COMP3411: Artificial Intelligence

Extension 4. Evolutionary Robotics

Outline

- Darwinian Evolution
- Evolutionary Computation
- Simulated Hockey
- Evolutionary Robotics

Charles Darwin

- Darwin’s theory of Natural Selection was largely inspired by what he observed on a visit to the Galapagos Islands
  - different species of finches from different islands
  - unusual adaptations such as the marine iguana
  - breeding habits of turtles
- Darwin was influenced by:
  - Charles Lyell’s “Principles of Geology”
  - Thomas Malthus’s “Essay on Population”
  - his grandfather Erasmus Darwin
  - his other grandfather, Josiah Wedgwood

Human Genome

- human genome consists of 3 billion DNA base pairs
- each base pair can be one of four nucleotides
  - A (Adenine)
  - G (Guanine)
  - C (Cytosine)
  - T (Thymine)
- approximately 30,000 “genes”, each coding for a specific protein
- 97% of genome does not code for proteins
  - once thought to be useless “junk” DNA
  - now thought to serve some other function(s)

Evolutionary Computation

- use principles of natural selection to evolve a computational mechanism which performs well at a specified task.
- start with randomly initialized population
- repeated cycles of:
  - evaluation
  - selection
  - reproduction + mutation
- any computational paradigm can be used, with appropriately defined reproduction and mutation operators

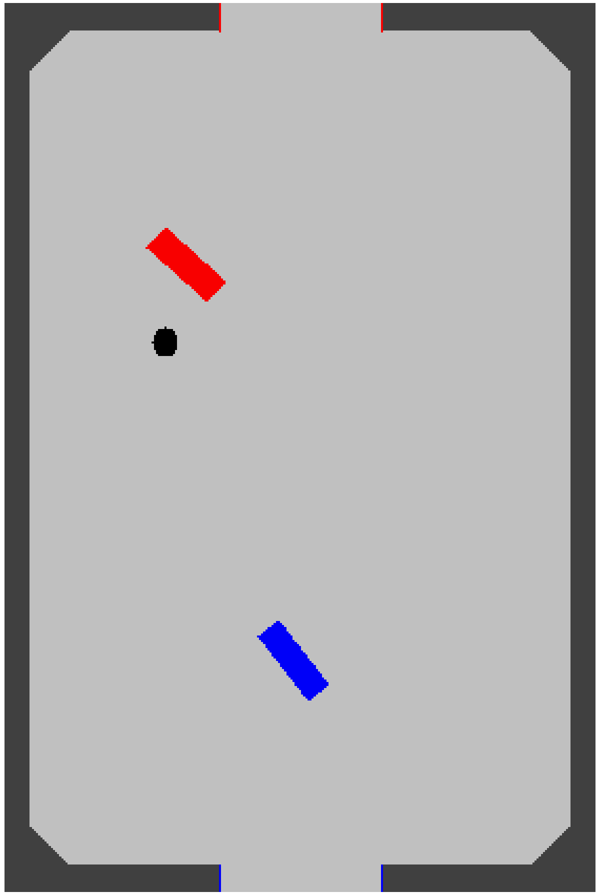
Evolutionary Computation Paradigms

- Bit Strings (Holland – “Genetic Algorithm”)
- S-expression trees (Koza – “Genetic Programming”)
- set of continuous parameters (Swefel – “Evolutionary Strategy”)
- Lindenmeyer system (e.g. Sims – “Evolving Virtual Creatures”)

Continuous Parameters (ES)

- reproduction = just copying
- mutation = add random noise to each weight (or parameter), from a Gaussian distribution with specified standard deviation
  - sometimes, the standard deviation evolves as well

