Homework 5

Convex Optimization 10-725/36-725

Due Tuesday November 22 at 5:30pm
submitted to Christoph Dann in Gates 8013
(Remember to a submit separate writeup for each problem, with your name at the top)

Out of the 4 questions, you are supposed to choose any 3 to solve. You may hand in a solution to all four questions for practice and receive feedback on all questions but only the scores of 3 questions count towards your grade. If you do decide to hand in all four problems, please indicate at the top of each question whether that question should count towards your grade. If you give no such indication, the *best score will be discarded (= worst 3 out of 4)*!

Total: 75 points
v1.3

1 Derivation of DFP update (25 points) [Christoph]

(a, 5pts) Assume $y, s \in \mathbb{R}^n$ are non-zero vectors and $B \in \mathbb{R}^{n\times n}$ is a symmetric matrix. Show that the solution to the Frobenius norm minimization problem

$$\begin{aligned} \min_{B^+} \quad & \|B^+ - B\|_F^2 \\ \text{subject to} \quad & (B^+)^T = B^+ \\ & B^+ s = y \end{aligned}$$

is

$$B^+ = B + \frac{(y-Bs)s^T}{s^Ts} + \frac{s(y-Bs)^T}{s^Ts} - \frac{(y-Bs)^Ts}{(s^Ts)^2} ss^T.$$

Hint: show that the KKT conditions hold for a suitable choice of multipliers for the contraints. Use a low-rank matrix for the first one.

(b, 5pts) Show via a counterexample that B^+ may not be positive definite even if B is symmetric positive definite and $y^Ts > 0$.

Hint: observe that B^+ can be written as

$$B^+ = \left(I - \frac{ss^\top}{s^\top s}\right) B \left(I - \frac{ss^\top}{s^\top s}\right) + \frac{ys^\top}{s^\top s} + \frac{sy^\top}{s^\top s} - \frac{y^\top s}{(s^\top s)^2} ss^\top.$$

(c, 5pts) Assume $y, s \in \mathbb{R}^n$ are such that $y^Ts > 0$. Prove that there exists a symmetric and positive definite matrix $M \in \mathbb{R}^{n\times n}$ such that

$$Ms = y.$$

In particular, there exists a non-singular matrix $W \in \mathbb{R}^{n\times n}$ such that $WW^Ts = y$.

Hint: use a matrix M that is the sum of the identity and low-rank matrices.

(d, 5pts) Assume $y, s \in \mathbb{R}^n$ are such that $y^Ts > 0$ and $B \in \mathbb{R}^{n\times n}$ is a symmetric matrix. Let $W \in \mathbb{R}^{n\times n}$ be a non-singular matrix such that $WW^Ts = y$. Show that the solution to the weighted Frobenius norm minimization problem

$$\begin{array}{rl} \min\limits_{B^+} & \|W^{-1}(B^+ - B)W^{-T}\|_F^2 \\ \text{subject to} & (B^+)^T = B^+ \\ & B^+s = y \end{array}$$

is the DFP update

$$B^+ = B + \frac{(y-Bs)y^T}{y^Ts} + \frac{y(y-Bs)^T}{y^Ts} - \frac{(y-Bs)^Ts}{(y^Ts)^2}yy^T = \left(I - \frac{ys^T}{y^Ts}\right)B\left(I - \frac{sy^T}{y^Ts}\right) + \frac{yy^T}{y^Ts}.$$

Hint: use part (a) and a suitable change of variables.

(e, 5pts) Show that if B in the previous question is positive definite then so is B^+.

