Pattern Recognition for System Monitoring - An Overview

Thierry Denoeux

University of Compiègne

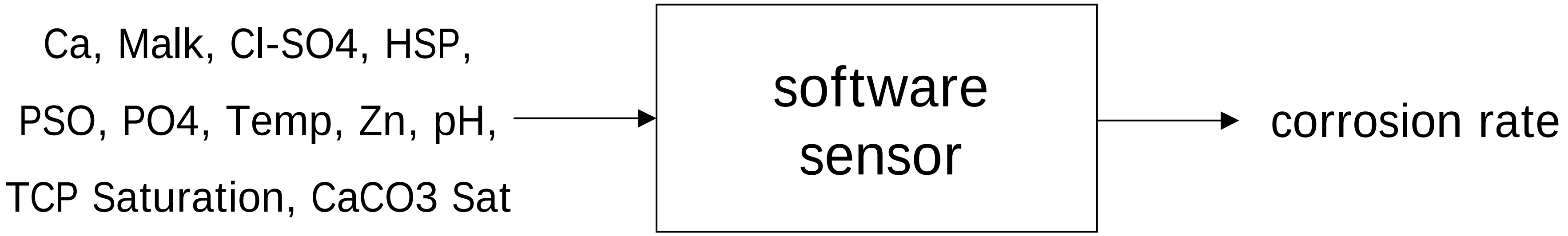
CNRS (National Center for Scientific Research)

Compiègne, France

Outline

- Pattern recognition for process control and monitoring: two generic applications
  - software sensors
  - system diagnosis
- The development cycle of a PR system
  - analysis (choice of sensors, data collection, ...)
  - design (training, model selection and performance assessment)
  - implementation (robustness, refinement, adaptation, ...)
- Case study: prediction of optimal coagulant dosage in water treatment plant.
- Conclusions

Software Sensor: Example



Ca, Malk, Cl-SO4, HSP,

PSO, PO4, Temp, Zn, pH,

TCP Saturation, CaCO3 Sat

software sensor

corrosion rate

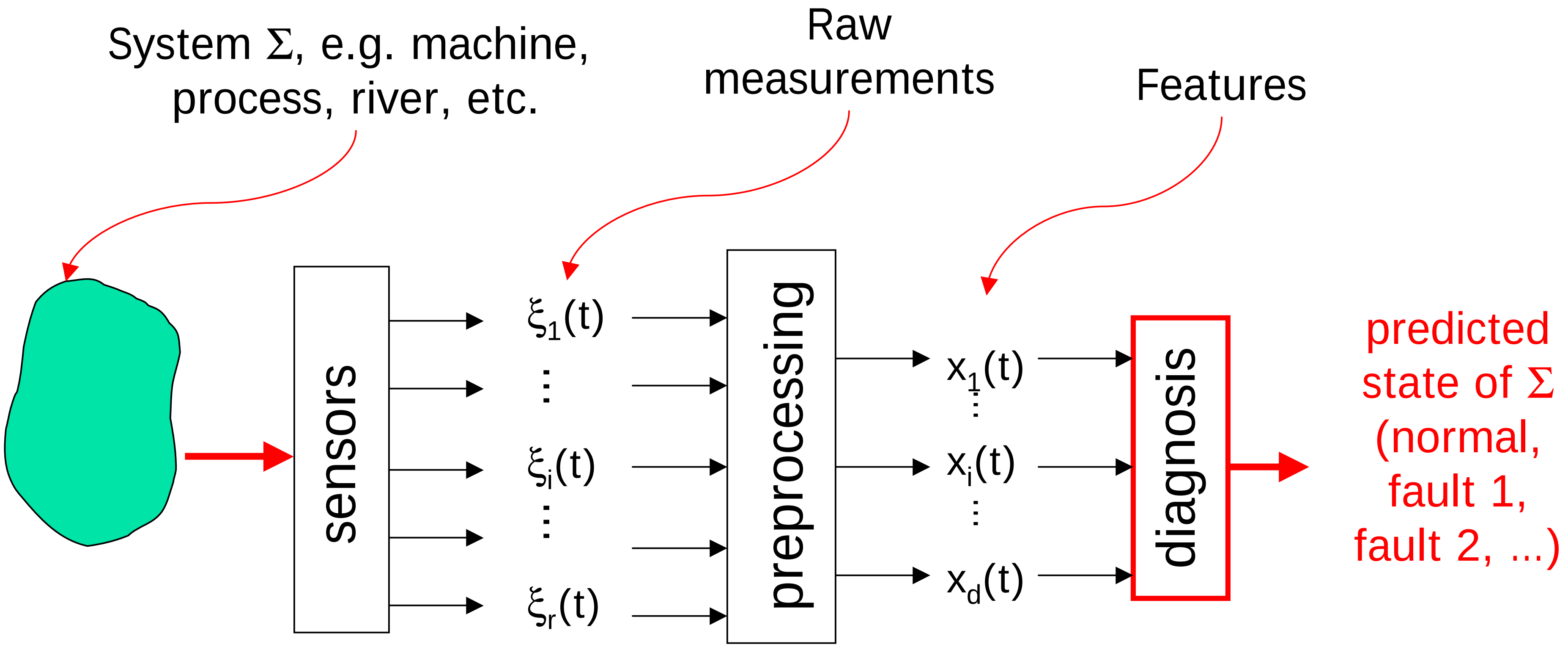
software sensor: procedure for estimating a quantity of interest (output, response variable) from observable quantities (input variables, features)

Buiding a software sensor

The methodology for building a software sensor depends on the available information:

- domain knowledge (equations, physical laws, expert rules, ...) → deterministic or conceptual modeling approach (domain-specific)
- statistical knowledge (past observations of the input and output variables) → supervised-learning, pattern recognition approach (generic)

System Diagnosis



Design of a diagnosis system

The methodology depends on the nature of the available knowledge:

- A (logical or numerical) model for some or all of the states

  → model-based approach (AI, control engineering)

- No model, but historical data of past measurements and observations of the system state

  → feature-based, pattern recognition-based diagnosis

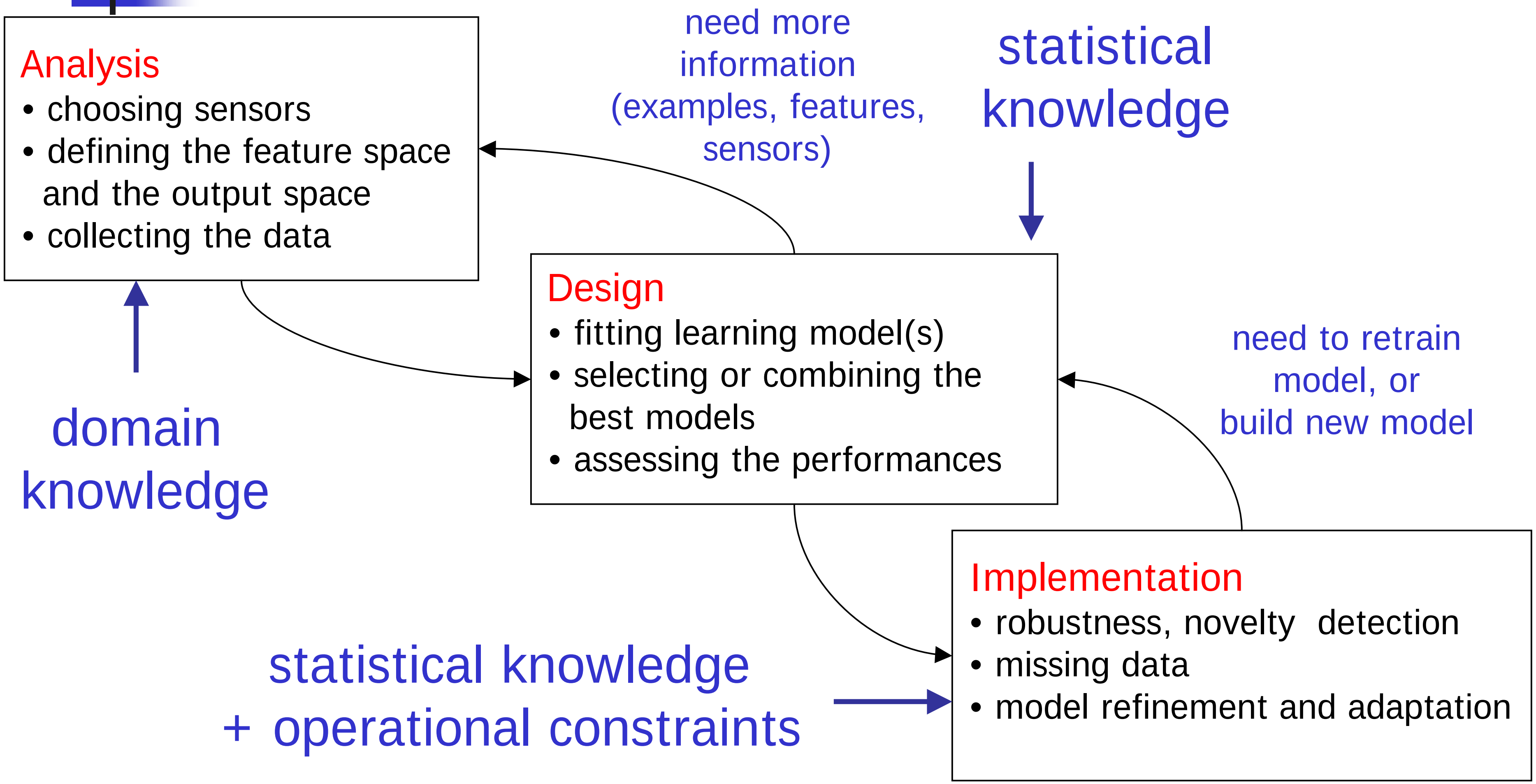
Common framework: Supervised learning

- Two groups of variables:
  - inputs, features, attributes $(x_1,\ldots,x_d)=\mathbf{x}$ (measurements, or functions of measurements)
  - output y
    - quantitative (regression),
    - qualitative $y \in \mathcal{G}=\{1,\ldots,K\}$ (classification)
- Learning set:

$$\mathcal{L}=\{(\mathbf{x}_i,y_i),\ i=1,\ldots,n\}$$

- Goal : predict y for a new case, based on observed input vector **x**.

The development cycle



Analysis
- choosing sensors
- defining the feature space and the output space
- collecting the data

domain knowledge

need more information (examples, features, sensors)

statistical knowledge

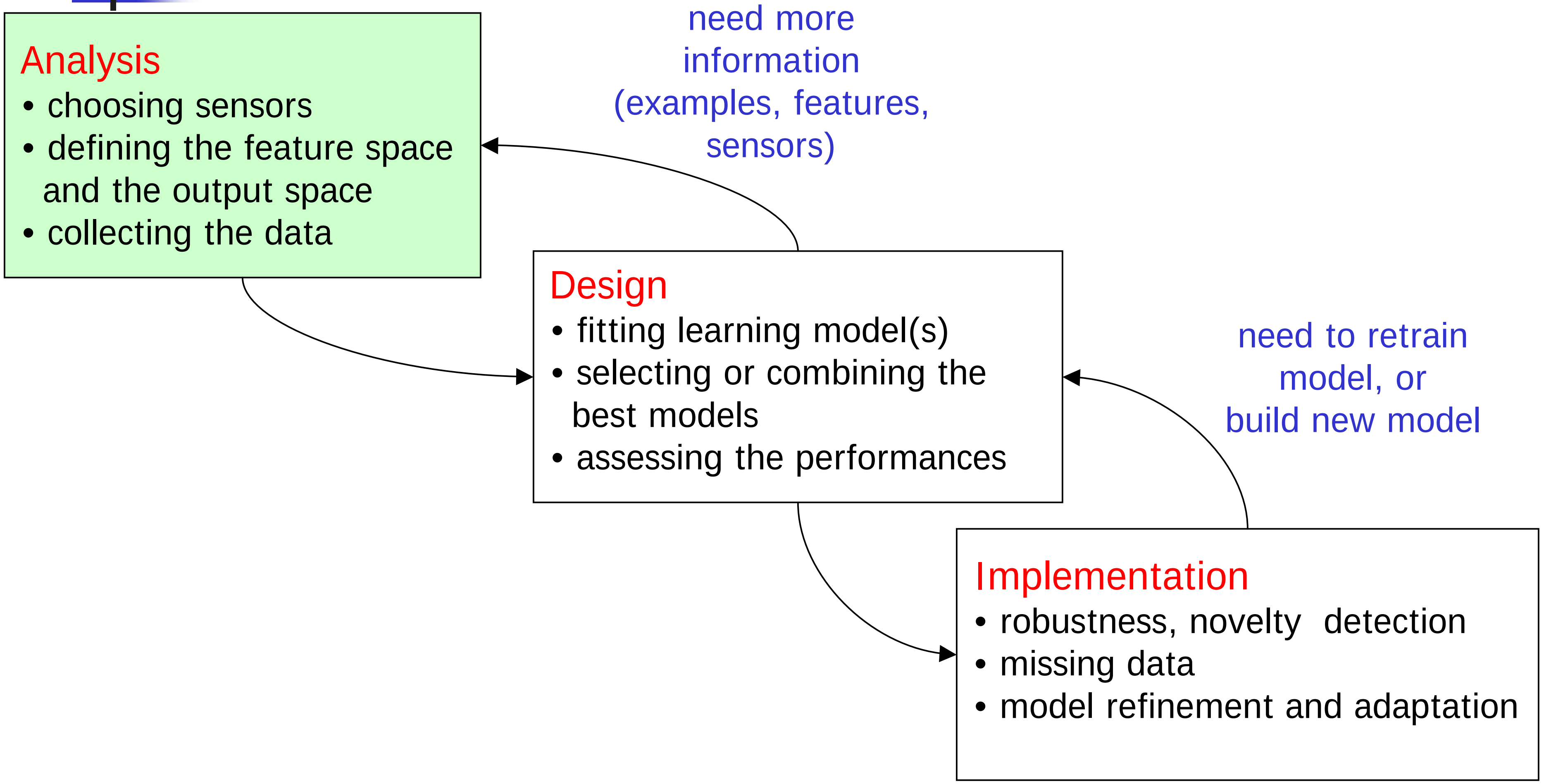
Design
- fitting learning model(s)
- selecting or combining the best models
- assessing the performances

need to retrain model, or build new model

Implementation
- robustness, novelty detection
- missing data
- model refinement and adaptation

statistical knowledge + operational constraints

Analysis



Analysis
- choosing sensors
- defining the feature space and the output space
- collecting the data

need more information (examples, features, sensors)

Design
- fitting learning model(s)
- selecting or combining the best models
- assessing the performances

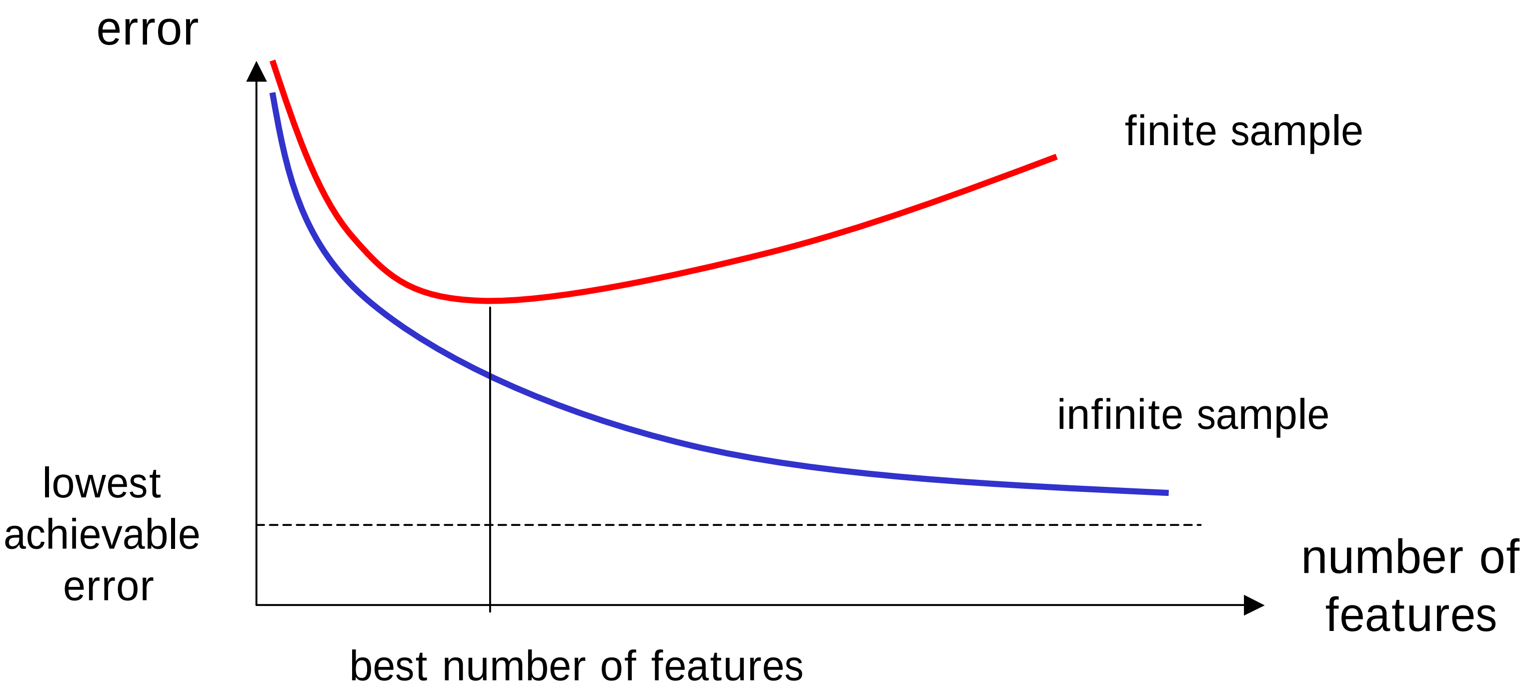
need to retrain model, or build new model

