EE263 Prof. S. Lall
Nov 5-9

Midterm exam

This is a 24 hour take-home midterm. 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 (*e.g.*, matlab), but you may not discuss the exam with others until Nov. 10, after everyone has taken the exam. The only exception is that you can ask the TAs or Professor Lall for clarification, by emailing to the staff email address `ee263-aut1314-staff@lists.stanford.edu`. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

- Since you have 24 hours, we expect your solutions to be legible, neat, and clear. Do not hand in your rough notes, and please try to simplify your solutions as much as you can. We will deduct points from solutions that are technically correct, but much more complicated than they need to be.

- Please check your email a few times during the exam, just in case we need to send out a clarification or other announcement. It's unlikely we'll need to do this, but you never know.

- Attach the official exam cover page (available when you pick up or drop off the exam, or from the midterm info page) to your exam, and assemble your solutions to the problems in order, *i.e.*, problem 1, problem 2, ..., problem 6. Start each solution on a new page.

- Please make a copy of your exam before handing it in. We have never lost one, but it might occur.

- When a problem involves some computation (say, using matlab), we do not want just the final answers. We want a clear discussion and justification of exactly what you did, the matlab source code that produces the result, and the final numerical result. Be sure to show us your verification that your computed solution satisfies whatever properties it is supposed to, at least up to numerical precision. For example, if you compute a vector x that is supposed to satisfy $Ax = b$ (say), show us the matlab code that checks this, and the result. (This might be done by the matlab code `norm(A*x-b)`; be sure to show us the result, which should be very small.) *We will not check your numerical solutions for you, in cases where there is more than one solution.*

- In the portion of your solutions where you explain the mathematical approach, you *cannot* refer to matlab operators, such as the backslash operator. (You can, of course, refer to inverses of matrices, or any other standard mathematical construct.)

- Some of the problems are described in a practical setting, such as energy consumption, population dynamics, or wireless communications. *You do not need to understand anything about the application area to solve these problems.* We've taken special care to make sure all the information and math needed to solve the problem is given in the problem description.

- Some of the problems require you to download and run a matlab file to generate the data needed. These files can be found at the URL

  `http://www.stanford.edu/class/ee263/mterm13_mfiles/FILENAME`

  where you should substitute the particular filename (given in the problem) for `FILENAME`. *There are no links on the course web page pointing to these files, so you'll have to type in the whole URL yourself.*

1. *Minimum-energy job scheduling.* You are given n jobs, labeled $1, \ldots, n$, and must schedule these jobs on a processor over T time periods, labeled $1, \ldots, T$. You cannot work on job j before a given start period s_j, or after a given finish period f_j. Let w_j be the workload of job j, and $\theta_{tj} \in [0, 1]$ be the fraction of job j that is performed during time period t, where w_j is given, and you must choose θ_{tj}. The energy consumed by the processor during period t is

$$e_t = \left(\sum_{j=1}^{n} \theta_{tj} w_j \right)^2,$$

and the total energy consumed over all the periods is

$$E = \sum_{t=1}^{T} e_t.$$

Since you can only work on a job after its start time and before its finish time, we have the constraints

$$\theta_{tj} = 0, \qquad t < s_j \text{ or } t > f_j; \qquad j = 1, \ldots, n.$$

Additionally, each job must be fully completed by time T:

$$\sum_{t=1}^{T} \theta_{tj} = 1, \qquad j = 1, \ldots, n.$$

These two requirements are called the scheduling constraints, and a specification of θ_{tj} for $t = 1, \ldots, T$ and $j = 1, \ldots, n$ is called a job schedule. A specific instance of this problem is defined in the file `min_energy_job_sched_data.m`.

(a) For this part of the problem, you may ignore the restriction $\theta_{tj} \in [0, 1]$. Explain how to find a job schedule that minimizes the total energy consumed by the processor subject to the scheduling constraints. Apply your method to the instance of the problem defined in `min_energy_job_sched_data.m`. Report the optimal total energy, and use the function `plot_job_sched.m` to generate a plot of your job schedule. Is the optimal job schedule unique? Does your job schedule happen to satisfy $\theta_{tj} \in [0, 1]$?

