Optimization

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

31 I 2017

Department of Computer and Information Science
Linköping University

Plan for today

- Introduction
- Mathematical definition of problem
- 1D optimization
- kD optimization
- `R` code examples

Optimization

Nearly everything is optimization **!**

- Chemistry
- Physics
- Economics, **Industry**
- Engineering

BUT EVEN

- Your mobile price plan
- Course scheduling
- Your lunch choice

STATISTICS

- Fit parameters to data
- Propose optimal decision

ANY BIOLOGICAL ORGANISM

YOU

Industry

How to produce a cylindrical (**WHY?**) $0.5L$ beer can so it requires minimum material?

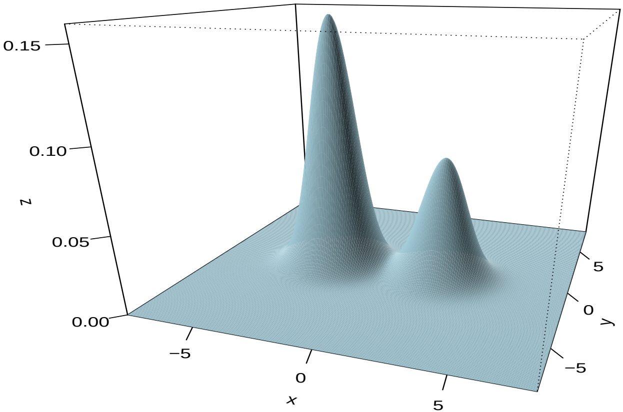
Given a certain product minimize e.g. material usage, production effort while still meeting consumer requirements.

Economics/Logistics

- Travelling Salesman Problem
- Windmills
- Flight schedule (especially “cheap” airlines)

Statistics

Maximize likelihood, model fitting



Maximal likelihood

An i.i.d. sample $(X_1, \ldots, X_n)$ is drawn from a probability distribution $P(X|\Theta)$, where Θ is an unknown parameter set.

The joint probability of all the observations is

$$P(X_1, \ldots, X_n|\Theta) = \prod_{i=1}^{n} P(X_i|\Theta).$$

Find Θ that maximizes $P(X_1, \ldots, X_n|\Theta)$.

Mathematical formulation

The goal is to minimize (maximize)

Objective function: $f(\theta)$
(reproduction, chances of survival, quality of life, cost, profit, likelihood, fit to data)

depending on

Parameters or **Unknowns** θ
(reproduction strategy, resource utilization, consumer choices, height & diameter, production, raw material choice, service times, route, flight routes/times ,parameters)

Mathematical formulation

$$\boxed{\min_{\theta \in \Theta} f(\theta) \;\; \text{subject to} \;\; \begin{array}{ll} c_i(\theta) = 0, & i \in E \\ c_i(\theta) \geq 0, & i \in I \end{array}}$$

QUESTION: What should we do if we are interested in maximization instead of minimization?

QUESTION: What should we do if the constraints are $c_i(x) \leq 0, \; i \in I$?

Constraints examples

- Available environment
- Volume: 0.5l of can
- Production: Factories (F_1, F_2), retail outlets (R_1, R_2, R_3), cost of shipping $i \to j$: c_{ij}, production a_i per week, requirement b_j per week **to optimize:** x_{ij} amount shipped $i \to j$ per week

$$
\begin{array}{ll}
\min\limits_{x \in \mathrm{R}^3} \sum_{ij} c_{ij} x_{ij} & \text{minimize shipping costs} \\
\sum\limits_{j=1}^{3} x_{ij} \leq a_i, i = 1, 2 & \text{production capacity} \\
\sum\limits_{i=1}^{3} x_{ij} \geq b_j, j = 1, 2, 3 & \text{demand} \\
\forall_{i,j} x_{ij} \geq 0 &
\end{array}
$$

Question: What would happen if we drop demand constraint?

