Homework 3

Due date: Friday, November 16.

In this homework, you will have to solve convex problems numerically. You are free to do this any way you like but I encourage you to check `cvx` (just google "cvx Stanford"). `cvx` is the wrapper I used in my demos. It is nice because it lets you solve convex problems the way you write them down on paper. For instance, suppose we want to solve

$$\min_{x\in\mathbb{R}^n} \|y - Ax\|_{\ell_1}.$$

In `cvx`, this is done as follows:

```
cvx_begin
    variable x(n);
    minimize(norm(y - A*x,1));
cvx_end
```

You can also add constraints. For instance,

$$\min_{x\in\mathbb{R}^n} \|y - Ax\|_{\ell_1} \quad \text{subject to} \quad x \geq 0$$

becomes

```
cvx_begin
    variable x(n);
    minimize(norm(y - A*x,1));
    subject to
        x >= 0;
cvx_end
```

Alternatively, you can use the command `linprog` to solve linear programs in matlab. A good description of how `linprog` operates is available at `http://www.mathworks.com` or alternatively you can type `help linprog` in matlab.

1. *Error correction*. How can we transmit information reliably? This problem reviews the example we studied in class.

   The problem is as follows: we wish to transmit a vector $x_0 \in \mathbb{R}^n$ using a linear code; a linear code is a given collection of codewords which are vectors $a_1, \ldots, a_n \in \mathbb{R}^m$—the columns of a matrix A. Given a vector $x_0 \in \mathbb{R}^n$ (the "message") we can then generate a vector Ax_0 in $\mathbb{R}^m$ (the "codeword"). A recurrent problem with real communication or storage devices is that some of the entries of Ax_0 may become corrupted; when this is the case, the corrupted entries are unreliable and may have nothing to do with the original values.

   In class, when have seen that when the fraction of corrupted data is not too large, the solution to the problem

   $$\min_{x\in\mathbb{R}^n} \|y - Ax\|_{\ell_1},$$

   is exact.

   (a) Set $n = 128$ and $m = 512$. Make up a message x_0 in $\mathbb{R}^n$, and generate a Gaussian matrix `A = randn(m,n)`.

(b) Make up Ax_0, corrupt about 10% of its entries (explain how you do this), and obtain the corrupted codeword y.

(c) Solve the optimization problem and compare its solution with the original message. What do you observe? Repeat this experiment for several instances of the message and of the corruption (keeping the corruption rate at about 10%).

(d) Summarize your findings.

2. *Error correction with gross and small errors*. The setup is as in the previous problem. In truth, it may not be realistic to assume that except for some gross errors, one is able to receive the values of Ax_0 with infinite precision. A better model would assume instead that the receiver gets

$$y = Ax_0 + z_0, \qquad z_0 = e + z, \tag{1}$$

where e is a possibly sparse vector of gross errors and z is a vector of small errors affecting all the entries. In other words, one is willing to assume that there are malicious errors affecting a fraction of the entries of the transmitted codeword *and* in addition, smaller errors affecting all the entries. For instance, one could think of z as some sort of quantization error which limits the precision/resolution of the transmitted information. In this more practical scenario, is it still possible to recover the signal x_0 accurately?

(a) Consider a first proposal where one solves

$$(P_2) \quad \min \|y - Ax - z\|_{\ell_1} \quad \text{subject to} \quad \begin{array}{r} \|z\|_{\ell_2} \le \varepsilon, \\ A^* z = 0, \end{array} \tag{2}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$. The parameter ε above depends on the magnitude of the small errors.

i. Does this make sense? Explain.

ii. Recast this as a type of convex optimization problem that we have seen.

(b) Consider now a second proposal where one solves

$$(P_1) \quad \min \|y - Ax - z\|_{\ell_1} \quad \text{subject to} \quad \begin{array}{r} \|z\|_{\ell_\infty} \le \delta, \\ A^* z = 0, \end{array} \tag{3}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$. The parameter δ above depends on the magnitude of the small errors.

i. Does this make sense? Explain.

ii. Recast this as a type of convex optimization problem that we have seen.

(c) Suppose there were no gross errors in the channel ($e = 0$), how would you recover the message? What performance would you expect? That is, what would the typical squared error $\|\hat{x}^{Ideal} - x_0\|_{\ell_2}^2$ be? ($\hat{x}^{Ideal}$ is your reconstruction.)

(d) We will now perform numerical experiments of the two proposals and compare their performance, both with each other and with the performance one expects if there are no gross errors.

i. Obtain a corrupted codeword $y = Ax_0 + e + z$ where $Ax_0 + e$ is generated the same way as in Problem 1 (take `x0 = randn(n,1)` for simplicity) and the small error vector z is generated `z = randn(m,1).*sigma` where `sigma = .21`. (`sigma` is chosen so that approximately the first four binary digits of Ax are reliable.)

ii. Solve the optimization problem (P_2) with $\varepsilon = 4.4$ to obtain an estimate $\hat{x}$ of x. (The parameter ε is chosen so that the true x_0 used to generate y will be feasible with very high probability.) Calculate

$$\rho = \frac{\|\hat{x} - x_0\|_{\ell_2}}{\|\hat{x}^{Ideal} - x_0\|_{\ell_2}}$$

where $\|\hat{x}^{Ideal} - x_0\|_{\ell_2}^2$ was calcuated in part (c). Repeat this experiment several times. What do you observe? What does it mean if ρ is close to one?

iii. Repeat (ii) but instead solve (P_1) with $\delta = .84$. (Again, δ is chosen so that the true x_0 is feasible with high probability.)

iv. Summarize your findings.

3. Boyd and Vandenberghe, Chapter 4, Problem 5.14.

4. *Bonus problem (you get lots of extra credit if you do this):* Implement the simplex algorithm. Your program should take an LP in standard form and return the solution. I am not asking you to handle large problems. If you can handle a number of variables and a number of equality constraints in the twenties and thirties, I will be happy.