EE364 Convex Optimization Prof. S. Boyd
June 7 – 8 or June 8 – 9, 2006.

Final exam

You may use any books, notes, or computer programs (*e.g.*, Matlab, `cvx`), but you may not discuss the exam with anyone until June 10, after everyone has taken the exam. The only exception is that you can ask the TAs or Stephen Boyd for clarification. 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 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.

Since this is a take home exam, you will be graded on clarity and conciseness as well as accuracy and correctness. **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 Matlab (or other) source code that produces the result, and the final numerical results or plots.

Some of the problems require you to download a Matlab file containing problem data. You'll have to type the whole URL given in the problem into your browser; there are no links on the course web page pointing to these files. To get a file called `data.m`, for example, you'd type `www.stanford.edu/class/ee364/data.m` into your browser.

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 easier than they might appear at first glance. (That's why there are 8 questions this year ...)

Very important: You will be using `cvx` during the exam. Be sure that you have downloaded and installed the most recent version of `cvx`. We've tried very hard to make sure that no `cvx` bug is going to rear its ugly head during the final exam, but of course this *might* happen. If you've written down some `cvx` code that you think should work, but it doesn't, don't hesitate to email us to check. Also, be sure to check your email often during the exam, just in case we do need to send out an important announcement.

1. *Optimizing processor speed.* A set of n tasks is to be completed by n processors. The variables to be chosen are the processor speeds $s_1, \ldots, s_n$, which must lie between a given minimum value $s_{\min}$ and a maximum value $s_{\max}$. The computational load of task i is α_i, so the time required to complete task i is $\tau_i = \alpha_i/s_i$.

   The power consumed by processor i is given by $p_i = f(s_i)$, where $f : \mathbf{R} \to \mathbf{R}$ is positive, increasing, and convex. Therefore, the total energy consumed is

   $$E = \sum_{i=1}^{n} \frac{\alpha_i}{s_i} f(s_i).$$

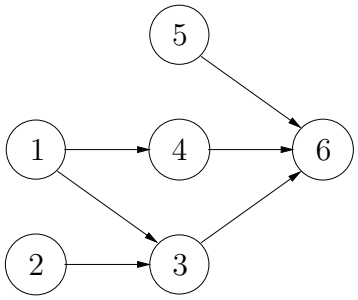
   (Here we ignore the energy used to transfer data between processors, and assume the processors are powered down when they are not active.)

   There is a set of *precedence constraints* for the tasks, which is a set of m ordered pairs $\mathcal{P} \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$. If $(i, j) \in \mathcal{P}$, then task j cannot start until task i finishes. (This would be the case, for example, if task j requires data that is computed in task i.) When $(i, j) \in \mathcal{P}$, we refer to task i as a *precedent* of task j, since it must precede task j. We assume that the precedence constraints define a directed acyclic graph (DAG), with an edge from i to j if $(i, j) \in \mathcal{P}$.

   If a task has no precedents, then it starts at time $t = 0$. Otherwise, each task starts as soon as all of its precedents have finished. We let T denote the time for all tasks to be completed.

   To be sure the precedence constraints are clear, we consider the very small example shown below, with $n = 6$ tasks and $m = 6$ precedence constraints.

   $$\mathcal{P} = \{(1,4),\ (1,3),\ (2,3),\ (3,6),\ (4,6),\ (5,6)\}.$$

   In this example, tasks 1, 2, and 5 start at time $t = 0$ (since they have no precedents). Task 1 finishes at $t = \tau_1$, task 2 finishes at $t = \tau_2$, and task 5 finishes at $t = \tau_5$. Task 3 has tasks 1 and 2 as precedents, so it starts at time $t = \max\{\tau_1, \tau_2\}$, and ends τ_3 seconds later, at $t = \max\{\tau_1, \tau_2\} + \tau_3$. Task 4 completes at time $t = \tau_1 + \tau_4$. Task 6 starts when tasks 3, 4, and 5 have finished, at time $t = \max\{\max\{\tau_1, \tau_2\} + \tau_3, \tau_1 + \tau_4, \tau_5\}$. It finishes τ_6 seconds later. In this example, task 6 is the last task to be completed, so we have

   $$T = \max\{\max\{\tau_1, \tau_2\} + \tau_3, \tau_1 + \tau_4, \tau_5\} + \tau_6.$$

