Introduction to Artificial Neural Networks

v 2.3 – August 2002

Introduction to Artificial Neural Networks

- Why ANNs ?
- Biological inspiration
- Some examples of problems
- Historical background
- Neural computation
  - feature extraction
  - classification – regression – adaptive filtering – projection
  - principle of learning (non-linearity, error criterion)
  - difficulties: learning and generalization, the curse of dimensionality, overfitting and model complexity
  - NN structure and learning rule
  - How to use ANN’s?

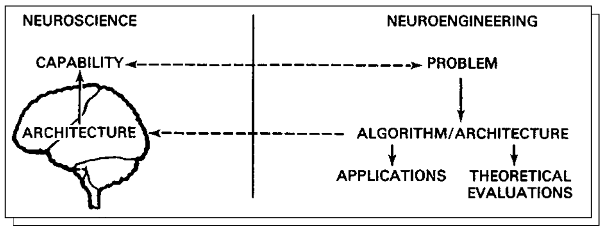
Why Anns ?

Von Neumann's computer	**(Human) brain**
determinism	fuzzy behaviour
sequence of instructions	parallelism
high speed	slow speed
repetitive tasks	adaptation to situations
programming	learning
uniqueness of solutions	different solutions
ex: matrix product	ex: face recognition

Brain "Properties"

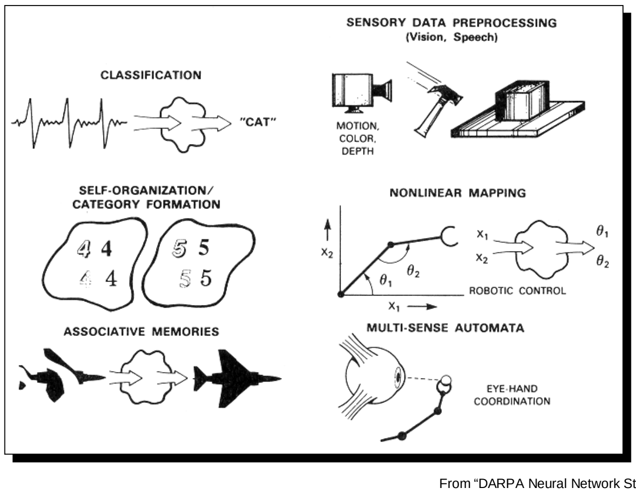
- Robust (cells die!) and fault-tolerant
- Flexible (does not need C or java…)
- Information is fuzzy, probabilistic, noisy, or inconsistent
- Highly parallel
- Learning
- Slow
- Small, compact, dissipates little power
- Auto-organization
- Non-unique behavior

Neurosciences - Neuroengineering



From "Handbook of neural computation", E. Fiesler, R. Beale eds., IOP and Oxford University Press, 1996, p. A2.2:3

Some examples of problems



From "DARPA Neural Network Study", 1988

Examples of tasks

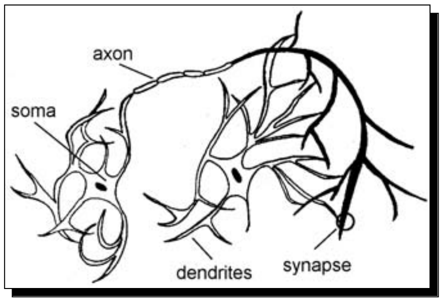
- face recognition
- time-series prediction
- process identification
- process control
- optical character recognition
- adaptive filtering
- etc.

Historical Background

1940 – 1965	Hebb	biological learning rule
	McCulloch & Pitts	binary decision units
	Rosenblatt	Perceptron - learning
1969	Minsky & Papert	limits to Perceptrons
1974	Werbos	back-propagation
1980 –	Hopfield	feedback networks
	Parker, LeCun, Rumelhart, McClelland	back-propagation
	Kohonen	Self-Organizing Maps

