Learning from nature

Olle Gällmo

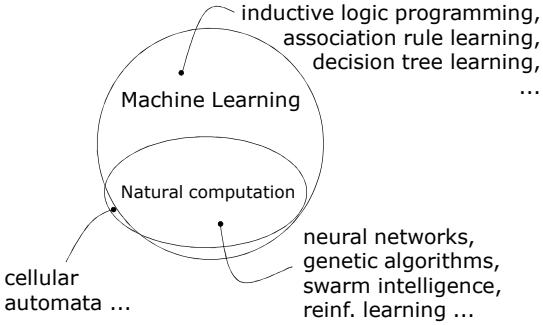
Uppsala University
Department of Information Technology

Natural computation

- Using computers to model/simulate natural phenomena
  - to learn more about these phenomena
  - to learn new ways to solve computational problems
  - to learn how to build computational devices from biological material

Computers and humans

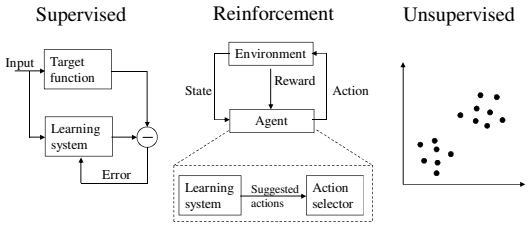
NC and Machine learning



What is learning?

- The ability to improve over time, based on experience
- Why?
  - Solutions to problems are not always programmable!
- Examples
  - Handwritten character recognition
  - Adaptive control of production processes
  - Game programs that adjust parameters and/or strategies over time
  - Learning to walk by trial-and-error

Three forms of learning



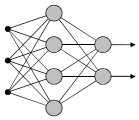
Techniques (examples)

- Artificial neural networks (ANNs)
  - Inspired by biological nervous systems
  - E.g. Multilayer perceptrons, Self-Organizing Maps
- Reinforcement learning (RL)
  - Inspired by psychology, ethology and behaviourism
  - E.g. Menace, Q-Learning, TD(λ)
- Evolutionary Computing (EC)
  - Inspired by genetics, natural selection and evolution
  - E.g. Genetic algorithms, Genetic Programming
- Swarm intelligence
  - Inspired by social animals (bird flocks, ants, etc.)
  - E.g. Particle Swarm Optimization, Ant Colony Optimization, Cellular automata

Techniques (examples)

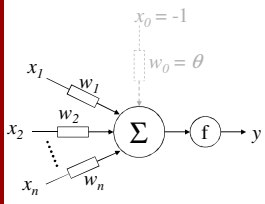
- **Artificial neural networks (ANNs)**
  - Inspired by biological nervous systems
  - E.g. Multilayer perceptrons, Self-Organizing Maps
- Reinforcement learning (RL)
  - Inspired by psychology, ethology and behaviourism
  - E.g. Menace, Q-Learning, TD(λ)
- Evolutionary Computing (EC)
  - Inspired by genetics, natural selection and evolution
  - E.g. Genetic algorithms, Genetic Programming
- Swarm intelligence
  - Inspired by social animals (bird flocks, ants, etc.)
  - E.g. Particle Swarm Optimization, Ant Colony Optimization, Cellular automata

Artificial neural networks



- Begun in the 1940's
- Many simple processing elements (neurons), operating in parallel and communicating through weighted connections
- Based on very simple models of biological neurons and synaptic connections
- Used both for industrial applications and as a model to study biological systems

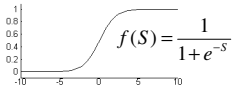
An artificial neuron



$$y = f(S)$$

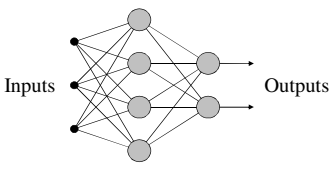
$$S = \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=0}^{n} w_i x_i$$

$f(S)$ = any non-linear, saturating function, e.g. a step function or a sigmoid:



$$f(S) = \frac{1}{1+e^{-S}}$$

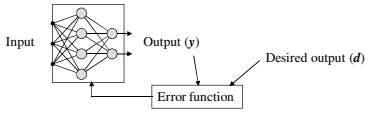
Multilayer perceptrons



Can approximate any function to any degree of accuracy, given a sufficiently rich internal structure (number of nodes and layers)

Most common training algorithm: *Back propagation*

Back propagation



The contribution to the error E from a particular weight w_{ji} is $\frac{\partial E}{\partial w_{ji}}$

The weight should be moved in proportion to that contribution, but in the other direction:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

⇨ Error function and activation function must both be differentiable.