(a) Formulate the problem of choosing processor speeds (between the given limits) to minimize completion time T, subject to an energy limit $E \leq E_{\max}$, as a convex optimization problem. The data in this problem are $\mathcal{P}$, $s_{\min}$, $s_{\max}$, $\alpha_1, \ldots, \alpha_n$, $E_{\max}$, and the function f. The variables are $s_1, \ldots, s_n$.

Feel free to change variables or to introduce new variables. Be sure to explain clearly why your formulation of the problem is convex, and why it is equivalent to the problem statement above.

Important:

- Your formulation must be convex for any function f that is positive, increasing, and convex. You cannot make any further assumptions about f.
- This problem refers to the general case, not the small example described above.

(b) Consider the specific instance with data given in `ps_data.m`, and processor power

$$f(s) = 1 + s + s^2 + s^3.$$

The precedence constraints are given by an $m \times 2$ matrix `prec`, where m is the number of precedence constraints, with each row giving one precedence constraint (the first column gives the precedents).

Plot the optimal trade-off curve of energy E versus time T, over a range of T that extends from its minimum to its maximum possible value. (These occur when all processors operate at $s_{\max}$ and $s_{\min}$, respectively, since T is monotone nonincreasing in s.) On the same plot, show the energy-time trade-off obtained when all processors operate at the same speed $\bar{s}$, which is varied from $s_{\min}$ to $s_{\max}$.

Note: In this part of the problem there is no limit $E^{\max}$ on E as in part (a); you are to find the optimal trade-off of E versus T.

2. *Exploring nearly optimal points.* An optimization algorithm will find *an* optimal point for a problem, provided the problem is feasible. It is often useful to explore the set of nearly optimal points. When a problem has a 'strong minimum', the set of nearly optimal points is small; all such points are close to the original optimal point found. At the other extreme, a problem can have a 'soft minimum', which means that there are many points, some quite far from the original optimal point found, that are feasible and have nearly optimal objective value. In this problem you will use a typical method to explore the set of nearly optimal points.

   We start by finding the optimal value $p^\star$ of the given problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p, \end{array}$$

   as well as an optimal point $x^\star \in \mathbf{R}^n$. We then pick a small positive number ϵ, and a vector $c \in \mathbf{R}^n$, and solve the problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \\ & f_0(x) \leq p^\star + \epsilon. \end{array}$$

   Note that any feasible point for this problem is ϵ-suboptimal for the original problem. Solving this problem multiple times, with different c's, will generate (perhaps different) ϵ-suboptimal points. If the problem has a strong minimum, these points will all be close to each other; if the problem has a weak minimum, they can be quite different.

   There are different strategies for choosing c in these experiments. The simplest is to choose the c's randomly; another method is to choose c to have the form $\pm e_i$, for $i = 1, \ldots, n$. (This method gives the 'range' of each component of x, over the ϵ-suboptimal set.)

   You will carry out this method for the following problem, to determine whether it has a strong minimum or a weak minimum. You can generate the vectors c randomly, with enough samples for you to come to your conclusion. You can pick $\epsilon = 0.01 p^\star$, which means that we are considering the set of 1% suboptimal points.

   The problem is a minimum fuel optimal control problem for a vehicle moving in $\mathbf{R}^2$. The position at time kh is given by $p(k) \in \mathbf{R}^2$, and the velocity by $v(k) \in \mathbf{R}^2$, for $k = 1, \ldots, K$. Here $h > 0$ is the sampling period. These are related by the equations

$$p(k+1) = p(k) + hv(k), \quad v(k+1) = (1-\alpha)v(k) + (h/m)f(k), \quad k = 1, \ldots, K-1,$$

   where $f(k) \in \mathbf{R}^2$ is the force applied to the vehicle at time kh, $m > 0$ is the vehicle mass, and $\alpha \in (0, 1)$ models drag on the vehicle; in the absense of any other force, the vehicle velocity decreases by the factor $1 - \alpha$ in each discretized time interval.

