The Genetic Algorithm is Useful to Fitting Input Probability Distributions for Simulation Models

Johann Christoph Strelen
Rheinische Friedrich–Wilhelms–Universität Bonn
Römerstr. 164, 53117 Bonn, Germany
E-mail: strelen@cs.uni-bonn.de

KEYWORDS

Stochastic Simulation, Fitting Probability Distributions, Genetic Algorithm

ABSTRACT

The genetic algorithm can be applied to selecting theoretical probability distributions so as to be representative for observed data. Two aspects are considered here: Using the genetic algorithm, one can decide which one of some different families of probability distributions is best suited, and parameters can be estimated.

INTRODUCTION

Many simulation models use random input which models influences from outside such as interarrival times, demand sizes, or times between failures of components. The distributions of these random variables must be chosen carefully since they influence the accuracy of the results and hence their trustworthiness. Typically, data are gathered in some real system, and a theoretical distribution is searched which is representative for them i.e. from which the data could be realizations. Usually this is done in three steps: In the first step, a specific family of distributions is hypothesized. Methods for this are histograms and the inspection of quantile summaries ("box plots"). In the second step, distribution parameters are estimated, usually using maximum-likelihood estimators. These are different for different families of distributions, and sometimes nonlinear equations must be solved. In the third step, one must decide if the found distribution is representative for the data: Graphic methods and goodness-of-fit tests can be applied like the chi-square test, the Kolmogorov-Smirnov test, or others.

This three-step procedure [6] applies if the data are statistically independent which can be decided with independence tests like the chi-square test. If the data are not independent, stochastic processes must be considered instead of single random variables which is much more difficult and hence not so common in simulation.

The genetic algorithm [2, 4, 5] is a stochastic global search method that mimics the metaphor of natural biological evolution. Genetic algorithms operate on populations of individuals applying the principle of survival of the fittest. Individuals are tuples of decision variable values. They are approximations of the desired solution, and their fitness measures the accuracy. At each generation, a new population is created by a randomized process of selecting individuals according to their level of fitness and breeding them together using operators borrowed from natural genetics: Properties of individuals mutate and are recombined in a random fashion. This process leads hopefully to the evolution of populations of individuals that are better approximations to the solution than their parents.

The genetic algorithm differs from traditional search and optimization methods. Significant differences are:

- Genetic algorithms search generations of approximations in parallel, not a single sequence
- Genetic algorithms require only the objective

function, no derivatives

- Genetic algorithms use probabilistic transition rules, not deterministic ones
- Genetic algorithms work on an encoding of the parameter set rather than the parameter set itself except for real-valued parameters

We applied a genetic algorithm on some tasks of the quoted steps one and two of the three-step procedure, and we remarked that this works very well and has some advantages compared to the traditional way via maximum-likelihood estimators: Here one has only two different algorithms, one for closed-form distribution functions and one with numerical integration of densities or with summation of discrete probabilities - on the other hand, maximum-likelihood estimators require individual solutions for different distributions. Many parameters can be considered, and the genetic algorithm can be used for the selection of different theoretical distributions.

For the numerical calculation we used MATLAB [1] with the Genetic Algorithm Toolbox [3] which provides very comfortable functions for genetic algorithms. It remains mainly to specify the number of decision variables (parameters), their type (integer or real), their degrees of accuracy (number of binary digits), and an objective function. Further specifications concern the features of the genetic algorithm like the number of generations, the number of individuals in a population, and others.

1 FITTING INPUT PROBABILITY DISTRIBUTIONS

The problem which is to be solved can be described as follows. Some data are given which were measured for a specific aspect of a real system. The modeller assumes that the data can be modelled as independent realizations of a random variable with a distribution chosen from about one and a half dozen different theoretical distributions which are known to be usefull for this purpose.

One of these distributions must be selected, and some parameters must be calculated. For, more precisely, each theoretical distribution is a whole family of distributions which differ with respect to some parameter values.