- ML: often no constraints

Exercise

- Split into pairs/triplets/quadruples
- Think of some human anatomy part/organ:
  - What is its function?
  - What could it have been optimized for over the course of time?
  - Is it still under selection?
  - What constraints was and is it under?
- Think of a situation where optimization is needed in your own student/professional/personal/financial situation.
- State the problem in terms of
  - Objective function
  - Parameters
  - Constraints
  - Does it have a trivial solution?
- **10** minutes

Optimization approaches

- Constrained optimization
  - Lagrange multipliers, linear programming
  - E.g. LASSO
  - **Not this lecture!**

- Unconstrained optimization
  - Steepest descent
  - Newton method
  - Quasi–Newton–Methods
  - Conjugate gradients

Why are there different methods?

1D Optimization

- Function of a single parameter, find minimum
- *What algorithm would you suggest?*

- **Golden–section search**
  local minimum on $[A, B]$ interval (constraint)

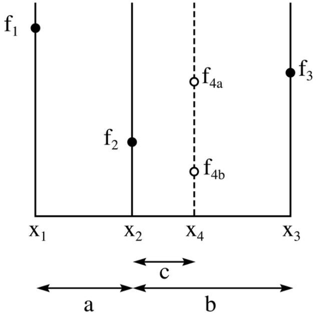
- Works by narrowing down the search interval with a constant reduction factor

$$1 - \alpha = \frac{\sqrt{5} - 1}{2} \approx 0.62$$

Question: Does α remind you of something?

Golden section (minimization)

$x_1 = A$, $x_3 = B$,
while $x_1 - x_3 > \epsilon$ **do**
    $a = \alpha(x_3 - x_1)$
    $x_2 = x_1 + a$, $x_4 = x_3 - a$
    if $f(x_4) > f(x_2)$ **then**
        $x_1 = x_1$, $x_3 = x_4$
    else
        $x_1 = x_2$, $x_3 = x_3$
    end if{We know the value at **3** points!}
end while



Wikipedia, Golden–section search

f has to be **UNIMODAL**

1D Optimization: Example

```r
f <- function (x, a) (x - a)^2
xmin <- optimize(f, c(0, 1), tol = 0.0001, a =
    1/3)
## See where the function is evaluated:
optimize(function(x) x^2*(print(x)-1), lower =
    0, upper = 10)

## "wrong" solution with unlucky interval and
   piecewise constant f():
f  <- function(x) ifelse(x > -1, ifelse(x < 4,
    exp(-1/abs(x - 1)), 10), 10)
fp <- function(x) { print(x); f(x) }
xmin1<-optimize(fp, c(-4, 20))   # doesn't see
    the minimum
xmin2<-optimize(fp, c(-7, 20))   # ok
```

Multi–dimensional optimization

Find

$$\boxed{\min_{\vec{x}\in\mathbb{R}^n} f(\vec{x})}$$

Using (known, or numerically evaluated)

$$\textbf{Gradient}\ \ \bigtriangledown f(\vec{x}) = \left(\frac{\partial f(\vec{x})}{\partial x_1}, \ldots, \frac{\partial f(\vec{x})}{\partial x_n}\right)^T$$

$$\textbf{Hessian}\ \ \bigtriangledown^2 f(\vec{x}) = \left[\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}\right]_{i,j=1}^n$$

General strategy

1. Provide a **(good)** starting point $\vec{x}_0$,
   $\vec{x} = \vec{x}_0$
2. Choose a direction $\vec{p}$ ($\|p\| = 1$) and step size a
3. Move to $\vec{x} := \vec{x} + \alpha\vec{p}$
4. Repeat step 2 until convergence

How to choose the direction?

Taylor's theorem

$$f(\vec{x} + a\vec{p}) = f(\vec{x}) + \boxed{\alpha\vec{p}^T \cdot \bigtriangledown f(\vec{x})} + o(\alpha^2)$$

$\vec{p}$ s.t. $\vec{p}^T \cdot \bigtriangledown f(\vec{x}) < 0$ is a *descent* direction.

Steepest descent is

$$\vec{p} = -(\bigtriangledown f(\vec{x}))/\| \bigtriangledown f(\vec{x})\|$$

How to choose the step size?