Computational note. To solve the equation $Ax = b$ when the coefficient matrix A is rank deficient, you can use the command `x = pinv(A)*b`. If the system is consistent, this will return a solution. After computing x in this way, you should confirm that you have found a solution by checking that the residual norm $\|b - Ax\|$ is approximately equal to zero.

(b) In (a), we ignored the fact that a job schedule must also satisfy $\theta_{tj} \in [0, 1]$ for all $j = 1, \ldots, n$ and $t = 1, \ldots, T$. Unfortunately, the solution you found in (a) does

not satisfy these conditions. We can try to enforce the constraint $\theta_{tj} \in [0, 1]$ by minimizing the objective

$$J = E + \rho \sum_{j=1}^{n} \sum_{t=s_j}^{f_j} (\theta_{tj} - \tfrac{1}{2})^2,$$

where $\rho > 0$ is a penalty parameter. (The schedule must also satisfy the scheduling constraints from before.) For the problem data in `min_energy_job_sched_data.m`, find (to 2 decimal places) the smallest value of $\rho > 0$ such that $\theta_{tj} \in [0, 1]$ for all t and j. Report this value of ρ, and the corresponding optimal values of E and J; use the function `plot_job_sched.m` to generate a plot of your job schedule. Is this job schedule unique?

2. *Iteratively reweighted least squares for ℓ_1-norm approximation*

In an ordinary least squares problem, we are given $A \in \mathbf{R}^{m\times n}$ (skinny and full rank) and $y \in \mathbf{R}^m$, and we choose $x \in \mathbf{R}^n$ in order to minimize

$$\|Ax - y\|_2^2 = \sum_{i=1}^{m} (\tilde{a}_i^T x - y_i)^2.$$

Note that the penalty that we assign to a measurement error does not depend on the sensor from which the measurement was taken. However, this is not always the right thing to do: if we believe that one sensor is more accurate than another, we might want to assign a larger penalty to an error in the measurement from the more accurate sensor. We can account for differences in the accuracies of our sensors by assigning sensor i a weight $w_i > 0$, and then minimizing

$$\sum_{i=1}^{m} w_i (\tilde{a}_i^T x - y_i)^2.$$

By giving larger weights to more accurate sensors, we can account for differences in the precision of our sensors.

(a) *Weighted least squares.* Explain how to choose x in order to minimize

$$\sum_{i=1}^{m} w_i (\tilde{a}_i^T x - y_i)^2,$$

where the weights $w_1, \ldots, w_n > 0$ are given.

(b) *Iteratively reweighted least squares for ℓ_1-norm approximation.* Consider a cost function of the form

$$\sum_{i=1}^{m} w_i(x) (\tilde{a}_i^T x - y_i)^2. \tag{1}$$

One heuristic for minimizing a cost function of the form given in (1) is *iteratively reweighted least squares*, which works as follows. First, we choose an initial point $x^{(0)} \in \mathbf{R}^n$. Then, we generate a sequence of points $x^{(1)}, x^{(2)}, \ldots \in \mathbf{R}^n$ by choosing $x^{(k+1)}$ in order to minimize

$$\sum_{i=1}^{m} w_i(x^{(k)}) (\tilde{a}_i^T x^{(k+1)} - y_i)^2.$$

Each step of this algorithm involves updating our weights, and solving a weighted least squares problem. Suppose we want to use this method to solve minimize the ℓ_1-norm approximation error, which is defined to be

$$\|Ax - y\|_1 = \sum_{i=1}^{m} |\tilde{a}_i^T x - y_i|,$$

where the matrix $A \in \mathbf{R}^{m \times n}$ and the vector $y \in \mathbf{R}^m$ are given. How should we choose the weights $w_i(x)$ to make the cost function in (1) equal to the ℓ_1-norm approximation error?

