Homework 6

Convex Optimization 10-725

Due Friday December 6 at 11:59pm

Submit your work as a single PDF on Gradescope. Make sure to prepare your solution to each problem on a separate page. (Gradescope will ask you select the pages which contain the solution to each problem.)

Note that the programming questions in this assignment, Q1(d) and Q3, will be **peer-graded**. Instructions to follow about how this will be done.

Total: 75 points + 25 Bonus

1 Frank-Wolfe and trace norms (30 points + 20 bonus points)

In this problem, we'll investigate the effectiveness of the Frank-Wolfe method for problems involving trace norm regularization.

(a, 3 pts) Show that if u, v are such that $\|u\|_2, \|v\|_2 \leq 1$ and $u^T X v = \sigma_{\max}(X)$, where $\sigma_{\max}(X)$ is the largest singular value of X, then

$$uv^T \in \partial \|X\|_{\text{op}}.$$

(b, 8 pts) Let

$$C = \{X : \|X\|_{\text{tr}} \leq t\},$$

Let $Y \notin C$, and let $Y = U\Sigma V^T$ be an SVD of Y, with (say) $U \in \mathbb{R}^{m\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, $V \in \mathbb{R}^{n\times r}$, and $r = \text{rank}(Y)$. Write $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$. Show that the projection of Y onto C with respect to the Frobenius norm, $P_C(Y) = \text{argmin}_{X\in C} \ \|Y - X\|_F$, is given by

$$P_C(Y) = U\Sigma_t V^T, \quad \text{where } \Sigma_t = \text{diag}\big((\sigma_1 - \theta_t)_+, \ldots, (\sigma_r - \theta_t)_+\big),$$
$$\text{and } \theta_t \geq 0 \text{ is such that } \quad \sum_{i=1}^{r} (\sigma_i - \theta_t)_+ = t.$$

Hint: argue that if U, V have orthonormal columns and are of appropriate dimensions, then $\|UX\|_F = \|X\|_F$ and $\|XV^T\|_F = \|X\|_F$.

(c, 3 pts) Consider the (constrained) matrix completion problem

$$\min_B \sum_{(i,j)\in\Omega} (Y_{ij} - B_{ij})^2 \ \text{ subject to } \ \|B\|_{\text{tr}} \leq t.$$

Using the results from parts (a) and (b), write out in explicit form the iterations of the Frank-Wolfe method and projected gradient descent for this problem. Compare and discuss the computational cost of an iteration of each.

(d, 16 pts) Conduct a suite of experiments to elucidate this difference in iteration costs, starting with the standard step sizes choices for each ($t_k = 2/(k+1)$ and $t_k = 1/L$ for projected gradient). Further compare the differences in convergence speed, and therefore ultimately any differences in total computational cost (running time comparison to reach solutions of comparable accuracy). Investigate the use of backtracking in Frank-Wolfe, and other interesting variants of either algorithm (away steps, acceleration, etc.).

Tip: you should try to implement the iterations for each of Frank-Wolfe and projected gradient as efficiently as possible. You *should not simply compute a full SVD* in each case, and then just read off the quantities you need. This will obviously be the dominant computation in each iteration and would essentially equate the practical computation costs between the two methods, thus defeating the purpose of the comparison. You should be able to get away with a *truncated SVD* (only compute the top k singular values/vectors) for each method. (A truncated SVD should be available in both R and Python.) Explain clearly what your implementations are doing.

Some general tips: be completely explicit about all of your experimental design choices; figures are generally easier to read than tables; it is generally a good idea to aggregate results over multiple simulation instances. You will be graded on the following criteria (3 points each):

- sanity check: have Frank-Wolfe and proximal gradient been demonstrated to be reaching the same solutions?
- depth check: are Frank-Wolfe and projected gradient implemented in a reasonably efficient way (truncated SVD or something of the like)?
- breadth check: is there at least more than one experimental setup considered, and more than just the standard step sizes considered (other algorithm variants considered)?
- in general, are the algorithm implementations and experimental setup clearly explained?
- in general, are the results clearly explained and the conclusions justified?

As usual, append all code in an appendix (1 point for readable/organized code).

