the Cramer-Rao bound
- Note : these nice properties are only valid for large values of  $N$ ...

## Exercise 1 [Duda]

- ML estimation may be used to estimate prior probabilities...
- Suppose we have  $N$  samples drawn by successive independent samplings and let  $z_{ik}$  the binary variable that denotes the state of nature (*one-hot-encoding*)
  - $z_{ik} = 1$  if the state of nature was  $\omega_i$  for the  $k$ -th sample
  - $z_{ik} = 0$  otherwise



- The  $z_{ik}$ 's are independent Bernoulli variables, hence:

$$P(z_{11}, \dots, z_{CN} | P(\omega_i)) = \prod_{i=1}^C \prod_{k=1}^N P(z_{ik} | P(\omega_i)) = \prod_{i=1}^C \prod_{k=1}^N P(\omega_i)^{z_{ik}} (1 - P(\omega_i))^{(1-z_{ik})}$$

- Now, show that the ML estimate of  $\theta_i = P(\omega_i)$  is

$$\hat{P}(\omega_i) = \frac{1}{N} \sum_{k=1}^N z_{ik} = \frac{N_i}{N}$$

## Solution of exercise 1

- We have to maximize

$$P(z_{11}, \dots, z_{CN} | P(\omega_i)) = \prod_{i=1}^C \prod_{k=1}^N \theta_i^{z_{ik}} (1 - \theta_i)^{(1-z_{ik})}$$

- i.e. to maximize its logarithm

$$\ln P(z_{11}, \dots, z_{CN} | P(\omega_i)) = \sum_{i=1}^C \sum_{k=1}^N z_{ik} \ln(\theta_i) + (1 - z_{ik}) \ln(1 - \theta_i)$$

- Canceling the derivative with respect to  $\theta_i = P(\omega_i)$ , every term in the sum  $\sum_{i=1}^d$  vanishes except the  $i$ -th and we obtain

$$0 = \sum_{k=1}^N \frac{z_{ik}}{\hat{\theta}_i} - \frac{1 - z_{ik}}{1 - \hat{\theta}_i}$$

- So

$$\hat{\theta}_i = \hat{P}(\omega_i) = \frac{1}{N} \sum_{k=1}^N z_{ik} = \frac{N_i}{N}$$

## Maximum A Posteriori (MAP) Estimation

- MAP is a way to introduce *a priori* knowledge about  $\theta$ 
  - The parameter vector  $\theta$  is viewed as a random variable
  - Prior distribution  $p(\theta)$
- Seek the mode (peak) of the *posterior* distribution  $p(\theta | \mathcal{D})$

$$\tilde{\theta} = \arg \max_{\theta} [p(\mathcal{D} | \theta) p(\theta)] \propto p(\theta | \mathcal{D})$$

- Then take

$$p(x) = p(x | \tilde{\theta})$$

- ML can be seen as MAP with a uniform (flat) prior distribution

## 1D MAP estimation (Gaussian case)

- Estimation of the mean  $\mu$  of a Gaussian distribution
  - known variance,  $\sigma^2$
  - under Gaussian Prior assumption:  $p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$ .

- The MAP criterion is

$$J_{MAP}(\theta) = -\ln[p(\mathcal{D} | \theta) p(\theta)] = N \ln \left[ \sqrt{2\pi\sigma^2} \right] + \sum_{k=1}^N \frac{(x_k - \mu)^2}{2\sigma^2} + \ln \left[ \sqrt{2\pi\sigma_0^2} \right] + \frac{(\mu - \mu_0)^2}{2\sigma_0^2}$$

- Canceling its derivative, we obtain

$$0 = -\sum_{k=1}^N \frac{(x_k - \mu)}{\sigma^2} + \frac{(\mu - \mu_0)}{\sigma_0^2} \quad \rightarrow \quad \tilde{\mu} = \frac{\sigma^2}{N\sigma_0^2 + \sigma^2} \mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \hat{\mu}$$

$$\sigma_0^2 \sum_{k=1}^N (x_k - \mu) = \sigma^2 (\mu - \mu_0) \quad \left[ \tilde{\mu} = (1 - \alpha) \mu_0 + \alpha \hat{\mu} \right] \quad \alpha = \frac{1}{1 + \sigma^2 / N\sigma_0^2}$$

$$\mu(\sigma^2 + N\sigma_0^2) = \mu_0\sigma^2 + N\sigma_0^2 \hat{\mu} \quad \sigma_0^2 \rightarrow \infty : \text{non-informative prior (MAP} \Leftrightarrow \text{ML estimate)}$$

## Bayesian learning: general theory

### Formulation of the problem (simplified notation)

- Estimate the class-conditional PDF,  $p(x)$  using a training sample set,  $\mathcal{D}$
- The best we can do with the data in hand is to estimate  $p(x|\mathcal{D})$ .
- This can be done by integrating the joint law  $p(x, \theta|\mathcal{D})$  w.r.t the parameter vector,  $\theta$  (marginalization)

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

### Recall that, by definition,

$$p(x, \theta|\mathcal{D}) = p(x|\theta, \mathcal{D}) p(\theta|\mathcal{D})$$

- But, knowing  $\mathcal{D}$  does not bring supplementary information about the density of  $x$  if the parameter vector,  $\theta$ , is known:  $p(x|\theta, \mathcal{D}) = p(x|\theta)$ . So, finally:

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

## Remark

### Let us compare the obtained expression:

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

### with the ML estimator

$$p(x|\omega_i) = p(x|\hat{\theta})$$

### Bayesian learning averages $p(x|\theta)$ over possible values of $\theta$

- In contrast with the estimative approach, it takes into account the variability of parameter estimation with respect to the sample data set...
- If the posterior density,  $p(\theta|\mathcal{D})$ , peaks very sharply, it may be considered as a Dirac and the integral simplifies.

### Implementation

- Calculate  $p(\theta|\mathcal{D})$
- Integrate...

## Implementation

### In the expression $p(x|\mathcal{D}) = \int p(x|\theta) p(\theta|\mathcal{D}) d\theta$

- The analytic form of  $p(x|\theta)$  is known
- The *posterior* distribution,  $p(\theta|\mathcal{D})$  encodes our knowledge about  $\theta$ 
  - ✓ *a priori*, i.e. before observing the data (hence, rather imprecise)
  - ✓ Information provided by the training sample set.