(c) *Numerical example.* The file `l1_irwls_data.m` contains data $(t_1, y_1), \ldots, (t_m, y_m)$. We want to fit an affine model to this data:

$$y_i = x_1 + x_2 t_i, \qquad i = 1, \ldots, m.$$

Choose $x^{(0)}$ to be the vector of least-squares parameter estimates: that is, choose $x^{(0)}$ in order to minimize

$$\sum_{i=1}^{m} ((x_1^{(0)} + x_2^{(0)} t_i) - y_i)^2.$$

Generate $x^{(1)}, x^{(2)}, \ldots$ using iteratively reweighted least squares for ℓ_1-norm approximation. You can stop generating iterates when $\|x^{(k+1)} - x^{(k)}\| < 10^{-6}$. Report your values of $x^{(0)}$ and the final $x^{(k)}$ in your sequence of points. Draw a scatterplot of the data points (t_i, y_i). Add the fitted lines corresponding to $x^{(0)}$ and the final $x^{(k)}$ to your scatterplot. What do you observe?

Remark. Suppose we fit the least-squares line to some data. Then, a point that is very far from the least-squares line may be an *outlier*: that is, a point that does not seem to follow the same model as the rest of the data. Because such points may not follow the same model as the rest of data, it may make sense to give such points less weight. This idea is the intuition behind iteratively reweighted least squares for ℓ_1-norm approximation.

3. *Certificates of infeasibility.* Suppose I give you $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, and I ask you to solve $Ax = b$. If the equation has a solution, then you can give me such a solution $x \in \mathbf{R}^n$, and I can easily check that $Ax = b$. But if the equation does not have a solution, how can you convince me that there is no solution? Define the sets

$$S_1 = \{x \in \mathbf{R}^n : Ax = b\} \quad \text{and} \quad S_2 = \{y \in \mathbf{R}^m : A^T y = 0, b^T y = 1\}.$$

(a) Prove that if S_2 is nonempty, then S_1 must be empty. Thus, you can convince me that $Ax = b$ does not have a solution by giving me a $y \in \mathbf{R}^m$ such that $A^T y = 0$ and $b^T y = 1$. We call y a *certificate of infeasibility* for $Ax = b$, and we call S_1 and S_2 *weak alternatives.*

(b) Prove that if S_1 is empty, then S_2 must be nonempty. Thus, if $Ax = b$ does not have a solution, then you can always a find a certificate of infeasibility in S_2. We call S_1 and S_2 *strong alternatives.*

(c) Suppose $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $C \in \mathbf{R}^{p\times n}$ and $d \in \mathbf{R}^p$ are given. The normal equations for the problem of minimizing $\|Ax-b\|$ subject to the constraint $Cx = d$ are

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^T b \\ d \end{bmatrix}.$$

Prove that this system of equations has a solution if $Cx = d$ has a solution.

4. *Removing lens distortion.* You have a camera that produces distorted images, and you want to design a filter to remove this distortion. More precisely, let (x, y) denote the actual location of a point, and let $(\tilde{x}, \tilde{y})$ denote the location of that point in the image taken by your camera. Your camera suffers from radial distortion:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = d(\|(x, y)\|) \begin{bmatrix} x \\ y \end{bmatrix},$$

where $d : \mathbf{R} \to \mathbf{R}$ is an unknown distortion function. You want to design a filter $g : \mathbf{R} \to \mathbf{R}$ that removes this distortion:

$$\begin{bmatrix} x \\ y \end{bmatrix} \approx \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = g(\|(\tilde{x}, \tilde{y})\|) \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix},$$

where $(\hat{x}, \hat{y})$ is your estimate for the actual location of a point that appears at location $(\tilde{x}, \tilde{y})$ in an image taken by your camera. In particular, you want to find a filter of the form

$$g(r) = \alpha_1 r + \alpha_2 r^2 + \alpha_3 r^3 + \alpha_4 r^4,$$

where the $\alpha_i \in \mathbf{R}$ are parameters of the filter. In order to determine the filter parameters, you take pictures of N calibration targets at known locations $(x_1, y_1), \ldots, (x_N, y_N)$; the corresponding locations of these targets in the images produced by your camera are $(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_N, \tilde{y}_N)$.