Biological Neuron - Hebb's Rule

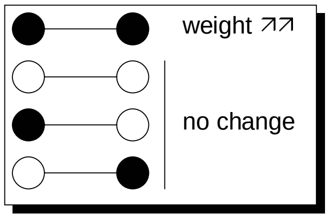
- 'Basic' scheme



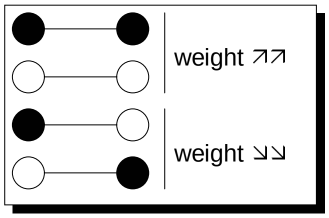
- Hebbs' rule: how coupling between neurons (synaptic values) influences and is influenced by the operation performed by a group of neurons

Hebb's rule

- *"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."*



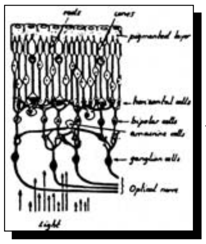
- Symmetrical Hebb's rule for artificial networks:
  - to avoid saturation
  - because 1 and 0 are conventional



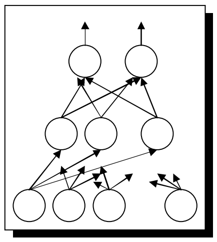
○ 0 ● 1

Similarity levels

- “elements” level
  - neurons → sums & non-linear functions
  - synapses → multiplication
  - other elements → not useful for computation
- functional blocks level
  - visual cortex → pattern recognition
  - sensory-motor cortex → control
  - …
- structure level
  - distributed information
  - ...



$\longleftrightarrow$



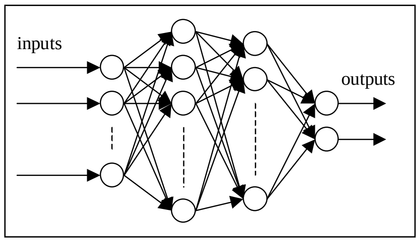
Artificial Neural Networks

- Artificial neural networks ARE NOT :
  - an attempt to understand biological systems
  - an attempt to reproduce the behavior of a biological system
- Artificial neural networks ARE :
  - a set of tools aimed to solve “perceptive” problems (“perceptive” <-> “rule-based”)

At least in the framework of this lecture, i.e. “Neural Computation”…

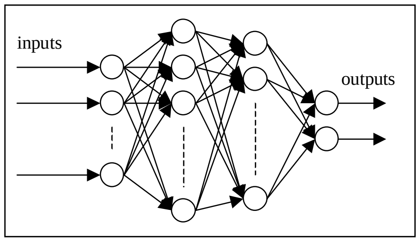
Why “Artificial neural networks” ?

- Structure: many computation units in parallel (McCulloch & Pitts 1943)
- Learning rule (adaptation of parameters): similar to Hebb’s rule (1949)



And that’s all...

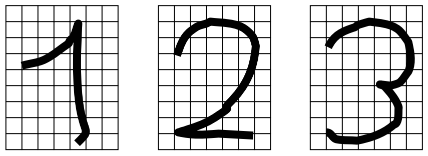
Learning



- Give - many - examples (*input-output pairs*, or *training samples*)
- Compute the parameters to fit the examples
- Test the result!

Why Neural Computation ?

- Hypothetical problem of optical character recognition (OCR)



- If: - image 256 x 256 pixels
  - 8-bits pixel values (256 gray levels)

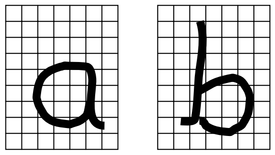
  then: - 10^{158000} different images

- Necessity to work with **features** !

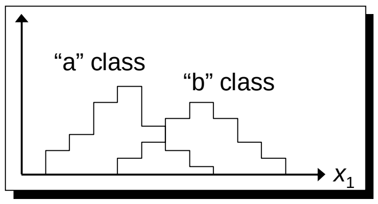
Michel Verleysen Introduction - 15

Feature Extraction

- Example of feature in OCR:
  ratio height/width (x_1) of the character



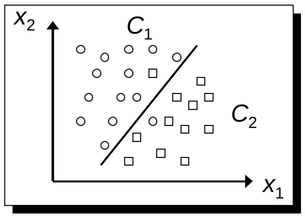
- Histogram of feature value



Michel Verleysen Introduction - 16

Multi-dimensional Features

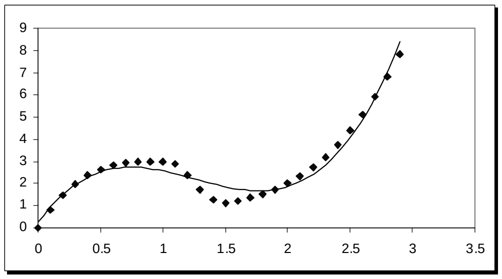
- Necessity to classify according to *several*, but *not too much* features