2 Quasi-Newton Methods in Practice (25 points) [Mariya & Christoph]

(a, 7pts) In this part, we'll use a quasi-Newton line-search algorithm to minimize the quadratic function:

$$f(x) = \frac{1}{2}x^TQx - b^Tx,$$

where $Q \in \mathbb{R}^{n\times n}$ is symmetric positive definite. The quasi-Newton line-search algorithm can be stated as:

```
1 Initialize $x^0 \in \mathbb{R}^n, k = 0, B_0 = I$;
2 while $\|\nabla f(x^k)\| > \epsilon$) and $(k \le k_{max})$ do
3     $p^k := -(B^k)^{-1}\nabla f(x^k)$;
4     $s^k := t_k p^k$, where $t_k = \text{argmin}\{f(x^k + tp^k) : t \ge 0\}$;
5     $x^{k+1} := x^k + s^k$;
6     Update $B^{k+1}$ using $B^k$, $s^k$, and $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$;
7     $k := k + 1$;
8 end
```

i. Implement this algorithm for both the DFP and BFGS updates (you may want to update $H_k = B_k^{-1}$ instead of B_k). Use $\epsilon = 10^{-8}$ and $k_{max} = 100$ quasi-Newton main iterations. Initialize x_0 as a vector of n normally distributed random numbers. Choose $b = 0_n$ and construct a random positive definite Q according to the following steps:

1. Construct a positive definite diagonal matrix D with appropriate entries so that its condition number $\kappa(D) = 10$.
2. Construct a random orthogonal matrix P.
3. Set $Q = P^TDP$.

ii. Evaluate your implementation for both updates using the following:
   1. Three different random matrices $Q \in \mathbb{R}^{2\times 2}$.
   2. Three different random matrices $Q \in \mathbb{R}^{3\times 3}$.
   3. Three different random matrices $Q \in \mathbb{R}^{5\times 5}$.

   For each random Q, report the normalized (divided by the number of elements) Frobenius norm of the difference between the last B_k (or H_k^{-1}) and Q. Report the final objective value and the number of iterations needed to achieve that value.

iii. Do the experiments above suggest a general convergence property for the iterates in the special case of convex quadratic functions with exact line search? Can you propose a concrete conjecture? (Just state a one sentence conjecture. No need to prove it.)

(b, 8pts) In this part, we'll implement a quasi-Newton backtracking line-search algorithm to minimize the Rosenbrock function:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

This algorithm is very similar to the one above, but instead of exact line-search to determine the step-size t_k, we use backtracking. For backtracking, we use the following condition to determine the step-size at each iteration k of the quasi-Newton method:

```
$t_k = 1$;
**while** $f(x^{(k)} + t_k p^{(k)}) > f(x^{(k)}) + 0.0001 t_k \nabla f(x^{(k)})^T p^{(k)}$ **do**
    $t_k := 0.9 t_k$
**end**
```

i. Implement the quasi-Newton backtracking line-search algorithm with 4 different updates for B_k:
   - $B^k = I$ for all k (i.e., steepest descent direction).
   - DFP update.
   - BFGS update.
   - $B^k = \nabla^2 f(x^{(k)})$ (i.e., Newton direction).

   Use $x^{(0)} = \begin{bmatrix} -1.2 \\ 1 \end{bmatrix}$ as the initial point. In the second and third case take $B^0 = H^0 = I$. Limit the number of main quasi-Newton iterations to 100.

ii. Plot $f^{(k)}$ versus k on a semi-log scale (i.e. where the y-axis is in log scale), where $f^{(k)}$ denotes the objective value obtained by the quasi-Newton method with each of the 4 updates at each quasi-Newton iteration k. Plot in the same graph and include a legend. Now, we would like to visualize the solution paths (i.e. x^k for each iteration k) of the different updates. First, plot the 2D contours of the Rosenbrock function with sufficient number of level sets. In MATLAB, such a plot can be obtained in the following way:

```
[X,Y] = meshgrid(-2:0.05:2);
Z = evalRosenbrock([X;Y]);
contour(X,Y,Z,50)
```

   where evalRosenbrock is a user-created function that evaluates the Rosenbrock function. Next, plot the solution paths for the 4 updates on top of the Rosenbrock contour. Plotting without interpolating between different values works best for visualization. Plot the final value of the iterate x^k in larger font so that it is easily visible. Include a legend.

iii. Based on the resulting plots, discuss the performance of each of the updates in minimizing the Rosenbrock function.