(a) Explain how to choose the filter parameters in order to minimize the mean-squared error of the filtered image:

$$\frac{1}{N} \sum_{i=1}^{N} \|(x_i, y_i) - (\hat{x}_i, \hat{y}_i)\|^2.$$

State any assumptions that are needed for your method to work.

(b) The file `lens_distortion_data.m` defines the following variables.

- `target_locations`, a $2 \times N$ matrix whose ith column gives the actual location (x_i, y_i) of the ith calibration target
- `target_images`, a $2 \times N$ matrix whose ith column gives the location $(\tilde{x}_i, \tilde{y}_i)$ of the ith calibration target in the image produced by your camera
- `img`, an $m \times n \times 3$ array representing an image taken by your camera; `img(i,j,:)` is a vector giving the RGB values for the pixel that is ith from the left, and jth from the top
- `img_pixel_coords`, an $mn \times 2$ array whose ith row gives the location of the ith pixel in `img`; note that the location of the pixel that is ith from the left, and jth from the top is not position (i, j), but is measured in normalized units, so that the origin is in the center

Apply your method to this data. Report the filter parameters α_i. We provide you with the function `update_image.m`, which you can use to apply your filter to `img`. In particular, you can use the command

```
fimg = update_image(img, img_pixel_coords, fimg_pixel_coords)
```

to produce the filtered image, where `fimg_pixel_coords` is an $mn \times 2$ array whose ith row is your estimate for the actual location of the ith pixel in `img`. Attach a copy of the filtered image. You can look in the data file to see how to plot an image in `MATLAB`.

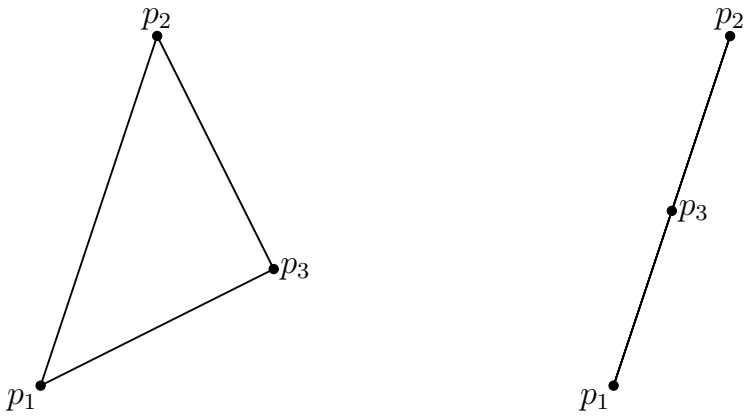
5. *Classifying polytopes.* Let $\mathcal{P} = \{p_1, \ldots, p_{n+1}\}$ be a set of $n+1$ points in $\mathbf{R}^n$. The polytope generated by $\mathcal{P}$ is the set

$$\Delta(\mathcal{P}) = \{\theta_1 p_1 + \cdots + \theta_{n+1} p_{n+1} \mid \theta_1 + \cdots + \theta_{n+1} = 1, \theta_1, \ldots, \theta_{n+1} \geq 0\}.$$

Recall that a hyperplane in $\mathbf{R}^n$ is a set of the form

$$\mathcal{H}(c, d) = \{x \in \mathbf{R}^n : c^T x + d = 0\},$$

where $c \in \mathbf{R}^n$ and $d \in \mathbf{R}$ are parameters, with $c \neq 0$. We say that $\Delta(\mathcal{P})$ is *degenerate* if it is contained in some hyperplane in $\mathbf{R}^n$. For example, suppose $n = 2$. Then, in the typical case, $\Delta(\mathcal{P})$ is a triangle, as shown below on the left. A degenerate case is shown on the right: $\Delta(\mathcal{P})$ is contained in a hyperplane in $\mathbf{R}^2$ (that is, a line).



