EE364a: Convex Optimization I S. Boyd
March 14–15, 15–16, or 16–17, 2019

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 9:30AM March 17, 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 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 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 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 (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.

Some of the data files generate random data (with a fixed seed), which are not necessarily the same for Matlab, Python, and Julia.

1. *Fitting a periodic Poisson distribution to data.* We model the (random) number of times that some type of event occurs in each hour of the day as independent Poisson variables, with

$$\textbf{Prob}(k \text{ events occur}) = e^{-\lambda_t}\frac{\lambda_t^k}{k!}, \quad k = 0, 1, \ldots,$$

with parameter $\lambda_t \geq 0$, $t = 1, \ldots, 24$. (For $\lambda_t = 0$, $k = 0$ events occur with probability one.) Here t denotes the hour, with $t = 1$ corresponding to the hour from midnight to 1AM, and $t = 24$ the hour between 11PM and midnight. (This is the periodic Poisson distribution in the title.) The parameter λ_t is the expected value of the number of events that occur in hour t; it can be thought of as the rate of occurence of the events in hour t.

Over one day we observe the numbers of events $N_1, \ldots, N_{24}$.

(a) *Maximum likelihood estimate of parameters.* What is the maximum likelihood estimate of the parameters $\lambda_1, \ldots, \lambda_{24}$? *Hint.* There is a simple analytical solution. You should consider the cases $N_t > 0$ and $N_t = 0$ separately.

(b) *Regularized maximum likelihood estimate of parameters.* In many applications it is reasonable to assume that λ_t varies smoothly over the day; for example, the rate of occurence of events for 3PM–4PM is not too different from the rate of occurence for 4PM–5PM. To obtain a smooth estimate of λ_t we maximize the log likelihood minus the regularization term

$$\rho\left(\sum_{t=1}^{23}(\lambda_{t+1} - \lambda_t)^2 + (\lambda_1 - \lambda_{24})^2\right),$$

where $\rho \geq 0$. Explain how to find the values $\lambda_1, \ldots, \lambda_{24}$ using convex optimization. If you change variables, explain.

(c) What happens as $\rho \to \infty$? You can give a very short answer, with an informal argument. *Hint.* As in part (a), there is a simple analytical solution.

(d) *Numerical example.* Over one day, we observe

$$N = (0, 4, 2, 2, 3, 0, 4, 5, 6, 6, 4, 1, 4, 4, 0, 1, 3, 4, 2, 0, 3, 2, 0, 1).$$

Find the regularized maximum likelihood parameters for $\rho \in \{0.1, 1, 10, 100\}$ using CVX*, and plot λ_t versus t for each value of ρ.

(e) *Choosing the hyper-parameter value by out-of-sample test.* One way to choose the value of ρ is to see which of the models found in part (d) has the highest log likelihood on a test set, *i.e.*, another day's data, that was not used to create the model. For each of the 4 values of the parameters you estimated in part (d), evaluate the log likelihood of another day's number of events,

$$N^{\text{test}} = (0, 1, 3, 2, 3, 1, 4, 5, 3, 1, 4, 3, 5, 5, 2, 1, 1, 1, 2, 0, 1, 2, 1, 0).$$

Which hyper-parameter value ρ would you choose?

2. *Currency exchange.* An entity (such as a multinational corporation) holds $n = 10$ currencies, with $c_i^{\text{init}} \geq 0$ denoting the number of units of currency i. The currencies are, in order, USD, EUR, GBP, CAD, JPY, CNY, RUB, MXN, INR, and BRL. Our goal is to exchange currencies on a market so that, after the exchanges, we hold at least c_i^{req} units of each currency i.

   The exchange rates are given by $F \in \mathbf{R}^{n \times n}$, where F_{ij} is the units of currency j it costs to buy one unit of currency i. We call $1/F_{ij}$ the bid price for currency j in terms of currency i, and F_{ji} the ask price for currency j in terms of currency i.

   For example, suppose that $F_{12} = 0.88$ and $F_{21} = 1.18$. This means that it takes 0.88 EUR to buy one USD, and it takes 1.18 USD to buy one EUR; the bid and ask prices for EUR in USD are 1.1364 USD and 1.1800 USD, respectively.

   We will value a set of currency holdings in USD, by valuing each unit of currency j at the geometric mean of the bid and ask price in USD, $\sqrt{F_{j1}/F_{1j}}$. In our example above, we would value one EUR as $\sqrt{1.1364 \cdot 1.1800} = 1.1580$ USD.