(c, 10pts) In this question, we will implement the limited-memory version of BFGS and apply it to ℓ_2-penalized logistic regression.

i. Implement the two-loop recursive version of LBFGS

- As line search use a basic Armijo rule of the form with parameters c_1 and η.

```
1 $t_k = 1$;
2 while $f(x^{(k)} + t_k p^{(k)}) > f(x^{(k)}) + c_1 t_k \nabla f(x^{(k)})^T p^{(k)}$ do
3     $t_k := \eta t_k$
4 end
```

- For computing the step direction use the two-loop recursive matrix-vector multiplication from slide 23 of the *Quasi-Newton Lecture slides.* Use the suggested choice $H^{0,k}$ on this slide. Make sure your algorithm has run- and memory-complexity $O(mn)$ where m is the length of the history considered by LBFGS.
- Stop the algorithm once at least one of the following three conditions are satisfied:

  Gradient condition: The gradient satisfies for each component $i \in \{1, \dots n\}$ the condition

$$\frac{\left|\nabla f(x^{(k)})_i\right| \max\{|x_i^{(k)}|, 1\}}{\max\{f(x^{(k)}), 1\}} < \epsilon_g \tag{1}$$

  where $\epsilon_g > 0$ is an algorithm parameter.

  Step condition: The difference in iterates $s^{(k)} = x^{(k+1)} - x^{(k)}$ satisfies for each component $i \in \{1, \dots n\}$ the condition

$$\frac{\left|s_i^{(k)}\right|}{\max\{\left|x_i^{(k)}\right|, 1\}} < 4\epsilon_f \tag{2}$$

  where ϵ_f is machine precision of double floating point numbers.

  Iteration limit: The number of iterations is at least $k \geq k_{\max}$ where $k_{\max}$ is a parameter of the algorithm.

- Try to use as few function and gradient evaluations of the objective function in your code as possible.

ii. Apply your LBFGS implementation to the Rosenbrock function from part (b) with $c_1 = 10^{-4}, \eta = 0.9, k_{\max} = 10^4$ and $\epsilon_g = 10^{-8}$. Use the same initial $x^{(0)}$ as in part (b).

Report (1) the number of iterations does your algorithm needs, (2) its runtime in seconds, (3) how often the Rosenbrock function evaluated, (4) how often its gradient is evaluated and (5) the final objective value for each of the following settings of $m \in \{3, 4, 5, 6, 7, 8, 9, 10, 20\}$.

How do the number of function and gradient evaluations compare against BFGS from part (b)?

iii. We will now use our LBFGS implementation to find a classifier for the age prediction task from Problem 4 of Homework 2. Instead of ℓ_1 logistic regression, we now want to optimize the ℓ_2-penalized logistic regression loss

$$f(\beta) = -y^\top \tilde{X}\beta + \sum_{i=1}^{n} \log(1 + \exp\{(\tilde{X}\beta)_i\}) + \lambda\|\beta\|_2^2 \tag{3}$$

where $\tilde{X} \in \mathbb{R}^{v\times n}$ are the movie ratings $X \in \mathbb{R}^{v\times(n-1)}$ with an additional column appended with entries 1, $y \in \{0,1\}^v$ are the age classes and β are the logistic regression parameters we want to optimize. Load X and y from the training set in `Q4c_movies.zip` available on the course webpage and use your LBFGS implementation with parameters $c_1 = 10^{-4}, \eta = 0.9, k_{\max} = 10^4, m = 7, \epsilon_g = 10^{-8}$ to find the minimizer of $f(\beta)$ with $\lambda = 100$. Use $\beta^{(0)} = \mathbf{0}$ as initial iterate.

Report the final objective value, the number of iterations, the number of objective function evaluations, the number of gradient evaluations as well as the total runtime of the optimization procedure in seconds.

iv. Use the solution obtained in the previous part to make predictions on the test set, available in `moviesTest.mat` in `Q4c_movies.zip`. What is the classification error?

v. Apply your BFGS implementation from part (b) to the same logistic regression problem with identical settings. Report the final objective value, the number of iterations, the number of objective function evaluations, the number of gradient evaluations as well as the total runtime of the optimization procedure in seconds. How does LBFGS and BFGS compare on this problem?