Bonus (10 pts): Additionally investigate the problem

$$\min_{U,V} \sum_{(i,j)\in\Omega} (Y_{ij} - [UV^T]_{ij})^2 + \lambda(\|U\|_F^2 + \|V\|_F^2),$$

which for wide enough U, V, is equivalent to the trace norm regularized matrix completion problem written above. Implement a coordinate descent algorithm (alternate minimization over U and V), and include it in your discussion and comparisons of Frank-Wolfe and proximal gradient.

Bonus bonus (10 pts): Show that the matrix completion problem is a semidefinite program (SDP), then derive and implement an interior point method for the SDP formulation, and include it in your comparisons.

2 Screening rules for SVMs (26 points + 5 bonus points)

As we've seen, the KKT conditions can be an extremely useful tool. In machine learning, a series of papers have emerged that use the KKT conditions to derive what are called *screening rules*, originally developed in the context of ℓ_1 regularization problems.[1] These are analytic (closed-form) rules that we can apply to any given data set $(x_i, y_i) \in \mathbb{R}^p \times \mathcal{Y}$, $i = 1, \ldots, n$, to determine *a priori*

[1] It all started with El Ghaoui et al. (2010), http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-126.pdf.

that certain dimensions of the feature space $\mathbb{R}^p$ would not contribute to (say) the lasso or logistic lasso solution, and thus these could be "safely" eliminated before solving (say) the lasso or logistic lasso optimization problem. The rules are usually based on manipulation of the KKT conditions, and typically, properties the solution to the optimization at hand at a "nearby" tuning parameter value.

Screening rules have also been developed for support vector machine (SVMs). In this problem, we'll follow one of the early analyses[2] and look at a constrained version of SVMs: given $y_i \in \{-1, 1\}$ and $x_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, we solve

$$\min_w \ \frac{1}{2}\|w\|_2^2 \ \text{ subject to } \ \sum_{i=1}^n [1 - y_i f_w(x_i)]_+ \leq s, \tag{1}$$

where $f_w(x) = w^T x$. We write $w^*(s)$ for the unique solution in (1). For convenience, we abbreviate $f_{w^*(s)}(x)$ by $f^*(x|s)$. We also abbreviate $J(w) = (1/2)\|w\|_2^2$ and $H(w) = \sum_{i=1}^n [1 - y_i f_w(x_i)]_+$.

(a, 5 pts) Prove that problem (1) and

$$\min_w \ \frac{1}{2}\|w\|_2^2 \ \text{ subject to } \ \sum_{i \notin \mathcal{R}} [1 - y_i f_w(x_i)]_+ \leq s,$$

have the same solution, where $\mathcal{R} = \{i : y_i f^*(x_i|s) > 1\}$. In other words, show that the instances $i \in \mathcal{R}$ do not affect the solution of (1), and can hence be safely discarded. Hint: look at the KKT conditions on all n points, and on only on the points in $\mathcal{R}$.

(b, 2 pts) Fix any $s_a > s_b$. Show that $s \in [s_b, s_a] \implies J(w^*(s)) \leq J(w^*(s_b))$.

(c, 4 pts) Show that $s \in [s_b, s_a] \implies w^*(s_a)^T (w^*(s) - w^*(s_a)) \geq 0$. Hint: it helps to consider the KKT conditions for (1), consider subgradients of $H(w)$, and primal feasibility of $w^*(s)$ for (1) when the tuning parameter is s_a.

(d, 3 pts) Show that we may safely discard a point i from the optimization in problem (1) with tuning parameter $s \in [s_b, s_a]$ if $g_{[s_b,s_a]}(i) > 1$ where:

$$g_{[s_b,s_a]}(i) = \min_{w \in \Theta_{[s_b,s_a]}} \ y_i f_w(x_i),$$

and $\Theta_{[s_b,s_a]} = \{w : J(w) \leq J(w^*(s_b)) \text{ and } w^*(s_a)^T (w - w^*(s_a)) \geq 0\}$.

(e, 3 pts) We now work to reduce the screening rule of $g_{[s_b,s_a]}(i) > 1$ to an analytical formula. Let $\gamma_b = J(w_b^*)$ and $\gamma_a = J(w_a^*)$. Write out the Lagrangian for the problem (2) with Lagrange multipliers of μ and ν for constraints $J(w) \leq \gamma_b$ and $w^*(s_a)^T (w - w^*(s_a)) \geq 0$ respectively.