  - Several: because of the overlap in histogram with 1 feature
  - Not too much: because classification methods are easier in small dimensional spaces

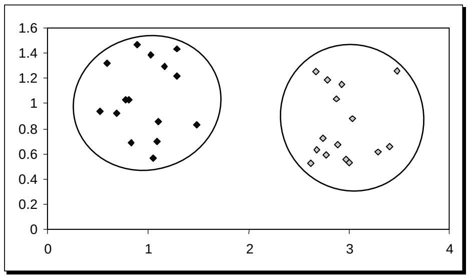
What can be done with ANNs ?

- regression
- classification
- adaptive filtering
- projection

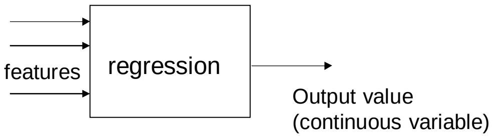


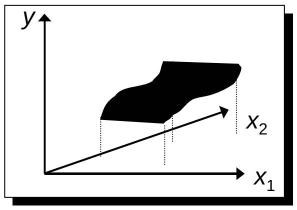
What can be done with ANNs ?

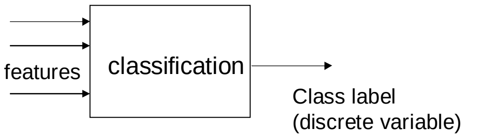
- regression
- classification
- adaptive filtering
- projection

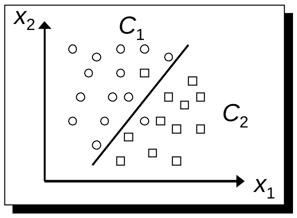


Classification - Regression



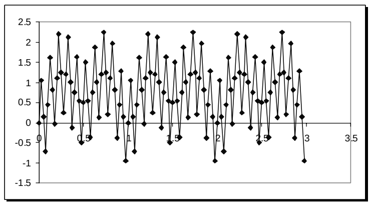


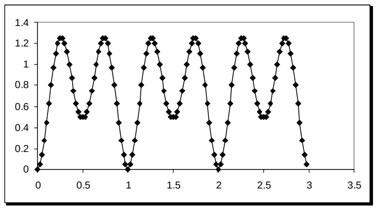




What can be done with ANNs ?

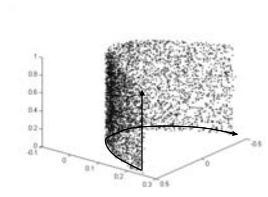
- regression
- classification
- adaptive filtering
- projection

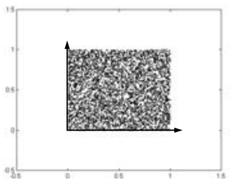




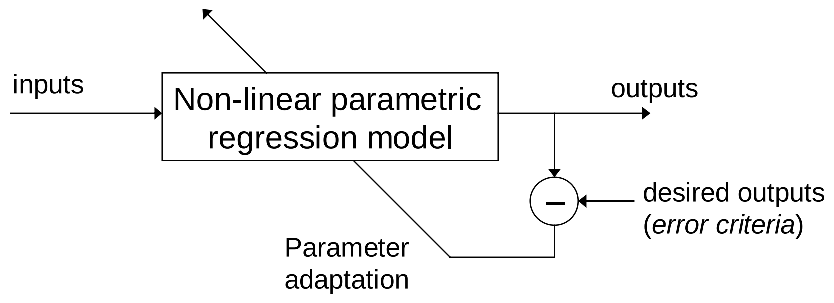
What can be done with ANNs ?