Implementation
- robustness, novelty detection
- missing data
- model refinement and adaptation

Analysis Phase: some guidelines

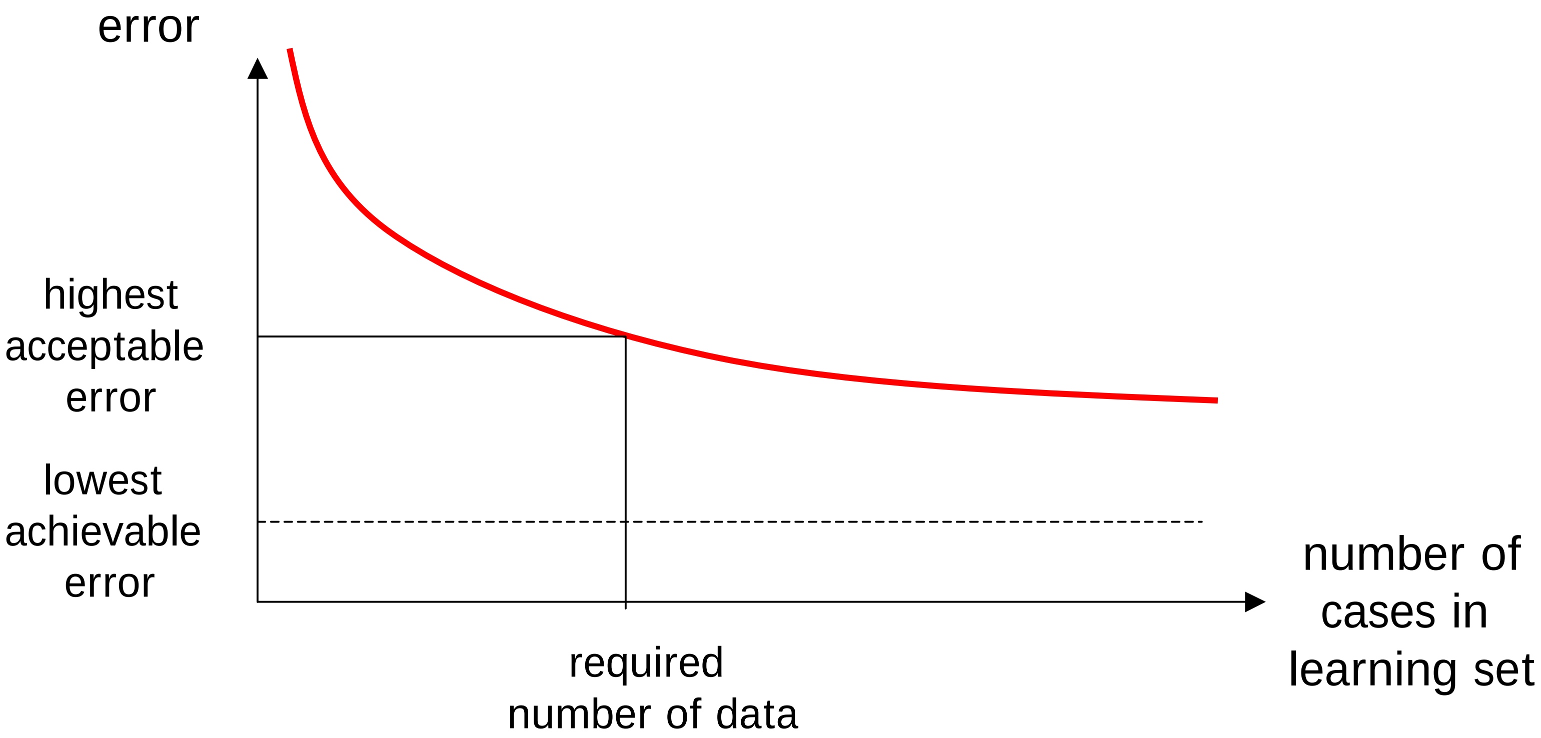
- Application specific, the choice of relevant sensor information can only be guided by domain knowledge.
- Typical questions:
  - How many features ?
  - How many data examples ?

How many features ?



Selected features should be limited to possibly relevant ones: Too many features may be harmul !

How many examples ?



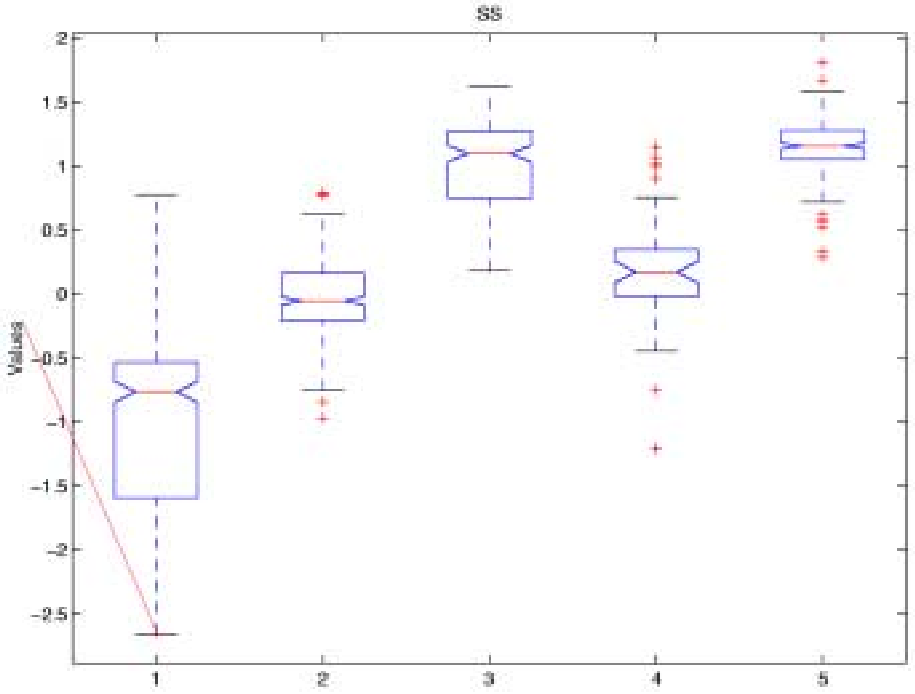
Exploratory data analysis

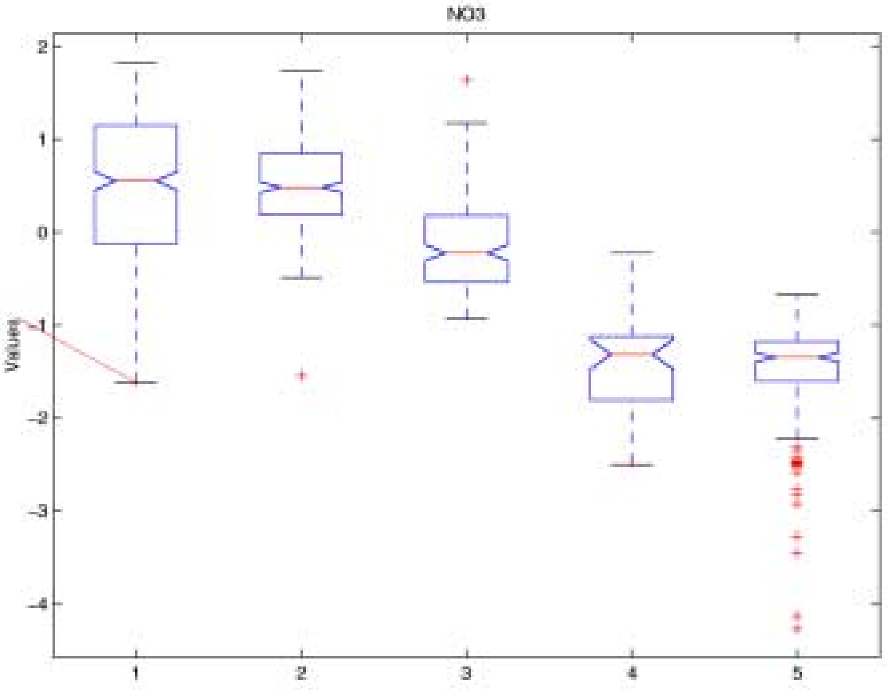
- A preliminary step to validate the data, and help selecting relevant features, using
  - Elementary techniques: visualize one or two variables at a time (histograms, boxplots, scatter plots, ...)
  - Multidimensional techniques: analyze the correlations between multiple features
- Examples of multidimensional techniques:
  - principal component analysis (PCA)
  - self-organizing feature maps

Example: classification of waste water for reuse

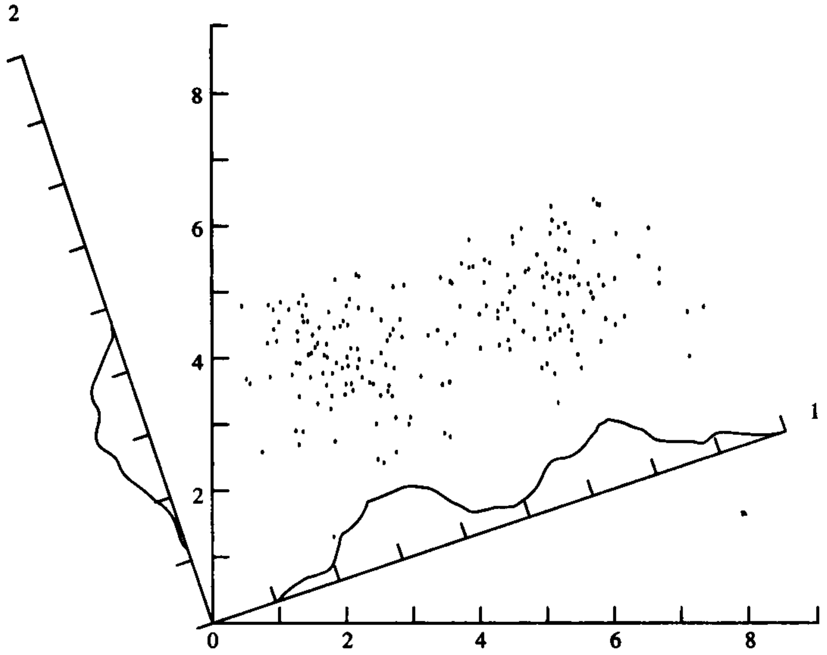
- Five classes of water quality, each class corresponding to a different possible usage
- 11 input features describing the chemical and bacteriological characteristics of water: suspended solids, TOC, conductivity, nitrate, etc.
- Problem: which features are relevant for classifying a water sample into one of the 5 categories ?

Analysis of a single input variable





Principal component analysis

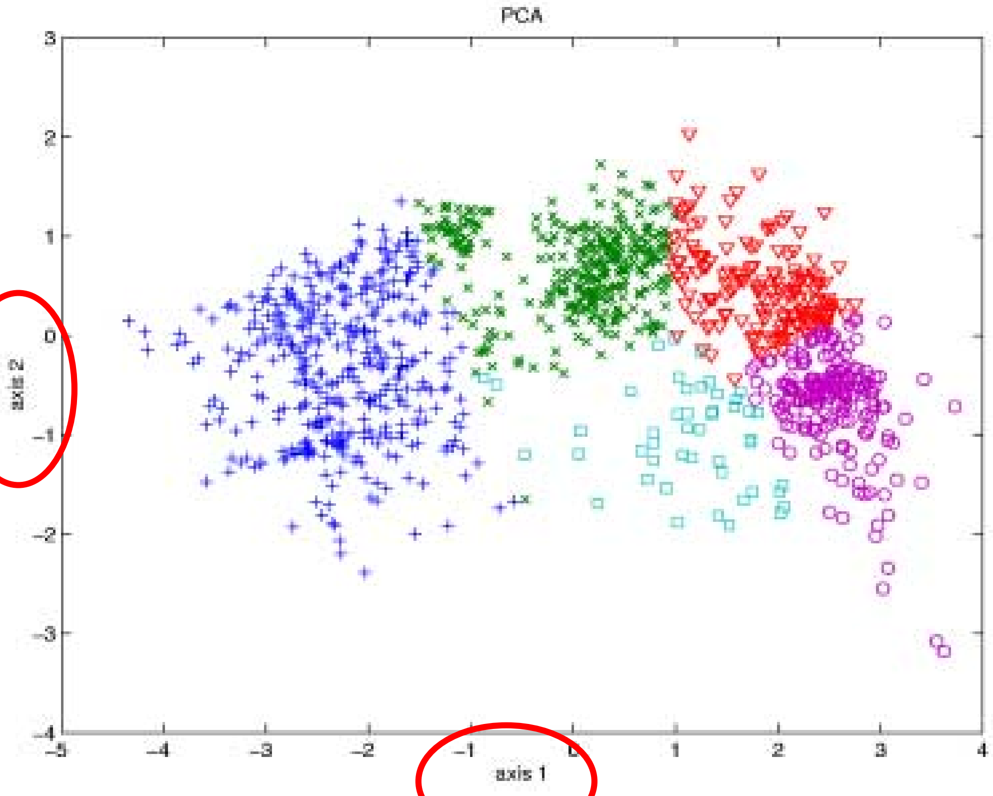


- Objective: summarize multi-dimensional data by defining a small number of “informative” features

- Approach: Find the directions in input space that maximize the variance (scatter) of projected data

- Each direction = linear combination of original features → new feature.

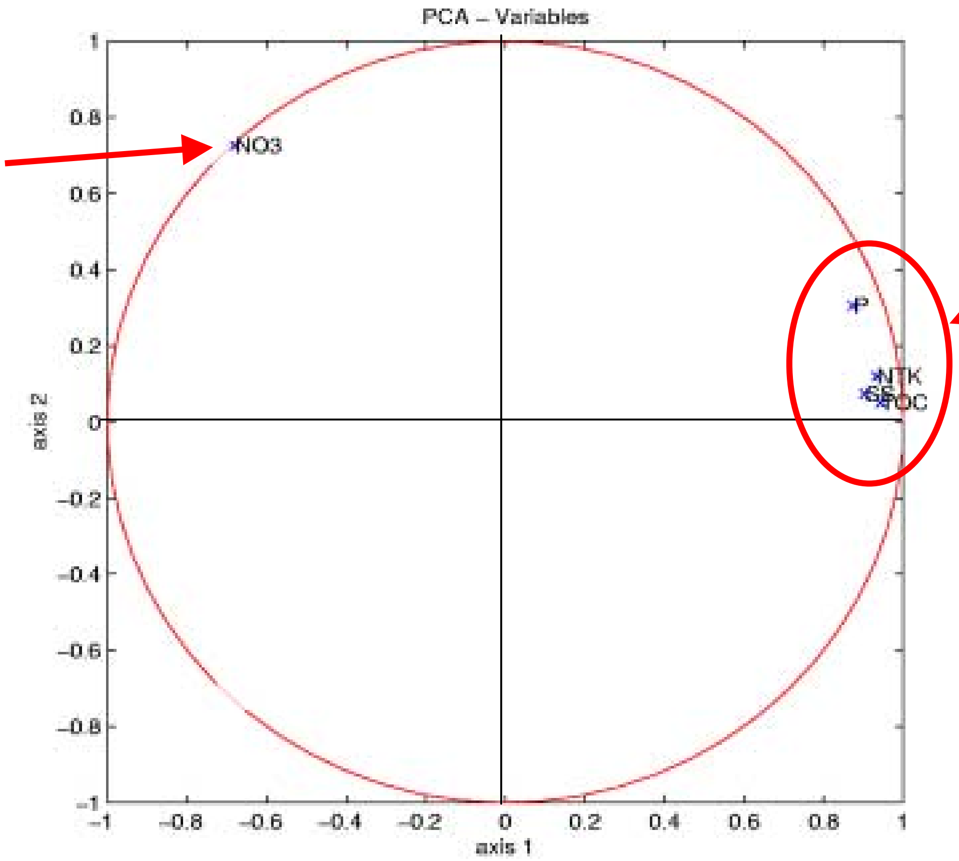
PCA: example (1)



2nd axis: 13 % of the total variance

1st axis: 76 % of the total variance (initial information)