These definitions are deliberately vague: an important part of the problem is making them precise. Suppose you are given a set $\mathcal{P} = \{p_1, \ldots, p_{n+1}\}$ of $n+1$ points in $\mathbf{R}^n$.

(a) Explain how to determine whether or not $\Delta(\mathcal{P})$ is degenerate.

(b) Suppose $\Delta(\mathcal{P})$ is *not* degenerate. Explain how to determine whether or not $\Delta(\mathcal{P})$ contains the origin.

(c) The file `classify_polytopes_data.m` defines a $2 \times 3 \times 50$ array `P`. For $k = 1, \ldots, 50$, `P(:,:,k)` is a 2×3 matrix specifying a set of 3 points in $\mathbf{R}^2$ (each column of the matrix is a different point). Thus, `P` describes 50 polytopes in $\mathbf{R}^2$. Give the indices of the degenerate polytopes, and the indices of the nondegenerate polytopes that contain the origin.

6. *Meta-analysis with unknown measurements.* Suppose k different research papers report the results of different experiments designed to measure a common parameter vector $x \in \mathbf{R}^n$. The ith paper gives the estimate $\hat{x}^{(i)} \in \mathbf{R}^n$, which was computed by minimizing $\|A^{(i)}\hat{x}^{(i)} - y^{(i)}\|$, where $A^{(i)} \in \mathbf{R}^{m_i \times n}$ is the measurement matrix, and $y^{(i)} \in \mathbf{R}^{m_i}$ is the vector of measurements. You want to combine the results of these various experiments to obtain another estimate of x. In particular, you choose your estimate $\hat{x} \in \mathbf{R}^n$ in order to minimize

$$\sum_{i=1}^{k} \|A^{(i)}\hat{x} - y^{(i)}\|^2.$$

This type of analysis is called meta-analysis because you are analyzing the results of other experiments, rather than running a new experiment. The catch in this problem is that paper i reports the measurement matrix $A^{(i)}$, and the corresponding estimate $\hat{x}^{(i)}$, but not the actual vector of measurements $y^{(i)}$.

(a) Explain how to use $A^{(1)}, \ldots, A^{(k)}$ and $\hat{x}^{(1)}, \ldots, \hat{x}^{(k)}$ to compute $\hat{x}$. State any assumptions that are needed for your method to work.

(b) Carry out your method on the specific instance of this problem defined in the file `meta_analysis_data.m`. This file defines a cell array `A`, each of whose entries is one of the $A^{(i)}$, and a cell array `x`, each of whose entries is one of the $\hat{x}^{(i)}$. Report your estimate $\hat{x}$.

7. *Some true-false questions.* Determine if the following statements are true or false. No justification or discussion is needed for your answers. What we mean by "true" is that the statement is true for all values of the matrices and vectors given. You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A + B = B + A$" is true, because no matter what the dimensions of A and B (which must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2 = A$ is false, because there are (square) matrices for which this doesn't hold. (There are also matrices for which it does hold, *e.g.*, an identity matrix. But that doesn't make the statement true.)

   (a) If $x^T Ax = x^T Bx$ for all x, then $A = B$.

   (b) If $x^T Ay = x^T By$ for all x and y, then $A = B$.

   (c) If $\|Ax\| = \|Bx\|$ for all x, then $A = B$.

   (d) If A and B are both stable, then $A + B$ is also stable.

   (e) The matrix $\begin{bmatrix} 2a & 3b \\ 4c & 5d \end{bmatrix}$ is equal to $A \begin{bmatrix} a & b \\ c & d \end{bmatrix} B$ for some matrices A and B.

   (f) If R is upper triangular and orthogonal, then R is diagonal.

   (g) If A is square, then there always exists a matrix C such that $AC = CA^T$.

   (h) If $x, y \in \mathbf{R}^n$ then the $n \times n$ matrix xy^T is diagonalizable.