In the literature, for example in [6], a three-step procedure is proposed for selecting input probability distributions. In the first step, a specific family of distributions is hypothesized. Instead of the graphical methods which are mentioned in the introduction, we propose here to try to fit a weighted sum of different distributions to the data using a genetic algorithm, see purpose five in the sequel. Only if this yields a substantial value for a weight, the according distribution is accepted. In the second step, the distribution parameters are estimated. We propose to accomplish this with a genetic algorithm. In the third step, one must decide if the found distribution is representative for the data, usually with goodness-of-fit tests. This is not a topic of this paper, but a hypothesized distribution is thrown away if it turns out to not being suited to fit observed data accurately. Accuracy can be measured as follows.

The given data are sorted with respect to their values, x_1 is the smallest, and so on. Thus one obtaines the sorted sample $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and the empirical distribution function $\hat{F}(x) = i/n,\ x_i \leq x < x_{i+1},\ i = 0, \ldots, n$, where $x_0 = -\infty,\ x_{n+1} = \infty$.

Our objective function for the genetic algorithm is

$$Z(\mathbf{d}) = \sum_{i=1}^{n} [\hat{F}(x_i) - F_{\mathbf{d}}(x_i)]^2$$

where $\mathbf{d}$ denotes the tuple of decision variables, and $F_{\mathbf{d}}(x)$ is the selected distribution function with the parameter tuple $\mathbf{d}$. This objective function measures how accurate the distribution function $F_{\mathbf{d}}(x)$ fits the given data.

We identified five more or less different purposes concerning step one or step two of the three-step procedure for which a genetic algorithm can be applied to advantage.

Purpose 1, parameter estimation: Given data and a family of distributions, parameter values are calculated such that the distribution fits the given data. The number of different parameters is not so crucial as with nonlinear equations. Hence, a location parameter and/or a scale parameter can be added if not present anyhow.

Purpose 2, similar to purpose one but with multi-mode distributions. Here one takes a distribution function

$$\begin{aligned} F(x) &= p_1 F_1(x) + p_2 F_2(x) + \ldots, \\ & p_1 + p_2 + \ldots = 1, \end{aligned}$$

where $F_1,\ F_2, \ldots$ belong to the same family of distributions but may have different parameter values.

Purpose 3, similar to purpose two but with mixed different distributions. The distribution function for fitting the data is as above but $F_1,\ F_2, \ldots$ belong to different families of distributions, for example F_1 is Weibull, F_2 is Lognormal and so on.

These three purposes concern step two of the general procedure, the estimation of parameter values, where the families of distributions are given. But the genetic algorithm can also be used for step one, the selection of a suited theoretical distribution:

Purpose 4, falsification of a selected theoretical distribution: If one tries to fit a distribution to the data, and even with the best parameter values the found distribution differs a good deal from the empirical distribution, one can conclude that the selected distribution is a wrong one.

Purpose 5, automatic selection: If one tries to fit data with a mixed distribution, and one finds $p_1 \approx 1,\ p_2 \approx 0, p_3 \approx 0, \ldots$, and the found distribution function $F_1(x)$ is close to the empirical distribution function, one may conclude that $F_1(x)$ is representative but the other ones are not. If more than one probability p_i is significantly greater than zero, one may accept the according mixed distribution function if it is close to the empirical distribution function.

Sometimes there is no closed form of the distribution function available, for example if the distribution is Gamma or Lognormal. Here we use the density $f_{\mathbf{d}}(x)$ of the distribution and calculate the distribution function $F_{\mathbf{d}}(x)$ at the values $x_i,\ i = 1, \ldots, n$, approximately with simple numerical integration. In particular, we take

$$\begin{aligned} \tilde{F}(x_1) &= 1/n, \\ \tilde{F}(x_i) &= \sum_{j=2}^{i} (x_j - x_{j-1}) f_{\mathbf{d}}(x_{j-1}),\ i = 2, \ldots, n, \\ F_{\mathbf{d}}(x_i) &\approx \tilde{F}(x_i)/\tilde{F}(x_n),\ i = 1, \ldots, n. \end{aligned}$$

