Evolutionary Programming


EP Overview

- Developed: USA in the 1960's
- Early names: D. Fogel
- Typically applied to:
  - traditional EP: machine learning tasks by finite state machines
  - contemporary EP: (numerical) optimization
- Main characteristics:
  - open framework: any representation and mutation operators
  - crossbred with ES (contemporary EP)
  - consequently: hard to say what is “standard” EP
- Special feature:
  - no recombination
  - self-adaptation of parameters in contemporary EP

EP: Main Features

<table>
<thead>
<tr>
<th>Representation</th>
<th>Real-valued vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination</td>
<td>None</td>
</tr>
<tr>
<td>Mutation</td>
<td>Gaussian perturbation</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Deterministic</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>Probabilistic ($\mu + \lambda$)</td>
</tr>
<tr>
<td>Special feature</td>
<td>Self-adaptation of mutation step sizes (in meta-EP)</td>
</tr>
</tbody>
</table>

Survivor Selection

- Recall ($\mu + \lambda$) selection, where offspring and parents participate in the selection
- What is ($\mu + \mu$) selection?

EP: Historical Perspective

- EP aimed at achieving intelligence
- Intelligence was viewed as adaptive behavior
- Prediction of the environment was considered a prerequisite to adaptive behavior
- Thus capability to predict is key to intelligence

Prediction by Finite State Machines

- Finite state machine (FSM):
  - States $S$
  - Inputs $I$
  - Outputs $O$
  - Transition function $\delta : S \times I \rightarrow S \times O$
  - Transforms input stream into output stream
- Can be used for predictions, e.g., to predict next input symbol in a sequence
FSM Example

- Consider the FSM with:
  - States $S = \{A, B, C\}$
  - $I = \{0, 1\}$
  - $O = \{a, b, c\}$
  - Transition function $\delta$ given by the diagram

FSM as a Predictor

- Consider the following FSM
- Task: predict next input
- Quality: % of $in_{i+1} = out_i$
- Given initial state C
- Input sequence 01101
- Leads to output 11011
- Quality: 3 out of 6 (50%)

Example: Evolving FSMs to Predict Primes

- $P(n) = 1$, if $n$ is prime, 0 otherwise
- $I = \mathbb{N} = \{1, 2, 3, \ldots, n, \ldots\}$
- $O = \{0, 1\}$
- Correct prediction: $Out_i = P(in_{i+1})$
- Fitness function:
  - 1 point for correct prediction of the next input
  - 0 point for incorrect prediction
  - Penalty for “too many” states

Example: Evolving FSMs to Predict Primes

- Parent selection: each FSM is mutated once
- Mutation operators (one selected randomly):
  - Change an output symbol
  - Change a state transition (i.e., redirect edge)
  - Add a state
  - Delete a state
  - Change the initial state
- Survivor selection: $(\mu + \mu)$
- Results: over-fitting, after 202 inputs best FSM had one state and both outputs were 0, i.e., it always predicted “not prime”

Modern EP

- In general no predefined representation
- Thus the mutation must match the representation
- Often applies self-adaptation of mutation parameters
- Next one EP variant is presented, not the canonical EP

Representation

- For continuous parameter optimization
- Chromosomes consist of two parts:
  - Variables: $x_1, \ldots, x_n$
  - Mutation step sizes: $\sigma_1, \ldots, \sigma_n$
- Full size representation: $\langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n \rangle$
Mutation
- Chromosomes: $\langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot (1 + \alpha \cdot N(0,1))$
- $x'_i = x_i + \sigma'_i \cdot N(0,1)$
- $\alpha \approx 0.2$
- Boundary rule: $\sigma' < \epsilon_0 \Rightarrow \sigma' = \epsilon_0$
- Other variants proposed and tried:
  - Lognormal scheme as in ES
  - Using variance instead of standard deviation
  - Mutate $\sigma$-last
  - Other distributions, e.g., Cauchy instead of Gaussian

Recombination
- None
- Rationale: one point in the search space stands for a specie, not for an individual and there can be no crossover between species
- Much historical debate "mutation vs. crossover"
- Pragmatic approach seems to prevail today

Survivor Selection
- Each individual creates one child by mutation
- $P(t): \mu$ parents, $P'(t): \mu$ offspring
- Pair-wise competition in round-robin format:
  - Each solution $x$ from $P(t) \cup P'(t)$ is evaluated against $q$ other randomly chosen solutions
  - For each comparison, a "win" is assigned, if $x$ is better than its opponent
  - The $\mu$ solutions with the greatest number of wins are retained to be parents of the next generation
- Parameter $q$ allows tuning selection pressure
- Typically $q = 10$

Example Application: The Ackley Function (Bäck et al. '93)
- The Ackley function (here used with $n = 30$):
  \[ f(x) = -20 \cdot \exp \left( -0.2 \cdot \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e \]
- Representation:
  - $-30 < x_i < 30$ (coincidence of 30's!)
  - 30 variances as step sizes
- Mutation with changing object variables first!
- Population size $\mu = 200$, selection with $q = 10$
- Termination: after 200,000 fitness evaluations
- Results: the average best solution is $1.4 \cdot 10^{-2}$

Example Application: Evolving Checkers Players (Fogel '02)
- Neural nets for evaluating future values of moves are evolved
- NNs have fixed structure with 5046 weights, these are evolved + one weight for "kings"
- Representation:
  - vector of 5046 real numbers for variables (weights)
  - vector of 5046 real numbers for $\sigma$'s
- Mutation:
  - Gaussian, lognormal scheme with $\sigma$-first
  - Plus special mechanism for the kings' weight
- Population size 15

Example Application: Evolving Checkers Players (Fogel '02)
- Tournament size $q = 5$
- Programs (with NN inside) play against other programs, no human trainer or hard-wired intelligence
- After 840 generation (6 months!) best strategy was tested against humans via Internet
- Program earned "expert class" ranking, outperforming 99.61% of all rated players