Practical Optimization: Applications

INSEAD, Spring 2006

Jean-Philippe Vert

Ecole des Mines de Paris

Jean-Philippe.Vert@mines.org

Solvers and modeling language

How to solve an optimization problem?

- Use your own optimization routines
- Use a solver
- Use a modeling language

Trade-off between the *effort* required to perform the implementation and the *freedom* to chose the optimization problem (e.g., little effort for LP but you must then formulate your problem as a LP).

Custom code

- Use you own Newton / interior point routines
- Requires to explicitly define functions, gradients, Hessian
- No publicly-available general-purpose interior method, custom code is required
- Determining a valid barrier function is not trivial, in particular if the inequality constraint is non-differentiable
- Useful for problem that do not fit the particular cases handled by general solvers and modeling languages.

Standard form and solvers

- Most CP solvers are designed to handle certain prototypical problems known as *standard forms*, e.g., LP, QP ...
- They trade generality for ease of use and performance.
- Limitation: the transformation from your problem to a standard form is often not trivial (and prone to errors..)

Solver example

- MATLAB's `linprog` is a program for solving LP:

  `x = linprog( c, A, b, A_eq, b_eq, l, u )`

- Problems must be expressed in the following standard form:

$$
\begin{aligned}
\text{minimize} \quad & c^\top x \\
\text{subject to} \quad & Ax \leq b\,, \\
& A_{eq}x = b_{eq}\,, \\
& l \leq x \leq u
\end{aligned}
$$

- Converting to standard often requires many tricks

Smoothed convex CP

- A problem is *smooth* if both the objective and the constraints are twice continuously differentiable
- Several software packages solve smooth CP:
  - LOQO (primal/dual interior point method)
  - MOSEK (homogeneous algorithm)
- Requires custom code for gradient and Hessians
- Other packages exist for solving nonconvex smooth problems (but based on local convexity for the search direction)

Other standard forms

- Other standard forms with dedicated solvers exist:
  - Conic programs (SDP, SOCP..): SeDuMi, CDSP, SDPA, SDPT3, DSDP..
  - Geometric programs

Modeling frameworks

- Provide a convenient interface for specifying problems, and then by automating many of the underlying mathematical and computational steps for analyzing and solving them.
- Many excellent frameworks for LP, QP, smooth NLP:
  - Custom modeling language that allows models to be specified in a text file using a natural mathematical syntax: AMPL, GAMS, LINGO
  - Use spreadsheets as a natural, graphical user interface: What'sBest!, Frontline.
- These frameworks are built upon solvers that are called without any user's intervention

Advantages of modeling languages

- Convenient problem specification
- Standard form detection (LP, QP, NLP) to decide the best solver
- Automatic differentiation (for smooth NLP)
- Solver control: automatically calls the solver, pass the data value and provide reasonable default values

Summary

- If you have a nice standard form problem (LP, QP..) then using a modeling framework (e.g., with Excel) is probably the simplest
- Alternatively use directly a solver (e.g., input your own functions with gradient and Hessian)
- Alternatively, use custom code (e.g., non-smooth constraints, tricky barrier functions)

The CVX package

Motivation

- A (new) modeling framework for convex programming in MATLAB.
- Offers functions that can be called within other scripts
- Intuitive syntax
- Powerful features (e.g., non-smooth convex functions) that go beyond this course

Disciplined convex programming

- CVX can solve any convex program expressed in a particular form called disciplined convex programming
- Two key elements
  - An expandable *atom library*: a collection of functions and sets with known properties of convexity, monotonicity and range
  - A *ruleset* which governs how those atoms can be used and combined to form a valid problem (e.g., a sum of convex functions is ok).
- We will only use basic features in this course, because there are already quite a few atoms defined.

General syntax

```
cvx_begin
    variable x
    minimize( ... );
    subject to
        ...
cvx_end
```

After the last command the problem is solved and the solution returned in the variable `x`. The value of the minimum is available in the variable `cvx_optval`

Dual variables

```
cvx_begin
    variable x
    dual variable y
    minimize( ... );
    subject to
        y : ...
cvx_end
```

After the last command the optimal dual variable is available

in the `y` dual variable

Example: linear program

$$\begin{array}{rl} \text{minimize} & c^\top x \\ \text{subject to} & Ax \leq b \end{array}$$

```
n = size(A,2)
cvx_begin
    variable x(n);
    minimize( c' * x );
    subject to
        A * x <= b;
cvx_end
```

(see example_lp.m and exampl_lp2.m)

Example: QP with inequality constraints

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}x^\top Px + q^\top x + r \\ \text{subject to} & -1 \leq x \leq 1 \end{array}$$

```
cvx_begin
    variable x(n)
    minimize ( (1/2)*quad_form(x,P) + q'*x + r)
    x >= -1;
    x <=  1;
cvx_end
```