2 THE GENETIC ALGORITHM

As stated in the introduction, the *genetic algorithm* [2, 4, 5] is a stochastic global search method that mimics the metaphor of natural biological evolution. Genetic algorithms operate on *populations* of individuals applying the principle of survival of the fittest. *Individuals* are tuples of decision variable values which are encoded as strings over the binary alphabet or other alphabets. The individuals are approximations of the desired solution, and their *fitness* measures the acuracy which is fixed by an objective function. At each *generation*, a new population is created by the process of selecting individuals according to their level of fitness and breeding them together using operators borrowed from natural genetics. The *recombination operator* is used to exchange parts of the strings between pairs, or larger groups, of individuals, according to some probabilistic rule. *Mutation* will cause a single bit to change its state with some probability, in the binary string representation. *Selection* serves the purpose to select individuals for the next generation.

This process leads hopefully to the evolution of populations of individuals that are better approximations to the solution than their parents.

In each problem which is to be solved here, the genetic algorithm is applied to searching for a minimum of the objective function. The result is a tuple of values of the real valued decision variables $d_i,\ i = 1, \ldots, N^{(\mathrm{var})}$. For each decision variable, a minimum and a maximum is given in advance, and the values are encoded with $PRECI$ binary digits. Grey coding is used, hence adjacent values differ in just one digit. An $N^{(\mathrm{var})}$-tuple of decision values is an individual, and $N^{(\mathrm{ind})}$ individuals are a population. The encoded values of an individual are concatenated, hence form a binary string of $N^{(\mathrm{var})} * PRECI$ digits which is termed the chromosome of the individual. The chromosomes of all individuals of a population are the matrix $CHROM$ with $N^{(\mathrm{ind})}$ rows, each a chromosom.

The genetic algorithm works as follows. The matrix $CHROM$ is initialized with uniformly distributed random numbers. For each individual, its chromosome is decoded into a decision variable tuple $\mathbf{d}$, and the objective function $Z(\mathbf{d})$ is calculated.

In a loop, $MAXGEN$ populations are calculated, one after the other: For each individual, the fitness with linear ranking is calculated according to the value of the objective function $Z(\mathbf{d})$ of its decision variables, as follows. The individuals are sorted according to descending values of their objective function values, i.e. the best individual gets the position $pos = N^{(\text{ind})}$. Each individual on position pos gets the rank $2(pos-1)/(N^{(\text{ind})}-1)$, rank 2 is the best, rank 0 the worst.

$GGAP$, $0 < GGAP \leq 1$, is the generation gap: The $(1-GGAP)*N^{(\text{ind})}$ individuals with the best fitness values remain unchanged, and the other $GGAP*N^{(\text{ind})}$ individuals are selected for breeding offspring with the method of stochastic universal sampling: Each individual i gets a probability p_i which is proportional to its fitness value. The selection is according to these probabilities where individual i is selected at least $\lfloor p_i N^{(\text{ind})} GGAP \rfloor$ times and at most $\lceil p_i N^{(\text{ind})} GGAP \rceil$ times ($\lfloor . \rfloor$ means the floor, and $\lceil . \rceil$ the ceil).

The selected chromosomes are pairwise recombined with the single-point crossover operator, each pair with probability 0.7, and with probability 0.3, the two individuals of a pair remain unchanged with respect to recombination. If the number of selected individuals is odd, one more remains unchanged.

For this crossover, a position within the two chromosomes of a pair is selected at random, and the left part of one chromosome is concatenated with the right part of the other, and vice versa. The result are two new chromosomes.

Now on all chromosomes, the mutation operator is applied. Each binary digit flips with probability $0.7/L^{(\text{ind})}$ where $L^{(\text{ind})} = N^{(\text{var})} * PRECI$ is the number of binary digits in each chromosome.

The objective function values are now calculated for offspring.

The reinsertion step performs insertion of offspring into the current population, replacing least fit parents with offspring.

Now the next population is ready and the process continues if the number of populations is less than $MAXGEN$.

The individual with the best objective function value is the result, and this value measures the accuracy of the fitted probability distribution.

