Maximum Entropy Formalism and Genetic Algorithms

Willi-Hans Steeb

International School for Scientific Computing, University of Johannesburg,
Auckland Park 2006, South Africa

Reprint requests to Prof. W.-H. S.; E-mail: WHS@NA.RAU.AC.ZA

Z. Naturforsch. **61a,** 556 – 558 (2006); received August 16, 2006

We apply genetic algorithms to find solutions of the maximum entropy formalism. As an application we consider a chaotic map.

Key words: Maximum Entropy Formalism; Genetic Algorithm.

The maximum entropy principle [1 – 3] is a powerful tool in the investigations of image reconstruction, spectral analysis, seismic inversion, inverse scattering etc. It is proven to be the only consistent method for inferring from incomplete information. Here we show that the maximum entropy principle can be cast into a unconstrained optimization problem and therefore genetic algorithms [4, 5] can be applied to solve it. Then we give an application to a chaotic system.

The missing information function (entropy) of a probability density p is defined as

$$S = -\int_\Omega p(x) \ln p(x)\, \mathrm{d}x.$$

In the maximum entropy formalism one maximizes the missing information subject to the constraints of the available information and to the normalization of the probability density. In our case we assume that we have the n lowest moments of the time evolution of the dynamical variables which, for ergodic systems, are equal to the moments of the temporal probability density. The constraints are introduced via the method of Lagrange multipliers $\lambda_0, \lambda_1, \ldots, \lambda_m$. Our aim is to find the approximate probability density p_{app} which minimizes

$$S' = -\int_\Omega p_{\mathrm{app}} \ln p_{\mathrm{app}}\, \mathrm{d}x + \lambda_0 \left(1 - \int_\Omega p_{\mathrm{app}}\, \mathrm{d}x\right) + \sum_{j=1}^{m} \lambda_j \left(\langle x^j \rangle - \int_\Omega x^j p_{\mathrm{app}}\, \mathrm{d}x\right),$$

where λ_j $(j = 0, 1, \ldots, m)$ are the Lagrange multipliers for the $m+1$ constraints. Performing the minimization we obtain

$$p_{\mathrm{app}}(x) = \exp\left(-1 - \sum_{j=0}^{m} \lambda_j x^j\right) = \frac{1}{Z} \exp\left(-\sum_{j=1}^{m} \lambda_j x^j\right),$$

where $Z := \mathrm{e}^{1+\lambda_0}$ is determined by the normalization of the probability density so that

$$1 = \frac{1}{Z} \int_\Omega \exp\left(-\sum_{j=1}^{m} \lambda_j x^j\right) \mathrm{d}x.$$

The remaining Lagrange multipliers are obtained by solving the following set of m coupled nonlinear equations for λ_j $(j = 1, 2, \ldots, m)$:

$$\langle x^j \rangle = \frac{1}{Z} \int_\Omega x^j \exp\left(-\sum_{k=1}^{m} \lambda_k x^k\right) \mathrm{d}x, \quad j = 1, 2, \ldots, m.$$

For the numerical study we proceed as follows. Without any loss of generality we restrict ourselves to $x \in [-1, 1]$. We start using the quadrature formula

$$S = -\int_{-1}^{1} p(x) \ln p(x)\, \mathrm{d}x \approx -\sum_{j=1}^{n} w_j p_j \ln p_j, \quad (1)$$

where $p_j = p(x_j)$. w_j and x_j are the weights and abscisses of any accurate quadrature formula, for example Gauss-Legrendre or Gauss-Chebyshev [6]. We maximize S subject to the discretized moment constraints

$$\sum_{j=1}^{n} w_j x_j^i p_j \equiv \sum_{j=1}^{n} x_j^i \tilde{p}_j = \mu_i, \quad i = 0, 1, \ldots, m, \quad (2)$$

0932–0784 / 06 / 1000–0556 $ 06.00

where μ_j are the given moments and $m+1$ is the number of moments. We define $\tilde{p}_j := w_j p_j$. Thus we optimize the Lagrangian function

$$L(\tilde{p}, \tilde{\boldsymbol{\lambda}}) = \sum_{j=1}^{n} \tilde{p}_j \ln\left(\frac{\tilde{p}_j}{w_j}\right) - \sum_{i=0}^{m} \tilde{\lambda}_i \left(\sum_{j=1}^{n} x_j^i \tilde{p}_j - \mu_i\right). \quad (3)$$

The solution can be written as

$$\tilde{p}_j = w_j \exp\left(\sum_{k=0}^{m} x_j^k \tilde{\lambda}_k - 1\right), \quad j = 1,2,\ldots,n. \quad (4)$$

Since the weights w_j $(j = 1,2,\ldots,n)$ are positive, it implies that $\tilde{p}_j \geq 0$ $(j = 1,2,\ldots,n)$. From (4) it follows that

$$\mu_i = \sum_{j=1}^{n} x_j^i \tilde{p}_j = \sum_{j=1}^{n} x_j^k w_j \exp\left(\sum_{k=1}^{m} a_{kj} \tilde{\lambda}_k - 1\right), \quad (5)$$
$$i = 0,1,\ldots,m.$$