   We let $X \in \mathbf{R}_+^{n \times n}$ denote the currency exchanges that we carry out, with $X_{ij} \geq 0$ the amount of currency j we exchange on the market for currency i, for which we obtain X_{ij}/F_{ij} of currency i. (You can assume that $X_{ii} = 0$.) The total of each currency j that we exchange into other currencies cannot exceed our initial holdings, c_j^{init}. After the currency exchange, we must end up with at least c_i^{req} of currency i. (The post-exchange amount we hold of currency i is our original holding c_i^{init}, minus the total we exchange into other currencies, plus the total amount we obtain from exchanging other currencies into currency i.)

   The cost of the exchanges is the decrease in value between the currency holdings before and after the exchanges, in USD. The cost can be interpreted as the transaction costs incurred by crossing the bid-ask spread (*i.e.*, if the bid and the ask were the same, there would be no cost.)

   Find the currency exchanges $X^\star$ that minimize the currency exchange cost for the data in `currency_exchange_data.*`. (These data are based on real exchange rates, but with artificially large spreads, to make sure that you don't encounter any numerical issues.) Explain your method, and give the optimal value, *i.e.*, the cost obtained.

3. *Optimal operation of a microgrid.* We consider a small electrical microgrid that consists of a photo-voltaic (PV) array, a storage device (battery), a load, and a connection to an external grid. We will optimize the operation of the microgrid over one day, in 15 minute increments, so all powers, and the battery charge, are represented as vectors in $\mathbf{R}^{96}$. The load power is p^{ld}, which is nonnegative and known. The power that we take from the external grid is p^{grid}; $p_i^{\text{grid}} \geq 0$ means we are consuming power from the grid, and $p_i^{\text{grid}} < 0$ means we are sending power back into the grid, in time period i. The PV array output, which is nonnegative and known, is denoted as p^{pv}. The battery power is p^{batt}, with $p_i^{\text{batt}} \geq 0$ meaning the battery is discharging, and $p_i^{\text{batt}} < 0$ meaning the battery is charging. These powers must balance in all periods, *i.e.*, we have

$$p^{\text{ld}} = p^{\text{grid}} + p^{\text{batt}} + p^{\text{pv}}.$$

(This is called the power balance constraint. The lefthand side is the load power, and the righthand side is the sum of the power coming from the grid, the battery, and the PV array.) All powers are given in kW.

The battery state of charge is given by $q \in \mathbf{R}^{96}$. It must satisfy $0 \leq q_i \leq Q$ for all i, where Q is the battery capacity (in kWh). The battery dynamics are

$$q_{i+1} = q_i - (1/4)p_i^{\text{batt}}, \quad i = 1, \ldots, 95, \quad q_1 = q_{96} - (1/4)p_{96}^{\text{batt}}.$$

(The last equation means that we seek a periodic operation of the microgrid.) The battery power must satisfy $-C \leq p_i^{\text{batt}} \leq D$ for all i, where C and D are (positive) known maximum charge and maximum discharge rates.

When we buy power (*i.e.*, $p_i^{\text{grid}} \geq 0$) we pay for it at the rate of R_i^{buy} (in \$/kWh). When we sell power to the grid (*i.e.*, $p_i^{\text{grid}} < 0$) we are paid for it at the rate of R_i^{sell}. These (positive) prices vary with time period, and are known. The total cost of the grid power (in \$) is

$$(1/4)\left(R^{\text{buy}}\right)^T\left(p^{\text{grid}}\right)_+ - (1/4)\left(R^{\text{sell}}\right)^T\left(p^{\text{grid}}\right)_-,$$

where $(p^{\text{grid}})_+ = \max\{p^{\text{grid}}, 0\}$ and $(p^{\text{grid}})_- = \max\{-p^{\text{grid}}, 0\}$ (elementwise). You can assume that $R_i^{\text{buy}} > R_i^{\text{sell}} > 0$, *i.e.*, in every period, you pay at a higher rate to consume power from the grid than you are paid when you send power back into the grid.

The data for the problem are

$$p^{\text{ld}}, \quad p^{\text{pv}}, \quad Q, \quad C, \quad D, \quad R^{\text{buy}}, \quad R^{\text{sell}}.$$

(a) Explain how to find the powers and battery state of charge that minimize the total cost of the grid power. Carry out your method using the data given in `microgrid_data.*`. Report the optimal cost of the grid power. Plot p^{grid}, p^{load}, p^{pv}, p^{batt}, and q versus i. *Note.* For CVXPY, you might need to specify `solver=cvx.ECOS` when you call the `solve()` method.

