CHEBYCHOP
A new method for constrained global optimization

John M. Vincent
vintronia@gmail.com

July 12, 2019

Abstract

A new method for multi-dimensional global optimization is presented. The fundamental idea is to obtain the Chebyshev norm from a set of L_p-norms; an original extrapolation procedure is used to converge to the optimum value, which is identical to the Chebyshev norm. This allows the optima to be efficiently pin-pointed by repeatedly bisecting the domain. Estimates of computational cost indicate that the method is fast.

1 Introduction

The generic optimization problem that CHEBYCHOP has been devised for is:

$$F_{MAX}(\mathcal{D}) = \max_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}), \tag{1}$$

where $f(\boldsymbol{x})$ is a continuous function of a D-dimensional vector, $\boldsymbol{x}$, and $\mathcal{D}$ is the domain. As well as calculating the optimum, F_{MAX}, the set of vectors, $\mathcal{X}$, where the optimum occurs also needs to be obtained.

In order to calculate the optimum, the approach taken is to calculate the Chebyshev norm, since $F_{MAX} = \|f\|_\infty$. The Chebyshev norm is the asymptotic limit of the L_p-norm:

$$\|f\|_p = \left(\int_{\mathcal{D}} |f|^p \, \mathrm{d}\mu \right)^{1/p},$$

and

$$F_{MAX} = \lim_{p \to \infty} \|f|\|_p.$$

The extrapolation process is discussed in the next section.

The set $\mathcal{X}$ is determined by 'chopping' the original domain into two sub-domains, $\mathcal{D}_a$ and $\mathcal{D}_b$; then $F_{MAX}(\mathcal{D}_a)$ and $F_{MAX}(\mathcal{D}_b)$ are calculated. The smallest is discarded and the chopping process is repeated until the optima are located with sufficient accuracy; if the maxima are almost identical then both sub-domains are retained.

2 The extrapolation process

Pragmatically we are extrapolating the data set,

$$\mathcal{S}_N = \{\|f\|_1, \cdots, \|f\|_n, \cdots, \|f\|_N\}$$

to its asymptotic limit, F_{MAX}. I have devised an efficient procedure that converges to the limit. It was obtained by performing a Taylor series expansion about $p = 1$ and involves a fair amount of 'handle turning'. The derivation of that procedure will be the topic of the next article.

The means by which the data set is obtained and the computational cost involved is the subject of this first article. This is in order to demonstrate that CHEBYCHOP is feasible and fast.

Let $\|f\|_n = \mathcal{I}_n^{1/n}$, where $\mathcal{I} = \int_{\mathcal{D}} f(\boldsymbol{x})^n \mathrm{d}\mu$. The extrapolation procedure requires the condition, $f(\boldsymbol{x}) \geq 1$. This is obtained through a linear transformation of the original function, given easily obtainable upper and lower bounds for it. Furthermore, the process of converting a minimization problem into a maximization problem also involves a linear transformation.

3 Classes of test problems

A good example of a standard test problem is the Rosenbrook function constrained to a disk [1]:

$$f(x, y) = (1 - x^2)^2 + 100(y - x^2)^2$$

The domain is a disk defined by $x^2 + y^2 \leq 2$ and the minimum is $f(1.0, 1.0) = 0$. The function is a polynomial and this is likewise for many optimization problems. Furthermore, many optimization problems have functions that can be feasibly approximated by polynomials. Project CHEBYCHOP will initially tackle these types of problems.

The algebraic manipulation involved in forming $f(\boldsymbol{x})^n$ is too much for a manual approach and so computer algebra is required. Contained within this initial article are some estimates of the computational cost of integrating $f(\boldsymbol{x})^n$, assuming 'brute force' polynomial multiplication. A better approach might be to decompose $f(\boldsymbol{x})$ into orthonormal polynomials, through a generalisation of the Legendre polynomials.

