General optimization, global optimization and Evolutionary algorithms

Jost Neigenfind

Gute Ideen in der theoretischen Biologie/Systembiologie
July 9, 2007

Abstract

Optimization is an important task in physics, engineering and informatics especially in bioinformatics. It allows the development of optimal solutions to a given problem. However, there are different problem instances which have to be treated differently. This paper will give an overview over these different instances and where they occure. Moreover the alternative ways to treat them correctly will be explained. In the end, a slight introduction is given to Gene Expression Programming [7].

Introduction

Optimization problems occure in physics (e.g. minimizing the internal energy of molecules), in engineering (e.g. chip design) [10][8] and in informatics (Travelling salesman problem) especially bioinformatics (e.g. micro array analysis). Each problem instance can be reformulated as target function which has to be minimized (or maximized):

$$f : \mathbb{R}^D \to \mathbb{R} \tag{1}$$

Unfortunately, these target functions and the corresponding function value can have different features. Target functions can differ in their dimensionality, i.e. the number of variable parameters where the parameters can be continous or discrete. Additionally, target functions can be differentiable (or not) and/or constraints are given so that only a given region of the solution space is of interest. In contrast, the function value is 0-dimensional and also can be continous or discrete. However, the target function can be multi-modal so that there can be several local optima and one global optimum.
Clearly, all the above mentioned classes of target functions have to be treated differently, since a method that works perfectly on differntiable target functions need not work on target functions with discret function values. In the following the different classes and a corresponding method for optimization are presented.

Optimization of differentiable target functions

Target functions must be two time differentiable at every point so that the gradient at a given point and the corresponding bending can be calculated. The gradient is given by the first partial differentiation:

$$f'(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{bmatrix} \tag{2}$$

where x is a vector with D elements representing the parameters. Since it is not known if an optimum is found in the general situation even if a parameter vector with $f'(x) = 0$ is found (e.g.

saddle point), the second partial differentiation is needed which is given by Hessian matrix:

$$f''(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial f}{\partial x_1 \partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_D \partial x_1} & \cdots & \partial x_D \partial x_D \end{bmatrix} \tag{3}$$

where x again is a vector with D elements representing the parameters. By means of the two differentiations the optima can be calculated directly if the target function is quadratic. In general this is not possible and alternative methods have to be used which work iteratively. There are many of those methods that find optima by means of the two differentiations [8]:

- Gauss-Newton
- Fletcher-Reeves
- Davidon-Fletcher-Powell
- Broyden-Fletcher-Goldfarb-Shanno
- Levenberg-Marquardt

However, the simplest gradient based method, Steepest Descent, optimizes without the second partial differntiation. Therefore, f''^{-1} is replaced by the identity matrix so that the corresponding iteration looks as follows: $x_{n+1} = x_n - \gamma \cdot g(x_n)$ where γ defines a step size. However, some problems

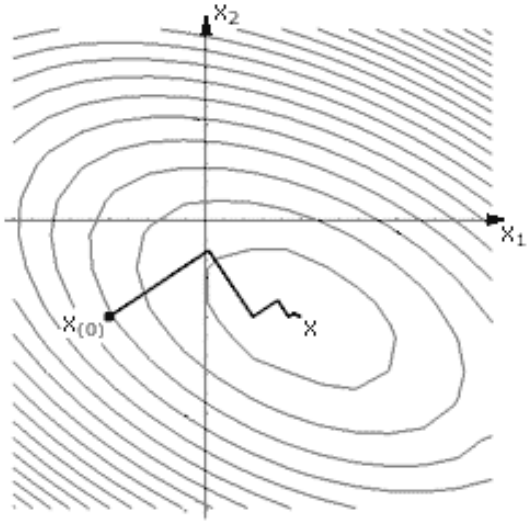

Figure 1: Path of the steepest descent method [1]

are left open by the gradient based optimization methods: First, these methods are useless if the solution space of the target functions is discrete. And second, in fact these methods guarantee to find an local optimum but they do not guarantee to find a global one.

