Université de Montréal
Département d'informatique et de recherche opérationnelle

IFT6390
Fondements de l'apprentissage machine

Formalizing learning problems
Histogram methods
Curse of Dimensionality

Professor: Ioannis Mitliagkas
Slides adapted from: Pascal Vincent

Steps of a machine learning project

- Can the problem at hand be expressed as one of the classical machine learning task?
  (classification, regression, density estimation, clustering, dimensionality reduction, ...)
  What is the cost/loss that we actually want to optimize for this problem?

- Look at the data available for training/testing.
  Are there enough examples? How is the data formatted?
  What is the semantic of the data/features?

- Design and implement the data preprocessing steps.
  (to transform the data into the appropriate form for the learning algorithms we plan to use.)

- Train/test and evaluate the learning algorithms using the correct methodology (notion of "out of sample" error).
  (ex: train/test split, cross validation, etc.).

Practical steps of a data-mining project

- Clearly identify the problem to solve
- Formalize the problem as a machine learning task given the available data.
- Extract and store the data in the appropriate format (in a file, a table, etc.).
- Pre-process the data to get the adequate form for the learning algorithms.

"data plumbing"

- Modeling: experiment with different learning algorithms on the data.
- Evaluate the performance of each algorithm, to select the best one for the problem at hand.

model selection

- Deploy the system in production for the client.

Today's lecture

- Terminology reminder
- Formalization of the learning problem.
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

Today's lecture

- Terminology reminder
- Formalization of the learning problem.
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

D

Training Set

Input dimension: d

entrées:

cibles:

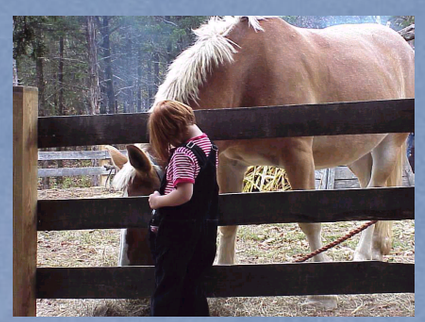
"horse"

"cat"

Training set size / number of examples:

n

etc...



"horse"

preprocessing, feature extraction

inputs: X outputs: Y

(feature vectors)

X_1	(3.5, -2, ... , 127, 0, ...)	+1	Y_1
	(-9.2, 32, ... , 24, 1, ...)	-1	
	etc...		
X_n	(6.8, 54, ... , 17, -3, ...)	+1	Y_n

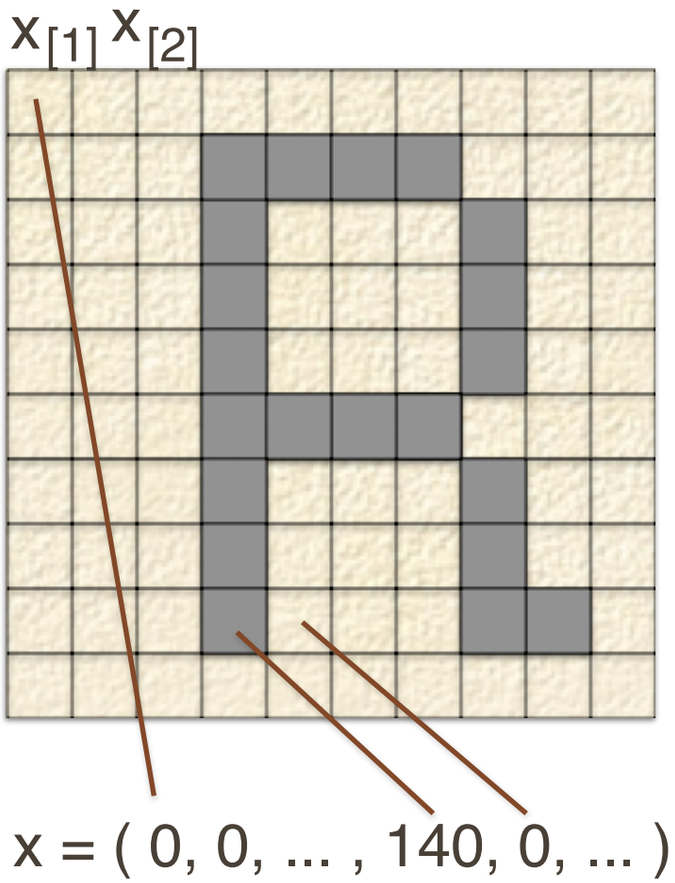
$X_{n,2}$

testing point:

→ ?

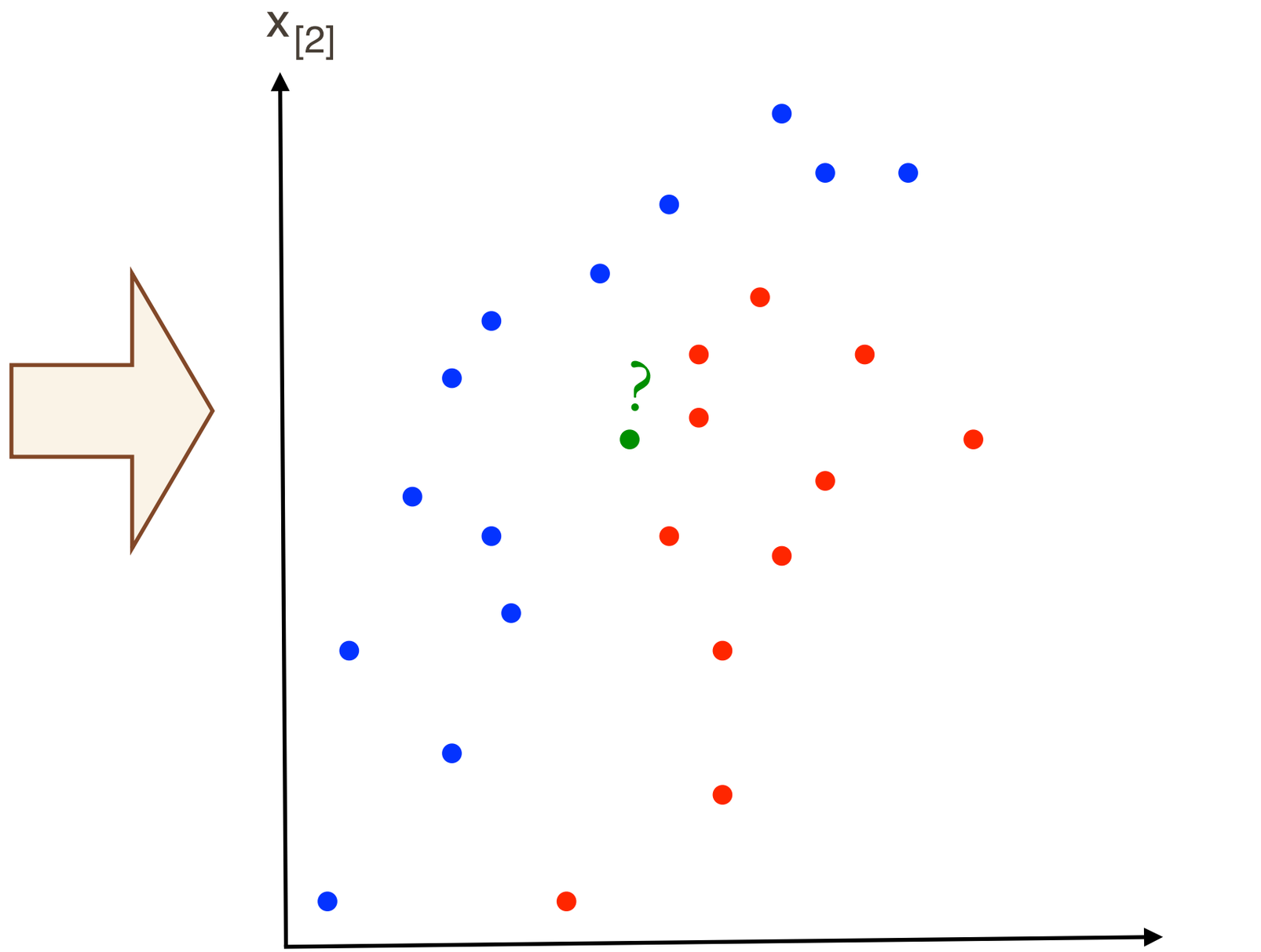
x = (5.7, -27, ... , 64, 0, ...) → ?