- **Expensive way**: find the global minimum in direction $\vec{p}$
- **Trade–off way**: find a decrease which is *sufficient*

BACKTRACKING

1: Choose (large) $\alpha_0 > 0$, $\rho \in (0,1)$, $c \in (0,1)$,
2: $\alpha = \alpha_0$
3: **repeat**
4: $\quad \alpha = \rho\alpha$
5: **until** $f(\vec{x} + \alpha\vec{p}) \leq f(\vec{x}) + c\alpha\vec{p}^T \bigtriangledown f(\vec{x})$

Newton's method

- Newton–Raphson method
- Hessian ignored in steepest descent
- If f is quadratic

$$f(\vec{p}) = \frac{1}{2}\vec{p}^T \mathbf{A}\vec{p} + \vec{b}^T\vec{p} + c,$$

then minimum

$$\vec{p}^* = \mathbf{A}^{-1}\vec{b}.$$

- Taylor expansion of f

$$f(\vec{x} + a\vec{p}) = f(\vec{x}) + \alpha\vec{p}^T \cdot \bigtriangledown f(\vec{x}) + \frac{\alpha^2}{2}\vec{p}^T \bigtriangledown^2 f(\vec{x})\vec{p} + o(\alpha^3)$$

- $x := x + \alpha\vec{p}$ where

$$\boxed{\vec{p} = -\left(\bigtriangledown^2 f(\vec{x})\right)^{-1} \bigtriangledown f(\vec{x})}$$

Newton's method

- $\left(\bigtriangledown^2 f(\vec{x})\right)^{-1}$ is expensive to compute, there are quicker approaches, e.g. Cholesky decomposition
- Hessian should be **positive definite** for $\vec{p}$ to be a descent direction (if not see book)
- Memory expensive — need to store $O(n^2)$ elements

BUT

- Method converges quickly esp. near optimum

Quasi–Newton methods

- k iteration number
- Compute an approximation to the Hessian, $\mathbf{B}$, that will allow for efficient choice of $\vec{p}$.
- **SECANT CONDITION:** (quasi–Newton condition)

$$\mathbf{B}_{k+1}\left(\vec{x}_{k+1} - \vec{x}_k\right) = \bigtriangledown f(\vec{x}_{k+1}) - \bigtriangledown f(\vec{x}_k)$$

BFGS Algorithm

Choose $\mathbf{B}_0 > 0$, $\vec{x}_0$, $k = 0$
repeat
    $\vec{p}_k$ is solution of $\mathbf{B}_k\vec{p}_k = \bigtriangledown f(\vec{x}_k)$
    find suitable α_k
    $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$
    calculate $\mathbf{B}_{k+1}$ {next slide}
    $k = k + 1$
until convergence of $\vec{x}_k$ at minimum

How to compute $\mathbf{B}_{k+1}$?

- We want $\mathbf{B}_{k+1}$ and $\mathbf{B}_k$ to be close to each other

- $$\min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_k\|$$
  $$s.t.\ \mathbf{B} = \mathbf{B}^T,\ \text{secant condition}$$

- $\vec{y}_k = \triangledown f(\vec{x}_{k+1}) - \triangledown f(\vec{x}_k)$, $\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$
- $$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \vec{y}_k \vec{y}_k^T \mathbf{B}_k}{\vec{y}_k^T \mathbf{B}_k \vec{y}_k} + \frac{\vec{s}_k \vec{s}_k^T}{\vec{y}_k \vec{s}_k^T}$$

- Closed form Sherman–Morrison formula for $\mathbf{B}_{k+1}^{-1}$
- We have to store $\mathbf{B}_k^{-1}$

BFGS

- BGFS: Broyden–Fletcher–Goldfarb–Shanno
- More iterations than Newton's method (uses approximation)
- Each iteration quicker, no numeric inversion
- Good for large scale problems
- Choice of $\mathbf{B}_0$?

Conjugate Gradient method—quadratic case

Minimize

$$f(\vec{x}) = \frac{1}{2}\vec{x}^T \mathbf{A}\vec{x} - \vec{b}^T \vec{x}$$

for $\mathbf{A}$ symmetric positive definite.
Gradient:

$$\bigtriangledown f(\vec{x}) = \mathbf{A}\vec{x} - \vec{b} = r(\vec{x})$$

Two vectors $\vec{p}$ and $\vec{q}$ are **conjugate** with respect to $\mathbf{A}$ if

$$\vec{p}^T \mathbf{A}\vec{q} = 0.$$

IDEA: $\vec{p}$ and $\vec{q}$ are orthogonal w.r.t. to an inner product associated with $\mathbf{A}$. Use this to find a basis that will allow for easy finding of $\vec{x}$.