(These formulas are approximations of more accurate formulas that involve matrix exponentials.)

The force comes from two thrusters, and from gravity:

$$f(k) = \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} u_1(k) + \begin{bmatrix} \cos\theta_2 \\ \sin\theta_2 \end{bmatrix} u_2(k) + \begin{bmatrix} 0 \\ -mg \end{bmatrix}, \quad k = 1, \ldots, K-1.$$

Here $u_1(k) \in \mathbf{R}$ and $u_2(k) \in \mathbf{R}$ are the (nonnegative) thruster force magnitudes, θ_1 and θ_2 are the directions of the thrust forces, and $g = 10$ is the constant acceleration due to gravity.

The total fuel use is

$$F = \sum_{k=1}^{K-1} \left(u_1(k) + u_2(k)\right).$$

(Recall that $u_1(k) \geq 0$, $u_2(k) \geq 0$.)

The problem is to minimize fuel use subject to the initial condition $p(1) = 0$, $v(1) = 0$, and the way-point constraints

$$p(k_i) = w_i, \quad i = 1, \ldots, M.$$

(These state that at the time hk_i, the vehicle must pass through the location $w_i \in \mathbf{R}^2$.) In addition, we require that the vehicle should remain in a square operating region,

$$\|p(k)\|_\infty \leq P^{\max}, \quad k = 1, \ldots, K.$$

Both parts of this problem concern the specific problem instance with data given in `thrusters_data.m`.

(a) Find an optimal trajectory, and the associated minimum fuel use $p^\star$. Plot the trajectory $p(k)$ in $\mathbf{R}^2$ (*i.e.*, in the p_1, p_2 plane). Verify that it passes through the way-points.

(b) Generate several 1% suboptimal trajectories using the general method described above, and plot the associated trajectories in $\mathbf{R}^2$. Would you say this problem has a strong minimum, or a weak minimum?

3. *Estimating a vector with unknown nonlinear measurement nonlinearity.* We want to estimate a vector $x \in \mathbf{R}^n$, given some measurements

$$y_i = \phi(a_i^T x + v_i), \quad i = 1, \ldots, m.$$

Here $a_i \in \mathbf{R}^n$ are known, v_i are IID $\mathcal{N}(0, \sigma^2)$ random noises, and $\phi : \mathbf{R} \to \mathbf{R}$ is an unknown monotonic increasing function, known to satisfy

$$\alpha \leq \phi'(u) \leq \beta,$$

for all u. (Here α and β are known positive constants, with $\alpha < \beta$.) We want to find a maximum likelihood estimate of x and ϕ, given y_i. (We also know a_i, σ, α, and β.)

This sounds like an infinite-dimensional problem, since one of the parameters we are estimating is a function. In fact, we only need to know the m numbers $z_i = \phi^{-1}(y_i)$, $i = 1, \ldots, m$. So by estimating ϕ we really mean estimating the m numbers $z_1, \ldots, z_m$. (These numbers are not arbitrary; they must be consistent with the prior information $\alpha \leq \phi'(u) \leq \beta$ for all u.)

(a) Explain how to find a maximum likelihood estimate of x and ϕ (*i.e.*, $z_1, \ldots, z_m$) using convex optimization.

(b) Carry out your method on the data given in `nonlin_meas_data.m`, which includes a matrix $A \in \mathbf{R}^{m \times n}$, with rows $a_1^T, \ldots, a_m^T$. Give $\hat{x}_{\mathrm{ml}}$, the maximum likelihood estimate of x. Plot your estimated function $\hat{\phi}_{\mathrm{ml}}$. (You can do this by plotting $(\hat{z}_{\mathrm{ml}})_i$ versus y_i, with y_i on the vertical axis and $(\hat{z}_{\mathrm{ml}})_i$ on the horizontal axis.)

Hint. You can assume the measurements are numbered so that y_i are sorted in nondecreasing order, *i.e.*, $y_1 \leq y_2 \leq \cdots \leq y_m$. (The data given in the problem instance for part (b) is given in this order.)

