Swarm Intelligence – W5:
Swarm Intelligence for Machine Learning:
An Introduction to Genetic Algorithms and Particle Swarm Optimization
Outline

• Machine-learning-based methods
  – Rationale for real-time, embedded systems
  – Classification and terminology

• Genetic Algorithms (GA)
  – Terminology
  – Main operators and features

• Particle Swarm Optimization (PSO)
  – Terminology
  – Main operators and features

• Comparison between GA and PSO
Rationale and Classification
Why Machine-Learning?

• Complementarity to a model-based/engineering approaches: when low-level details matter (optimization) and/or good models do not exist (design)!

• When the design/optimization space is too big (infinite)/too computationally expensive to be systematically searched

• Automatic design and optimization techniques

• Role of engineer reduced to specifying performance requirements and problem encoding
Why Machine-Learning?

• There are design and optimization techniques robust to noise, nonlinearities, discontinuities

• Individual real-time adaptation to new environmental conditions; i.e. increased individual flexibility when environmental conditions are not known/cannot predicted a priori

• Search space: parameters and/or rules
ML Techniques: Classification

– **Supervised techniques**: “a trainer/teacher” is available.
  • Ex: a set of input-output examples is provided to the system, performance error given by difference between system output and true/teacher-defined output, error fed to the system using optimization algorithm so that performance is increased over trials
  • The generality of the system after training is tested on examples not previously presented to the system (i.e. a “test set” exclusive from the “training set”)

– **Unsupervised techniques**: “trial-and-error”, “evaluative” techniques; no teacher available.
  • The system judges its performance according to a given metric (fitness function) to be optimized
  • The metrics does not refer to any specific input-to-output mapping
  • The system tries out possible design solutions, does mistakes, and tries to learn from its mistakes
  • Number of possible examples is very large, possibly infinite, and not known a priori
ML Techniques: Classification

– **Off-line**: in simulation, download the learned/evolved solution onto real hardware when certain criteria are met

– **Hybrid**: most of the time in simulation (e.g. 90%), last period (e.g. 10%) of the process on real hardware

– **On-line**: from the beginning on real hardware (no simulation). Depending on the algorithm more or less rapid
ML Techniques: Classification

ML algorithms require sometimes fairly important computational resources (in particular for multi-agent search algorithms), therefore a further classification is:

– **On-board**: machine-learning algorithm run on the system to be learned or evolved (no external unit)

– **Off-board**: the machine-learning algorithm runs off-board and the system to be learned or evolved just serves as phenotypical, embodied implementation of a candidate solution
Selected Unsupervised ML Techniques Robust to Noisy Fitness/Reinforcement Functions

• Evolutionary computation
  – Genetic Algorithms (GA)
  – Genetic Programming (GP)
  – Evolutionary Strategies (ES)
  – Particle Swarm Optimization (PSO)

• Learning
  – In-Line Adaptive Learning
  – Reinforcement Learning
Genetic Algorithms
Genetic Algorithms Inspiration

• In natural evolution, organisms adapt to their environments – better able to survive over time

• Aspects of evolution:
  – Survival of the fittest
  – Genetic combination in reproduction
  – Mutation

• Genetic Algorithms use evolutionary techniques to achieve parameter optimization
GA: Terminology

- **Population**: set of m candidate solutions (e.g. m = 100); each candidate solution can also be considered as a genetic individual endowed with a single chromosome which in turn consists of multiple genes.

- **Generation**: new population after genetic operators have been applied (n = # generations e.g. 50, 100, 1000).

- **Fitness function**: measurement of the efficacy of each candidate solution

- **Evaluation span**: evaluation period of each candidate solution during a given generation. The time cost of the evaluation span differs greatly from scenario to scenario: it can be extremely cheap (e.g., simply computing the fitness function in a benchmark function) or involve an experimental period (e.g., evaluating the performance of a given control parameter set on a robot)

- **Life span**: number of generations a candidate solution survives

- **Population manager**: applies genetic operators to generate the candidate solutions of the new generation from the current one

- **Principles**: selection (survival of the fittest), recombination, and mutation
Evolutionary Loop: Several Generations

Ex. of end criteria:
- # of generations
- best solution performance
- …
GA: Coding & Decoding

phenotype coding genotype decoding phenotype

- phenotype: usually represented by a vector of dimension D, D being the dimension of the hyperspace to search; vector components are usually real numbers in a bounded range
- genotype: chromosome = string of genotypical segments, i.e. genes, or mathematically speaking, again a vector of real or binary numbers; vector dimension varies according to coding schema (≥ D)

