EE364a: Convex Optimization I J. Duchi
March 13–14, 14–15, or 18–19, 2020

Final Exam

This is a 24 hour take-home final. Please turn it in by uploading to Gradescope no more than 24 hours after you receive it by email. We will *not* be very lenient with upload times: you must upload no more than 24 hours after you receive the final.

You may use any books, notes, or computer programs, but you may not discuss the exam with anyone until 6:30PM March 19, after everyone has finished the exam. The only exception is that you can ask us for clarification, via the course staff email address or private Piazza post. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

Please attach the cover page to the front of your exam. Assemble your solutions in order (problem 1, problem 2, problem 3, ...), starting a new page for each problem. Put everything associated with each problem (*e.g.*, text, code, plots) together; do not attach code or plots for all problems at the end of the final.

We will deduct points from long, needlessly complex solutions, even if they are correct. Our solutions are not long, so if you find that your solution to a problem goes on and on for many pages, you should try to figure out a simpler one. We expect neat, legible exams from everyone, including those enrolled Cr/N.

When a problem involves computation you must give all of the following: a clear discussion and justification of exactly what you did, the source code that produces the result, and the final numerical results or plots.

Files containing problem data can be found in the following location:

`http://www.stanford.edu/~jduchi/MhTDSk/`

Please respect the honor code. Although we allow you to work on homework assignments in small groups, you cannot discuss the final with anyone, at least until everyone has taken it.

All problems have equal weight. Some are (quite) straightforward. Others, not so much.

Be sure you are using the most recent version of CVX, CVXPY, or Convex.jl. Check your email often during the exam, just in case we need to send out an important announcement.

Some problems involve applications. But you do not need to know *anything* about the problem area to solve the problem; the problem statement contains everything you need.

1. *Deconvolution of a known filter.* In a (simplified) imaging system, instead of observing a true image, one observes an image with slight blurring (or other aberrations due to sensor error), and wishes to recover the true image. We let $Z \in \mathbf{R}^{d \times d}$ by the true image, which is unobsered, and $Y \in \mathbf{R}^{d \times d}$ be the observed image. Here, $Y = F * Z$ is the convolution of Z with a known filter F (this is the *point spread function*), with entries

$$Y_{kl} = (F * Z)(k, l) := \sum_{i,j=1}^{d} F_{i,j} Z_{k-i,l-j}.$$

For those indices $(k - i, l - j)$ out of the range $\{1, \ldots, d\}^2$, we define $Z_{k-i,l-j} = 0$. Let $m = d^2$ be the number of measurements we take. If we let $z = \mathbf{vec}(Z)$, that is, the vectorized version of Z, and $y = \mathbf{vec}(Y)$, then there is a matrix $A \in \mathbf{R}^{m \times m}$ such that

$$y = Az.$$

(You do not need to know what the matrix A is or precisely what **vec** does.) Sensor failures at some pixels (k, l) mean that instead of observing $Y_{kl} = (F * Z)(k, l)$ we observe a $Y_{kl} = 0$.

A vectorized image $z \in \mathbf{R}^m$ is represented in an overcomplete basis $B \in \mathbf{R}^{m \times n}$, $n > m$, so there exist vectors $x \in \mathbf{R}^n$ such that $z = Bx$. Given Y with $y = \mathbf{vec}(Y)$, the image deconvolution problem is to find x minimizing an objective $f(x)$ while reconstructing the observed image, *i.e.* satisfying $y = ABx$.

Formulate the following as convex optimization problems:

(a) The deconvolution problem with objective $f(x) = \|x\|_1$.

(b) The deconvolution problem with objective $f(x) = \|x\|_2$.

(c) The deconvolution problem with objective $f(x) = \|x\|_\infty$.

Solve your optimization problems from parts (a), (b), and (c) on the data in `deconvolution_data.*`. In the file we have defined a vector $y \in \mathbf{R}^m$ and filter matrix $A \in \mathbf{R}^{m \times m}$, with zeroed-out entries indicating sensor failures, as well as a basis matrix $B \in \{-1, 0, 1\}^{m \times n}$.

(d) For each of (a)–(c), display the estimated true image $z = Bx^\star$ that is reconstructed from $x^\star$. In addition, display the initial sensed image y. Explain your results in one or two sentences.

Note. You can view an image $Z \in \mathbf{R}^{d \times d}$ from a vector $z \in \mathbf{R}^m$ with $m = d^2$ by reshaping and displaying. A few commands to view $z \in \mathbf{R}^m$ as an image follow.

