1. Introduction

- mathematical optimization
- least squares and linear programming
- convex optimization
- example
- course information

Mathematical optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

- $x = (x_1, \ldots, x_n)$: optimization variables (decision variables)
- f_0: objective function (cost function)
- $f_1, \ldots, f_m$, $h_1, \ldots, h_p$: inequality and equality constraint functions

Examples

Optimal design and control

- variables represent design parameters, decisions, control actions
- objective function measures performance, cost, deviation from desired outcome
- constraints represent design specifications, restrict allowable choices

Model fitting and approximation

- variables are model parameters
- objective includes approximation or prediction error, regularization terms
- constraints represent prior knowledge, restrictions on possible values

Solving optimization problems

General optimization problem

- very difficult to solve with guarantees of global optimality
- good suboptimal solutions are often sufficient in applications

Exceptions: important classes of problems can be solved globally and efficiently

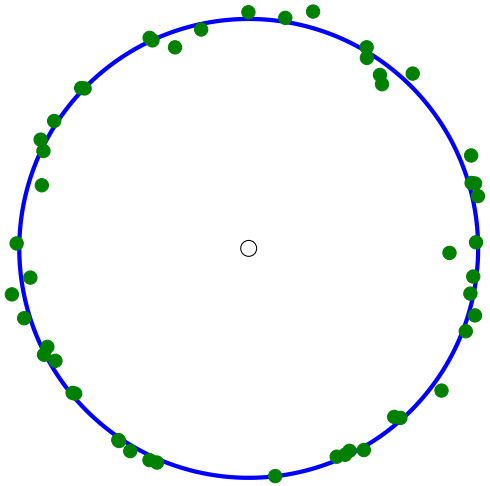
- least squares
- linear programming
- convex optimization

Least squares

$$\text{minimize} \quad \|Ax - b\|_2^2 = \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j - b_i)^2$$

- A is an $m \times n$ matrix, b is an m-vector
- $\|y\|_2 = \sqrt{y_1^2 + \cdots + y_m^2}$ is the Euclidean norm of m-vector y
- optimal solutions satisfy the *normal equations* $A^T A x = A^T b$
- if A has full column rank, there is a unique solution $x = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- easy to recognize in applications
- flexibility is increased by adding weights, quadratic regularization terms

Example: fit sphere to set of points



$$\text{minimize} \quad \sum_{i=1}^{m}(\|y_i - u\|_2^2 - R^2)^2 = \sum_{i=1}^{m}(\|y_i\|_2^2 - 2y_i^T u + \|u\|_2^2 - R^2)^2$$

- $y_1, \ldots, y_m$ are m given points in $\mathbf{R}^p$
- optimization variables are center $u \in \mathbf{R}^p$ and radius $R \in \mathbf{R}$ of the fitted sphere
- not a least squares problem, due to the nonlinear terms R^2, $\|u\|_2^2$

Least squares formulation

$$\text{minimize} \quad \sum_{i=1}^{m} (\|y_i\|_2^2 - 2y_i^T u + w)^2$$

- use u and $w := \|u\|_2^2 - R^2$ as variables
- a least squares problem: minimize $\|Ax - b\|_2^2$ where

$$A = \begin{bmatrix} 1 & -2y_1^T \\ 1 & -2y_2^2 \\ \vdots & \vdots \\ 1 & -2y_m^T \end{bmatrix}, \qquad x = \begin{bmatrix} w \\ u \end{bmatrix}, \qquad b = \begin{bmatrix} -\|y_1\|_2^2 \\ -\|y_2\|_2^2 \\ \vdots \\ -\|y_m\|_2^2 \end{bmatrix}$$

- from least squares solution w, u, compute radius

$$R = \sqrt{\|u\|_2^2 - w}$$

Exactness of least squares formulation

we omitted the constraint in

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} (\|y_i\|_2^2 - 2y_i^T u + w)^2 \\ \text{subject to} & \|u\|_2^2 - w \geq 0 \end{array}$$