- Using Bayes theorem (see MAP)  $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta) d\theta} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$

- And, since data are drawn independently (see ML)  $p(\mathcal{D}|\theta) = \prod_{k=1}^N p(x_k|\theta)$

### In general, the integral is difficult to calculate analytically. One can use numerical techniques (Monte-Carlo...) instead

- See Chap. 4 in [Webb]
- Exception: the Gaussian case

## Bayesian learning: the Gaussian case (1/4)

### A 1D example:

- $x$  sample from a normal law, with known variance:  $p(x|\theta) = \mathcal{N}(\mu, \sigma^2)$
- Unknown parameter:  $\theta = \mu$ , the mean, supposed to be normally distributed:  $p_0(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$ . The variance,  $\sigma_0^2$ , is taken very large to measure our uncertainty about the initial guess,  $\mu_0$ .

### Using Bayes rule and the above definitions, we form the *posterior* distribution:

$$p(\mu|\mathcal{D}) = \frac{p_0(\mu) \prod_{k=1}^N p(x_k|\mu)}{P(\mathcal{D})} = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left(-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right) \frac{1}{P(\mathcal{D})} \prod_{k=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_k - \mu)^2}{2\sigma^2}\right)$$

### It can be shown that

$$p(\mu|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma_N^2)$$

## Bayesian learning: the Gaussian case (2/4)

### ■ The parameters of the posterior $p(\mu|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma_N^2)$ are

- mean 
$$\mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \hat{\mu}_N + \frac{\sigma^2}{N\sigma_0^2 + \sigma^2} \mu_0$$
- variance 
$$\sigma_N^2 = \frac{\sigma^2 \sigma_0^2}{N\sigma_0^2 + \sigma^2}$$
- where  $\hat{\mu}_N = \frac{1}{N} \sum_{k=1}^N x_k$  is the sample mean, i.e. the ML estimate

### ■ So, for $p(\mu|\mathcal{D})$ :

- $N = 0$ : one finds the *a priori* distribution,  $\mathcal{N}(\mu_0, \sigma_0^2)$
- $N \rightarrow \infty$ : the distribution becomes sharply peaked around the ML solution
- In general, the mean  $\mu_N$  is a linear combination of the prior mean  $\mu_0$  and the sample mean  $\hat{\mu}_N$ :  $\mu_N = \alpha \hat{\mu}_N + (1 - \alpha) \mu_0$  with  $\alpha = 1/(1 + \sigma^2/N\sigma_0^2)$

## Bayesian learning: the Gaussian case (2/4)

### ■ The parameters of the posterior $p(\mu|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma_N^2)$ are

- mean 
$$\mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \hat{\mu}_N + \frac{\sigma^2}{N\sigma_0^2 + \sigma^2} \mu_0$$
- variance 
$$\sigma_N^2 = \frac{\sigma^2 \sigma_0^2}{N\sigma_0^2 + \sigma^2} = \frac{\sigma^2}{N + \sigma^2/\sigma_0^2}$$
- where  $\hat{\mu}_N = \frac{1}{N} \sum_{k=1}^N x_k$  is the sample mean, i.e. the ML estimate

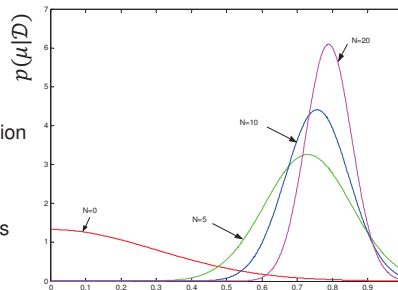
### ■ Indeed, the compromise also depends on variances

- Degenerate case  $\sigma_0^2 = 0 \rightarrow$  extremely strong prior, data have no effect  $\mu_N = \mu_0$
- Other extreme case  $\sigma_0^2 \gg \sigma^2 \rightarrow$  extremely loose prior, only data matter  $\mu_N = \hat{\mu}_N$
- In general, compromise according to the ratio  $\sigma^2/\sigma_0^2$ ,  $\rightarrow$  “dogmatism”

## Bayesian learning: the Gaussian case (3/4)

### ■ Toy example

- $p(x|\theta) = \mathcal{N}(0.8, 0.9)$ ,
- We take  $\mu_0$  far from the true solution and  $\sigma_0^2$  large:  $p_\theta(\mu) = \mathcal{N}(0, 0.9)$ .
- We plot  $p(\mu|\mathcal{D})$  for different values of the number of samples,  $N$



- One can see that, when  $N \rightarrow \infty$ , the estimate gets closer from the true value  $\mu = 0.8$  and that the distribution becomes sharply peaked (the uncertainty about the parameter value decreases)

- This suggest a recursive implementation which, indeed, is possible.

## Bayesian learning: the Gaussian case (4/4)

### ■ Back to the expression of $p(x)$ :

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

- In the Gaussian case, the integral can be computed analytically and one can show that

$$p(x|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma^2 + \sigma_N^2)$$

- i.e.  $\mu_N$  is treated as the true mean
- the known variance is increased to account for the additional uncertainty on  $x$  due to our lack of knowledge of  $\mu$ .
- when  $N \rightarrow \infty$ ,  $\sigma_N^2 \rightarrow 0$  and  $\mu_N \rightarrow \hat{\mu}_N$

## The Gaussian case (recap)

- **Likelihood: known variance, unknown mean**  $p(x|\mu) = \mathcal{N}(\mu, \sigma^2)$
- **Prior**  $p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$
- **Posterior: analytically**  $p(\mu|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma_N^2)$
- **Estimated distribution (analytically, also)**  $p(x|\mathcal{D}) = \mathcal{N}(\mu_N, \sigma^2 + \sigma_N^2)$   
 $\mu_N = \alpha \hat{\mu}_N + (1 - \alpha) \mu_0$   $\alpha = \frac{N}{N + \sigma^2 / \sigma_0^2}$
- **Recall that, for the MAP,**  $p(x) = \mathcal{N}(\mu_N, \sigma^2)$   
**and for the ML,**  $p(x) = \mathcal{N}(\hat{\mu}_N, \sigma^2)$