Conjugate Gradient method

- $\vec{p}_0 = \vec{r}_0$
- $\vec{p}_{k+1} = -\vec{r}_k + \beta_{k+1}\vec{p}_k$

- Conjugate condition has to be satisfied so

$$\beta_{k+1} = \frac{\vec{r}_k^T \mathbf{A} \vec{p}_{k-1}}{\vec{p}_k^T \mathbf{A} \vec{p}_k}$$

Exercise: check this

- Convergence in $\dim(\mathbf{A})$ steps
  (or unless cutoff for $\vec{r}_k$)

Nonlinear CG method

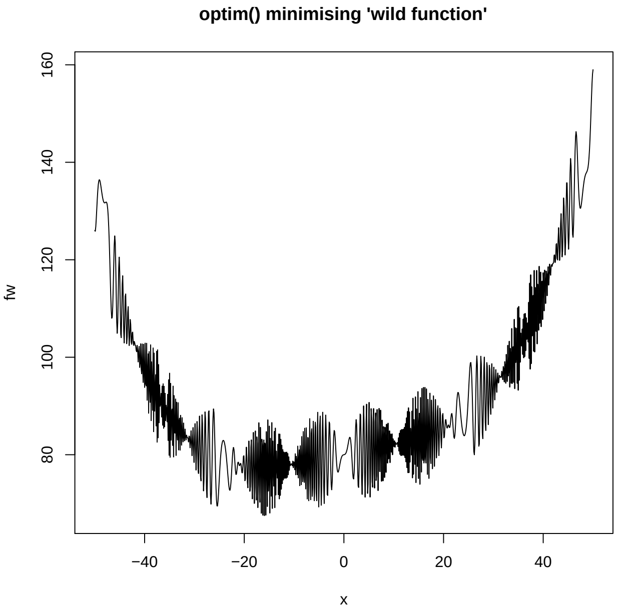
- If $f(\cdot)$ general, use $\bigtriangledown f(\cdot)$ instead of $r(\cdot)$

Choose $\vec{x}_0$, $\vec{p}_0 = -\bigtriangledown f(\vec{x}_0)$, $k = 0$
while $\bigtriangledown f(x_k) \neq \vec{0}$ **do**
  find suitable α_k
  $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$ {and now update step}
  $\beta_{k+1} = \left(\bigtriangledown^T f(\vec{x}_{k+1}) \bigtriangledown f(\vec{x}_{k+1})\right) / \left(\bigtriangledown^T f(\vec{x}_k) \bigtriangledown f(\vec{x}_k)\right)$
  {Fletcher–Reeves update, other possible}
  $\vec{p}_{k+1} = -\bigtriangledown f(\vec{x}_{k+1}) + \beta_{k+1}\vec{p}_k$
  $k = k + 1$
end while

Nonlinear CG method

- Local minimum convergence
- But this is true of all methods that cannot "jump out" of descent path
- Faster than steepest descent
- Slower than Newton and Quasi–Newton but significantly less memory

kD Optimization: Example



kD Optimization: Example

```
## "wild" function , global minimum at about
    -15.81515
fw <- function (x)
           10*sin(0.3*x)*sin(1.3*x^2) + 0.00001*
              x^4 + 0.2*x+80
plot(fw, -50, 50, n = 1000, main = "optim() 
    minimising 'wild function'")
# method = c("Nelder-Mead", "BFGS", "CG", "L-
    BFGS-B", "SANN","Brent")
res <- optim(50, fw, method = "SANN",lower = -
    Inf, upper = Inf, control = list(maxit =
    20000, temp = 20, parscale = 20), hessian =
     FALSE)
# res$par= -15.8144, res$value=67.47249, res$$
    counts["function"]=20000,res$$counts["
    gradient"]=NA, res$convergence=0 (did!),
    res$message=NULL
```

Summary

- Optimization is everywhere
- Numerical methods for finding minimum
- 1D: Golden section (unimodal), `optimize()`
- kD: choose step size and direction (gradient), `optim()`