W1005
Intro to CS and Programming in MATLAB

Optimization

Fall 2014
Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/w1005

Outline

- Problem Formulation
- Optimization
- Toolbox

Writing a Best Seller

- Consider the following scenario: you are an aspiring writer trying to publish the next best-seller. You come across a secret report describing the “quality” of each word in the English vocabulary with respect to many characteristics (aspects)
- Each word also has a “popularity penalty” associated with it (people don’t like long and complicated text)
- Your goal is to choose your words wisely ensuring that they complement each other in terms of characteristics while minimizing the penalty

WBS – Problem Formulation

- More specifically:
  - Your text is represented by fraction (%) of each word in it
  - You have a vocabulary of N words to choose from
  - Each word 'x', has a penalty of C(x)
  - The quality report consists of K marks for each word. Each mark is a real value in the range [0.0,1.0] indicating the word's quality with respect to an aspect (i.e. 0.0 is 'no effect', 1.0 is 'maximum effect')
  - To ensure that your text is well written, you require that the total quality of all words for every aspect 'k' is at least some value B(k)
- Given all the relevant info (vocabulary, penalty, quality report), which words do you choose and how often (%), to make your text a best-seller?

WBS – Problem Formulation

- Assumptions:
  - Each word $0 \le x \le 1$, vocabulary size = N
  - C (penalty) is an N-by-1 (column) vector of positive real values
  - R is an N-by-K matrix of real values in the range [0.0,1.0]. Each row represents a word. Each column represents an aspect
  - B (total quality) is a 1-by-K (row) vector of positive real values
- All the relevant info is given. That is, {N,C,R,B} must be supplied to us
- Note: no guarantee that the parameters are set correctly
- Note: words are represented by percentages, hence the lower/upper bounds on ‘x’

Optimization

- "Do things best under the given circumstances"
- Applications in almost every field imaginable: planning, scheduling, resource allocation, management, traffic control
- Optimization problem:
    - Make a decision
    - Express/control the quality of the decision by the objective function
    - Typically a minimization/maximization task
    - Express "circumstances" that affect the decision as constraints
    - The type of opt. prob. is determined by the nature of the objective function and the constraints

Optimization – General Form

General From:

minimize $F(x)$

subject to: $g_i(x) \leq b_i \qquad i = 1,\dots,m$

- The problem is characterized by the objective function $F(x)$, and the constraints $g_i(x)$
- The variable or vector x, belongs to some domain/set S specified by the constraints
- Linear and Quadratic programs are the most frequent problems you are likely to encounter

MATLAB Optimization Toolbox

- Extensive package. Many routines, options. Plenty of documentation (‘help’ is not sufficient)
  - First step: define your problem clearly, write down your equations
  - Second step: find the appropriate solver (what type of problem are you solving?)
  - Third step: convert your problem to solver form. This might require combining equations, switching sign of equations & objective, adding equations
  - Fourth step: set options, call solver, examine the solution

1st Step – Approach

- Approach:
  1. Collect all parameters {N,C,R,B} % Input/load data
  2. Verify that parameters are set correctly % Error checking
  3. State the problem in mathematical notation:
     - We are clearly solving a constrained optimization problem
     - We are trying to minimize a linear objective (minimize the total penalty), subject to:
     - One equality constraint, total fractions of words sum to 1
     - K linear inequality constraints (total quality for some aspect is one constraint, we have K aspects)
     - Our variables must be real [0,1], text may consist of a single word or not contain a given word at all
  4. Output/verify the solution % Output/save solution

1st Step – Notation

- Mathematical notation:
  1. Let X be the variable/solution (column) vector, $X \in [0,1]^N$
  2. We wish to minimize the objective function C^TX
  3. One equality constraint: $\sum x_i = 1$
  4. K linear inequality constraints: $R^TX \geq B^T$
  5. Complete form:

$$\begin{array}{ll} \text{minimize} & C^TX \\ \text{subject to:} & R^TX \geq B^T \\ & 1^TX = 1 \\ & \forall i, X_i \in [0,1] \end{array}$$

2nd Step – Choose Solver

- Find appropriate solver:
  - http://www.mathworks.com/help/optim/index.html
  - Frequent solvers: linprog(), quadprog(), fmincon()

- Solver form example:
  - Linear program

$$\min_x f^T x \ such\ that \begin{cases} A \cdot x \le & b, \\ Aeq \cdot x = & beq, \\ lb \le x \le ub & \end{cases}$$

3rd Step – Solver Form

- MATLAB Toolbox notation:
  - Appropriate solver: linprog()
  - Why? Solution vector is a vector of real numbers (with bounds), objective is linear and the constraints are linear
  - Convert to solver form:

```
1. f = C;
2. A = -R';                % Change sign, transpose
3. b = -B';                % Change sign, column vector
4. Aeq = ones(1,N);        % 1^T X = 1
5. beq = 1;
6. lb = zeros(N,1);
7. ub = ones(N,1);
```

$$\min_x f^T x \text{ such that } \begin{cases} A \cdot x \leq & b, \\ Aeq \cdot x = & beq, \\ lb \leq x \leq ub & \end{cases}$$

4th Step – Call Solver

- Converted to solver form, variables {f, A, b, Aeq, beq, lb, ub}
- Function call options (syntax):

```
x = linprog(f)
x = linprog(f,A,b)
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,Beq,lb,ub,x0,options)
x = linprog(problem)
[x,fval] = linprog(...)
[x,fval,exitflag] = linprog(...)
[x,fval,exitflag,output] = linprog(...)
```

General Solver Rules

- Syntax rules (for all solvers):
  - Parameter not passed, assume it is empty
  - Parameter order is important
  - To include a subsequent parameter, but omit a preceding one, pass an empty array [ ]
  - ‘options’ is a struct specifying optimization method details. Ignore it, unless you know a thing or two about the field
  - Instead of passing many parameters, can pass a single struct ‘problem’ with appropriate fields
  - Output parameters include solution (x), value of objective function at the solution (fval), flag indicating outcome of call (exitflag), and details about the execution (output)

linprog() – Examples

- Parameter not passed, assume it is empty:
  - x = linprog(f)   % Minimize objective without constraints
- Include subsequent parameter, omit preceding one:
  - x = linprog(f, [], [],Aeq,beq)   % No inequality constraints
- Pass a single struct:
  1. problem.f = C;
  2. problem.Aineq = -R’;
  3. problem.solver = ‘linprog’;
  4. x = linprog(problem);
  - Notice that field names are slightly different
  - Must set all fields except x0 (set to empty if doesn’t exist)

linprog() – Examples

- Output parameters:
  - ‘x’: the solution vector
  - ‘exitflag’: if returns 1, problem solved successfully
  - ‘output’: a structure with solution details. For example, output.time is execution time
  - Can name parameters in any way you wish
  - [soln, fval, the_flag, soln_details] = linprog(problem)
  - If solution vector is not what it should be, you must check all output parameters to discover the problem. You should start with the ‘exitflag’, though there is no prescribed approach to detect a problem