## Take-away remarks

- **Two families of approaches**
  - Estimative: the parameter vector is estimated and its value is then incorporated into the expression of the PDF (plug-in estimates)
    - ✓ E.g. ML, MAP estimators
    - ✓ Are simple to implement
    - ✓ But their result may highly depend on the set of training samples,  $\mathcal{D}$ .
  - Predictive: the PDF is directly determined by integration of the joint probability law with respect to the parameter (marginalization):
    - ✓ E.g. Bayesian learning
    - ✓ Accounts for the variability of sample sets
    - ✓ But is more complex (except in simple cases, e.g. Gaussian).
  - The estimates:
    - ✓ are different for small number of samples,
    - ✓ ...but the methods converge to the same solution for  $N \rightarrow \infty$
- **Frequencies are a ML estimator of probabilities**  $\hat{P}(\omega_i) = N_i / N$

## Supplementary material

## One-hot-encoding: example

- **Suppose we have  $N = 5$  samples in hand, with their labels**

Sample	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
Label	$\omega_1$	$\omega_1$	$\omega_3$	$\omega_2$	$\omega_1$

- **The one-hot-encoding of this sample set  $\mathcal{D}$  is**

Sample	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
$\omega_1$	1	1	0	0	1
$\omega_2$	0	0	0	1	0
$\omega_3$	0	0	1	0	0

## One-hot-encoding (a.k.a one-of-K encoding)

- One vector  $\mathbf{z}_k$ , of length  $C$  for each of the  $N$  sample,  $\mathbf{x}_k$ 
  - $z_{ik} = 1$  if the state of nature was  $\omega_i$  for the  $k$ -th sample,  $\mathbf{x}_k$
  - $z_{ik} = 0$  otherwise

Sample / Class	$\mathbf{x}_1$	...	$\mathbf{x}_k$	...	$\mathbf{x}_N$	Sum
$\omega_1$	$z_{11}$	...	$z_{1k}$	...	$z_{1N}$	$N_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$\omega_i$	$z_{i1}$	...	$z_{ik}$	...	$z_{iN}$	$\sum_k z_{ik} = N_i$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$\omega_C$	$z_{C1}$	...	$z_{Ck}$	...	$z_{CN}$	$N_C$
Sum	1	...	$\sum_i z_{ik} = 1$	...	1	$N$



## Bernoulli variable

- If  $X$  is a binary random variable with Bernoulli Distribution

$$P(X = 1) = P$$

$$P(X = 0) = 1 - P(X = 1) = 1 - P = Q$$

where  $P$  is the expected value of  $X$

$$E[X] = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) = P$$

- We may summarize

$$P(X = 1|P) = P$$

$$P(X = 0|P) = 1 - P$$

as

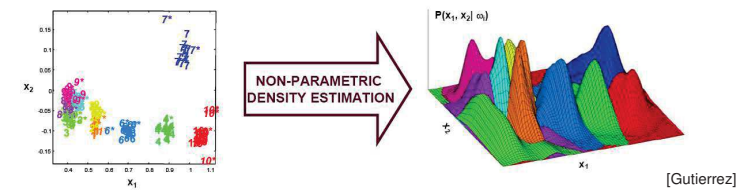
$$P(X = x|P) = P^x(1 - P)^{1-x}$$

## Probability Density Estimation

NON parametric methods  
Histograms, kernel methods (Parzen)  
Nearest-neighbors methods

## Summary of episodes 1-3

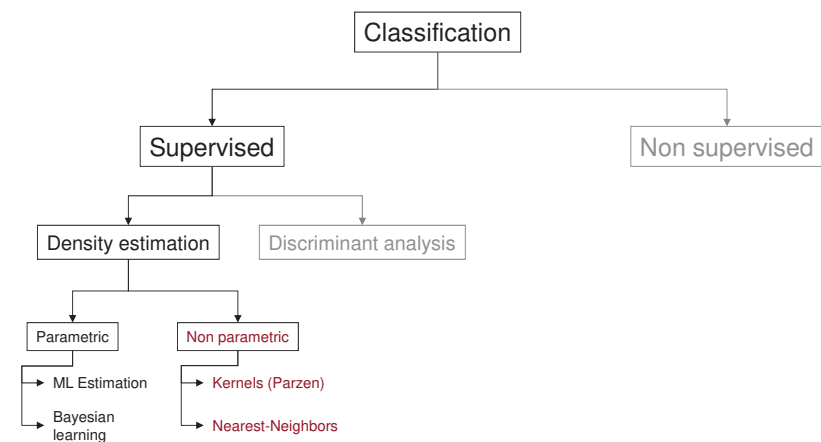
- So far, we supposed that we knew
  - Class-conditional PDFs  $p(\mathbf{x}|\omega_i)$  and prior probabilities  $P(\omega_i)$
  - At least, the parametric form of the likelihood functions
- Most of the time, the situation is more complicated...
- What can we do?
  - We shall estimate the density directly from the data samples, without making any parametric assumption about the underlying pdf...



## Introduction

- Once again, we will find 2 families of methods
- Kernel Density Estimation methods
  - May be seen as extensions of PDF approximation using histograms
  - Decision made in the usual Bayesian framework, once  $p(\mathbf{x}|\omega_i)$  is estimated
- Nearest-neighbors methods
  - Must be used with caution
  - Computing posterior probabilities  $P(\omega_i|\mathbf{x})$  yields the well-known k-NN classifier
- Note
  - We are still in the context of supervised learning
  - So, we use simplified notations:  $p(\mathbf{x})$  for  $p(\mathbf{x}|\omega_i)$  when no ambiguity...

## A hierarchy of methods



## Histograms

### Probably the simplest and most familiar PDF estimation method:

- The feature space is divided into bins
- The density is approximated by the fraction of the  $N$  training samples that fall into the considered bin / volume of the bin

$$P_H(\mathbf{x}) = \frac{1}{N} \frac{[\text{number of samples in the same bin as } \mathbf{x}]}{[\text{volume of the bin } \mathbf{x} \text{ belongs to}]}$$

- 2 « parameters »: bin dimensions and location of first bin

### This representation is simple and easy to implement

### Density shape depends on the position/orientation of bins

- The estimates show discontinuities that may cause misinterpretations
- They are not due to the underlying density

## Non-parametric density estimation, general formulation

### Let us suppose that a set of $N$ sample vectors is drawn according to the same distribution: $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

