Bi7740: Scientific computing

Optimization: a brief summary

Vlad Popovici
popovici@iba.muni.cz

Institute of Biostatistics and Analyses
Masaryk University, Brno

Book:
Venkataraman P., Applied optimization using Matlab, Wiley & Sons, 2002

Outline

Problem setting

- minimization problem: $f : \mathbb{R}^n \to \mathbb{R}$, $S \subseteq \mathbb{R}^n$, find $\mathbf{x}^* \in S$: $f(\mathbf{x}) \leq f(\mathbf{y}), \forall \mathbf{y} \in S \setminus \{\mathbf{x}\}$
- $\mathbf{x}^*$ is called minimizer (minimum, extremum) of f
- maximization is equivalent to minimizing $-f$
- f is called objective function and considered, *here*, differentiable with continuous second derivative
- constraint set S (or feasible region) is defined by a system of equations and/or inequations
- $\mathbf{y} \in S$ is called a feasible point
- if $S = \mathbb{R}^n$ the optimization is unconstrained

Optimization problem

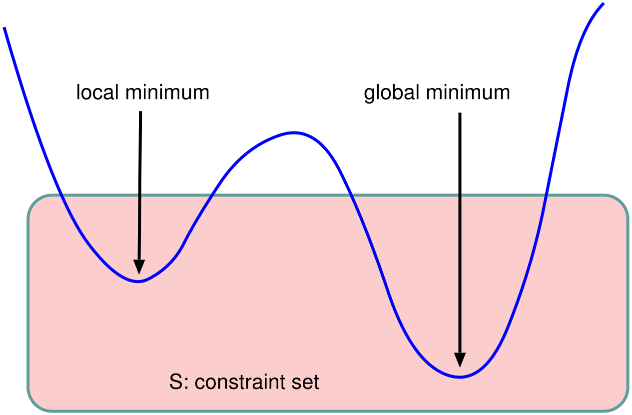
$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}$$
$$h_k(\mathbf{x}) \leq 0$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$, $h_k : \mathbb{R}^n \to \mathbb{R}$.

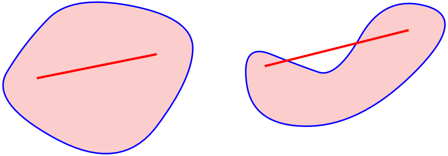
If f, $\mathbf{g}$ and $\mathbf{h}_k$ functions are linear: linear programming.


local minimum
global minimum
S: constraint set

Some theory

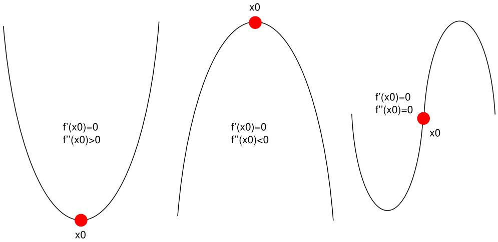
- *Rolle's thm*: f cont. on $[a, b]$ and differentiable on (a, b) with $f(a) = f(b)$, then $\exists c \in (a, b) : f'(c) = 0$
- *Weierstrass' thm*: f cont. on a compact set with values in a subset of $\mathbb{R}$ attains its extrema
- *Fermat's thm*: $f : (a, b) \to \mathbb{R}$ then in a stationary point $x_0 \in (a, b)$, $f'(x_0) = 0$. Generalization: $\nabla f(\mathbf{x}_0) = 0$.
- convex function: $f''(x) > 0$; concave function: $f''(x) < 0$
- if $f'(x_0) = 0$ and $f''(x_0) < 0$ then x_0 is a minimizer
- if $f'(x_0) = 0$ and $f''(x_0) > 0$ then x_0 is a maximizer
- if $f'(x_0) = f''(x_0) = 0$, then x_0 is an inflection point

Set convexity



Formally: a set S is convex if $\alpha x_1 + (1 - \alpha x_2) \in S$ for all $x_1, x_2 \in S$ and $\alpha \in [0, 1]$.

Function convexity



Formally: f is said to be convex on a convex set S if $f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$ for all $x_1, x_2 \in S$ and $\alpha \in [0,1]$.