Optimization of non differentiable target functions

Since the differentiable methods do not work on discret target functions, other methods are needed. One class of methods is based on a simplex. In geometry, a simplex or n-simplex is an n-dimensional analogue of a triangle [2]. Specifically, a simplex is the convex hull of a set of $(n + 1)$ affinely independent points in some Euclidean space of dimension n or higher [2]. One example for simplex based methods is the Nelder-Mead method [3]. This method works as follows on a target function with two parameters [8] :

(1) sample randomly three points into solution space

(2) sort the three points by the function value

(3) reflect the worst point at the line generated by the other two

(4) try to improve:

  (4.1) if reflection returns function value better than the other three go one step further into the same direction assuming that further improvment is still possible (expansion step)

  (4.2) if expansion succeeds, add corresponding point to set of three points and drop the worst point from this set

  (4.3) if expansion fails, add point returned by the reflection to the set of three points and drop the worst point from this set

  (4.4) if reflection fails but returned function value is still better than the worst of the three others, add it this point to the set and drop the worst point

  (4.5) if (4.1) - (4.4) fail, shrink the simplex

(5) go to (2) or finish iteration if some criterion for finishing is reached

Point (4.5) guarantees that the simplex is able to find optima for which the starting simplex is too unsophisticated. Clearly, this method works on continous or discrete target functions but does not solve the problem of multi-modal target functions.

Simulated annealing

Simulated annealing (SA) is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global optimum of a given function in a large search space [4]. This search method is inspired by metallurgy where annealing describes the process after hardening the metal. Hardening is done by fast cooling down the metal. By this procedure, the atoms of the metal can not reach the configuration of minimized energy of the system since they are "frozen" at their current configuration. Such a configuration contains a lot of defects that can provide an early breaking of the metal. Because of that, the metal is heated again (but not melted) and slowly cooled down (annealed). This annealing procedure makes it possible to get from the current, local stable configuration of atoms and corresponding energy of the system to configurations with higher energies. From these "higher" points the annealing procedure finds other local minima or even the global minimum. Clearly, accepting worse energies of the system as new configuration is not arbitrary and in physics this is described by a probability distribution dependent on the temperature. Finally, the atoms end up in their configuration with the minimal energy of the system which is mostly crystal like. The probability to observe a given state (configuration) j with energy niveau E_j is described by the Boltzmann distribution [5]:

$$P(E_j) = \frac{\exp\left(\frac{-E_j}{k_B T}\right)}{Z(T)} \tag{4}$$

where k_B is the Boltzmann constant, T the temperature and $Z(T)$ the so called partition function:

$$Z(T) = \sum_j \exp\left(\frac{-E_j}{k_B T}\right) \tag{5}$$

where the partition function is a normalization factor for the Boltzmann distribution. In 1953, Metropolis *et. al.* [9] developed a simple algorithm which provides an efficient simulation of a collection of atoms in equillibrium at a given temperature. The algorithm computes a small displacement for an atom and calculates the corresponding energy difference in the system, ΔE. If ΔE is less than 0, the new configuration is accepted and the new configuration of atoms is used as starting point for the next iteration step. If ΔE is bigger than 0, i.e. the energy of the system

increases, the acceptance is treated probabilistically. The new, worse configuration is accepted with the probability [10]:

$$P(\Delta E) = \exp\left(\frac{-\Delta E}{k_B T}\right) \tag{6}$$

Thus, the system evolves into a Boltzmann distribution [10].
The method described by Metropolis *et. al.* [9] can be easily applied on arbitrary target functions where the atomic configuration is represented by a given set of parameters and the ΔE is represented by the difference between the function values of two different parameter sets. Additionally, the Boltzmann constant has to be changed to a problem dependent control variable. The temperature T is equally treated as in physics or in the algorithm described by Metropolis [9][10]. Since the problem dependant control variable, corresponding to the Boltzmann constant, has to be chosen very carefully, the whole annealing schedule has to be chosen carefully. I.e. if the temperature T is lowered to fast, the algorithm would behave like a local search algorithm and would be caught in a local optimum. If the temperature is lowered to slow, the running time of the algorithm might last too long. However, finding the optimal annealing schedule is an optimization problem of an optimization problem.

