Cross-Entropy Method*

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

November 22, 2010
Technical Report: CA-TR-20101122a-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 describes the Cross-Entropy Method using the standardized algorithm description template.

Keywords: `Clever, Algorithms, Description, Optimization, Cross-Entropy, Method`

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 [1]. 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 [2]. This report describes the Cross-Entropy Method using the standardized algorithm description template.

2 Name

Cross-Entropy Method, Cross Entropy Method, CEM

3 Taxonomy

The Cross-Entropy Method is a probabilistic optimization belonging to the field of Stochastic Optimization. It is similar to other Stochastic Optimization and algorithms such as Simulated Annealing, and to Estimation of Distribution Algorithms such as the Probabilistic Incremental Learning Algorithm.

4 Inspiration

The Cross-Entropy Method does not have an inspiration. It was developed as an efficient estimation technique for rare-event probabilities in discrete event simulation systems and was adapted for use in optimization. The name of the technique comes from the Kullback-Leibler cross-entropy method for measuring the amount of information (bits) needed to identify an event from a set of probabilities.

5 Strategy

The information processing strategy of the algorithm is to sample the problem space and approximate the distribution of good solutions. This is achieved by assuming a distribution of the problem space (such as Gaussian), sampling the problem domain by generating candidate solutions using the distribution, and updating the distribution based on the better candidate solutions discovered. Samples are constructed step-wise (one component at a time) based on the summarized distribution of good solutions. As the algorithm progresses, the distribution becomes more refined until it focuses on the area or scope of optimal solutions in the domain.

6 Procedure

Algorithm 1 provides a pseudo-code listing of the Cross-Entropy Method algorithm for minimizing a cost function.

Algorithm 1: Pseudo Code for the Cross-Entropy Method algorithm.

Input: $Problem_{size}$, $Samples_{num}$, $UpdateSamples_{num}$, $Learn_{rate}$, $Variancemin$
Output: S_{best}

```
Means ← InitializeMeans();
Variances ← InitializeVariances();
S_best ← 0;
while Max(Variances) ≤ Variancemin do
    Samples ← 0;
    for i = 0 to Samples_num do
        Samples ← GenerateSample(Means, Variances);
    end
    EvaluateSamples(Samples);
    SortSamplesByQuality(Samples);
    if Cost(Samples_0) ≤ Cost(S_best) then
        S_best ← Samples_0;
    end
    Samples_selected ←SelectBestSamples(Samples, UpdateSamples_num);
    for i = 0 to Problem_size do
        Means_i ← Means_i + Learn_rate × Mean(Samples_selected, i);
        Variances_i ← Variances_i + Learn_rate × Variance(Samples_selected, i);
    end
end
return S_best;
```

7 Heuristics

- The Cross-Entropy Method was adapted for combinatorial optimization problems, although has been applied to continuous function optimization as well as noisy simulation problems.
- A alpha (α) parameter or learning rate $\in [0.1]$ is typically set high, such as 0.7.
- A smoothing function can be used to further control the updates the summarizes of the distribution(s) of samples from the problem space. For example, in continuous function optimization a β parameter may replace α for updated the standard deviation, calculated at time t as $\beta_t = \beta - \beta \times (1 - \frac{1}{t})^q$, where β is initially set high $\in [0.8, 0.99]$ and q is a small integer $\in [5, 10]$.

8 Code Listing

Listing 1 provides an example of the Cross-Entropy Method 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 $n = 3$. The optimal solution for this basin function is $(v_0, \ldots, v_{n-1}) = 0.0$.

The algorithm was implemented based on a description of the Cross-Entropy Method algorithm for continuous function optimization by Rubinstein and Kroese in Chapter 5 and Appendix A of their book on the method [6]. The algorithm maintains means and standard deviations of the distribution of samples for connivence. The means and standard deviations were initialized based on random positions in the problem space and the bounds of the whole problem space respectively. A smoothing parameter is not used on the standard deviations.