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>G₃</th>
<th>G₄</th>
<th>...</th>
<th>Gₙ</th>
</tr>
</thead>
</table>

\( G₁ = \text{gene} = \text{binary or real number} \)

Coding: real-to-real or real-to-binary via Gray code (minimization of nonlinear jumping between phenotype and genotype)

Decoding: inverted operation

Rem:

- Artificial evolution: usually one-to-one mapping between phenotypic and genotypic space
- Natural evolution: 1 gene codes for several functions, 1 function coded by several genes.
GA: Basic Operators

- **Selection**: roulette wheel (selection probability determined by normalized fitness), ranked selection (selection probability determined by fitness order), elitist selection (highest fitness individuals always selected)
- **Crossover**: 1 point, 2 points (e.g. $p_{\text{crossover}} = 0.2$)

![Crossover Example]

- **Mutation** (e.g. $p_{\text{mutation}} = 0.05$)

![Mutation Example]

**Note**: examples for fixed-length chromosomes!
GA: Discrete vs Continuous

• For default GA, all parameters discrete (e.g., binary bits, choice index)

• Common adaptation for continuous optimization:
  – Parameters are real values
  – Mutation: apply randomized adjustment to gene value (i.e. $G_i' = G_i + m$) instead of replacing value

• Selection of adjustment range affects optimization progress
Particle Swarm Optimization
Reynolds’ Rules for Flocking

1. Separation: avoid collisions with nearby flockmates

2. Alignment: attempt to match velocity (speed and direction) with nearby flockmates

3. Cohesion: attempt to stay close to nearby flockmates

More on Week 7
PSO: Terminology

- **Population**: set of candidate solutions tested in one time step, consists of \( m \) particles (e.g., \( m = 20 \))

- **Particle**: represents a candidate solution; it is characterized by a velocity vector \( \mathbf{v} \) and a position vector \( \mathbf{x} \) in the hyperspace of dimension \( D \)

- **Evaluation span**: evaluation period of each candidate solution during one a time step; as in GA the evaluation span might take more or less time depending on the experimental scenario.

- **Fitness function**: measurement of efficacy of a given candidate solution during the evaluation span

- **Population manager**: update velocities and position for each particle according to the main PSO loop

- **Principles**: imitate, evaluate, compare
Evolutionary Loop: Several Generations

Start

Initialize particles

Perform main PSO loop

End criterion met?

Ex. of end criteria:
- # of time steps
- best solution performance
- ...

End
Initialization: Positions and Velocities
The Main PSO Loop – Parameters and Variables

• Functions
  – \text{rand}() = \text{uniformly distributed random number in } [0,1]

• Parameters
  – \( w \): velocity inertia (positive scalar)
  – \( c_p \): personal best coefficient/weight (positive scalar)
  – \( c_n \): neighborhood best coefficient/weight (positive scalar)

• Variables
  – \( x_{ij}(t) \): position of particle \( i \) in the \( j \)-th dimension at time step \( t \) (\( j = [1,D] \))
  – \( v_{ij}(t) \): velocity particle \( i \) in the \( j \)-th dimension at time step \( t \)
  – \( x_{ij}^*(t) \): position of particle \( i \) in the \( j \)-th dimension with maximal fitness up to iteration \( t \)
  – \( x_{ij}'(t) \): position of particle \( i' \) in the \( j \)-th dimension having achieved the maximal fitness up to iteration \( t \) in the neighborhood of particle \( i \)
The Main PSO Loop

At each time step \( t \)
for each particle \( i \)
for each component \( j \)

update the velocity

\[
v_{ij}(t+1) = w v_{ij}(t) + c_p \text{rand}() (x_{ij}^* - x_{ij}) + c_n \text{rand}() (x_{i,j}^* - x_{ij})
\]

then move

\[
x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)
\]
The main PSO Loop
- Vector Visualization

Here I am!

My position for optimal fitness up to date

The position with optimal fitness of my neighbors up to date

p-best

n-best
Neighborhoods Types

- **Size:**
  - Neighborhood index considers also the particle itself in the counting
  - **Local:** only \( k \) neighbors considered over \( m \) particles in the population \((1 < k < m)\); \( k=1 \) means no information from other particles used in velocity update
  - **Global:** \( m \) neighbors

- **Topology:**
  - Geographical
  - Social
  - Indexed
  - Random
  - ...
Neighborhood Examples: Geographical vs. Social
Neighborhood Example: Indexed and Circular (lbest)

