The University of Texas at Austin
Department of Computer Science

Online Learning and Optimization

Problem Set Two

Constantine Caramanis — Due: Not Due.

Submission. There is nothing to hand in here as the problems are optional.

About this problem set. There are two aspects of this problem set. The first is to provide us with some high level motivation for this class. As the lectures have mentioned, this class is not about formulations – it is about designing algorithms. So before we dive into that, this problem set gives some taste as to what are the kinds of problems you'll be able to solve. But since we have not yet learned about any algorithms, you will use a solver instead. In other words, you will use a black box to solve some problems. The rest of this class will essentially be digging into this black box.

The second is to give you a few practice linear algebra problems. As Lecture 1.4 mentions, the main tools we need for this class are linear algebra, and some basics of multivariable calculus.

Computational Problems

Set up CVXOPT `http://cvxopt.org/` – a suite of solvers for convex optimization problems and CVXPY `http://www.cvxpy.org/en/latest/` – a convenient modeling language for convex optimization problems, and frontend for CVXOPT. CVXPY is also able to call other optimization solvers.

1. Read Example 6.3 (Optimal input design) from BV, and reproduce the results using Python and CVXPY.

2. If you are not familiar with denoising and smoothing, read about it, for example, in Section 6.3.3 of the freely available book by Stephen Boyd and Lieven Vandenberghe (`http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`). This is the section on Reconstruction, smoothing and de-noising. Look at the in-painting-using-denoising example here `http://nbviewer.jupyter.org/github/cvxgrp/cvxpy/blob/master/examples/notebooks/WWW/tv_inpainting.ipynb` and make sure you can run the Notebook (you will have to download the images).

   (a) Figure out how to use your own grayscale image (of you, or of something else you choose).

   (b) Add typed noise, as in the example, and try to recover the original image. Display the recovered version, and the difference to the original.

   (c) Delete each pixel at random, with probability p. For which values of p can you recover something close to the original image?

   (d) Delete an entire $B \times B$ block somewhere in the image. Can you recover the deleted pixels using TV? Explain why or why not.

(e) The original problem notes that the SCS solver is much faster than the CVXOPT solver. Try the CVXOPT solver. What is the difference in solving time? This class will give us the tools to try to understand why one solver works well (quickly) in some domains but not others.

3. Sparse Recovery. Consider the following problem. There is an *unknown* vector $\beta \in \mathbb{R}^p$. We get n noisy linear measurements of β: $y_i = \langle x_i, \beta \rangle + e_i$, $i = 1, \ldots, n$. We write this in matrix notation:

$$y = X\beta + e,$$

where X is a $n \times p$ matrix. Here we have $n < p$. Typically, this means that the problem is *under-determined*: there are more unknowns than constraints. However, it turns out that if β is sparse, then in many cases it is possible to recover β from fewer measurements than dimensions. In this problem you will explore solving this problem when β is sparse.

(a) Generate random data for three versions of this problem, all of different size:

$$(p, n) \in \{(50, 500), (500, 5000), (5000, 50000)\}.$$

In all three cases, let X be a $n \times p$ matrix where each entry is drawn at random from a standard Gaussian ($N(0, 1)$). Let $\beta \in \mathbb{R}^p$ be a vector that has all zeroes, except for a "+1" at $k = 5$ randomly chosen locations. For each of the three sizes, generate y according to $y = X\beta + e$, where each entry of e is a $N(0, 0.1)$ random variable.

Now also generate $100 \times p$-size *testing data*, $(X_{\text{test}}, y_{\text{test}})$, using the same random procedure to generate X_{test} and the noise, but using the same values of β you created above. You will use this test data to check the quality of solutions you obtain.

(b) Algorithm 1: Least Squares.

Compute a least-squares solution to the problem by solving:

$$\text{minimize :} \qquad \|X\beta - y\|_2^2$$

Figure out how to solve this problem using CVXPY. Ask CVXPY to solve each of the three problems, and report: (a) Did CVXPY succeed? (b) If so, how long did it take to solve each instance? (Specify the solver you used). (c) Plot the β you recover, and plot the true β. Are these close? (d) Report the Regression error of the solution computed: $\|X\beta^* - y\|_2$ and also the Testing error: $\|X_{\text{test}}\beta - y_{\text{test}}\|_2$.

(c) Algorithm 2: Sparse Recovery via an optimization-based algorithm called LASSO. (We will talk about Lasso in great detail, in particular, in Part 1, Topic 5).

Instead of minimizing the squared error, the idea is to "encourage" the solution to be sparse, by minimizing a *penalized* version of the squared error – the squared error plus a small multiple of the ℓ^1-norm of the solution:

$$\text{minimize :} \qquad \|X\beta - y\|_2^2 + \lambda\|\beta\|_1.$$

Ask CVXPY to solve each of the three problems, using $\lambda = 0.001$, and report: (a) Did CVXPY succeed? (b) If so, how long did it take to solve each instance? (c) Report the Regression error of the solution computed: $\|X\beta^* - y\|_2$ and also the Testing error: $\|X_{\text{test}}\beta - y_{\text{test}}\|_2$. (d) What is the support of β? That is, what are the non-zero coefficients of β. (e) Can you find a better λ?

Written Problems

(1) Consider the matrix

$$X = \begin{bmatrix} 4 & 5 & 12 \\ 5 & 16 & 1 \\ 12 & 1 & 16 \end{bmatrix}.$$

This matrix is symmetric, and therefore it has real eigenvalues. Check this. However, this matrix is not positive semidefinite. Find a vector $\boldsymbol{v} \in \mathbb{R}^3$ such that

$$\boldsymbol{v}^\top X \boldsymbol{v} < 0.$$

(2) The notes claim that if a matrix is symmetric, you can write it in its eigen-decomposition, namely, you can find real scalars $\lambda_1, \lambda_2, \lambda_3$ and vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ such that

$$X = \lambda_1 \boldsymbol{v}_1 + \lambda_2 \boldsymbol{v}_2 + \lambda_3 \boldsymbol{v}_3,$$

where $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ are *orthonormal vectors*, i.e., $\|\boldsymbol{v}_i\|_2 = 1$ for $i = 1, 2, 3$, and their dot products are zero:

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = \boldsymbol{v}_i^\top \boldsymbol{v}_j = 0, \quad \forall i \neq j.$$

Find the λ_i and $\boldsymbol{v}_i$, and show that in fact the $\boldsymbol{v}_i$ are orthonormal, as claimed.