The University of Texas at Austin
Department of Electrical and Computer Engineering

EE381K: Large Scale Optimization — Fall 2015

Problem Set Nine

Constantine Caramanis   Due: Thursday, November 19, 2015.

Matlab and Computational Assignments. Please provide a printout of the Matlab code you wrote to generate the solutions to the problems below. Problems marked with (?) are optional.

1. (**FISTA**) In Problem Set 8, we revisited the Lasso problem:

$$\min_{\beta}: \ \|X\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_1,$$

and you compared subgradient descent with the proximal gradient method. Combine the proximal gradient method with Nesterov's acceleration, and plot the result compared to the results you obtained on last week's homework. As you derived on last week's homework, the proximal operator of the $\|\cdot\|_1$-norm is a soft thresholding operator. Thus, the proximal gradient algorithm for Lasso is known as the Iterative Soft Thresholding Algorithm, or ISTA. The accelerated version of this, which you implement here, is called the Fast ISTA, or, FISTA. You can see this clearly described in Chapter 5 of Sebastien Bubeck's notes, which I posted last time.

2. (**Frank-Wolfe**) Again for the same problem as above, use the conditional gradient (Frank-Wolfe) algorithm to obtain a sparse solution, and compare running time, error and sparsity, to your previous solutions. For Frank-Wolfe, we need to solve the problem over a polytope. For the sake of illustrating this method, we will "cheat" and solve it over the polytope $\{\beta \mid \|\beta\|_1 \leq r, \, \beta \geq 0\}$, where $r = \|\hat{\beta}\|$ is the ℓ^1-norm of the solution you obtained through a different method in the previous problem.

3. (**Mirror Descent**) Consider now the problem of robust regression, where some small number of measurements have been potentially completely corrupted. One way to formulate an optimization problem to solve this robust regression is as follows:

$$\begin{aligned} \min_{\beta}: & \quad \|X\beta - \mathbf{y}\|_1 \\ \text{s.t.}: & \quad \beta \in \mathfrak{X}. \end{aligned}$$

The rationale for this formulation stems from the idea that because of the ℓ^1-error, huge errors are not disproportionally penalized, as they would be in the squared error formulation (this is the formulation we have worked with before, including in the previous problem), and therefore the optimal solution is less sensitive to outliers. You will solve this problem using Projected Subgradient Descent, and also Mirror Descent. Let the constraint set be the simplex:

$$\mathfrak{X} = \{\beta : \beta \geq 0, \sum \beta_i = 1\}.$$

   (a) Write down the update for projected gradient descent.

(b) Write down the mirror descent update, using your work from the last written problem below. For this, we will use the same mirror map that we used in class: $\Phi(\beta) = \sum \beta_i \log \beta_i$. Compute the Bregman divergence, $D_{phi}(\mathbf{x}, \mathbf{y})$ explicitly. Write down the mirror descent update, using stepsize η.

(c) Using the data in the file `robust_regression.mat`, and using stepsizes of your choosing, compare the projected subgradient method with mirror descent. What is β?

(d) (?) Explore the robustness properties of this formulation. Create your own data set and corrupt a few values. Compare the performance of standard regression (least squares), with the above formulation.

4. (**Matrix Completion**) In this problem we investigate **low-rank matrix completion**, the problem of finding a low-rank matrix given only a few (randomly sampled entries). While this is (clearly) not possible in general, somewhat remarkably, it is possible once some additional assumptions are made on the problem setup (for example, for a "random" low-rank matrix and random samples). We will learn more about these assumptions next semester, but for now we will develop a projected subgradient algorithm to solve such a problem.

Suppose there is a true matrix $M \in \mathbb{R}^{m \times n}$ that we want to recover, but we are only given elements in the set $\Omega \subset [m] \times [n]$ (i.e. we know the value of m_{ij} if $(i,j) \in \Omega$). We want to solve the following constrained optimization problem

$$\begin{array}{lll} \min\limits_{X} & & \|X\|_* \\ & s.t. & x_{ij} \; = \; m_{ij} \;\; \text{for all } (i,j) \in \Omega \end{array}$$

where the variable of optimization $X \in \mathbb{R}^{m \times n}$ is a matrix. Here $\|\cdot\|_*$ is the "nuclear" norm, equal to the sum of singular values of the matrix. This norm is a convex but not smooth function of X; we will implement projected sub gradient descent for this problem.