- regression
- classification
- adaptive filtering
- projection

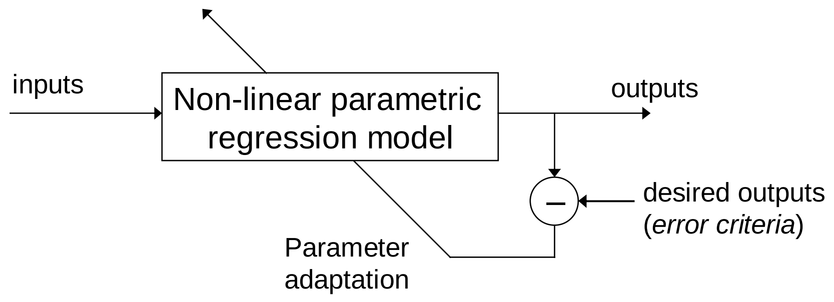




Principle

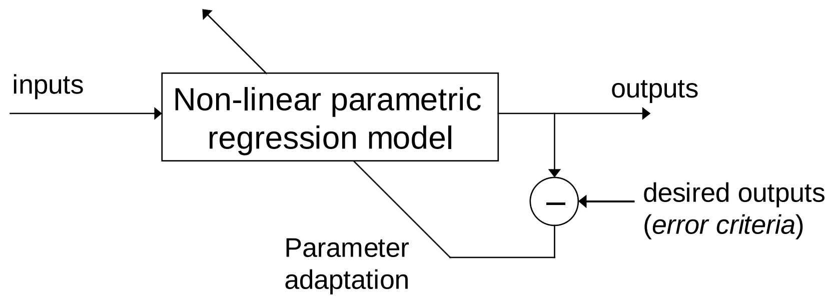

inputs
Non-linear parametric regression model
outputs
–
desired outputs (*error criteria*)
Parameter adaptation

Principle


inputs
Non-linear parametric regression model
outputs
–
desired outputs (*error criteria*)
Parameter adaptation

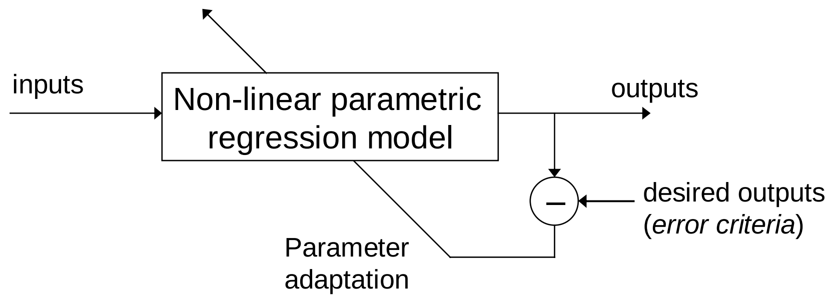
- Model: restrictions about the possible relation

Principle



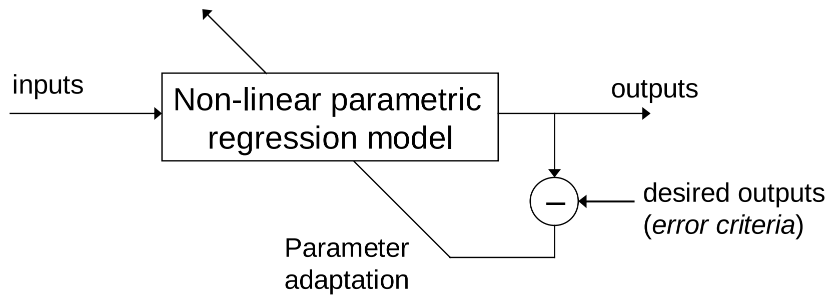
- Parametric: learning parameters to adjust (contain the information)

Principle



- Non-linear: more powerful than linear

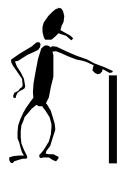
Principle


inputs
Non-linear parametric regression model
outputs
–
desired outputs (*error criteria*)
Parameter adaptation

Universal approximator !

Why non-linear ?