3 EXAMPLES

Now we present some numerical examples which indicate the capabilities of the genetic algorithm technique for fitting distributions. In each of these experiments, an independent sample $\mathbf{x}$ is generated according to a known distribution which is two-mode in example 2. In the examples 1, 2, and 3, this known theoretical distribution is fitted to the sample. In example 4, an other distribution is fitted, and the result is very inaccurate. This indicates that the tried distribution is not suited. In example 5, a distribution which is mixed from two different theoretical distributions is tried to fit. The result indicates that one of them is suited, the other one is not. The value of the objective function is given for all fitted distributions as a measure of the accuracy.

Example 1, Fitting Data Drawn from a Weibull Distribution. The sample consists in 800 realizations of a Weibull random variable with the distribution function $F(x) = 1 - \exp[-(x/\beta)^{\alpha}]$ where $\alpha = 2$ and $\beta = 3$. After 200 generations in the genetic algorithm, the best approximated parameters where 2.047 and 3.031, respectively, and the accuracy is $Z(2.047, 3.031) = 0.0293$.

In the figures, the smooth curve is the fitted theoretical distribution function, and the scribbling curve is the empirical distribution function. In figure 1 for this example, a smaller sample size is considered for the figure in order to have the two curves clearer separated: the sample size is 200 and the number of generations 800.

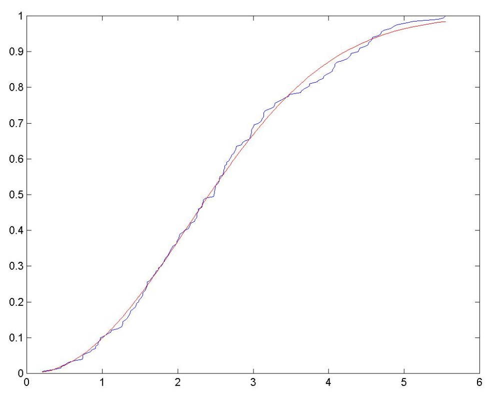

Figure 1: Fitted Weibull Distribution Function

Example 2, Fitting Data Drawn from a Two-Mode Weibull Distribution. The sample consists in 800 realizations according to the mixed distribution function $F(x) = 1-0.5\exp[-(x/3)^2]-0.5\exp[-(x/17)^5]$, the fitted distribution function (figure 2) is $F(x) = 1 - 0.51\exp[-(x/2.95)^{2.05}] - 0.49\exp[-(x/16.98)^{5.14}]$. The accuracy is 0.024.

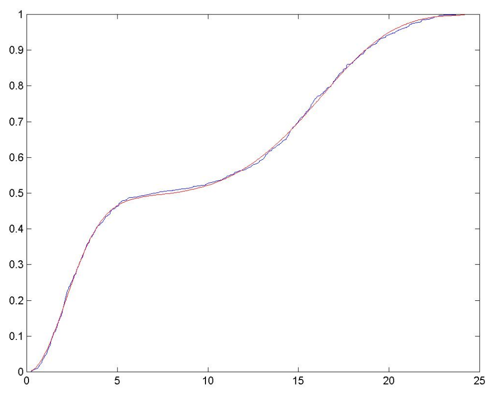

Figure 2: A fitted two-mode Weibull distribution function

Example 3, Gamma Distribution with Numerical Integration. The sample consists in 800 realizations of a Gamma random variable with the density

$$f(x) = \beta^{-\alpha}x^{\alpha-1}\exp[-(x/\beta)]/\Gamma(\alpha)$$

for $x \geq 0$ where $\alpha = 3$ and $\beta = 3$. The fitted distribution functions are obtained with numerical integration, for each parameter set. After 100 generations in the genetic algorithm, the best approximated parameters where 3.01 and 2.94, respectively, and the accuracy is $Z(3.01, 2.94) = 0.031$. See figure 3.