Uniqueness of the solution

- any local minimum of a convex function f on a convex set $S \subseteq \mathbb{R}^n$ is global minimum of f on S
- any local minimum of a *strictly* convex function f on a convex set $S \subseteq \mathbb{R}^n$ is unique global minimum of f on S

Optimality criteria

For $\mathbf{x}^* \in S$ to be an extremum of $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$

- first order condition: $\mathbf{x}^*$ must be a *critical point*:

$$\nabla f(\mathbf{x}^*) = 0$$

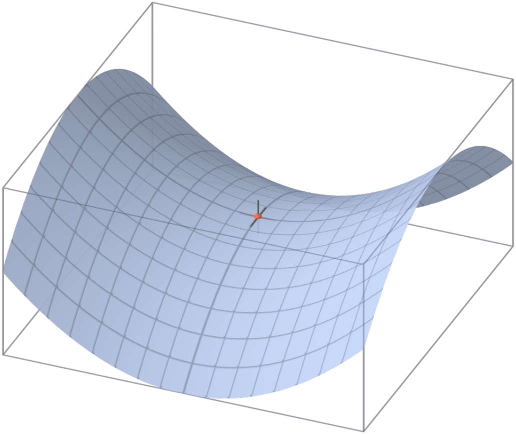
- second order condition: the Hessian matrix $\mathbf{H}_f(\mathbf{x}^*)$ must be positive or negative definite

$$[\mathbf{H}_f(\mathbf{x})]_{ij} = \frac{\partial f(\mathbf{x})}{\partial x_i \partial x_j}$$

If the Hessian is

- positive definite, then $\mathbf{x}^*$ is a minimum of f
- negative definite, then $\mathbf{x}^*$ is a maximum of f
- indefinite, then $\mathbf{x}^*$ is a saddle point of f
- singular, then different degenerated cases are possible...

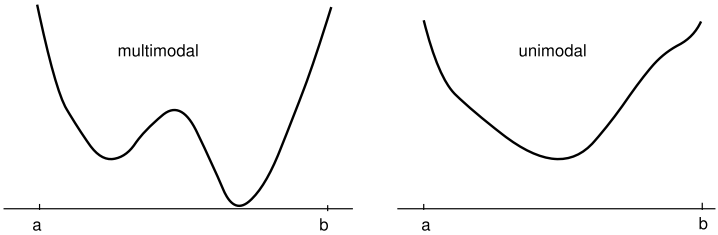
Saddle point



source: Wikipedia

Outline

Unimodality



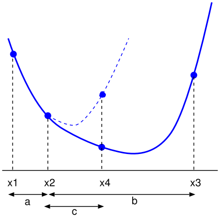
Unimodality allows discarding safely parts of the interval, without loosing the solution (like in the case of interval bisection).

Golden section search

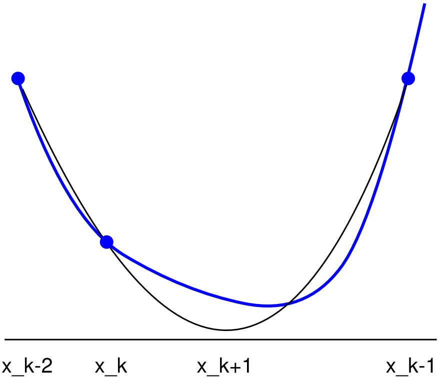
- evaluate the function at 3 points and decide which part to discard
- ensure that the sampling space remains proportional:

$$\frac{c}{a} = \frac{a}{b} \Rightarrow \frac{b}{a} = \frac{1+\sqrt{5}}{2} = 1.618\ldots$$

- convergence is linear, with $C \approx 0.618$



Successive parabolic interpolations



Convergence is superlinear, with $r \approx 1.32$.

Newton's method

From Taylor's series:

$$f(x+h) \approx f(x) + f'(x)h + \frac{f''(x)}{2}h^2$$

whose minimum is at $h = -f'(x)/f''(x)$. HOMEWORK: prove it!
Iteration scheme:

$$x_{k+1} = x_k - f'(x)/f''(x)$$

(That's Newton's method for finding the zero of $f'(x) = 0$.)
Quadratic convergences, but needs to start close to the solution.