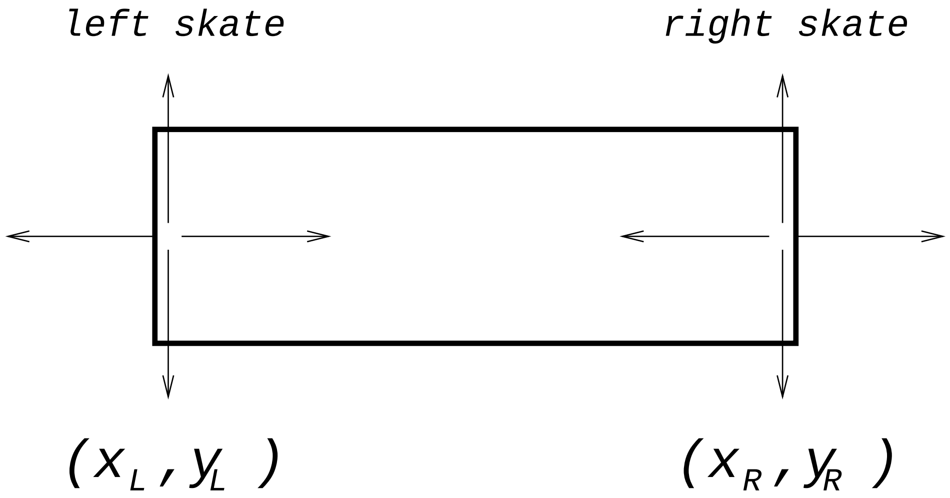
Case Study – Simulated Hockey



Shock Physics

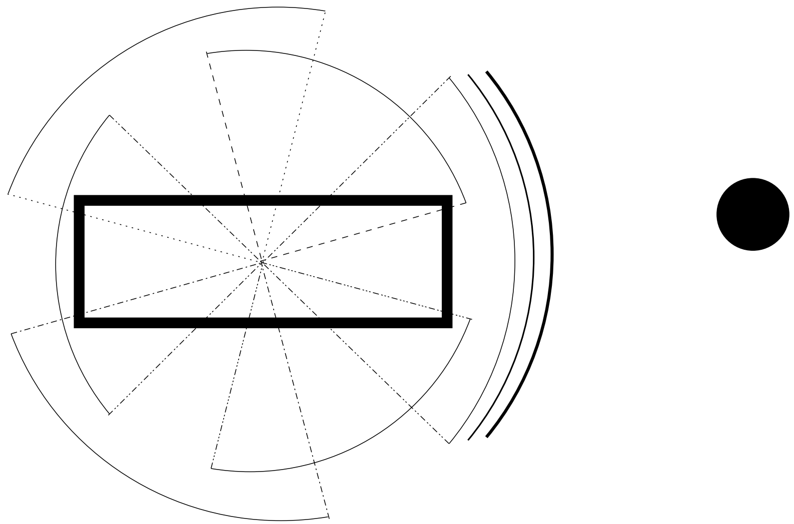
- rectangular rink with rounded corners
- near-frictionless playing surface
- “spring” method of collision handling
- frictionless puck (never acquires any spin)

Shock Actuators



- a skate at each end of the vehicle with which it can push on the rink in two independent directions