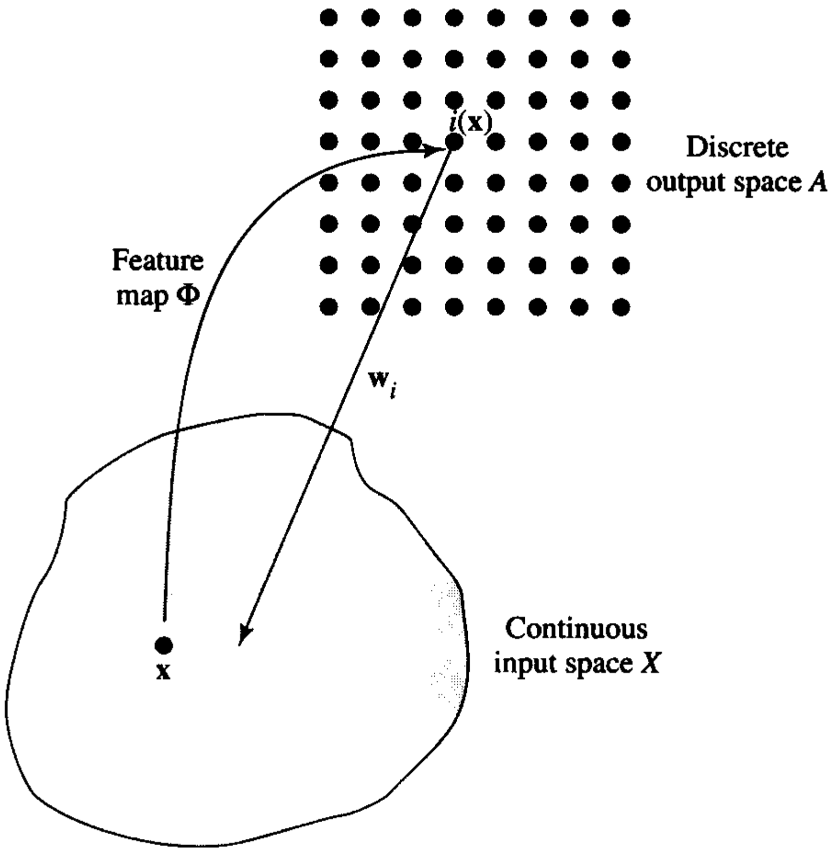
PCA: example (2)



the 2nd axis
is mostly explained
by this variable

group of
correlated
variables
→ 1st axis

Self-organizing feature maps

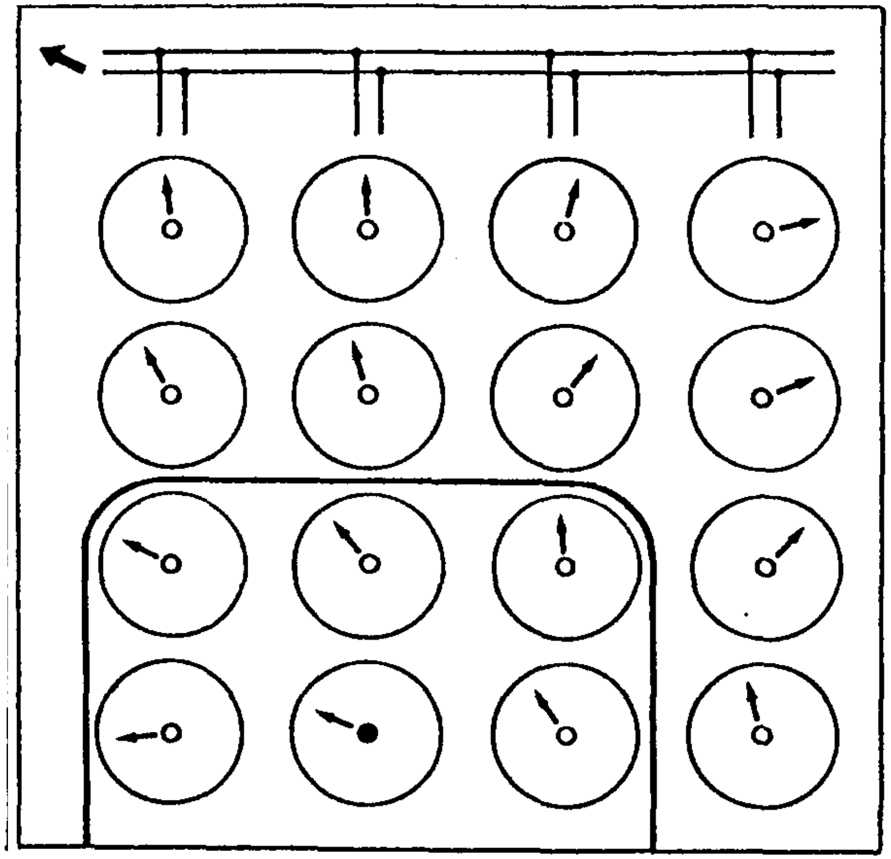


- A connectionist (artificial neural network) model.

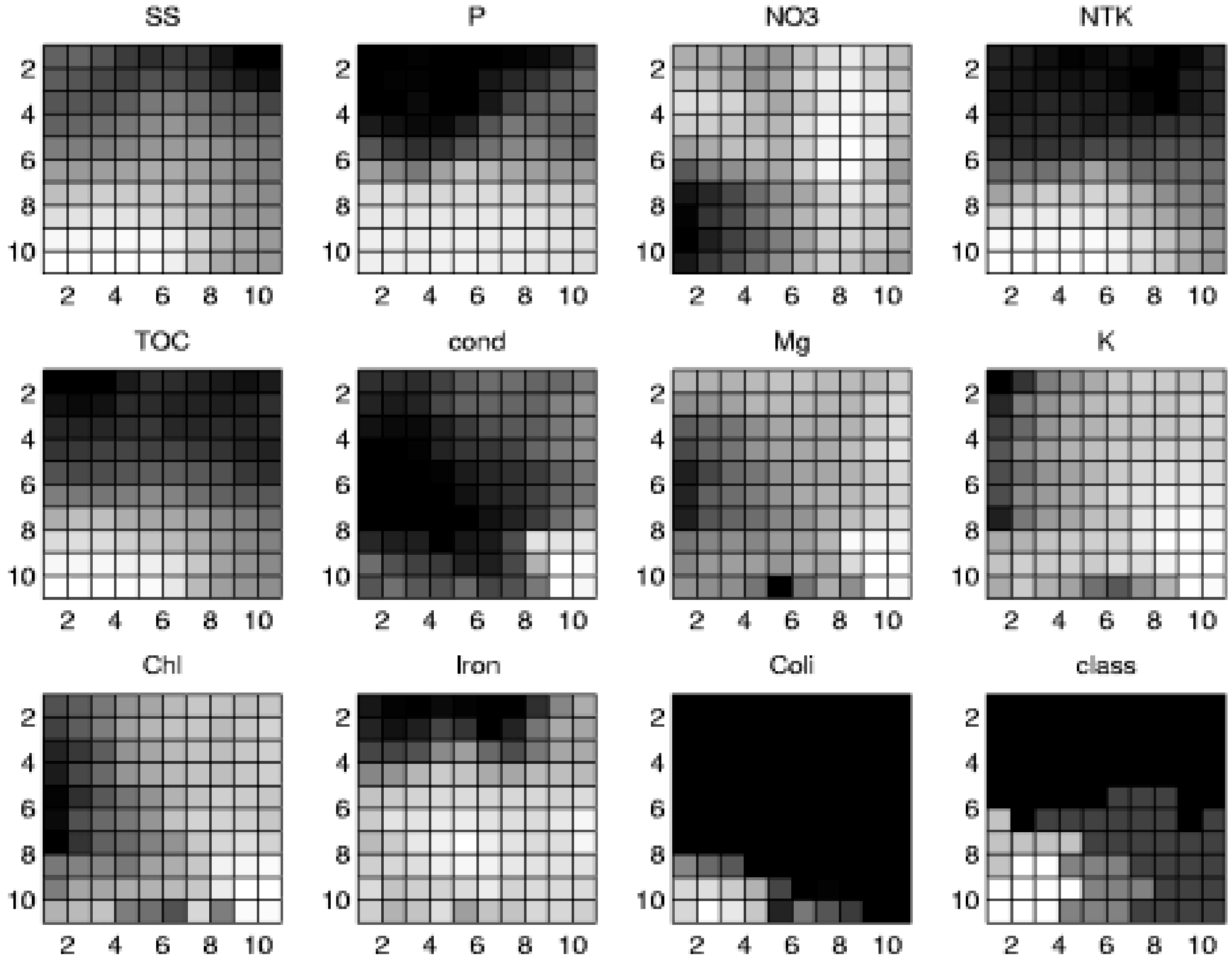
- Goal : map high-dimensional data to a 2-D grid of neurons, in such a way that similar input vectors are mapped to neighboring nodes.

- This « topology preservation » property is obtained by a simple learning algorithm.

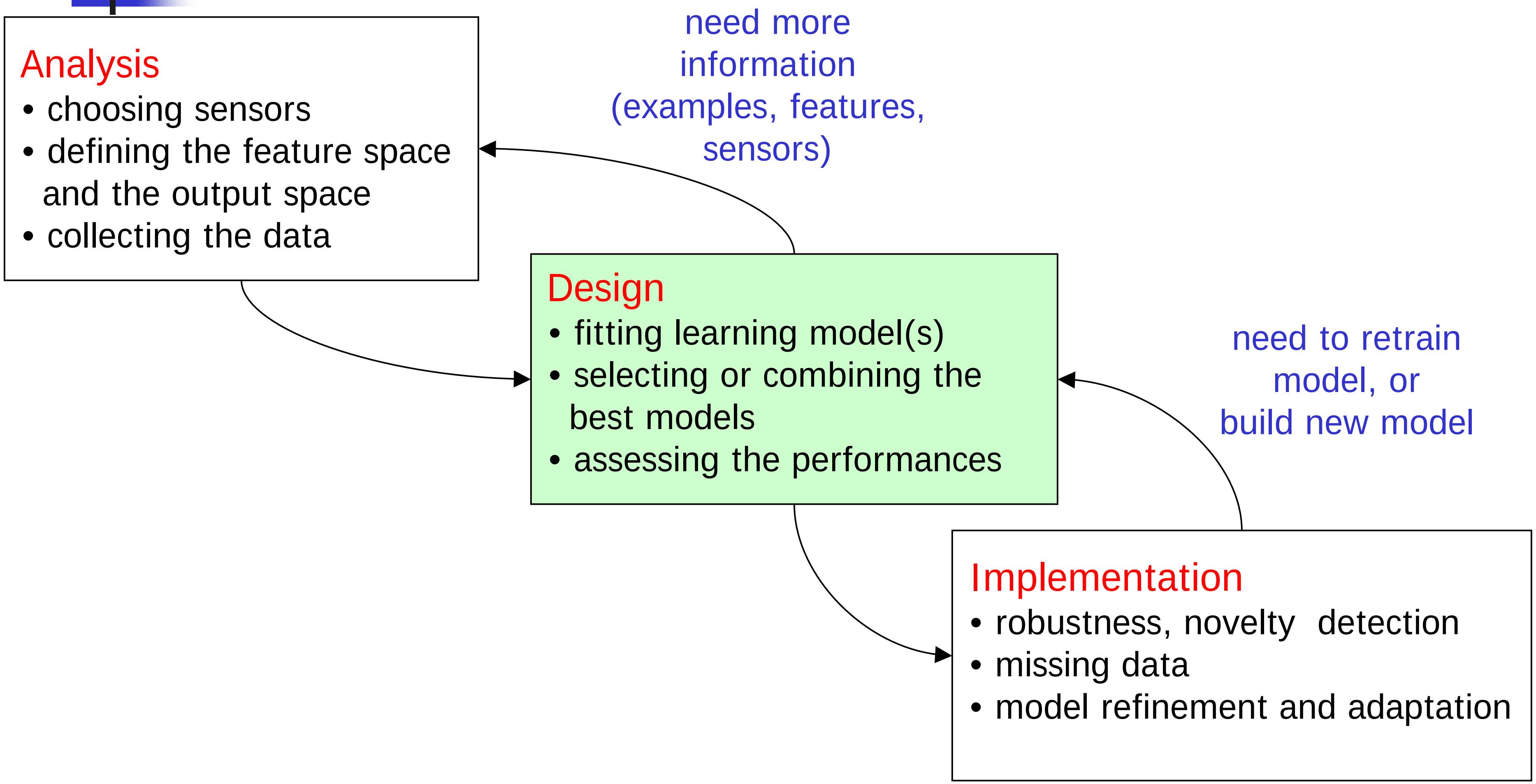
Learning algorithm



Correlation Analysis Using SOM's



Design



Analysis

- choosing sensors
- defining the feature space and the output space
- collecting the data

need more information (examples, features, sensors)

Design

- fitting learning model(s)
- selecting or combining the best models
- assessing the performances

need to retrain model, or build new model

Implementation

- robustness, novelty detection
- missing data
- model refinement and adaptation

Design - Statistical decision theory

- Let X be a random vector, Y real-valued random variable, joint distr. Pr(X,Y).
- We seek a function f(X) for predicting Y given X.
- We need to quantify errors using a loss function L(Y,f(X)).
  - Regression: $L(Y,f(X))=(Y-f(X))^2$
  - Classification: $L(Y,f(X))=1$ if $f(X)\neq Y$, 0 otherwise.
- The optimal f should maximize the *expected prediction error:*

$$EPE(f) = E(L(Y,f(X)))$$

Stat. decision theory (cont.)

- The optimal solution:
  - regression: the regression function

$$f(x)=E(Y \mid X=x)$$

  - classification: the Bayes rule

$f(X)$=class g_k with highest posterior probability $P(g_k|x)$

(the Bayes rule has minimal EPE= error probability)

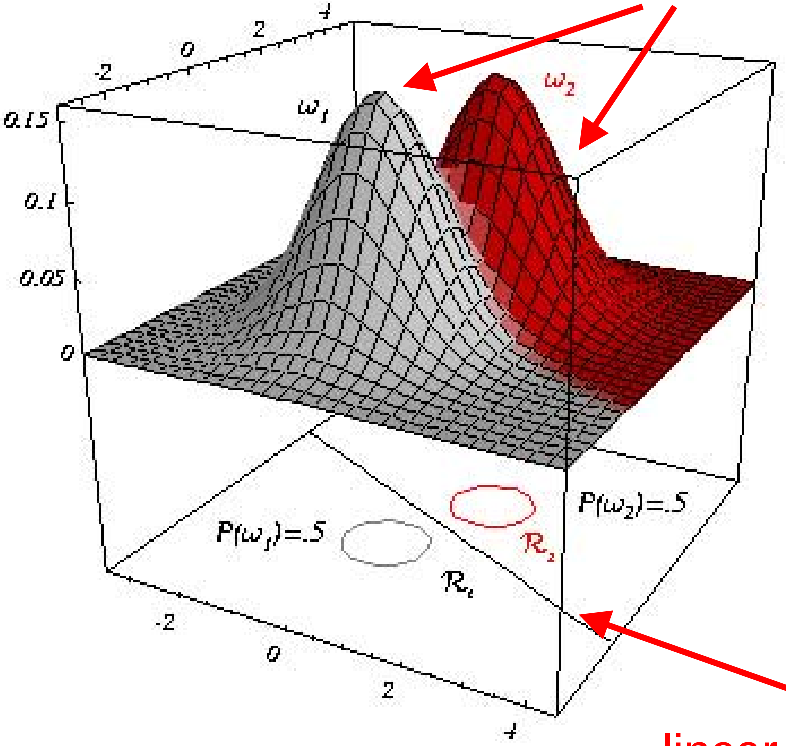
- Most learning methods aim at approximating the regression function or the Bayes rule.

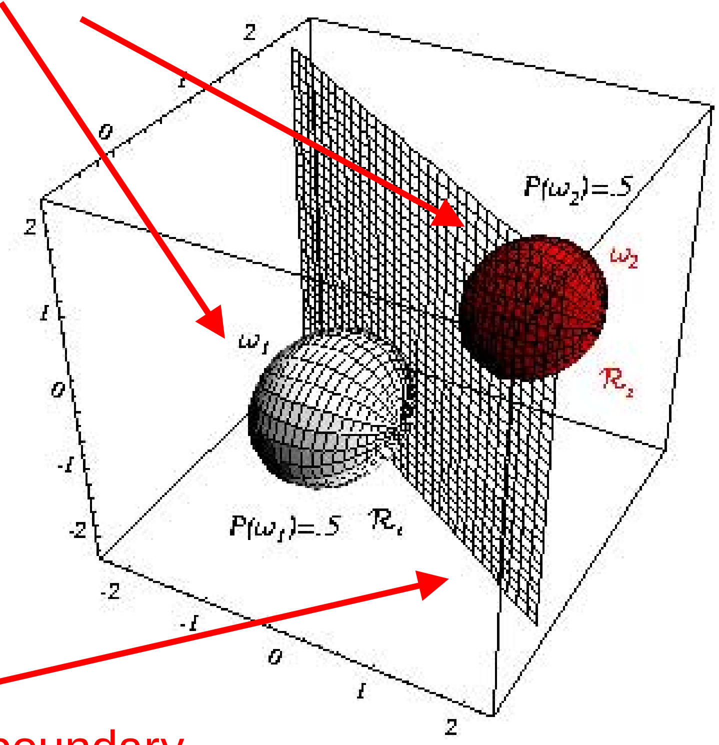
Learning models

- Hundreds of methods for classification and regression.
- Some of the most popular models:
  - linear methods (linear/logistic regression, LDA, ...)
  - non parametric methods (k-NN, Kernel methods)
  - neural network techniques (multilayer perceptrons, LVQ,...)
  - Support vector machines,
  - decision trees,
  - fuzzy systems, ...

