# Optimization

732A90
Computational Statistics

Maryna Prus
(maryna.prus@liu.se)

Slides originally by Krzysztof Bartoszek

November 9, 2021

Department of Computer and Information Science
Linköping University

# Plan for today

- Introduction
- Mathematical definition of problem
- 1D optimization, function `optimize()`
- $k$D optimization, function `optim()`

# 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

# Industry

How to produce a cylindrical $0.5L$ 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

# Optimization

- Mathematical optimization
  - Linear optimization
  - Non-linear optimization
  - Combinatorial optimization
  - Integer optimization
  - ...

    **Not this lecture**

- Computational approach
  - Algorithms, heuristics

    Convergence ?

    **This lecture**

# 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** $\theta$
(reproduction strategy, resource utilization, consumer choices, height & diameter, production, raw material choice, service times, route, flight routes/times ,parameters)

# Mathematical formulation

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

**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

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

# 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
  - **This lecture**

# 1D Optimization: analytical approach

- To minimize / maximize

$$f(x), \quad x \in \mathbb{R}$$

  $x$ - one dimensional variable
- Compute first derivative: $\frac{\partial f}{\partial x}$
- Find $x^*$ for which first derivative is 0
- Compute second derivative: $\frac{\partial^2 f}{\partial x^2}$
- Compute value of second derivative at $x = x^*$
  - Second derivative positive
    $\Rightarrow \quad x^*$ local minimum
  - Second derivative negative
    $\Rightarrow \quad x^*$ local maximum
  - Second derivative 0
    $\Rightarrow \quad x^*$ saddle point

# 1D Optimization: computational approach

- To minimize / maximize

$$f(x), \quad x \in \mathbb{R}$$

$x$ - one dimensional variable

- Algorithms

*Golden-Section Search* (see next slide)

  - local minimum / maximum on interval $[A, B]$
  - Works by narrowing down the search interval with a constant reduction factor

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

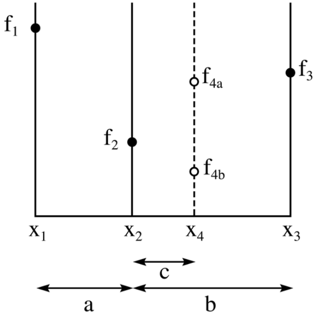
- In **R**:

function `optimize()`

- Brent's method (modified Golden-Section Search)

# Golden-Section Search (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**
**end while**




Wikipedia, Golden-Section Search

- $f$ has one local minimum $\rightarrow$ minimum will be found
- $f$ has several local minima $\rightarrow$ one of them will be found

# 1D Optimization: Example

```
f <- function (x, a) (x - a)^2
xmin <- optimize(f, c(0, 1), tol = 0.0001, a =
    1/3)
xmin

## "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)
plot(f, -20, 20)
xmin1<-optimize(f, c(-4, 20))    # doesn't see
   the minimum
xmin1
xmin2<-optimize(f, c(-7, 20))    # ok
xmin2
```

# $k$D Optimization: analytical approach

- To minimize / maximize

$$f(\vec{x}), \quad \vec{x} = (x_1, \ldots, x_n)^\top$$

  $\vec{x}$ - $n$-dimensional vector

- Compute gradient:

$$\nabla f(\vec{x}) = \left( \frac{\partial f(\vec{x})}{\partial x_1}, \ldots, \frac{\partial f(\vec{x})}{\partial x_n} \right)^\top$$

  - vector of all partial derivatives

- Find $\vec{x}^*$ for which gradient is $\mathbf{0}$
  - all partial derivatives are 0

# $k$D Optimization: analytical approach

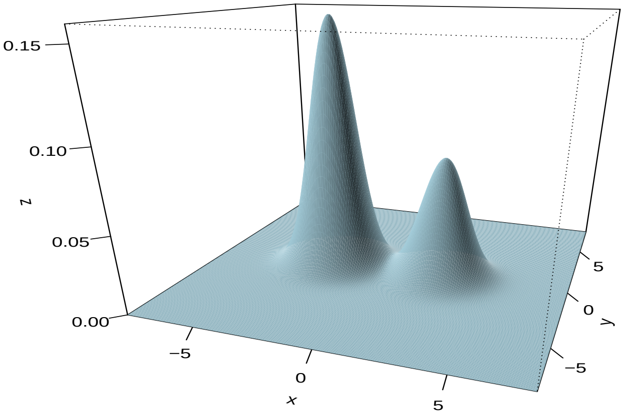
- Compute Hessian matrix:

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

  - matrix of all second partial derivatives
- Compute values of Hessian matrix at $x = x^*$
  - Hessian matrix positive definite
    $\Rightarrow \quad \vec{x}^*$ local minimum
  - Hessian matrix negative definite
    $\Rightarrow \quad \vec{x}^*$ local maximum
  - Otherwise
    $\Rightarrow \quad \vec{x}^*$ saddle point

# Statistics

**Maximize likelihood**



# Maximal (maximum) likelihood

$X_1, \ldots, X_n$ i.i.d. - sample is drawn from probability distribution $P(X|\theta)$, where $\theta$ is unknown parameter set