Shock Sensors

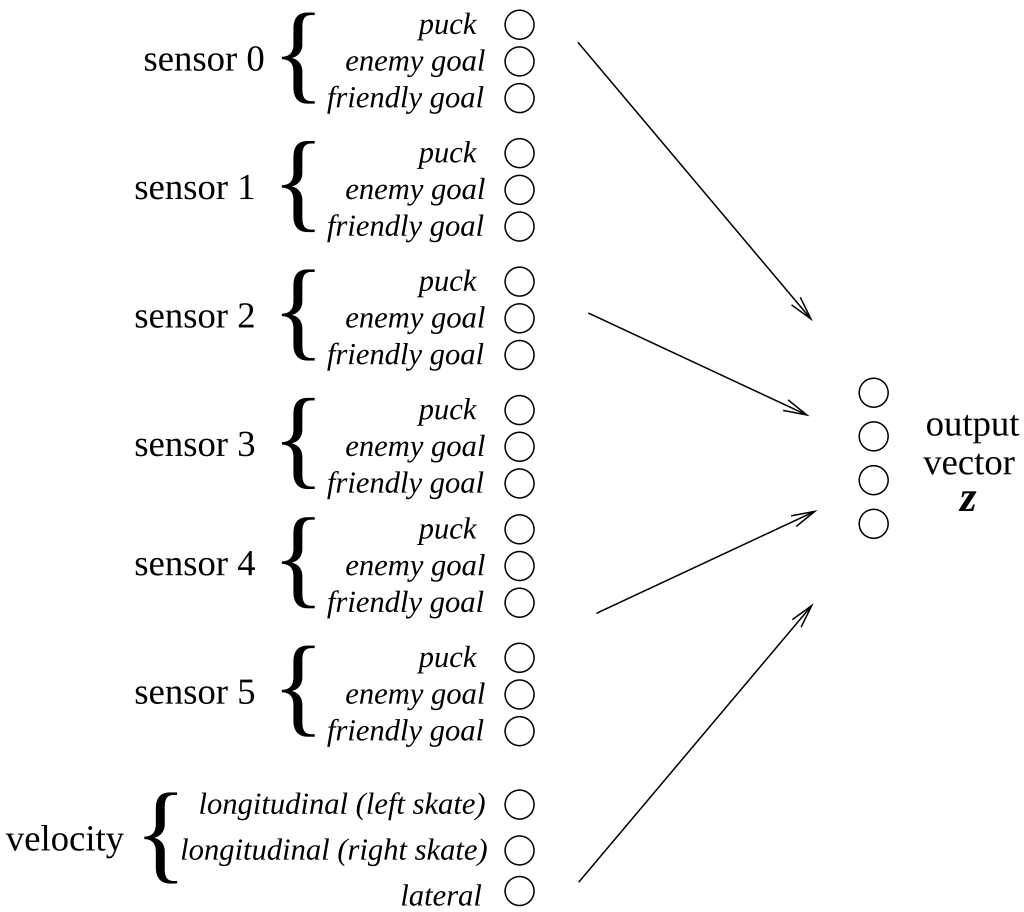


- 6 Braitenberg-style sensors equally spaced around the vehicle
- each sensor has an angular range of 90° with an overlap of 30° between neighbouring sensors

Shock Inputs

- each of the 6 sensors responds to three different stimuli
  - ball / puck
  - own goal
  - opponent goal
- 3 additional inputs specify the current velocity of the vehicle
- total of $3 \times 6 + 3 = 21$ inputs

Shock Agent


sensor 0
puck
enemy goal
friendly goal
sensor 1
puck
enemy goal
friendly goal
sensor 2
puck
enemy goal
friendly goal
sensor 3
puck
enemy goal
friendly goal
sensor 4
puck
enemy goal
friendly goal
sensor 5
puck
enemy goal
friendly goal
velocity
longitudinal (left skate)
longitudinal (right skate)
lateral
output vector
z

Shock Agent

- Perceptron with 21 inputs and 4 outputs
- total of $4 \times (21+1) = 88$ weights
- our "genome" (for Evolutionary Computation) consists of a vector of these 88 parameters
- mutation = add Gaussian random noise to each parameter, with standard deviation 0.05

Shock Task

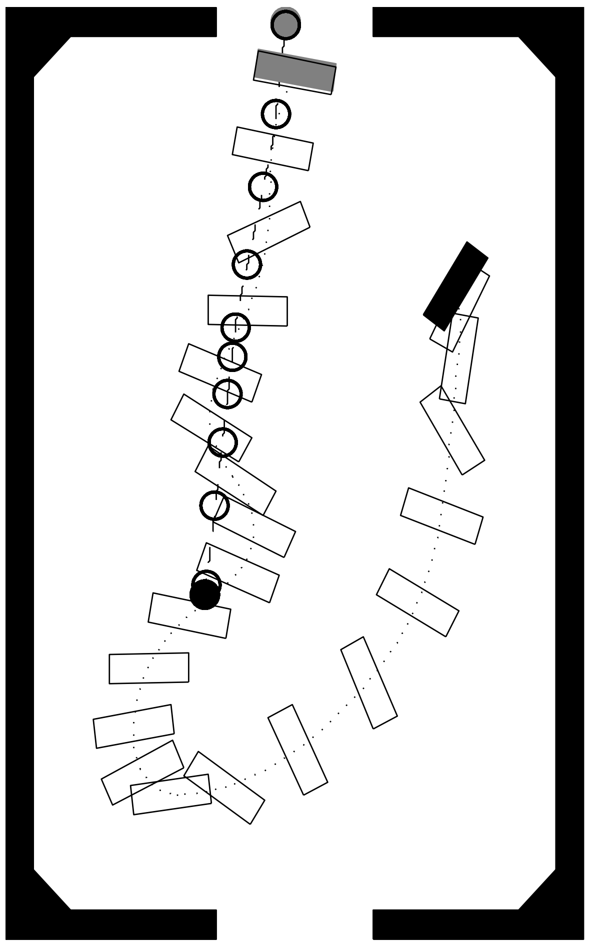
- each game begins with a random “game initial condition”
  - random position for puck
  - random position and orientation for player
- each game ends with
  - +1 if puck $\rightarrow$ enemy goal
  - -1 if puck $\rightarrow$ own goal
  - 0 if time limit expires