Evolutionary algorithms

Evolutionary algorithms (EAs) are search heuristics for global optima. There are two classes of algorithms, evolution strategies and genetic algorithms. Both classes are inspired by biology. It is assumed that living organisms adapt to the environment perfectly and that evolution is able to find local and global optima by mutation, recombination and selection. On the one hand, evolutionary strategies were developed by Rechenberg and Schwefel and are applicable to continous target functions with continous parameter representation (float, double) [8]. On the other hand, genetic algorithms were developed by Holland and Goldberg and are applicable on combinatorial problems with discrete (binary) representation of parameters [8]. Mutation mostly consists of adding a random vector to the offspring vector, where recombination can be done directly by exchanging elements of the parental vectors. How mutation and recombination are realized differs from method to method but are always directly inspired by biology. However, if the parameter representation of an evolutionary strategy can be transformed to a discret representation, the evolutionary strategy is converted to an genetic algorithm (and vice verca) [8].The meta algorithm for an evolutionary algorithm is given in Figure 2.

```
Initialization();       //choose starting population of μ members
while (not converged)   //decide the number of iterations
{
   for (i=0; i<λ; i++) //child vector generation: λ > μ
   {
     p1(i) = rand(μ);     //pick a random parent from μ parents
     p2(i) = rand(μ);     //pick another random parent  p2(i) != p1(i)
     c1(i) = recombine(p1(i),p2(i)); //recombine parents
     c1(i) = mutate(c1(i));          //mutate child
     save(c1(i));         //save child in an intermediate population
    }
   selection();          //μ new parents out of either λ, or λ+μ
}
```

Figure 2: Meta-algortihm for evolutionary strategies and genetic algorithms [8]

	target f.			optimum			
method	continous	discret	varying	local	global	# of f. values	running time
steepest d.	yes	no	parameters	yes	no	single	fast
simplex	yes	yes	parameters	yes	no	single/multi	fast
simulated ann.	yes	yes	parameters	yes	yes*	single	slow
EAs	yes	yes	parameters	yes*	yes*	multi	slow
GEP	no	yes	function	yes	no	single	fast

Table 1: Advantages and disadvantages of the single methods (* but not guaranteed)

Gene expression programming

Gene expression programming (GEP) [7] is a very new method for optimizing whole functions or programs. In contrast to the above mentioned methods, gene expression programming is more similar to polynomial interpolation: given some data, calculate a function that represents this data with smallest distance. Optimization of these functions again is inspired by biology and work similar to the evolutionary algorithms. Functions are represented as genes which can be reformulated to expression trees (ETs). Genes are "simple" strings of operators and variables where the corresponding expression tree defines how the function must be evaluated. Moreover, these genes can be organized in whole chromosomes where the single genes respectively functions are combined by other functions (e.g. "if"). However, since this method is relatively new the interested reader is referred to the original paper [7].

Summary

In Table 1, the advantages and disadvantages of the single methods are summarized. Additionally, there is a general proceeding for optimization:

- try as much algorithms as possible which are applicable to the given problem
- try different sets of parameters of the algorithm instead of default values
  - annealing schedule
  - population size ($\geq$ 10x number of parameters)
- start algorithm at different coordinates of solution space ($\geq$ 10x)

However, the methods presented in this paper give only a slight overview about possible optimization methods. There are new heuristics inspired by biology for the search in high dimensional solution spaces (e.g. Ant colony optimization [6]) which are not presented in this paper.

References

[1] http://www.basegroup.ru/images/neural/conjugate/pict2.gif.

[2] http://en.wikipedia.org/wiki/Simplex.

[3] http://en.wikipedia.org/wiki/Nelder-Mead_simplex_method.

[4] http://en.wikipedia.org/wiki/Simulated_annealing.

[5] http://de.wikipedia.org/wiki/Boltzmann-Verteilung.

[6] http://en.wikipedia.org/wiki/Ant_colony_optimization.

[7] C. Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, 13, 2:87–129, 2001.

[8] K. V. Price, J. Lampinen, R. Storn. *Differential Evolution.* Springer.

[9] M. Rosenbluth A. Teller E. Teller N. Metropolis, A. Rosenbluth. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, vol. 6, no. 21:1087–1092, 1953.

[10] Jr. M.P. Vecchi S. Kirkpatrick, C.D. Gelatt. Optimization by simulated annealing. *Science*, 220, 4598:671–680, 1983.