(b) *Price and payments.* Let $\nu \in \mathbf{R}^{96}$ denote the optimal dual variable associated with the power balance constraint. The vector 4ν can be interpreted as the (time-varying) price of electricity at the microgrid, and is called the *locational marginal price* (LMP). The LMP is in \$/kWh, and is generally positive; the factor 4 converts between 15 minute power intervals and per kWh prices. Find and plot the LMP, along with the grid buy and sell prices, versus i. Make a very brief comment comparing the LMP prices with the buy and sell grid prices. *Hint.* Depending on how you express the power balance constraint, your software might return $-\nu$ instead of ν. Feel free to use -4ν instead of ν, or to switch the left-hand and righ-hand sides of your power balance constraint.

(c) The LMPs can be used as a system for payments among the load, the PV array, the battery, and the grid. The load pays $\nu^T p^{\text{ld}}$; the PV array is paid $\nu^T p^{\text{pv}}$; the battery is paid $\nu^T p^{\text{batt}}$; and the grid is paid $\nu^T p^{\text{grid}}$. Note carefully the directions of these payments. Also note that the battery and grid, whose powers can have either sign, can be paid in some time intervals and pay in others.

Use this pricing scheme to calculate the LMP payments made by the load, and to the PV array, the battery, and the grid. If all goes well, these payments will balance, *i.e.*, the load will pay an amount equal to the sum of the others.

When you execute the script that contains the data, it will create plots showing the various powers and prices versus time. Your are welcome to use these as templates for plotting your results. You are very welcome to look inside the script to see how the data is generated.

Remark. (Not needed to solve the problem.) The given data is approximately consistent with a group of ten houses, a common or pooled PV array of around 100 panels, and two Tesla Powerwall batteries.

4. *Curvature of some order statistics.* For $x \in \mathbf{R}^n$, with $n > 1$, $x_{[k]}$ denotes the kth largest entry of x, for $k = 1, \ldots, n$, so, for example, $x_{[1]} = \max_{i=1,\ldots,n} x_i$ and $x_{[n]} = \min_{i=1,\ldots,n} x_i$. Functions that depend on these sorted values are called order statistics or order functions. Determine the curvature of the order statistics below, from the choices convex, concave, or neither. For each function, explain why the function has the curvature you claim. If you say it is neither convex nor concave, give a counterexample showing it is not convex, and a counterexample showing it is not concave. All functions below have domain $\mathbf{R}^n$.

   (a) $\mathrm{median}(x) = x_{[(n+1)/2]}$. (You can assume that n is odd.)

   (b) The range of values, $x_{[1]} - x_{[n]}$.

   (c) The midpoint of the range, $(x_{[1]} + x_{[n]})/2$.

   (d) Interquartile range, defined as $x_{[n/4]} - x_{[3n/4]}$. (You can assume that $n/4$ is an integer.)

   (e) Symmetric trimmed mean, defined as

   $$\frac{x_{[n/10]} + x_{[n/10+1]} + \cdots + x_{[9n/10]}}{0.8n + 1},$$

   the mean of the values between the 10th and 90th percentiles. (You can assume that $n/10$ is an integer.)

   (f) Lower trimmed mean, defined as

   $$\frac{x_{[1]} + x_{[2]} + \cdots + x_{[9n/10]}}{0.9n + 1},$$

   the mean of the entries, excluding the bottom decile. (You can assume that $n/10$ is an integer.)

   Remark. For the functions defined in (d)–(f), you might find slightly different definitions in the literature. Please use the formulas above to answer each question.

5. *Control with various objectives.* We consider a standard optimal control problem, with dynamics $x_{t+1} = Ax_t + Bu_t$, $t = 0, 1, \ldots, T-1$. Here $x_t \in \mathbf{R}^n$ is the state, and $u_t \in \mathbf{R}^m$ is the control or input, at time period t, $A \in \mathbf{R}^{n \times n}$ is the dynamics matrix, and $B \in \mathbf{R}^{n \times m}$ is the input matrix. We are given the initial state, $x_0 = x^{\rm init}$, and we require that the final state be zero, $x_T = 0$. (In applications, the state 0 corresponds to some desirable state.) Your job is to choose the sequence of inputs $u_0, \ldots, u_{T-1}$ that minimize an objective. Values for $x^{\rm init}$, A, B, and T are given in `various_obj_regulator_data.*`.

   We consider various objectives, all of which measure the size of the inputs (or, in control dialect, the *control effort*).

