Evolution Strategies*

Jason Brownlee
jasonb@CleverAlgorithms.com
The Clever Algorithms Project
http://www.CleverAlgorithms.com

March 26, 2010
Technical Report: CA-TR-20100326-1

Abstract

The Clever Algorithms project aims to describe a large number of Artificial Intelligence algorithms in a complete, consistent, and centralized manner, to improve their general accessibility. The project makes use of a standardized algorithm description template that uses well-defined topics that motivate the collection of specific and useful information about each algorithm described. This report described the Evolution Strategies algorithm using the standardized template.

Keywords: `Clever, Algorithms, Description, Optimization, Evolution, Strategies`

1 Introduction

The Clever Algorithms project aims to describe a large number of algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics in a complete, consistent and centralized manner [4]. The project requires all algorithms to be described using a standardized template that includes a fixed number of sections, each of which is motivated by the presentation of specific information about the technique [5]. This report describes the Evolution Strategies algorithm using the standardized template.

2 Name

Evolution Strategies, Evolution Strategy, Evolutionary Strategies, ES

3 Taxonomy

Evolution Strategies is a global optimization algorithm and is an instance of an Evolutionary Algorithm from the field of Evolutionary Computation. Evolution Strategies is a sibling technique to other Evolutionary Algorithms such as Genetic Algorithms, Genetic Programming, Learning Classifier Systems, and Evolutionary Programming. A popular descendant of the Evolution Strategy algorithm is the Covariance Matrix Adaptation Evolution Strategy.

*© Copyright 2010 Jason Brownlee. Some Rights Reserved. This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 2.5 Australia License.

4 Inspiration

Evolution Strategies is inspired by the theory of evolution by means of natural selection. Specifically, the technique is inspired by macro-level or the species-level process of evolution (phenotype, hereditary, variation) and is not concerned with the genetic mechanisms of evolution (genome, chromosomes, genes, alleles).

5 Metaphor

Evolution Strategies only briefly flirted with explanation via metaphor, and is less preferred to grounded probabilistic explanations.

6 Strategy

The objective of the Evolution Strategies algorithm is to maximize the suitability of collection of candidate solutions in the context of an objective function from a domain. The objective was classically achieved through the adoption of dynamic variation, a surrogate for descent with modification, where the amount of variation was adapted dynamically with performance-based heuristics. Contemporary approaches co-adapt parameters that control the amount and bias of variation with the candidate solutions.

7 Procedure

Instances of Evolution Strategy algorithms may be concisely described with a custom terminology in the form $(\mu, \lambda) - ES$, where μ is number of candidate solution in the parent generation, and λ is the number of candidate solutions generated from and replace the parent generation. In addition to the so-called comma-selection Evolution Strategy, a plus-selection variation may be defined $(\mu + \lambda) - ES$, where the best members of the union of the μ and λ generations complete based on objective fitness for a position in the next generation. The simplest configuration is the $(1+1) - ES$ which is a type of greedy hill climbing algorithm. Algorithm 1 provides a pseudo-code listing of the $(\mu + \lambda) - ES$ Evolution Strategy algorithm for minimizing a cost function. The algorithm shows the adaptation of candidate solutions that co-adapt their own strategy parameters that influence the amount of mutation applied to a candidate solutions descendants.

8 Heuristics

- Evolution Strategies uses problem specific representations, such as real values for continuous function optimization.
- The algorithm is commonly configured such that $1 < \mu < \lambda < \infty$.
- The ratio of μ to λ influences the amount of selection pressure (greediness) exerted by the algorithm.
- A contemporary update to the algorithms notation includes a ρ as $(\mu/\rho, \lambda) - ES$ that specifies the number of parents that will contribute to each new candidate solution using a recombination operator.
- A classical rule used to govern the amount of mutation (standard deviation used in mutation for continuous function optimization) was the $\frac{1}{5}$-rule, where the ratio of successful

Algorithm 1: Pseudo Code for the Evolution Strategies algorithm.