Attach all relevant code to the end of this problem.

3 Row Subset Selection for Linear Regression (25 points) [Han]

In this question we will consider the optimal subset selection problem in the setting of linear regression. A typical linear regression model is:

$$y = X\beta + \varepsilon$$

where $X \in \mathbb{R}^{n\times p}$ is the design matrix, $y \in \mathbb{R}^n$ is the response, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ is a vector of i.i.d. Gaussian noise with variance σ^2, and β is a fixed but unknown p-dimensional coefficient vector. Given the design matrix X and the response y, the goal is to estimate the regression model β. In this problem we focus on the low-dimensional setting where $n > p$ and $\text{rank}(X) = p$. An effective estimator of β in the low-dimensional setting is the *ordinary least square* (OLS) estimator, given by:

$$\hat{\beta} = \underset{\beta\in\mathbb{R}^p}{\text{argmin}}\, ||y - X\beta||_2^2 = (X^TX)^{-1}X^Ty$$

A simple calculation shows that the mean squared error associated with the OLS estimator is:

$$\mathbb{E}_\varepsilon||\beta - \hat{\beta}||_2^2 = \sigma^2 \text{tr}\left((X^TX)^{-1}\right).$$

In many cases where $n \gg p$, it is favorable to select a representative subset of rows in X so that the subsampled regression still accurately estimates β. However, for a given k, with $p \le k \le n$, determining the optimal k-subset of rows that minimizes the mean squared error leads to a combinatorial optimization problem, which is computationally intractable to solve exactly. To this end, we propose the following convex relaxation of the k-subset selection problem to be solved:

$$\begin{array}{ll} \text{minimize}_{\pi\in\mathbb{R}^n} & \text{tr}\left((X^T\,\text{diag}(\pi)X)^{-1}\right) \\ \text{subject to} & \pi \ge 0, 1^T\pi = 1 \end{array} \tag{4}$$

where we define the objective function to be ∞ whenever $X^T\,\text{diag}(\pi)X$ is not invertible.

(a, 5pts) Show that (4) is a convex optimization problem. (*Hint*: There are two ways to go about this: (i) show directly that the above objective is convex, or (ii) reformulate the above problem into one that is of "canonical" convex form, i.e., one of the LP, QP, SOCP, SDP, Conic problem classes.)

(b, 5pts) Derive the gradient of the objective function (as a function of π).

(c, 5pts) List an algorithm that you know of from the course that applies to (4), and describe in as much detail as possible the updates. If an update involves a projection operator, then provide a precise description/pseudocode as to how exactly this projection will be computed.

(d, 10pts) Implement your proposed algorithm to solve the optimization problem (4) on the dataset `madelon.txt`, available on the course website. Report your optimal objective function value, and the number of nonzero components in the optimal π^*. To answer this question, you can treat all the values less than 10^{-6} to be 0. (*Hint*: In your implementation, make sure that after each update iteration, the new π is in the feasible region and has at least p nonzero components).

(e, 10 pts, Bonus) Describe any other algorithm(s), inspired by what you learned in the course or otherwise, that you believe to be efficient for solving (4). For full bonus points, implement and run on the data from part (d).

Attach all relevant code to the end of this problem.

4 Quantile regression and ADMM (25pts) [Justin and Alnur]

Quantile regression is a way to estimate the conditional α-quantile, for some $\alpha \in [0,1]$ fixed by the implementer, of some response variable Y given some predictors $X_1, \ldots, X_p$. (Just to remind you: the α-quantile of $Y|X_1, \ldots, X_p$ is given by $\inf_t \{t : \text{Prob}(Y \leq t | X_1, \ldots, X_p) \geq \alpha\}$.) It turns out that we can estimate the α-quantile by solving the following (convex) optimization problem:

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad \textstyle\sum_{i=1}^n \ell_i^{(\alpha)} \left(y - X\beta\right), \tag{5}$$

where $y \in \mathbb{R}^n$ is a response vector, $X \in \mathbb{R}^{n \times p}$ is a data matrix, and the map $\ell^{(\alpha)} : \mathbb{R}^n \to \mathbb{R}^n$ is the *quantile loss*, defined as

$$\ell_i^{(\alpha)}(a) = \max\left\{\alpha a_i, (\alpha - 1)a_i\right\}, \quad i = 1, \ldots, n,$$

for some $a \in \mathbb{R}^n$. (It is not too hard to show why this is the case, but you can just assume it is true for this question.) In Figure 1, we plot some estimated α-quantiles, for a few values of α, obtained by solving problem (5) on some synthetic data.

We often want to estimate multiple quantiles simultaneously, but would like them to not intersect: e.g., in Figure 1, we would like to *avoid* the situation where the 0.1-quantile line (maybe eventually) lies above the 0.9-quantile line, since this doesn't make any sense. To do so, we can instead solve the following (again convex) optimization problem, referred to as the *multiple quantile regression problem with non-crossing constraints*:

$$\begin{array}{ll} \underset{B \in \mathbb{R}^{p \times r}}{\text{minimize}} & \sum_{i=1}^n \sum_{j=1}^r \ell_{ij}^{(\mathcal{A})} \left(y 1^T - XB\right) \\ \text{subject to} & XBD^T \geq 0. \end{array} \tag{6}$$

Here, r is the number of α's (i.e., quantile levels); $\mathcal{A} = \{\alpha_1, \ldots, \alpha_r\}$ is a set of user-defined quantile levels; y, X are as before; 1 is the r-dimensional all-ones vector; and $B \in \mathbb{R}^{p \times r}$ is now a parameter

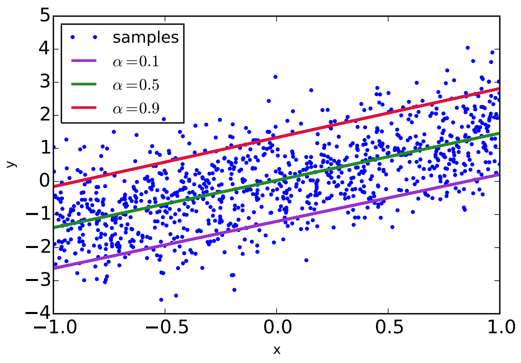

Figure 1: Various quantiles estimated from synthetic data.

matrix, structured as follows:

$$B = \begin{bmatrix} | & & | \\ \beta_1 & \cdots & \beta_r \\ | & & | \end{bmatrix},$$

for $\beta_i \in \mathbb{R}^p,\ i = 1, \ldots, r$. Here, we have also extended the definition of the quantile loss: it is now a map $\ell^{(\mathcal{A})} : \mathbb{R}^{n\times r} \to \mathbb{R}^{n\times r}$, defined as

$$\ell_{ij}^{(\mathcal{A})}(A) = \max\left\{\alpha_j A_{ij}, (\alpha_j - 1)A_{ij}\right\}, \quad i = 1, \ldots, n,\ j = 1, \ldots, r.$$

Lastly, $D \in \mathbb{R}^{(r-1)\times r}$ is the first-order finite differencing operator, given by

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}.$$

Hence, you can convince yourself that the constraints in (6) prevent the estimated quantiles from crossing.

Phew! Now, finally, for the problem.

(a, 5pts) Derive the prox operator for the quantile loss $\ell_{ij}^{(\mathcal{A})}$ (defined above). In other words, write down an expression for $\text{prox}_{\lambda \ell_{ij}^{(\mathcal{A})}}(A)$, where $\lambda > 0$ is a user-defined constant, and $A \in \mathbb{R}^{m\times n}$ is the point at which the prox operator is evaluated.