Hybrid methods

- idea: combine "slow-but-sure" methods with "fast-but-risky"
- most library routines are using such approach
- popular combination: golden search and successive parabolic interpolation

MATLAB functions for optimization in $\mathbb{R}$

- first, check `optimset` for managing the optimization options
- `fminbnd`: bounded function minimization
- you can use functions for multivariate case as well

Try in MATLAB:

```
>> opts = optimset('display','iter'); % what's for?
>> f = ...
    @(x)(1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6);
>> [x,fx] = fminbnd(f, .2, 1, opts)
>> g = @(x) (cos(x) - 2*log(x));
>> [x, gx] = fminbnd(g, 2, 4, opts); % explain
```

Outline

Outline

Nelder-Mead (simplex) method

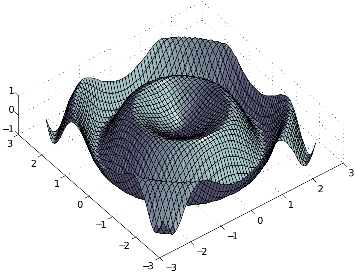
- *direct search* methods simply compare the function values at different points in S
- Nelder-Mead selects $n + 1$ points (in $\mathbb{R}^n$) forming a simplex (i.e. a segment in $\mathbb{R}$, a triangle in $\mathbb{R}^2$, a tetrahedron in $\mathbb{R}^3$, etc)
- along the line from the point with highest function value through the centroid of the rest, select a new vertex
- the new vertex replaces the worst previous point
- repeat until convergence
- useful procedure for non-smooth functions, but expensive for large n

Nelder-Mead in MATLAB

Use the function `fminsearch`.

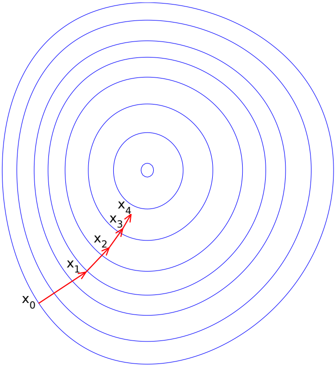
Example:

```
>> f = @(x) (sin(norm(x,2)^2));
>> x = fminsearch(f, [.5,.5], ...
    opts)
>> x = fminsearch(f, ...
    [.25,.25], opts)
>> x = fminsearch(f, [1,1], opts)
```



Steepest descent (gradient descent)

- $f : \mathbb{R}^n \to \mathbb{R}$: the negative gradient, $-\nabla f(\mathbf{x})$ is locally the steepest descent towards a (local) minimum
- $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ where α_k is *line search* parameter



- $\alpha_k = \arg\min_\alpha f(\mathbf{x}_k - \nabla f(\mathbf{x}_k))$
- the method always progresses towards minimum, as long as the gradient is non-zero
- the convergence is slow, the search direction may zig-zag
- the method is "myopic" in its choices

Newton's method

- exploit the 1st and 2nd derivative
- Newton iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_f^{-1}(\mathbf{x}_k)\nabla f(\mathbf{x}_k)$$

- no need to invert the Hessian; solve the system

$$\mathbf{H}_f(\mathbf{x}_k)\mathbf{s}_k = -\nabla f(\mathbf{x}_k)$$

  and then

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$$

- variation: damped Newton method uses a line search along the direction of $\mathbf{s}_k$ to make the method more robust

Newton's method, cont'd

- close to minimum, the Hessian is symmetric positive definite, so you can use Cholesky decomposition
- if initialized far from minimum, the Newton step may not be in the direction of steepest descent:

$$(\nabla f(\mathbf{x}_k))^T \mathbf{s}_k < 0$$

- choose a different direction based on negative gradient, negative curvature, etc

Quasi-Newton methods

- improve reliability and reduce overhead
- general form

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{B}_k^{-1} \nabla f(\mathbf{x}_k)$$

where α_k is a line search parameter and $\mathbf{B}_k$ is an approximation to the Hessian

BFGS (Broyden-Fletcher-Goldfarb-Shanno) method

Algorithm 1: BFGS method

$\mathbf{x}_0$ = some initial value
$\mathbf{B}_0$ = initial approximation of the Hessian
for $k = 0, 1, 2, \ldots$ **do**
  solve $\mathbf{B}_k \mathbf{s}_k = -\nabla f(\mathbf{x}_k)$ for $\mathbf{s}_k$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
  $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
  $\mathbf{B}_{k+1} = \mathbf{B}_k + (\mathbf{y}_k \mathbf{y}_k^T)/(\mathbf{y}_k^T \mathbf{s}_k) - (\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k)/(\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k)$