Input: μ, λ, ProblemSize
Output: S_{best}

```
Population ← InitializePopulation(μ, ProblemSize);
EvaluatePopulation(Population);
S_best ← GetBest(Population, 1);
while ¬StopCondition() do
    Children ← 0;
    i ← 0;
    while Size(Children) < λ do
        S_i ← 0;
        Si_problem ← Mutate(Pi_problem, Pi_strategy);
        Si_strategy ← Mutate(Pi_strategy);
        Children ← S_i;
        i ← i + 1;
    end
    EvaluatePopulation(Children);
    S_best ← GetBest(Children + S_best, 1);
    Population ← GetBest(Children, μ);
end
return S_best;
```

mutations should be $\frac{1}{5}$ of all mutations. If it is greater the variance is increased, otherwise if the ratio is is less, the variance is decreased.

- The comma-selection variation of the algorithm can be good for dynamic problem instances given it's capability for continued exploration of the search space, whereas the plus-selection variation can be good for refinement and convergence.

9 Code Listing

Listing 1 provides an example of the Evolution Strategies algorithm implemented in the Ruby Programming Language. The demonstration problem is an instance of a continuous function optimization that seeks $minf(x)$ where $f = \sum_{i=1}^{n} x_i^2$, $-5.0 \leq x_i \leq 5.0$ and $n = 2$. The optimal solution for this basin function is $(v_0, \ldots, v_{n-1}) = 0.0$. The algorithm is a implementation of Evolution Strategies based on simple version described by Bäck and Schwefel [2], which was also used as the basis of a detailed empirical study [13]. The algorithm is an $(30+20)-ES$ Evolutionary Strategy that adapts both the problem and strategy (standard deviations) variables. More contemporary implementations may modify the strategy variables differently, and include an additional set of adapted strategy parameters to influence the direction of mutation (see [9] for a concise description).

```ruby
def objective_function(vector)
  return vector.inject(0.0) {|sum, x| sum + (x ** 2.0)}
end

def random_vector(problem_size, search_space)
  return Array.new(problem_size) do |i|
    search_space[i][0] + ((search_space[i][1] - search_space[i][0]) * rand())
  end
end

def gaussian
```

```ruby
  u1 = u2 = w = g1 = g2 = 0
  begin
    u1 = 2 * rand() - 1
    u2 = 2 * rand() - 1
    w = u1 * u1 + u2 * u2
  end while w >= 1
  w = Math::sqrt((-2 * Math::log(w)) / w)
  g2 = u1 * w;
  g1 = u2 * w;
  return g1
end

def mutate_problem(vector, stdevs, search_space)
  child = Array(vector.length)
  vector.each_with_index do |v, i|
    child[i] = v + stdevs[i] * gaussian()
    child[i] = search_space[i][0] if child[i] < search_space[i][0]
    child[i] = search_space[i][1] if child[i] > search_space[i][1]
  end
  return child
end

def mutate_strategy(stdevs)
  tau = Math.sqrt(2.0*stdevs.length.to_f)**-1.0
  tau_prime = Math.sqrt(2.0*Math.sqrt(stdevs.length.to_f))**-1.0
  child = Array.new(stdevs.length) do |i|
    stdevs[i] * Math::exp(tau_prime*gaussian() + tau*gaussian())
  end
  return child
end

def mutate(parent, search_space)
  child = {}
  child[:vector] = mutate_problem(parent[:vector], parent[:strategy], search_space)
  child[:strategy] = mutate_strategy(parent[:strategy])
  return child
end

def search(max_generations, problem_size, search_space, pop_size, num_children)
  strategy_space = Array.new(problem_size) do |i|
    [0, (search_space[i][1]-search_space[i][0])*0.02]
  end
  population = Array.new(pop_size) do |i|
    {:vector=>random_vector(problem_size, search_space),
      :strategy=>random_vector(problem_size, strategy_space)}
  end
  population.each{|c| c[:fitness] = objective_function(c[:vector])}
  gen, best = 0, population.sort{|x,y| x[:fitness] <=> y[:fitness]}.first
  max_generations.times do |gen|
    children = Array.new(num_children) {|i| mutate(population[i], search_space)}
    children.each{|c| c[:fitness] = objective_function(c[:vector])}
    union = children+population
    union.sort!{|x,y| x[:fitness] <=> y[:fitness]}
    best = union.first if union.first[:fitness] < best[:fitness]
    population = union[0...pop_size]
    puts " > gen #{gen}, fitness=#{best[:fitness]}"
  end
  return best
end

max_generations = 200
pop_size = 30
num_children = 20
```

```ruby
problem_size = 2
search_space = Array.new(problem_size) {|i| [-5, +5]}

best = search(max_generations, problem_size, search_space, pop_size, num_children)
puts "done! Solution: f=#{best[:fitness]}, s=#{best[:vector].inspect}"
```

