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

inductive logic programming, association rule learning, decision tree learning, ...

Machine Learning

Natural computation

neural networks, genetic algorithms, swarm intelligence, reinforcement learning ...

cellular automata ...
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

**Supervised**
- Input
- Target function
- Learning system
- Error

**Reinforcement**
- Environment
- State
- Reward
- Agent
- Action
- Learning system
- Suggested actions
- Action selector

**Unsupervised**

Institutionen för informationsteknologi | www.it.uu.se
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

$$E = \frac{\partial E}{\partial w_{ji}}$$

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.
Deep Learning

- A very deep multilayer structure
- Many layers → Back propagation ineffective
- Layers are more sparsely connected and often trained successively, to build up a sequence of progressively more complex feature extractors
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
Early neurocomputers
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($\lambda$)

- **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)
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(\(\lambda\))

- 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 computing

Population

Selection
- Fitness
- Rank

Genetic ops.
- Reproduction
- Recombination
- Mutation
- Crossover

Individual — Encoding
- Genotype
- Phenotype

Individual

Encoding

Genotype

Phenotype
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
  - ...

Applications

- Automated design (circuitry, antennae, lenses, turbines, ...)
- Industrial process control
- Chemistry and Bioinformatics (e.g. predicting RNA structures)
- Games (racing games in particular)
- Scheduling
- Neural 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($\lambda$)

- **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
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

At first, the ants select at random

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

Drawings by Marco Dorigo

Olle Gällmo | olle.gallmo@it.uu.se
Stigmergy

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

![Life demo image]
Machine Learning course

- Spring semester (10 credits)
- Natural computation algorithms
  - all the above, and more
  - emphasis on neural networks
- Flexible choice of end project