EE364a: Convex Optimization I S. Boyd
March 11–12 or March 12–13, 2016

Final Exam

This is a 24-hour take-home final. Please turn it in at Bytes Cafe in the Packard building, 24 hours after you pick it up.

You may use any books, notes, or computer programs, but you may not discuss the exam with anyone until March 16, after everyone has taken the exam. The only exception is that you can ask us for clarification, via the course staff email address. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

Please make a copy of your exam, or scan it, before handing it in.

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 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 usual place,

`http://www.stanford.edu/~boyd/cvxbook/cvxbook_additional_exercises/`

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 easy. 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. *Portfolio optimization using multiple risk models.* Let $w \in \mathbf{R}^n$ be a vector of portfolio weights, where negative values correspond to short positions, and the weights are normalized such that $\mathbf{1}^T w = 1$. The expected return of the portfolio is $\mu^T w$, where $\mu \in \mathbf{R}^n$ is the (known) vector of expected asset returns. As usual we measure the risk of the portfolio using the variance of the portfolio return. However, in this problem we do not know the covariance matrix Σ of the asset returns; instead we assume that Σ is one of M (known) covariance matrices $\Sigma^{(k)} \in \mathbf{S}^n_{++}$, $k = 1, \ldots, M$. We can think of the $\Sigma^{(k)}$ as representing M different risk models, associated with M different market regimes (say). For a weight vector w, there are M different possible values of the risk: $w^T \Sigma^{(k)} w$, $k = 1, \ldots, M$. The worst-case risk, across the different models, is given by $\max_{k=1,\ldots,M} w^T \Sigma^{(k)} w$. (This is the same as the worst-case risk over all covariance matrices in the convex hull of $\Sigma^{(1)}, \ldots, \Sigma^{(M)}$.)

   We will choose the portfolio weights in order to maximize the expected return, adjusted by the worst-case risk, *i.e.*, as the solution $w^\star$ of the problem

$$\begin{array}{ll} \text{maximize} & \mu^T w - \gamma \max_{k=1,\ldots,M} w^T \Sigma^{(k)} w \\ \text{subject to} & \mathbf{1}^T w = 1, \end{array}$$

   with variable w, where $\gamma > 0$ is a given risk-aversion parameter. We call this the mean-worst-case-risk portfolio problem.

   (a) Show that there exist $\gamma_1, \ldots, \gamma_M \geq 0$ such that $\sum_{k=1}^M \gamma_k = \gamma$ and the solution $w^\star$ of the mean-worst-case-risk portfolio problem is also the solution of the problem

$$\begin{array}{ll} \text{maximize} & \mu^T w - \sum_{k=1}^M \gamma_k w^T \Sigma^{(k)} w \\ \text{subject to} & \mathbf{1}^T w = 1, \end{array}$$

   with variable w.

   Remark. The result above has a beautiful interpretation: We can think of the γ_k as allocating our total risk aversion γ in the mean-worst-case-risk portfolio problem across the M different regimes.

   Hint. The values γ_k are not easy to find: you have to solve the mean-worst-case-risk problem to get them. Thus, this result does not help us solve the mean-worst-case-risk problem; it simply gives a nice interpretation of its solution.

   (b) Find the optimal portfolio weights for the problem instance with data given in `multi_risk_portfolio_data.*`. Report the weights and the values of γ_k, $k = 1, \ldots, M$. Give the M possible values of the risk associated with your weights, and the worst-case risk.

2. *Minimum possible maximum correlation.* Let Z be a random variable taking values in $\mathbf{R}^n$, and let $\Sigma \in \mathbf{S}^n_{++}$ be its covariance matrix. We do not know Σ, but we do know the variance of m linear functions of Z. Specifically, we are given nonzero vectors $a_1, \ldots, a_m \in \mathbf{R}^n$ and $\sigma_1, \ldots, \sigma_m > 0$ for which

$$\mathbf{var}(a_i^T Z) = \sigma_i^2, \quad i = 1, \ldots, m.$$

For $i \neq j$ the correlation of Z_i and Z_j is defined to be

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}.$$

Let $\rho^{\max} = \max_{i \neq j} |\rho_{ij}|$ be the maximum (absolute value) of the correlation among entries of Z. If $\rho^{\max}$ is large, then at least two components of Z are highly correlated (or anticorrelated).