- If world was linear:

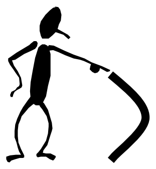


Push on a flexible bar, and it will shrink…

(one equilibrium point, stability, linearity, etc.)

Why non-linear ?

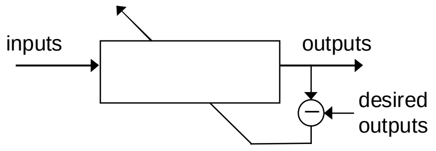
- But world is non-linear:



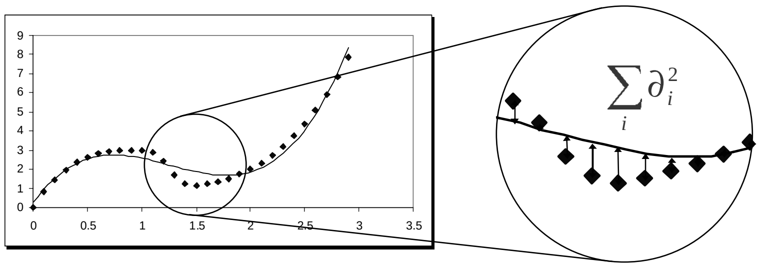
Push on a flexible bar, and it will bend, to the left or to the right !

(unstable equilibrium point, bifurcation, non-linearity, etc.)

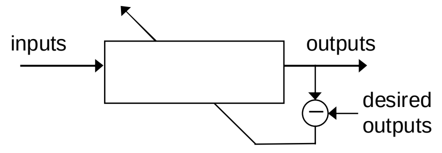
Error criterion



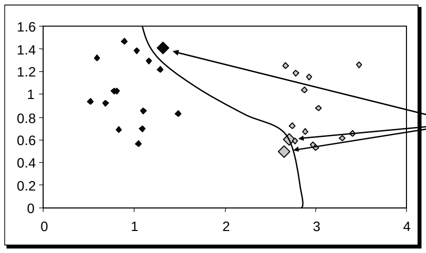
- regression
- classification
- adaptive filtering
- projection



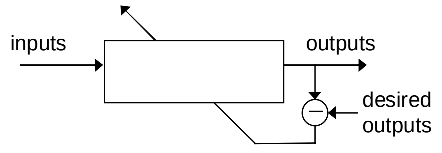
Error criterion



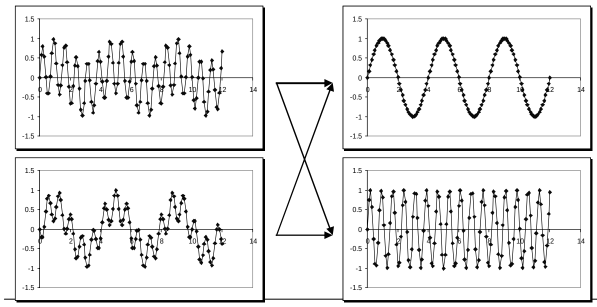
- regression
- classification
- adaptive filtering
- projection



