

## 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 anyone until 6pm Oct. 26, after everyone has taken the exam. The only exception is that you can ask the TAs or Stephen Boyd for clarification, by emailing to the staff email address `ee263-aut0809-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 7. 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 signal processing, communications, machine learning, or control. *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.
- We *do not* expect you to be able to solve all parts of all problems, so don't worry if you cannot finish them all.
- 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/midterm_M-files/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.*

- Please respect the honor code. Although we encourage you to work on homework assignments in small groups, *you cannot discuss the midterm with anyone*, with the exception of Stephen Boyd and the TAs, until everyone has taken it.

1. *Element-wise nonnegative matrix and inverse.* Suppose a matrix  $A \in \mathbf{R}^{n \times n}$ , and its inverse  $B$ , have all their elements nonnegative, *i.e.*,  $A_{ij} \geq 0$ ,  $B_{ij} \geq 0$ , for  $i, j = 1, \dots, n$ . What can you say must be true of  $A$  and  $B$ ?

Please give your answer first, and then the justification. Your solution (which includes what you can say about  $A$  and  $B$ , as well as your justification) must be short; we won't read more than one page. We want the most specific description possible; no credit will be given for true but obvious statements (such as that  $A$  and  $B$  must be invertible).

2. *Least-squares classification.* For each of  $N$  documents we are given a feature vector  $x^{(i)} \in \mathbf{R}^n$ , and a label  $y_i \in \{-1, 1\}$ . (This is called a binary label.) Each component of the feature vector could be, for example, the number of occurrences of a certain term in the document; the label could be decided by a person working with the documents, with  $+1$  meaning the document is interesting or useful, and  $-1$  meaning the document is not (for example, spam). From this data set we construct  $w \in \mathbf{R}^n$  and  $v \in \mathbf{R}$  that minimize

$$\sum_{i=1}^N (w^T x^{(i)} + v - y_i)^2.$$

We can now use  $w$  and  $v$  to predict the label for other documents, *i.e.*, to guess whether an as-yet-unread document is interesting, by forming  $\hat{y} = \mathbf{sign}(w^T x + v)$ . For scalar  $a$ , we define  $\mathbf{sign}(a) = +1$  for  $a \geq 0$  and  $\mathbf{sign}(a) = -1$  for  $a < 0$ ; for vector arguments,  $\mathbf{sign}()$  is taken elementwise.

- (a) Explain (briefly) how to find  $w$  and  $v$ . If you need to make an assumption about the rank of a matrix, say so.
- (b) Find  $w$  and  $v$  for the data in `ls_classify_data.m`, which defines  $\mathbf{X}$ , whose columns are  $x^{(i)}$ , and  $\mathbf{y}$ . This M-file will also define a second data set,  $\mathbf{X}_{\text{test}}$  and  $\mathbf{y}_{\text{test}}$ , of the same size (*i.e.*,  $n$  and  $N$ ). Use the  $w$  and  $v$  you found to make predictions about whether the documents in the test set are interesting. Give the number of correct predictions (for which  $\hat{y}_i = y_i$ ), false positives ( $\hat{y}_i = +1$  while  $y_i = -1$ ), and false negatives ( $\hat{y}_i = -1$  while  $y_i = +1$ ) for the test set.

You may find the Matlab function `sign()` useful. To count false positives, for example, you can use `sum((yhat == 1) & (y == -1))`.

*Remark.* There are better methods for binary classification, which you can learn about in a modern statistics or machine learning course, or in EE364a. But least-squares classification can sometimes work well.

3. *Minimum time control.* We consider a discrete-time linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \quad t = 0, 1, \dots,$$

with  $x(t) \in \mathbf{R}^n$ ,  $u(t) \in \mathbf{R}^m$ .

- (a) You are given  $A$ ,  $B$ , and  $x(0) = x_{\text{init}}$ . Explain how to find an input sequence  $u(0), u(1), \dots, u(N-1)$ , so that  $x(N) = 0$ , with  $N$  is as small as possible. Your answer can involve any of the concepts used in the course so far, *e.g.*, range, rank, nullspace, least-squares,  $QR$  factorization, etc.
- (b) Apply the method described in part (a) to the specific problem instance with data

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad x_{\text{init}} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

(where  $n = 4$  and  $m = 1$ ). You must give us the (minimum) value of  $N$ , and a sequence of inputs  $u(0), \dots, u(N-1)$  that results in  $x(N) = 0$ .

