Neural and Evolutionary Computing

- What is this course about ?
- Computational Intelligence
- Neural Computing
- Evolutionary Computing
- Related techniques

What is this course about ?

- As almost all courses in Computer Science it is about problem solving
- Its main aim is to present techniques to solve hard problems
- There are problems which are hard:
  - both for humans and computers (e.g. large combinatorial optimization problems, multimodal / multiobjective optimization problems etc) – computationally hard problems
  - for computers but rather easy for humans (e.g. character recognition, face recognition, speech recognition etc) – ill posed problem

Computationally hard problems

- Problems characterized by a large space of solutions for which there are no exact methods of polynomial complexity (so-called NP hard problems) – they are characterized by a huge size search space (which cannot be exhaustively searched)

Examples:

- Satisfiability problem (SAT): find the values of boolean variables for which a logical formula is true. For n variables the search space has the size 2^n
- Travelling Salesman Problem (TSP): find a minimal cost tour which visits n towns. The search size is (n-1)! (in the symmetric case, it is (n-1)!/2)

Ill-posed problems

- The particularity of problems which are easy for humans but hard for computers is that they are ill-posed, i.e. there is difficult to construct an abstract model which reflects all particularities of the problem

- Let us consider the following two problems:
  - Classify the employees of a company in two categories: first category will contain all of those who have an income larger than the average salary per company and the second category will contain the other employees

  - Classify the employees of a company in two categories: first category will contain all those which are good candidates for a bank loan and the second category will contain the other employees

Ill-posed problems

- In the case of the first problem there is easy to construct a rule-based classifier:

  IF income > average THEN Class 1
  ELSE Class 2

- In the case of the second problem it is not so easy to construct a classifier because there are a lot of other interrelated elements (health status, family, career evolution etc) to be taken into account in order to decide if a given employee is reliable for a bank loan. A bank expert relies on his experience (previous success and failure cases) when he makes a decision

Ill-posed problems

- Differences between well-posed and ill-posed problems:

Well-posed problems:

- There is an abstract model which describes the problem
- Consequently, there is a solving method, i.e. an algorithm

Ill-posed problems:

- They cannot be easily formalized
- There are only some examples for which the results is known
- The data about the problem could be incomplete or inconsistent
- Thus, traditional methods cannot be applied

Ill-posed problems

The methods appropriate for ill-posed problems should be characterized by:

- Ability to extract models from examples (learning)
- Ability to deal with dynamic environments (adaptability)
- Ability to deal with noisy, incomplete or inconsistent data (robustness)
- Ability to provide the answer in a reasonable amount of time (efficiency)

The field dealing with such kind of methods is called “computational intelligence” or “soft computing”

Computational Intelligence and Soft Computing

Computational Intelligence

"is a branch of the study of artificial intelligence; it aims to use learning, adaptive, or evolutionary algorithms to create programs that are, in some sense, intelligent. " [Wikipedia 2012]

"it deals with the study of adaptive mechanisms which allow the simulation of the intelligent behaviour in complex and/or dynamic environments"

Soft Computing

"is a collection of new techniques in computer science, especially in artificial intelligence; unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost." [Wikipedia 2012]

Computational Intelligence and Soft Computing

Computational Intelligence

"is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional approaches, i.e., first principles modeling or explicit statistical modeling, are ineffective or infeasible. It primarily includes artificial neural networks, evolutionary computation and fuzzy logic." [Wikipedia - 2014]

Soft Computing

"is a term applied to a field within computer science which is characterized by the use of inexact solutions to computationally hard tasks such as the solution of NP-complete problems, for which there is no known algorithm that can compute an exact solution in polynomial time. Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind." [Wikipedia - 2014]

