Antanas Žilinskas

Algorithm. 44. MIMUN. Optimization of one-dimensional multimodal functions in the presence of noise

_Aplikace matematiky_, Vol. 25 (1980), No. 3, 234–240

Persistent URL: [http://dml.cz/dmlcz/103855](http://dml.cz/dmlcz/103855)

Terms of use:

© Institute of Mathematics AS CR, 1980

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these Terms of use.
OPTIMIZATION OF ONE-DIMENSIONAL MULTIMODAL FUNCTIONS IN THE PRESENCE OF NOISE

ANTANAS ŽILINSKAS

Institute of mathematics and cybernetics, Academy of sciences of the Lithuanian SSR, Lenino pr. 3, 232600, Vilnius, USSR

The problems of minimization in the presence of noise occur in various fields of science and engineering. But, as far as it is known to the author, among the currently available issues there is not a single publication of a computer program for solving such problems. A rather efficient and quick-operating algorithm for one-dimensional multimodal minimization in the presence of noise is proposed in [1]. This algorithm is based on the usage of a Wiener process for a statistical model of an objective function [2]. The results of investigation of a former version of this algorithm are given in [3], [4]. The algorithm for one-dimensional multimodal minimization without noise based on similar assumptions [5] is more efficient than other algorithms of analogous destination as shown in [6]. Only a brief description of this algorithm is given here to explain the meaning of formal parameters while its full description is given in [1].

Let a function \( f(x), a \leq x \leq b, \) be minimized where only the values \( z(x_i) = f(x_i) + \xi_i \) may be observed where \( \xi_i \) are independent Gaussian random numbers (noise), whose mean is equal to zero and the dispersion is \( \sigma_i^2, i \) being the number of observation. Before the minimization the variance analysis of the results \( z_{ij} \) is carried out, where \( z_{ij} = z(x(i)), x(i) = a + (b - a)(i - 1)/(m - 1), i = 1, ..., m, j = 1, ..., m_3, \) i.e. at each point \( x(i) \) the objective function \( z(\cdot) \) is observed \( m_3 \) times. If the hypothesis of equality of \( f(x(i)) \) is accepted (the significance level being equal to 0.05) then the algorithm terminates indicating that the noise level is too high. If the variance of \( f(\cdot) \) is significant then the dispersion of noise and the parameter of a Wiener process, chosen as a statistical model of an objective function, are estimated [1]. After that the minimization begins. To simplify the algorithm the lattice \( x^i = a + (b - a)(i - 1)/(m - 1), i = 1, ..., m, \) is substituted for the interval of
minimization \([a, b]\). The additional error caused by discretization of \([a, b]\) may obviously be reduced to a desirable value choosing sufficiently large \(m_4\): the value \(m_4 = 101\) is large enough for many practical problems. The coordinate of the current observation is defined by the condition of maximum of the expected improvement \([1, 3]\). The algorithm terminates if the number of objective function evaluations reaches the maximally allowable amount or if the \(P\)-probability of evaluating the global minimum with given accuracy \(\varepsilon_1\) exceeds 0.9; this probability is calculated according to the chosen statistical model of objective function \([1]\). Note that in the case when \(\sigma_i/\varepsilon_1 \geq 10\) and \(\sigma_i\) is of the same order as variance of \(f(\cdot)\), more than a thousand observations of an objective function are necessary for \(P\) to reach 0.9. On the other hand, if the noise level is not so high, a practically acceptable solution is usually obtained after 200—500 observations \([1]\).

Using this algorithm the following remarks must be taken into account:

1. The variable \(k_{\text{max}}\) is the machine dependent constant which is initialised as 19. If the maximal real number of the user’s computer is \(10^k\) where \(k < 19\) then the value of the variable \(k_{\text{max}}\) must be set equal to \(k\).