- constraint is needed to guarantee that we can compute $R = \sqrt{\|u\|_2^2 - w}$
- constraint can be omitted because least squares solution satisfies $\|u\|_2^2 - w \geq 0$
- this follows from the normal equations $A^T(Ax - b) = 0$: first equation is

$$\begin{aligned} 0 &= \mathbf{1}^T(Ax - b) && (\mathbf{1} \text{ is } m\text{-vector of ones}) \\ &= \sum_{i=1}^{m} (w - 2y_i^T u + \|y_i\|_2^2) \\ &= m(w - \|u\|_2^2) + \sum_{i=1}^{m} \|y_i - u\|_2^2 \end{aligned}$$

Linear programming

$$\begin{array}{ll} \text{minimize} & c^T x = c_1 x_1 + \cdots + c_n x_n \\ \text{subject to} & a_i^T x + b_i \leq 0, \quad i = 1, \ldots, m \end{array}$$

- no analytical formula for solution
- reliable and efficient algorithms and software
- not as easy to recognize as least squares problems
- a few standard techniques are used to convert problems into linear programs

  e.g., handling 1-norms or ∞-norms, piecewise-linear functions

Example: 1-norm approximation

$$\text{minimize} \quad \|Ax - b\|_1$$

- A is an $m \times n$ matrix, b is an m-vector
- $\|y\|_1 = |y_1| + |y_2| + \cdots + |y_m|$ is 1-norm of y
- linear programming formulation:

$$\begin{array}{ll} \text{minimize} & t_1 + t_2 + \cdots + t_m \\ \text{subject to} & -t_1 \leq a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n - b_1 \leq t_1 \\ & -t_2 \leq a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n - b_2 \leq t_2 \\ & \cdots \\ & -t_m \leq a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n - b_m \leq t_m \end{array}$$

a linear program with variables x and $u_1, \ldots, u_m$

Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- objective and inequality constraint functions are convex: for $0 \leq \theta \leq 1$,

$$f_i(\theta x + (1-\theta)y) \leq \theta f_i(x) + (1-\theta) f_i(y)$$

  (see lecture 3)

- equality constraints are linear

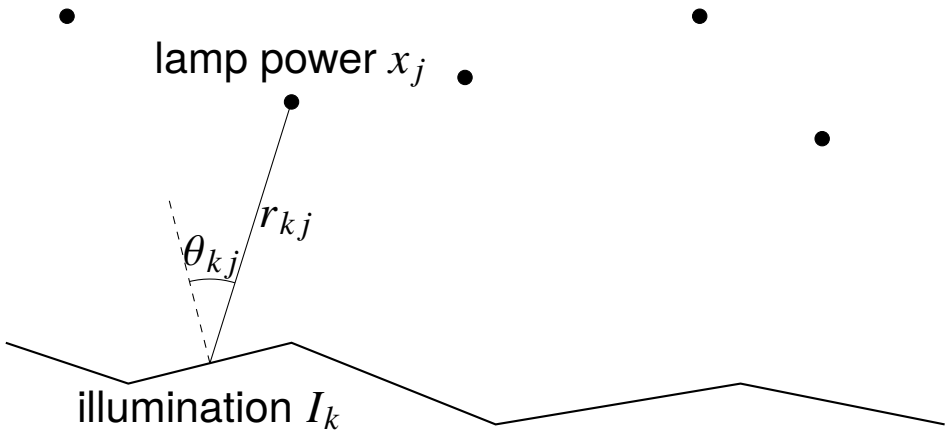
- includes least squares problems and linear programs as special cases

Using convex optimization

- no analytical formula for solution
- reliable and efficient algorithms
- may be difficult to recognize in applications
- many techniques available for transforming problems into convex form
- surprisingly many problems can be solved via convex optimization
- modeling languages (CVXPY, CVX, ...) greatly simplify interface with solvers

Example

- n lamps illuminate m (small, flat) patches



- intensity I_k at patch k depends linearly on lamp powers x_j:

$$I_k(x) = \sum_{j=1}^{n} a_{kj} x_j, \qquad \text{where } a_{kj} = r_{kj}^{-2} \max\{\cos\theta_{kj}, 0\}$$

Problem: achieve desired illumination I_{des} with bounded lamp powers

$$\begin{array}{ll} \text{minimize} & \max\limits_{k=1,\ldots,m} |\log I_k(x) - \log I_{\text{des}}| \\ \text{subject to} & 0 \le x_j \le p_{\max}, \quad j = 1, \ldots, n \end{array}$$