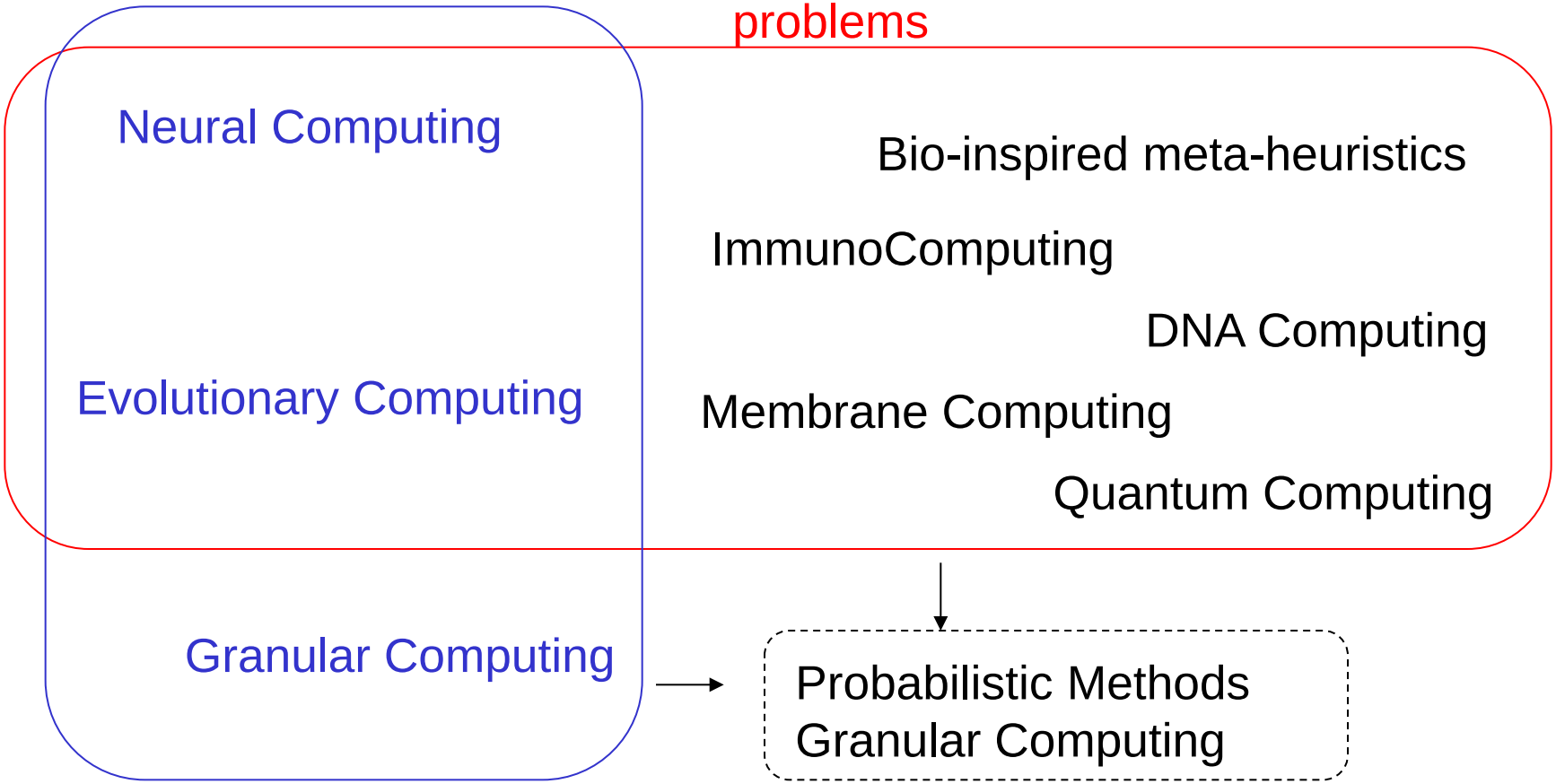
Computational Intelligence

Main components:	Tool/technique	Inspiration source:
Neural Computing	Artificial Neural Networks	Human brain
Evolutionary Computing	Evolutionary Algorithms	Biological evolution
Granular Computing	Fuzzy Sets/Rough Sets/ Probabilistic Reasoning	Human reasoning and natural language

CI covers all branches of science and engineering that are concerned with understanding and solving problems for which effective traditional algorithms do not exist.

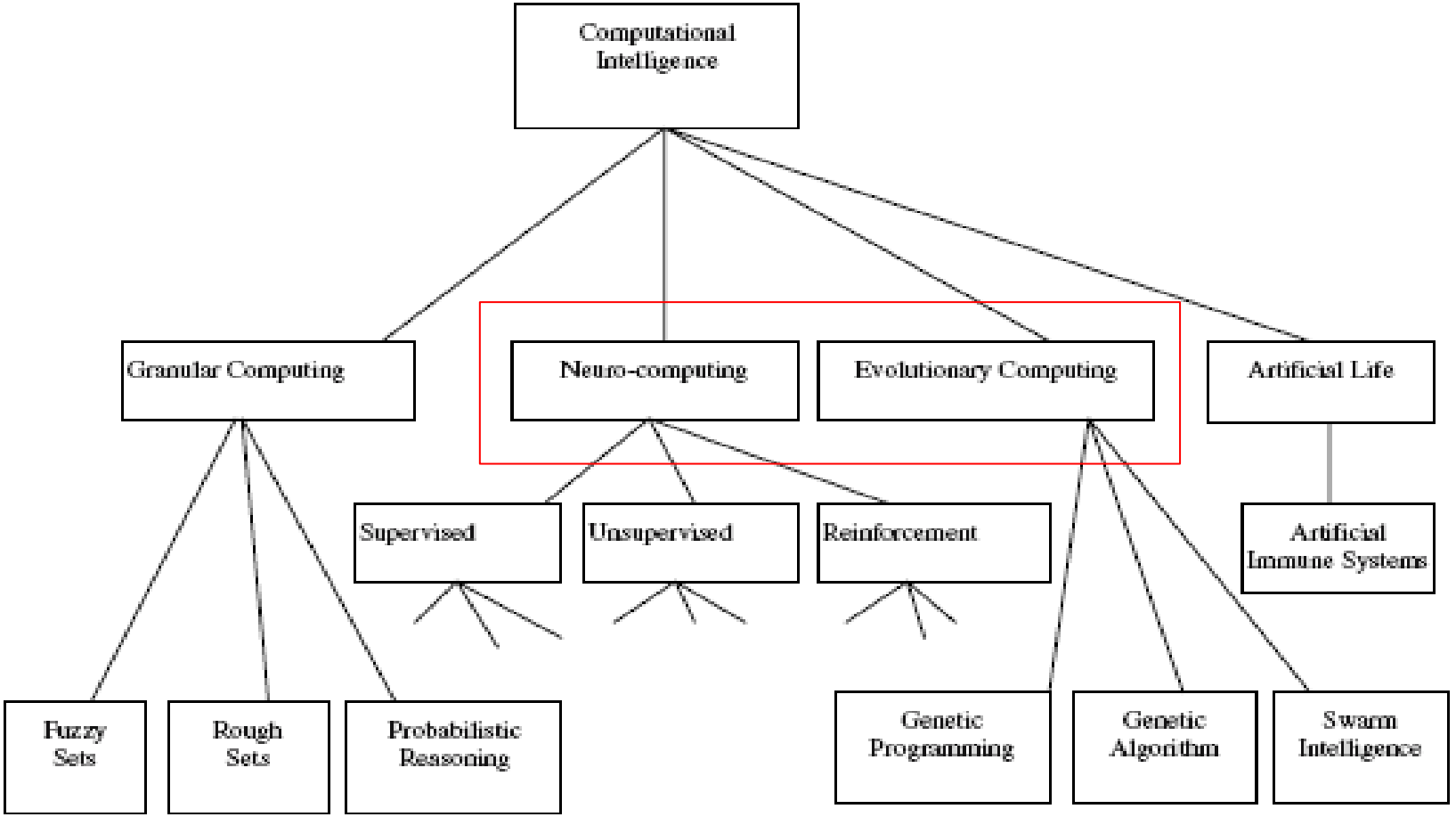
Computational Intelligence and Natural Computing