4. *Optimizing rates and time slot fractions.* We consider a wireless system that uses time-domain multiple access (TDMA) to support n communication flows. The flows have (nonnegative) rates $r_1, \ldots, r_n$, given in bits/sec. To support a rate r_i on flow i requires transmitter power

$$p = a_i(e^{br} - 1),$$

where b is a (known) positive constant, and a_i are (known) positive constants related to the noise power and gain of receiver i.

TDMA works like this. Time is divided up into periods of some fixed duration T (seconds). Each of these T-long periods is divided into n time-slots, with durations $t_1, \ldots, t_n$, that must satisfy $t_1 + \cdots + t_n = T$, $t_i \geq 0$. In time-slot i, communications flow i is transmitted at an instantaneous rate $r = Tr_i/t_i$, so that over each T-long period, Tr_i bits from flow i are transmitted. The power required during time-slot i is $a_i(e^{bTr_i/t_i} - 1)$, so the average transmitter power over each T-long period is

$$P = (1/T)\sum_{i=1}^{n} a_i t_i (e^{bTr_i/t_i} - 1).$$

When t_i is zero, we take $P = \infty$ if $r_i > 0$, and $P = 0$ if $r_i = 0$. (The latter corresponds to the case when there is zero flow, and also, zero time allocated to the flow.)

The problem is to find rates $r \in \mathbf{R}^n$ and time-slot durations $t \in \mathbf{R}^n$ that maximize the log utility function

$$U(r) = \sum_{i=1}^{n} \log r_i,$$

subject to $P \leq P^{\max}$. (This utility function is often used to ensure 'fairness'; each communication flow gets at least some positive rate.) The problem data are a_i, b, T and $P^{\max}$; the variables are t_i and r_i.

(a) Formulate this problem as a convex optimization problem. Feel free to introduce new variables, if needed, or to change variables. Be sure to justify convexity of the objective or constraint functions in your formulation.

(b) Give the optimality conditions for your formulation. Of course we prefer simpler optimality conditions to complex ones. *Note:* We do not expect you to *solve* the optimality conditions; you can give them as a set of equations (and possibly inequalities).

Hint. With a log utility function, we cannot have $r_i = 0$, and therefore we cannot have $t_i = 0$; therefore the constraints $r_i \geq 0$ and $t_i \geq 0$ cannot be active or tight. This will allow you to simplify the optimality conditions.

5. *Feature selection and sparse linear separation.* Suppose $x^{(1)}, \ldots, x^{(N)}$ and $y^{(1)}, \ldots, y^{(M)}$ are two given nonempty collections or classes of vectors in $\mathbf{R}^n$ that can be (strictly) separated by a hyperplane, *i.e.*, there exists $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$ such that

$$a^T x^{(i)} - b \geq 1, \quad i = 1, \ldots, N, \qquad a^T y^{(i)} - b \leq -1, \quad i = 1, \ldots, M.$$

This means the two classes are (weakly) separated by the slab

$$S = \{z \mid |a^T z - b| \leq 1\},$$

which has thickness $2/\|a\|_2$. You can think of the components of $x^{(i)}$ and $y^{(i)}$ as *features*; a and b define an affine function that combines the features and allows us to distinguish the two classes.

To find the thickest slab that separates the two classes, we can solve the QP

$$\begin{array}{ll} \text{minimize} & \|a\|_2 \\ \text{subject to} & a^T x_i - b \geq 1, \quad i = 1, \ldots, N \\ & a^T y_i - b \leq -1, \quad i = 1, \ldots, M, \end{array}$$

with variables $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$. (This is equivalent to the problem given in (8.23), p424, §8.6.1; see also exercise 8.23.)

In this problem we seek (a, b) that separate the two classes with a thick slab, and also has a sparse, *i.e.*, there are many j with $a_j = 0$. Note that if $a_j = 0$, the affine function $a^T z - b$ does not depend on z_j, *i.e.*, the jth feature is not used to carry out classification. So a sparse a corresponds to a classification function that is parsimonius; it depends on just a few features. So our goal is to find an affine classification function that gives a thick separating slab, and also uses as few features as possible to carry out the classification.