2. The formal parameter \(\text{ifault}\) is the failure indicator. The normal termination of the algorithm is indicated by \(\text{ifault} = 0\). If \(\text{ifault} = 1\) then the cause of termination of the algorithm is a too high level of noise. \(\text{ifault} = 2\) means that the number of observations reaches the maximally allowable value. The scale of values of an objective function must be chosen so that \(|f(x)|\) does not exceed \(10^K\) where \(K = k_{\text{max}}/2\); the violation of this condition is indicated as \(\text{ifault} = 3\). The termination with \(\text{ifault} = 4\) means that the variance of the objective function is insignificant as shown by the results before the minimization investigation; the scale of the values of an objective function must be changed or the different number of points \(m_2\) must be taken (for example, \(2m_2 + 1\)).

3. The algorithm calls the auxiliary \textbf{real procedure \textit{ndtr}}, which is the ALGOL version of \textit{SUBROUTINE NDTR} \([7]\) and which calculates the value of the Gaussian distribution function.

Algorithm \textit{minun}:

\begin{verbatim}
procedure minun(be, en, sn1, am, m, f, e1, e, nf, xm, ym, ifault, anm, b, c, y);
  comment: be ... input ... start of interval of optimization,
            en ... input ... end of interval of optimization,
            sn1 ... input ... if(sn1 > 0) then sn1 is variance of noise,
                             else variance of noise to be evaluated by algorithm,
            am ... input ... if am = 1 then minimization,
                             if am = -1 then maximization,
\end{verbatim}
\section*{Optimization Parameters}

\begin{itemize}
    \item \texttt{m} \ldots input \ldots \texttt{m}[1] \ldots maximal allowed number of observations of objective function \texttt{f},
    \item \texttt{m}[2] \ldots number of observation points for parameters estimation, it is recommended = 6,
    \item \texttt{m}[3] \ldots number of observations at each point for parameters estimation, it is recommended = 5,
    \item \texttt{m}[4] \ldots number of points of lattice, it is recommended = 101,
\end{itemize}

\texttt{f} \ldots input \ldots objective function,

\texttt{e1} \ldots input \ldots if(e1 \geq 0) then \texttt{e1} is required accuracy of \texttt{ym},
if(e1 < 0) then required accuracy is equal to \texttt{sqrt(variance of noise/abs(e1))},

\texttt{e} \ldots output \ldots estimation of mean-root-square error of \texttt{ym},
\texttt{nf} \ldots output \ldots number of observations of \texttt{f},
\texttt{xm} \ldots output \ldots estimation of optimum point,
\texttt{ym} \ldots output \ldots estimation of optimum,
\texttt{ifault} \ldots output \ldots failure indicator

\texttt{anm}, \texttt{b}, \texttt{c}, \texttt{y} \ldots workspace, dimension of these arrays \geq \texttt{m}[4];

\begin{verbatim}
value be, en, sn1, am, e1; integer nf, ifault;
real be, en, sn1, am, e1, xm, ym; integer array m;
array anm, b, c, y; real procedure f;
begin integer n, kmax, k, k1, n1, n2, j, k3, k4, km, km2;
real dt, cv, eps2, an, pp, amax, ym1, p1, p2, p3, p4, p5, p6, cv2, ym2, sn2,
aw, av1, av2, am1, v1, v2, c1, c2, pr, pr1, ppab, va, d, d1,
a11, a12, a21, a22, eb, sf, eps1, eps3;
real procedure av(k, co);
comment: auxiliary procedure for mimun: calculates conditional mean and variance of Wiener process;
integer k; real co;
begin integer i; real a, p;
a := y[k]; p := b[k]; co := 1.0; i := k;
for i := i - 1 while co \times eps2 < p \land i > 0 do
if anm[i] > 0.0 then begin a := a + p \times y[i]; co := co + p; p := p \times b[i]
end;
p := c[k]; i := k;
for i := i + 1 while co \times eps2 < p \land i \leq n do
if anm[i] > 0.0 then begin a := a + p \times y[i]; co := co + p; p := p \times c[i]
end;
av := a/\texttt{co};
end \texttt{av};
real procedure \texttt{fi}(x, nr);
\end{verbatim}

value \( x \), \( nr \); integer \( nr \); real \( x \);

comment: auxiliary procedure for minun;

begin integer \( k \); real \( a \);