Linear discriminant analysis

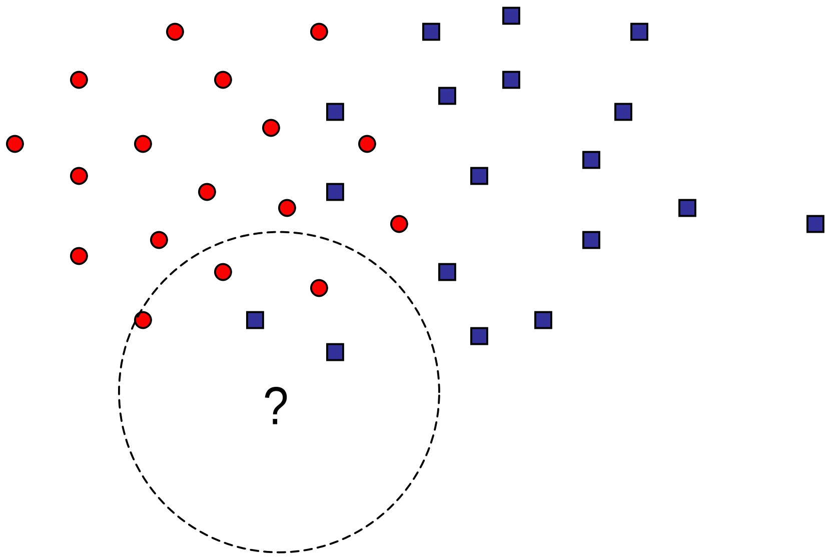
- Gaussian distribution (parametric method)
- equal covariance matrices





linear decision boundary

Nearest neighbor method



- The k-NN method for classification: approximate the Bayes classifier by classifying to the majority class among the k nearest neighbors of x.

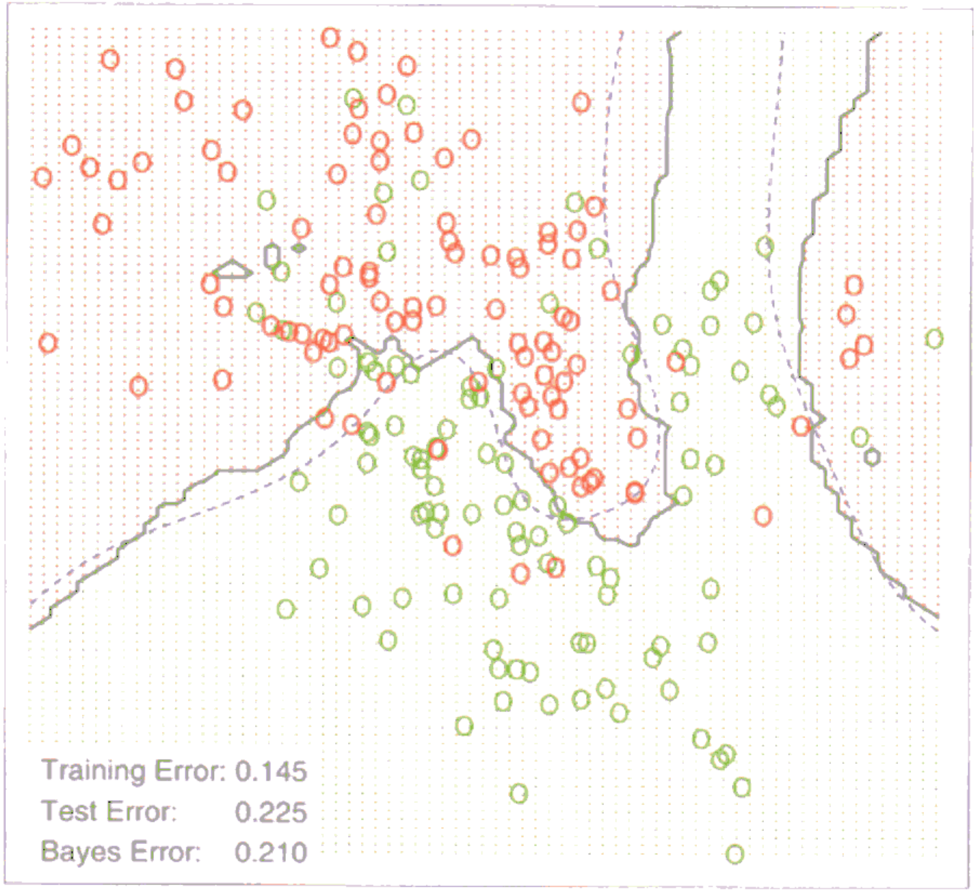
- Similar method for regression

- Non-parametric method: works for any distribution (but high storage and computational requirements)

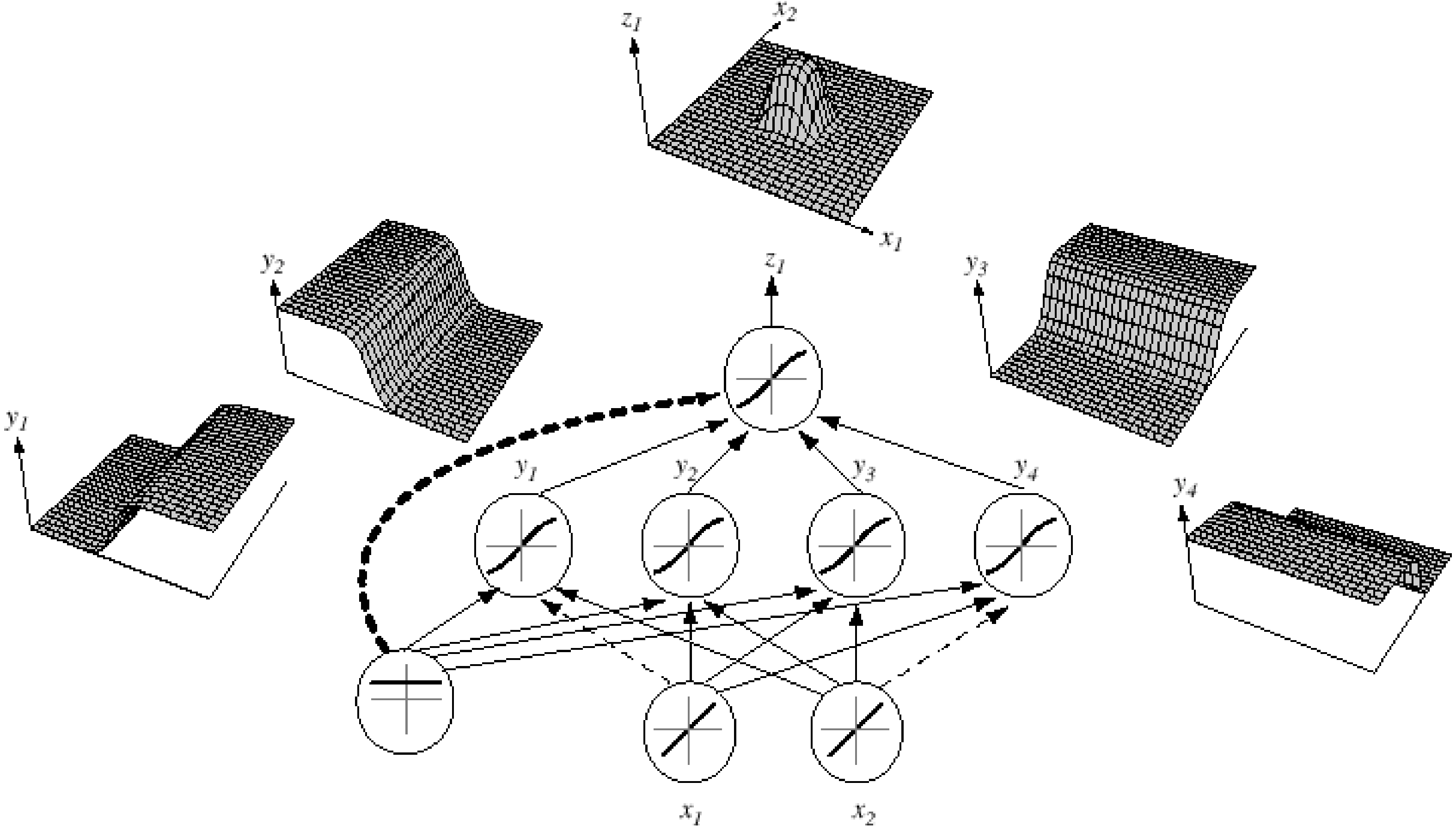
k-NN rule - Example

——— Bayes decision boundary

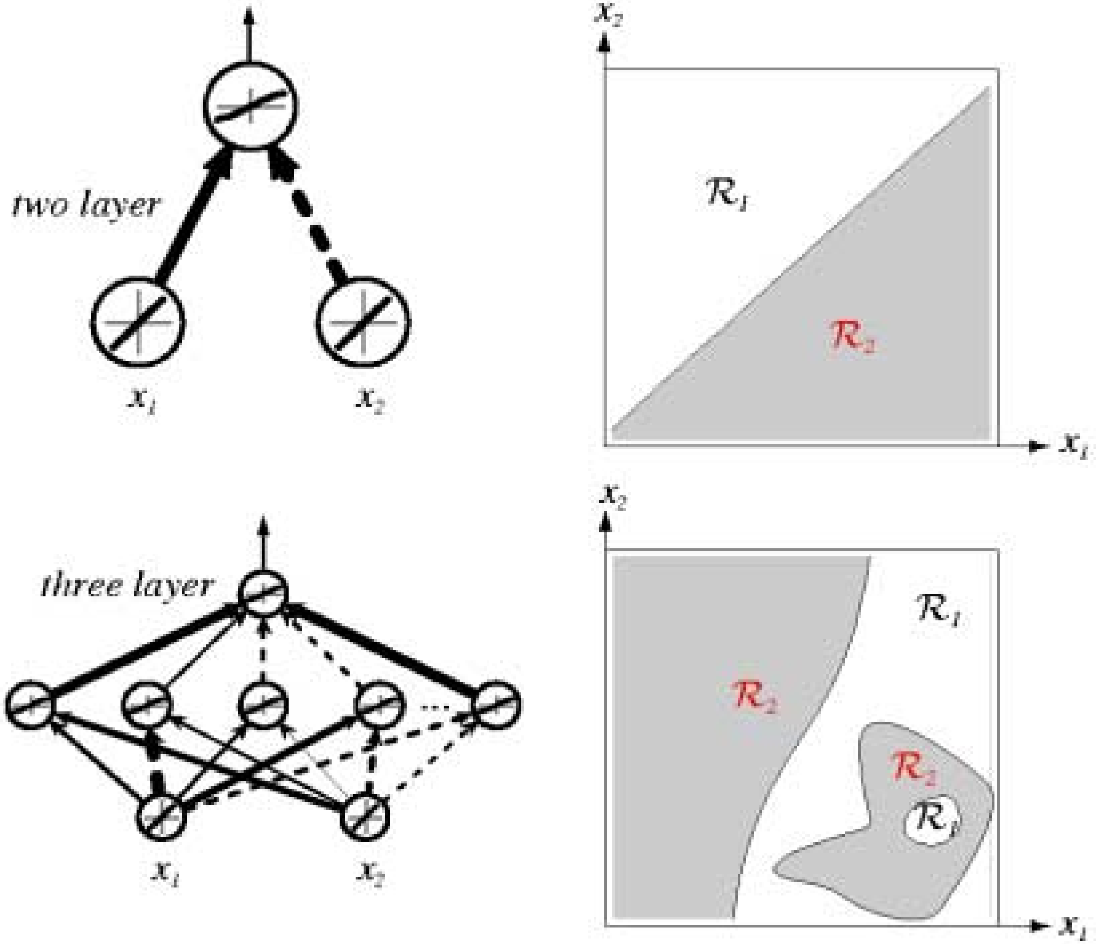
---------- 7-NN decision boundary



Multiplayer perceptrons



PMC (cont.)


two layer
x_1
x_2
three layer
x_1
x_2
$\mathcal{R}_1$
$\mathcal{R}_2$
x_1
x_2
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_2$
$\mathcal{R}_1$
x_1
x_2

Training of MLP's

- Principle: maximize a measure of fit

weight vector

input vector

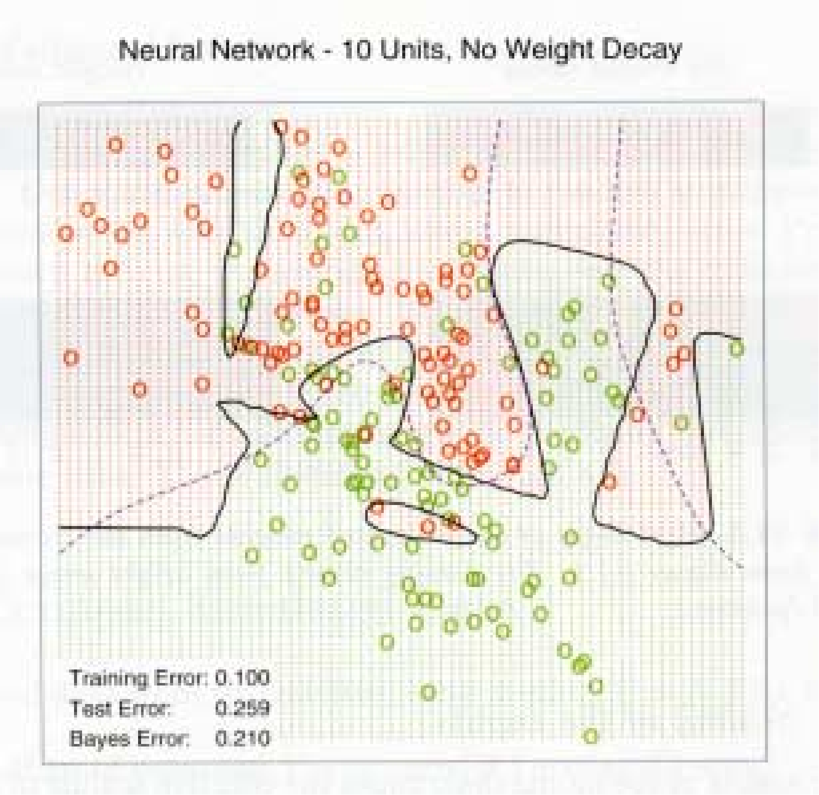
$$R(\mathbf{w}) = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_{ik} - f_k(x_i; \mathbf{w}))^2 \quad \underbrace{\left(+ \lambda \sum_{j=1}^{m} w_j^2 \right)}_{\text{regularization term}}$$

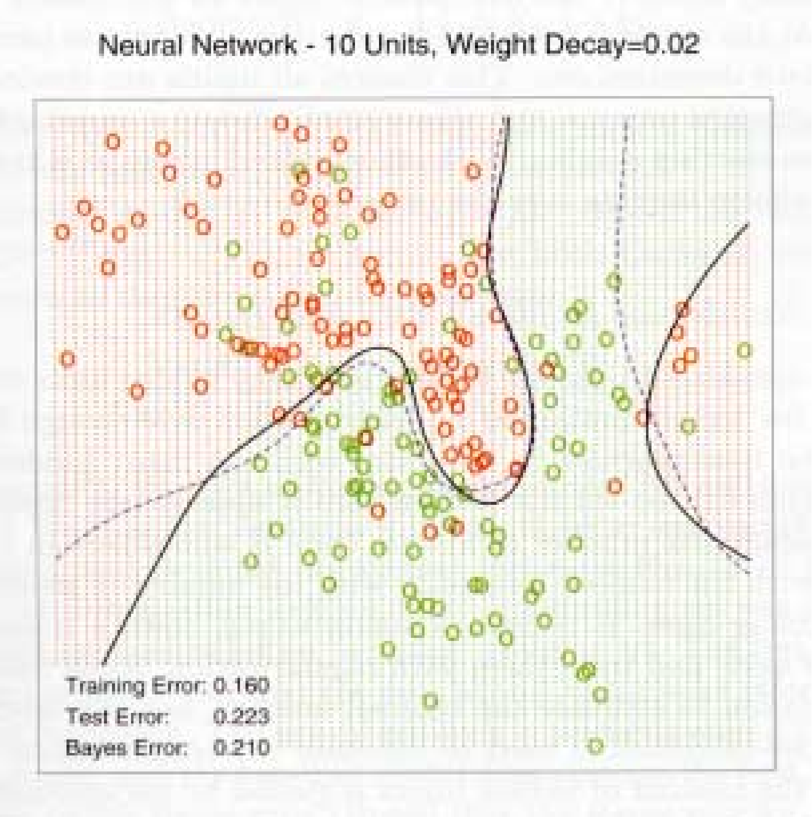
desired output

network output

- R(**w**) is a non linear function of **w** → minimized using an iterative gradient-based non linear optimization algorithm.

