EE263 Homework 9

Fall 2025

due Thursday 12/4, at 11:59 PM

15.2170. Positive semidefinite (PSD) matrices.

a) Show that if A and B are PSD and $\alpha \in \mathbb{R}$, $\alpha \geq 0$, then so are αA and $A + B$.

b) Show that any (symmetric) submatrix of a PSD matrix is PSD. (To form a symmetric submatrix, choose any subset of $\{1, \ldots, n\}$ and then throw away all other columns and rows.)

c) Show that if $A \geq 0$, $A_{ii} \geq 0$.

d) Show that if A is (symmetric) PSD, then $|A_{ij}| \leq \sqrt{A_{ii}A_{jj}}$. In particular, if $A_{ii} = 0$, then the entire ith row and column of A are zero.

15.2320. Approximate left inverse with norm constraints. Suppose $A \in \mathbb{R}^{m \times n}$ is full rank with $m \geq n$. We seek a matrix $F \in \mathbb{R}^{n \times m}$ that minimizes $\|I - FA\|$ subject to the constraint $\|F\| \leq \alpha$, where $\alpha > 0$ is given. Note that $\|I - FA\|$ gives a measure of how much F fails to be a left inverse of A. Give an explicit description of an optimal F. Your description can involve standard matrix operations and decompositions (eigenvector/eigenvalue, QR, SVD, ...).

16.2710. The EE263 search engine. In this problem we examine how linear algebra and low-rank approximations can be used to find matches to a search query in a set of documents. Let's assume we have four documents: **A**, **B**, **C**, and **D**. We want to search these documents for three terms: *piano*, *violin*, and *drum*. We know that:
in **A**, the word *piano* appears 4 times, *violin* 3 times, and *drum* 1 time;
in **B**, the word *piano* appears 6 times, *violin* 1 time, and *drum* 0 times;
in **C**, the word *piano* appears 7 time, *violin* 4 times, and *drum* 39 times; and
in **D**, the word *piano* appears 0 times, *violin* 0 times, and *drum* 5 times.
We can tabulate this as follows:

	A	**B**	**C**	**D**
piano	4	6	7	0
violin	3	1	4	0
drum	1	0	39	5

This information is used to form a *term-by-document* matrix A, where A_{ij} specifies the frequency of the ith term in the jth document, *i.e.*,

$$A = \begin{bmatrix} 4 & 6 & 7 & 0 \\ 3 & 1 & 4 & 0 \\ 1 & 0 & 39 & 5 \end{bmatrix}.$$

Now let q be a *query vector*, with a non-zero entry for each term. The query vector expresses a criterion by which to select a document. Typically, q will have 1 in the entries corresponding to the words we want to search for, and 0 in all other entries (but other weighting schemes are

possible.) A simple measure of how relevant document j is to the query is given by the inner product of the jth column of A with q:

$$a_j^\mathsf{T} q.$$

However, this criterion is biased towards large documents. For instance, a query for *piano* ($q = [1\ 0\ 0]^\mathsf{T}$) by this criterion would return document **C** as most relevant, even though document **B** (and even **A**) is probably much more relevant. For this reason, we use the inner product normalized by the norm of the vectors,

$$\frac{a_j^\mathsf{T} q}{\|a_j\|\,\|q\|}.$$

Note that our criterion for measuring how well a document matches the query is now the cosine of the angle between the document and query vectors. Since all entries are non-negative, the cosine is in $[0, 1]$ (and the angle is in $[-\pi/2, \pi/2]$.) Define $\tilde{A}$ and $\tilde{q}$ as normalized versions of A and q (A is normalized column-wise, *i.e.*, each column is divided by its norm.) Then,

$$c = \tilde{A}^\mathsf{T} \tilde{q}$$

is a column vector that gives a measure of the relevance of each document to the query. And now, the question. In the file `term_by_doc.json` you are given m search terms, n documents, and the corresponding term-by-document matrix $A \in \mathbb{R}^{m \times n}$. (They were obtained randomly from Stanford's *Portfolio* collection of internal documents from the 1990s.) The variables `term` and `document` are lists of strings. The string `term[i]` contains the ith word. Each document is specified by its former URL, *i.e.*, the jth document used to be at the URL `document[j]`; the documents are no longer available online, but you might be able to find some of them on some internet archive (like the Wayback Machine) if you're curious. (You don't need to in order to solve the problem.) The matrix entry `A[i,j]` specifies how many times term i appears in document j.