Data Representation



x = (0, 0, ... , 140, 0, ...)

x : point in $\mathbb{R}^d$



A dataset is seen as a set of vectors in a (high-dimensional) vector space.

Terminology of Supervised Learning

Input is usually a vector of dimension d.

$x \in \mathbb{R}^d$

input dimension

Training Set

1	entrée, observation, *input*, x_1	cible, *target*, sortie désirée, y_1
2	entrée, observation, *input*, x_2	cible, *target*, sortie désirée, y_2
3	entrée, observation, *input*, x_3	cible, *target*, sortie désirée, y_3
⋮	*etc…*	⋮
n	entrée, observation, *input*, x_n	cible, *target*, sortie désirée, y_n

Training set size, number of examples/ samples

testing point  ? ? ?

Goal: design an algorithm producing an output which is a good prediction of the target

The algorithm finds a "good" function x -> y

Today's lecture

- Terminology reminder
- Formalization of the learning problem
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

Learning problem

- Dataset: $D = (Z_1, Z_2, \cdots, Z_n)$ generated by *nature*

In the supervised setting: $Z_i=(X_i, Y_i)$

- I.I.D. data (independent and identically distributed):
  - Drawn from the same unknown distribution $p(Z)$, each example drawn independently from the others.

⇨ Ordering of examples typically does not matter
(shuffling the data has no influence on the formulation of the learning problem)

⚠ **Not all datasets are I.I.D.** ⚠

Learning problem

binary: $\mathcal{Y} = \{-1, 1\}, \{0, 1\}$

multiclass: $\mathcal{Y} = \{1, m\}, \{0, m-1\}$

- Three families of problems:
  - Classification: $Z = (X, Y) \in \mathbb{R}^d \times \mathcal{Y}$
  - Regression: $Z = (X, Y) \in \mathbb{R}^d \times \mathbb{R}$
  - Density estimation: $Z = X \in \mathbb{R}^d$
- Hypothesis space (space of possible solutions/functions/predictors):
  - Classification: $f : \mathbb{R}^d \to \mathcal{Y}$
  - Regression: $f : \mathbb{R}^d \to \mathbb{R}$
  - Density estimation: $f \in \mathcal{P}$ (space of probability density functions / distributions)

Common tasks and problems in machine learning

	supervised	supervised	unsupervised
	Classification	**Regression**	**Density estimation**
What is the target y	one category out of m possible classes.	a real number	no target y!
Output domain	$y \in$ {-1,1} or $y \in$ {1, ..., m} or $y \in$ {0, 1, ... , m-1}, ...	$y \in \mathbb{R}$	no target y!
What does $f(x)$ predict?	class/label of x (the most likely class associated with x)	expectation of the output ("mean" y) corresponding to x. $E[\, Y \mid X=x \,]$	the density $p(x)$ (how likely is x?)
Loss/cost function usually optimized	Classification error: $L((x,y),f) = I_{\{f(x) \neq y\}}$	Squared error: $L((x,y),f) = (f(x)-y)^2$	Negative log likelihood: $L(x,f) = -\log f(x)$

Today's lecture

- Terminology reminder
- Formalization of the learning problem
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

Example of classification problem

Iris flower data set

From Wikipedia, the free encyclopedia

The **Iris flower data set** or **Fisher's Iris data set** is a multivariate data set introduced by Sir Ronald Aylmer Fisher (1936) as an example of discriminant analysis.[1] It is sometimes called **Anderson's Iris data set** because Edgar Anderson collected the data to quantify the geographic variation of *Iris* flowers in the Gaspé Peninsula.[2]

The dataset consists of 50 samples from each of three species of *Iris* flowers (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample, they are the length and the width of sepal and petal, in centimeters. Based on the combination of the four features, Fisher developed a linear discriminant model to distinguish the species from each other. It is used as a typical test for many other classification techniques.

$\mathbf{x} \in \mathbb{R}^4$

Fisher's Iris Data

Y

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*

etc...

5.7	2.8	4.5	1.3	*versicolor*
6.3	3.3	4.7	1.6	*versicolor*
4.9	2.4	3.3	1.0	*versicolor*
6.6	2.9	4.6	1.3	*versicolor*
5.2	2.7	3.9	1.4	*versicolor*
5.0	2.0	3.5	1.0	*versicolor*

etc...

7.7	3.0	6.1	2.3	*virginica*
6.3	3.4	5.6	2.4	*virginica*
6.4	3.1	5.5	1.8	*virginica*
6.0	3.0	4.8	1.8	*virginica*
6.9	3.1	5.4	2.1	*virginica*

etc...