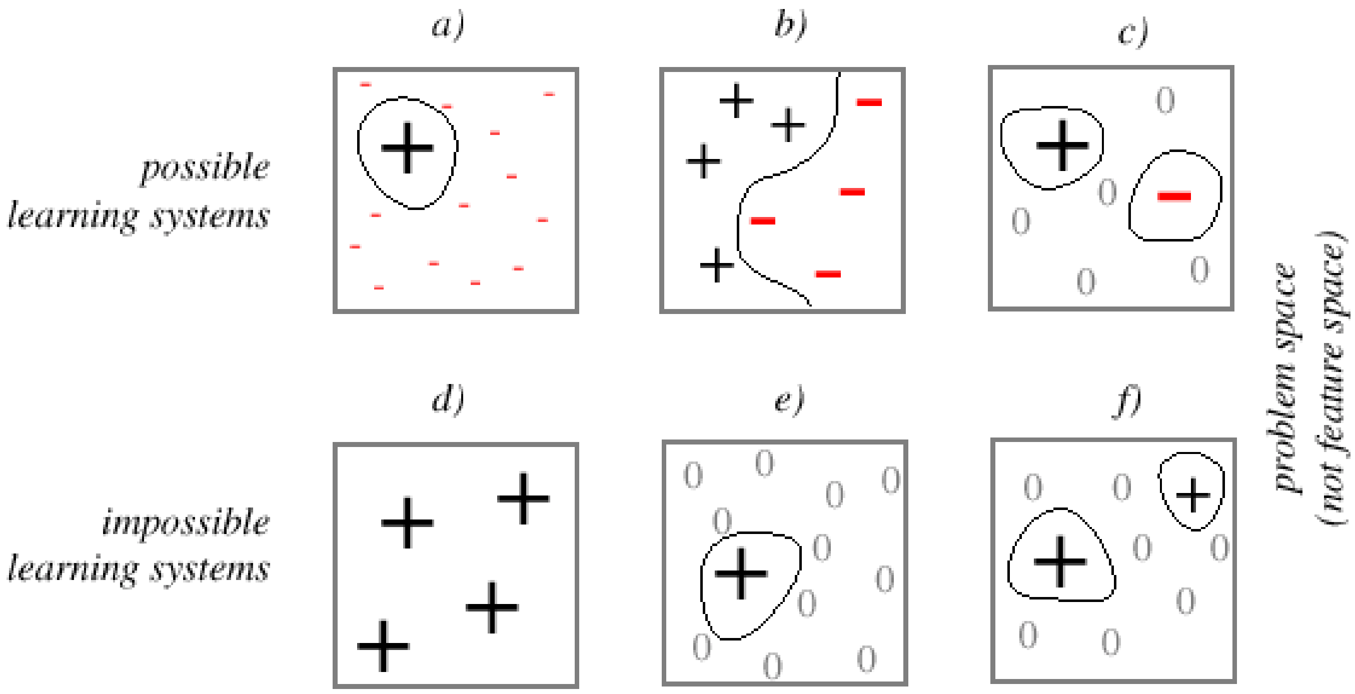
Example


Neural Network - 10 Units, No Weight Decay


Neural Network - 10 Units, Weight Decay=0.02

Is there a "best" learning system ?

No free lunch theorem:
No classifier is better than others for all problems


a)
b)
c)
possible learning systems
+
-
0
d)
e)
f)
impossible learning systems
problem space (not feature space)

Pros and cons of learning algorithms

Classifier	Pros	Cons
LDA	- simple to implement - fast learning and operation	- restrictive assumptions
k-NN rule	- no learning - arbitrary decision boundaries	- high storage and time requirements in operation
MLP	- arbitrary decision boundaries	- slow learning

Tuning learning algorithms

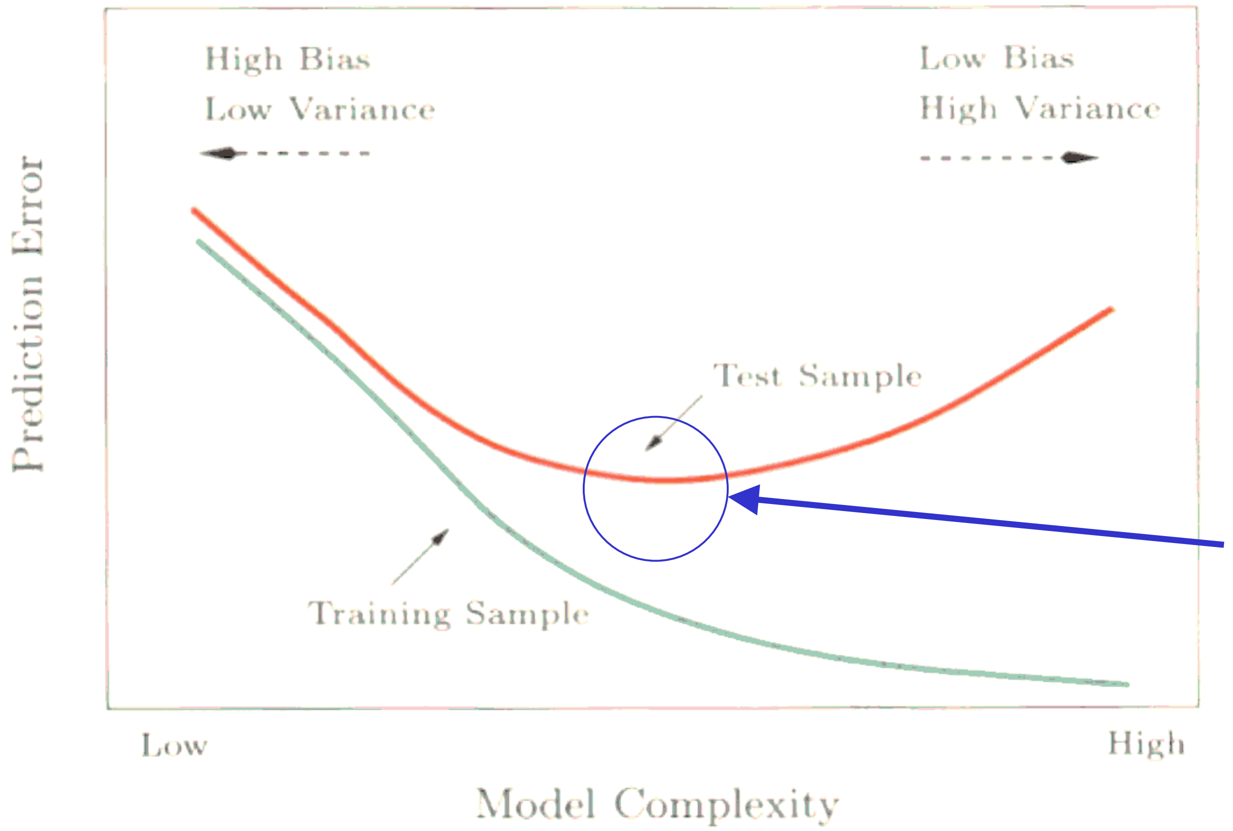
- Each classification/regression method has one or more tuning parameters:

LDA	number of input features d
k-NN rule	d, k
MLP	number of hidden units n_H, λ

- Each tuning parameter controls the complexity of the model: greater complexity results in smaller bias, but greater variance

  → bias/variance dilemma

The bias-variance dilemma



How to determine the optimal model complexity ?

→ model selection

Model selection

- Model selection: given M models, find the one with the smallest expected prediction error

$$EPE = \mathbb{E}(L(Y, \widehat{f}(X)))$$

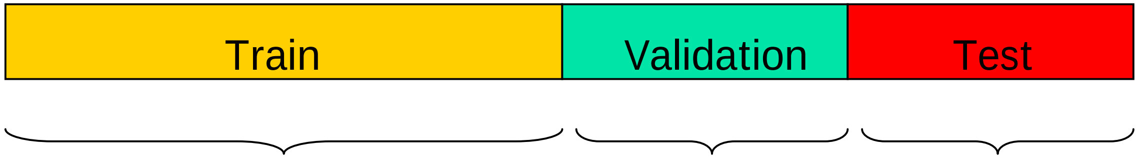
- Problem: we know only the training error

$$\overline{err} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{f}(x_i))$$

which is a strongly biased (optimistic) estimate of EPE.

The hold-out estimation method

The simplest approach, when enough data is available.



Train

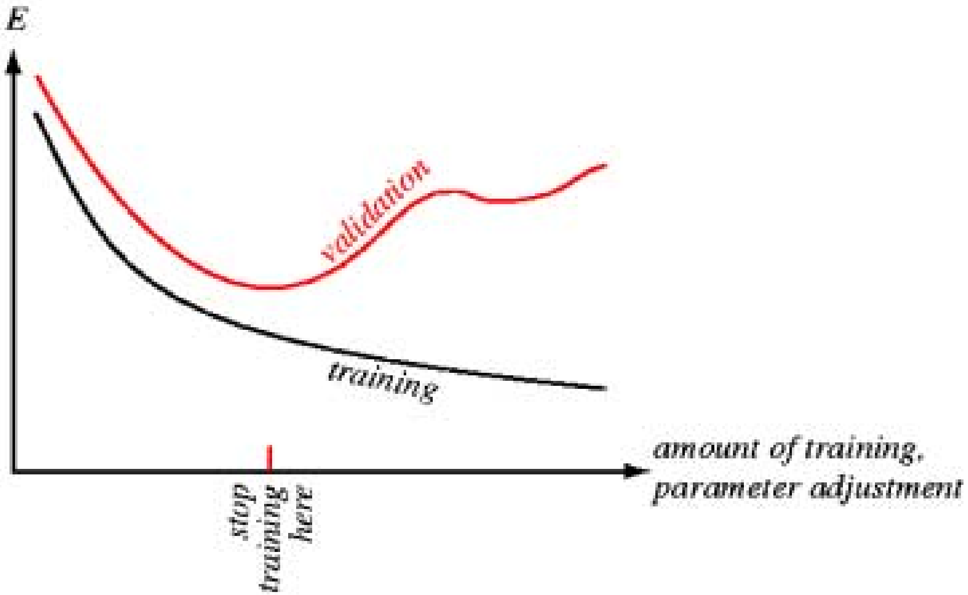
Validation

Test

≈ 50 %
fit the models

≈ 25 %
model selection

≈ 25 %
estimate generalization error of selected model



Cross-validation

Random partition of the data set (typically $5 \leq K \leq 10$):

1	2	3	4	5
Train	Train	Test	Train	Train

- For k=1,...,K
  - fit the model using the data with part k removed
  - test the resulting model on part k
- Combine the K estimates of prediction error

$$CV(\alpha) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

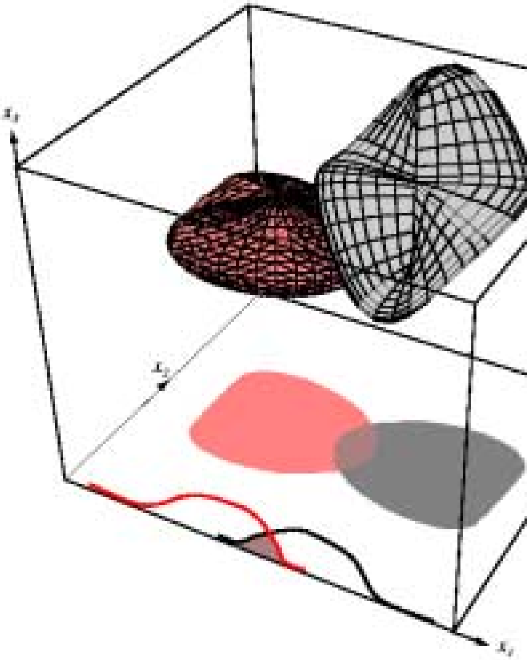
tuning parameter

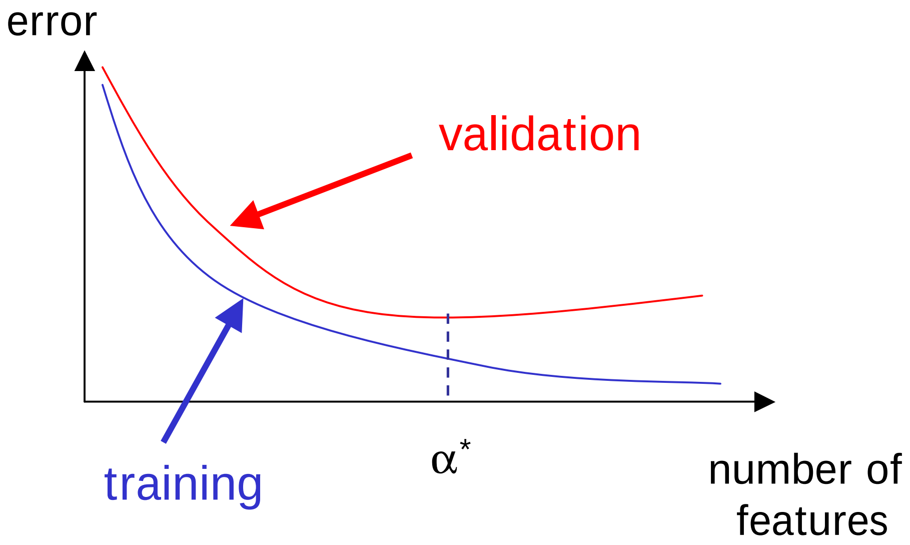
model fit without κ(i)

subset of example i

What to do is the estimated prediction error is too high ?

1) Add new features

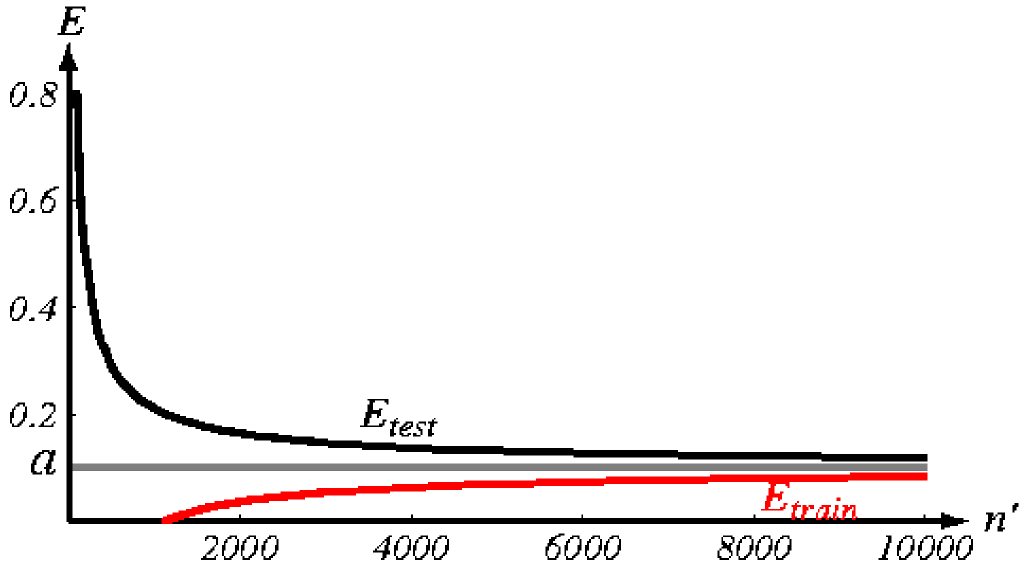




Additional features may or may not reduce the error
(too many features may be harmful !)

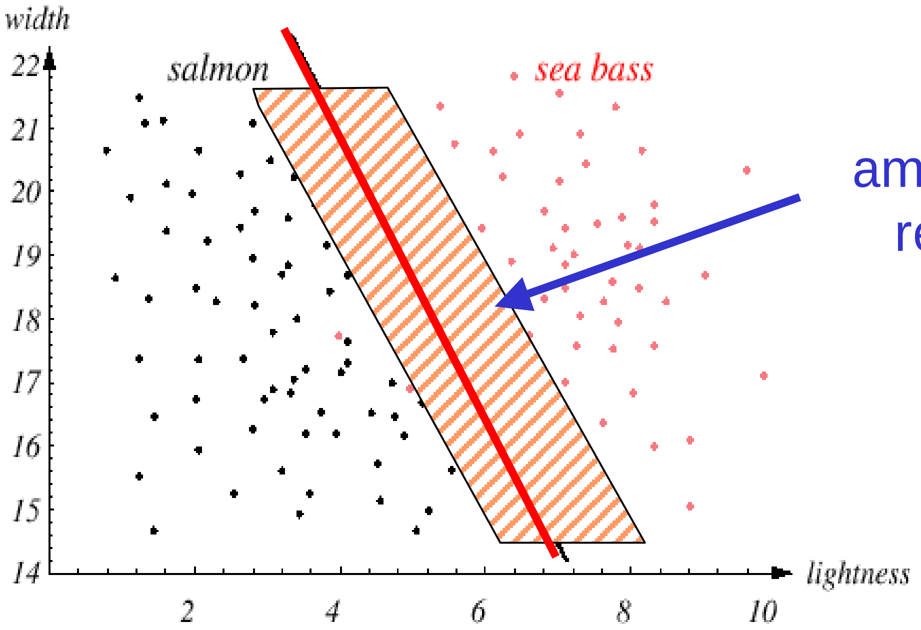
What do do is the estimated prediction error is too high ? (cont.)

2. Add new examples: this can only reduce the error, but may be costly. How many ? → the number of examples allowing to achieve a given error can be predicted by extrapolating the learning curves.