4. *Interference cancelling equalizers.* Two vector signals  $x \in \mathbf{R}^p$  and  $y \in \mathbf{R}^q$  are to be transmitted to two receivers. The transmitter broadcasts the signal  $z = Ax + By \in \mathbf{R}^n$  to each receiver. (The matrices  $A$  and  $B$  are called the coding matrices, and are known.) Receiver 1 forms an estimate of the signal  $x$  using the linear decoder  $\hat{x} = Fz$ ; receiver 2 forms an estimate of the signal  $y$  using the linear decoder  $\hat{y} = Gz$ . (The matrices  $F \in \mathbf{R}^{p \times n}$  and  $G \in \mathbf{R}^{q \times n}$  are called the decoding matrices.)

The goal is to find  $F$  and  $G$  so that  $\hat{x} = x$  and  $\hat{y} = y$ , no matter what values  $x$  and  $y$  take. This means that both decoders are perfect; each reconstructs the exact desired signal, while completely rejecting the other (undesired) signal. For this reason we call decoding matrices with this property *perfect*.

- (a) When is it possible to find perfect decoding matrices  $F$  and  $G$ ? (The conditions, of course, depend on  $A$  and  $B$ .) Your answer can involve any of the concepts we've seen so far in EE263.
- (b) Suppose that  $A$  and  $B$  satisfy the conditions in part (a). How would you find perfect decoding matrices that, among all perfect decoding matrices, minimize

$$\sum_{i=1}^p \sum_{j=1}^n F_{ij}^2 + \sum_{i=1}^q \sum_{j=1}^n G_{ij}^2.$$

We call such decoding matrices minimum norm perfect decoding matrices.

- (c) Find minimum norm perfect decoding matrices for the data (*i.e.*,  $A$  and  $B$ ) given in the M-file `mn_perf_dec_data.m`.

5. *Piecewise affine fitting.* In this problem we refer to vectors in  $\mathbf{R}^N$  as signals. We say that a signal  $z$  is *piecewise-affine* (PWA), with kink points  $i_1, \dots, i_{K+1}$ , which are integers satisfying  $i_1 = 1 < i_2 < \dots < i_{K+1} = N + 1$ , if

$$z_j = \alpha_k j + \beta_k, \quad \text{for } i_k \leq j < i_{k+1},$$

for  $k = 1, \dots, K$ . Thus, the signal value is an affine function of the index  $j$  (which we might interpret as time in this problem) over the (integer) intervals

$$1, \dots, i_2 - 1; \quad i_2, \dots, i_3 - 1; \quad \dots \quad i_K, \dots, N.$$

We call  $\alpha_k$  and  $\beta_k$  the slope and offset, respectively, in the  $k$ th interval. (It is very common to refer to such a signal as piecewise-linear, since ‘linear’ is sometimes used to mean ‘affine’.)

We can also add a *continuity requirement*,

$$\alpha_k i_{k+1} + \beta_k = \alpha_{k+1} i_{k+1} + \beta_{k+1}, \quad k = 1, \dots, K - 1.$$

This means that if each piecewise affine segment were extrapolated to the next index, the value would agree with the starting value for the next segment. When a PWA signal satisfies this condition, we say that it is continuous. (Of course, it doesn’t make sense to refer to a discrete signal as continuous; this is just special notation for PWA signals that refers to the condition above.)

Finally, we get to the problem.

- (a) You are given a signal  $y \in \mathbf{R}^N$ , and some kink points  $i_1, \dots, i_{K+1}$ . How would one find the best PWA approximation  $\hat{y}^{\text{pwa}}$  of  $y$ , with approximation error measured in the RMS sense,

$$\left( \frac{1}{N} \sum_{j=1}^N (\hat{y}_j^{\text{pwa}} - y_j)^2 \right)^{1/2}.$$

- (b) Repeat part (a), but this time, you are to find the continuous PWA approximation  $\hat{y}^{\text{pwac}}$  that minimizes the RMS deviation from  $y$ .
- (c) Carry out your methods from parts (a) and (b) on the data given in `pwa_data.m`. Running this data file will define  $y$  and the kink points. The data file also includes a commented out code template for plotting your results. Using this template, plot the original signal along with the PWA and continuous PWA approximations. Give us the RMS approximation error in both cases.

6. *Robust input design.* We are given a system, which we know follows  $y = Ax$ , with  $A \in \mathbf{R}^{m \times n}$ . Our goal is to choose the input  $x \in \mathbf{R}^n$  so that  $y \approx y^{\text{des}}$ , where  $y^{\text{des}} \in \mathbf{R}^m$  is a given target outcome. We'll assume that  $m \leq n$ , *i.e.*, we have more degrees of freedom in our choice of input than specifications for the outcome. If we knew  $A$ , we could use standard EE263 methods to choose  $x$ . The catch here, though, is that we don't know  $A$  exactly; it varies a bit, say, day to day. But we do have some possible values of  $A$ ,

$$A^{(1)}, \dots, A^{(K)},$$

which might, for example, be obtained by measurements of  $A$  taken on different days. We now define  $y^{(i)} = A^{(i)}x$ , for  $i = 1, \dots, K$ . Our goal is to choose  $x$  so that  $y^{(i)} \approx y^{\text{des}}$ , for  $i = 1, \dots, K$ .