(f, 3 pts) Write out the KKT conditions for the problem (2). Use these conditions to get an expression for solution to this problem (call it) z in terms of the optimal dual variables μ, ν, and furthermore, an expression for $y_i z^T x_i$ in terms of μ, ν and $f^*(x_i|s_a)$.

(g, 4 pts) Show that:

$$g_{[s_b,s_a]}(i) = \begin{cases} -\sqrt{2\gamma_b}\|x_i\| & \text{if } -\frac{y_i f^*(x_i|s_a)}{\|x_i\|} \geq \frac{\sqrt{2}\gamma_a}{\sqrt{\gamma_b}} \\ y_i f^*(x_i|s_a) - \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}\left(2\gamma_a\|x_i\|_2^2 - f^*(x_i|s_a)^2\right)} & \text{otherwise.} \end{cases}$$

[2] Ogawa et al. (2013), http://jmlr.csail.mit.edu/proceedings/papers/v28/ogawa13b.pdf; you may read this paper if it helps, but you must write out arguments to all parts of this problem in your own words.

Hint: consider the two complementary slackness conditions from part (f), and substitute z for w. Then use the sign of $-\frac{y_i f^*(x_i|s_a)}{\|x_i\|} - \frac{\sqrt{2}\gamma_a}{\sqrt{\gamma_b}}$ to guide whether $\nu = 0$. (Discuss the cases when $\nu = 0$ and $\nu \neq 0$. When $\nu \neq 0$, you can use the two equations from complementary slackness to solve μ and ν.)

(h, 2 pts) Further simplify the screening rule of $g_{[s_b,s_a]}(i) > 1$ to:

$$y_i f^*(x_i|s_a) > 1 \text{ and } y_i f^*(x_i|s_a) - \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}\left(2\gamma_a\|x_i\|_2^2 - f^*(x_i|s_a)^2\right)} > 1.$$

Bonus (5 pts): Empirically validate the correctness of this screening rule by setting up your own empirical example with the constrained SVM problem (1).

3 Distributed ADMM or SGD and variants (19 points)

Design a suite of experiments implementing *either* distributed ADMM *or* SGD and variants (variance reduction, momentum, acceleration, adaptive step sizes), for an optimization problem of interest to you. Compare implementation choices/variants against each other. For example, with ADMM, you could consider solving a large-scale lasso problem, split the data into subsets over samples/features/both, and compare the convergence of distributed ADMM as the number of subsets varies, for different choices of the augmented Lagrangian parameter, for different levels of regularization (sparse versus dense solutions), for different levels of accuracy in solving the inherent linear systems (using direct versus indirect methods), etc. Bonus points for actually doing this in a bonafide distributed fashion, so that communication and synchronization costs become real issues. And with SGD, for example, you could consider solving a large-scale logistic regression problem, and compare the convergence of SGD and variants as you vary the batch size, add ridge regularization, make the data more or less noisy (linearly separable being the noiseless case), make the problem more or less overparametrized, etc. In short, in both cases, you should come up with your own questions and answer them through empirical analysis. Your solution here does not have to be excessively long (no longer than your answer to the Q1(d)), and should be as self-contained as possible: it should be easily readable and understandable by a student of this class.

To be clear, you will you need to implement the algorithms of your choice here instead of simply calling built-in libraries. And at a minimum you will need to compare the running times of your algorithms acrossthe different variants/scenarios/implementations choices/etc. The same general tips given in Q1(d) apply to this problem: be completely explicit about your experimental design choices (this is especially true here, as you are crafting your own questions and thus your own empirical directions); figures are generally easier to read than tables; it is generally a good idea to aggregate results over multiple simulation instances. You will be graded on the following criteria (3 points each):

- setup check: are the questions/directions of interest clearly defined, and the empirical setup a reasonable way to seek answers?
- sanity check: have the algorithms been demonstrated to be convergent (checked against CVX, or each other)?
- depth check: have the algorithms been implemented in a nontrivial way, or is the simulation nontrivial to implement?
- breadth check: is there nontrivial breadth in the design of experimental conditions and/or variants of the algorithms been explored?

- in general, are the algorithm implementations and experimental setup clearly explained?
- in general, are the results clearly explained and the conclusions justified?

As usual, append all code in an appendix (1 point for readable/organized code).