(b, 5pts) We can put problem (6) into "ADMM form" as follows:

$$\begin{array}{ll} \underset{\substack{B\in\mathbb{R}^{p\times r},\\ Z_1,Z_2\in\mathbb{R}^{n\times r},\\ Z_3\in\mathbb{R}^{n\times(r-1)}}}{\text{minimize}} & \sum_{i=1}^n \sum_{j=1}^r \ell_{ij}^{(\mathcal{A})}(Z_1) + I_+(Z_3) \\ \text{subject to} & Z_1 = y1^T - XB, \quad Z_2 = XB, \quad Z_3 = Z_2 D^T \end{array} \tag{7}$$

where we introduced the variables Z_1, Z_2, Z_3, and $I_+(\cdot)$ is the projection map onto the non-negative orthant.

The augmented Lagrangian associated with problem (7) is then (in "scaled" form)

$$L_\rho(B, Z_1, Z_2, Z_3, U_1, U_2, U_3) = \sum_{i=1}^{n} \sum_{j=1}^{r} \ell_{ij}^{(\mathcal{A})}(Z_1) + I_+(Z_3) + \frac{\rho}{2}\Big(\|y1^T - XB - Z_1 + U_1\|_F^2 + \|XB - Z_2 + U_2\|_F^2 + \|Z_2 D^T - Z_3 + U_3\|_F^2 - \|U_1\|_F^2 - \|U_2\|_F^2 - \|U_3\|_F^2 \Big),$$

where U_1, U_2, U_3 are dual variables.

Write down the ADMM updates for problem (7). You can assume that the quantity $X^T X$ is invertible.

(c, 7 pts) Implement your ADMM algorithm from part (b) for the quantiles $\mathcal{A} = \{0.1, 0.5, 0.9\}$, using synthetic data from hw4q4.zip which contain a predictor matrix `X`, response vector `y`, and in csv files. Note that the predictor matrix in this example data is 1000×2, with the 2nd column being the column of all 1s, hence providing an intercept term in the quantile regression.

Set initial values for all primal and dual variables as zero matrices of appropriate size, use an augmented Lagrangian parameter of $\rho = 1$, and run ADMM for 50 steps. Report your optimal criterion value after 50 steps, and plot the optimal criterion value across the iterations. Recreate the plot similar to Figure 1, plotting data as points and overlaying as lines the *conditional quantile* predictions for the quantiles in $\mathcal{A}$. (Hint: recall that B is a matrix whose r columns are coefficient estimates for each quantile level $\alpha_1, \cdots, \alpha_r$, so that $X_{new}^T B$ at a new data point $X_{\text{new}} = (x_{\text{new}}, 1)$ is the fitted estimates for each quantile in $\mathcal{A}$.)

(d, 8 pts) An alternative (more efficient) way to parameterize problem (6) in "ADMM form" is as follows:

$$\begin{aligned} \underset{\substack{B \in \mathbb{R}^{p \times r}, \\ Z_1, Z_2 \in \mathbb{R}^{n \times r}}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{r} \ell_{ij}^{(\mathcal{A})}(Z_1) + I_+(Z_2 D^T) \\ \text{subject to} \quad & Z_1 = y1^T - XB, \quad Z_2 = XB. \end{aligned} \tag{8}$$

The augmented Lagrangian associated with problem (8) is then

$$L_\rho(B, Z_1, Z_2, U_1, U_2) = \sum_{i=1}^{n} \sum_{j=1}^{r} \ell_{ij}^{(\mathcal{A})}(Z_1) + I_+(Z_2 D^T) + \frac{\rho}{2}\Big(\|y1^T - XB - Z_1 + U_1\|_F^2 + \|XB - Z_2 + U_2\|_F^2 - \|U_1\|_F^2 - \|U_2\|_F^2 \Big).$$

Write down and implement the ADMM updates for problem (8). Explain why the Z_2 update reduces to solving n isotonic regressions. Give the name of an algorithm (by looking around in the literature) for isotonic regression that runs in linear-time (or very nearly linear-time). Most programming languages (Python, Matlab, R) should provide access to an implementation of such an algorithm.

Then, run the ADMM steps for (8) on the data from part (c), and reproduce the same things requested in the part. Compare the optimal criterion curves between the implementation in part (c) and the one from the current implementation—is there a noticeable difference?

Attach all relevant code to the end of this problem.