ЧИСЛЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ

Константин Ловецкий

Сентябрь 2012

Кафедра систем телекоммуникаций

Методы прямого поиска

В типичном методе поиска направления минимизации полностью
определяются на основании последовательных вычислений целевой
функции f(x).

При решении задач нелинейного программирования при отсутствии
ограничений градиентные методы и методы, использующие вторые
производные, сходятся быстрее, чем прямые методы поиска. Тем не
менее, применяя на практике методы, использующие производные,
приходится сталкиваться с двумя главными препятствиями.

Во-первых, в задачах с достаточно большим числом переменных
довольно трудно или даже невозможно получить производные в виде
аналитических функций, необходимых для градиентного алгоритма или

Методы прямого поиска

Методы поиска не требуют ***регулярности и непрерывности*** целевой функции и существования производных.

- **Голоморфная функция**, также называемая **регулярной функцией** — функция комплексного переменного, определённая наоткрытом подмножестве комплексной плоскости и комплексно дифференцируемая в каждой точке.
- В отличие от вещественного случая, это условие означает, что функция бесконечно дифференцируема и может быть представлена сходящимся к ней рядом Тейлора.
- Голоморфные функции также называют иногда *аналитическими*, хотя второе понятие гораздо более широкое, так как аналитическая функция не обязана быть определена на множестве комплексных чисел. Тот факт, что для комплекснозначных функций комплексной переменной множества голоморфных и аналитических функций совпадают, является нетривиальным и весьма замечательным результатом комплексного анализа.

Методы прямого поиска

Вторым обстоятельством, правда, связанным с предыдущей проблемой, является то, что при использовании методов оптимизации, основанных на вычислении первых и при необходимости вторых производных, требуется по сравнению с методами поиска довольно большое время на подготовку задачи к решению.

Алгоритмы оптимизации, использующие прямой поиск, хотя и медленнее реализуются в случае простых задач, на практике могут оказаться более удовлетворительными с точки зрения пользователя, чем градиентные методы или методы, использующие вторые производные, и решение задачи с их помощью может обойтись дешевле, если стоимость подготовки задачи к решению высока по сравнению со стоимостью машинного времени.

Методы прямого поиска

Методы поиска простейшего типа заключаются в изменении каждый раз

одной переменной, тогда как другие остаются постоянными, пока не x_1 будет

достигнут минимум. Например, x_2 в одном из таких методов переменная

устанавливается постоянной, а x_2 изменяют до тех пор, x_1 пока не будет x_2

получен минимум.

Затем, x_1 x_2 сохраняя новое значение постоянным, изменяют пока

не будет достигнут оптимум при выбранном x_1 x_2 значении и т. д.

Однако такой алгоритм работает плохо, если имеет место взаимодействие

между и , т. е., например, если в выражение для целевой функции

Алгоритм Хука и Дживса

Pattern search (PS) refers to a family of numerical optimization methods that do not require the gradient of the problem to be optimized and PS can hence be used on functions that are not continuous or differentiable. Such optimization methods are also known as direct-search, derivative-free, or black-box methods

The name, pattern search, was coined by Hooke and Jeeves [1]. An early and simple PS variant is attributed to Fermi and Metropolis when they worked at the Los Alamos National Laboratory as described by Davidon [2] who summarized the algorithm as follows:

They varied one theoretical parameter at a time by steps of the same magnitude, and when no such increase or decrease in any one parameter further improved the fit to the experimental data, they halved the step size and repeated the process until the steps were deemed sufficiently small.

Hooke, R.; Jeeves, T.A. (1961). "'Direct search' solution of numerical and statistical problems". *Journal of the Association for Computing Machinery (ACM)* **8** (2): 212–229.

Алгоритм Хука и Дживса

Хук и Дживс предложили логически простую стратегию поиска,
использующую априорные сведения и в то же время отвергающую
устаревшую информацию относительно характера топологии целевой E^n
функции в . Алгоритм включает два основных этапа: «исследующий
поиск» вокруг базисной точки и «поиск по образцу», т. е. в x направлении,
выбранном для минимизации. Δx
Прежде всего задаются начальные значения всех элементов $f(x)$, а также
начальное приращение . Чтобы начать «исследующий поиск», следует
вычислить значение функции в базисной точке (базисная точка
представляет собой начальный вектор предполагаемых искомых

Алгоритм Хука и Дживса

Затем в циклическом порядке изменяется каждая переменная (каждый раз
только одна) на выбранные величины приращений, пока все параметры не
будут таким образом изменены.

В частности, $x_1^{(0)}$ изменяется на величину $\Delta x_1^{(0)}$, так что $x_1^{(1)} = x_1^{(0)} + \Delta x_1^{(0)}$

Если приращение не улучшает целевую функцию, $x_1^{(0)}$ изменяется на $-\Delta x_1^{(0)}$ и значение $f(x)$ проверяется, как и ранее. $x_1^{(1)} = x_1^{(0)} - \Delta x_1^{(0)}$

Если значение $x_1^{(0)}$ не улучшают ни $x_1^{(1)}$, ни $x_2^{(0)}$, то $\Delta x_2^{(0)}$ оставляют без изменений.