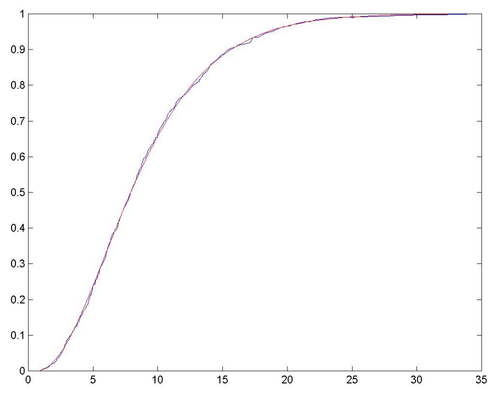

Figure 3: Fitted Gamma distribution using numerical integration

Example 4, Fitting a Wrong Distribution. The sample consists in 3200 realizations of a Weibull random variable with the parameter values $\alpha = 2$ and $\beta = 3$. It is tried to fit these data with a Gamma distribution. After 800 generations in the genetic algorithm, the accuracy is only 0.76; that means, the Gamma distribution is not suited for these data, see figure 4.

Example 5, Decision between Gamma and Weibull Distribution. The sample consists in 400 realizations of a Weibull random variable with the parameter values $\alpha = 3$ and $\beta = 3$. It is tried to fit these data with a distribution function, mixed Weibull and Gamma,

$$F(x) = pF_1(x) + (1-p)F_2(x)$$

where F_1 is Weibull and F_2 is Gamma. After 400 generations, the best parameters are $p = 0.998$, $\alpha = 3.00$, $\beta = 2.96$ where the accuracy is 0.047, see figure 5. That means, the genetic algorithm found out that the data should be fitted with the Weibull distribution, not with the Gamma distribution, and calculated the parameters.

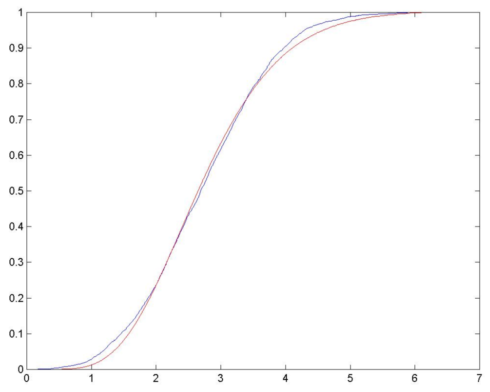

Figure 4: Gamma distribution fitted to Weibull data - not accurate

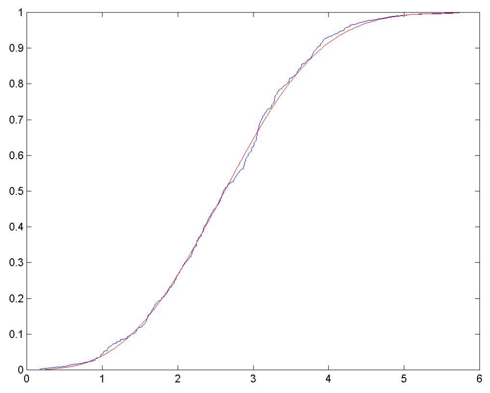

Figure 5: Decision in favour of a Weibull distribution

References

[1] *MATLAB - http://www.mathworks.com/.*

[2] T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation.* Oxford University Press, Bristol, New York, 1997.

[3] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca. *Genetic Algorithm Toolbox for Use with MATLAB.* http://www.shef.ac.uk/∼gaipp/ga-toolbox/.

[4] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Norstrand Reinhold, New York, 1991.

[5] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading,MA, 1989.

[6] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, New York, third edition, 2000.

Johann Christoph Strelen was born in Wiesbaden, Germany, in 1941. He received the Dipl.-Math. and Dr. rer. nat. degrees in mathematics and the Habilitation degree in Computer Science from the Technische Hochschule Darmstadt, Germany, in 1968, 1973, and 1981, respectively. There he was affiliated to the Computing Center (1968-1973), and assistant at the Computer Science Department (1974-1982). In 1973 he was a post doctoral fellow at the IBM Scientific Center, Grenoble, France. Since 1982 he has been a Professor of Computer Science at the Rheinische Friedrich–Wilhelms–Universität Bonn, Germany. His research interests include performance evaluation, distributed systems, and simulation. Dr. Strelen is a member of the Gesellschaft für Informatik (GI).