Evolutionary Algorithm

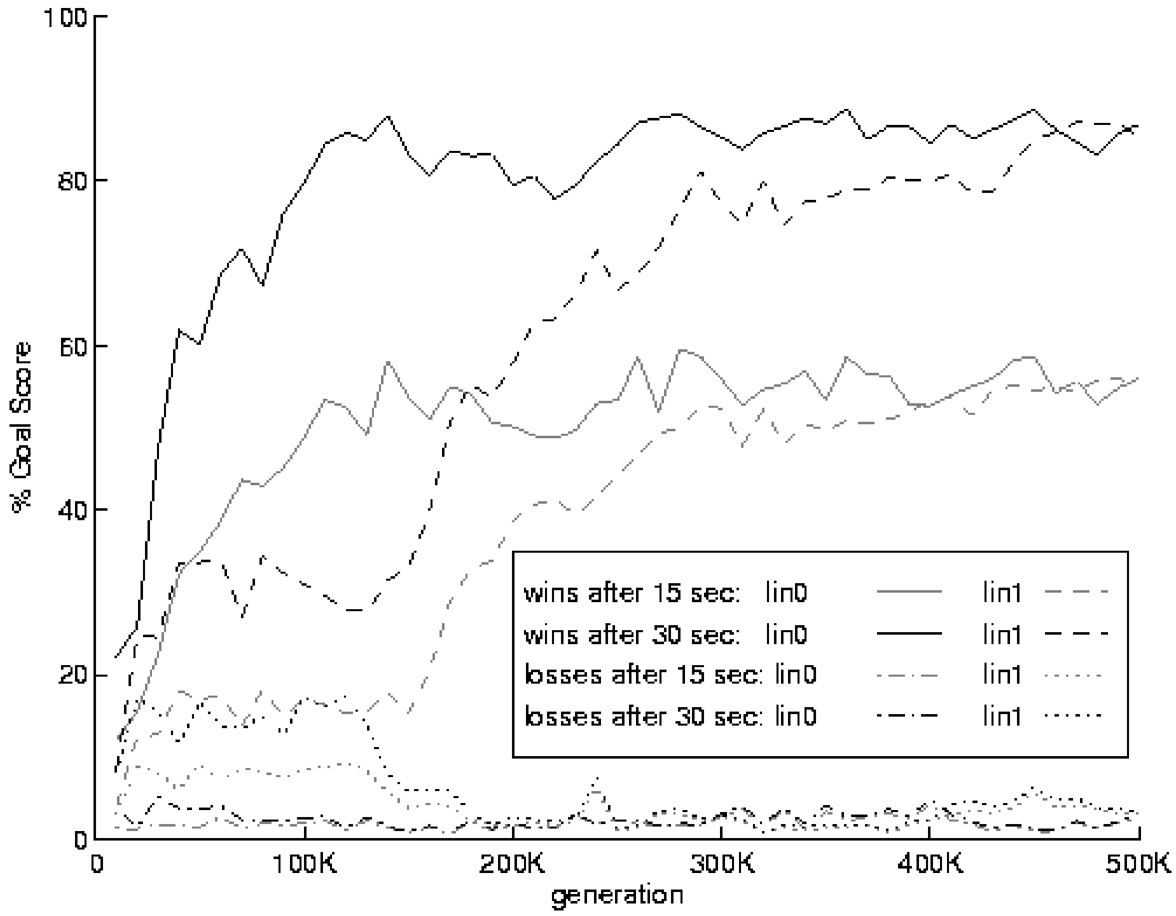
- mutant $\leftarrow$ champ + Gaussian noise
- champ and mutant play up to 5 games with same game initial conditions
- if mutant does “better” than champ,

  $$\text{champ} \leftarrow (1-\alpha) * \text{champ} + \alpha * \text{mutant}$$
- “better” means the mutant must score higher than the champ in the first game, and at least as high as the champ in each subsequent game