\( a := 0.0; \) for \( k := 1 \) step 1 until \( nr \) do \( a := a + f(x); \)

end fi;

procedure updata;

comment: auxiliary procedure for minun:

updates array of parameters \( c, b \);

begin integer \( k, k1, kp, kp1 \); real \( b, bs, cs \);

\( kp := 1; \) \( kp1 := n; \) \( bs := cs := b[1] := c[n] := 1.0; \)

for \( k := 2 \) step 1 until \( n \) do

begin if \( anm[k] > 0.0 \) then

begin \( b1 := dt((dt/anm[kp] + (k - kp) \times an \times bs) \times anm[k]); \)

\( bs := bs \times b1 + 1.0; \) \( b[k] := b1; \) \( kp := k \)

end;

\( k1 := n + 1 - k; \) if \( anm[k1] > 0.0 \) then

begin \( b1 := dt((dt/anm[kp1] + (kp1 - k1) \times an \times cs) \times anm[k1]); \)

\( cs := cs \times b1 + 1.0; \) \( c[k1] := b1; \) \( kp1 := k1 \)

end

end updata;

real procedure ndtr(x);

value \( x \); real \( x \);

comment: Gaussian distribution function, algol version of subroutine ndtr:

system/360 scientific subroutine package;

begin real \( t, d, p, ax; \) \( ax := abs(x); \) \( t := 1.0/(1.0 + 0.2316419 \times ax); \)

\( d := 0.3989423 \times \exp((-x \times x/2.0); \) \( p := 1.0 - d \times t \times (((1.330274 \times t - 1.821256) \times t + 1.781478) \times t - 0.3565638) \times t + 0.3193815); \)

if \( x > 0 \) then \( ndtr := p \) else \( ndtr := 1.0 - p \)

end Any other procedure of analogous destination may be used instead of \( ndtr; \)

\( kmax := 19; \) \( pp := 2.0; \) \( eps2 := 0.001; \) \( ifault := 0; \)

\( n := m[4]; \) \( n1 := m[2]; \) \( n2 := m[3]; \) \( cv := 1.0; \) \( amax := 10 \uparrow (kmax \div 2 - 1); \)

\( eb := (en - be)/(n - 1); \) \( p2 := p5 := p6 := 0.0; \) \( ym1 := amax; \) \( an := 1/(n - 1); \)

for \( k := 1 \) step 1 until \( n \) do begin \( y[k] := 0.0; \) \( anm[k] := -1.0/amax \) end;

for \( k := 1 \) step 1 until \( n \) do

begin \( p3 := 0.0; \) for \( k1 := 1 \) step 1 until \( n2 \) do

begin \( p4 := f1(be + eb \times (((n - 1) \times (k - 1)) \div (n1 - 1)), 1); \)

if \( abs(p4) < amax \) then begin \( p3 := p3 + p4; \) \( p2 := p2 + p4 \times p4 \) end

else begin \( ifault := 3; \) go to fn \) end

end;

\( y[k] := p3/n2; \) if \( ym1 > p3 \) then \( ym1 := p3; \)