Listing 1: Evolution Strategies algorithm in the Ruby Programming Language

10 References

10.1 Primary Sources

Evolution Strategies was developed by three students (Bienert, Rechenberg, Schwefel) at the Technical University in Berlin in 1964 in an effort to robotically optimize an aerodynamics design problem. The seminal work in Evolution Strategy was by Rechenberg's PhD thesis [8] that was later published as a book [7], both in German. Many technical reports and papers were published by Schwefel and Rechenberg, although the seminal paper published in English was by Klockgether and Schwefel on the two–phase nozzle design problem [6].

10.2 Learn More

Schwefel published his PhD dissertation [11] not long after Rechenberg that too was later published as a book [10] both in German. Schwefel's book was later translated into English was represents a classical reference for the technique [12]. Bäck, et al. provide a classical introduction to the technique, covering the history, development of the algorithm, and the steps that lead it to where it was in 1991 [1]. Beyer and Schwefel provide a contemporary introduction to the field that includes a detailed history of the approach, the developments and improvements since its inception, and an overview of the theoretical findings that have been made [3].

11 Conclusions

This report provided a description of the Evolution Strategies algorithm.

12 Contribute

Found a typo in the content or a bug in the source code? Are you an expert in this technique and know some facts that could improve the algorithm description for all? Do you want to get that warm feeling from contributing to an open source project? Do you want to see your name as an acknowledgment in print?

Two pillars of this effort are i) that the best domain experts are people outside of the project, and ii) that this work is wrong by default. Please help to make this work less wrong by emailing the author 'Jason Brownlee' at jasonb@CleverAlgorithms.com or visit the project website at http://www.CleverAlgorithms.com.

References

[1] Thomas Bäck, Frank Hoffmeister, and Hans paul Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

[2] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[3] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing: an international journal*, 1(1):3–52, 2002.

[4] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[5] Jason Brownlee. A template for standardized algorithm descriptions. Technical Report CA-TR-20100107-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[6] Jürgen Klockgether and Hans-Paul Schwefel. Two–phase nozzle and hollow core jet experiments. In *Proc. Eleventh Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148. California Institute of Technology, 1970.

[7] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, 1973.

[8] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* PhD thesis, Technical University of Berlin, Department of Process Engineering, 1971.

[9] Günter Rudolph. *Evolutionary Computation 1: Basic Algorithms and Operations*, chapter 9: Evolution Strategies, pages 81–88. IoP Press, 2000.

[10] H. P. Schwefel. *Numerische Optimierung von Computer – Modellen mittels der Evolutionsstrategie.* Birkhaeuser, 1977.

[11] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, Technical University of Berlin, Department of Process Engineering, 1975.

[12] Hans-Paul Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, 1981.

[13] Xin Yao and Yong Liu. Fast evolution strategies. In *Proceedings of the 6th International Conference on Evolutionary Programming VI*, pages 151–162, 1997.