Particle 1’s 3-neighbourhood

Virtual circle
PSO Animated Illustration

Global optimum

© M. Clerc
## GA vs. PSO - Qualitative

<table>
<thead>
<tr>
<th>Parameter/function</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>General features</td>
<td>Multi-agent, probabilistic search</td>
<td>Multi-agent, probabilistic search</td>
</tr>
<tr>
<td>Individual memory</td>
<td>no</td>
<td>yes (randomized hill climbing)</td>
</tr>
<tr>
<td>Individual operators</td>
<td>mutation</td>
<td>personal best position history, velocity inertia</td>
</tr>
<tr>
<td>Social operators</td>
<td>selection, crossover</td>
<td>neighborhood best position history</td>
</tr>
<tr>
<td>Particle’s variables (tracked by the population manager)</td>
<td>position</td>
<td>position and velocity</td>
</tr>
</tbody>
</table>
# GA vs. PSO - Qualitative

<table>
<thead>
<tr>
<th>Parameter/function</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td># of algorithmic parameters (basic)</td>
<td>$p_m, p_c, \text{selection par.}$ (1), $m$, position range (1) = 5</td>
<td>$w, c_n, c_p, k, m$, position and velocity range (2) = 7</td>
</tr>
<tr>
<td>Population diversity</td>
<td>Somehow tunable via $p_c/p_m$ ratio and selection schema</td>
<td>Mainly via local neighborhood</td>
</tr>
<tr>
<td>Global/local search balance</td>
<td>Somehow tunable via $p_c/p_m$ ratio and selection schema</td>
<td>Tunable with $w$ ($w^{\uparrow}$ → global search; $w^{\downarrow}$ → local search)</td>
</tr>
<tr>
<td>Particle’s variables (tracked by the population manager)</td>
<td>position</td>
<td>position and velocity</td>
</tr>
</tbody>
</table>
## GA vs. PSO - Quantitative

- **Goal:** minimization of a given $f(x)$
- **Standard benchmark functions with thirty terms ($n = 30$) and a fixed number of iterations**
- **All $x_i$ constrained to $[-5.12, 5.12]$**

### Standard Benchmark Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>$f_1(\bar{x}) = \sum_{i=1}^{n} x_i^2$</td>
<td>400</td>
</tr>
<tr>
<td>Generalized Rosenbrock</td>
<td>$f_2(\bar{x}) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$</td>
<td>20000</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>$f_3(\bar{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$</td>
<td>400</td>
</tr>
<tr>
<td>Griewank</td>
<td>$f_4(\bar{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$</td>
<td>400</td>
</tr>
</tbody>
</table>
GA vs. PSO - Quantitative

- **GA**: Roulette Wheel for selection, mutation applies numerical adjustment to gene
- **PSO**: lbest ring topology with neighborhood of size 3
- Algorithm parameters used (but not thoroughly optimized):

<table>
<thead>
<tr>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>Swarm Size</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>Personal Best Attraction</td>
</tr>
<tr>
<td>0.6</td>
<td>2.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>Neighborhood Best Attraction</td>
</tr>
<tr>
<td>0.05</td>
<td>2.0</td>
</tr>
<tr>
<td>Mutation Range</td>
<td>Inertia Factor</td>
</tr>
<tr>
<td>[-0.5, 0.5]</td>
<td>0.6</td>
</tr>
</tbody>
</table>
# GA vs. PSO - Quantitative

**Bold: best results; 30 runs; no noise on the performance function**

<table>
<thead>
<tr>
<th>Function (no noise)</th>
<th>GA (mean ± std dev)</th>
<th>PSO (mean ± std dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>0.02 ± 0.01</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Generalized Rosenbrock</td>
<td>34.6 ± 18.9</td>
<td>7.38 ± 3.27</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>157 ±21.8</td>
<td>48.3 ± 14.4</td>
</tr>
<tr>
<td>Griewank</td>
<td>0.01 ± 0.01</td>
<td>0.01 ± 0.03</td>
</tr>
</tbody>
</table>
GA vs. PSO – Overview

• According to most recent research, PSO outperforms GA on most (but not all!) continuous optimization problems

• No-Free-Lunch Theorem

• GA still much more widely used in general research community

• Because of random aspects, very difficult to analyze either metaheuristic or make guarantees about performance
Conclusion
Take Home Messages

• A key difference in machine-learning is supervised vs. unsupervised techniques

• Unsupervised techniques are key for robotic learning

• Two robust multi-agent probabilistic search techniques are GA and PSO

• They share some similarities and some fundamental differences

• PSO is a younger technique than GA but extremely promising; it has been invented by the swarm intelligence community
Books