Approximate solutions

1. use uniform power: $x_j = p$ for $j = 1, \ldots, n$, vary p
2. use least squares: solve

$$\text{minimize} \quad \sum_{k=1}^{m} (I_k(x) - I_{\text{des}})^2$$

   and round x_j if $x_j > p_{\max}$ or $x_j < 0$
3. use weighted least squares:

$$\text{minimize} \quad \sum_{k=1}^{m} (I_k(x) - I_{\text{des}})^2 + \sum_{j=1}^{n} w_j (x_j - \tfrac{1}{2} p_{\max})^2$$

   iteratively adjust weights w_j until $0 \leq x_j \leq p_{\max}$
4. use linear programming:

$$\begin{array}{ll} \text{minimize} & \max\limits_{k=1,\ldots,m} |I_k(x) - I_{\text{des}}| \\ \text{subject to} & 0 \leq x_j \leq p_{\max}, \quad j = 1, \ldots, n \end{array}$$

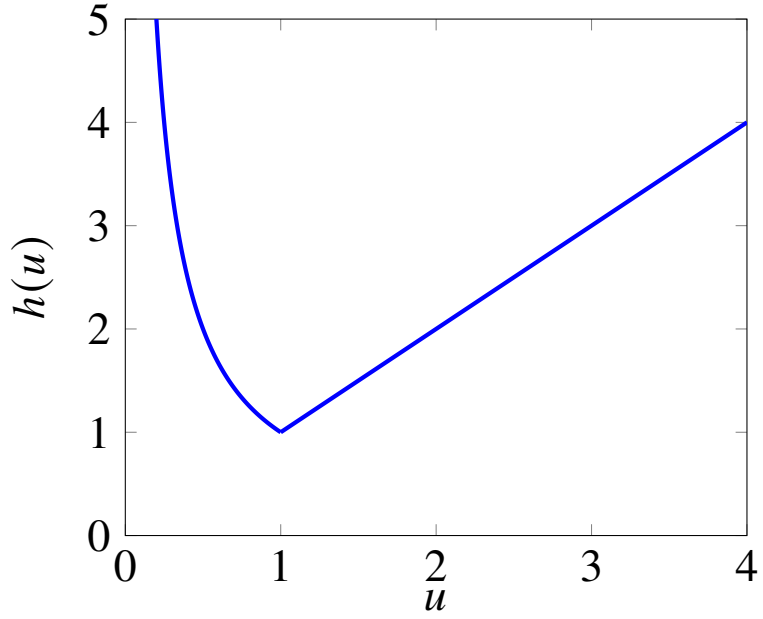
   which can be solved via linear programming

Convex formulation

problem is equivalent to

$$\begin{array}{ll} \text{minimize} & f_0(x) = \max\limits_{k=1,\ldots,m} h(I_k(x)/I_{\text{des}}) \\ \text{subject to} & 0 \leq x_j \leq p_{\max}, \quad j = 1, \ldots, n \end{array}$$

with $h(u) = \max\{u, 1/u\}$



f_0 is a convex function (see lecture 3)

exact solution obtained with effort $\approx$ modest factor $\times$ least squares effort

Nonconvex optimization

algorithms for general nonconvex optimization

Local optimization (nonlinear programming)

- find a solution that minimizes objective among feasible points near it
- fast algorithms, handle large problems
- often require initial guess
- provide no information about distance to (global) optimum

Global optimization

- find the global solution, with guarantee of optimality
- worst-case complexity grows exponentially with problem size

these algorithms are often based on iteratively solving convex subproblems

Course information

Course material

- textbook available online at web.stanford.edu/~boyd/cvxbook
- lecture slides, homework assignments on Bruin Learn course website bruinlearn.ucla.edu/courses/221577
- slides from previous years available on www.seas.ucla.edu/~vandenbe/ee236b

Course requirements (see syllabus on the on the course website)

- weekly homework
- computational problems will use the Python package CVXPY (cvxpy.org) or the MATLAB package CVX (cvxr.com)
- open-book final exam (Tuesday, March 17, 6:30pm–9:30pm)