BFGS, cont'd

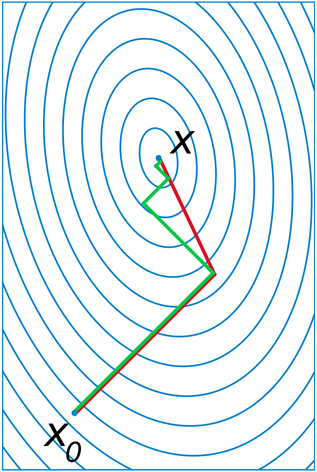
- update only the factorization of $\mathbf{B}_k$ rather than factorizing it at each iteration
- no 2nd derivative is needed
- can start with $\mathbf{B}_0 = \mathbf{I}$
- $\mathbf{B}_k$ does not necessarily converge to true Hessian

Conjugate gradient (CG)

- does not need 2nd derivative, does not construct an approximation of the Hessian
- searches on conjugate directions, implicitly accumulating information about the Hessian
- for quadratic problems, it converges in n steps to exact solution (theoretically)
- two vectors $\mathbf{x}$, $\mathbf{y}$ are *conjugate with respect to a matrix* $\mathbf{A}$ is $\mathbf{x}^T\mathbf{A}\mathbf{y} = 0$
- idea: start with an initial guess $\mathbf{x}_0$ (could be $\mathbf{0}$); go along the negative gradient at the current point; compute the new direction as a combination of previous and new gradients

Algorithm 2: CG method

$\mathbf{x}_0 =$ some initial value
$\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$
$\mathbf{s}_0 = -\mathbf{g}_0$
for $k = 0, 1, 2, \ldots$ **do**
  $\alpha_k = \arg\min_\alpha f(\mathbf{x}_k + \alpha \mathbf{s}_k)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$
  $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
  $\beta_{k+1} = (\mathbf{g}_{k+1}^T \mathbf{g}_{k+1})/(\mathbf{g}_k^T \mathbf{g}_k)$
  $\mathbf{s}_{k+1} = -\mathbf{g}_{k+1} + \beta_{k+1} \mathbf{s}_k$



source: Wikipedia

Other methods

- we barely scratched the surface!
- heuristic methods
- genetic algorithms
- stochastic methods
- hybrid methods
- etc etc etc

MATLAB functions

- linear and quadratic optimization: `linprog`, `quadprog`
- linear least squares: `lsqlin`, `lsqnonneg`
- nonlinear minimization:
  - `fminbnd` - scalar bounded problem;
  - `fmincon` - multidimensional constrained nonlinear minimization
  - `fminsearch` - Nelder-Mead unconstrained nonlinear minimization
  - `fminunc` - multidimensional unconstrained nonlinear minimization
  - `fseminf` -multidimensional constrained minimization, semi-infinite constraints

Outline

1. Problem setting
2. Optimization in $\mathbb{R}$
3. Optimization in $\mathbb{R}^n$
   - Unconstrained optimization in $\mathbb{R}^n$
4. Important classes of optimization problems
   - Linear programming
   - Quadratic programming
   - Constrained nonlinear optimization

Outline

1. Problem setting
2. Optimization in $\mathbb{R}$
3. Optimization in $\mathbb{R}^n$
   - Unconstrained optimization in $\mathbb{R}^n$
4. Important classes of optimization problems
   - Linear programming
   - Quadratic programming
   - Constrained nonlinear optimization

Linear programming (LP)

General form:

$$\text{minimize } \mathbf{f}^T\mathbf{x}$$

subject to

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$
$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$
$$lb \leq \mathbf{x} \leq ub$$

MATLAB:

```
X = linprog(f,A,b,Aeq,beq,LB,UB,X0)
```

LP - Example

Solve the LP:

$$\text{maximize } 2x_1 + 3x_2$$

such that

$$\begin{aligned} x_1 + 2x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 10 \\ x_2 &\leq 3 \end{aligned}$$

```
c = [-2,-3]';
A = [1,2;2,1;0,1];
b = [8,10,3]';
x = linprog(c,A,b,[],[],[],[],[])
```