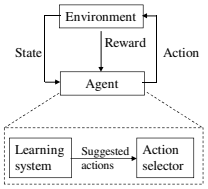
Artificial neural networks …

- store information in the weights, not in the nodes
- are trained, by adjusting the weights, not programmed
- can generalize to previously unseen data
- are adaptive
- are concurrent
  - well suited for parallel simulation and/or hardware implementation
- are fault tolerant

Techniques (examples)

- Artificial neural networks (ANNs)
  - Inspired by biological nervous systems
  - E.g. Multilayer perceptrons, Self-Organizing Maps
- **Reinforcement learning (RL)**
  - Inspired by psychology, ethology and behaviourism
  - E.g. Menace, Q-Learning, TD(λ)
- Evolutionary Computing (EC)
  - Inspired by genetics, natural selection and evolution
  - E.g. Genetic algorithms, Genetic Programming
- Swarm intelligence
  - Inspired by social animals (bird flocks, ants, etc.)
  - E.g. Particle Swarm Optimization, Ant Colony Optimization, Cellular automata

Reinforcement learning



- Reward: an evaluation of the environmental state (only indirectly an evaluation of the agent's actions)
- Goal: To make decisions (find actions) that maximise the long term reward received by the agent.
- The agent must be allowed to *explore*, i.e. sometimes do actions that at the time seem sub-optimal.
- Learning by trial-and-error

MENACE
(D. Michie 1961)

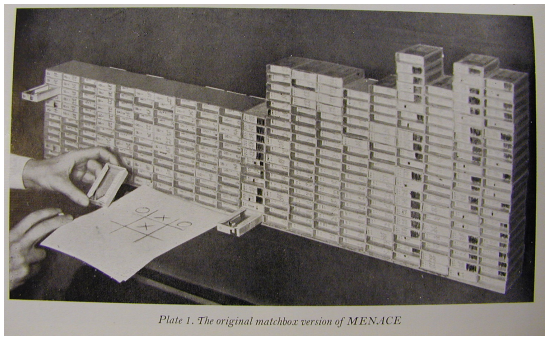


Plate 1. The original matchbox version of MENACE

Typical RL problems

- Games
- Autonomous robots
- Control of unstable systems
  - Learning to ride a bicycle
  - Auto-pilot for helicopters
- Sequential optimization problems, for example:
  - Controlling the elevators in an office building
  - Resource allocation in computer networks

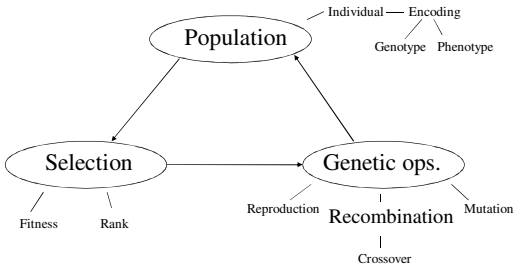
Techniques (examples)

- Artificial neural networks (ANNs)
  - Inspired by biological nervous systems
  - E.g. Multilayer perceptrons, Self-Organizing Maps
- Reinforcement learning (RL)
  - Inspired by psychology, ethology and behaviourism
  - E.g. Menace, Q-Learning, TD(λ)
- **Evolutionary Computing (EC)**
  - Inspired by genetics, natural selection and evolution
  - E.g. Genetic algorithms, Genetic Programming
- Swarm intelligence
  - Inspired by social animals (bird flocks, ants, etc.)
  - E.g. Particle Swarm Optimization, Ant Colony Optimization, Cellular automata