   (a) *Sum of squares of 2-norms.* $\sum_{t=0}^{T-1} \|u_t\|_2^2$. This is the traditional objective.

   (b) *Sum of 2-norms.* $\sum_{t=0}^{T-1} \|u_t\|_2$.

   (c) *Max of 2-norms.* $\max_{t=0,\ldots,T-1} \|u_t\|_2$.

   (d) *Sum of 1-norms.* $\sum_{t=0}^{T-1} \|u_t\|_1$. In some applications this is an approximation of the fuel use.

   For each objective, plot (the components of) optimal input, as well as $\|u_t\|_2$, versus t. Make a very brief comment on each plot of optimal control inputs, explaining why you might expect what happened.

6. *Morphing between two discrete distributions.* Consider two distributions for a random variable that takes values in $\{1, 2, \ldots, n\}$, given by $q, r \in \mathbf{R}^n$, with $q \succeq 0$, $\mathbf{1}^T q = 1$, and $r \succeq 0$, $\mathbf{1}^T r = 1$. We seek a sequence of distributions $p^{(i)}$, $i = 1, \ldots, N$, that 'morph' between q and r. This means that $p^{(1)} = q$, $p^{(N)} = r$, and $p^{(i+1)}$ is close to $p^{(i)}$ for $i = 1, \ldots, (N-1)$, in some sense. Specifically we will minimize

$$\sum_{i=1}^{N-1} d(p^{(i)}, p^{(i+1)})$$

where d is a distance function.

(a) *Euclidean morphing.* What is the solution when the distance function is the sum of squares, $d^{\mathrm{sq}}(u, v) = \|u - v\|_2^2$? The solution is simple; you can just give it without justification.

(b) *Hellinger morphing.* Now we use the Hellinger distance function

$$d^{\mathrm{hel}}(u, v) = \sum_{i=1}^{n} (\sqrt{u_i} - \sqrt{v_i})^2.$$

Explain how to solve the Hellinger morphing problem using convex optimization.

(c) *Kolmogorov morphing.* Now we use the Kolmogorov distance function

$$d^{\mathrm{kol}}(u, v) = \max_{i=1,\ldots,n} \left| \sum_{j=1}^{i} u_j - \sum_{j=1}^{i} v_j \right|,$$

which is the ℓ_∞ distance between the respective cumulative distributions (using the order of the outcomes). Explain how to solve the Kolmogorov morphing problem using convex optimization.

(d) Find the Euclidean, Hellinger, and Kolmogorov morphings for $N = 10$, $n = 100$. Use q and r provided in `morphing_data.*`. Plot each $p^{(i)}$ versus n. Produce one figure for each choice of distance function.

Note. In Python and Julia, you should use the ECOS solver.

7. *Constrained maximum likelihood estimation of mean and covariance.* You are given some independent samples $x_1, \ldots, x_N \in \mathbf{R}^n$ from a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Explain how to find the maximum-likelihood estimate of μ and Σ, subject to the constraint that $\Sigma^{-1}\mu \succeq 0$, using convex optimization. You must fully justify any change of variables.

   Finance interpretation. (Not needed to solve the problem.) Suppose $x \sim \mathcal{N}(\mu, \Sigma)$ is the return of n assets. The portfolio vector h that maximizes the risk-adjusted return $\mu^T h - \gamma h^T \Sigma h$, where $\gamma > 0$ is the risk aversion parameter, is $h = (1/2\gamma)\Sigma^{-1}\mu$. So the constraint in the problem above is that the optimal portfolio has nonnegative entries, *i.e.*, is a long-only portfolio. The constrained maximum-likelihood estimate finds the maximum likelihood mean and covariance of the return distribution, subject to the constraint that the associated optimal portfolio is long-only.

   Probability interpretation. (Not needed to solve the problem.) The constraint $\Sigma^{-1}\mu \succeq 0$ is the same as $\nabla p(0) \succeq 0$, where p is the density of the $\mathcal{N}(\mu, \Sigma)$ distribution. In other words, at 0, the density is nondecreasing in each coordinate.

8. *Minimizing tax liability.* You will liquidate (sell) some stocks that you hold to raise a given amount of cash C. The stocks are divided into n tax lots; a tax lot is a group of stocks you bought at the same time. For each tax lot i, you have the cost basis $b_i > 0$, the current market value $v_i > 0$ (both in \$), and its short term / long term status. (Long term means that you acquired the stock in the tax lot more than one year ago, and short term means that you acquired it less than one year ago.) We assume that tax lots $i = 1, \ldots, L$ are long term, and tax lots $i = L+1, \ldots, n$ are short term.