Error criterion

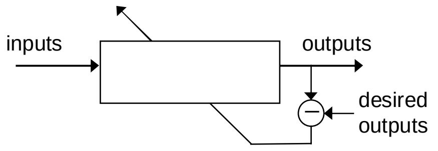


- regression
- classification
- adaptive filtering
- projection

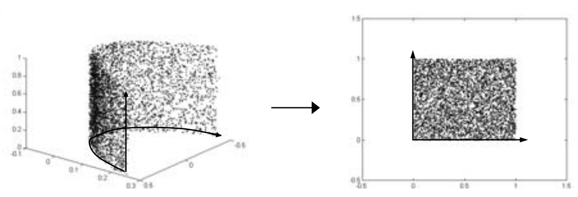


Independence between outputs

Error criterion



- regression
- classification
- adaptive filtering
- projection



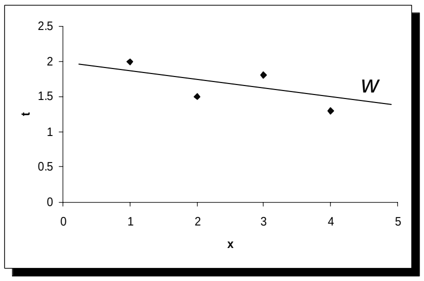
Independence between outputs

Polynomial Curve Fitting

- Strong similarity between polynomial curve fitting and regression with neural networks
  - polynoms $y = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$
  - neural networks $y = f(\boldsymbol{w}, \boldsymbol{x})$
- Error criterion $E = \sum_{p=1}^{P} \left(y^p - \underbrace{t^p}_{\text{desired output } (\textbf{t}\text{arget})}\right)^2$
- Polynoms: E is quadratic in $\boldsymbol{w} \rightarrow$ min(E) is the solution of a set of linear equations
  Neural networks: E is non-linear in $\boldsymbol{w} \rightarrow$ local minima

Learning - generalization

- Learning: process of
  - finding parameters ***w*** which
  - minimize the error function *E*
  - given a set of *input-output pairs* (x^p, t^p)

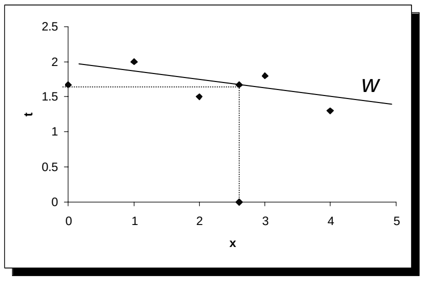


Michel Verleysen

Introduction - 35

Learning - generalization

- generalization: process of
  - assigning an output value *y* to
  - a new input *x*
  - given the parameters vector ***w***

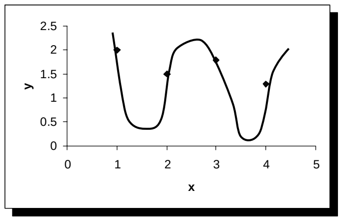


Michel Verleysen

Introduction - 36

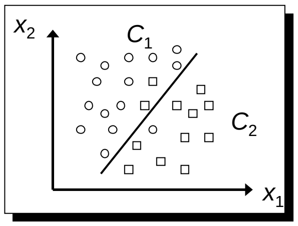
Overfitting

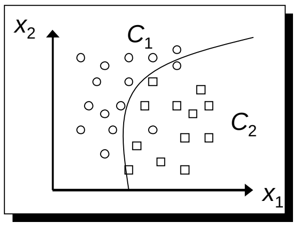
- Compromise between
  - model complexity and
  - number of learning pairs

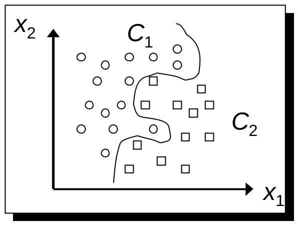


- Neural networks are
  - universal approximation models with
  - a (relatively) low complexity
- But overfitting must be a serious concern !

Overfitting in classification



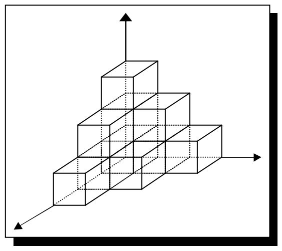




Model complexity

- Compromise between performance and complexity: difficult to evaluate a priori
- Alternative: to control the *effective* complexity:
  - new error function $\tilde{E} = E + \lambda\Omega$
  - penalty term (example) $\Omega = \frac{1}{2}\int \left(\frac{d^2y}{dx^2}\right)^2 dx$
- But:
  - many possible forms of penalty terms
  - λ difficult to choose/adjust

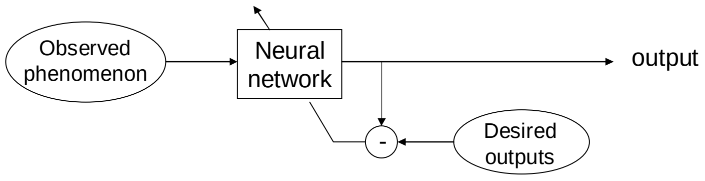
The Curse of Dimensionality



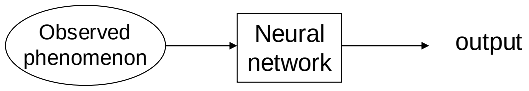
- Histograms: the number of boxes is exponential with the dimension
- More features → reduced performances
- If # data is limited → dimension must be low
- Concept of intrinsic dimensionality of data