We will consider two different methods to choose  $x$ .

- *Least norm method.* Define  $\bar{A} = (1/K) \sum_{i=1}^K A^{(i)}$ . Choose  $x^{\text{ln}}$  to be the least-norm solution of  $\bar{A}x = y^{\text{des}}$ . (You can assume that  $\bar{A}$  is full rank.)
- *Mean-square error minimization method.* Choose  $x^{\text{mmse}}$  to minimize the mean-square error

$$\frac{1}{K} \sum_{i=1}^K \|y^{(i)} - y^{\text{des}}\|^2.$$

- (a) Give formulas for  $x^{\text{ln}}$  and  $x^{\text{mmse}}$ , in terms of  $y^{\text{des}}$  and  $A^{(1)}, \dots, A^{(K)}$ . You can make any needed rank assumptions about matrices that come up, but please state them explicitly.
- (b) Find  $x^{\text{ln}}$  and  $x^{\text{mmse}}$  for the problem with data given in `rob_inp_des_data.m`. Running this M-file will define `ydes` and the matrices  $A^{(i)}$  (given as a 3 dimensional array; for example, `A(:, :, 13)` is  $A^{(13)}$ ). Also included in the data file (commented out) is code to produce scatter plots of your results. Write down the values of  $x^{\text{ln}}$  and  $x^{\text{mmse}}$  you found. Produce and submit scatter plots of  $y^{(i)}$  for  $x^{\text{ln}}$  and  $x^{\text{mmse}}$ . Use the code we provided as a template for your plots.

7. *Unbiased estimation with deadlines.* We consider a standard measurement set up, with  $y = Ax + v$ , where  $y \in \mathbf{R}^m$  is the vector of measurements,  $v \in \mathbf{R}^m$  is the vector of measurement noises,  $x \in \mathbf{R}^n$  is the vector of parameters to be estimated, and  $A \in \mathbf{R}^{m \times n}$  characterizes the measurement system. In this problem, you should think of the index for  $y$  as denoting a time period, and you should imagine the measurements (*i.e.*, the components of  $y$ ) as arriving sequentially. In the first time period,  $y_1$  becomes available, in the next time period  $y_2$  becomes available, and so on, so that all  $m$  measurements are available in the  $m$ th time period.

You are to design a linear estimator, given by a matrix  $B \in \mathbf{R}^{n \times m}$ , with the estimate of  $x$  given by  $\hat{x} = By$ . We require that the estimator be unbiased, *i.e.*, that  $\hat{x} = x$  when  $v = 0$ .

In addition, we have *deadline constraints*, which we now explain. We require that  $\hat{x}_i$  can be computed after  $k_i$  time periods, *i.e.*, we require that  $\hat{x}_i$  must be a function of  $y_1, \dots, y_{k_i}$  only. We say that  $k_i$  is the *deadline* for computing  $\hat{x}_i$ , our estimate of the  $i$ th parameter to be estimated. You are given increasing deadlines,

$$0 < k_1 < k_2 < \dots < k_n = m.$$

Thus,  $\hat{x}_1$  may only be computed from  $y_1, \dots, y_{k_1}$ , while  $\hat{x}_n$  may be computed from all of the measurements  $y_1, \dots, y_m$ .

The data in this problem are the measurement matrix  $A$  and the deadlines  $k_1, \dots, k_n$ .

- (a) How would you determine whether or not an unbiased linear estimator, which respects the given deadlines, exists? Your answer does not have to be a single condition, such as ‘ $A$  is skinny and full rank’; it can involve a sequence of tests.
- (b) Assume that it is possible to find an unbiased linear estimator that respects the deadlines. Explain how to find the smallest such estimator matrix, *i.e.*, the  $B$  that minimizes

$$J = \sum_{i=1}^m \sum_{j=1}^n B_{ij}^2.$$

If your method requires some matrix or matrices to be full rank, you can just assume they are, but you must state this clearly.

- (c) Carry out the method described in part (b) on the data found in the M-file `unbdl_data.m`. Compare the value of  $J$  found for your estimator with the value of  $J$  for  $B = A^\dagger$ . The increase in  $J$  can be thought of as the cost of imposing the deadlines, in terms of the size of estimator matrix.