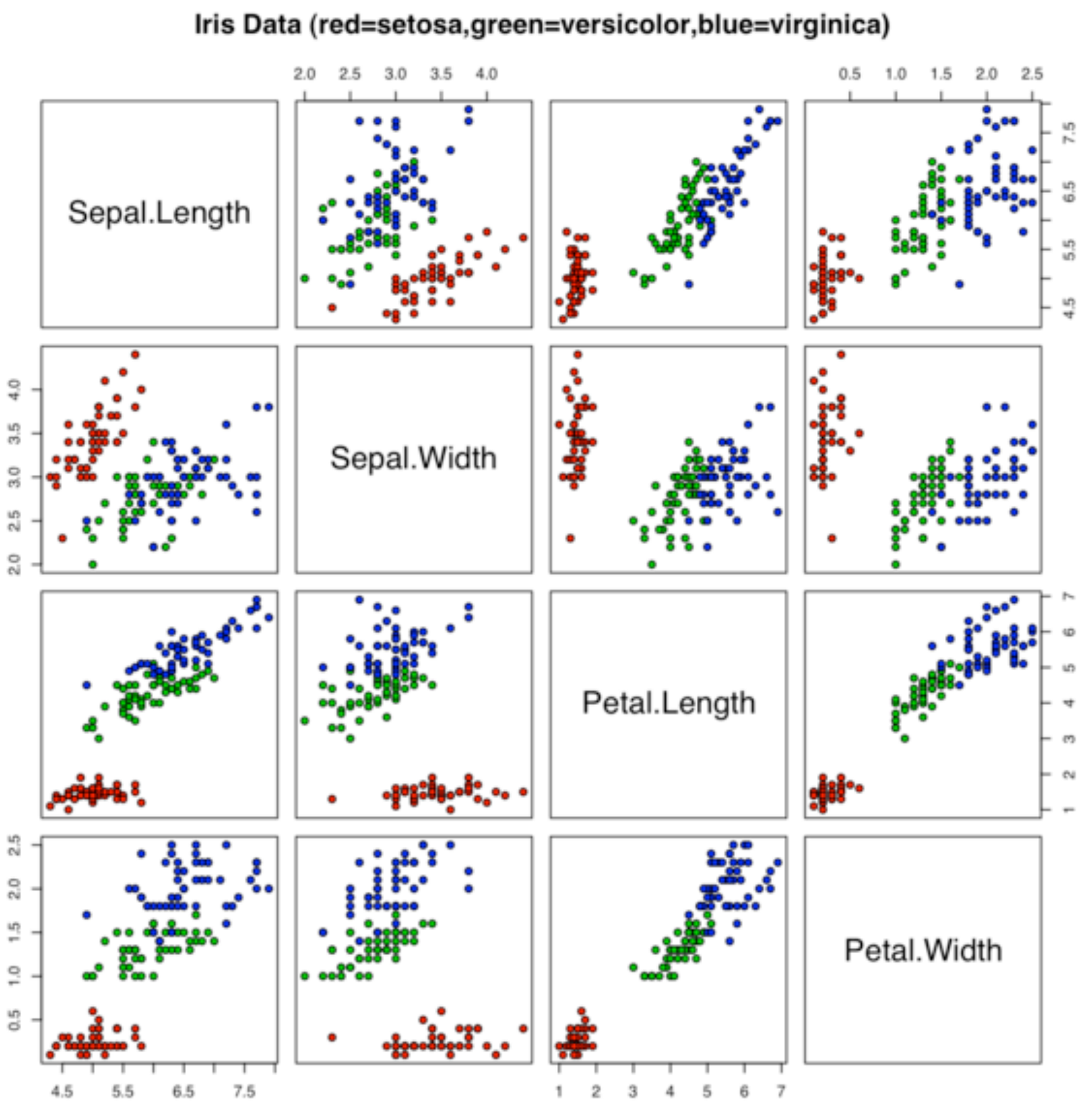
n= 150

Iris setosa

Iris versicolor

Iris virginica

Labels as a function of two of the features



Different tasks can be tackled on the same dataset!

- Classification: predict the iris species given petal and sepal lengths.
- Regression: predict petal lengths given sepal dimensions and iris species.
- Density estimation: given sepal and petal lengths of an unknown flower, how likely is it that this flower is an iris setosa?

Today's lecture

- Terminology reminder
- Formalization of the learning problem.
- Examples of regression, classification and density estimation problems
- **Histogram methods**, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

Histogram methods

Algorithms relying on splitting up the input space (histogram methods)

A simple idea: cut the space into small cubes...

A simple classification algorithm

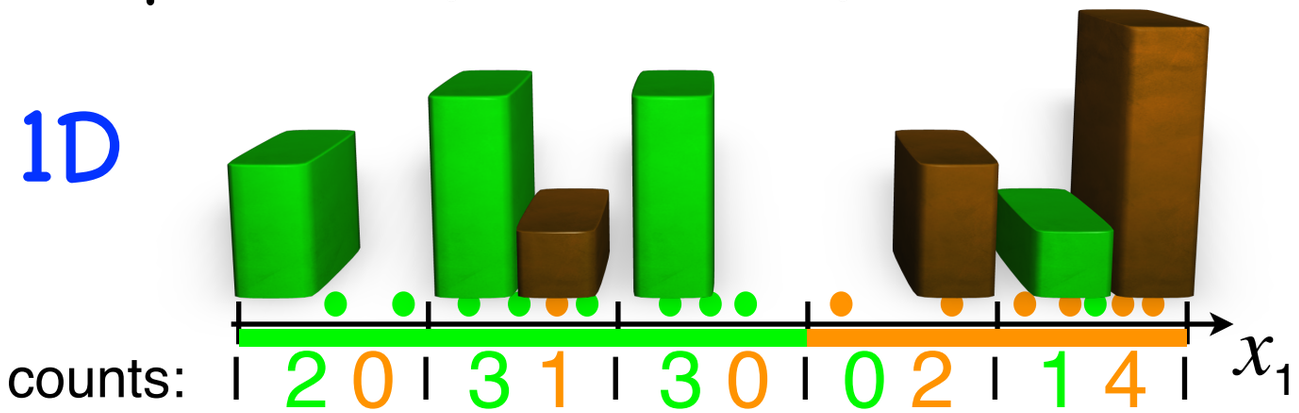
Any learning algorithm needs to be able to predict an output for any point in the input space. (ex: $x \in \mathbb{R}^d$)
With this in mind, here's a simple idea:

- Split the space in small regions!
- Training: Count how many training points are in each region, for each of the classes.
- Testing: Identify the region in which the test point lies, and return the majority class for this region.

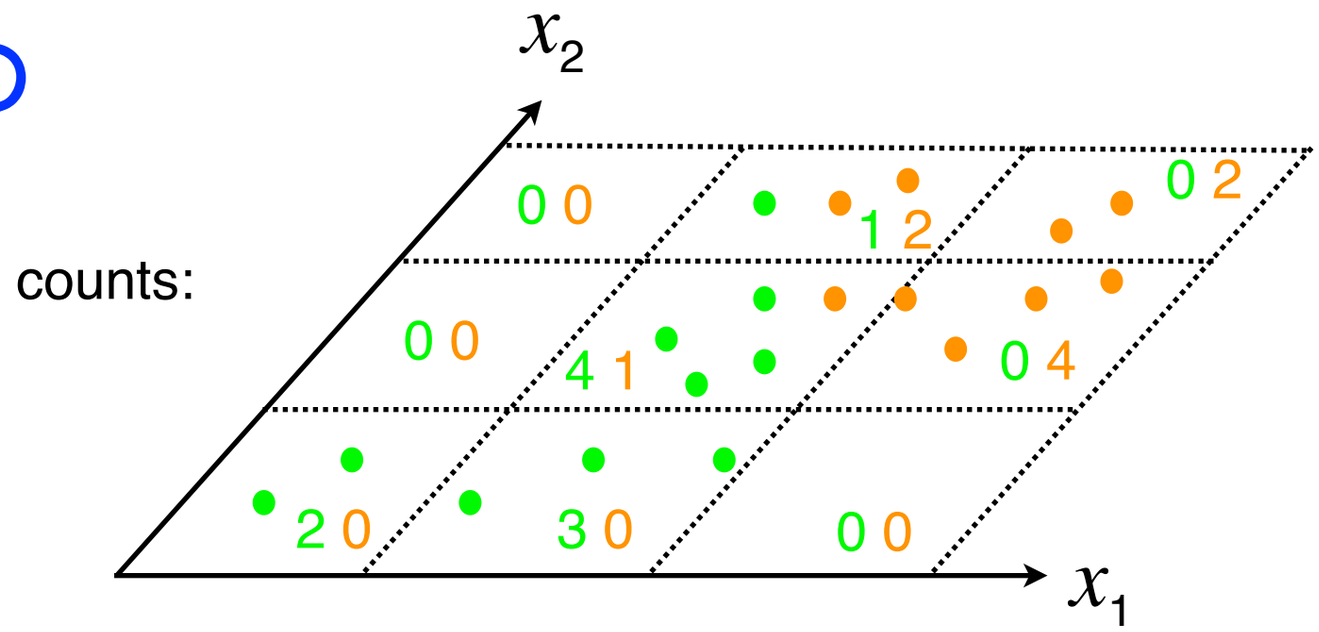
Classification

- Suppose there exists an (unknown) process generating pairs (x,y), where y is the label (• or •)