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

def random_variable(minmax)
  min, max = minmax
  return min + ((max - min) * rand())
end

def random_gaussian(mean=0.0, stdev=1.0)
  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)
  g1, g2 = u1 * w, u2 * w
  return mean + g1 * stdev
end

def generate_sample(search_space, means, stdevs)
  vector = Array.new(means.length)
  means.length.times do |i|
    vector[i] = random_gaussian(means[i], stdevs[i])
    vector[i] = search_space[i][0] if vector[i] < search_space[i][0]
    vector[i] = search_space[i][1] if vector[i] > search_space[i][1]
  end
  return {:vector=>vector}
end

def mean_parameter(samples, i)
```

```ruby
  sum = samples.inject(0.0) do |s,sample|
    s + sample[:vector][i]
  end
  return (sum / samples.size.to_f)
end

def stdev_parameter(samples, mean, i)
  sum = samples.inject(0.0) do |s,sample|
    s + (sample[:vector][i] - mean)**2.0
  end
  return Math.sqrt(sum / samples.size.to_f)
end

def update_distribution!(samples, alpha, means, stdevs)
  means.length.times do |i|
    means[i] = alpha*means[i] + ((1.0-alpha)*mean_parameter(samples, i))
    stdevs[i] = alpha*stdevs[i] + ((1.0-alpha)*stdev_parameter(samples, means[i], i))
  end
end

def search(search_space, max_iter, num_samples, num_update, learning_rate)
  means = Array.new(search_space.length){|i| random_variable(search_space[i])}
  stdevs = Array.new(search_space.length){|i| search_space[i][1]-search_space[i][0]}
  best = nil
  max_iter.times do |iter|
    samples = Array.new(num_samples) {generate_sample(search_space, means, stdevs)}
    samples.each {|sample| sample[:cost]=objective_function(sample[:vector])}
    samples.sort!{|x,y| x[:cost]<=>y[:cost]}
    best = samples.first if best.nil? or samples.first[:cost] < best[:cost]
    selected = samples[0...num_update]
    update_distribution!(selected, learning_rate, means, stdevs)
    puts " > iteration=#{iter}, fitness=#{best[:cost]}"
  end
  return best
end

if __FILE__ == $0
  problem_size = 3
  search_space = Array.new(problem_size) {|i| [-5, 5]}
  max_iter = 100
  num_samples = 50
  num_update = 5
  learning_rate = 0.7

  best = search(search_space, max_iter, num_samples, num_update, learning_rate)
  puts "done! Solution: f=#{best[:cost]}, s=#{best[:vector].inspect}"
end
```

Listing 1: Cross-Entropy Method algorithm in the Ruby Programming Language

9 References

9.1 Primary Sources

The Cross-Entropy method was proposed by Rubinstein in 1997 [7] for use in optimizing discrete event simulation systems. It was later generalized by Rubinstein and proposed as an optimization method for combinatorial function optimization in 1999 [5]. This work was further elaborated by Rubinstein providing a detailed treatment on the use of the Cross-Entropy method for combinatorial optimization [4].

9.2 Learn More

De Boer, et al. provide a detailed presentation of Cross-Entropy method including its application in rare event simulation, its adaptation to combinatorial optimization, and example applications to the max-cut, traveling salesman problem, and a clustering numeric optimization example [3]. Rubinstein and Kroese provide a thorough presentation of the approach in their book, summarizing the relevant theory and the state of the art [6].

10 Conclusions

This report described the Cross-Entropy Method using the standardized algorithm description template.

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

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

[3] P. T. De Boer, D. P Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[4] R. Rubinstein. *Stochastic optimization: algorithms and applications*, chapter Combinatorial optimization, cross-entropy, ants and rare events, pages 303–364. Springer, 2001.

[5] Reuven Rubinstein. The simulated entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[6] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization*. Springer, 2004.

[7] R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1997.