Evolved Behavior



Wins and Losses



Evolutionary/Variational Methods

- initialize mean $\mu = \{\mu_i\}_{1 \le i \le m}$ and standard deviation $\sigma = \{\sigma_i\}_{1 \le i \le m}$
- for each trial, collect k samples from a Gaussian distribution

$$\theta_i = \mu_i + \eta_i \sigma_i \quad \text{where} \quad \eta_i \sim \mathcal{N}(0,1)$$

- sometimes include "mirrored" samples $\overline{\theta}_i = \mu_i - \eta_i \sigma_i$
- evaluate each sample θ to compute score or "fitness" $F(\theta)$
- update mean μ by $$\mu \leftarrow \mu + \alpha(F(\theta) - \overline{F})(\theta - \mu)$$
  - α = learning rate, $\overline{F}$ = baseline
- sometimes, σ is updated as well

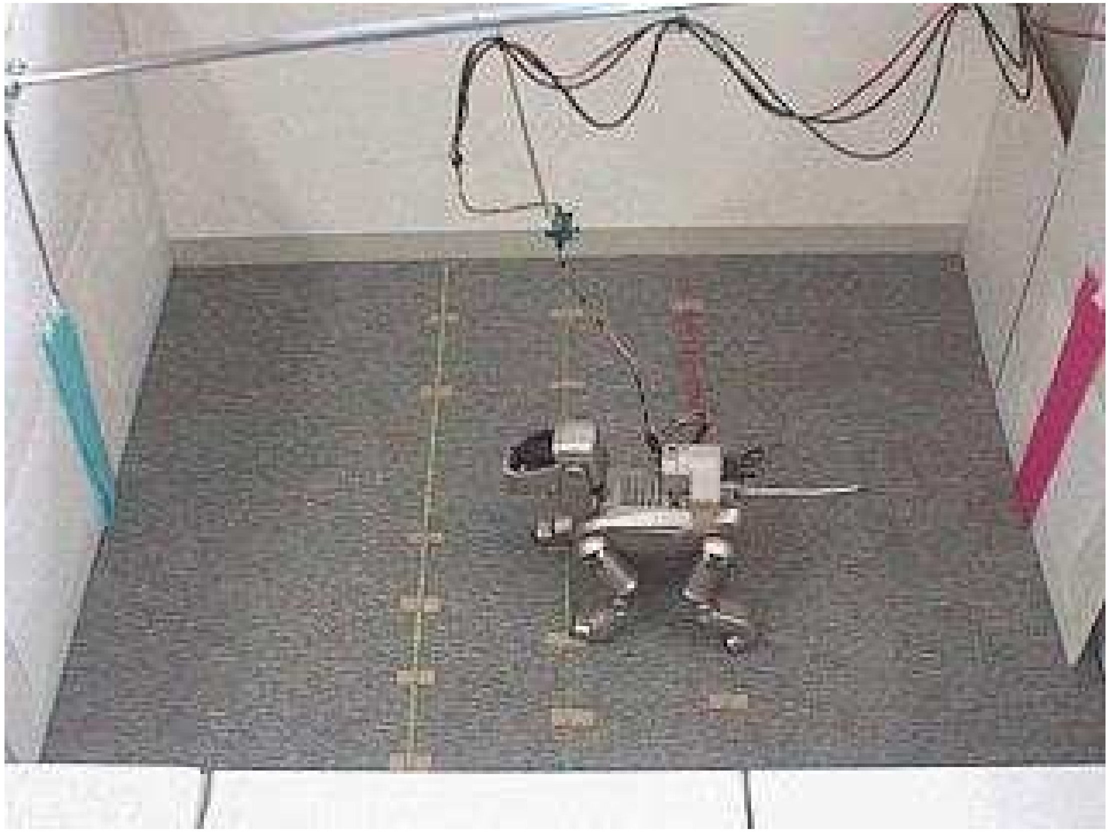
OpenAI Evolution Strategies

- Evolutionary Strategy with fixed σ
- since only μ is updated, computation can be distributed across many processors
- applied to Atari Pong, MuJoCo humanoid walking
- competitive with Deep Q-Learning on these tasks