- The probability that  $k$  of these vectors fall in a region  $\mathcal{R}$  of the feature space is given by a binomial distribution:

$$P(k) = \frac{N!}{k!(N-k)!} P^k (1-P)^{N-k}$$

where  $P$ =probability that one vector falls into  $\mathcal{R}$ , and it may be shown that:

$$E\left[\frac{k}{N}\right] = P \quad \text{and} \quad E\left[\left(\frac{k}{N} - P\right)^2\right] = \frac{P(1-P)}{N}$$



### In other words, the frequency, $k/N$ , is an estimator of $P$

- Unbiased
- Asymptotically consistent: when  $N \rightarrow \infty$ , its distribution becomes sharply peaked around the value of  $P$  (the variance tends to 0)

## Non-parametric density estimation, general formulation

### We will then write:

$$P \equiv \frac{k}{N}$$

### By definition of probability density,

- the probability that a sample  $\mathbf{x}$  drawn according to  $p(\mathbf{x})$  falls in  $\mathcal{R}$  is:

$$P = \int_{\mathcal{R}} p(\mathbf{x}') d\mathbf{x}'$$

### If we suppose that the volume $V$ of region $\mathcal{R}$ is small enough

- $p(\mathbf{x})$  is almost constant and thus:

$$P \cong p(\mathbf{x})V$$

### Gathering these results:

$$p(\mathbf{x}) \equiv \frac{k}{N.V}$$

## Non-parametric density estimation: the trade-off...

### Issues

- If  $V$  is fixed and  $N \rightarrow \infty$ , we have that  $(k/N) \rightarrow P$  but then,

$$\frac{k}{N.V} \rightarrow \frac{P}{V} = \frac{\int_{\mathcal{R}} p(\mathbf{x}') d\mathbf{x}'}{\int_{\mathcal{R}} d\mathbf{x}'}$$

- So, what we obtain is an averaged version of the sought density...
- We must have  $V \rightarrow 0$  to obtain  $p(\mathbf{x})$ , but if we fix  $N$ , the risk is either:
  - ✓ To have no sample into  $\mathcal{R}$ :  $p(\mathbf{x})=0$ ...
  - ✓ To have one or more samples exactly in  $\mathbf{x}$ : divergence of the estimate

### In theory, we simultaneously need, when $N \rightarrow \infty$

- $V \rightarrow 0$ : avoid averaging,
- $k \rightarrow \infty$ : (if  $p(\mathbf{x}) \neq 0$ ) assure that  $k/N \rightarrow P$ ,
- $k/N \rightarrow 0$ : avoid divergence of  $(k/N)/V$



## Take-away remarks

### General expression of non-parametric density estimator:

$$p(x) \equiv \frac{k}{N.V} \text{ where } \begin{cases} V: \text{volume around } x \\ N: \text{total number of examples} \\ k: \text{number examples in } V \end{cases}$$

### In practice, 2 approaches apply this result:

- Choose a value for  $V$  and evaluate  $k$  from the data: kernel-based density estimation
- Choose a value for  $k$  and evaluate the corresponding volume from the data: k-nearest neighbors methods

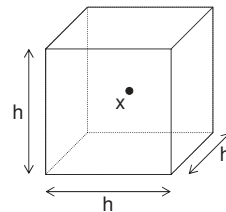
### The estimator converges to the sought probability density when $N \rightarrow \infty$ , if $V \rightarrow 0$ and $k/N \rightarrow 0$ simultaneously...

## Kernel Density Estimation (KDE)

## Kernel methods: Parzen windows (1962)

### Assume that the region $\mathcal{R}$ is a hypercube

- Centered on  $x$
- Of edge length  $h$



### Its volume is $V = h^D$ ,

- where  $D$  denotes the number of dimensions of the feature space

### To obtain the analytic expression of $k$ , the number of samples falling in region $\mathcal{R}$ , let us define a *window function*, $K(u)$

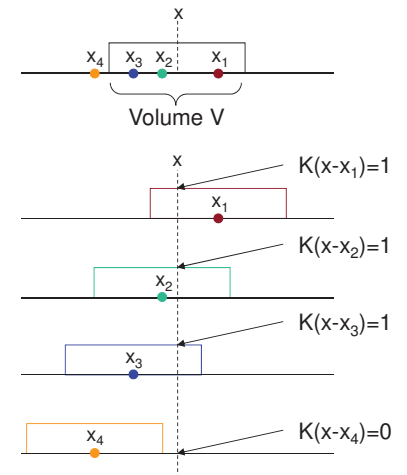
$$K(u) = \begin{cases} 1 & |u_j| \leq 1/2 \\ 0 & \text{otherwise} \end{cases} \quad \forall j = 1, \dots, D$$

- We have  $K((x - x_i)/h) = 1$  if  $x_i$  falls within the hypercube centered on  $x$ , and 0 otherwise
- The unit hypercube is called Parzen window

## Parzen windows

### The total number of points falling within the hypercube is

$$k = \sum_{i=1}^N K\left(\frac{x - x_i}{h}\right)$$



### Substituting $k$ into the general expression of density estimate:

$$p_{\text{Kernel}}(x) = \frac{1}{Nh^D} \sum_{i=1}^N K\left(\frac{x - x_i}{h}\right)$$

- Parzen density estimate resembles the histogram, but the difference is that the location of bins depends on the data!

Following [Gutierrez]

## Properties of the estimator

- The expectation of the density estimator is

$$E[p_{\text{kernel}}(x)] = \frac{1}{Nh^D} \sum_{i=1}^N E\left[K\left(\frac{x-x_i}{h}\right)\right] = \frac{1}{h^D} E\left[K\left(\frac{x-x_i}{h}\right)\right] = \frac{1}{h^D} \int K\left(\frac{x-x'}{h}\right) p(x') dx'$$

(supposing the  $x_i$ 's drawn from the true law,  $p(x)$ )

- The expectation of the estimator is the convolution of the true density with the kernel
  - The width parameter,  $h$ , is a smoothing parameter: the higher it is, the smoother the estimate  $p_{\text{kernel}}$  is.
  - If  $h \rightarrow 0$ , the kernel tends to the Dirac, and  $p_{\text{kernel}}$  tends to the true density
    - ✓ In practice,  $h$  must not be chosen too small, otherwise the estimate might degenerate into a set of pulses centered on the learning samples...

## Exercise 1: numerical application (from [Gutierrez])



- Consider the following learning sample (in 1-D)

$$\mathcal{D} = \{x_1, x_2, \dots, x_N\} = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$$

- Plot the training data to get an idea of their repartition

- Estimate  $p(x)$  using Parzen windows, at points  $x = 3, 10, 15$   
Use a window width  $h = 4$ .