(a) Explain how to find the smallest value of $\rho^{\max}$ that is consistent with the given information, using convex or quasiconvex optimization. If your formulation involves a change of variables or other transformation, justify it.

(b) The file `correlation_bounds_data.*` contains $\sigma_1, \ldots, \sigma_m$ and the matrix A with columns $a_1, \ldots, a_m$. Find the minimum value of $\rho^{\max}$ that is consistent with this data. Report your minimum value of $\rho^{\max}$, and give a corresponding covariance matrix Σ that achieves this value. You can report the minimum value of $\rho^{\max}$ to an accuracy of 0.01.

3. *Bandlimited signal recovery from zero-crossings.* Let $y \in \mathbf{R}^n$ denote a *bandlimited* signal, which means that it can be expressed as a linear combination of sinusoids with frequencies in a band:

$$y_t = \sum_{j=1}^{B} a_j \cos\left(\frac{2\pi}{n}(f_{\min} + j - 1)t\right) + b_j \sin\left(\frac{2\pi}{n}(f_{\min} + j - 1)t\right), \quad t = 1, \ldots, n,$$

where $f_{\min}$ is lowest frequency in the band, B is the bandwidth, and $a, b \in \mathbf{R}^B$ are the cosine and sine coefficients, respectively. We are given $f_{\min}$ and B, but not the coefficients a, b or the signal y.

We do not know y, but we are given its sign $s = \text{sign}(y)$, where $s_t = 1$ if $y_t \geq 0$ and $s_t = -1$ if $y_t < 0$. (Up to a change of overall sign, this is the same as knowing the 'zero-crossings' of the signal, *i.e.*, when it changes sign. Hence the name of this problem.)

We seek an estimate $\hat{y}$ of y that is consistent with the bandlimited assumption and the given signs. Of course we cannot distinguish y and αy, where $\alpha > 0$, since both of these signals have the same sign pattern. Thus, we can only estimate y up to a positive scale factor. To normalize $\hat{y}$, we will require that $\|\hat{y}\|_1 = n$, *i.e.*, the average value of $|y_i|$ is one. Among all $\hat{y}$ that are consistent with the bandlimited assumption, the given signs, and the normalization, we choose the one that minimizes $\|\hat{y}\|_2$.

(a) Show how to find $\hat{y}$ using convex or quasiconvex optimization.

(b) Apply your method to the problem instance with data in `zero_crossings_data.*`. The data files also include the true signal y (which of course you cannot use to find $\hat{y}$). Plot $\hat{y}$ and y, and report the relative recovery error, $\|y - \hat{y}\|_2 / \|y\|_2$. Give one short sentence commenting on the quality of the recovery.

4. *Satisfying a minimum number of constraints.* Consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \text{ holds for at least } k \text{ values of } i, \end{array}$$

with variable $x \in \mathbf{R}^n$, where the objective f_0 and the constraint functions f_i, $i = 1, \ldots, m$ (with $m \geq k$), are convex. Here we require that only k of the constraints hold, instead of all m of them. In general this is a hard combinatorial problem; the brute force solution is to solve all $\binom{m}{k}$ convex problems obtained by choosing subsets of k constraints to impose, and selecting one with smallest objective value.

In this problem we explore a convex restriction that can be an effective heuristic for the problem.

(a) Suppose $\lambda > 0$. Show that the constraint

$$\sum_{i=1}^{m} (1 + \lambda f_i(x))_+ \leq m - k$$

guarantees that $f_i(x) \leq 0$ holds for at least k values of i. ($(u)_+$ means $\max\{u, 0\}$.)
Hint. For each $u \in \mathbf{R}$, $(1 + \lambda u)_+ \geq 1(u > 0)$, where $1(u > 0) = 1$ for $u > 0$, and $1(u > 0) = 0$ for $u \leq 0$.

(b) Consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & \sum_{i=1}^{m} (1 + \lambda f_i(x))_+ \leq m - k \\ & \lambda > 0, \end{array}$$

with variables x and λ. This is a restriction of the original problem: If (x, λ) are feasible for it, then x is feasible for the original problem. Show how to solve this problem using convex optimization. (This may involve a change of variables.)

(c) Apply the method of part (b) to the problem instance

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \leq b_i \text{ holds for at least } k \text{ values of } i, \end{array}$$

with $m = 70$, $k = 58$, and $n = 12$. The vectors b, c and the matrix A with rows a_i^T are given in the file `satisfy_some_constraints_data.*`.