(see example_qp.m)

Example: sensitivity analysis for QCQP

We consider (ex. 5.1, homework 5):

$$\begin{array}{rl} \text{minimize} & x^2 + 1 \\ \text{subject to} & (x-2)(x-4) \leq u \end{array}$$

Compute the optimal value p^* as a function of u, and check that the optimal dual variable λ^* satisfies:

$$\frac{dp^*}{du} = -\lambda^*.$$

(see example_qcqp_sensitivity.m)

Example (cont.)

```
u = linspace(-0.9,10,50);
p_star = zeros(1,length(u));
lambda_star = zeros(1,length(u))
for i = 1:length(u)
    cvx_begin
        variable x(1)
        minimize ( quad_form(x,1) + 1 )
        lambda : quad_form(x,1) - 6*x + 8 <= u(i
    cvx_end
    p_star(i) = cvx_optval;
    lambda_star(i) = lambda
end
plot(u,-lambda_star,u,p_star)
```

Log-optimal investment strategy

The problem

- n assets held over N periods
- At the beginning of each period we re-invest our total wealth, redistributing it over the n assets using a fixed, constant, allocation strategy $x \in \mathbb{R}^n$ where $x \geq 0$ and $\sum_{i=1}^n x_i = 1$.
- We want to determine an allocation strategy x that maximizes growth of our total wealth for large N.

The model

- We use a discrete stochastic model to account for the uncertainty in the returns
- During each period there are m possible scenarios with probabilities $\pi_1, \ldots, \pi_m$.
- In scenario j the return for asset i over one period is given by p_{ij}
- We assume the same scenarios for each period, with identical independent distributions.

Formalization

- Let $W(t-1)$ our wealth at the beginning of period t.
- During period t we therefore invest $x_i W(t-1)$ in asset i.
- If scenario j happens in period t then our wealth at the end of period t is:

$$W(t) = \sum_{i=1}^{n} p_{ij} x_i W(t-1)$$

- The total return during period t is therefore:

$$\lambda(t) = \frac{W(t)}{W(t-1)} = p_j^\top x \, .$$

Growth rate

- At the end of the N periods our wealth has been multiplied by the factor $\prod_{t=1}^{N} \lambda(t)$
- The growth rate of the investment over the N periods is

$$G_N = \frac{1}{N} \sum_{t=1}^{N} \log \lambda(t)$$

- By the law of large numbers, for large N:

$$\lim_{N \to \infty} G_N = E \log \lambda(t) = \sum_{j=1}^{m} \pi_j \log \left(p_j^\top x \right) \ .$$

Optimization problem

- The problem can therefore be formulated as:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{m} \pi_j \log\left(p_j^\top x\right) \\
\text{subject to} \quad & x \geq 0 \, , \\
& 1^\top x = 1 \, .
\end{aligned}
$$

- The investment strategy $x \in \mathbb{R}^n$ that solves this problem is called the *log-optimal investment strategy*.
- This is a convex optimization problem with differentiable objective and constraints.

Example

- 5 assets, 10 equiprobable scenarios.
- Asset 1 is very risky, with occasional large return but (most of the time) substantial loss
- Asset 5 gives a fixed and certain return of 1%.

(see example_logoptimalportfolio.m)

Scenarios

```
P = [3.5000    1.1100    1.1100    1.0400    1.0100;
     0.5000    0.9700    0.9800    1.0500    1.0100;
     0.5000    0.9900    0.9900    0.9900    1.0100;
     0.5000    1.0500    1.0600    0.9900    1.0100;
     0.5000    1.1600    0.9900    1.0700    1.0100;
     0.5000    0.9900    0.9900    1.0600    1.0100;
     0.5000    0.9200    1.0800    0.9900    1.0100;
     0.5000    1.1300    1.1000    0.9900    1.0100;
     0.5000    0.9300    0.9500    1.0400    1.0100;
     3.5000    0.9900    0.9700    0.9800    1.0100];
```

Solving the problem with CVX

```
[m,n] = size(P);

cvx_begin
    variable x_opt(n)
    maximize(geomean(P*x_opt))
    sum(x_opt) == 1
    x_opt >= 0
cvx_end

x_opt
x_unif = ones(n,1)/n
R_opt = sum(log(P*x_opt))/m
R_unif = sum(log(P*x_unif))/m
```

Solution

- The log-optimal investment strategy is:

$$x_{opt} = [0.0580 \quad 0.3995 \quad 0.2921 \quad 0.2504 \quad 0.0000]^\top$$

- The long-term growth rate achieved is $R_{opt} = 2.31\%$
- The long-term growth rate achieved by the uniform strategy is $R_{unif} = 1.14\%$
- The optimal strategy is to invest very little on the very risky asset, and nothing on the sure asset. Most of the wealth goes to asset 2.