$P(X|\theta)$ - probability function (for discrete random variables)
- density function (for continuous random variables)

The joint probability / density function for all observations:

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

Find $\theta$ that maximizes $P(X_1, \ldots, X_n|\theta)$

**Example**: $X_1, \ldots, X_n \sim N(\mu, \sigma^2) \quad \Rightarrow \quad \theta = (\mu, \sigma^2)$

# $k$D Optimization: computational approach

**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 $\alpha$
3. Move to $\vec{x} := \vec{x} + \alpha\vec{p}$
4. Repeat step 2 until convergence

# Choice of direction

**Taylor's theorem**

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

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

***Steepest*** **descent** is

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

*Hessian matrix ignored in steepest descent*

# Choice of 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** $f(\vec{x} + \alpha\vec{p}) \leq f(\vec{x}) + c\alpha\vec{p}^\top \bigtriangledown f(\vec{x})$

# Newton-Raphson Method

- If $f$ is quadratic

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

then minimum

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

- Taylor expansion of $f$

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

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

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

# Newton-Raphson Method

"–"

- $(\bigtriangledown^2 f(\vec{x}))^{-1}$ is expensive to compute
  $\rightarrow$ 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

"+"

- 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)

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

**BFGS Algorithm**

1: Choose $\mathbf{B}_0 > 0$, $\vec{x}_0$, $k = 0$
2: **repeat**
3: $\quad$ $\vec{p}_k$ is solution of $\mathbf{B}_k \vec{p}_k = \triangledown f(\vec{x}_k)$, i.e.
$\quad$ $\vec{p}_k = \mathbf{B}_k^{-1} \triangledown f(\vec{x}_k)$
4: $\quad$ find suitable $\alpha_k$
5: $\quad$ $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$
6: $\quad$ calculate $\mathbf{B}_{k+1}$ {next slide}
7: $\quad$ $k = k + 1$
8: **until** convergence of $\vec{x}_k$ at minimum

# Computing $\mathbf{B}_{k+1}$

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

$$\min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_k\|$$
$$s.t.\ \mathbf{B} = \mathbf{B}^\top$$

- For $\vec{y}_k = \triangledown f(\vec{x}_{k+1}) - \triangledown f(\vec{x}_k)$, $\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$
  We receive

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

- 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
- Choice of $\mathbf{B}_0$?

# Conjugate Gradient Method—quadratic case

Minimize

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

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

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

Method is based on this property

# Conjugate Gradient Method

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

  where

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

- 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 $\alpha_k$
    $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$ {and now update step}

---

$\beta_{k+1} = \left(\bigtriangledown f(\vec{x}_{k+1})^\top \bigtriangledown f(\vec{x}_{k+1})\right) / \left(\bigtriangledown f(\vec{x}_k)^\top \bigtriangledown f(\vec{x}_k)\right)$
{Fletcher–Reeves update, other possible}

---

$D_{k+1} := \bigtriangledown f(\vec{x}_{k+1}) - \bigtriangledown f(\vec{x}_k$
$\beta_{k+1} = \left(\bigtriangledown f(\vec{x}_{k+1})^\top D_{k+1}\right) / \left(\bigtriangledown f(\vec{x}_k)^\top \bigtriangledown f(\vec{x}_k)\right)$
{Polak–Ribiére 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
- Cannot "jump out" of descent path
- Faster than steepest descent
- Slower than Newton and Quasi–Newton but significantly less memory

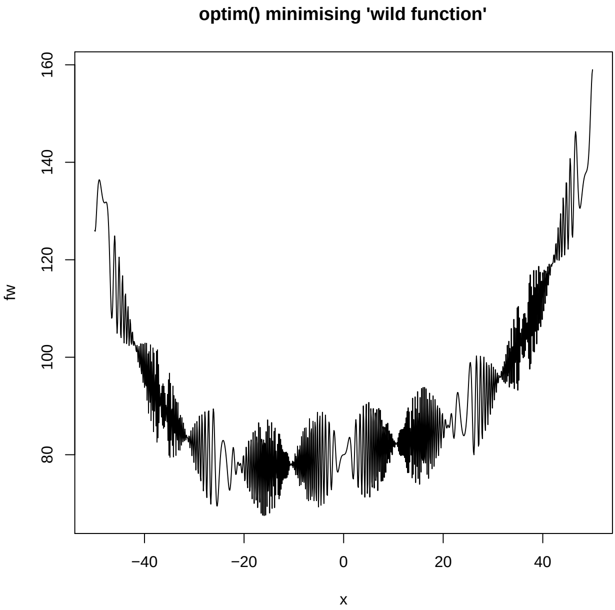
## $k$D Optimization in **R**

**Function** `optim()`

Methods:

- Nelder-Mead
- **BFGS**
- **Conjugate Gradient**
- L-BFGS-B
- SANN
- Brent

# $k$D Optimization: Example



# $k$D 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'")
res <- optim(50, fw, method = "CG", control =
   list(maxit = 20000, parscale = 20))
res
```

Thank you for attention!