Evolutionary Robotics

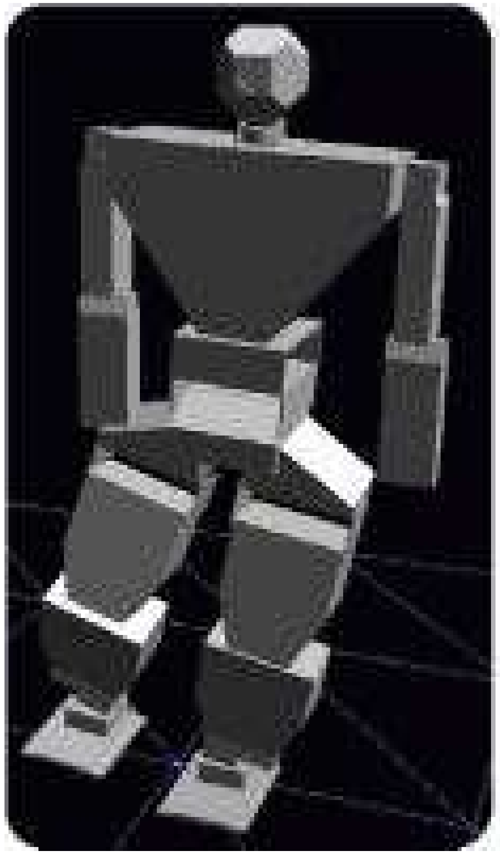
- Aibo walk learning
- Humanoid walk learning
- Evolving body as well as controller
- Simulation to Reality

Aibo Walk Learning (Hornby)



- Learning done on actual robot.

Guroo – Humanoid Walk Learning

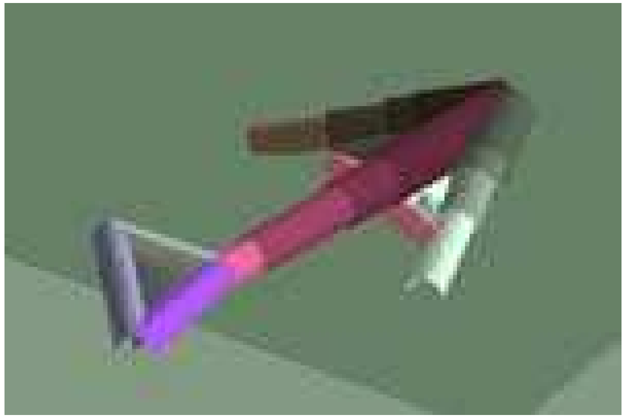


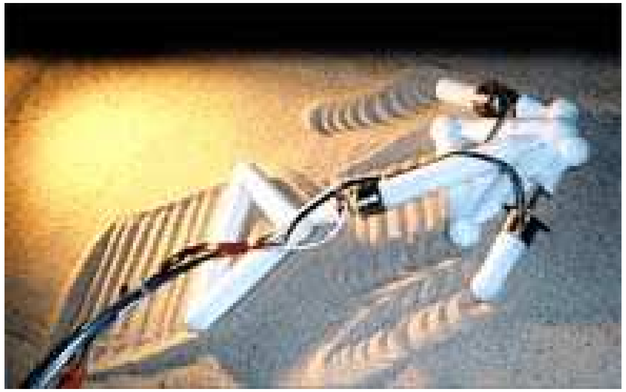
- Learning done in simulator(s), then tested on actual robot.

Evolving Virtual Creatures (Sims)

- Body evolves as a Lindenmeyer system
- Controller evolves as a neural network

Golem (Lipson)





- Evolved in simulation, tested in reality.

Evolved Antenna

One example of the use of Evolutionary Algorithms for a real world application is the antenna that was evolved by Hornby et al in 2006 for NASA's Space Technology 5 (ST5) mission.