The sub gradient of the $\|\cdot\|_*$ function is as follows: for any matrix X, if its SVD is $U\Sigma V'$, then a matrix $Z \in \partial\|X\|_*$ is in its sub gradient if and only if

$$Z \; = \; UV' + W$$

where W is such that *(a)* the column and row spaces of W are perpendicular to the corresponding ones of X, and *(b)* the spectral norm $\|W\|_2 \leq 1$. Recall that the spectral norm of a matrix is its maximum singular value. Also recall that if X is rank r, then the matrices U, V are of sizes $m \times r$ and $n \times r$ respectively, and have orthonormal columns.

(a) Given a matrix X, how will you generate an element $Z \in \partial\|X\|_*$ using the `svd` function in matlab ?

(b) Given a matrix X, how will you project it onto the feasible set (i.e. the set of matrices that satisfy the constraints) ?

(c) Implement projected sub gradient descent with two choices for step sizes: $\eta_k = \frac{1}{k}$ and $\eta_k = \frac{1}{\sqrt{k}}$. You will need to use the file `matrix_completion.mat`, which contains two 100×100 matrices: a low-rank matrix M, and the matrix O that represents the set Ω by having entries that are 0 or 1 (in particular, $o_{ij} = 1$ means $(i,j) \in \Omega$).

(d) Plot the relative error $\frac{1}{100^2}\|M - X_k\|_F^2$ between the true matrix and the k^{th} iterate, as a function of k, for both step size choices; do so on one plot.

(e) What is the rank of the intermediate iterates ? Why is this the case ?

Written Problems

1. Prove that a matrix Z as described above in the matrix completion problem is indeed a sub gradient to the nuclear norm function at X. You can use the following fact about the nuclear norm: for any matrix $M \in \mathbb{R}^{m\times n}$, let $s = \min(m,n)$. Then for any matrices $A \in R^{m\times s}$ and $B \in R^{n\times s}$ that have orthonormal columns, we have that

$$\|M\|_* \ \geq \ \langle M, AB' \rangle$$

2. For sub-gradient descent, we have shown that we have error $O(1/\sqrt{k})$ after k iterations, *when we take a fixed step size.* We can also use a step size that is decaying in the iteration. While using step size $t_k = 1/k$ is guaranteed to converge, it is not the optimal choice. In fact, there are simple functions where sub gradient descent with this sequence of step sizes might take exponential time to converge. Find such a function, and show that we need an exponential number of steps for ϵ-suboptimality. (You can find a simple 1-dimensional function, $f : [0,1] \to \mathbb{R}$.)

3. (?) Re-do the computation we did in class, showing that if we use the mirror function

$$\Phi(\mathbf{x}) = \frac{1}{2}\|x\|_2^2,$$

then the Mirror Descent update for:

$$\begin{aligned} \min_{\mathbf{x}} : \quad & f(\mathbf{x}) \\ \text{s.t.} : \quad & \mathbf{x} \in \mathfrak{X}, \end{aligned}$$

is exactly projected subgradient descent.

4. Here you will do some work that helps compute the Mirror Descent update you need for the computational problem above. Mirror Descent is only computationally useful if we can easily solve the problem:

$$\min_{u\in\mathcal{X}} : \ \langle z, u\rangle + \Phi(u).$$

In this problem, you will show that when $\mathcal{X}$ is the simplex, i.e., $\mathcal{X} = \Delta_n$, then this problem is indeed easy.

   (a) Consider the optimization problem:

$$\begin{aligned} \min : \quad & \langle z, u\rangle + \Phi(u) \\ \text{s.t.} : \quad & \sum_i u_i = 1. \end{aligned}$$

   (Note that the constraints $\{u_i \geq 0\}$ are implicitly included as they are part of domΦ.) Write the Lagrangian for the problem. The variables will be u and a single variable λ for the single constraint. Write the KKT conditions for the problem.

   (b) Using the KKT conditions, derive a closed form expression for u as a function of z.

   (c) Now go back to the Mirror Descent update and write explicitly the Mirror Descent update using your work above.