Natural computing = methods inspired by the nature way of solving problems



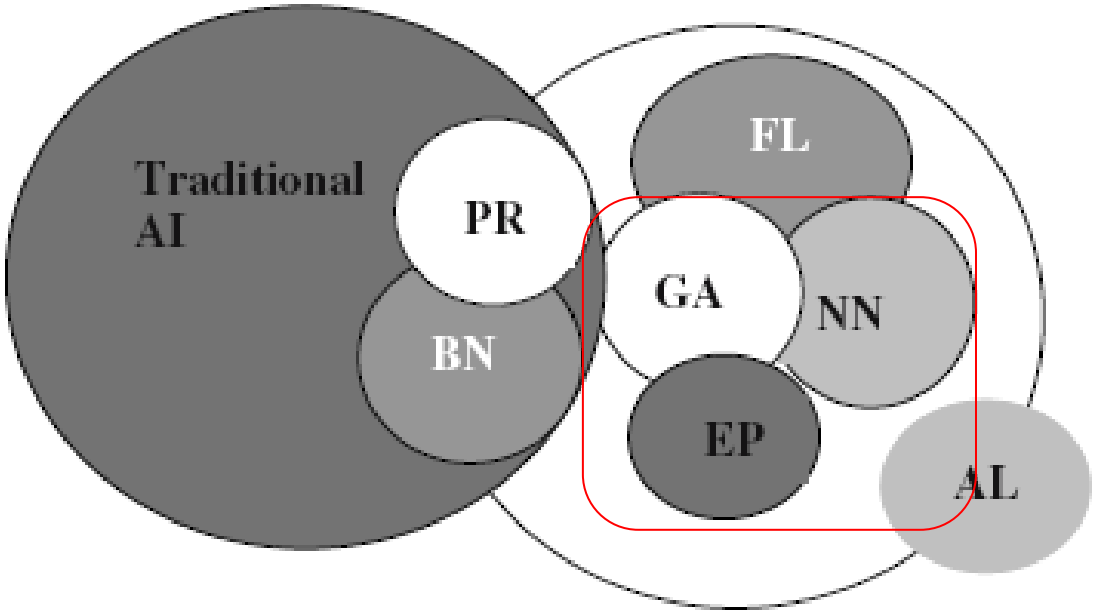
Computational Intelligence

[A. Konar – Computational Intelligence, 2007]



Computational Intelligence

[A. Konar – Computational Intelligence, 2007]



PR= Probabilistic reasoning, BN= Belief networks, FL= Fuzzy logic, NN= Neural nets, GA= Genetic algorithms, EP= Evolutionary programming, AL= Artificial life.

Neural Computing

Basic principles

The biological model

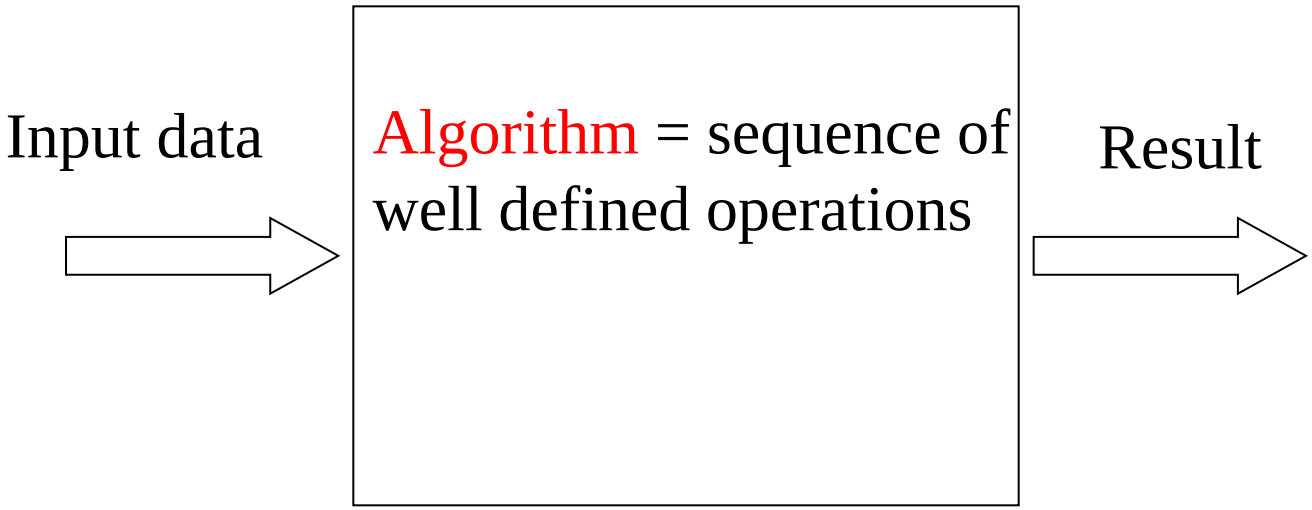
Elements of an artificial neural network

