Homework 5

ORF522: Linear and Nonlinear Optimization

Instructor: Bartolomeo Stellato
AI: Irina Wang

Due on December 18

Problem 1 – Optimality conditions for ℓ_1-regularized minimization.

Consider the problem of minimizing

$$\phi(x) = f(x) + \lambda \|x\|_1,$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and differentiable, and $\lambda \geq 0$. The number λ is the regularization parameter, and is used to control the trade-off between small f and small $\|x\|_1$. When ℓ_1-regularization is used as a heuristic for finding a sparse x for which $f(x)$ is small, λ controls (roughly) the trade-off between $f(x)$ and the cardinality (number of nonzero elements) of x.

1. Show that $x = 0$ is optimal for this problem (*i.e.,* minimizes ϕ) if and only if $\|\nabla f(0)\|_\infty \leq \lambda$. In particular, for $\lambda \geq \lambda^{\max} = \|\nabla f(0)\|_\infty$, ℓ_1 regularization yields the sparsest possible x, the zero vector.
2. Did your proof in (Q1.1) use the fact that f is convex? Explain why or why not.

Remark. The value $\lambda^{\max}$ gives a good reference point for choosing a value of the penalty parameter λ in ℓ_1-regularized minimization. A common choice is to start with $\lambda = \lambda^{\max}/2$, and then adjust λ to achieve the desired sparsity/fit trade-off. Useful values of λ typically range between $0.05\lambda^{\max}$ and $0.9\lambda^{\max}$.

Problem 2 – Diode relation

The operator D defined by its graph

$$\mathbf{gph} D = \{(x_1, x_2) \mid x_1 x_2 = 0, \quad x_1 \leq 0, \quad x_2 \geq 0\},$$

is called the *diode relation*, since it is the V-I characteristic of an ideal diode.

1. Show that D is monotone
2. Find the resolvent and the Cayley operator of λD (with $\lambda > 0$).
3. Plot the operator D, the resolvent $R_{\lambda D}$, and the Cayley operator $C_{\lambda D}$.

Problem 3 – Solving LPs via alternating projections

Consider an LP in standard form,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array}$$

with variable $x \in \mathbf{R}^n$, and where $A \in \mathbf{R}^{m \times n}$.

A tuple $(x, \nu, \lambda) \in \mathbf{R}^{2n+m}$ is primal-dual optimal if and only if

$$Ax = b, \quad x \geq 0, \quad \lambda \geq 0, \quad -A^T\nu - c + \lambda = 0, \quad c^T x + b^T \nu = 0.$$

These are the KKT optimality conditions of the LP. The last constraint, which states that the duality gap is zero, can be replaced with an equivalent condition, $\lambda^T x = 0$, which is complementary slackness.

1. Let $z = (x, \nu, \lambda)$ denote the primal-dual variable. Express the optimality conditions as $z \in \mathcal{A} \cap \mathcal{C}$, where $\mathcal{A}$ is an affine set, and $\mathcal{C}$ is a simple cone. Give $\mathcal{A}$ as $\mathcal{A} = \{z \mid Fz = g\}$, for appropriate F and g.
2. Explain how to compute the Euclidean projections onto $\mathcal{A}$ and also onto $\mathcal{C}$.
3. Here is a simple method to generate LP instances that are feasible. First, generate a random vector $\omega \in \mathbf{R}^n$. Let $x^\star = \max\{\omega, 0\}$ and $\lambda^\star = \max\{-\omega, 0\}$, where the maximum is taken elementwise. Choose $A \in \mathbf{R}^{m \times n}$ and $\nu^* \in \mathbf{R}^m$ with random entries, and set $b = Ax^\star$. Construct a cost vector c such that ($x^\star$, $\nu^\star$, $\lambda^\star$) are the optimal primal-dual variables of our LP in standard form. What is the expression of c?
4. Implement *alternating projections* to solve the standard form LP, *i.e.*,

$$\begin{aligned} \tilde{z}^{k+1} &= \Pi_{\mathcal{A}}(z^k) \\ z^{k+1} &= \Pi_{\mathcal{C}}(\tilde{z}^{k+1}) \end{aligned}$$

   Your implementation must exploit factorization caching in the projection onto $\mathcal{A}$, and but you do not need to exploit the structure in the matrix F. See the documentation for `scipy.sparse.linalg.splu` or `scipy.sparse.linalg.factorized` for factorization caching. You are welcome to use other factorization methods.

Test your solver on a problem instance with $m = 100, n = 500$. Plot the residual $\|z^{k+1} - \tilde{z}^{k+1}\|_2$ over 1000 iterations. (This should converge to zero, although perhaps slowly.)

5. Write down the steps and implement ADMM on the same problem instances from point (Q3.4). Verify that you obtain a speedup, and plot the same residual as in point (Q3.4). You can plot the residuals from (Q3.4) and (Q3.5) on the same plot.