Evolutionary computing

- Used for learning problems where the task is to maximize some measure of success (fitness)
- Essentially the same family of *problems* as in reinforcement learning, but the *methods* are different
- Methods inspired by genetics, natural selection and evolution
- However, the ”evolution” is controlled, so it’s more like breeding

Evolutionary loop



Genotypes

- A solution to the problem is encoded by the individual’s *genotype* (*genome*, *artificial chromosome*)
  - In genetic algorithms, a string or parameter vector (e.g. bit string)
  - In genetic programming, a computer program
  - In evolutionary programming, a representation of a state machine
  - …

Genetic Programming

- Usually operates on parse trees of computer programs
- E.g. the expression 5+3*4

In Lisp

(+ 5 (* 3 4))

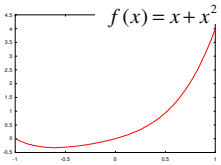
As a tree

- Crossover: Swap sub trees
- Works particularly well in Lisp!

GP example: function approximation

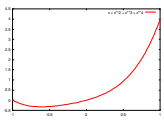
- Task: Given a training set, discover the function

$$f(x) = x + x^2 + x^3 + x^4, \text{ for } x \in [-1,1]$$



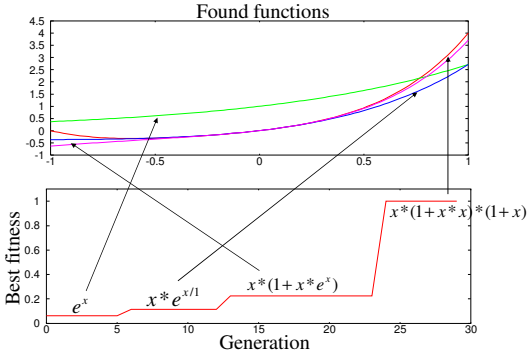
- A neural network would do a *numerical* approximation
- GP is *combinatorial* – it should be able to find the *exact* function (if given the necessary building blocks)

Implementation



- Create a population of random expressions, $y(x)$, using the functions +, –, *, /, *sin*, *cos*, *exp* and *log*, and the terminals 1 and x
- Many lures (*exp* in particular)
- Fitness: 0 if illegal expression, else $1/(d+1)$, where $d = |f(x) - y(x)|$
- This way, fitness stays in]0,1])

Test run (100 individuals)



Techniques (examples)

- Artificial neural networks (ANNs)
  - Inspired by biological nervous systems
  - E.g. Multilayer perceptrons, Self-Organizing Maps
- Reinforcement learning (RL)
  - Inspired by psychology, ethology and behaviourism
  - E.g. Menace, Q-Learning, TD(λ)
- Evolutionary Computing (EC)
  - Inspired by genetics, natural selection and evolution
  - E.g. Genetic algorithms, Genetic Programming
- **Swarm intelligence**
  - Inspired by social animals (bird flocks, ants, etc.)
  - E.g. Particle Swarm Optimization, Ant Colony Optimization, Cellular automata

Swarm Intelligence

- Bird flocks and fish schools move in a coordinated way, but there is no coordinator (leader)
  - So, what decides the behaviour of a leader-less flock?
- Ants and termites quickly find the shortest path between the nest and a food source
  - ... and solve many other advanced problems as well
    - keeping cattle, building (ventilated) housing, coordinated heavy transports, tactical warfare, cleaning house, etc.
  - A single ant is essentially a blind, memory-less, random walker!
- Distributed systems without central control
- Useful not only to simulate but also to solve optimization problems

Bird flocks and fish schools

- Local interaction
- No leader
- Simple local rules – a weighted combination of several goals
  - match velocity of your neighbours
  - avoid collisions with your neighbours
  - avoid getting too far from your neighbours
    - or strive for centre of the flock (fish)

- To simulate an insect swarm, remove the match-velocity rule
- Sufficient to make very realistic simulations of fish schools and bird flocks
  - used in movies and computer graphics

Stampede in "Lion King"