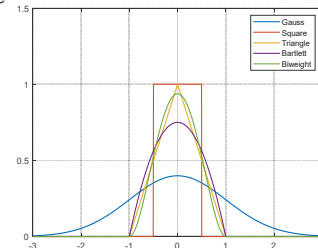
## Continuous Kernels

- Parzen windows yield discontinuous estimates
- Solution: use a smooth, positive kernel, such that  $\int K(x') dx' = 1$

- ✓ Square kernel  $K(u) = 1$  if  $|u| \leq 1/2$ , 0 otherwise
- ✓ Triangle  $K(u) = 1 - |u|$  if  $|u| \leq 1$ , 0 otherwise
- ✓ Biweight  $K(u) = \frac{15}{16}(1 - u^2)^2$  if  $|u| \leq 1$ , 0 otherwise
- ✓ Bartlett  $K(u) = \frac{3}{4}(1 - u^2)$  if  $|u| \leq 1$ , 0 otherwise
- ✓ Gaussian  $K(u) = \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)$

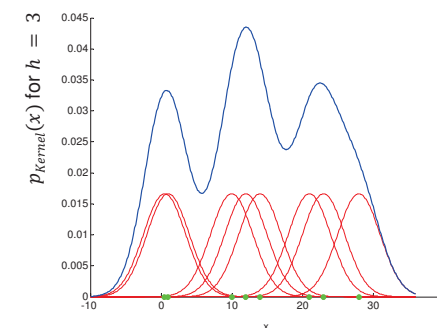
- The expression of the density remains the same

$$p_{\text{kernel}}(x) = \frac{1}{Nh^D} \sum_{i=1}^N K\left(\frac{x-x_i}{h}\right)$$



## Bandwidth selection

- The probability density estimate is a sum of functions placed at each data point
  - The parameter  $h$  determines the width of these functions
  - It is called smoothing parameter or bandwidth parameter

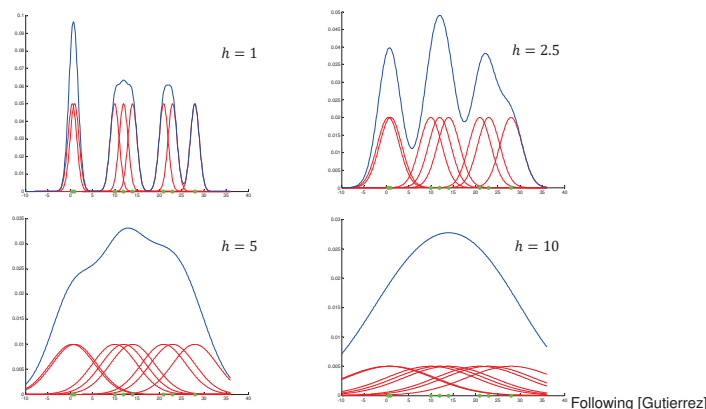


Following [Gutierrez]

## Bandwidth selection

### The choice of $h$ is crucial

- Too high: over-smoothing masks the structure of data
- Too small: irregular estimate, difficult to interpret



## Bandwidth selection methods

### $h$ cannot be estimated directly from the data

- Maximizing the likelihood  $p(x_1, \dots, x_N | h)$  leads to  $h = 0$   
 $\Rightarrow$  Estimated density = 1 Dirac on each sample and 0 elsewhere...

### Subjective choice

- Draw several curves with different smoothing parameters and choose the “best” one... Not satisfying, nor practical in multiple dimensions!

### Reference to a known distribution: MISE

- e.g. Gaussian:  $h = 1.06 \sigma N^{-1/5}$  (rule of thumb)

### Cross-validation

### Adaptive bandwidth (two-step estimation)

### Distance between closest points

### And many others... (see e.g. [Webb])

## The multi-dimensional case

### The $D$ -variate Kernel Density Estimator with kernel $K(\mathbf{x})$ and $(D \times D)$ bandwidth matrix $\mathbf{H}$ is

$$p_{\text{Kernel}}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i)$$

where  $K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2} \mathbf{x})$ ,  $\mathbf{H}$  is symmetric and positive definite.

### The multivariate kernel $K(\mathbf{x})$ can be generated from a symmetric univariate kernel $K_1(x)$ in two ways

- Product kernel (next slide):  $K(\mathbf{x}) = \prod_{i=1}^D K_1(x_i)$
- Radially symmetric kernel:  $K(\mathbf{x}) = K_1(\|\mathbf{x}\|) / \int_{\mathbb{R}^D} K_1(\|\mathbf{x}\|) d\mathbf{x}$

e.g. Gaussian kernel

$$p_{\text{Kernel}}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \frac{1}{(2\pi h^2)^{1/2}} \exp - \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2h^2}$$

## Product kernels

### Kernel considered as a product of one-dimensional kernels

$$p_{\text{Kernel}}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}, \mathbf{x}_i, h_1, \dots, h_D)$$

$$K(\mathbf{x}, \mathbf{x}_i, h_1, \dots, h_D) = \frac{1}{h_1 \dots h_D} \prod_{d=1}^D K_d\left(\frac{\mathbf{x}_d - (\mathbf{x}_i)_d}{h_d}\right)$$

where the subscript  $d$  denotes the dimension index.

### Most of the time, kernels have the same shape along all dimensions, but their bandwidths may be different

$$p_{\text{Kernel}}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \frac{1}{h_1 \dots h_D} \prod_{d=1}^D K_d\left(\frac{\mathbf{x}_d - (\mathbf{x}_i)_d}{h_d}\right)$$