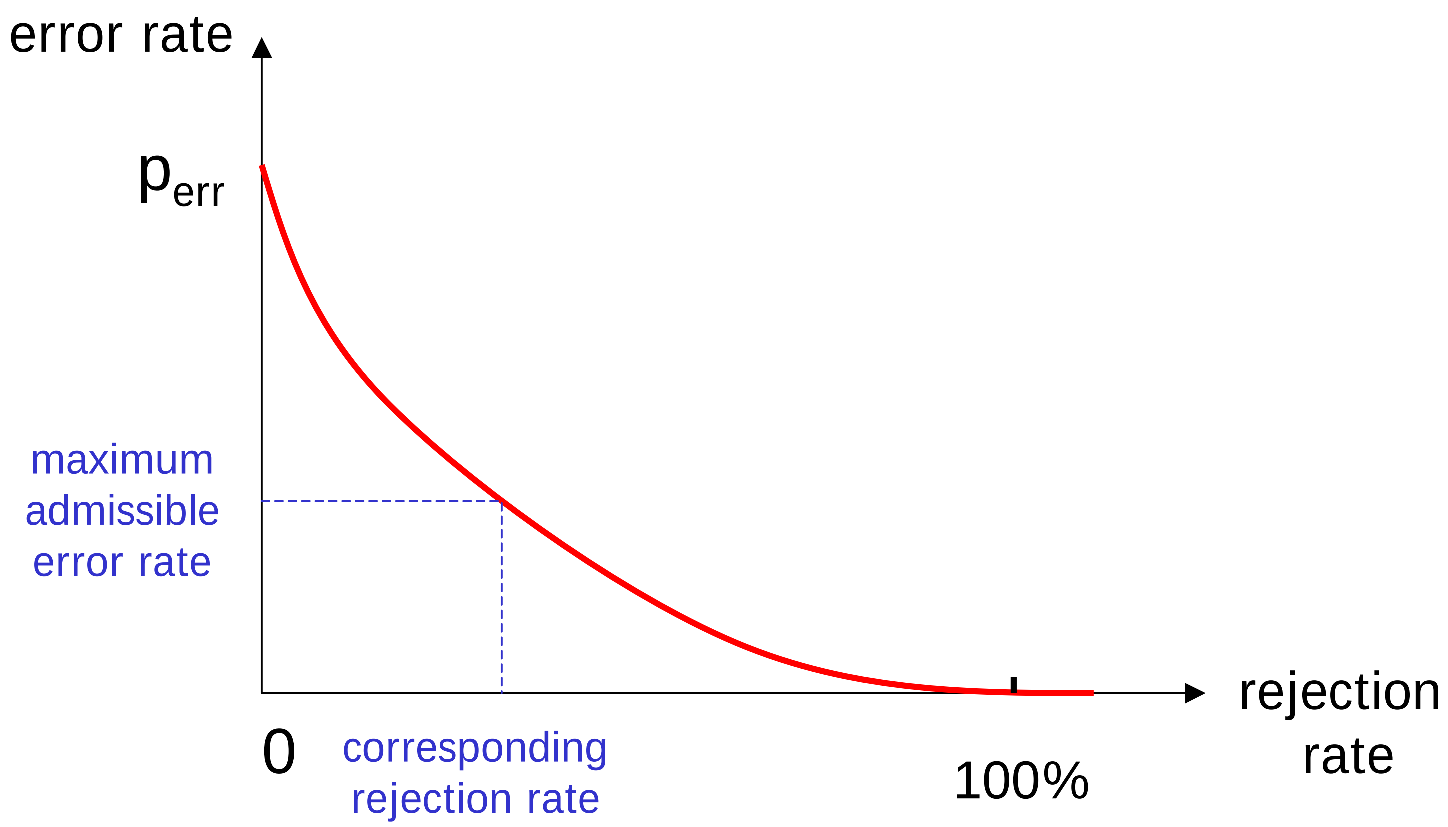
What do do is the estimated prediction error is too high ? (cont.)

3. Reject ambiguous patterns (classification)

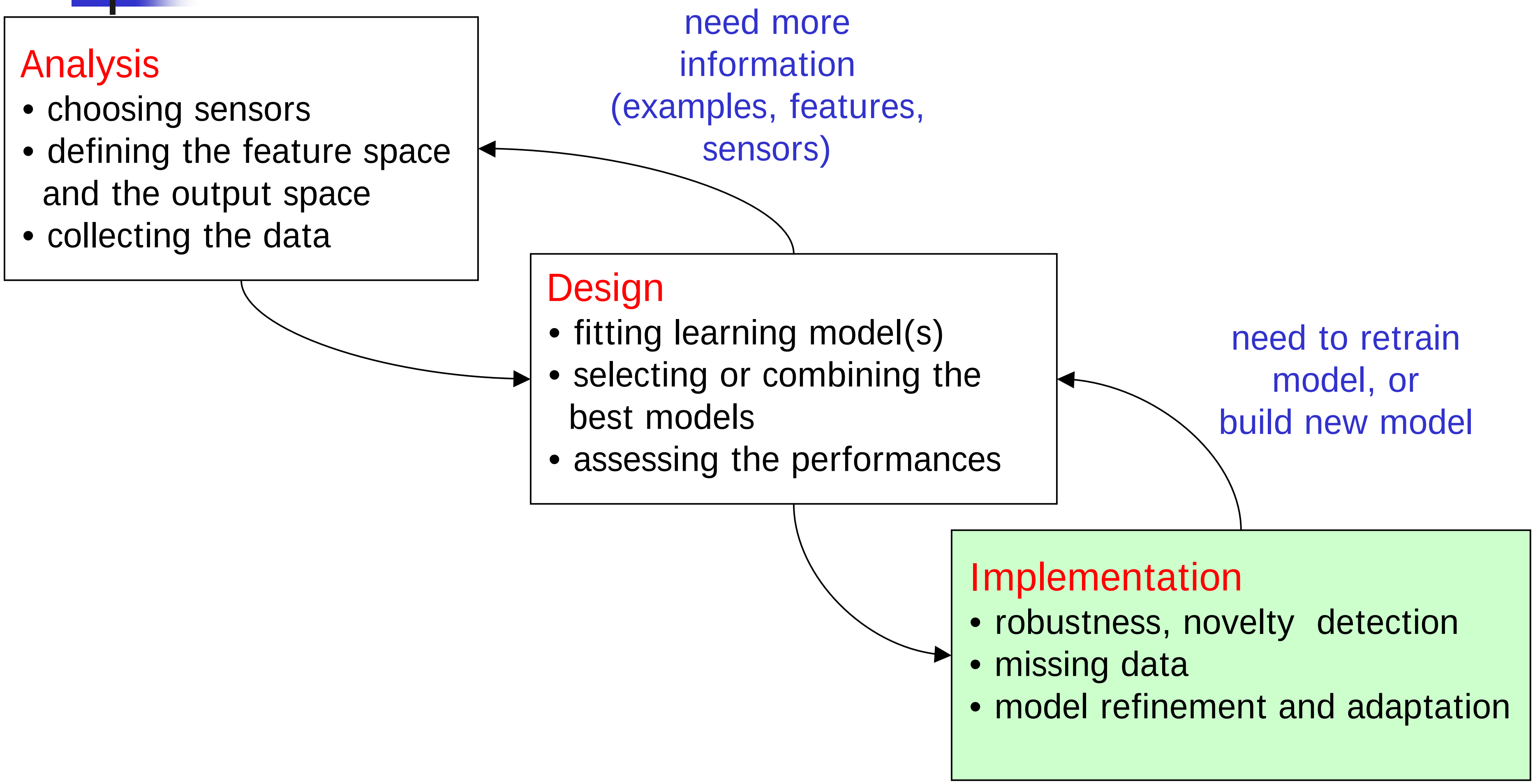


ambiguity region

Rejection/error tradeoff



Implementation



Analysis

- choosing sensors
- defining the feature space and the output space
- collecting the data

need more information (examples, features, sensors)

Design

- fitting learning model(s)
- selecting or combining the best models
- assessing the performances

need to retrain model, or build new model

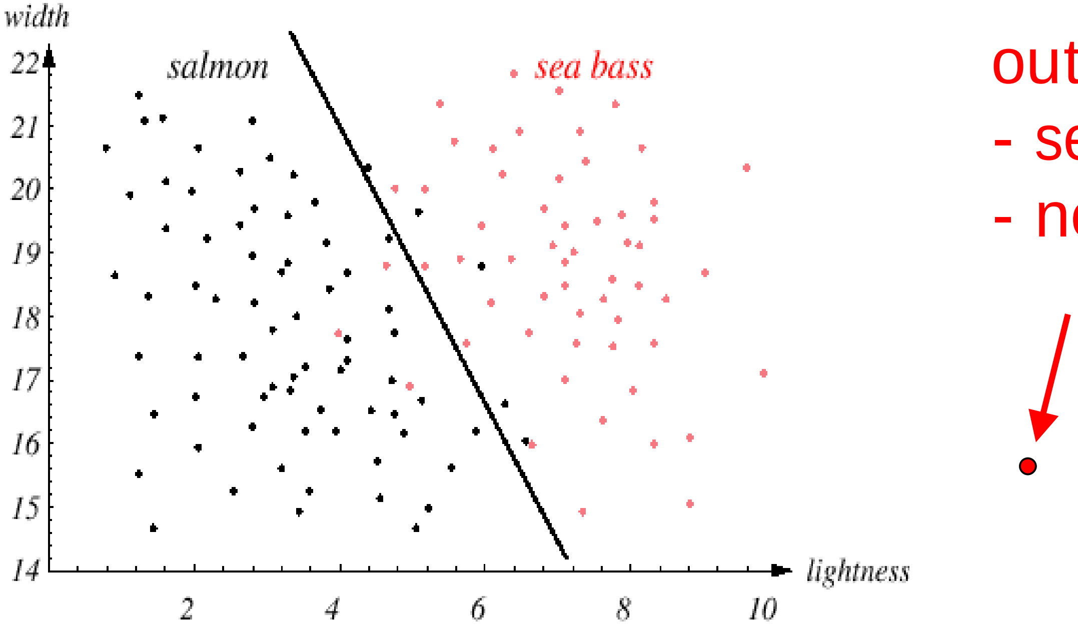
Implementation

- robustness, novelty detection
- missing data
- model refinement and adaptation

Outlier/novelty detection

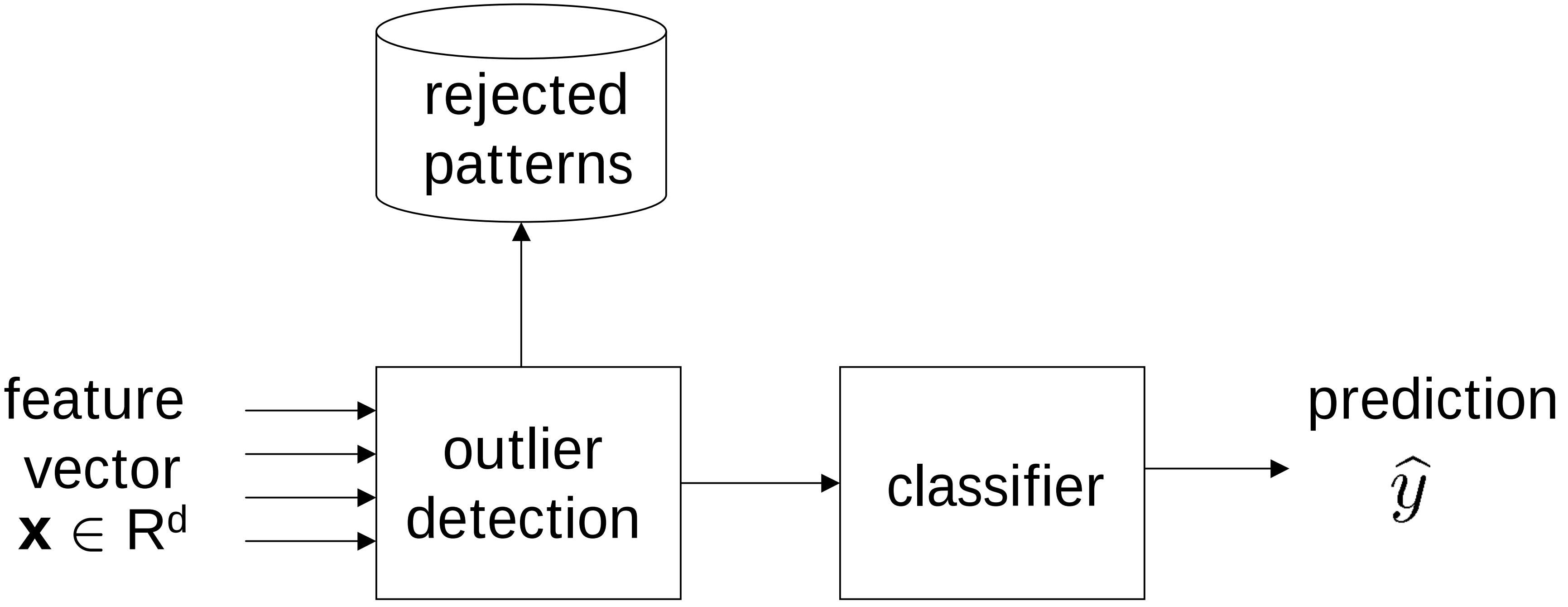
- Learning framework: (X,Y) have joint probability distribution Pr(X,Y).
- The training set composed of validated, quality-controlled data → realization of a random sample from Pr(X,Y).
- In operational conditions, the distribution of (X,Y) may change due to:
  - different operating conditions
  - sensor faults
  - occurrence of new, previously unseen system states
- The output from the learning system may become unreliable, unless some outlier and novelty detection mechanism is implemented

Outlier: example

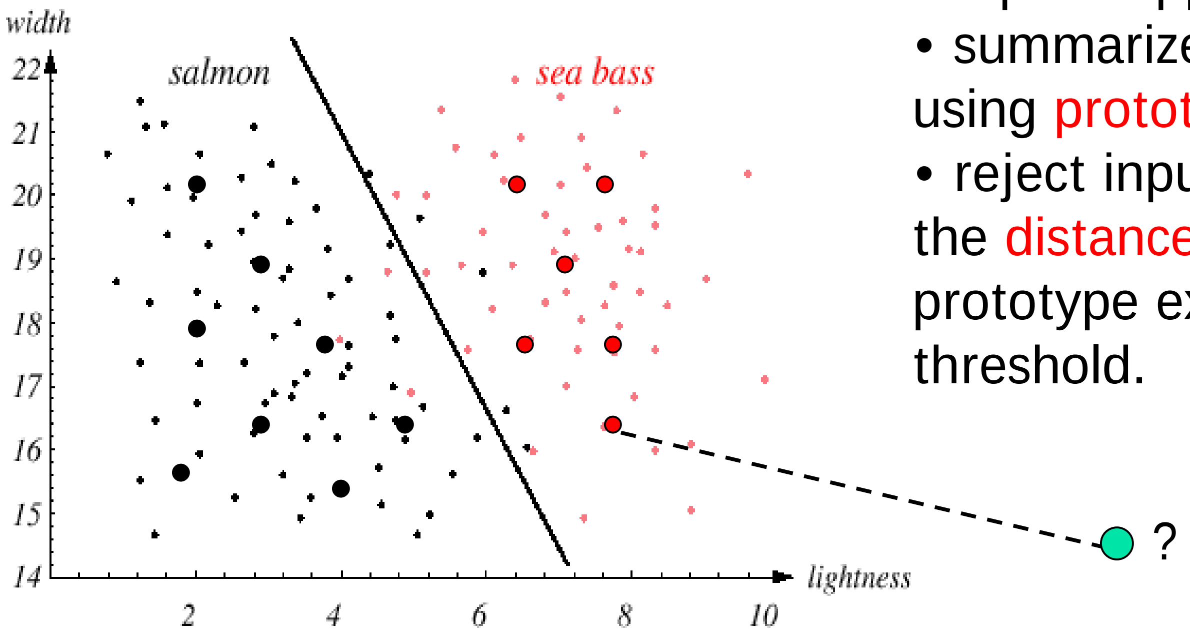


outlier:
- sensor fault ?
- new class ?

