I214 System Optimization

Brian Kurkoski

Japan Advanced Institute of Science and Technology

2023 February

Outline

7.1 Optimization Using Matlab and Python

7.1.1 How to Solve an LP Using Matlab

Consider the "running example" from Chapter 2:

$$\begin{aligned} \textbf{Maximize} \quad & x_1 + x_2 \\ \textbf{subject to} \quad & x_1 + 2x_2 \leq 3 \\ & x_1 \leq 2 \\ & x_2 \leq 1 \end{aligned} \tag{1}$$

Recall we can write in matrix form:

$$\begin{aligned} \textbf{Minimize} \quad & \mathbf{c}^t\mathbf{x} \\ \textbf{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 1 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Solve an LP with Matlab `linprog`

```
c = [−1 −1]';
A = [1 2 ; ...
     1 0 ; ...
     0 1 ];
b = [3 2 1]';
x = linprog(c,A,b)

Optimal solution found.

x =


    2.0000
    0.5000
```

Include Equality Constraints. Include Upper Bound, Lower Bound

```
>> help linprog

linprog Linear programming.

   X = linprog(f,A,b,Aeq,beq) solves the problem above while
   additionally satisfying the equality constraints Aeq*x = beq.
   (Set A=[] and B=[] if no inequalities exist.)
```

Include Equality Constraints. Include Upper Bound, Lower Bound

```
>> help linprog

linprog Linear programming.

   X = linprog(f,A,b,Aeq,beq) solves the problem above while
   additionally satisfying the equality constraints Aeq*x = beq.
   (Set A=[] and B=[] if no inequalities exist.)
```

```
   X = linprog(f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper
   bounds on the design variables, X, so that the solution is in
   the range LB <= X <= UB. Use empty matrices for LB and UB
   if no bounds exist. Set LB(i) = -Inf if X(i) is unbounded below;
   set UB(i) = Inf if X(i) is unbounded above.
```

Solutions Do Not Always Exist

```
>>> help linprog
  [X,FVAL,EXITFLAG] = linprog(f,A,b) returns an EXITFLAG that describes
  the exit condition. Possible values of EXITFLAG and the corresponding
  exit conditions are

    1  linprog converged to a solution X.
    0  Maximum number of iterations reached.
   -2  No feasible point found.
   -3  Problem is unbounded.
   -4  NaN value encountered during execution of algorithm.
   -5  Both primal and dual problems are infeasible.
   -7  Magnitude of search direction became too small; no further
       progress can be made.
```

7.1.2 Solve an LP Using Python `scipy`

Solve an LP Using Python scipy

```
>>> from scipy.optimize import linprog
>>> c = [-1, -1]
>>> A = [[1, 2], [1, 0], [0, 1]]
>>> b = [3, 2, 1]
>>> result = linprog(c, A_ub=A, b_ub=b)
>>> result.x
array([2. , 0.5])

>>> result.fun
-2.5
```

Install scipy using pip

```
# python3 textbookExample07.py
...
ModuleNotFoundError: No module named 'scipy'

# pip3 install scipy
Successfully installed scipy-1.10.0
```

7.1.3 Quadratic Program with Linear Constraints (Matlab)

$$\begin{aligned} \textbf{Minimize} \quad & (x_1 + x_2)^2 \\ \textbf{subject to} \quad & x_1 + x_2 \leq -2 \end{aligned}$$

Write in matrix form:

$$\begin{aligned} \textbf{Minimize} \quad & \tfrac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x} + \mathbf{g}^{\mathrm{t}}\mathbf{x} \\ \textbf{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \mathbf{f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Quadratic Program with Linear Constraints (Matlab)

```
>>> help quadprog
X = quadprog(H,f,A,b) attempts to solve quadratic programming problem:
        min 0.5*x'*H*x + f'*x   subject to:  A*x <= b
```

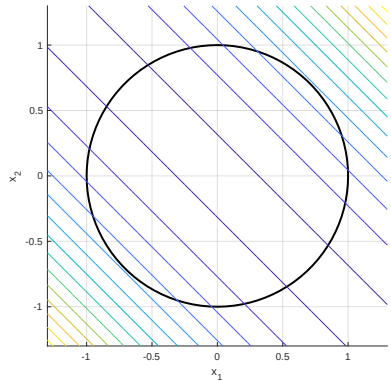
```
>> H = [2 2 ; 2 2];          f = [0 ; 0];
>> A = [1 1];                b = -2;
>> [x, fval] = quadprog(H,f,A,b)

x =
   -1.0000
   -1.0000

fval =
    4.0000
```