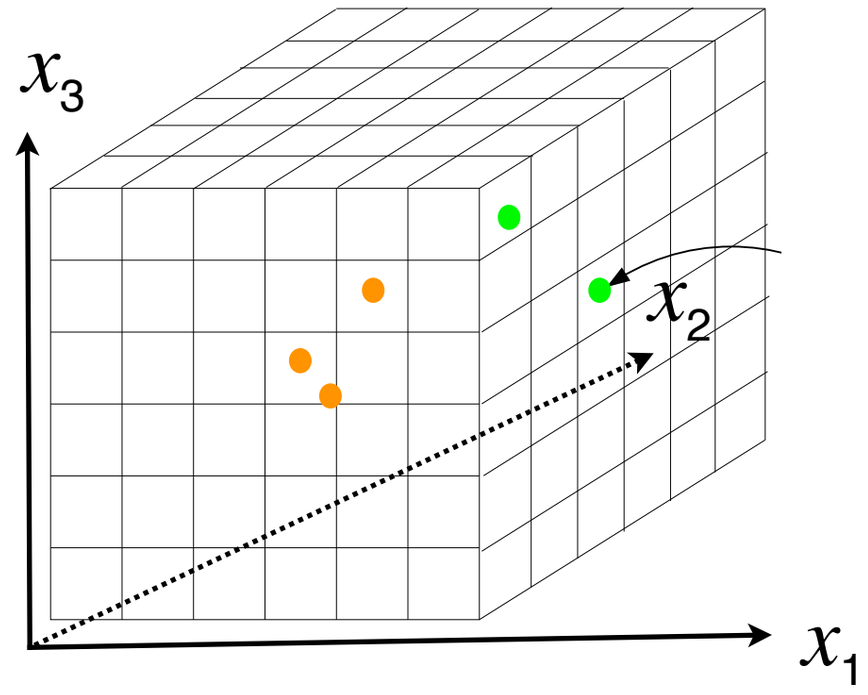
1D



2D



3D

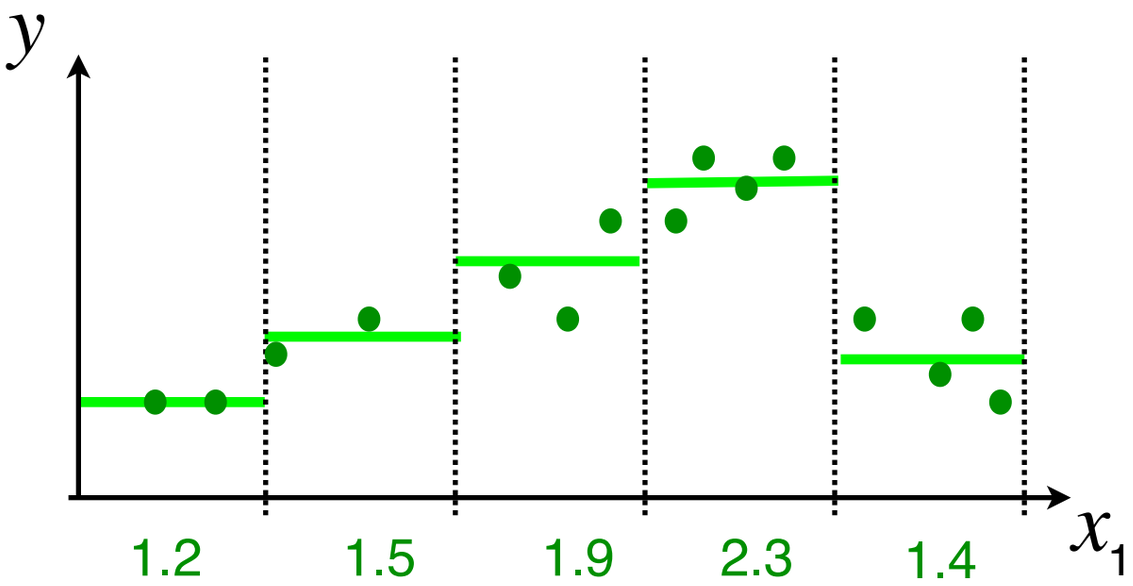


dD ...

Regression

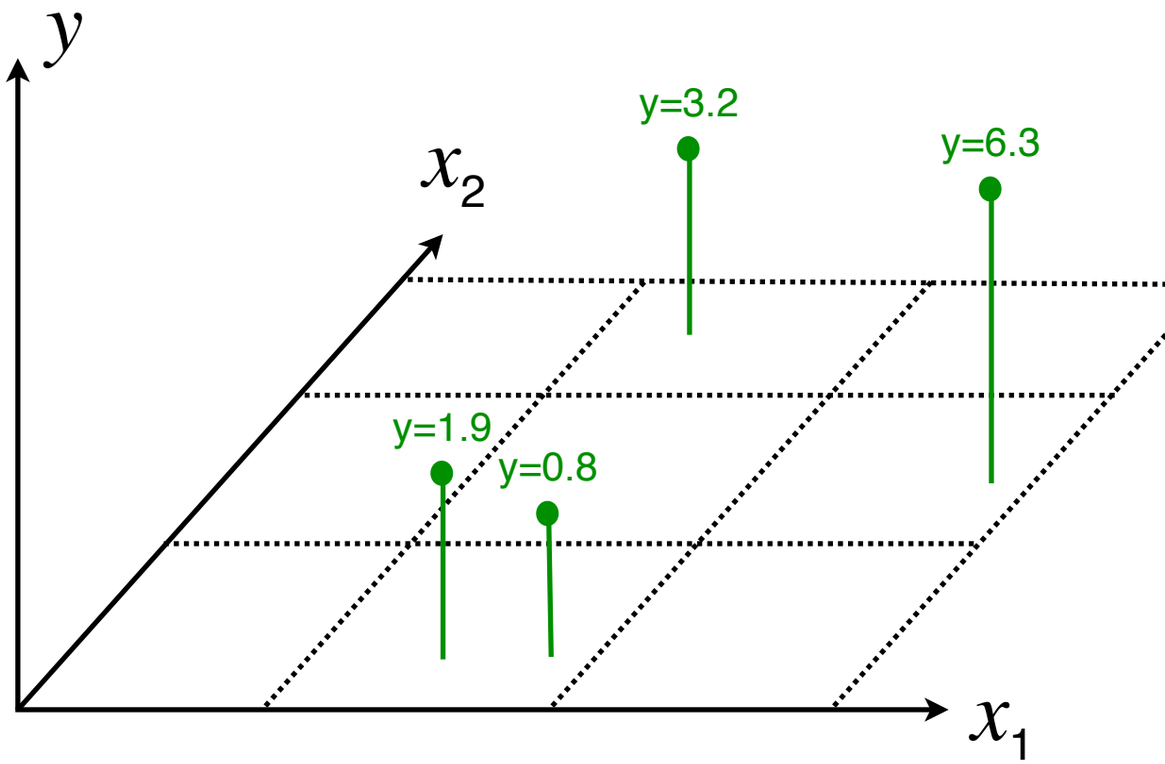
- Suppose there exists an (unknown) process generating pairs (x,y), where y is a real value.