Classes of neural networks

Applications

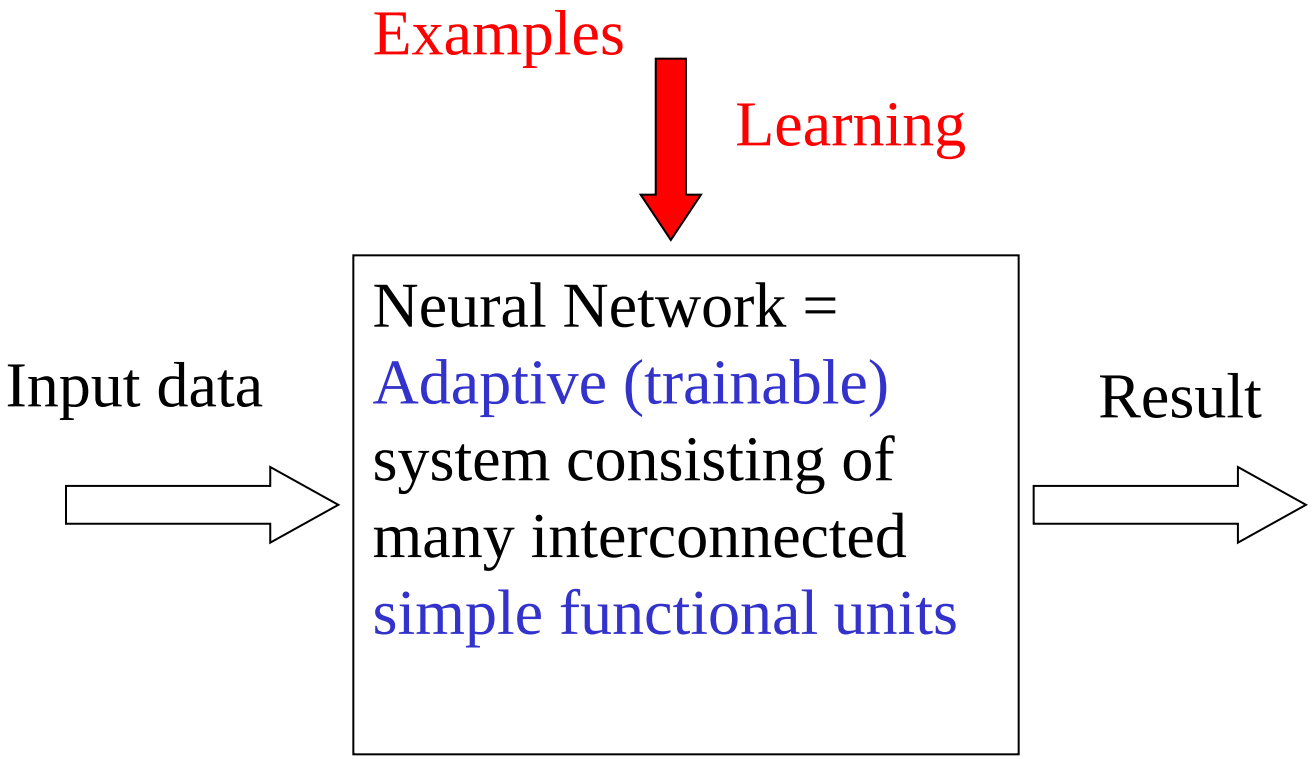
Neural Computing

Traditional problem solving approach (appropriate for well-defined problems)



Neural Computing

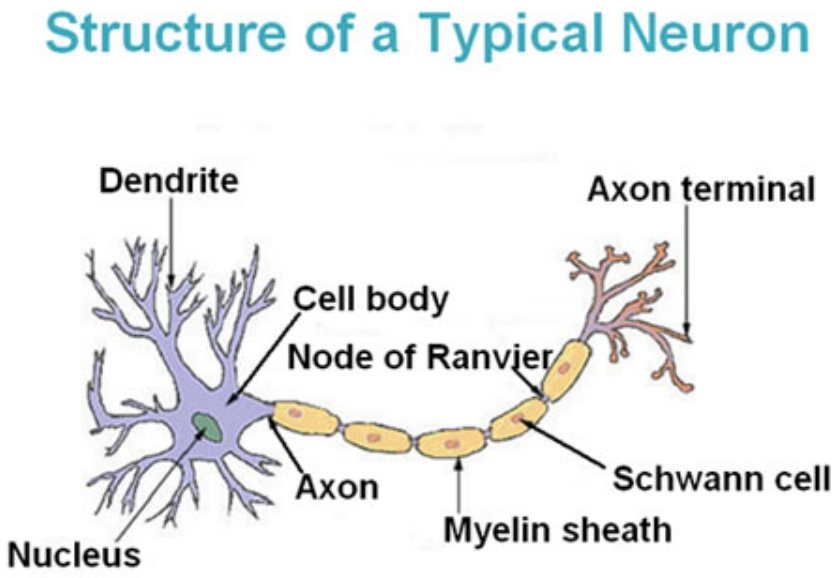
The neural (machine learning) approach:

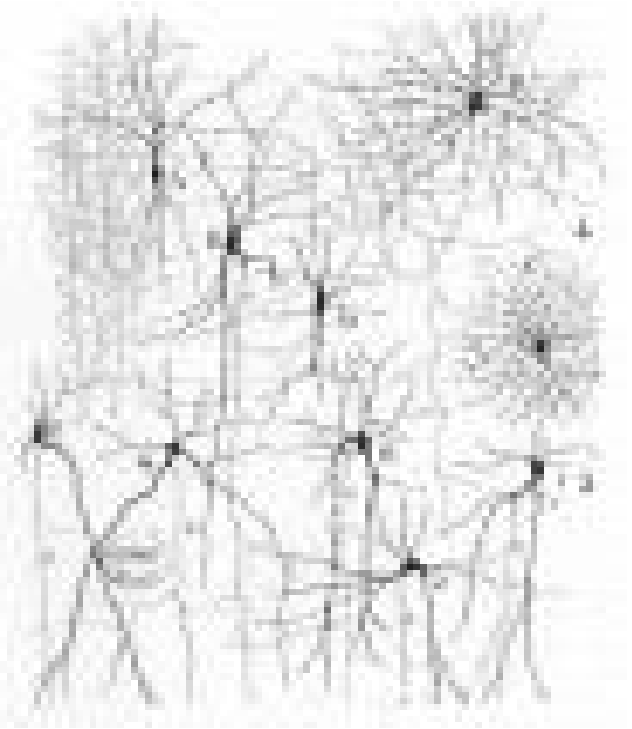


Neural Computing

Human brain

cca 10^{10} neurons, cca 10^{14} connections

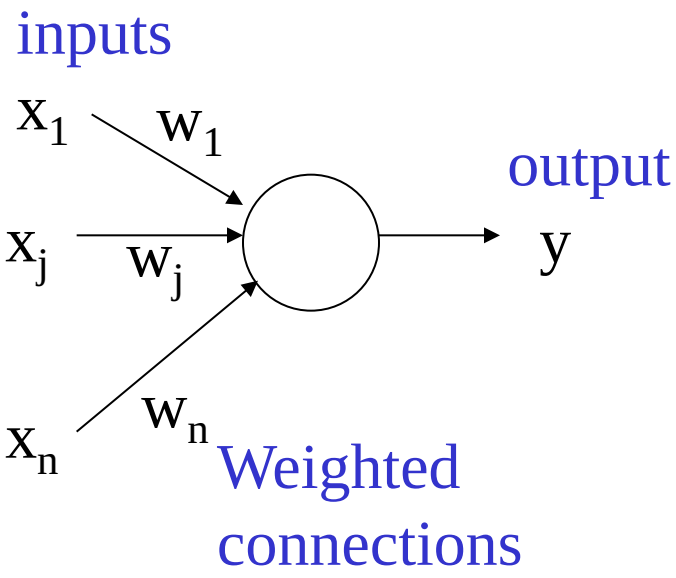




Artificial neural networks

ANN = set of interconnected functional units

Functional unit = input connections + aggregation function + activation function



Aggregation function

$$y = f\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

$$f(u) = \mathrm{sgn}(u) \qquad f(u) = \tanh(u)$$

$$f(u) = H(u) \qquad f(u) = \frac{1}{1+\exp(-u)}$$

activation functions

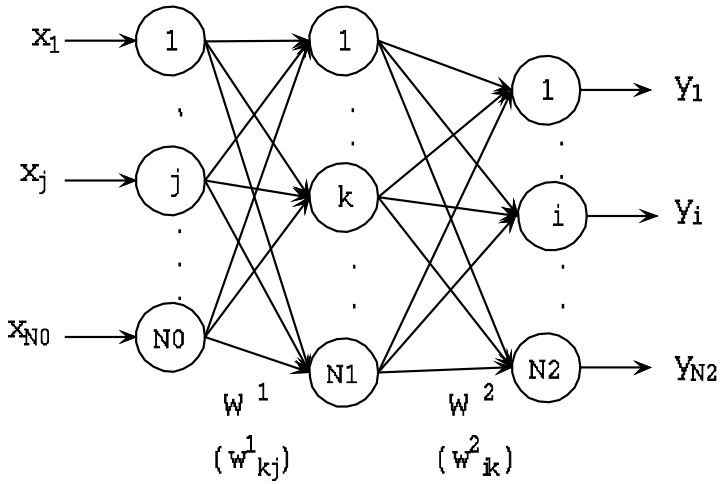
Artificial neural networks

ANN components:

- Architecture

- Functioning

- Learning: find the adaptive weights

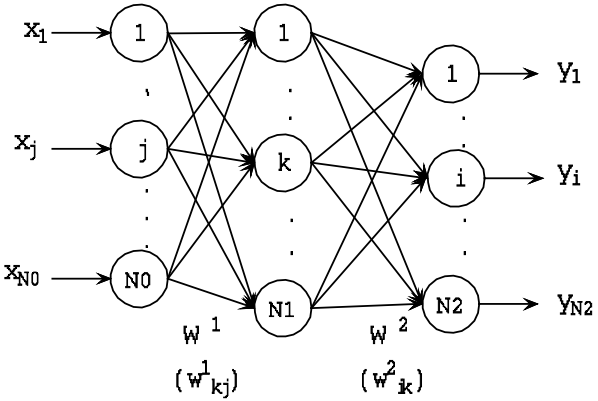


$$y_i = f\left(\sum_{k=0}^{N1} w_{ik}^2 f\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),\ i = \overline{1, N2}$$

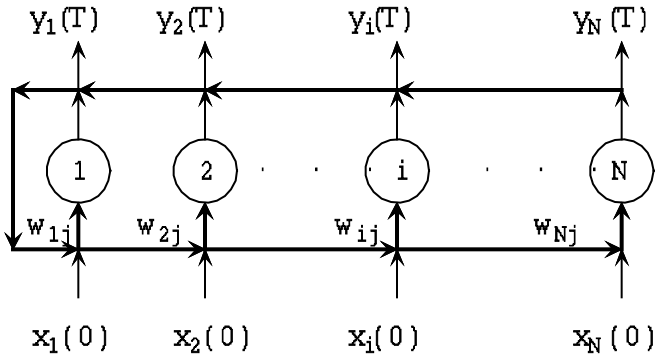
A feedforward neural network

Artificial neural networks

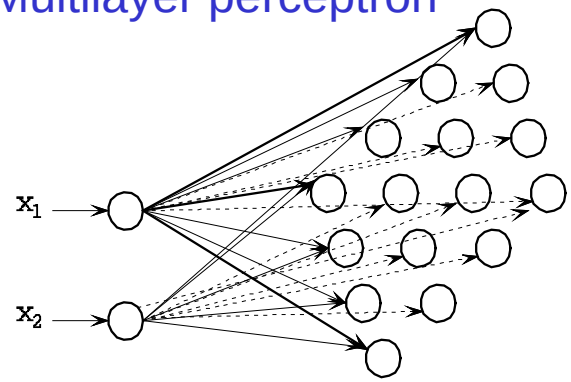
Neural networks variants:



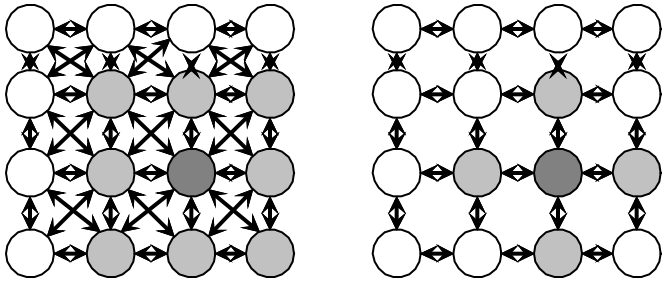
Multilayer perceptron



Hopfield model



Kohonen network



Cellular neural network

Artificial neural networks

Learning = process of extracting a (computational) model from examples
= compute the adaptive parameters (weights) of the network

Learning variants:

- Supervised (with a teacher): use the difference between the desired (correct) output and the actual output of the network to compute the adjustments of the adaptive weights
- Unsupervised (without a teacher): use only the correlations between input data (no knowledge of the correct answer)
- Reinforcement: use reward and penalty values to adjust the weights (no use of the difference between the correct and actual outputs)

ANN applications

- Classification
  - Supervised and unsupervised classification of data
  - Character/image/speech recognition
- Approximation
  - Estimate the relationship between different variables (e.g. estimate the software projects effort based on software size, …)
- Prediction
  - Extract time series models from data (e.g. predict de evolution of the exchange rate, …)
- Control
  - Nonlinear systems modelling
- Optimization
  - Electronic circuits design
- Signal analysis
  - Adaptive filters