   The goal is to choose how much of each lot to sell. We let s_i denote the amount of tax lot i we sell (in \$). These must satisfy $0 \leq s_i \leq v_i$, and we must have $\mathbf{1}^T s = C$.

   When $v_i < b_i$, the sale is called a loss, and when $v_i > b_i$, the sale is called a gain. The amount of the gain or loss is given by $g_i = (s_i/v_i)(v_i - b_i)$, with positive values meaning a gain, and negative values meaning a loss. We define the (net) long and short term gains as

   $$N^{\mathrm{l}} = \sum_{i=1}^{L} g_i, \qquad N^{\mathrm{s}} = \sum_{i=L+1}^{n} g_i.$$

   When $N^{\mathrm{l}} > 0$ ($N^{\mathrm{l}} < 0$), we say that we have had a long term capital gain (loss), and similary for short term gain.

   These two net gains determine the total tax liability. The long and short term net gains are taxed at two different rates, ρ^{l} and ρ^{s}, respectively, which satisfy $0 < \rho^{\mathrm{l}} < \rho^{\mathrm{s}}$.

   The simplest case is when both net gains are nonnegative, in which case the tax is $\rho^{\mathrm{l}} N^{\mathrm{l}} + \rho^{\mathrm{s}} N^{\mathrm{s}}$. Another simple case occurs when both net gains are nonpositive, in which case the tax is zero.

   In the case when one of the net gains is positive and the other is negative, you are allowed to use the net loss in one to offset the net gain in the other, up to the value of the net gain. Specifically, if $N^{\mathrm{l}} < 0$ (you have a long term loss), the tax is $\rho^{\mathrm{s}}(N^{\mathrm{s}} + N^{\mathrm{l}})_+$; if $N^{\mathrm{s}} < 0$ (you have a short term loss), the tax is $\rho^{\mathrm{l}}(N^{\mathrm{s}} + N^{\mathrm{l}})_+$. (Here $(u)_+ = \max\{u, 0\}$.) Note that you have zero tax liability if $N^{\mathrm{s}} + N^{\mathrm{l}} \leq 0$, *i.e.*, your total long and short net gains is less than or equal to zero.

   Apology. Sorry this sounds complicated. In fact, this is a highly simplified version of the way taxes really work.

   Hint. The tax liability is neither a convex nor quasiconvex function of the long and short term net gains N^{l} and N^{s}.

   (a) Explain how to find s that minimizes the tax liability, subject to the constraints listed above, using convex optimization. Your solution can involve solving a modest number of convex problems.

   (b) Suppose you want to raise $C = 2300$ dollars from $n = 10$ tax lots, and the cost basis and values of each lot are given by

$$\begin{aligned} b &= (400, 80, 400, 200, 400, 400, 80, 400, 100, 500), \\ v &= (500, 100, 500, 200, 700, 300, 120, 300, 150, 600). \end{aligned}$$

Carry out your method on this data with $L = 4$, $\rho_l = 0.2$, and $\rho_s = 0.3$. Give optimal values of s_i, and the optimal value of the tax liability. Compare this to the tax liability when you liquidate all tax lots proportionally, *i.e.*, $s = (C/\mathbf{1}^T v)v$.

9. *Fitting with a nonnegative combination of vectors from ellipsoids.* You are given ellipsoids $\mathcal{E}_1, \ldots, \mathcal{E}_n \subset \mathbf{R}^k$, and the vector $b \in \mathbf{R}^k$. Explain how to use convex optimization to choose $a_i \in \mathcal{E}_i$, $i = 1, \ldots, n$, and nonnegative $x_1, \ldots, x_n \in \mathbf{R}$, that minimize

$$\left\| \sum_{i=1}^n x_i a_i - b \right\|_2 .$$

You can use any parametrization of the ellipsoids you like, for example,

$$\mathcal{E}_i = \{a \mid \|P_i a + q_i\|_2 \leq 1\},$$

or

$$\mathcal{E}_i = \{P_i u + q_i \mid \|u\|_2 \leq 1\},$$

or

$$\mathcal{E}_i = \left\{a \mid (a - c_i)^T P_i^{-1} (a - c_i) \leq 1\right\},$$

with $P_i \in \mathbf{S}^k_{++}$ and $c_i \in \mathbf{R}^k$.

Remark. This is the opposite situation from robust approximation. In robust approximation, the a_i's would be chosen to maximize the objective, once you choose x. Here, however, the a_i's are chosen to minimize the objective, along with x.