1D



means: $E(y|x \in case)$

2D



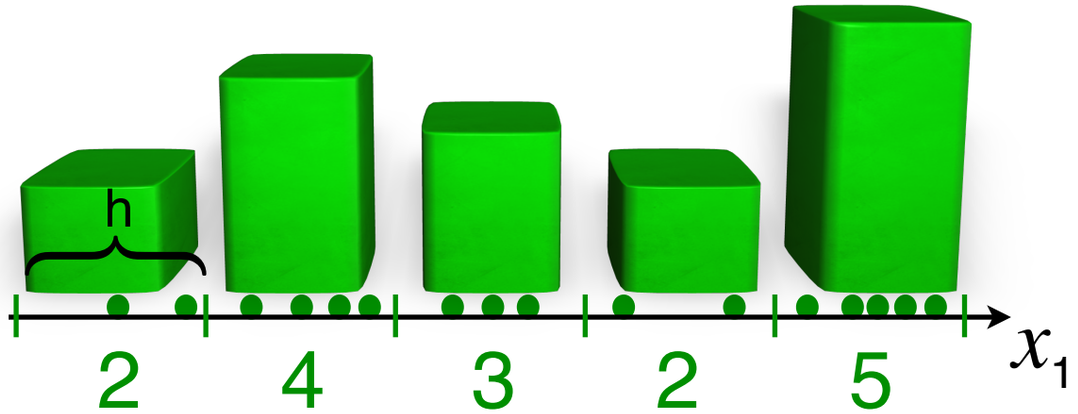
3D …

dD …

Density estimation

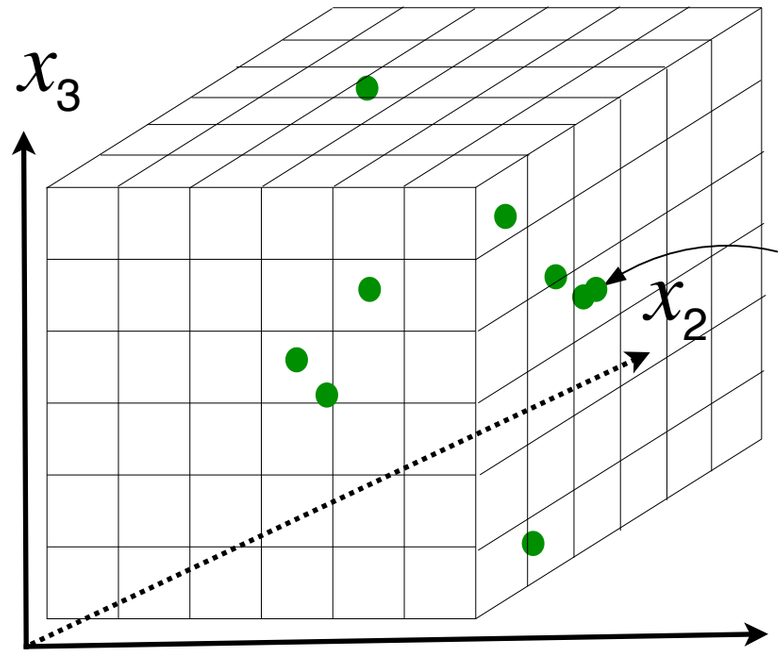
- Suppose there exists an (unknown) process generating observations x.

1D



comptes:	2	4	3	2	5
densité de probabilité: $p(x)$	2/16/h	4/16/h	3/16/h	2/16/h	5/16/h

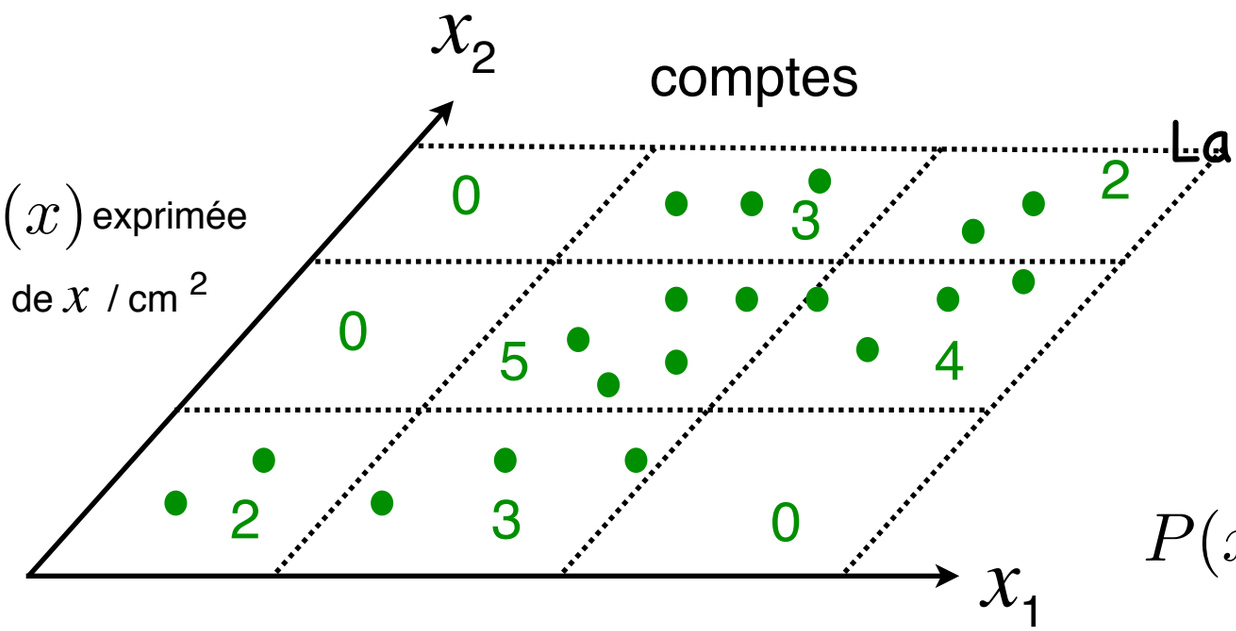
3D



2D

Densité de probabilité $p(x)$ exprimée

par ex. en probabilité de x / cm ^{2}



La densité de prob. doit intégrer à 1 sur tout l'espace:

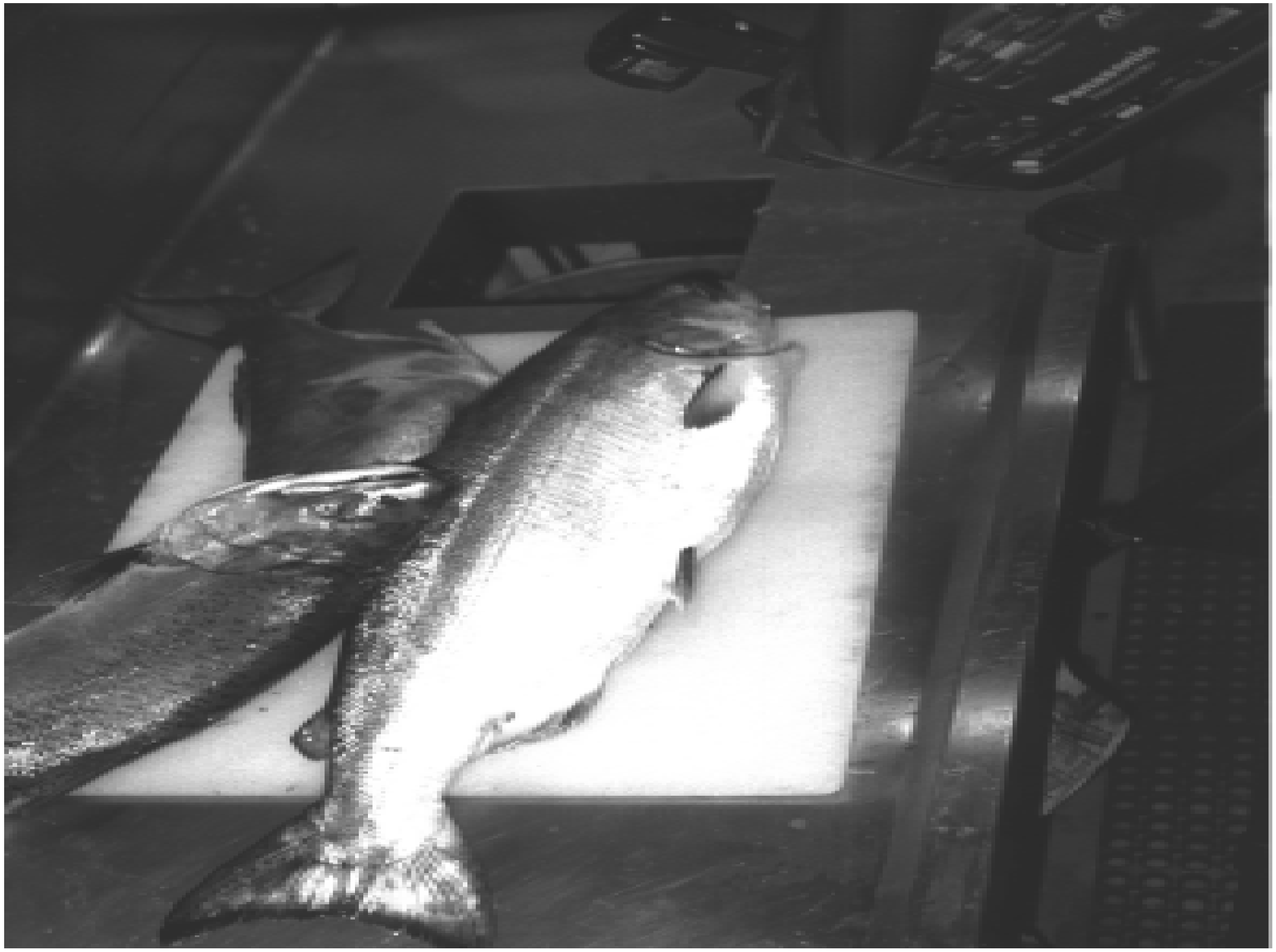
$$\int p(x)dx = 1$$

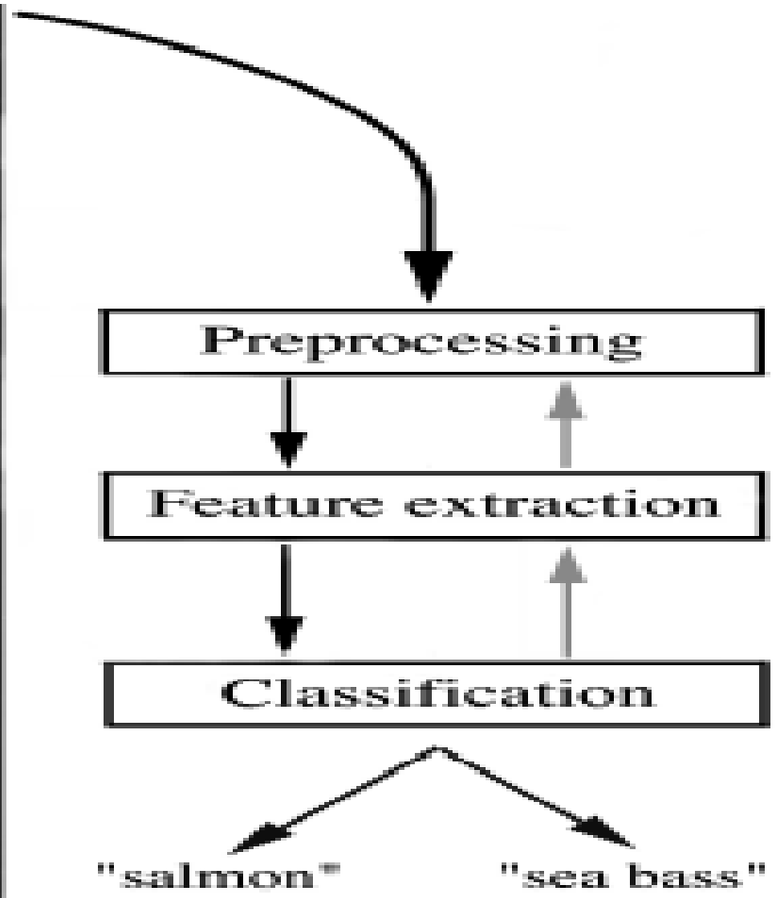
$$P(x \in region) = \int_{region} p(x)dx$$