Report the optimal value of λ, the objective value, and the actual number of constraints that are satisfied (which should be larger than or equal to k). To determine if a constraint is satisfied, you can use the tolerance $a_i^T x - b_i \leq \epsilon^{\text{feas}}$, with $\epsilon^{\text{feas}} = 10^{-5}$.

A standard trick is to take this tentative solution, choose the k constraints with the smallest values of $f_i(x)$, and then minimize $f_0(x)$ subject to these k constraints (*i.e.*, ignoring the other $m - k$ constraints). This improves the objective value over the one found using the restriction. Carry this out for the problem instance, and report the objective value obtained.

5. *Ideal preference point.* A set of K choices for a decision maker is parametrized by a set of vectors $c^{(1)}, \ldots, c^{(K)} \in \mathbf{R}^n$. We will assume that the entries c_i of each choice are normalized to lie in the range $[0, 1]$. The *ideal preference point model* posits that there is an ideal choice vector c^{ideal} with entries in the range $[0, 1]$; when the decision maker is asked to choose between two candidate choices c and $\tilde{c}$, she will choose the one that is closest (in Euclidean norm) to her ideal point. Now suppose that the decision maker has chosen between all $K(K-1)/2$ pairs of given choices $c^{(1)}, \ldots, c^{(K)}$. The decisions are represented by a list of pairs of integers, where the pair (i, j) means that $c^{(i)}$ is chosen when given the choices $c^{(i)}, c^{(j)}$. You are given these vectors and the associated choices.

   (a) How would you determine if the decision maker's choices are consistent with the ideal preference point model?

   (b) Assuming they are consistent, how would you determine the bounding box of ideal choice vectors consistent with her decisions? (That is, how would you find the minimum and maximum values of c_i^{ideal}, for c^{ideal} consistent with being the ideal preference point.)

   (c) Carry out the method of part (b) using the data given in `ideal_pref_point_data.*`. These files give the points $c^{(1)}, \ldots, c^{(K)}$ and the choices, and include the code for plotting the results. Report the width and the height of the bounding box and include your plot.

6. *Matrix equilibration.* We say that a matrix is ℓ_p *equilibrated* if each of its rows has the same ℓ_p norm, and each of its columns has the same ℓ_p norm. (The row and column ℓ_p norms are related by m, n, and p.) Suppose we are given a matrix $A \in \mathbf{R}^{m\times n}$. We seek diagonal invertible matrices $D \in \mathbf{R}^{m\times m}$ and $E \in \mathbf{R}^{n\times n}$ for which DAE is ℓ_p equilibrated.

   (a) Explain how to find D and E using convex optimization. (Some matrices cannot be equilibrated. But you can assume that all entries of A are nonzero, which is enough to guarantee that it can be equilibrated.)

   (b) Equilibrate the matrix A given in the file `matrix_equilibration_data.*`, with

   $$m = 20, \quad n = 10, \quad p = 2.$$

   Print the row ℓ_p norms and the column ℓ_p norms of the equilibrated matrix as vectors to check that each matches.

   Hints.

   - Work with the matrix B, with $B_{ij} = |A_{ij}|^p$.
   - Consider the problem of minimizing $\sum_{i=1}^m \sum_{j=1}^n B_{ij} e^{u_i+v_j}$ subject to $\mathbf{1}^T u = 0$, $\mathbf{1}^T v = 0$. (Several variations on this idea will work.)
   - We have found that expressing the terms in the objective as $e^{\log B_{ij}+u_i+v_j}$ leads to fewer numerical problems.

7. *Colorization with total variation regularization.* A $m \times n$ color image is represented as three matrices of intensities $R, G, B \in \mathbf{R}^{m \times n}$, with entries in $[0, 1]$, representing the red, green, and blue pixel intensities, respectively. A color image is converted to a monochrome image, represented as one matrix $M \in \mathbf{R}^{m \times n}$, using

$$M = 0.299R + 0.587G + 0.114B.$$

(These weights come from different perceived brightness of the three primary colors.)