A robust pattern recognition system



Distance rejection

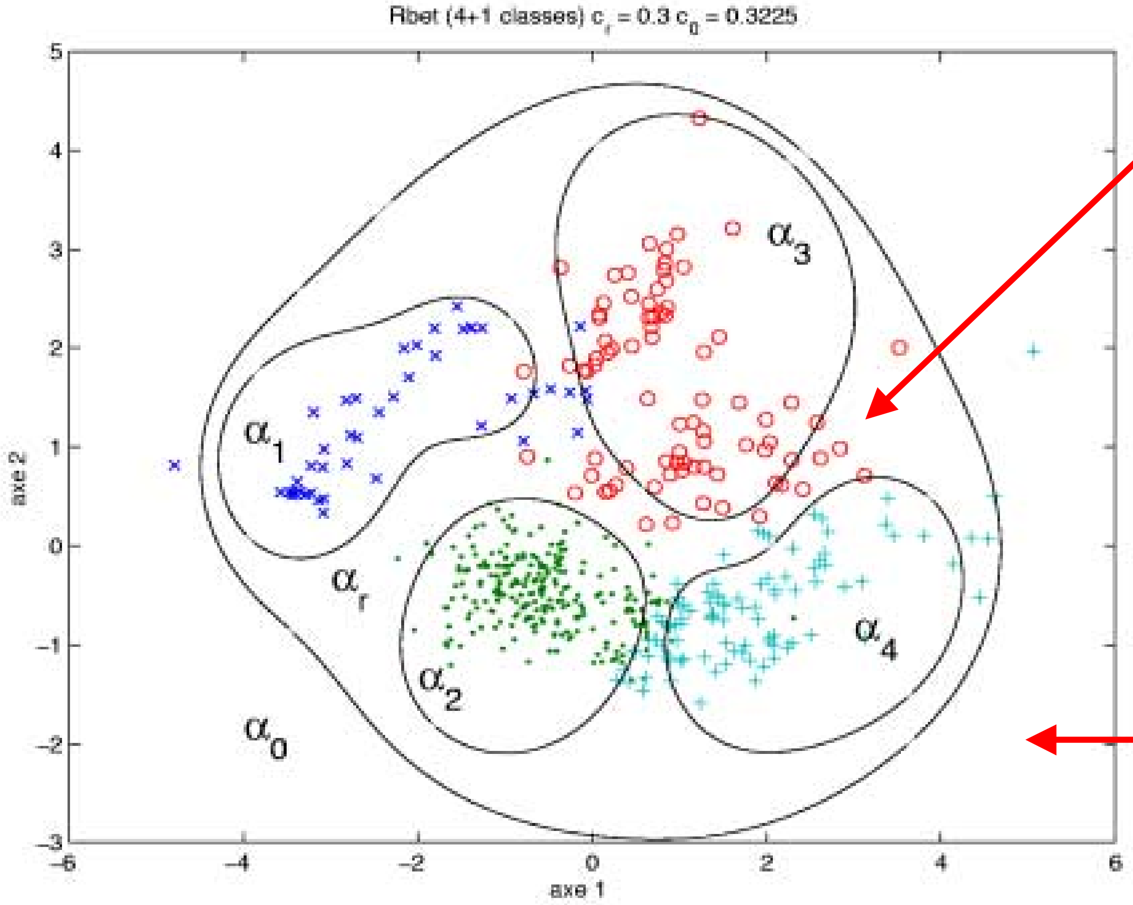


Simplest approach:

- summarize the learning set using prototypes
- reject input patterns for which the distance to the closest prototype exceeds a given threshold.

- Determination of the prototypes = clustering problem
- Algorithms: SOM, c-means

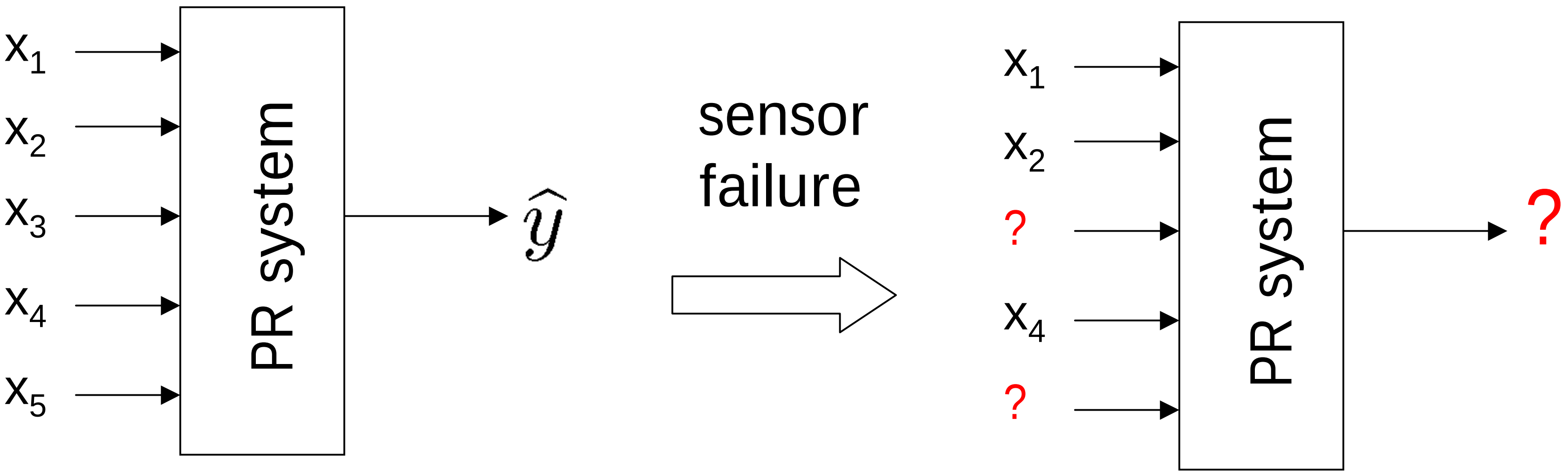
Example



ambiguity rejection

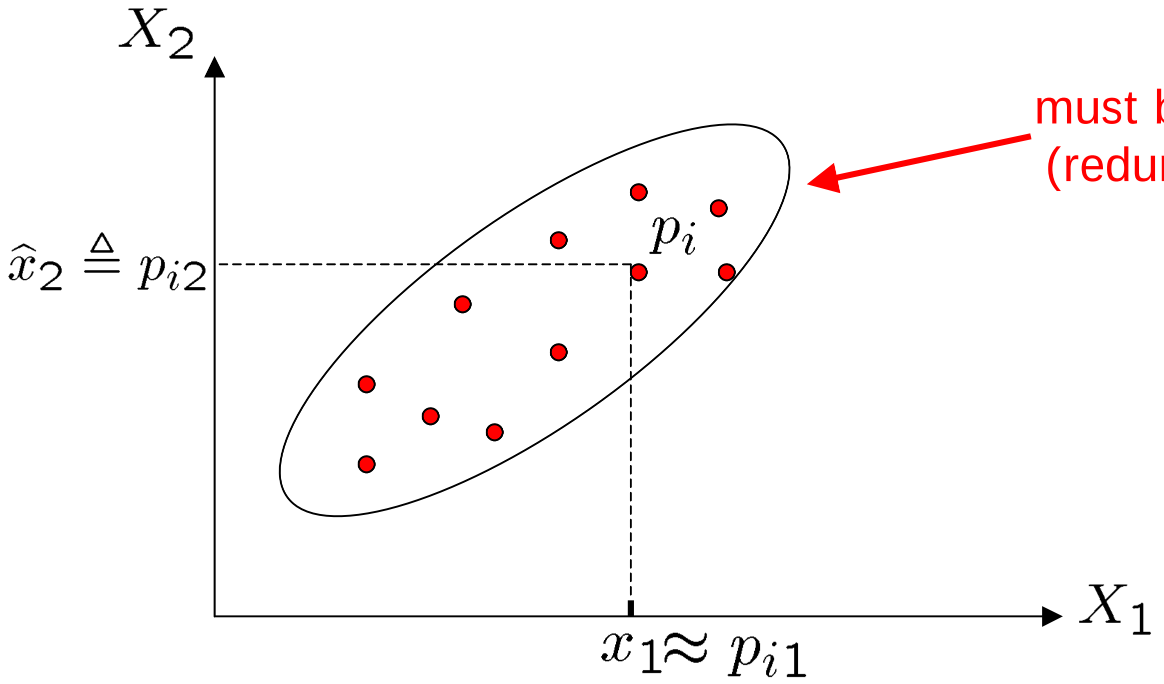
distance rejection

Missing data



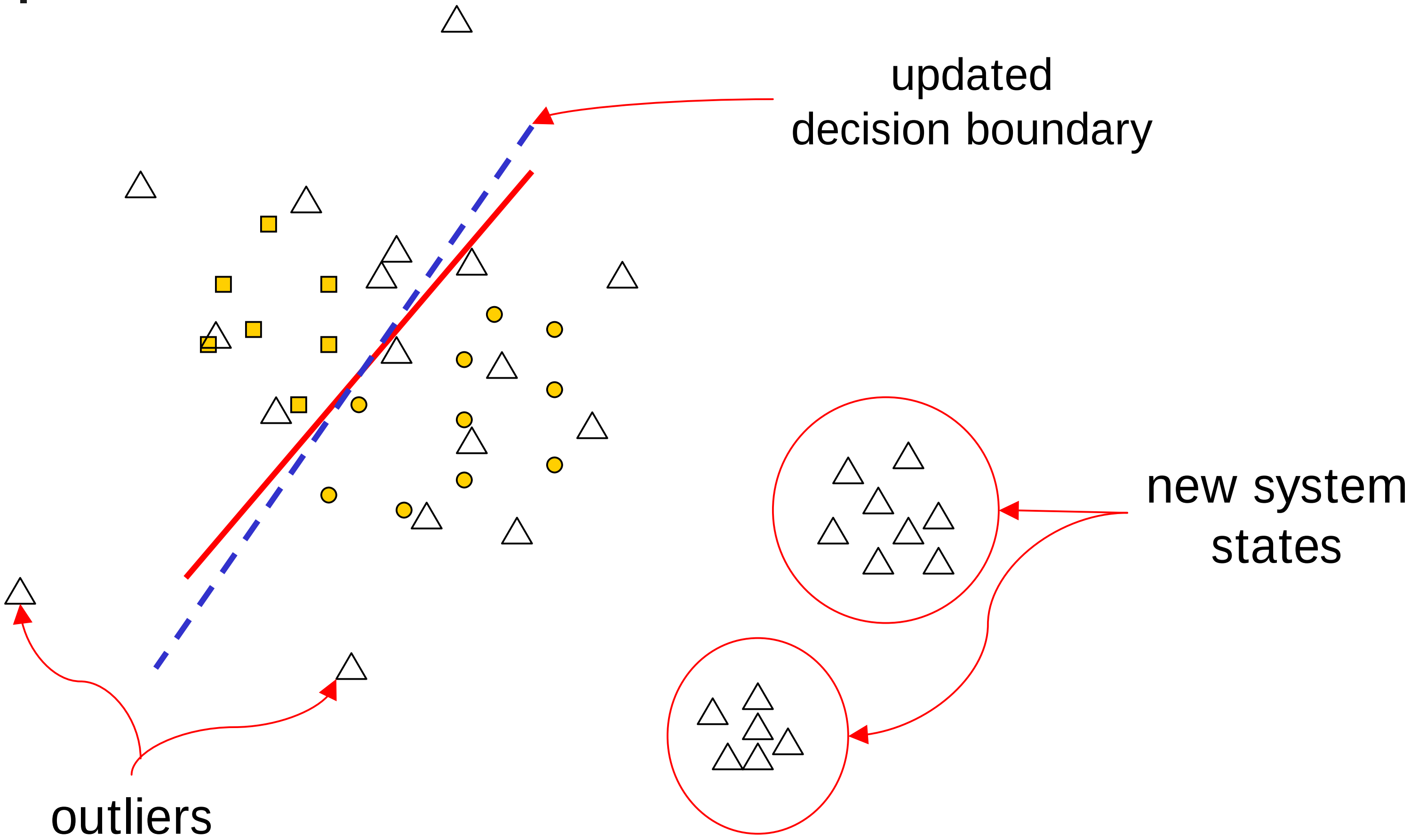
Missing data reconstruction

- Let I = indices of missing features
  J = indices of available features
- Approach: estimate $E(X_I|X_J=x_J)$



features
must be correlated !
(redundancy helps)

Adaptation of learning systems



updated
decision boundary

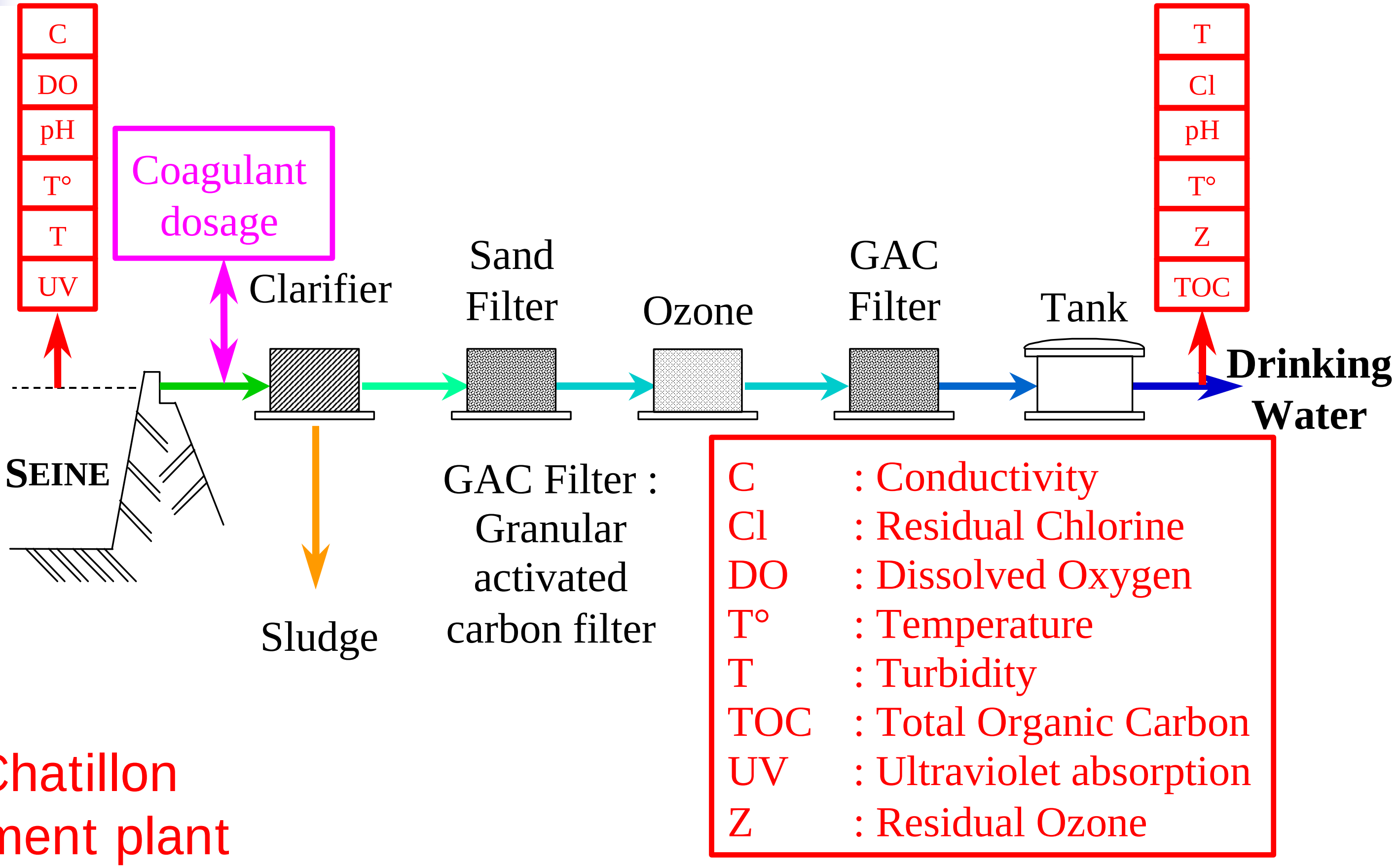
new system
states

outliers

Adaptation of learning systems (cont.)

