Machine learning and robotics

Lecture notes, INF3480

Kristian Nymoen

Machine Learning

Machine Learning (ML):

Programming computers to generate rules based on example data

Some applications of ML in robotics:

- Learning complicated dynamics
- Evolving locomotion patterns
- Adapting to changing conditions in the robot's surroundings

Machine Learning?

Some techniques:

- Artificial Neural Networks
- Evolutionary Algorithms
- Reinforcement Learning

Artificial neural networks

- Multiple inputs are passed to a "hidden" layer, and then to an output layer
- Learning by adapting weights for the different nodes in the network.

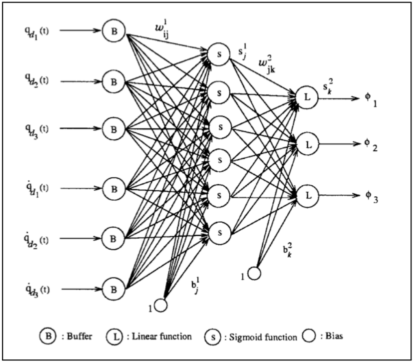


Figure borrowed from: Jung and Hsia (2000): "Neural network inverse control techniques for PD controlled robot manipulator" in *Robotica* vol. 18, pp. 305314.

Artificial neural networks

An artificial neural network can be used to learn the dynamics and friction of a robot system.

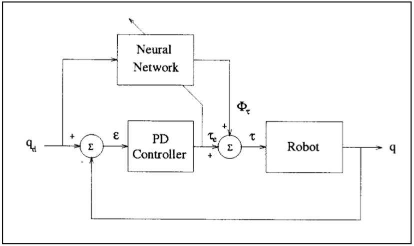
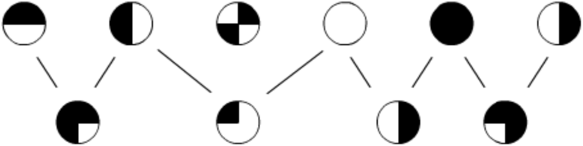


Figure borrowed from: Jung and Hsia (2000): "Neural network inverse control techniques for PD controlled robot manipulator" in *Robotica* vol. 18, pp. 305314.

Evolutionary algorithms - Inspiration from biology

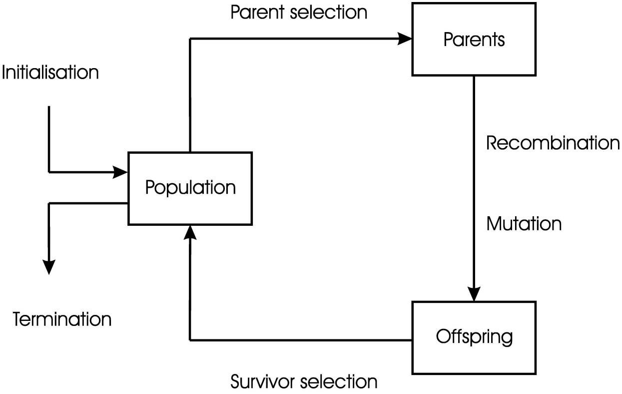


Evolution in biology: some individuals are more fit than others and will be able to reproduce. Properties of the parents can be found in the offspring. In this way, species in nature evolve into more fit species and adapt to changing conditions such as climate changes etc.

By implementing the same principle in computer algorithms, we can find near-optimal solutions to difficult problems.

Evolutionary algorithms - Overview

Overview of an evoltionary algorithm



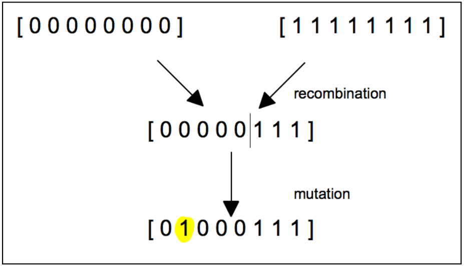
Evolutionary algorithms - Overview (contd.)

Pseudocode for a typical genetic algorithm

```
BEGIN
    INITIALIZE population with random solutions
    EVALUATE each candidate
    REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
        1 SELECT parents
        2 RECOMBINE pairs of parents
        3 MUTATE the resulting offspring
        4 EVALUATE new candidates
        5 SELECT individuals for the next generation
    OD
END!
```

Evolutionary algorithms - Operators

The most common operators in an evolutionary algorithm are *Recombination* and *Mutation*



Recombination

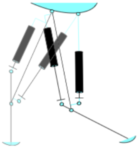
Combines two "parents" to create an "offspring".
The offspring has properties from both parents.

Mutation

A (usually small) change in the chromosome.
For instance by flipping a bit from 0 to 1.

Example 1: Henriette - The chicken robot

"Henriette"



http://www.youtube.com/watch?v=mXpz5khMY2c

Henriette - The chicken robot (contd.)