Supervised - unsupervised learning

- Supervised learning: building an input-output relation known through examples (input-output pairs)



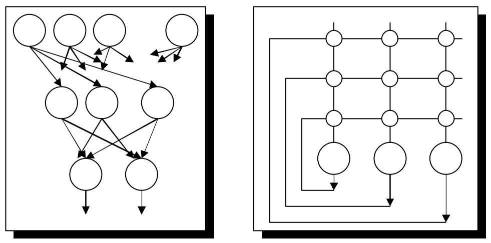
- Unsupervised learning: modeling a property of the input data (for example density)



- Reinforcement learning: outputs are good/bad (but *how much* good or bad is not know!)

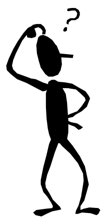
What is a neural network ?

- Model = structure + learning rule
  - structure



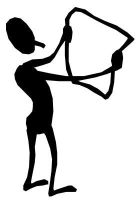
  - learning rule
    How to compute (estimate) the parameters of the network ?
- Several learning rules for one structure
- Similar learning rules for several structures

How to use ANNs



1) examine problem

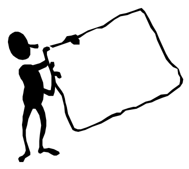
How to use ANNs



1) examine problem
2) formulate problem in terms of data analysis

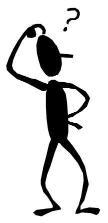
How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning

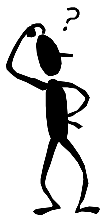


How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data



How to use ANNs

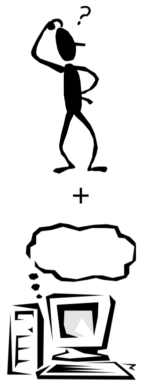


1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model

How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model
6) learn

How to use ANNs

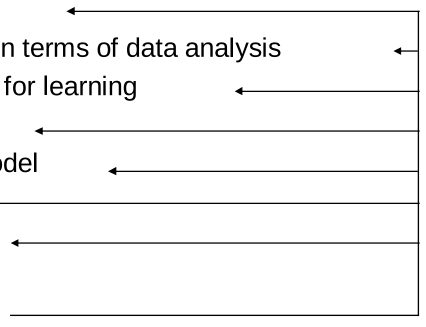


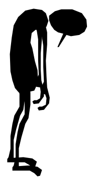
1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model
6) learn
7) test the results

How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model
6) learn
7) test the results

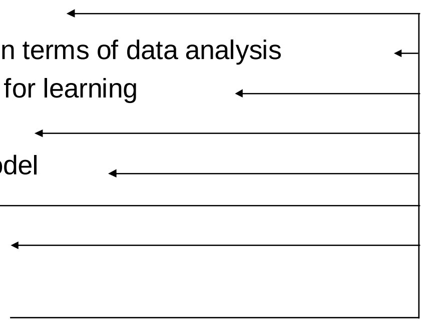
unsuccessful :





How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model
6) learn
7) test the results



successful !

How to use ANNs

1) examine problem
2) formulate problem in terms of data analysis
3) collect -many- data for learning
4) understand data
5) choose an ANN model
6) learn
7) test the results

successful !

Sources and References

- Some ideas developed in these slides come from:
  - Handbook of neural computation, E. Fiesler, R. Beale eds., IOP and oxford university press, 1996
  - Neural networks for pattern recognition, C. Bishop, Clarendon press, Oxford, 1995
- Other references:
  - Les Réseaux de Neurones Artificiels, F. Blayo, M. Verleysen, Presses Universitaires de France, collection "Que sais-je?", Paris (1996), 126 pages. *Short introduction in French, not sufficient, but easy to read.*
  - Pattern classification, R.O. Duda, P.E. Hart, D.G. Strok, second edition, Wiley, 2000 (a good book on pattern classification, including, but not limited to, neural networks).