When you specify documents in your results, please just specify the indices, that is, give us `j` rather than `document[j]`.

a) Compute $\tilde{A}$, the normalized term-by-document matrix. Compute and plot the singular values of $\tilde{A}$.

b) Perform a query for the word *students* ($i = 53$) on $\tilde{A}$. What are the 5 top results?

c) We will now consider low-rank approximations of $\tilde{A}$, that is

$$\hat{A}_r = \min_{\hat{A},\, \mathbf{rank}(\hat{A}) \leq r} \|\tilde{A} - \hat{A}\|.$$

Compute $\hat{A}_{32}$, $\hat{A}_{16}$, $\hat{A}_8$, and $\hat{A}_4$. Perform a query for the word *students* on these matrices. Comment on the results.

d) Are there advantages of using low-rank approximations over using the full-rank matrix? (You can assume that a very large number of searches will be performed before the term-by-document matrix is updated.)

Note: Variations and extensions of this idea are actually used in commercial search engines (although the details are closely guarded secrets . . .) Issues in real search engines include the fact that m and n are enormous and change with time. These methods are very interesting because they can recover documents that don't include the term searched for. For example, a search for *automobile* could retrieve a document with no mention of *automobile*, but many references to cars (can you give a justification for this?) For this reason, this approach is sometimes called *latent semantic indexing*.

Julia hints: You may find the command `sortperm` useful. It returns the index permutation that would sort its argument into an ascending order, or if the `rev=true` argument is supplied, into a descending order. Here's some sample code that prints the indices of the two largest elements of a vector `c`:

```
p = sortperm(c, rev=true)
@show p[1:2]                # indices of two largest elements
c_sorted = c[p]             # equivalent of sort(c, rev=true)
@show c[p[1:2]]             # two largest elements of c
```

17.1340. Navigation. Suppose a ship is located at position $x \in \mathbb{R}^2$. We measure distances to beacons located at

$$q_1 = \begin{bmatrix} 500 \\ 0 \end{bmatrix} \qquad q_2 = \begin{bmatrix} 0 \\ 500 \end{bmatrix} \qquad q_3 = \begin{bmatrix} 500 \\ -200 \end{bmatrix}$$

As usual we model this as $y = Ax + w$, where $y \in \mathbb{R}^3$ is a vector of measurements (with appropriate constants to make the relationship linear) such that $y_i = \frac{1}{\|q_i\|} q_i^\top x$ and $w \in \mathbb{R}^3$ is Gaussian noise, $w \sim \mathcal{N}(0, \Sigma_w)$. We also have prior information $x \sim \mathcal{N}(\mu_x, \Sigma_x)$. Here

$$\mu_x = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \qquad \Sigma_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \Sigma_w = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a) The MMSE estimate has the form

$$\phi_{\text{mmse}}(y) = \mu_x + L_{\text{mmse}}(y - A\mu_x)$$

Find the estimator gain L_{mmse} and compute it using Julia.

b) Perform the experiment in Julia; i.e., simulate x and w with the above distributions and compute y. Using this y, plot the 90% confidence ellipsoid for x conditioned on your measurement, centered on your estimate.

Hint: In Julia, you can sample from a multivariate normal distribution with the code

```
using Distributions
mu = zeros(2)
sigma = I(2)
dist = MvNormal(mu, sigma)
sample = rand(dist)
```

c) Now collect data, consisting of 500 samples of $y = Ax + w$. For each measurement y, compute the corresponding estimate $\phi_{\text{mmse}}(y)$, and plot the error in $\mathbb{R}^2$ given by $x - \phi_{\text{mmse}}(y)$. Also plot the 90% confidence ellipsoid for the error (centered at the origin.)

d) Suppose instead we were to use the least squares estimator

$$\phi_{\text{ls}}(y) = A^{\dagger} y$$

What is the mean square error

$$\mathrm{E}\|\phi_{\text{ls}}(y) - x\|^2$$

achieved by this estimator?

e) Using the same data as above, plot the error in $\mathbb{R}^2$ given by $x - \phi_{\text{ls}}(y)$. Also plot the 90% confidence ellipsoid for the error, centered at the origin.

f) Is it true that the ellipsoid of part (c) is always contained in the ellipsoid of part (e)? Prove or give a counterexample.

17.1360. MMSE with correlated noise. Suppose

$$y = Ax + w$$

and x and w are Gaussian, with

$$\mathrm{E}\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} \mu_x \\ \mu_w \end{bmatrix} \qquad \operatorname{cov}\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} \Sigma_x & \Sigma_{xw} \\ \Sigma_{wx} & \Sigma_w \end{bmatrix}$$

In particular, x and w are *correlated*. Find the minimum mean-square-error estimate of x given a measurement of y.

17.1365. Orthogonality. Suppose x, y are jointly Gaussian random variables. Let $\phi(y_{\text{meas}})$ be the MMSE estimate of x given $y = y_{\text{meas}}$. Let z be the estimation error

$$z = \phi(y) - x$$

a) Show that

$$\mathrm{E}\, zy^T = 0$$

b) This is called the *orthogonality principle* in estimation. It says that the estimation error is *uncorrelated* with the measurement y. Explain why you would expect this to be the case.