Problem 4 – ADMM for smart grid device coordination

We consider an electrical grid consisting of N devices that exchange electricity over T time periods. Device i has energy profile $p^i \in \mathbf{R}^T$, with p^i_t denoting the energy consumed by device i in time period t, for $t = 1, \ldots, T,\ i = 1, \ldots, N$. (When $p^i_t < 0$, device i is producing energy in time period t.)

Each device has a convex objective function $f_i : \mathbf{R}^T \to \mathbf{R}$, which we also use to encode constraints, by setting $f_i(p^i) = \infty$ for profiles that violate the constraints of device i. In each time period the energy flow has to balance, which means

$$\sum_{i=1}^{N} p^i_t = 0, \quad t = 1, \ldots, T.$$

The optimal profile coordination problem is to minimize the total cost, $\sum_{i=1}^{N} f_i(p^i)$, subject to the balance constraint, with variables $p^i, i = 1, \ldots, N$.

In this problem you will use ADMM to solve the optimal profile coordination problem in a distributed way, with each device optimizing its own profile, and exchanging messages to coordinate all of the profiles.

From this point on, we consider a specific (and small) problem instance. There are three devices: a generator, a fixed load, and a battery, with cost functions described below.

- *Generator.* The generator has upper and lower generator limits:

$$p^{\min} \le -p_t \le p^{\max}, \quad t = 1, \ldots, T.$$

  Note the minus sign, since a generator's profile is typically negative, using our convention. The objective is

$$f_{\text{gen}}(p) = \sum_{t=1}^{T} (\alpha(-p_t)^2 + \beta(-p_t)),$$

  where $\alpha, \beta > 0$ are given constants.

- *Fixed load.* The fixed load has zero objective function and the constraint that its power profile must equal a given consumption profile $d \in \mathbf{R}^T$.
- *Battery.* The battery has zero objective function, and charge/discharge limits given by C and D, respectively:

  $$-D \leq p_t \leq C, \quad t = 1, \dots, T.$$

  The battery is initially uncharged (*i.e.,* $q_1 = 0$), so its charge level in period t is $q_t = \sum_{\tau=1}^{t-1} p_\tau$ (we neglect losses for this problem). The charge level must be nonnegative, and cannot exceed the battery capacity:

  $$0 \leq q_t \leq Q, \quad t = 1, \dots, T+1.$$

  The charge level is extended to time $T+1$ to allow the battery to charge/discharge in time T, subject to the operational constraints.

1. Use CVXPY to solve the problem with data generated by following codes.

   ```
   import numpy as np
   T = 100               # planning horizon
   alpha, beta = 0.2, 1  # generator cost parameters
   pmin, pmax = 0, 10    # generator limits
   Q, C, D = 5, 1, 1     # battery parameters

   # fixed load schedule
   d = np.zeros((T,1))
   d[0:20]=3; d[20:56]=5; d[56:60]=11;
   d[60:80]=4; d[80:90]=10; d[90:100]=3
   ```

   Plot the (optimal) power profile for the generator and battery, as well as the battery charge level. Show that the optimal value is 1065.004.

2. Write down the steps of ADMM and implement for this problem (you may use CVXPY to solve each device's local optimization in each ADMM iteration). Experiment with a few values of the parameter $\rho = \frac{1}{\lambda}$ from lecture to see its effect on the convergence rate of the algorithm. Plot the norm of the energy balance residual, versus iteration. Plot the power profiles of the generator and battery, as well as the energy balance residual, for several values of iteration (say, after one iteration, after 10 iterations, and after 50). Check the results against the solution found by CVXPY.

Problem 5 – The shortest LP solver

Consider the linear program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array}$$

with $c \in \mathbf{R}^n, A \in \mathbf{R}^{m \times n}, b \in \mathbf{R}^m$. In this problem, we will consider the shortest known LP solver. For a history of short LP solves and a few other examples, we refer the interested reader to *Jacob Mattingly's collection*. The LP solver we consider is adapted from one developed by Borja Peleato.

1. Show that the following (Julia) code implements ADMM for the LP above, starting from any initial iterates x and $z \in \mathbf{R}^n$.

   ```
   for i=1:99;z=max(z-x,x-c);x=z+A\(b-A*z);end
   ```

   Derive the ADMM steps.

2. Implement this LP solver using Python, and try it on a few randomly generated LPs. Generate random LPs in the same way as you did for question 3 with $m = 10$ and $n = 20$. This will give you an optimal $x^\star$, but it may not be unique. Run your algorithm for 1000 iterations with $\rho = 1$. Plot the objective difference, $c^T x^k - c^T x^\star$, over the iterations. Additionally, show that your solution in the last iteration is feasible.

3. Extra credit: find an LP solver whose code is fewer than 44 character (in your language of choice).