CVXPY Exercises

Steven Diamond

TCMM 2014

1. *Hello world.* Solve the following optimization problem using CVXPY:

$$\begin{array}{ll} \text{minimize} & x^2 - 2\sqrt{y} \\ \text{subject to} & 2 \geq e^x \\ & x + y = 5, \end{array}$$

   where $x, y \in \mathbf{R}$ are variables.

   Find the optimal values of x and y.

2. *Non-negative least squares.* We wish to recover a sparse, non-negative vector $x \in \mathbf{R}^n$ from measurements $y \in \mathbf{R}^m$. Our measurement model tells us that

$$y = Ax + v,$$

   where $A \in \mathbf{R}^{n \times m}$ is a known matrix and $v \in \mathbf{R}^m$ is unknown measurement error. The entries of v are drawn IID from the distribution $\mathcal{N}(0, \sigma^2)$.

   We can recover a good estimate of x by solving the optimization problem

$$\begin{array}{ll} \text{minimize} & ||Ax - y||^2 \\ \text{subject to} & x \geq 0. \end{array}$$

   This problem is called non-negative least squares.

   The file `nnls.py` defines n, m, A, x, and y. Use CVXPY to estimate x from y. First try standard regression, *i.e.*, solve

$$\text{minimize} \quad ||Ax - y||^2.$$

   Use the plotting code in `nnls.py` to compare the estimated x with the true x. Add the constraint $x \geq 0$ and see how it affects the estimate.

   How many measurements n are needed for standard regression to find an accurate x? How about non-negative least squares?

3. *Minimum fuel optimal control.* We consider a vehicle moving along a 2D plane. The vehicle's state at time t is described by $x_t \in \mathbf{R}^4$, where $(x_{t,0}, x_{t,1})$ is the position of the vehicle in two dimensions and $(x_{t,0}, x_{t,1})$ is the vehicle velocity. At each time t a drive force $(u_{t,0}, u_{t,1})$ is applied to the vehicle.

   The dynamics of the vehicle's motion is given by the the linear recurrence

   $$x_{t+1} = Ax_t + Bu_t, \quad t = 0, \ldots, N-1,$$

   where $A \in \mathbf{R}^{4\times 4}$ and $B \in \mathbf{R}^{4\times 2}$ are given. We assume that the initial state is zero, *i.e.*, $x_0 = 0$.

   The *minimum fuel optimal control problem* is to choose the drive force $u_0, \ldots, u_{N-1}$ so as to minimize the total fuel consumed, which is given by

   $$F = \sum_{t=0}^{N-1} f(u_t),$$

   subject to the constraint that $x_N = x_{\mathrm{des}}$, where N is the (given) time horizon, and $x_{\mathrm{des}} \in \mathbf{R}^4$ is the (given) desired final or target state. The function $f : \mathbf{R}^2 \to \mathbf{R}$ is the *fuel use map* and gives the amount of fuel used as a function of the drive force.

   We will use

   $$f(a) = \|a\|_2^2 + \gamma\|a\|_1.$$

   The file `optimal_control.py` defines N, A, B, and x_{des}. Use CVXPY to solve the minimum fuel optimal control problem for $\gamma \in \{0, 1, 10, 100\}$.

   Use the plotting code in `optimal_control.py` to plot x and u for each γ.

4. *Power grid single commodity flow.* Recall the definition of a single commodity flow problem from the talk:

   $$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \phi_i(f_i) + \sum_{j=1}^p \psi_j(s_j), \\ \text{subject to} & \text{zero net flow at each node} \end{array}$$

   where f_i is the flow on edge i, s_j is the external source/sink flow into node j, and ϕ_i, ψ_j are convex cost functions.

   We will apply the single commodity flow framework to a power grid. Let nodes $\{1, \ldots, k\}$ be generators. The output at generator j is s_j. Each generator has the constraint $0 \leq s_j \leq U_j$ for some maximum output U_j and the cost function $\psi_j(s_j) = s_j^2$.

   Nodes $\{k+1, \ldots, p\}$ are consumers. Each consumer j has a fixed load L_j, meaning $s_j = L_j$.

   Each edge i has the constraint $|f_i| \leq c_i$ for some capacity c_i and the cost function $\phi_i(f_i) = f_i^2$. The edge flow cost represents power loss.

Explicitly, the power grid single commodity flow problem is

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{n} f_i^2 + \sum_{j=1}^{k} s_j^2, \\
\text{subject to} & \text{zero net flow at each node} \\
& 0 \leq s_j \leq U_j, \quad j = 1, \ldots, k \\
& s_j = L_j, \quad j = k+1, \ldots, p \\
& |f_i| \leq c_i, \quad i = 1, \ldots, n.
\end{array}
$$

The file `power_grid.py` defines the power grid graph, the maximum generator outputs U, loads L, and edge capacities c. Complete the classes `Generator`, `Consumer`, and `CapEdge` and use them to solve the power grid single commodity flow problem.

Use the plotting code in `power_grid.py` to plot the edge flows in the solution.

5. *Extra* I've included the code for the total variation in-painting example from the talk in `inpainting.py`. Feel free to play around with it if you have time. Try changing `PROB_PIXEL_LOST` to increase or decrease the number of known pixels.