7.1.4 Nonlinear Optimization

Consider the following constrained optimization problem:

$$\textbf{Minimize} \quad (x_1 + x_2)^2$$

$$\textbf{subject to} \quad x_1^2 + x_2^2 = 1.$$



Nonlinear Optimization in Matlab

```
>> help fmincon

fmincon finds a constrained minimum of a function of several variables
  fmincon attempts to solve problems of the form:
   min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq (linear constraints)
                          C(X) <= 0, Ceq(X) = 0   (nonlinear con.)
                             LB <= X <= UB        (bounds)

  fmincon implements four different algorithms: interior point, SQP,
  active set, and trust region reflective. Choose one via the option
  Algorithm.
```

```
function f = objfun(x)
%objective function
f = (x(1)+x(2))^2;
```

```
function [c,ceq] = confun(x)
%inequality constraints
c = [];
%equality constraints
ceq = [x(1)^2 + x(2)^2 - 1];
```

```
>>> x0 = [0 0];  %starting value for iterative algorithm
>>> x = fmincon('objfun',x0,[],[],[],[],[],[],'confun')
x =

    0.7071    0.7071
```

Found Solution Depends on Starting Value

Recall the analytical solution is $\mathbf{x}^* = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$.

This numerical algorithm found only one solution.

If we give a different starting value, a different solution is possible

```
x0 = [−1 −1];  %new starting value
x = fmincon('objfun',x0,[],[],[],[],[],[],'confun')

x =


   −0.7071   −0.7071
```

Numerical algorithms often find only one solution, even if there are multiple optimum values.

7.2 Homework 8

Let $\mathcal{S}$ be a finite set of symbols, and L_1, L_2 be two strings of these symbols. We can find a sequence of symbols which commonly appear in both L_1 and L_2, and we call it a "common subexpression" of L_1 and L_2.

For example, for $\mathcal{S} = \{a,\ b,\ c\}$, and given two strings $L_1 = a\ b\ b\ a\ c\ c\ b\ a\ c$ and $L_2 = b\ a\ c\ b\ a\ a\ b\ c\ a$, we can find "$a\ b\ a\ b\ c$" as one of common subexpressions for L_1 and L_2.

"$a\ b\ a\ b\ c$" is one of common subexpressions:

$$L_1 = \underline{a}\ \underline{b}\ b\ \underline{a}\ c\ c\ \underline{b}\ a\ \underline{c}$$
$$L_2 = b\ \underline{a}\ c\ \underline{b}\ \underline{a}\ a\ \underline{b}\ \underline{c}\ a$$

$L_1[i]$ ($L_2[i]$) denotes ith symbol in L_1 (L_2).

Our problem considered here is to find a longest common subexpression for given two strings. We will approach this **longest common subexpression problem** with a **dynamic programming method**.

Let $p(i, j)$ be the length of the longest common subexpression of two strings $L_1[1]\ L_1[2] \cdots L_1[i]$ and $L_2[1]\ L_2[2] \cdots L_2[j]$.

7.3 Summary of the Course

The following topics will be studied in this course:

- Linear programming problems,
- Network optimization problems,
- Nonlinear programming problems,
- Combinatorial optimization problem.

The term "programming" means optimization and not computer programming.

Other Exciting Courses At JAIST

I232 Information Theory (1-2 Semester, April–June)

- Fundamentals of communication limits
- Discover limits without regard of block length or complexity
- Information compression, reliable transmission, multiuser communications
- Coding theory and information theory are two sides of a coin

I437 Coding Theory (2-1 Semester, October–December)

- Error-correcting codes are widely used in everyday communication systems
  - Wireless data, Wifi, data storage, ethernet
- Practical means for reliable communications over unreliable channels
- Other applications: Distributed storage, group testing, block chain, many more

Class Info

- Homework 8 (dynamic programming). Do not submit. Solutions will be published before final exam.
- Final exam: Wednesday, February 8 at 13:30–15:10.

Final Exam

The midterm is Wednesday, February 8 at 13:30–15:10. The exam is closed book. You may use:

- One page of notes, A4-sized paper, double-sided OK.
- Blank scratch paper

You may not use anything else: No printed materials, including books, lecture notes, and slides. No notes (except as above). No internet-connected devices. No calculators.

Exam Content

- Covers Chapters 5 and 6 only
- Differs from midterm: not all numerical problems, some analytical problems
- Problems similar to Homework 6–8. Solutions to Homework 6–8 are provided.
- Additional exercises to practice for exam:
  - Example 5.9 (try to solve without looking at solution)
  - Section 6.1.3, Exercise [1]
  - Section 6.3.4, Exercise [1]
  - Section 6.3.4, Exercise [2]

Course Evaluation Questionnaire

https: //gakumu.jaist.ac.jp/hcampus/

What was the best point about this course?

It was the best class in most aspects, with a level of completion not seen in any other class

Youtube movie

Best point is learning before the classes

Homework and pop quizzes kept me studying the course

Do you have any ideas for improving this course?

I want more numerical examples in lecture notes for each theory

More examples and clearer explanation, especially related to the process of Markov chain

What is the code rate mean in the actual world/ what is compression / how practical they are

More explanation about the coding homework