- In Julia, assuming you are using PyPlot, use
  `imshow(reshape(z, (d, d)), cmap = "gray", interpolation="nearest")`
  If you are using Plots and Images, you can use
  `display(Gray.(reshape(z, (d, d))))`
- In Matlab use
  `imshow(reshape(z, d, d))`
- In Python assuming you are using `Matplotlib.pyplot as plt`, use
  `plt.imshow(np.reshape(z, (d,d)).T, "gray", interpolation="nearest")`

2. *Multi-period liability clearing.* We consider a financial system with n financial entities or agents, such as banks, who make payments to each other over discrete time periods $t = 1, 2, \ldots$. We let $c_t \in \mathbf{R}^n_+$ denote the cash held at the beginning of time period t, where $(c_t)_i$ is the amount held by the ith entity in dollars.

We let $L_t \in \mathbf{R}^{n\times n}_+$ denote the liability between the entities at the beginning of time period t, where $(L_t)_{ij}$ is the amount in dollars that entity i owes entity j. You can assume that $(L_t)_{ii} = 0$, *i.e.*, the entities do not owe anything to themselves. Note that $L_t\mathbf{1}$ is the vector of total liabilities of the entities, *i.e.*, the total amount owed to other entities, and $L_t^T\mathbf{1}$ is the vector of total amount owed to the entities by others, at time period t.

We let $P_t \in \mathbf{R}^{n\times n}_+$ denote the amount paid between each entity during time period t, where $(P_t)_{ij}$ is the amount, in dollars, paid from entity i to entity j. Thus $P_t\mathbf{1}$ is the vector of total cash payments made by the entities to others in period t (*i.e.*, $(P_t\mathbf{1})_i$ is the total payments from entity i to all other entities), and $P_t^T\mathbf{1}$ is the vector of total cash received by the entities from others in period t.

The liabilities and cash follows the dynamics

$$L_{t+1} = L_t - P_t, \quad t = 1, 2, \ldots,$$
$$c_{t+1} = c_t - P_t\mathbf{1} + P_t^T\mathbf{1}, \quad t = 1, 2, \ldots.$$

Each entity cannot pay more than the cash that it has on hand, so we have the constraint

$$P_t\mathbf{1} \preceq c_t, \quad t = 1, 2, \ldots.$$

We are given the initial cash held c_1 and the initial liabilities L_1. You can assume that for each entity, the cash held plus the cash owed to it are at least as much as the amount it owes, *i.e.*, $c_1 - L_1\mathbf{1} + L_1^T\mathbf{1} \succeq 0$.

(a) *Minimum time to clear the liabilities.* Explain how to find the minimum T for which there is a feasible sequence of payments $P_1, \ldots, P_{T-1}$ that results in $L_T = 0$. (Reducing the liabilities to zero is called *clearing* them.) Your method can involve solving a reasonable number of convex problems.

(b) Carry out the method of part (a) on the data given in `clearing_data.*`.

3. *Inverse of log-concave scalar random variable.* Suppose the random variable X defined on $\mathbf{R}_+$ has a log-concave and decreasing density p_X.

(a) *Square root.* Let $Y = \sqrt{X}$. Does Y have a log-concave density? Either show that it does, or give a counter-example, *i.e.*, a specific log-concave decreasing density for X for which the density of Y is not log-concave.

(b) *Square.* Let $Z = X^2$. Does Z have a log-concave density? Either show that it does, or give a counter-example, *i.e.*, a specific log-concave decreasing density for X for which the density of Z is not log-concave.

4. *Meta learning.* In the *meta-learning* or *one-shot learning* problem, one is given m tasks represented by loss functions $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$, and wishes to find a point x from which it is easy for a given algorithm initialized at x to find a minimizer of f_i. (Here the interpretation of *meta-learning* is that it is easy to "learn" from the initialization x.)

We formulate this as follows. For a point $x \in \mathbf{R}^n$, convex $f : \mathbf{R}^n \to \mathbf{R}$, and $\lambda > 0$ the *proximal mapping* is

$$\mathbf{prox}_f^\lambda(x) := \operatorname*{argmin}_y \left\{ f(y) + \frac{\lambda}{2} \|y - x\|_2^2 \right\}.$$

We measure the performance of a point x for task i by its distance to the proximal point update $\mathbf{prox}_{f_i}^\lambda(x)$ and the loss $f_i(\mathbf{prox}_{f_i}^\lambda(x))$ at this updated point. Accordingly, we seek x minimizing

$$h(x) := \frac{\lambda}{2} \sum_{i=1}^m \left\| x - \mathbf{prox}_{f_i}^\lambda(x) \right\|_2^2 + \sum_{i=1}^m f_i \left(\mathbf{prox}_{f_i}^\lambda(x) \right).$$

In this problem, the f_i are closed convex functions.

(a) Formulate the problem of minimizing $h(x)$ as a convex optimization problem. You may introduce new variables.

