Swarm Intelligence Seminar: Training Recurrent Neural Network with PSO

Xiwen Cheng & Barry van Veen

LIACS

November 30, 2010

Goal

- To implement a Particle Swarm Optimizer (PSO) and use this to train a Recurrent Neural Network (RNN).
- To use the optimized RNN to control a car in a Box2D world

Implementation

- C++
- Box2D
- Using code of last year

Implementation: PSO

- Generic PSO
  - Neighbourhood: swarm (*global best*)
  - Inertia weight w: 0.8
  - Social parameters: 2.0
  - X_{max}: $[-10, 10]$
  - V_{max}: $[-1, 1]$
  - Iterations: 100
  - Particles: 5

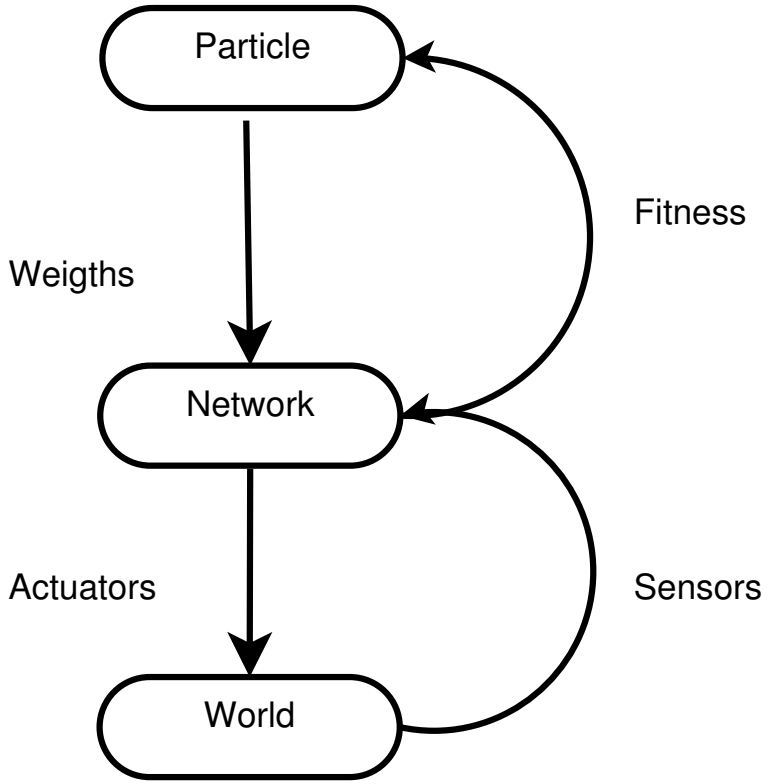
$$
\begin{aligned}
V(t+1) = \quad & w \times V(t) \\
& + c_1 \times rand_1() \times (pBest - P(t)) \\
& + c_2 \times rand_2() \times (nBest - P(t)) \\
P(t+1) = \quad & P(t) + V(t+1)
\end{aligned}
$$

Implementation: RNN

- Fully connected recurrent network
- Sigmoid activation function
- Input:
  - motor speed left wheel
  - motor speed right wheel
  - angle of the vehicle
  - linear velocity
  - angular velocity right axle
  - angular velocity left axle
  - position on y-axis
  - position on x-axis
  - position relative to starting point
- Output:
  - motor speed left wheel
  - motor speed right wheel

Implementation: putting it all together

- Evaluation of a particle
  - initialize the Box2D world and RNN
  - start loop
    - ★ extract input from Box2D
    - ★ run RNN with inputs
    - ★ simulate in Box2d
  - compute fitness (distance traveled)



Problems

- Implementation of last year
- Difficult to connect algorithms to Box2D environment
- Hard to tune parameters and quantify results

Results

- Our car can overcome obstacles in the world
- Multiple worlds tested: good results

Demo

Demo!

Further research

- Robustness
  - improve fitness function
  - NN parameter tuning
  - training setup (random initialization)
- Real time (online) training