Example of a classification task

- Differentiate between *salmons* and *see bass* on a conveyor belt:
  - input data (camera)
  - image pre-processing
  - feature extraction: width, height, luminosity, etc.
  - design of a classification function:

$$f : \{features\} \mapsto \{salmon, bar\}$$

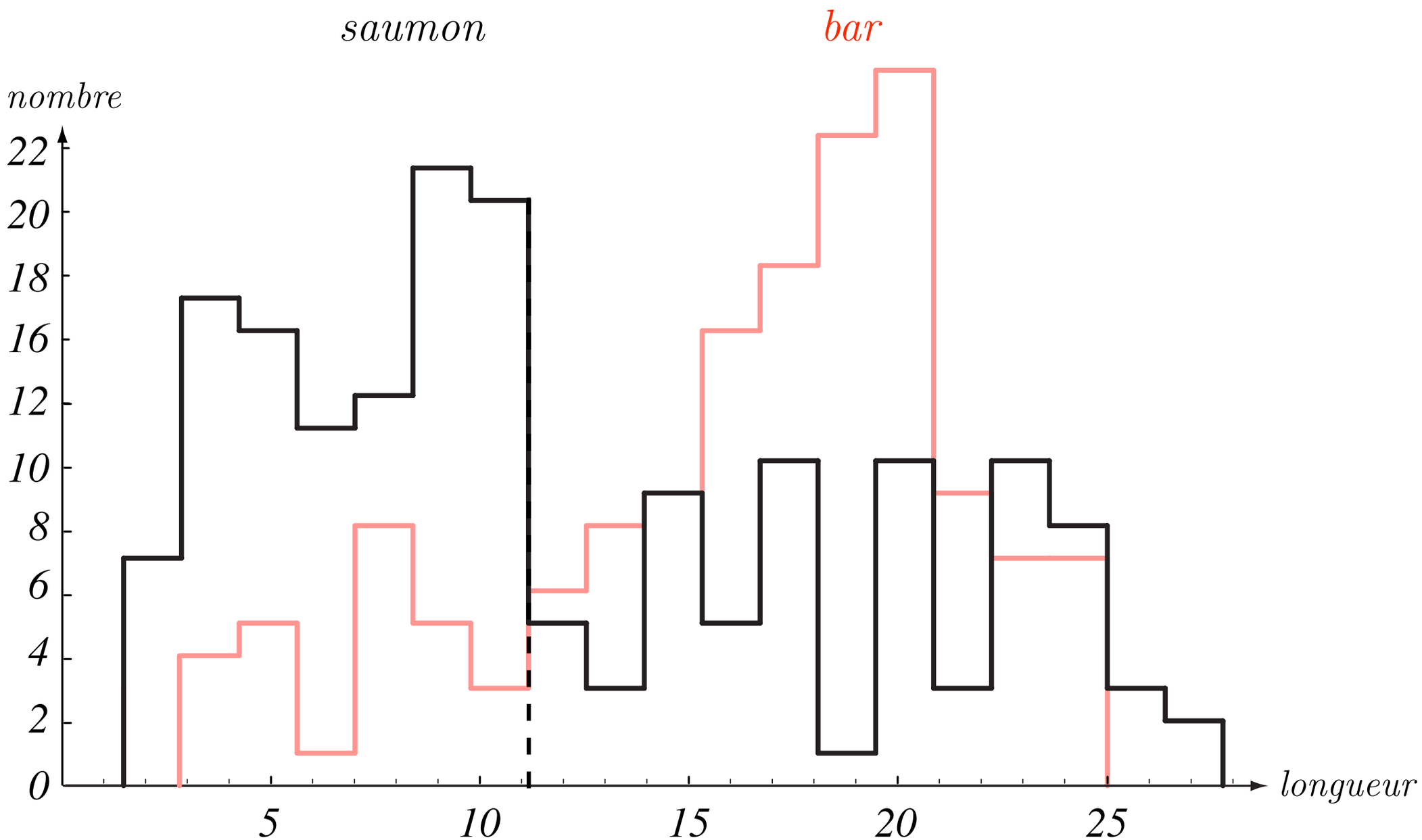




Examples of features extracted after preprocessing:

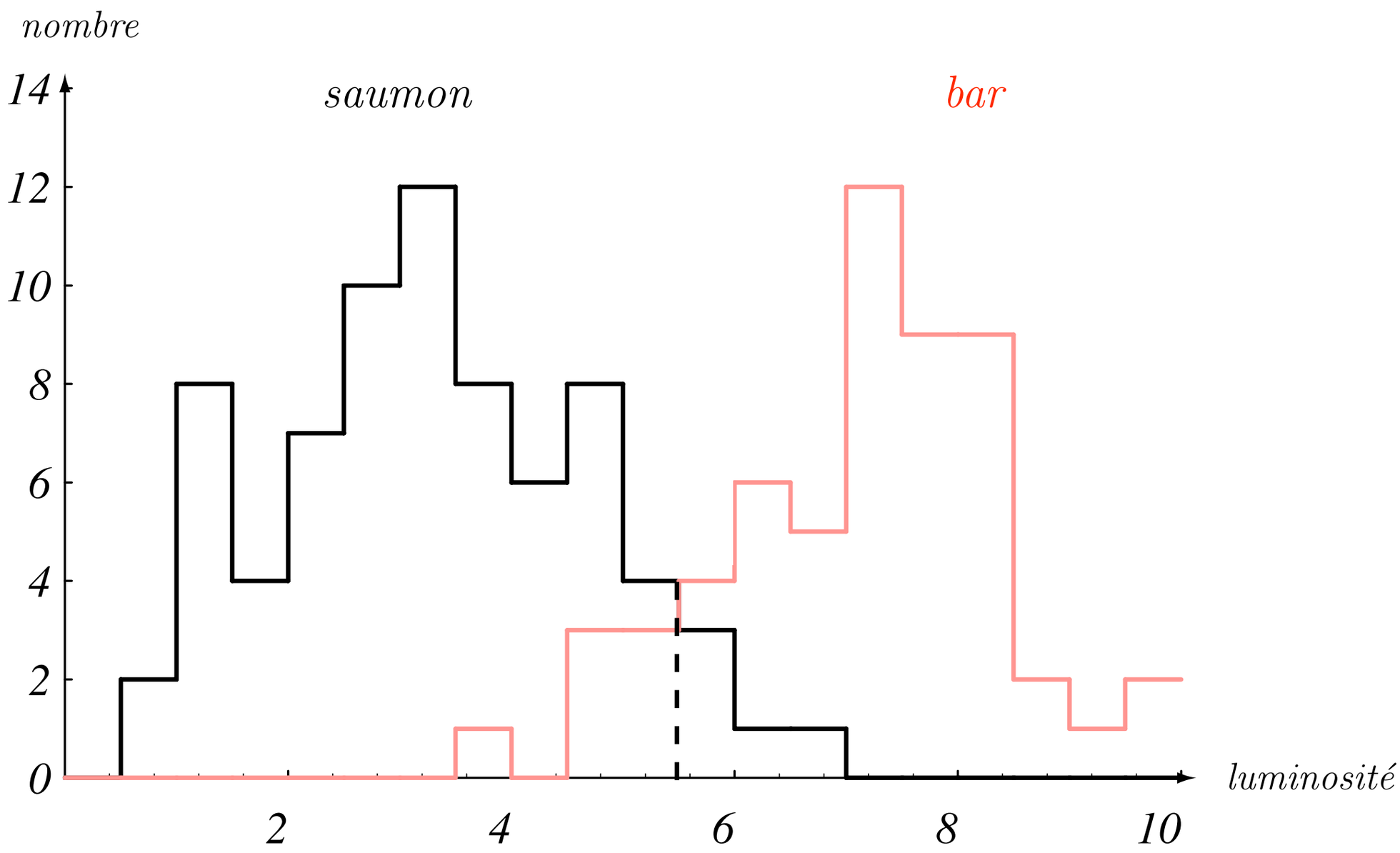
x = (width, height, luminosity, dorsal fin length, mouth position, etc.)