\( p5 := p5 + p3; \) \( p6 := p6 + p3 \times p3 \)
end;
nf := n_1 \times n_2; p5 := p5 \times p5/nf; p6 := p6/n2; sf := abs(p6 - p5)/(n1 - 1);
if sn1 > 0.0 then sn2 := sn1 else sn2 := abs(p2 - p6)/(n1 \times (n2 - 1));
sf := abs(p6 - p5)/(n1 - 1);
if sf < sn2 \times 2.5 then begin ifault := 1; go to fin end;
comment: estimation of parameters;
p1 := y[1]; cv2 := 0.0;
for fc := 2 step 1 until n1 do
begin p2 := y[fc];
   cv2 := cv2 + (p2 - p1) \times 2; p1 := p2
end;
cv := sqrt(cv2); if cv < 1.0/amin then begin ifault := 4; go to fin end;
dt := sn2/cv2;
for k := 1 step 1 until n1 do
begin k1 := ((n - 1) \times (k - 1)) \div (n1 - 1) + 1;
y[k1] := y[k]/cv; anm[k1] := n2
end;
if e1 > 0.0 then eps3 := e1/cv else eps3 := sqrt(dt/abs(e1));
eps1 := eps3/pp;
comment: begin of optimization;
ym1 := ym1/(cv \times n2); vl := 0.0; update;
lopt: ppab := 1.0; if vl \geq eps1 then ppab := 0.0; ym2 := av2 := av(1, c2);
k2 := k2 := 1; pr := 0.0;
comment main loop, computing of point of current observations;
for k := 1 step 1 until n do
begin if k < n \land anm[k] > 0.0 then
   begin c1 := c2; av1 := av2; k3 := k;
      for j := k + 1 step 1 until n do if anm[j] > 0.0 then
         begin av2 := av(j, c2); if av2 < ym2 then
            begin ym2 := av2; k2 := j
            end;
      end;
   end;
k4 := j; a11 := 1.0/c1; a12 := a11 \times b[k3]; a21 := a11 \times c[k3];
a22 := 1.0/c2; go to l1
end
end;
l1: d := (k - k3)/(k4 - k3); d1 := (1 - d); aw := av1 \times d1 + av2 \times d;
va := sqrt(d \times d1 \times (k4 - k3)/(n - 1) + (d1 \times (d1 \times a11 + d \times a21)/
anm[k3] + d \times (d1 \times a12 + d \times a22)/anm[k4] \times dt);
if k = k2 then v2 := va;
am1 := ym1 - aw; p1 := -0.2 \times am1;
comment: computing of probability of finding
global optimum with required accuracy ppab;
if vl < eps1 \land va \geq 1.5 \times vl \land ppab \geq 0.9 then
238
ppab := ppab × (1 − ndtr((am1 − eps3)/va));
comment computing of mean improvement;
va := va × 7.0; if va > p1 then
    begin pr1 := am1 × 0.65 × exp(-0.443 × (0.75 − am1/va) ↑ 2) +
          va × 0.3989 × exp(-(am1 × am1)/(2.0 × va × va));
    if pr1 > pr ∧ va > eps3 then begin km := k; pr := pr1 end
end
end main loop;
if ppab ≥ 0.9 then go to l2; d := ann[m][km];
j := 0.1 × d + 1.0; d1 := d + j; p4 := fi(be + eb × (km − 1), j);
if abs(p4) > amax then begin ifault := 3; go to fin end;
y[km] := (y[km] × d + am × j × p4)/d1; ann[m][km] := d1;
nf := nf + j; ym1 := ym2; v1 := v2; update;
if nf < m[1] then go to lopt; ifault := 2;
l2: ym := ym2 × cv/am; xm := be + (km2 − 1) × eb; e := v2 × pp × cv;
fin:
end;
Example: The test function:
real procedure f(x); value x; real x;
comment: test function for mimun, integer
parameter kun must be declared in driver program
and initialised there as kun = 127;
begin real a, b; integer i;
comment: generation of pseudo-random number a;
kun := kun × 3125; kun := kun − entier(kun/67108864) × 67108864;
a := kun/33554432 − 1.0; b := 0.0;
for i := 1 step 1 until 5 do
    b := b − i × sin((i + 1) × x + i);
f := a + b
end
was minimized with the input parameters: be = −10.0, en = 10.0, am = 1.0,
following results were obtained (computer BESM-6):
xm = 5.800000000, ym = −12.07391983, nf = 86, e = 0.2136057320, ifault = 0.
The FORTRAN codes of this algorithm are available from the author.

References