

Optimization

732A90

Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

24 I 2019 (P42)

Department of Computer and Information Science
Linköping University

Plan for today

- Introduction
- Mathematical definition of problem
- 1D optimization
- k D 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?

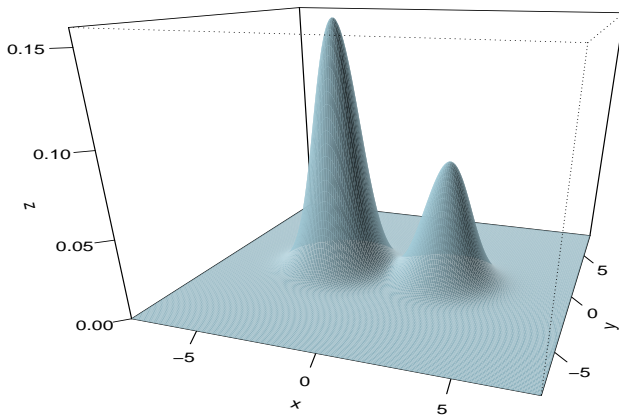
Given a certain product minimize e.g. material usage, production effort while still meeting consumer requirements.

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, \dots, 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

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

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

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


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



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 $i \rightarrow j$: c_{ij} , production a_i per week, requirement b_j per week **to optimize:** x_{ij} amount shipped $i \rightarrow j$ per week

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \sum_{ij} c_{ij} x_{ij} & \quad \text{minimize shipping costs} \\ \sum_{j=1}^3 x_{ij} \leq a_i, i = 1, 2 & \quad \text{production capacity} \\ \sum_{i=1}^3 x_{ij} \geq b_j, j = 1, 2, 3 & \quad \text{demand} \\ \forall_{i,j} x_{ij} \geq 0 & \end{aligned}$$

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

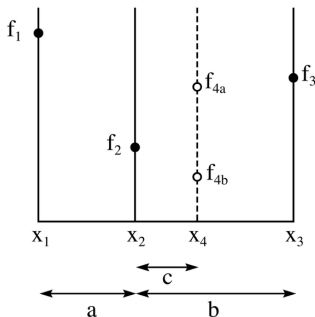
- Function of a single parameter, find minimum
- *What algorithm would you suggest?*
- **Golden-section search** 黄金分割法
local minimum on $[A, B]$ interval (constraint)
- Works by narrowing down the search interval with a constant reduction factor

$$1 - \alpha = \frac{\sqrt{5} - 1}{2} \approx 0.62 \quad \text{golden ratio}$$

Question: Does α remind you of something?

Golden section (minimization)

```
1:  $x_1 = A, x_3 = B,$   
2: while  $x_1 - x_3 > \epsilon$  do  
3:    $a = \alpha(x_3 - x_1)$   
4:    $x_2 = x_1 + a, x_4 = x_3 - a$   
5:   if  $f(x_4) > f(x_2)$  then  
6:      $x_1 = x_1, x_3 = x_4$   
7:   else  
8:      $x_1 = x_2, x_3 = x_3$   
9:   end if {We know the value at 3 points!}  
10: end while
```



Wikipedia, Golden-section search

f has to be **UNIMODAL** 只有极值的函数模型

1D Optimization: Example

732A90_ComputationalStatisticsVT2019_Lecture02codeSlide16.R

Multi-dimensional optimization

Find

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

Using (known, or numerically evaluated)

Gradient $\nabla f(\vec{x}) = \left(\frac{\partial f(\vec{x})}{\partial x_1}, \dots, \frac{\partial f(\vec{x})}{\partial x_n} \right)^T$
梯度下降

Hessian $\nabla^2 f(\vec{x}) = \left[\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} \right]_{i,j=1}^n$
海森矩阵 牛顿优化

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

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

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

Steepest descent is

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

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 缩小step

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

Newton's method

- Newton–Raphson method 牛頓-拉弗森方法
- 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

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

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

$$\vec{p} = -(\nabla^2 f(\vec{x}))^{-1} \nabla f(\vec{x})$$

Newton's method

- $(\nabla^2 f(\vec{x}))^{-1}$ is expensive to compute, there are quicker approaches, e.g. Cholesky decomposition 临界点 x_0 是一个局部的极小值
any $x \geq 0$, $x^T x > 0$
- 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)

$$\text{去掉极限,} \\ \mathbf{B}_{k+1} (\vec{x}_{k+1} - \vec{x}_k) = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$$

BFGS Algorithm

- 1: Choose $\mathbf{B}_0 > 0$, \vec{x}_0 , $k = 0$
- 2: **repeat**
- 3: \vec{p}_k is solution of $\mathbf{B}_k \vec{p}_k = -\nabla f(\vec{x}_k)$
- 4: find suitable α_k
- 5: $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$
- 6: calculate \mathbf{B}_{k+1} {next slide}
- 7: $k = k + 1$
- 8: **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



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

- $\vec{y}_k = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$, $\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$



$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \vec{y}_k \vec{y}_k^T \mathbf{B}_k}{\vec{y}_k^T \mathbf{B}_k \vec{y}_k} + \frac{\vec{s}_k \vec{s}_k^T}{\vec{y}_k \vec{s}_k^T}$$

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

- BFGS: 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 ? 初始值的选择对结果影响大

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:

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

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)

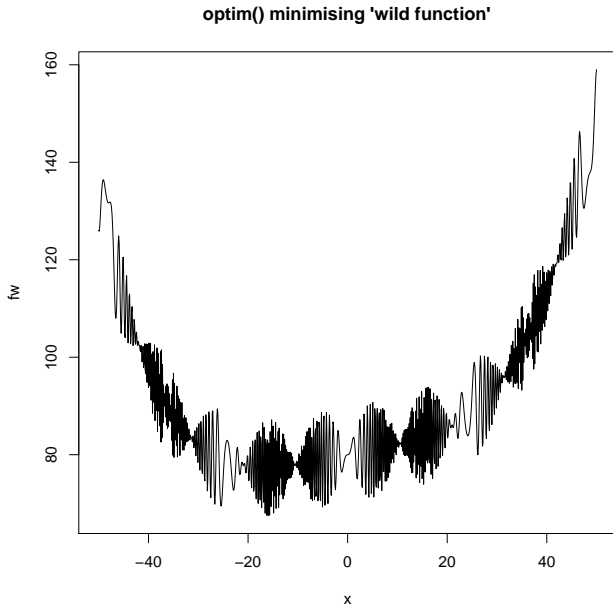
Nonlinear CG method

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

- 1: Choose $\vec{x}_0, \vec{p}_0 = -\nabla f(\vec{x}_0), k = 0$
- 2: **while** $\nabla f(\vec{x}_k) \neq \vec{0}$ **do**
- 3: find suitable α_k
- 4: $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$ {and now update step}
- 5: $\beta_{k+1} = (\nabla^T f(\vec{x}_{k+1}) \nabla f(\vec{x}_{k+1})) / (\nabla^T f(\vec{x}_k) \nabla f(\vec{x}_k))$
 {Fletcher-Reeves update, other possible}
- 6: $\vec{p}_{k+1} = -\nabla f(\vec{x}_{k+1}) + \beta_{k+1} \vec{p}_k$
- 7: $k = k + 1$
- 8: **end while**

- 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

*k*D Optimization: Example



k D Optimization: Example

732A90_ComputationalStatisticsVT2019_Lecture02codeSlide31.R

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