This is in general a hard combinatorial (bi-criterion) optimization problem, so we use the standard heuristic of solving

$$\begin{array}{ll} \text{minimize} & \|a\|_2 + \lambda \|a\|_1 \\ \text{subject to} & a^T x_i - b \geq 1, \quad i = 1, \ldots, N \\ & a^T y_i - b \leq -1, \quad i = 1, \ldots, M, \end{array}$$

where $\lambda \geq 0$ is a weight vector that controls the trade-off between separating slab thickness and (indirectly, through the ℓ_1 norm) sparsity of a.

Get the data in `sp_ln_sp_data.m`, which gives x_i and y_i as the columns of matrices `X` and `Y`, respectively. Find the thickness of the maximum thickness separating slab. Solve the problem above for 100 or so values of λ over an appropriate range (we recommend log spacing). For each value, record the separation slab thickness $2/\|a\|_2$ and $\mathbf{card}(a)$, the cardinality of a (*i.e.*, the number of nonzero entries). In computing the cardinality, you can count an entry a_j of a as zero if it satisfies $|a_j| \leq 10^{-4}$. Plot these data with slab thickness on the vertical axis and cardinality on the horizontal axis.

Use this data to choose a set of 10 features out of the 50 in the data. Give the indices of the features you choose. You may have several choices of sets of features here; you can just choose one. Then find the maximum thickness separating slab that uses only the chosen features. (This is standard practice: once you've chosen the features you're going to use, you optimize again, using only those features, and without the ℓ_1 regularization.

6. *Bounding object position from multiple camera views.* A small object is located at unknown position $x \in \mathbf{R}^3$, and viewed by a set of m cameras. Our goal is to find a box in $\mathbf{R}^3$,

$$\mathcal{B} = \{z \in \mathbf{R}^3 \mid l \preceq z \preceq u\},$$

for which we can guarantee $x \in \mathcal{B}$. We want the smallest possible such bounding box. (Although it doesn't matter, we can use volume to judge 'smallest' among boxes.)

Now we describe the cameras. The object at location $x \in \mathbf{R}^3$ creates an image on image plane of camera i at location

$$v_i = \frac{1}{c_i^T x + d_i}(A_i x + b_i) \in \mathbf{R}^2.$$

The matrices $A_i \in \mathbf{R}^{2\times 3}$, vectors $b_i \in \mathbf{R}^2$ and $c_i \in \mathbf{R}^3$, and real numbers $d_i \in \mathbf{R}$ are known, and depend on the camera positions and orientations. We assume that $c_i^T x + d_i > 0$. The 3×4 matrix

$$P_i = \begin{bmatrix} A_i & b_i \\ c_i^T & d_i \end{bmatrix}$$

is called the *camera matrix* (for camera i). It is often (but not always) the case the that the first 3 columns of P_i (*i.e.*, A_i stacked above c_i^T) form an orthogonal matrix, in which case the camera is called *orthographic*.

We do not have direct access to the image point v_i; we only know the (square) pixel that it lies in. In other words, the camera gives us a measurement $\hat{v}_i$ (the center of the pixel that the image point lies in); we are guaranteed that

$$\|v_i - \hat{v}_i\|_\infty \leq \rho_i/2,$$

where ρ_i is the pixel width (and height) of camera i. (We know nothing else about v_i; it could be any point in this pixel.)

Given the data A_i, b_i, c_i, d_i, $\hat{v}_i$, ρ_i, we are to find the smallest box $\mathcal{B}$ (*i.e.*, find the vectors l and u) that is guaranteed to contain x. In other words, find the smallest box in $\mathbf{R}^3$ that contains all points consistent with the observations from the camera.

(a) Explain how to solve this using convex or quasiconvex optimization. You must explain any transformations you use, any new variables you introduce, etc. If the convexity or quasiconvexity of any function in your formulation isn't obvious, be sure justify it.

(b) Solve the specific problem instance given in the file `camera_data.m`. Be sure that your final numerical answer (*i.e.*, l and u) stands out.