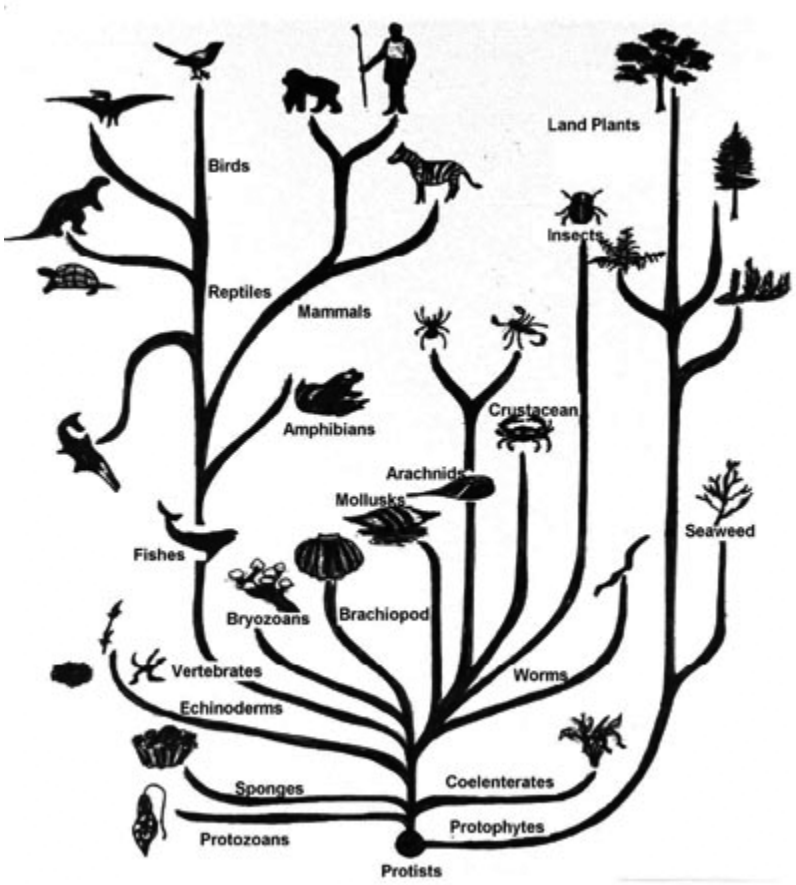
Evolutionary Computing

Basic principles

The structure of an Evolutionary Algorithm

Traditional classes of Evolutionary Algorithms

Applications of Evolutionary Computing



Evolutionary Computing

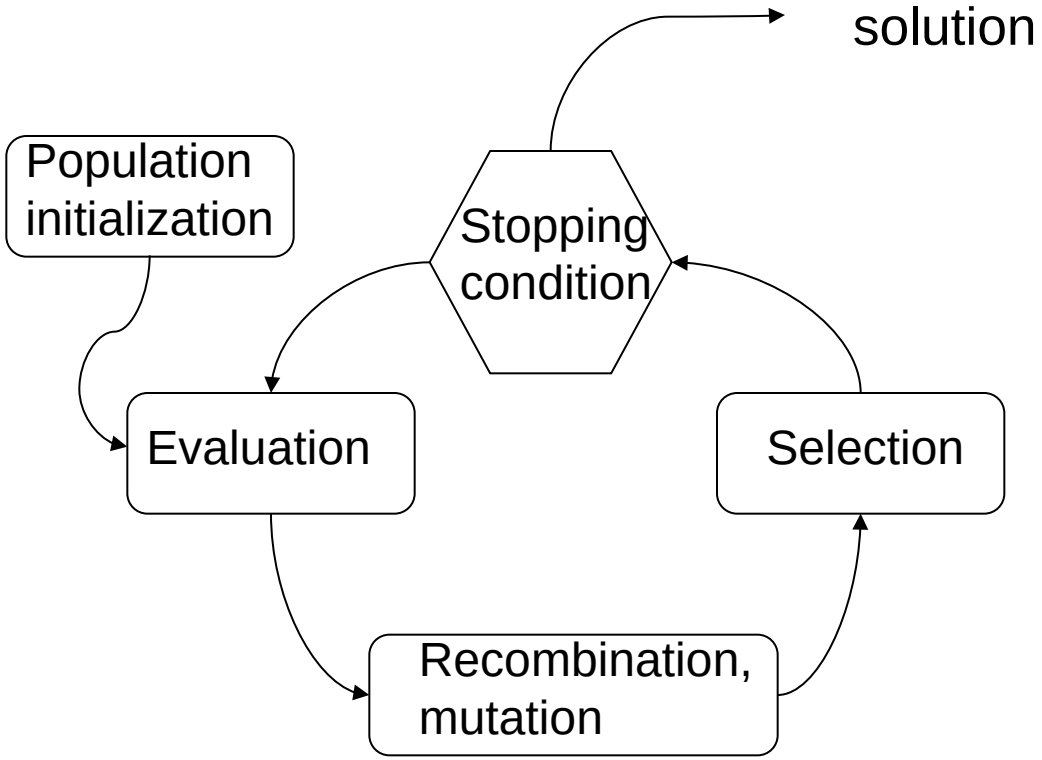
- It is inspired by the biological evolution; it stands on the principles of Darwin's natural evolution theory: genetic inheritance and survival of the fittest
- The solution of a problem is identified by searching the solution space using a population of agents (individuals or chromosomes)
- The elements of the population are encoded depending on the particularities of the problem (strings of binary or real values, trees, graphs etc)

Evolutionary Computing

There is an analogy between the evolution in nature and problem solving

Natural Evolution		**Problem Solving**
Environment	⟷	Problem
Individual	⟷	Potential solution
Fitness	⟷	Solution quality

Structure of an EA



EA =
Iterative process based on the sequential application of several operators:
- recombination (crossover)
- mutation
- selection

on a randomly initialized population

EA classes

- Genetic Algorithms (GA):
  - Binary encoding of population elements
  - The main operator is the recombination (crossover)
  - The mutation is applied with a small probability
  - Appropriate for combinatorial optimization problems (search in discrete spaces)

- Evolution Strategies (ES):
  - Real encoding of population elements
  - The main operator is the mutation
  - Appropriate for solving optimization/search problems over continuous domains

EA Classes

- Genetic Programming (GP):
  - The population elements are computational structures (tress, arithmetical/logical expressions, programs etc.)
  - Appropriate for evolutionary design of computational structures (programs, circuits etc)
- Evolutionary Programming (EP):
  - Real encoding of population elements
  - The mutation is the only operator
  - Used to solve optimization problems on continuous domains

Current variants: hybrid techniques

Classes of optimization problems

- Constrained and unconstrained optimization
  - Non-differentiable or discontinous functions or functions without a closed form (their evaluation is based on simulations)
  - Such kind of problems frequently appear in engineering design and in planning
- Multimodal optimization
  - For functions having many local/global optima
  - Typical for industrial design
- Multiobjective optimization
  - There are several conflicting objectives to be optimized
  - Typical for industrial design, data analysis and decision making
- Optimization in dynamic and/or noisy environments
  - The optimization criteria changes in time or its evaluation is influenced by random factors

Applications

- Planning (e.g. timetabling, tasks scheduling)
- Prediction (e.g. currency exchange rate evolution)
- Data and image analysis
- Structure prediction (e.g. protein structure prediction starting from the aminoacids sequence)
- Neural networks design
- Evolutionary art

Related techniques

- Models inspired by the intelligence of swarms

  PSO – Particle Swarm Optimization (Eberhart, Kennedy -1995)

  http://www.swarmintelligence.org/

  http://www.particleswarm.info/

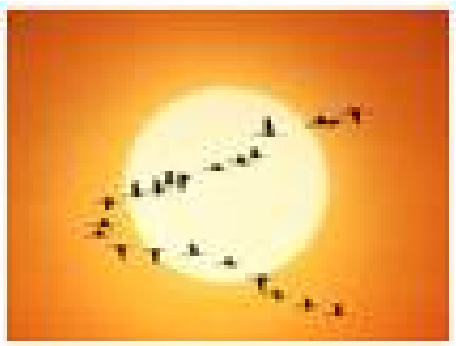
  – They are inspired by birds flocking, fish schooling, bees swarms and the behaviour of other "social" entities

  – During the search process each individual is guided by:

    - The collective experience
    - The individual Experience

  – Applications:

    - Optimization
    - Control (nano robots used in medicine)
    - Creating complex interactive environments (in games or cartoons)



Related techniques

- Ant based models

  ACO – Ant Colony Optimization (M. Dorigo, 1992)
  [http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html]

  AS – Ant Systems

  - They are inspired by the behaviour of ant colonies when they search for food or organize their nest
  - Stigmergy is a main concept which expresses the indirect communication between ants by using the pheromone trails

  Applications:

  - Optimization (routing problems)
  - Planning (allocation problems)
  - Data analysis (clustering)
  - Image classification