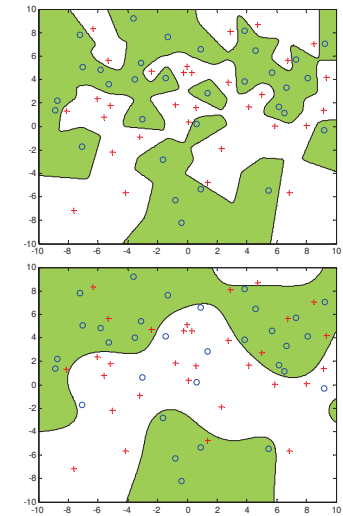
## Classification method

- Estimate the PDF for every class:  $p(\mathbf{x}|\omega_i) = p_{\text{kernel}}(\mathbf{x}|\omega_i)$
- Estimate the *a priori* probability for every class:  $P(\omega_i)$
- Use a Bayesian classifier (e.g. MAP)
  - Form  $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) \propto p_{\text{kernel}}(\mathbf{x}|\omega_i) \cdot P(\omega_i)$
  - Choose  $\omega_i$  if  $g_i(\mathbf{x}) > g_j(\mathbf{x})$  for all  $j \neq i$
- Attractive features
  - No assumption about the shape of distributions, that might be arbitrarily complicated.
  - The training error can be made very small if  $h$  is small (which does NOT imply that the classification error will be small on a novel test set !!!)
- The shape of decision boundaries clearly depends on the choice of the bandwidth

## An example with 2 classes

(following [DUDA], p. 171)

- Class-conditional PDFs:
  - Estimated Using Parzen windows with isotropic Gaussian kernels
    - ✓ Top, with  $h = 0,5$
    - ✓ Bottom, with  $h = 2$
  - No single bandwidth seems ideal overall
- Prior Probabilities:
  - Frequencies  $N_i/N$
- MAP Classification
- Visualization
  - Decision Regions
    - ✓ In green: class 1 (blue circles)
    - ✓ In white: class 2 (red crosses)



## (Naive) multi-dimensional classification

- As the number of dimensions increases, more samples are needed to correctly estimate PDFs...
- Alternative: *naïve Bayes* classification
  - Consider class-conditionally independent features (where  $d$ =dimension index)

$$p(\mathbf{x}|\omega_i) = \prod_{d=1}^D p(x_d|\omega_i)$$

- The MAP discriminant function is

$$g_i(\mathbf{x}) = P(\omega_i) \prod_{d=1}^D p(x_d|\omega_i)$$

- Requires 1D only densities  $\rightarrow$  simple, but may provide good results

## *k* nearest-neighbors (kNN) methods

## Basis idea

- Recall the expression of non-parametric density estimator

$$p(x) \equiv \frac{k}{N \cdot V} \text{ where } \begin{cases} V: \text{volume around } x \\ N: \text{total number of examples} \\ k: \text{number examples in } V \end{cases}$$

- k nearest neighbors (kNN) estimate**

- Starting from  $x$ , seek its  $k$  nearest neighbors
- $V$  is the volume of the hyper-sphere enclosing the  $k$ NN, and

$$p(x) = \frac{k}{N \cdot C_D \cdot R_k^D(x)}$$

Volume of unit sphere in  $D$  dimensions

Distance from  $x$  to its  $k$ -th nearest neighbor  $x_{kNN}$

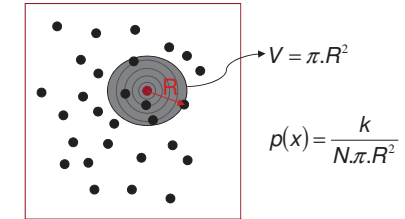
$$C_1 = 2, C_2 = \pi, C_3 = \frac{4\pi}{3}, \dots, C_D = \frac{\pi^{D/2}}{\Gamma(1 + D/2)}$$

## Remarks

- E.g in 2D:

following [Gutierrez]

( $k=5$ )



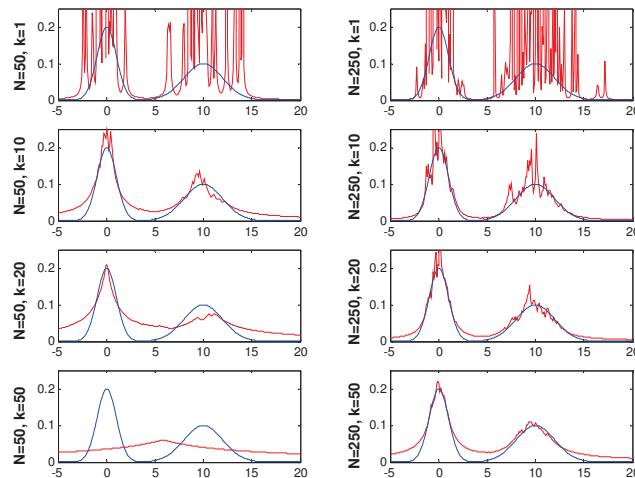
- kNN**

- Often provide unsatisfactory PDF estimates when  $N$  is not large enough (see following slides): noisy, non integrable (hence, no true PDFs), discontinuous...
- ...unless  $k \rightarrow \infty$  and  $k/N \rightarrow 0$  when  $N \rightarrow \infty$ : unbiased, consistent estimator

- However, they lead to a very simple approximation of the MAP classifier, as we will see in a moment...

## An example in 1D

$$p(x) = \frac{k}{2N|x - x_{kNN}|}$$



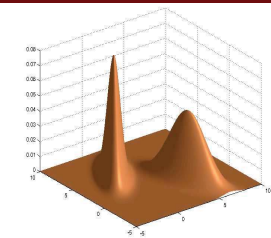
Following [Gutierrez]

## An example in 2D

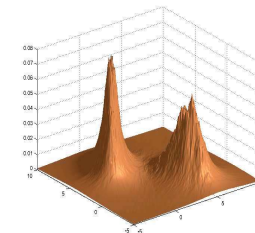
- $P(x) = 0.5\mathcal{N}(\mu_1, \Sigma_1) + 0.5\mathcal{N}(\mu_2, \Sigma_2)$

$$\mu_1 = \begin{pmatrix} 0 \\ 5 \end{pmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

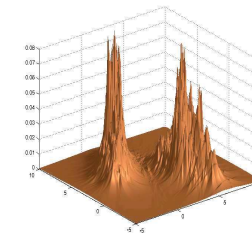
$$\mu_2 = \begin{pmatrix} 5 \\ 0 \end{pmatrix} \quad \Sigma_2 = \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix}$$