- Continuous adaptation of a learning system requires:
  - outlier/novelty detection mechanisms
  - unsupervised algorithms for discovering new classes
  - a posteriori knowledge of class labels for predictive accuracy improvement
- Can be done on-line but difficult
  - stability/plasticity dilemma
  - many tunable parameters
- More safely done off-line, using human supervision
  - some expertise in data analysis is necessary
  - increases maintenance costs

Case study: prediction of optimal coagulant dosage in WTP



GAC Filter : Granular activated carbon filter

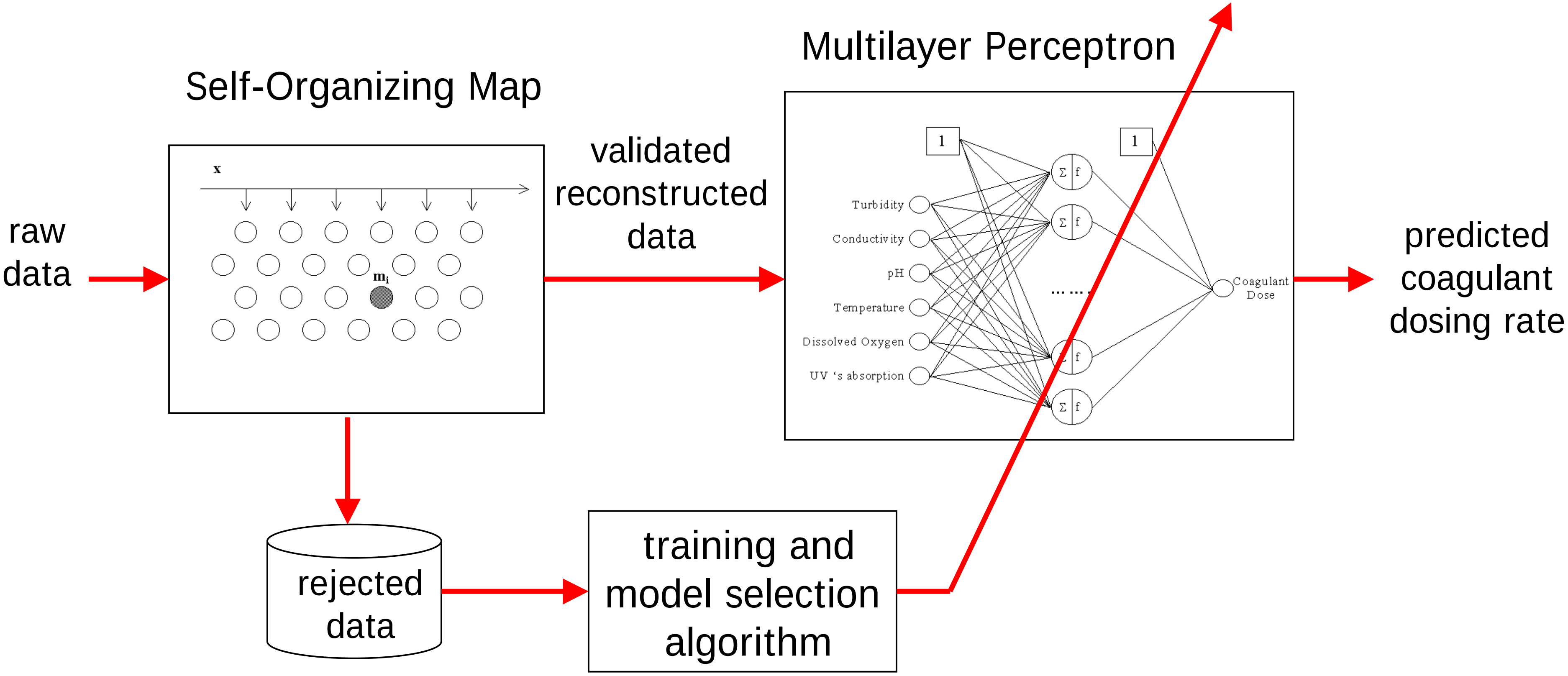
C	: Conductivity
Cl	: Residual Chlorine
DO	: Dissolved Oxygen
T°	: Temperature
T	: Turbidity
TOC	: Total Organic Carbon
UV	: Ultraviolet absorption
Z	: Residual Ozone

Viry-Chatillon treatment plant

Requirements

- Determine optimum coagulant dosage using on-line measurements of raw water quality parameters: turbidity, pH, conductivity, temperature,….
- Operation without human supervision:
  - robustness against erroneous data
  - estimation of missing input data
  - detection of and adaptation to changes in water characteristics
- Portability of the system to different sites:
  - methodology for fitting the whole system automatically to new data

The system



Self-Organizing Map

Multilayer Perceptron

raw data

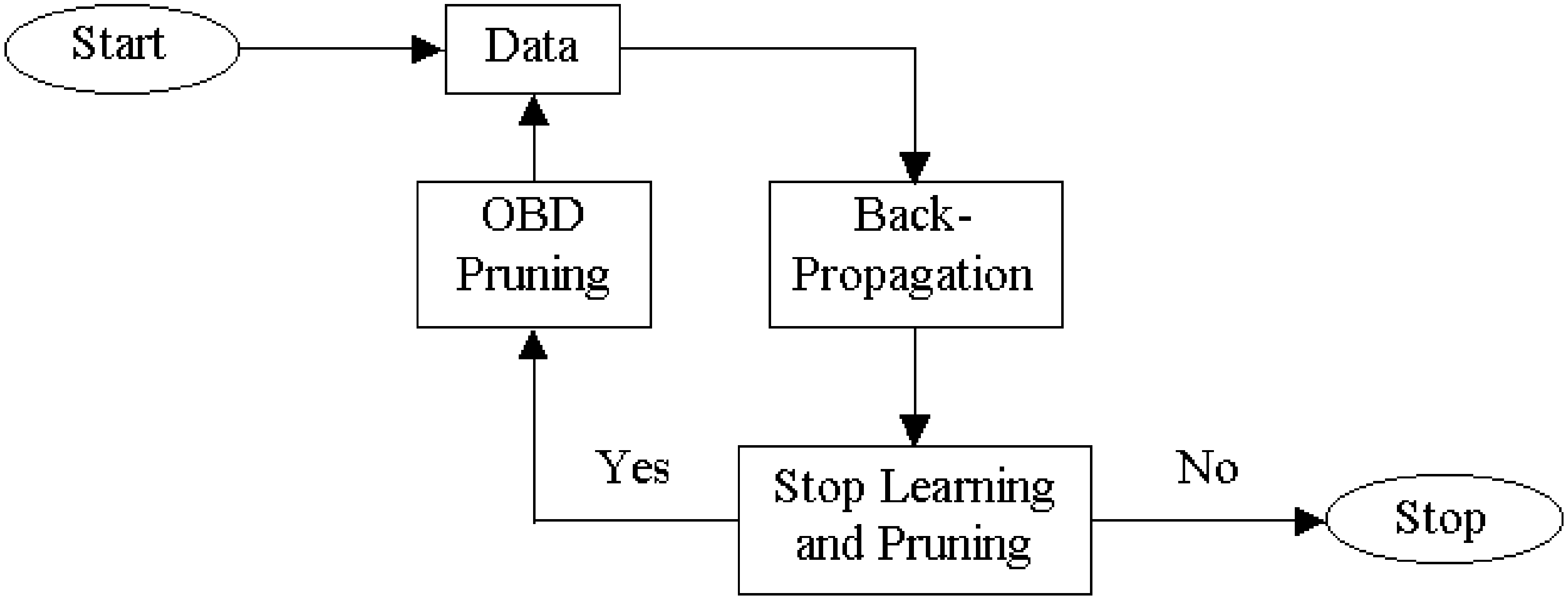
validated reconstructed data

predicted coagulant dosing rate

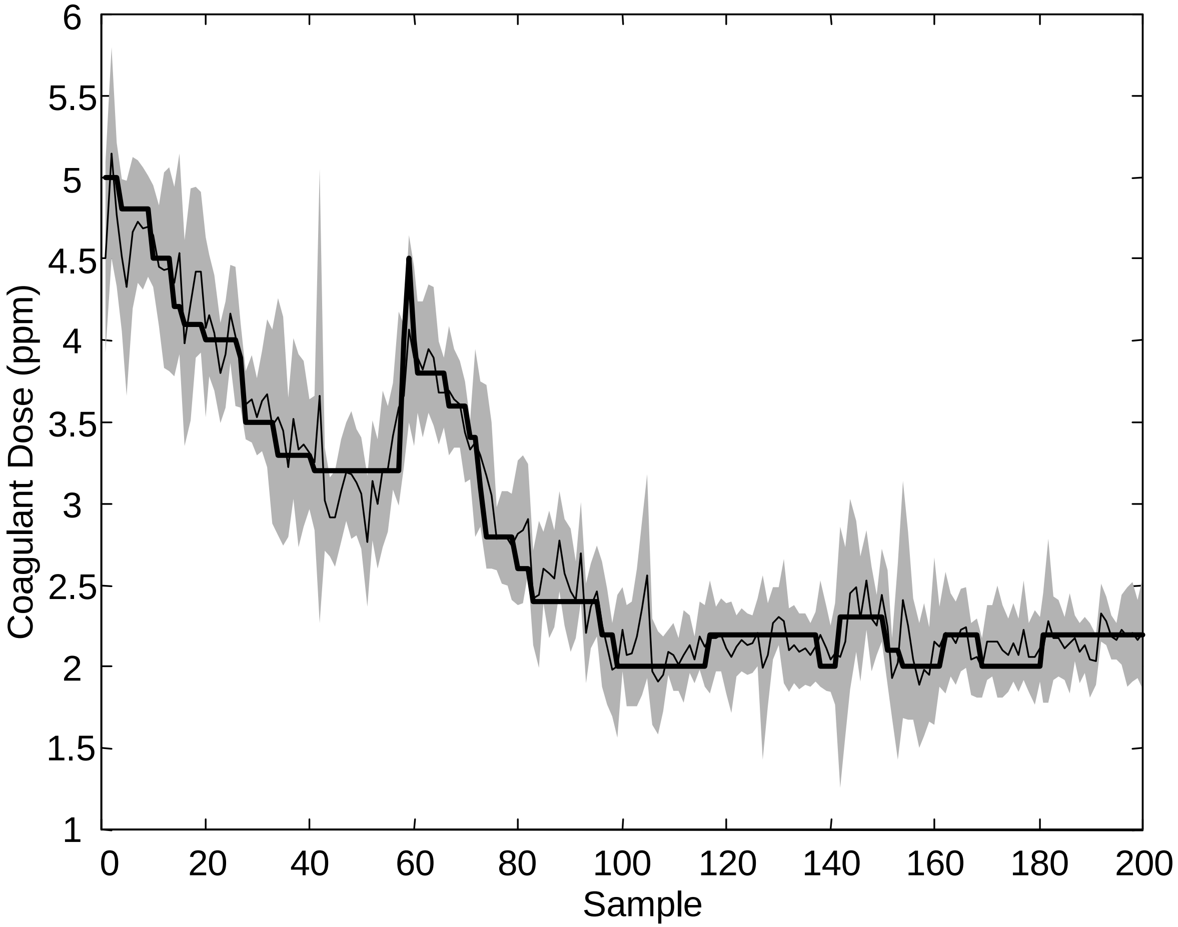
rejected data

training and model selection algorithm

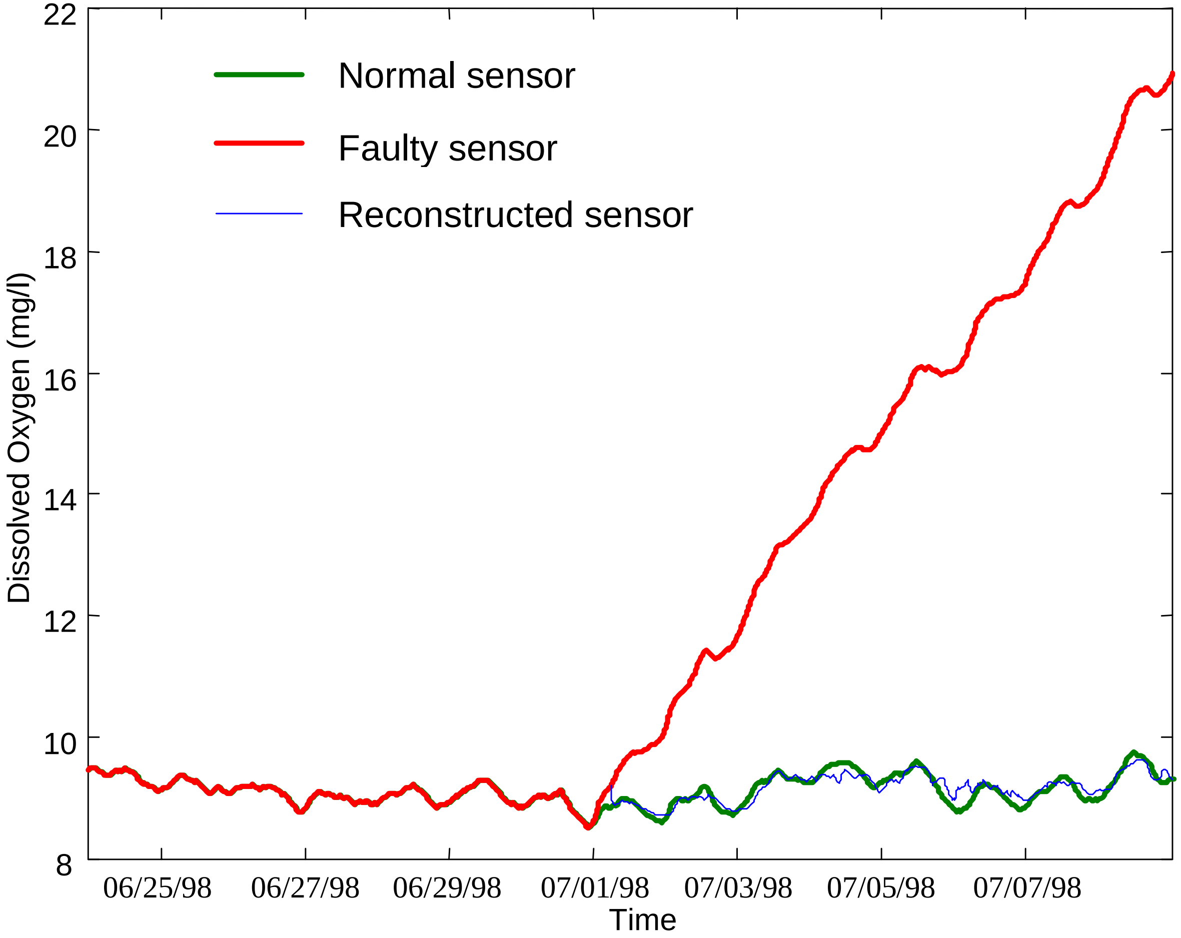
Learning algorithm



Results

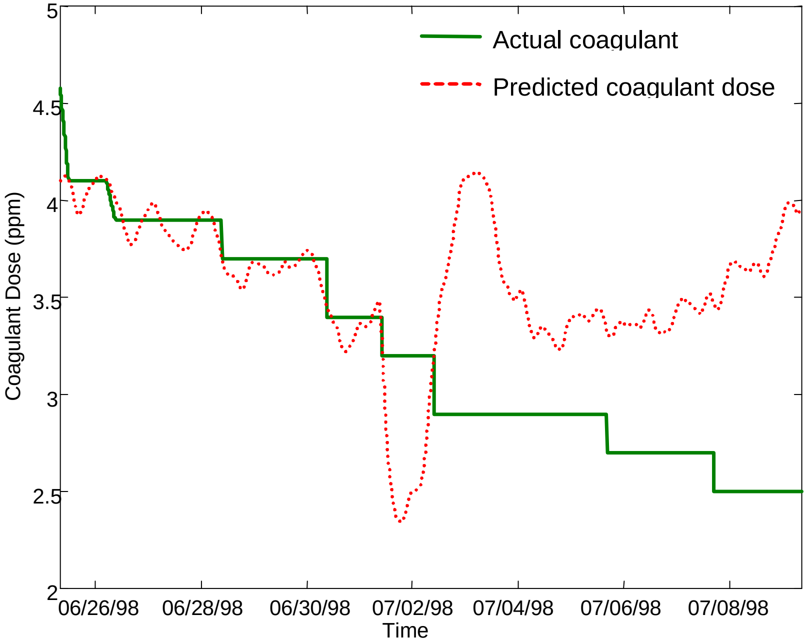

Coagulant Dose (ppm)
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
20
40
60
80
100
120
140
160
180
200
Sample

Simulation of sensor fault

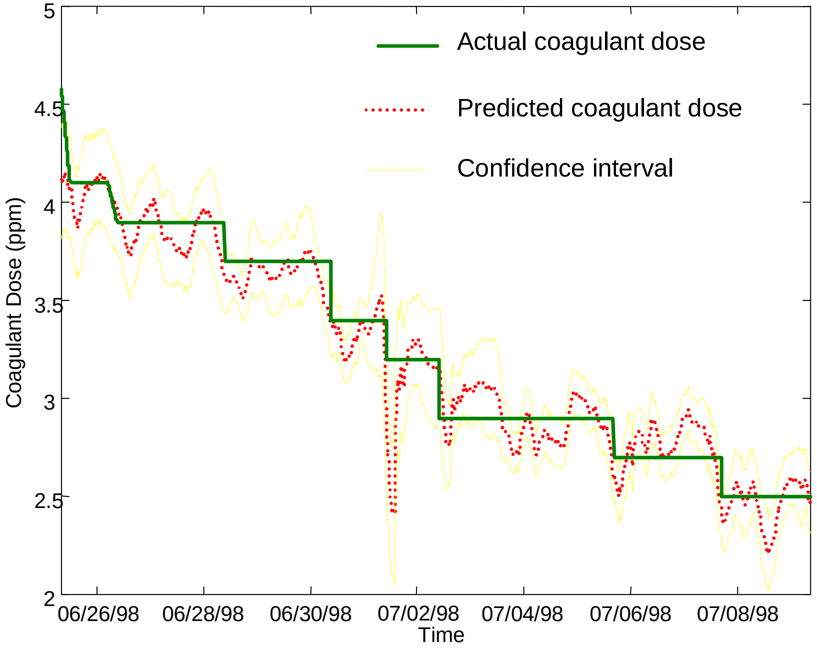


Results

Without preprocessing



With preprocessing



Conclusions

- Pattern recognition (supervised learning) techniques allow to build statistical models of the relationship between input and output variables, using observation data.
- Applications:
  - software sensor design
  - system diagnosis
  - data mining: text/image categorization, credit scoring, financial decision making, ...
- The three phases in the development of a PR system:
  - analysis (choice of sensors, definition of features, data)
  - design (model fitting and selection)
  - implementation (robustness, adaptation)

Conclusions (continued)

- The design of a pattern recognition system requires a close cooperation between
  - domain experts (choice of input and output spaces, selection of a representative learning set), and
  - statisticians (selection of learning techniques, interpretation of results).
  - end-users (knowledge of operational constraints and objectives)
- Two pitfalls:
  - expect too much from statistical techniques when too few data is available
  - expect too much from huge data sets when domain knowledge is weak or the learning task has not been thoroughly specified