Затем $x_2^{(0)}$ изменяют на величину $\Delta x_2^{(0)}$ и т, д., пока не будут изменены все независимые переменные, что завершает один исследующий поиск.

Исследующий поиск

На каждом шаге или сдвиге по независимой переменной значение целевой

функции сравнивается с ее значением в предыдущей точке. Если целевая

функция улучшается на данном шаге, то ее старое значение x заменяется на $f(x)$

новое при последующих сравнениях. Однако если проведенное возмущение

по неудачно, то сохраняется прежнее значение .

После проведения ($f(x)$ *exploration step*) ***исследующего*** поиска применяется

стратегия поиска по образцу. Удачные изменения переменных в

исследующем поиске [т. е, те изменения переменных, $f(x)$

которые уменьшили] определяют вектор в , указывающий

некоторое направление минимизации, которое может привести к

Исследующий поиск и поиск по образцу

Длина шага при поиске по образцу в данном координатном направлении

приблизительно пропорциональна числу удачных шагов, имевших место

ранее в этом координатном направлении во время исследующих поисков за Δx

несколько предыдущих циклов.

Для ускорения процесса оптимизации изменение размера шага Δx в

поиске по образцу осуществляется путем введения некоторого множителя

при величине , используемой в исследующих поисках. Исследующий

поиск, проводимый после поиска по образцу, называется исследующим

поиском типа II.

Успех или неудачу поиска по данному образцу нельзя установить до

завершения исследующего поиска типа II.

Алгоритм Хука и Дживса

Один из алгоритмов минимизации (библиотека МГУ) использует метод

Хука - Дживса для решения задачи минимизации функции многих переменных без вычисления производных при наличии двухсторонних ограничений на переменные методом покоординатного спуска с построением аппроксимирующей параболы.

http://www.srcc.msu.su/num_anal/lib_na/cat/mn/mnn1r.htm

Тексты программ приводятся на фортране, Си и паскале.

Немного отличный от этого метод приводится в книге:
Alfio Quarteroni, Riccardo Sacco, Fausto Saleri. Numerical Mathematics. Springer, 2nd ed., 2007.

Алгоритм Хука и Дживса -II

Assume we are searching for the minimizer of ***f*** *starting from a given initial* point **x(0)** and requiring that the error on the residual is less than a certain fixed tolerance ε. The Hooke and Jeeves method computes a new point **x(1)** using the values of ***f*** *at suitable points along the orthogonal coordinate directions* around **x(0).**

The method consists of two steps: an *exploration step and* an *advancing step.*

The exploration step starts by evaluating $f(x^{(0)} + h_1 e_1)$, where e_1 is the first vector of the canonical basis of R^n *and* h_1 *is a positive real number to* be suitably chosen.

Алгоритм Хука и Дживса -II

If $f(x^{(0)} + h_1 e_1) < f(x^{(0)})$, *then a success is recorded and the starting* point is moved in $x^{(0)} + h_1 e_1$, *from which an analogous check is carried out* at point $x^{(0)} + h_1 e_1 + h_2 e_2$ *with* $h_2 \in R^+$.

If, instead, $f(x^{(0)} + h_1 e_1) \geq f(x^{(0)})$, *then a failure is recorded and a* similar check is performed at $x^{(0)} - h_1 e_1$. *If a success is registered, the method* explores, as previously, the behavior of ***f*** *in the direction* e_2 *starting from this* new point, while, in case of a failure, the method passes directly to examining direction e_2, keeping $x^{(0)}$ as starting point for the exploration step.

Алгоритм Хука и Дживса -II

To achieve a certain accuracy, the step lengths h_i *must be selected in such* a way that the quantities

$$\left| f(x^{(0)} \pm h_j e_j) - f(x^{(0)}) \right|, \quad j = 1, \ldots, n$$

have comparable sizes.

The exploration step terminates as soon as all the *n Cartesian directions* have been examined. Therefore, the method generates a new point, $y^{(0)}$, after at most 2*n + 1 functional evaluations.*

Only two possibilities may arise:

1. $y^{(0)} = x^{(0)}$. In such a case, if $\max_{i=1,\ldots n} h_i \le \varepsilon$ *the method terminates* and yields the approximate solution $x^{(0)}$.

Otherwise, the step lengths h_i are halved and another exploration step is performed starting from $x^{(0)}$;

Алгоритм Хука и Дживса -II

2. $y^{(0)} \neq x^{(0)}$ **. If** $\max_{i=1,\ldots n} |h_i| < \varepsilon$ *then the method terminates yielding* $y^{(0)}$ *as*

an approximate solution, otherwise the advancing step starts.

The advancing step consists of moving further from $y^{(0)}$ along the direction $y^{(0)} - x^{(0)}$, (which is the direction that recorded the maximum decrease of ***f*** *during* the exploration step), rather then simply setting $y^{(0)}$ as a new starting point. This new starting point is instead set equal to $x^{(1)}$. *From this point* a new series of exploration moves is started. If this exploration leads to a point $y^{(1)}$ such that $f(y^{(1)}) < f(y^{(0)})$, *then a new starting point* for the next exploration step has been found, otherwise the initial guess for further explorations is set equal to $x^{(0)}$.