Related techniques

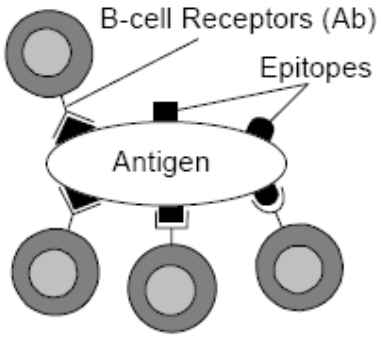
- Immune Systems Model

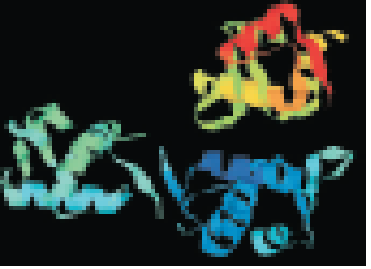
  AIS– Artificial Immune Systems (L. Castro, 1999)
  [http://www.dca.fee.unicamp.br/~lnunes/immune.html]

  – It is inspired by the ability of the biological immune systems to recognize the pathogen agents and to react to an attack

Applications:

- Intruder Detection Systems
- Multimodal optimization
- Data mining (clustering)





Related techniques

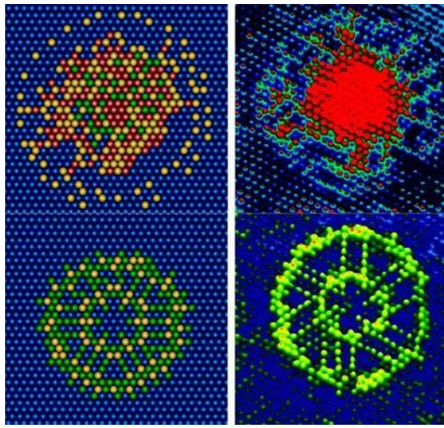
- DNA (molecular) Computing

First approach: Adleman's experiment (1994) – solving the TSP problems for 7 towns by using tools from molecular biology

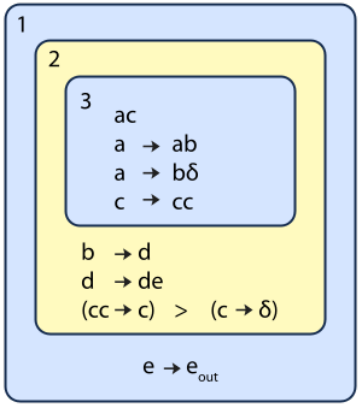
Current status:

- autonomous biomolecular computer of molecular scale (2004)
- a superthin computer of just two molecules thick has been designed (it generates various patterns) (2010)
- Algorithms inspired by operations on DNA structures (splicing, cloning, filtering)

A DNA computer is basically a collection of specially selected DNA strands whose combinations will result in the solution to some problem



Related techniques



- Membrane Computing (P-systems) [http://ppage.psystems.eu]

- First model: P-systems proposed by Gh. Paun (1998)
- A P system is a computing model which abstracts from the way the alive cells process chemical compounds in their compartmental structure.
- They process multisets of symbol objects placed in a hierarchically structured system of membranes (inspired by the structure of cells)
- Many theoretical results concerning their computation power but less practical applications (however there are reported applications in applications, in biology, linguistics, computer science, management)

Related techniques

- Artificial Bee Colony (Karaboga, 2005) http://mf.erciyes.edu.tr/abc/
  - Inspired by the artificial behavior of honey bees
- Biogeography Based Optimization (BBO Algorithms) - D. Simon, 2008 http://academic.csuohio.edu/simond/bbo/
  - Inspired by geographical distribution of biological organisms
- Fireflies Algorithm (X.S. Yang, 2008)
  - inspired by the flashing behaviour of fireflies
- Cuckoo Search (X.S. Yang & S. Deb, 2009)
  - Inspired by the habits of some cuckoo species to lay their eggs in the nests of other birds
- Bat Algorithm (X.S. Yang & A.H Gandomi, 2010)
  - Inspired by echolocation abilities of bats

Course structure

- Artificial Neural Networks for classification, approximation, prediction, optimization
  - Feedforward neural networks (BackPropagation, Radial Basis Functions)
  - Recurrent neural networks (Hopfield model)
- Random optimization algorithms
  - Random Search
  - Simulated annealing
- Evolutionary algorithms
  - Genetic algorithms
  - Evolutionary strategies
  - Evolutionary and Genetic Programming
  - Evolutionary algorithms for multi-objective optimization
  - Evolutionary design
  - Parallel and distributed evolutionary algorithms
- Related techniques: PSO (particle swarm optimization), ACO (ant colony optimization), AIS (artificial immune systems), DE (differential evolution), EDA (estimation of distribution algorithms) etc

Lab structure

Lab 1: Classification problems (pattern recognition) - feedforward NN

Lab 2: Approximation and prediction problems – feedforward NN

Lab 3: Combinatorial optimization problems – Simulated Annealing, Genetic Algorithms

Lab 4: Continuous optimization problems (nonlinear programming)- Evolution Strategies

Lab 5: Evolutionary design problems - Genetic Programming

Lab 6: Multiobjective optimization problems - MOEA

Lab 7: Applications of related techniques (ACO, PSO, AIS)

Testing environments:

Scilab, Weka, R

References

Course materials:

http://web.info.uvt.ro/~dzaharie/nec2014

References:

A.Engelbrecht: Computational Intelligence. An Introduction, John Wiley and Sons, 2007

L. Rutkowski: Computational Intelligence: Methods and Techniques, Springer, 2008

A.Konar: Computational Intelligence: Principles, Techniques and Applications, Springer, 2005

Z. Michalewicz, D. Fogel: How to Solve It. Modern Heuristics. Springer, 1999

Evaluation

Grading:

Project: 60-80%

Written test (open books): 20%

Lab activity: 20%