Chebyshev data approximation

Let (x_i, y_i) be a set of points. Find the best approximation with a d-degree polynomial $p(x) = \alpha_d x^d + \alpha_{d-1} x^{d-1} + \cdots + \alpha_0$:

$$\text{minimize } \max_i |y_i - p(x_i)|$$

Solution: let $f = \max_i |y_i - p(x_i)|$. The problem can be formulated as a LP problem:

$$\text{minimize } f \text{ with respect to } \alpha_i$$

such that

$$-f \leq y_i - p(x_i) \leq f$$

...which is equivalent to

$$\text{minimize } f$$

such that

$$-p(x_i) - f \leq -y_i$$
$$p(x_i) - f \leq y_i$$

Example

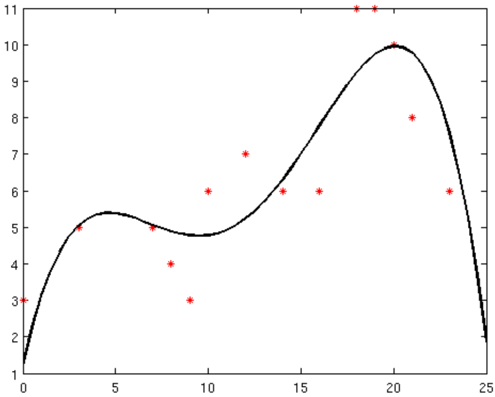
Approximate a set of 14 points with a 4-degree polyomial.

```
% given data: x, y
x = [0,3,7,8,9,10,12,14,16,18,19,20,21,23]';
y = [3,5,5,4,3,6,7,6,6,11,11,10,8,6]';

% ineq. constraints:
A1 = [-x.^4,-x.^3,-x.^2,-x,-ones(14,1),-ones(14,1)];
A2 = [x.^4,x.^3,x.^2,x,ones(14,1),-ones(14,1)];
A = [A1; A2];
b = [-y;y];

f = zeros(6,1); f(6)=1;  % objective function

[alpha, fval, exitflag] = linprog(f,A,b);
```


11
10
9
8
7
6
5
4
3
2
1
0
5
10
15
20
25

Outline

Quadratic programming (QP)

General form:

$$\text{minimize } \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x}$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$
$$lb \leq \mathbf{x} \leq ub$$

with $\mathbf{H} \in \mathbb{R}^{n \times s}$ symmetric. MATLAB:

```
X = quadprog(H,f,A,b,Aeq,beq,LB,UB,X0,OPTIONS)
```

QP - Example

Solve:
minimize $x_1^2 + x_1x_2 + 2x_2^2 + 2x_3^2 + 2x_2x_3 + 4x_1 + 6x_2 + 12x_3$
subject to

$$x_1 + x_2 + x_3 \geq 6$$
$$-x_1 - x_2 + 2x_3 \geq 2$$
$$x_1, x_2, x_3 \geq 0$$

```
H = [2,1,0;1,4,2;0,2,4];
f = [4,6,12];
A = [-1,-1,-1;1,1,-2]; b = [-6,-2];
lb = [0;0;0]; ub = [inf;inf;inf];
opts=optimoptions('quadprog', 'algorithm', ...
    'interior-point-convex');
[x,fval,exitflag,output] = ...
    quadprog(H, f, A, b, [], [], lb, ub, [],opts);
```

Outline

Constrained nonlinear optimization - `fmincon`

Problem:

$$\text{minimize } f(\mathbf{x})$$

subject to

$$
\begin{aligned}
c(\mathbf{x}) &\leq 0 \\
c_{eq}(\mathbf{x}) &= 0 \\
\mathbf{A}\mathbf{x} &\leq \mathbf{b} \\
\mathbf{A}_{eq}\mathbf{x} &= \mathbf{b}_{eq} \\
lb \leq \mathbf{x} &\leq ub
\end{aligned}
$$

MATLAB:

```
[x,fval,exitflag,output] = fmincon(fun, x0, A, b, ...
    Aeq, beq, lb, ub, nonlcon, options)
```

Algorithms for `fmincon`

- trust-region reflective: requires the gradient and allows only bounds or linear equality constraints, *but not both*. Works on large sparse and small dense problems efficiently.
- active-set can take large steps to converge fast. It is effective on some small problems with nonsmooth constraints.
- sqp satisfies bounds at each iteration. Not for large-scale problems.
- interior-point: for large+sparse or small+dense problems. Designed for large problems, can recover from `NaN` or `Inf` results. Satisfies bounds at each iteration.

Use the documentation for `fmincon` and `optimoptions` functions for details. You can use `optimtool` for a graphical user interface to optimization toolbox!

Exercise

Design the optimal beer can! It must be:

- cylindrical
- ecological: uses the minimum amount of materials (i.e. minimum total surface)
- of exact volume $V = 333 cm^3$
- not higher than twice its diameter

Tasks:

1. identify the variables
2. write the mathematical formulation of the problem
3. write the formula of the gradient of the objective function and the Jacobian of the nonlinear constraint function
4. implement in Matlab