In *colorization*, we are given M, the monochrome version of an image, and the color values of *some* of the pixels; we are to guess its color version, *i.e.*, the matrices R, G, B. Of course that's a very underdetermined problem. A very simple technique is to minimize the total variation of (R, G, B), defined as

$$\mathbf{tv}(R, G, B) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left\| \begin{bmatrix} R_{ij} - R_{i,j+1} \\ G_{ij} - G_{i,j+1} \\ B_{ij} - B_{i,j+1} \\ R_{ij} - R_{i+1,j} \\ G_{ij} - G_{i+1,j} \\ B_{ij} - B_{i+1,j} \end{bmatrix} \right\|_2,$$

subject to consistency with the given monochrome image, the known ranges of the entries of (R, G, B) (*i.e.*, in $[0, 1]$), and the given color entries. Note that the sum above is of the norm of 6-vectors, and not the norm-squared. (The 6-vector is an approximation of the spatial gradient of (R, G, B).)

Carry out this method on the data given in `image_colorization_data.*`. The file loads `flower.png` and provides the monochrome version of the image, `M`, along with vectors of known color intensities, `R_known`, `G_known`, and `B_known`, and `known_ind`, the indices of the pixels with known values. If `R` denotes the red channel of an image, then `R(known_ind)` returns the known red color intensities in Matlab, and `R[known_ind]` returns the same in Python and Julia. The file also creates an image, `flower_given.png`, that is monochrome, with the known pixels colored.

The `tv` function, invoked as `tv(R,G,B)`, gives the total variation. CVXPY has the `tv` function built-in, but CVX and CVX.jl do not, so we have provided the files `tv.m` and `tv.jl` which contain implementations for you to use.

In Python and Julia we have also provided the function `save_img(filename,R,G,B)` which writes the image defined by the matrices `R`, `G`, `B`, to the file `filename`. To view an image in Matlab use the `imshow` function.

The problem instance is a small image, 75×75, so the solve time is reasonable, say, under ten seconds or so in CVX or CVXPY, and around 60 seconds in Julia.

Report your optimal objective value and, if you have access to a color printer, attach your reconstructed image. If you don't have access to a color printer, it's OK to just give the optimal objective value.

8. *Computing market-clearing prices.* We consider n commodities or goods, with $p \in \mathbf{R}^n_{++}$ the vector of prices (per unit quantity) of them. The (nonnegative) demand for the products is a function of the prices, which we denote $D : \mathbf{R}^n \to \mathbf{R}^n$, so $D(p)$ is the demand when the product prices are p. The (nonnegative) supply of the products (*i.e.*, the amounts that manufacturers are willing to produce) is also a function of the prices, which we denote $S : \mathbf{R}^n \to \mathbf{R}^n$, so $S(p)$ is the supply when the product prices are p. We say that the market *clears* if $S(p) = D(p)$, *i.e.*, supply equals demand, and we refer to p in this case as a set of *market-clearing prices*.

   Elementary economics courses consider the special case $n = 1$, *i.e.*, a single commodity, so supply and demand can be plotted (vertically) against the price (on the horizontal axis). It is assumed that demand decreases with increasing price, and supply increases; the market clearing price can be found 'graphically', as the point where the supply and demand curves intersect. In this problem we examine some cases in which market-clearing prices (for the general case $n > 1$) can be computed using convex optimization.

   We assume that the demand function is *Hicksian*, which means it has the form $D(p) = \nabla E(p)$, where $E : \mathbf{R}^n \to \mathbf{R}$ is a differentiable function that is concave and increasing in each argument, called the *expenditure function*. (While not relevant in this problem, Hicksian demand arises from a model in which consumers make purchases by maximizing a concave utility function.)

   We will assume that the producers are independent, so $S(p)_i = S_i(p_i)$, $i = 1, \ldots, n$, where $S_i : \mathbf{R} \to \mathbf{R}$ is the supply function for good i. We will assume that the supply functions are positive and increasing on their domain $\mathbf{R}_+$.

   (a) Explain how to use convex optimization to find market-clearing prices under the assumptions given above. (You do not need to worry about technical details like zero prices, or cases in which there are no market-clearing prices.)

   (b) Compute market-clearing prices for the specific case with $n = 4$,

$$E(p) = \left(\prod_{i=1}^{4} p_i \right)^{1/4},$$

$$S(p) = (0.2p_1 + 0.5, 0.02p_2 + 0.1, 0.04p_3, 0.1p_4 + 0.2).$$

   Give the market-clearing prices and the demand and supply (which should match) at those prices.

   Hint: In CVX and CVXPY, `geo_mean` gives the geometric mean of the entries of a vector argument. Julia does not yet have a vector argument `geom_mean` function, but you can get the geometric mean of 4 variables a, b, c, d using `geomean(geomean(a, b), geomean(c, d))`.