Particle Swarm Optimization

- Originally intended to simulate bird flocks and to model social interaction
  - but stands on its own as an optimization tool
- A population of *particles*
  - Population size, typically 10-50 (smaller than in EC)
- A particle, i, has a position, $\boldsymbol{x}_i$, and a velocity, $\boldsymbol{v}_i$
  - Both vectors in n-dimensional space
- Each particle's position, $\boldsymbol{x}_i$, represents one solution to the problem
- Each particle remembers the best position it has found, so far, $\boldsymbol{p}_i$

The flying particle

- The particles "fly" through n-dimensional space, in search for the best solution

$$x_i(t) = x_i(t-1) + v_i(t)$$

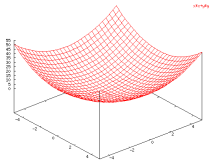
- The velocities, $\boldsymbol{v}$, depend on previous experience of this particle and that of its neighbours

$$v_{id}(t) = v_{id}(t-1) + \underbrace{U(0,\varphi_1)*(p_{id} - x_{id}(t-1))}_{\text{Cognitive component}} + \underbrace{U(0,\varphi_2)*(p_{ld} - x_{id}(t-1))}_{\text{Social component}}$$

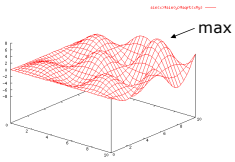
Simulation in 2D

Lbest with φ_1=1.8, φ_2=2.3

- Nhood: the 2 immediate neighbours
- V_{max} = range/25



Bowl

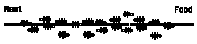


Alpine 2D

What about the ants?

- How do they find the shortest route?
  - They don't (not the individual ants, that is)
  - The colony does!
- Ant colonies are much more intelligent than ants
  - Ant colonies adapt, ants don't (much)
  - Ants have almost no memory and can not build cognitive maps. Ant colonies can (and do)
    - Mammals build cognitive maps in their brains
    - Ant colonies build them in their environment, through pheromone trails
- Ants are better thought of as cells in a greater organism – the colony
  - Also without leader – the queen is not a controller

Ants find shortest paths



Ant path

An obstacle appears

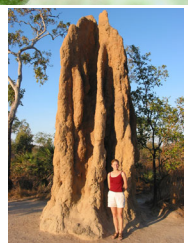
At first, the ants select at random

After a while, pheromones become more concentrated on the shortest route

Drawings by Marco Dorigo

Stigmergy

Indirect communication and coordination, by local modification and sensing of the environment



Walls, tunnels and bridges

Ant Colony Optimization

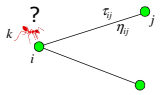
- Family of combinatorial optimization algorithms, based on ant behaviour
- Common benchmark: the Travelling Salesman Problem (TSP)
- Common 'real' applications
  - Scheduling and
  - Network routing (AntNet)
- Members: ACS, Ant-Q, MMAS, AS_{rank}, ...
  - most of which are extensions to Dorigo's Ant System (AS)

Ant System for TSP

Each ant (k)

- is placed in a randomly selected city
- remembers the partial solution found so far (initially, the start city only)
- moves stochastically from city (i) to city (j), by some transition probability

$$p_{ij}^{k}(t)$$



which depends on

- pheromone intensity, τ_{ij}
- local information, η_{ij} (distance)
- whether j is feasible (not already visited)

Ant System TSP Demo

- 20 cities (19!/2 = $6.1*10^{16}$ possible tours)
- 20 ants (one in each city)
- α=β=1
- Evaporation rate, ρ=0.9

Cellular automata

- Massively parallel system of identical communicating state machines (cells)
- A cell's *state* (e.g. on/off) is a function of the states of it communicates with (its neighbours)
  - The neighbourhood is usually topolocial
- Used to model/animate fluids *(Find Nemo)*, gases, bacterial growth, swaying grass (*Shreck?*), social interaction, epidemics, in ecological simulations etc.

Conway's Game of Life

- World: a 2D grid. Each square represents a cell
- States: Living or dead
- Neighbourhood: The eight surrounding cells
- Initialize with a random number of living cells
- State transition rules:
  - A living cell with <2 living neighbours dies (loneliness)
  - A living cell with >3 living neighbours dies (overcrowded)
  - A dead cell with exactly 3 living neighbours comes alive
  - All other cells keep their current state

Life demo