Theoretical



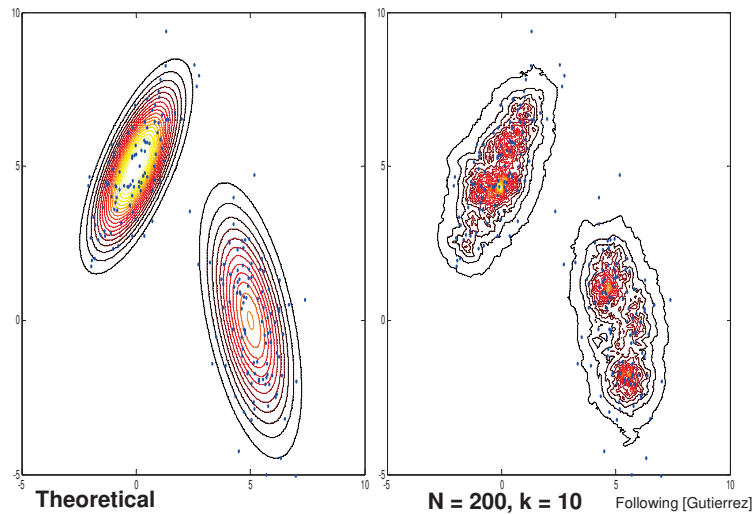
$N = 1000, k = 100$



$N = 200, k = 10$

Following [Gutierrez]

## An example in 2D: iso-probability level sets



## Bayesian classification using kNN

### ■ Suppose we have $N$ samples, $N_i$ of which from class $\omega_i$

- Consider a hyper-sphere with volume  $V$  around  $x$
- It encloses  $k$  samples,  $k_i$  of which from class  $\omega_i$
- The class-conditional likelihood is approximated by

$$p(x|\omega_i) = \frac{k_i}{N_i \cdot V}$$

- Similarly, the (overall) PDF of  $x$  is approximated by  $p(x) = \frac{k}{N \cdot V}$

- And the priors, by  $P(\omega_i) = \frac{N_i}{N}$

### ■ The (approximate) Bayesian classifier maximizes:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} = \frac{\frac{k_i}{N_i \cdot V} \cdot \frac{N_i}{N}}{\frac{k}{N \cdot V}}$$

$$P(\omega_i|x) = \frac{k_i}{k}$$

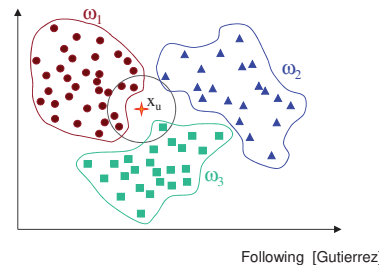
## The kNN classification rule (in French, kPPV)

- [Fix & Hodges, 1951], [Cover & Hart, 1967]
- Let  $x_u$  be an unlabeled sample, that must be classified
  1. Find the  $k$  nearest neighbors of  $x_u$  in the training data set
  2. Assign  $x_u$  to the class that appears most frequently within the kNN
- kNN require: an integer,  $k$ , a training set (with labels), a proximity measure (metric)

### ■ An example with 3 classes

- Use Euclidean distance
- Consider  $k = 5$  neighbors
- Among the 5 NN of  $x_u$ , 4 belong to  $\omega_1$  and 1, to  $\omega_3$

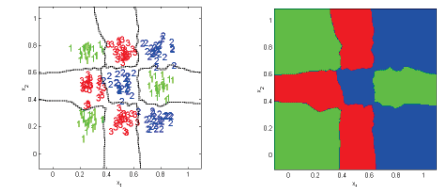
→ Assign  $x_u$  to class  $\omega_1$



## Examples

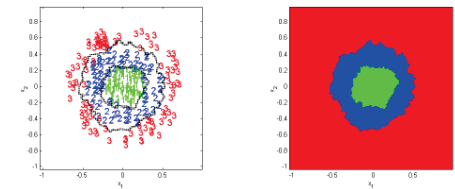
### ■ 2D problem, 3 classes

- Non linearly separable
- Multimodal PDFs
- $k = 5$
- Euclidean metric



### ■ 2D, 3 classes

- Non linearly separable
- Concentric PDFs
- $k = 5$
- Euclidean metric

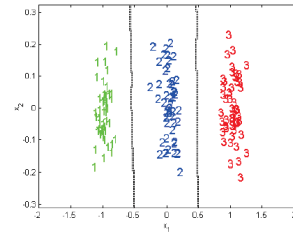


From [Gutierrez]

## kNN and the choice of metric

- The basis algorithm uses **Euclidean distance: it is sensitive to noise...**

- E.g. 3 classes in 2D
- 1st axis: useful information
- 2nd axis: white noise

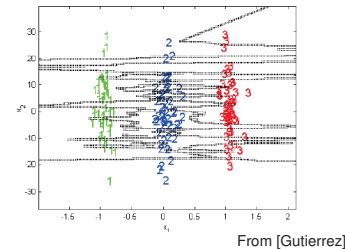


- 1st example: correct scale

- 2nd example: multiply by 100 the second axis scale

- the kNN are biased due to the high values of the 2nd, noisy coordinates

⇒ Bad results



From [Gutierrez]

## The choice of metric

- Noise is the Achilles' heel of kNN

- A possible solution: normalize axes
- ...but this may change decision boundaries
- Normalization is not enough if pertinent information is bound to a small number of features in high dimensional problems...

$$d(x, y) = \sqrt{\sum_{d=1}^D (x_d - y_d)^2}$$

- A better solution: use a weighted distance

$$d_w(x, y) = \sqrt{\sum_{d=1}^D w_d (x_d - y_d)^2}$$

- It is a way of selecting certain features (see next chapter)
- Several ways of setting weights have been proposed...

- Other metrics exist.

## kNN as a lazy learner

- Lazy learning algorithms

- Simply store training data and do nothing before having received any unlabelled, sample
- Answer a request using stored data and forget both the answer and intermediate results

- ...as opposed to eager learning algorithms

- Compile training data into a compact description or model
  - ✓ E.g. a PDF or its parameters (statistical approach)
- Discard training data after model compilation
- Classify unlabelled samples using the model, which stays in memory for future requests

- kNN

- Less computations for learning but more memory required, and more computations needed for answering requests

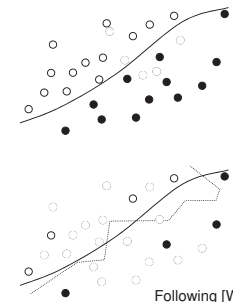
## Decreasing memory requirements

- The basis algorithm uses all N training samples

- Many points may be redundant
- The most important information (for classification) lies around decision boundaries

- 2 strategies for reducing the number of points

- « Editing »: classify all learning examples and discard misclassified ones, to better separate regions
- « Condensing »: discard correctly classified points without changing decision boundaries



Following [Webb]

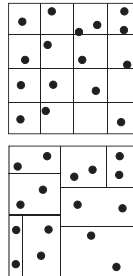
## Improving the efficiency of search procedure

### ■ Problem:

- Find nearest neighbors of  $x_q$  among  $N$  points, in  $D$ -dimensional space
- Naïve approach: compute all  $N$  distances and seek the  $k$  closest ones  
⇒ Impracticable as soon as  $N$  (or  $D$ ) becomes too large.