7. $\ell_{1.5}$ *optimization.* Optimization and approximation methods that use both an ℓ_2-norm (or its square) and an ℓ_1-norm are currently very popular in statistics, machine learning, and signal and image processing. Examples include Huber estimation, LASSO, basis pursuit, SVM, various ℓ_1-regularized classification methods, total variation denoising, etc. Very roughly, an ℓ_2-norm corresponds to Euclidean distance (squared), or the negative log-likelihood function for a Gaussian; in contrast the ℓ_1-norm gives 'robust' approximation, *i.e.*, reduced sensitivity to outliers, and also tends to yield sparse solutions (of whatever the argument of the norm is). (All of this is just background; you don't need to know any of this to solve the problem.)

   In this problem we study a natural method for blending the two norms, by using the $\ell_{1.5}$-norm, defined as

$$\|z\|_{1.5} = \left(\sum_{i=1}^{k} |z_i|^{3/2}\right)^{2/3}$$

   for $z \in \mathbf{R}^k$. We will consider the simplest approximation or regression problem:

$$\text{minimize} \quad \|Ax - b\|_{1.5},$$

   with variable $x \in \mathbf{R}^n$, and problem data $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$. We will assume that $m > n$ and the A is full rank (*i.e.*, rank n). The hope is that this $\ell_{1.5}$-optimal approximation problem should share some of the good features of ℓ_2 and ℓ_1 approximation.

   (a) Give optimality conditions for this problem. Try to make these as simple as possible. Your solution should have the form 'x is optimal for the $\ell_{1.5}$-norm approximation problem if and only if ...'.

   (b) Explain how to formulate the $\ell_{1.5}$-norm approximation problem as an SDP. (Your SDP can include linear equality and inequality constraints.)

   (c) Solve the specific numerical instance generated by the following code:

```
randn('state',0);
A=randn(100,30);
b=randn(100,1);
```

   Numerically verify the optimality conditions. Give a histogram of the residuals, and repeat for the ℓ_2-norm and ℓ_1-norm approximations. You can use any method you like to solve the problem (but of course you must explain how you did it); in particular, you do not need to use the SDP formulation found in part (b).

8. *Three-way linear classification.* We are given data

$$x^{(1)}, \ldots, x^{(N)}, \quad y^{(1)}, \ldots, y^{(M)}, \quad z^{(1)}, \ldots, z^{(P)},$$

three nonempty sets of vectors in $\mathbf{R}^n$. We wish to find three affine functions on $\mathbf{R}^n$,

$$f_i(z) = a_i^T z - b_i, \quad i = 1, 2, 3,$$

that satisfy the following properties:

$$\begin{array}{l} f_1(x^{(j)}) > \max\{f_2(x^{(j)}), f_3(x^{(j)})\}, \quad j = 1, \ldots, N, \\ f_2(y^{(j)}) > \max\{f_1(y^{(j)}), f_3(y^{(j)})\}, \quad j = 1, \ldots, M, \\ f_3(z^{(j)}) > \max\{f_1(z^{(j)}), f_2(z^{(j)})\}, \quad j = 1, \ldots, P. \end{array}$$

In words: f_1 is the largest of the three functions on the x data points, f_2 is the largest of the three functions on the y data points, f_3 is the largest of the three functions on the z data points. We can give a simple geometric interpretation: The functions f_1, f_2, and f_3 partition $\mathbf{R}^n$ into three regions,

$$\begin{array}{rcl} R_1 & = & \{z \mid f_1(z) > \max\{f_2(z), f_3(z)\}\}, \\ R_2 & = & \{z \mid f_2(z) > \max\{f_1(z), f_3(z)\}\}, \\ R_3 & = & \{z \mid f_3(z) > \max\{f_1(z), f_2(z)\}\}, \end{array}$$

defined by where each function is the largest of the three. Our goal is to find functions with $x^{(j)} \in R_1$, $y^{(j)} \in R_2$, and $z^{(j)} \in R_3$.

Pose this as a convex optimization problem. You may not use strict inequalities in your formulation.