The classes of test problems that will be considered in the early stages of Project CHEBYCHOP are restricted to polynomials constrained by three types of domains: 'CIRCLOIDAL', SIMPLEX, or HYPERCUBE. 'CIRCLOIDAL' domains include disks and hyperspheres. The very first class of problems that will be dealt with are polynomials constrained by hypercubes; let this class be called VANILLA. An important sub-problem is finding the integral of a generic term of each polynomial:

$$T = a \int_{\mathcal{D}} \prod_{d=1}^{D} x_d^{n_d} \mathrm{d}x_1 \cdots \mathrm{d}x_D,$$

where $\mathcal{D}$ is a hypercube that needs to be specified in terms of a format.

4 Estimating computational cost

The VANILLA class of optimization problems will be used here, in order to gauge how computational complexity increases as the most fundamental parameters of problems increase.

Clearly, when the domain is a hypercube, the integral of a function can be obtained from the indefinite integrals at the vertices and the number of vertices is 2^D. Let the cost of computing the indefinite integral of a polynomial be C_P and the total cost of computing the integral be C_T multiplications and divisions; clearly, $C_T = 2^D C_P$. The indefinite integral of term T is:

$$a \prod_{d=1}^{D} \frac{x_d^{n_d+1}}{n_d+1}$$

The powers of the independent variables are reusable. This implies that C_P is proportional to the numebr of dimensions, D, i.e. $C_P = \alpha D$, where $\alpha \approx 2$. Therefore the cost of computing the definite integral of a polynomial, C multiplications and divisions, can be expressed by the following formula:

$$C = \alpha D 2^D N_T,$$

where N_T is the number of terms. Let C_n be the cost of computing $\mathcal{I}_n$. Also, let the $n-th$ polynomial have M_n terms, given that the starting polynomial has $M = M_1$ terms. It may be shown that an upper bound on M_n is $M + n - 1$. Let C_n be the cost of computing $\mathcal{I}_n$. A working formula for this is:

$$C_n = \alpha D 2^D (M + n - 1)$$

This formula can be used to obtain a formula for $C_{\mathcal{S}}(D, M, N)$, the cost of obtaining the data set, $\mathcal{S}$:

$$C_{\mathcal{S}}(D, M, N) = \alpha D 2^D (M + N(N+1)/2 - 1)$$

This then needs to be multiplied by an estimate of the number of iterations or 'chops', Ch, in order to gauge the cost of solving a problem; let this total cost be C_{TOTAL}. The number of 'chops' will be roughly proportional to the number of dimensions, D, so let $Ch = kD$. Therefore, a formula for estimating C_{TOTAL} is:

$$C_{TOTAL} = \alpha k D^2 2^D (M + N(N+1)/2 - 1)$$

This formula indicates low computational cost, unless D is large. To obtain an idea of how it grows, let $\alpha = 2$, $k = 100$ and $M = 10$. Hence $C_{TOTAL}(D, N) = 200 D^2 2^D (9 + N(N+1)/2)$.

$C_{TOTAL}(1, 5) = 9,600$
$C_{TOTAL}(1, 20) = 87,600$
$C_{TOTAL}(5, 5) = 3.8 \times 10^6$
$C_{TOTAL}(5, 20) = 3.5 \times 10^7$

$C_{TOTAL}(10, 5) = 4.8 \times 10^8$
$C_{TOTAL}(10, 20) = 4.5 \times 10^9$
$C_{TOTAL}(20, 5) = 2 \times 10^{12}$
$C_{TOTAL}(20.20) = 1.8 \times 1.8^{13}$

The above calculations are for a range of problems from small to very large and the costs are feasible using current computing technology.

5 Conclusions

Project CHEBYCHOP is new and non-commercial. It has the aim of putting theory into practise through software development. Estimates of computational cost indicate that CHEBYCHOP is very fast. However, the amount of software development could be considerable. The intention is to use computer algebra software, such as Symbolic C++, in order to reduce the amount of software development required. A useful introduction to Symbolic C++ is [2].

The author would welcome feedback and any offers of collaboration and co-publication! As mentioned in the section on the extrapolation process, the next article on vixra.org will focus on how to efficiently converge to F_{MAX}, using a small data set, $\mathcal{S}_N$.

References

[1] `https://en.wikipedia.org/wiki/Test_functions_for_optimization`.

[2] Tan Kiat Shi, Willi-Hans Steeb and Yorick Hardy, *Symbolic C++: An introduction to Computer Algebra using Object-Oriented Programming*, second edition, Springer (2000).