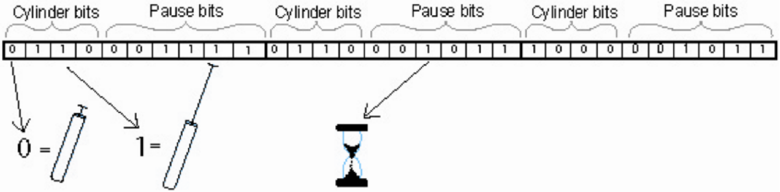


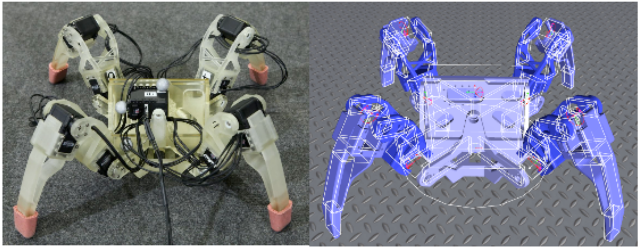
Figure: The "Henriette" chromosome

- Walking pattern coded into bit strings.
- 3 "states" consisting of leg configuration and pause length
- An evolutionary algorithm was used to evolve the leg configurations and the pause length.
- For each leg configuration, 4 bits denote the position of 4 actuators, 6 bits denote the length of the pause.
- Total chromosome length: 30 bits

Example 2: Mars Rover

Able to reconfigure itself, if for instance a wheel is damaged in the landing process.

Example 3: Quadrabot

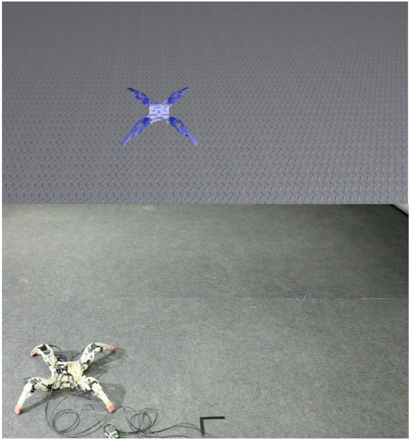


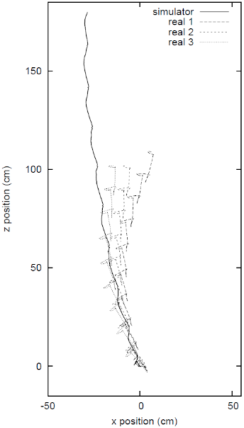
3D printed parts
AX12/18 servos
Silicone rubber socks

NVIDIA PhysX
Revolute motor joints
Rigid bodies (boxes)

Using a simulator to evolve gaits (walking patterns).

Quadrabot (contd.)





Reality gap: The evolved pattern is not necessarily as good on the real robot as in the simulation.

Reinforcement learning - a simple mapping example

Reinforcement learning can (among other things) be used for *mapping* in robotics.

An autonomous robot must be able to construct a *map* or floor plan and to localize itself in it.

An example of so-called Q-learning follows...

Mapping through reinforcement learning (contd.)

- In a simple 2 x 3 room, a robot tries out different paths to create a map of its surroundings.
- The figure (top) shows the 6 different states (positions) that the robot can be in.
- The table is a so-called *Q-table* with the scores associated with each action that can be taken in each state.

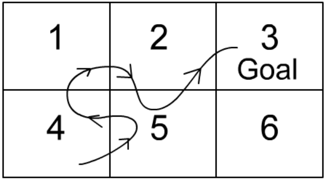
1	2	3 Goal
4	5	6

Initial Q-table

State	Move up	Move down	Move left	Move right
1)	-	0	-	0
2)	-	0	0	0
3)	-	-	-	-
4)	0	-	-	0
5)	0	-	0	0
6)	0	-	0	-

Mapping through reinforcement learning (contd.)

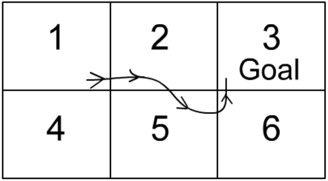
- Training starts by putting the robot in a random position (here, position 4).
- The robot follows a random trajectory, and when it reaches its goal, it receives an award of 100.
- We update the Q-table so that the last move: *move right from state 2* is awarded a score of 100.



Episode 1				
State	Move up	Move down	Move left	Move right
1)	-	0	-	0
2)	-	0	0	**0(100)**
3)	-	-	-	-
4)	0	-	-	0
5)	0	-	0	0
6)	0	-	0	-

Mapping through reinforcement learning (contd.)

- Another training session, following the path shown.
- When reaching square 2, the highest possible score in the next move is 100, so this move should be awarded, but not as high as reaching the final goal. We use a scaling factor of 0.9 compared to the highest possible score from state 2, giving a score of 90 for the move: *move right from state 1*.
- Again, the final move, here *moving up from state 6*, is awarded a score of 100.



Episode 2				
State	Move up	Move down	Move left	Move right
1)	-	0	-	**0 (90)**
2)	-	0	0	100
3)	-	-	-	-
4)	0	-	-	0
5)	0	-	0	0
6)	**0(100)**	-	0	-

Mapping through reinforcement learning (contd.)

- If this is continued, the Q-table will eventually converge towards this one.
- This represents a complete list of all possible moves for the robot in any possible state.
- By using this table the robot can always find the fastest possible way to the goal.

Complete Q-table

State	Move up	Move down	Move left	Move right
1)	-	72.9	-	90
2)	-	81	81	100
3)	-	-	-	-
4)	81	-	-	81
5)	90	-	72.9	90
6)	100	-	81	-