Histogram for height attribute (on training set)



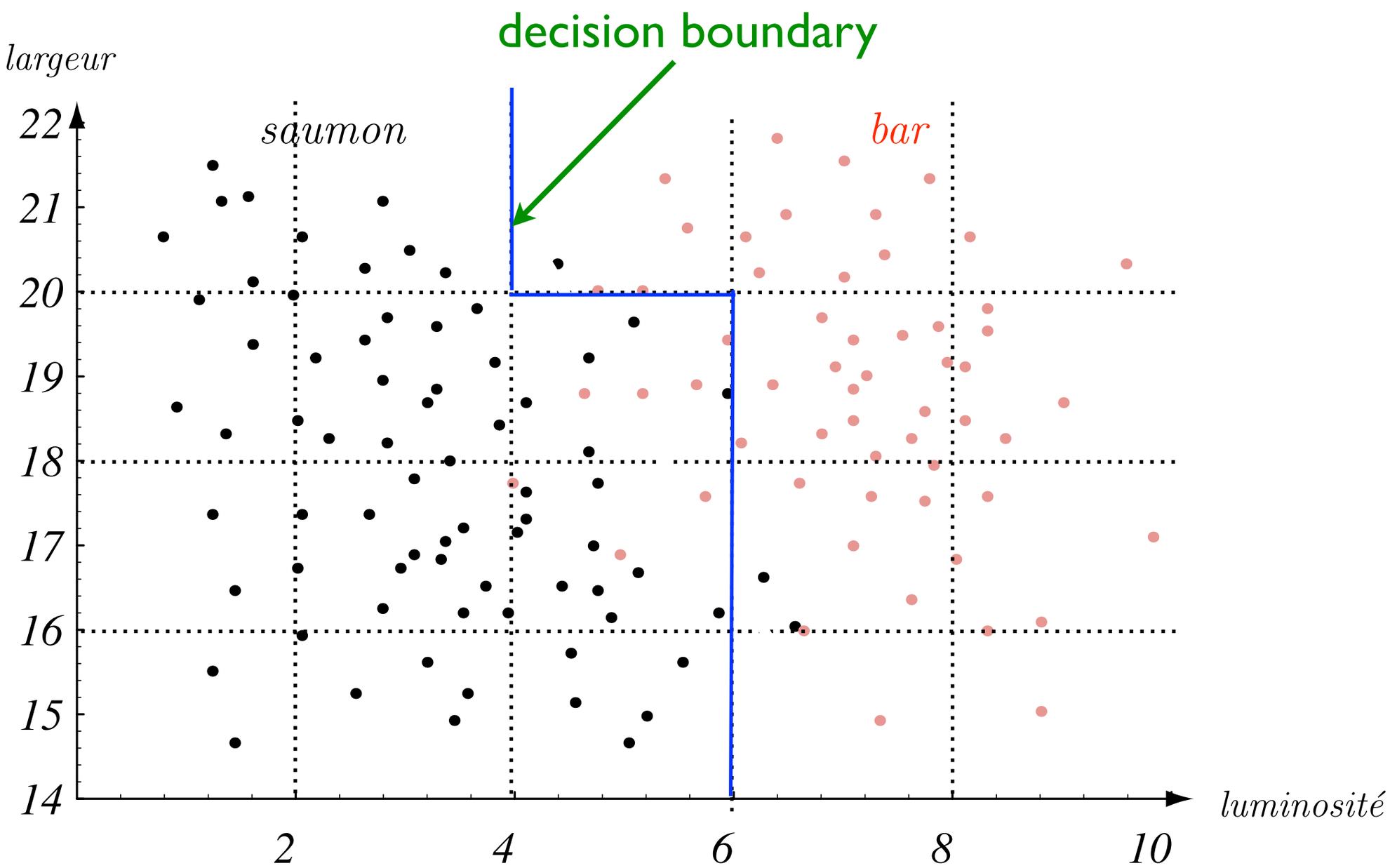
#errors: 40+48 = 88

Histogram using luminosity attribute (on training set)



#errors: 7+5 = 12

Histogram using two features (width and luminosity)



#errors : 3 + 7 =10

- More dimensions (i.e. more features) usually imply more informations to make the good prediction.
- As we get more features, the classes are more easily separated
- This sounds good but....

Today's lecture

- Terminology reminder
- Formalization of the learning problem.
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

CURSE OF DIMENSIONALITY

Ex: how many regions for a grid with 10 bins in a space of dimension d?

- d=1 : 10 regions
- d=2 : 10x10=100 regions
- d=3 : 10x10x10=1000 regions
- d=10 : 10 = 10 000 000 000 regions
- If each dimension is divided in m bins we get m^d regions!

The "size" of the hypothesis space (the space of functions to explore) grows exponentially with d!

With n=100 000 training points ± uniformly distributed:

- d=1 : 100 000/10 = 10000 points/region
- d=2 : 100 000/100 = 1000 points/region
- d=3 : 100 000/1000 = 100 points/region
- d=10 : $100\,000/10^{10} = 10^{-5}$ points/region
- d=100 : $100\,000/10^{100} = 10^{-95}$ points/region

In high dimension, most of the regions (where a test point is likely to appear) are empty!!!

How bad is this curse?

- Histogram methods work well in low dimension
  (1, 2, sometimes 3...)
- But they are worthless in high dimension!
- The curse of dimension affects more or less all learning algorithms... But some are considerably more sensitive than others.

Today's lecture

- Terminology reminder
- Formalization of the learning problem.
- Examples of regression, classification and density estimation problems
- Histogram methods, applied on classification, regression and density estimation tasks.
- Curse of dimensionality
  First discussion on capacity of models

Steps for designing a learning algorithm

- Intuitive understanding of the algorithm. Know how to explain/describe it with words!
- Mathematic formalization of the algorithm.
- Write the algorithm in pseudo-code.
- Implement it in a programming language.
- Training/Testing (debugging...) the algo on simple problems in low dimension, where we can visually check the data/outcome/decision boundary and where training is fast (don't wait on experiment while debugging)!
- Evaluate the algorithm on real world problems, and compare it with other learning algorithms.

Ex: Classification with histograms

hyper-parameters

Histograme-2D-Classifier (m1, x1min, x1max, m2, x2min, x2max, nclasses)

```
C ← array(m1, m2, nclasses) initialized to 0.
taille1 = (x1max-x1min)/m1,   taille2 = (x2max-x2min)/m2
```

training set

```
train(Dn):
    For each (x,y) ∈ Dn :
        i = floor((x[0]-x1min)/taille1),   j = floor((x[1]-x2min)/taille2)
        C[i,j,y]++
```

test point

```
class_prob(x):      # return a probability vector: [P(y=1 | x), ..., P(y=nclasses | x)]
    i = floor((x[0]-x1min)/taille1),   j = floor((x[1]-x2min)/taille2)
    count_vector = C[i,j,:]
    return count_vector / sum(count_vector)
```

test point

```
f(x):      # predict the class of x (return the index of the most likely class)
    probs = class_prob(x)
    return ArgMax[probs]
```

Complexity analysis

Classifier with histograms

Variables

n: number of training examples,
d: input dimension
C: number of classes
m: number of divisions for each dimensions,

Memory complexity (both for training and testing)

$O(C\ m^d)$

Time complexity for training

O(d n)

Time complexity for testing

O(d+C)