### ■ Improve neighbor search procedure.

- Divide search space into identical blocks: « *bucketing* », then first perform search on blocks, then on points.
- Use a  $D$ -dimensional generalization of binary search trees: «  $k$ -D trees ». Similar principle, but the partition depends on the density of points.



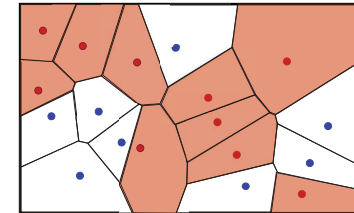
Following [Gutierrez]

## The nearest-neighbor rule

[Cover and Hart 1967]

### ■ Particular case where $k = 1$

- Choose the nearest sample to the point that must be classified
- Amounts to partition feature space in Voronoï cells centered on each sample
- E.g. (2 classes):



### ■ This approach is clearly sub-optimal:

- Its classification error is always higher than the error of an ideal Bayesian classifier
- However, for an infinite number of samples, it can be shown that it is always less than twice the classification error of the ideal Bayesian classifier.

## Summary

### ■ The expression of non-parametric density estimator.

- A trick to remember it:
  - ✓ Probabilities may be approximated by frequencies,
  - ✓ Probability density is a measure of probability per unit volume !

$$p(x) \equiv \frac{k}{N.V} \text{ where } \begin{cases} V: \text{volume around } x \\ N: \text{total number of examples} \\ k: \text{number examples in } V \end{cases}$$

### ■ Two well known implementations for classification

- Fix  $V$  and determine  $k$  from Parzen or smooth kernels, then use Bayesian classification, e.g. MAP.
- Fix  $k$  and determine  $V$ , which leads to a simple classification rule: the  $k$ -nearest neighbor algorithm.

## Supplementary material



## Moments of the binomial law

- The probability that  $k$  learning samples, among  $N$ , fall into a region  $\mathcal{R}$  of the feature space is:

$$P(k) = \frac{N!}{k!(N-k)!} P^k (1-P)^{N-k}$$

- Let us calculate  $E[k]$ :  $E[k] = \sum_{k=0}^N k P(k) = \sum_{k=0}^N k \frac{N!}{k!(N-k)!} P^k Q^{N-k}$

- The term for  $k=0$  is null, so we may change the beginning index of the sum

$$E[k] = \sum_{k=1}^N k \frac{N!}{k!(N-k)!} P^k Q^{N-k} = \sum_{k=1}^N \frac{N!}{(k-1)!(N-k)!} P^k Q^{N-k}$$

- Now, let us change variable:  $k'=k-1$

$$E[k] = \sum_{k'=0}^{N-1} \frac{N!}{k'!(N-k'-1)!} P^{k'+1} Q^{N-k'-1} = NP \sum_{k'=0}^{N-1} \frac{(N-1)!}{k'!(N-1-k')!} P^{k'} Q^{N-1-k'}$$



## Moments of the binomial law (2)

- It remains to apply Newton's binomial formula

$$(A+B)^N = \sum_{k=0}^N \frac{N!}{k!(N-k)!} A^k B^{N-k}$$

- Which gives the result

$$E[k] = NP(P+Q)^{N-1} = NP$$

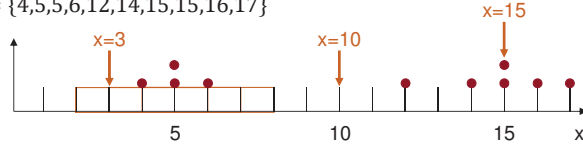
- NB : another solution is to use the moment-generating function

- Let us calculate  $\text{Var}(k) = E[k^2] - E[k]^2 = E[k^2] - N^2 P^2$ .

- The second term is straightforward, but calculating the first one is difficult!
- It is easier to calculate  $E[k(k-1)]$  because it is possible to use the above technique, "dropping" the first two terms of the sum. One then obtains:  $E[k(k-1)] = N(N-1)P^2$
- But, by linearity,  $E[k(k-1)] = E[k^2] - E[k]$ , so  $E[k^2] = N(N-1)P^2 + NP$
- So, finally,  $\text{Var}(k) = E[(k-NP)^2] = NP(1-P)$

## Solution of exercise 1

$$\mathcal{D} = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$$



- Recall that

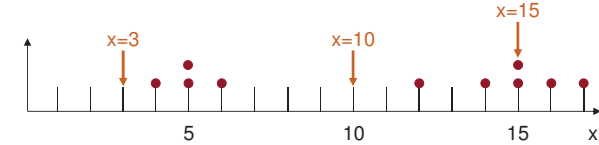
$$K(u) = \begin{cases} 1 & |u| \leq 1/2 \\ 0 & \text{otherwise} \end{cases}, \quad u = \frac{x - x_i}{4} \quad K(x - x_i) = \begin{cases} 1 & |x - x_i| \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

- Let us compute  $p(x=3)$

$$\begin{aligned} p_{\text{Kernel}}(x=3) &= \frac{1}{Nh^D} \sum_{i=1}^N K\left(\frac{x-x_i}{h}\right) = \frac{1}{10 \cdot 4^1} \left[ K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + K\left(\frac{3-5}{4}\right) + \dots + K\left(\frac{3-17}{4}\right) \right] \\ &= \frac{1}{10 \cdot 4^1} [1 + 1 + 1 + 0 + 0 + 0 + 0 + 0 + 0] \\ &= \frac{3}{40} = 0.075 \end{aligned}$$



## Solution of exercise 1



- Similarly

- 1 point ( $x=10$ ) lies at a distance lower than 2 from  $x=10$

$$\Rightarrow p_{\text{Kernel}}(x=10) = \frac{1}{40} = 0.025$$

- 5 points are less than 2 away from  $x=15$

$$\Rightarrow p_{\text{Kernel}}(x=15) = \frac{5}{40} = 0.125$$

- Distribution obtained