(b) Suppose the loss for task i is the squared error

$$f_i(x) = \frac{1}{2}(a_i^T x - b_i)^2,$$

where $a_i \in \mathbf{R}^n, b_i \in \mathbf{R}$. Formulate minimizing h as a quadratic program with the single variable $x \in \mathbf{R}^n$. Be as explicit as you can.

5. *Combining partial rankings.* In the rank aggregation problem, we are given several ordered lists of items, and want to construct a single list that reflects the orderings in the given lists. Suppose we have m ordered lists of k indices $i \in \{1, \ldots, n\}$, each of the form

$$\sigma^j = (i_1^j, i_2^j, \ldots, i_k^j), \quad j = 1, \ldots, m.$$

The meaning of the lists is that, according to list j, i_1^j is preferred to i_2^j, is preferred to i_3^j, and so on. (The lists have the same number of items, k, for notational simplicity. Everything works in the more realistic case when the lists have different lengths.)

We will search for a set of *scores* for the items, denoted by $s \in \mathbf{R}^n$. This set of scores induces a ranking of the items, with the item of highest score the first, second highest second, and so on. (If there are repeated entries in s, the ranking is ambiguous.)

We say that a score s is *consistent* with a ranking σ^j if

$$s_{i_1^j} > s_{i_2^j} > \cdots > s_{i_k^j}.$$

(a) *Finding a consistent score.* Explain how to use convex optimization to find a set of scores (and therefore also a ranking) that is consistent with the given lists, assuming there is one. *Note.* No, you cannot solve convex problems with strict inequalities.

(b) Use your method on the data in `ranked_lists_data.*`. Give the ordering you get. (Be sure to include your code.)

(c) *Finding a score consistent with many of the lists.* Suppose there is no consistent score for the set of lists. Suggest a convex optimization problem that is a heuristic for finding a score that is consistent with as many of the given lists as possible. Note that this is not the same as finding a score for which many of the inequalities hold; all $k - 1$ inequalities in a given list must hold for the scores to be consistent with the list. *Note.* We will accept any reasonable solution; there are several we can think of.

(d) Use your method on the data in `ranked_lists_inconsistent_data.*`. Give the ordering you get as well as the number of lists for which your solution is inconsistent (*i.e.*, where at least one of the $k - 1$ pairs in the list is mis-ordered).

6. *Probabilistic centers.* Let $C \subset \mathbf{R}^n$ be a bounded convex set. For a vector $w \in \mathbf{R}^n$, the distance of x to the boundary of C in the direction w is

$$d_C(x, w) := \sup_{y \in C} \left| w^T (y - x) \right| / \|w\|_2 .$$

For a given probability density p on $w \in \mathbf{R}^n$, we let

$$f(x) := \mathbf{E}\, d_C(x, w) = \int d_C(x, w) p(w) dw$$

be the expected distance. (We could easily define this for discrete random variables w as well.) The *p-centers of* C are all points $x^\star \in C$ minimizing $f(x)$, *i.e.*, satisfying $f(x^\star) = \inf_{x \in C} f(x)$.

(a) Let $w_1, \ldots, w_m \in \mathbf{R}^n$ and consider the empirical average

$$f_m(x) = \frac{1}{m} \sum_{i=1}^{m} d_C(x, w_i). \tag{1}$$

Formulate minimizing f_m as a convex optimization problem using the support functions $\sigma_C(w) := \sup_{y \in C} y^T w$.

(b) If C is convex and symmetric (*i.e.* $C = -C$) and p is an arbitrary distribution on w, is $x^\star = \mathbf{0}$ a minimizer of f? If so, why? If not, give a counterexample.

(c) Suppose that the set C is a polyhedron, that is,

$$C = \{x \in \mathbf{R}^n \mid Ax \preceq b\}$$

for some $A \in \mathbf{R}^{k \times n}, b \in \mathbf{R}^k$. Formulate minimizing f_m over C as a linear program.

(d) Suppose that p is the uniform distribution over $\{w \in \mathbf{R}^n \mid \|w\|_2 = 1\}$, where $n = 10$, and let $C = \{x \in \mathbf{R}^n_+ \mid \mathbf{1}^T x \leq 1\}$ be the unit simplex. Repeat the following experiment 20 times for each of the values of $m \in \{10, 20, 40, 80, 160, 320, 640, 1280, 2560\}$.

i. Draw m samples $w_1, \ldots, w_m$, where each w_i is uniform on the sphere, then

ii. Solve problem (1) to get a (random) minimizer $\hat{x}_m = \operatorname{argmin}_{x \in C} f_m(x)$.

Plot the average of the errors $\|\hat{x}_m - \mathbf{1}/n\|_2$ you get in each experiment against m, the sample size. (Be sure to include your code in your solution.)