The method is now ready to restart from the point $x^{(1)} = y^{(1)} - y^{(0)}$ just computed.

Алгоритм Хука и Дживса

function [x,minf,iter]=hookejeeves(f,n,h,x0,tol)
{HOOKEJEEVES HOOKE and JEEVES method for function minimization.
[X, MINF, ITER] = HOOKEJEEVES(F, N, H, X0, TOL) attempts to compute the minimizer of a function of N variables with the Hooke and Jeeves method. F is a string variable containing the functional expression of f. H is an initial step. X0 specifies the initial guess. TOL specifies the tolerance of the method. ITER is the iteration number at which X is computed. MINF is the value of F at the mimimizer X.}

```
x = x0; minf = eval(f); iter = 0;
while h > tol
   [y] = explore(f,n,h,x);
   if y == x
      h = h/2;
  else
    x = 2*y-x;
   [z] = explore(f,n,h,x);
   if z == x
     x = y;
   else
     x = z;
   end {if z == x}
  end {if y == x}
  iter = iter +1;
end {while}
minf = eval(f);
return
```

Алгоритм Хука и Дживса

function [x]=explore(f,n,h,x0)
{ EXPLORE Exploration step for function minimization.
[X] = EXPLORE(F, N, H, X0) executes one exploration step of size H in the Hooke and Jeeves method for function minimization.}

```
x = x0; f0 = eval(f);
for i=1:n
  x(i) = x(i) + h(i); ff = eval(f);
  if ff < f0
    f0 = ff;
  else
    x(i) = x0(i) - h(i);
    ff = eval(f);
    if ff < f0
      f0 = ff;
    else
      x(i) = x0(i);
    end {if ff < f0}
  end {if ff < f0}
end {for}
return
```

Hooke and Jeeves -I

Program 60 - hookejeeves : The method of Hooke and Jeeves (HJ)

```
function [x,minf,iter]=hookejeeves(f,n,h,x0,tol)
%HOOKEJEEVES HOOKE and JEEVES method for function minimization.
%  [X, MINF, ITER] = HOOKEJEEVES(F, N, H, X0, TOL) attempts to compute the
%  minimizer of a function of N variables with the Hooke and Jeeves method. F is
%  a string variable containing the functional expression of f. H is an initial
%  step. X0 specifies the initial guess. TOL specifies the tolerance of the method.
%  ITER is the iteration number at which X is computed. MINF is the value of F at
%  the mimimizer X.
x = x0; minf = eval(f); iter = 0;
while h > tol
    [y] = explore(f,n,h,x);
    if y == x
        h = h/2;
    else
        x = 2*y-x;
        [z] = explore(f,n,h,x);
        if z == x
            x = y;
        else
            x = z;
        end
    end
    iter = iter +1;
end
minf = eval(f);
return
```

Hooke and Jeeves -II

Program 61 - explore : Exploration step in the HJ method

```
function [x]=explore(f,n,h,x0)
%EXPLORE Exploration step for function minimization.
%  [X] = EXPLORE(F, N, H, X0) executes one exploration step of size H in the Hooke
%  and Jeeves method for function minimization.
x = x0; f0 = eval(f);
for i=1:n
    x(i) = x(i) + h(i);  ff = eval(f);
    if ff < f0
        f0 = ff;
    else
        x(i) = x0(i) - h(i);
        ff = eval(f);
        if ff < f0
            f0 = ff;
        else
            x(i) = x0(i);
        end
    end
end
return
```

Example – Tests for Simplex and Hooke and Jeeves methods

Example 7.5 Let us compare the performances of the Simplex method with the Hooke and Jeeves method, in the minimization of the Rosembrock function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{7.24}$$

This function has a minimizer at $[1, 1]^T$ and represents a severe benchmark for testing numerical methods in minimization problems. The starting point for both methods is set equal to $\mathbf{x}^{(0)} = [-1.2, 1]^T$, while the step sizes are taken equal to $h_1 = 0.6$ and $h_2 = 0.5$, in such a way that (7.23) is satisfied. The stopping tolerance on the residual is set equal to 10^{-4}. For the implementation of Simplex method, we have used the MATLAB function `fmins`.

Figure 7.2 shows the iterates computed by the Hooke and Jeeves method (of which one in every ten iterates have been reported, for the sake of clarity) and by the Simplex method, superposed to the level curves of the Rosembrock function. The graph demonstrates the difficulty of this benchmark: actually, the function is like a curved, narrow valley, which attains its minimum along the parabola of equation $x_1^2 - x_2 = 0$.

The Simplex method converges in only 165 iterations, while 935 are needed for the Hooke and Jeeves method to converge. The former scheme yields a solution equal to $[0.999987, 0.999978]^T$, while the latter gives the vector $[0.9655, 0.9322]^T$. •

Examples for Simplex and Hooke and Jeeves methods

http://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method

http://en.wikipedia.org/wiki/Pattern_search_(optimization)

http://www.serc.iisc.ernet.in/~amohanty/SE288/hja.html