This equation can be derived by minimizing

$$\mathrm{d}(\tilde{\boldsymbol{\lambda}}) = \sum_{j=1}^{n} w_j \exp\left(\sum_{i=0}^{m} x_j^i \tilde{\lambda}_i - 1\right) - \sum_{j=0}^{m} \mu_j \tilde{\lambda}_j. \quad (6)$$

Thus we have removed the constraint (2). The most difficult problem in applying genetic algorithms is the inclusion of constraints. To take into account contraints the penalty method [4] is used. In a penalty method, a constrained problem in optimization is transformed to an unconstrained problem by associating a cost or penalty with all constraint violations. This cost is included in the objective function evaluation. This technique is very clumsy and not very efficient in actual numerical calculation. Thus the removal of the contraint is of benefit for the application of genetic algorithms. Given the moments μ_i, the weights w_i and abscisses x_i the function $\mathrm{d}(\tilde{\boldsymbol{\lambda}})$ can be used as fitness function for the genetic algorithm to find the $\tilde{\lambda}_i$'s. The applicability of the maximum entropy to chaotic maps has been described by Steeb and Stoop [3] and Steeb [4].

Genetic algorithms are a family of computational models inspired by evolution [4, 5]. The algorithms encode a potential solution of a specific problem on a simple chromosome-like data structure (in our case a bitstring) and apply recombination operators (crossing, mutation) to these structures so as to preserve critical information. In most cases genetic algorithms are used to find the optimum of functions (these functions are called fitness functions). Hardy et al. [7] showed that one can directly manipulate the bits in floating point numbers. The use of floating point numbers (for example `double` in C and C++) instead of a standard bit string (for example from the Standard Template Library) will speed up the actual computation since we directly operate on the bit string of the floating point number which is the argument of the function we minimize [7]. This technique will be used in the following.

The Gauss-Legendre integration formula can be expressed as

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{j=1}^{n} w_j f(x_j),$$

where the arguments x_j are the zeros of the Legendre polynomials $P_n(x)$ and the coefficients (weights) w_j are given by

$$w_j = \frac{2(1-x_j^2)}{n^2 (P_{n-1}(x_j))^2}.$$

The Gauss-Chebyshev quadratures

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} w_k f(x_k)$$

are more suitable for our problem. Here the x_k are roots of the Chebyshev polynomials of order n and w_k are weights. The roots of the Chebyshev polynomials of order n are

$$x_k = \cos\left(\frac{(k-1/2)\pi}{n}\right), \quad k = 1,2,\ldots,n$$

and the weights are given by $w_k = \pi/n$ $(k = 1,2,\ldots,n)$. It follows that

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,\mathrm{d}x \approx \frac{\pi}{n} \sum_{k=1}^{n} f(x_k).$$

As an example we consider the map $f : [-1,1] \to [-1,1]$, $f(x) = 1 - 2x^2$ or $x_{t+1} = f(x_t)$ with $t = 0,1,2,\ldots$ This map shows chaotic behaviour and is ergodic. The Liapunov exponent is given by $\ln(2)$. The invariant density is given by

$$p(x) = \frac{1}{\pi\sqrt{1-x^2}}$$

with

$$\int_{-1}^{1} p(x)\,\mathrm{d}x = 1.$$

The moments are

$$\langle x_t^n \rangle = \int_{-1}^{1} y^n p(y)\,\mathrm{d}y.$$

For $n = 0$ we have $\langle x_t^0 \rangle = 1$, for $n = 1$ we have $\langle x_t \rangle = 0$ and for $n = 2$ we have $\langle x_t^2 \rangle = 1/2$. Applying genetic algorithm (using mutation and crossing) to the fitness function (6), where λ_0, λ_1 and λ_2 are considered as bit-strings of floating point type `double` in C++ [7], we obtain

$$\lambda_0 \approx 0.4, \quad \lambda_1 \approx 0, \quad \lambda_2 \approx -1.7.$$

We use from ergodic theory [4]

$$\lambda \approx \int_{-1}^{1} p_{\mathrm{app}}(x) \ln \left| \frac{\mathrm{d}f}{\mathrm{d}x} \right| \mathrm{d}x$$

to find an approximation for the Liapunov exponent λ. The integration yields 0.72 which agrees well with the exact value of $\ln 2$, if we take into account that we only considered two moments. Since we integrate $\ln |x|$ around 0 we have to take $n = 100$ to obtain the required precision.

[1] E. T. Jaynes, Phys. Rev. **106**, 620 (1957).
[2] W.-H. Steeb and R. Stoop, J. Math. Phys. **35**, 6604 (1994).
[3] W.-H. Steeb, F. Solms, and R. Stoop, J. Phys. A: Math. Gen. **27**, L399 (1994).
[4] W.-H. Steeb, The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Gene Expression Programming, Support Vector Machine, Wavelets, Hidden Markov Models, Fuzzy Logic with C++, Java and SymbolicC++ Programs, 3rd ed., World Scientific Publishing, Singapore 2005.
[5] W.-H. Steeb, Y. Hardy, A. Hardy, and R. Stoop, Problems and Solutions in Scientific Computing, World Scientific Publishing, Singapore 2004.
[6] F. Scheid, Numerical Analysis, Schaum's Outline Series in Mathematics, McGraw-Hill, New York 1968.
[7] Y. Hardy, W.-H. Steeb, and R. Stoop, Int. J. Mod. Phys. C **16**, 1811 (2005).