University of Toronto
Department of Electrical and Computer Engineering

ECE367 MATRIX ALGEBRA AND OPTIMIZATION

Problem Set #1

Fall 2023

Prof. Wei Yu **Due:** 11:59pm (Toronto time) September 27, 2023

Homework policy: Problem sets must be turned by the due date and time. Late problem sets will not be accepted. See the information sheet for further details. The course text "Optimization Models" is abbreviated as "OptM" and "Introduction to Applied Linear Algebra" as "IALA".

Problems are categorized as:

- **"Theory" problems:** These are mostly mathematical questions designed to give you deeper insight into the fundamentals of the ideas introduced in this class.
- **"Application" problems:** These questions are designed to expose you to the breadth of application of the ideas developed in class and to introduce you to useful numerical toolboxes. Problems of this sort often ask you to produce plots and discuss your results; said plots and discussions should be included in and form part of your submission – think of your submitted solution like a lab book. Your attached code simply provides back-up evidence.
- **"Optional" problems:** Optional problems provide extra practice or introduce interesting connections or extensions. They need not be turned in. I will assume you have reviewed and understood the solutions to the optional problems when designing the exams.

Hand-in procedure:

- **Initial submission:** Your initial submission of the "Theory" and "Application" questions must be submitted via Quercus upload by the due date.
- **Self-assessment:** After the problem set is due we will post solutions. You will have one week from the initial due date to submit a commented version of your assignment in which, using as a reference the posted solutions, you highlight your errors or omissions in red. Please annotate the PDF you initially submitted. If you have not submitted a solution you cannot submit the self-assessment.
- **Late problem sets are not accepted**
- **Grading:** Per the course handout problem sets are graded for completion only. Points are assigned to (i) Initial submission of theory part, (ii) Initial submission of application part, (iii) Self-assessment.

These problem sets were initially developed by Prof. Stark Draper.

Points allocation

- Theory parts (initial submission): 1 pt
- Application parts (initial submission): 1 pt
- Self-assessment: 1 pt

Problem categorization and main concepts covered

Theory

- Norms: Problems 1.1, 1.2
- Linear independence and orthogonality: Problems 1.3, 1.4
- Inner products: Problems 1.5, 1.6

Application

- Inner products: Problem 1.7
- Projection onto Fourier Basis: Problem 1.8

Optional

- Projection onto Polynomial Basis: Problem 1.9

Final note and recommendation: If it feels it has been a long time since you took linear algebra, and you are looking for additional resources and practice problems, I direct you to "Introduction to linear algebra" by Gilbert Strang, Wellesley-Cambridge Press. This classic text is the text I used to learn linear algebra. Prof. Strang's website has lots of online resources, including old exams. Please follow the following link: link

THEORY PROBLEMS

Problem 1.1 (Showing ℓ_1 and ℓ_∞ are both norms)
Show

(a) that the functions $\ell_1 : \mathbb{R}^n \to \mathbb{R}$ is a norm, and

(b) that $\ell_\infty : \mathbb{R}^n \to \mathbb{R}$ is a norm

by verifying that they satisfy all the properties of a norm, cf., OptM Definition 2.1.

Problem 1.2 (Norm inequalities)
OptM Prob. 2.6.
Note: This problem shows that each norm (l_1, l_2, l_∞) is both upper- and lower-bounded by each of the other norms to within a constant (dimension-dependent) factor.

Problem 1.3 (Linear independence of stacked vectors)
IALA Prob. 5.1.

Problem 1.4 (Orthogonality)
OptM Prob. 2.5.

Problem 1.5 (Inner products)
OptM Prob. 2.4.

Problem 1.6 (Distance versus angle nearest neighbors)
IALA Problem 3.24.

APPLICATION PROBLEMS

Problem 1.7 (Angles between word vectors)
In this problem you investigate how geometric concepts such as distance and angle can be applied to quantify similarity between text documents. Download the files `wordVecArticles.txt`, `wordVecTitles.txt`, `wordVecWords.txt` and `wordVecV.mat` from the course website. The first two files each have ten lines. Each line in the first file consists of the text of one Wikipedia article. The corresponding line of the second file is the title of the article. The last two files are described in detail below.

Denote by $\mathcal{D}$ the set of documents where the number of documents is $|\mathcal{D}|$. (In our dataset $|\mathcal{D}| = 10$). Let $\mathcal{W}$ denote the union of words in all articles, i.e., the lexicon of the set of documents. We denote the cardinality of $\mathcal{W}$ by $|\mathcal{W}|$. Assume the lexicon is ordered "lexiographically" (e.g., alphabetically) so that there is a one-to-one mapping from each word $w \in \mathcal{W}$ to an element of the index set $t \in [|\mathcal{W}|]$. Let $f_{\text{term}}(t, d)$ denote the number of times the word $w \in \mathcal{W}$ that is indexed as $t \in [|\mathcal{W}|]$ appears in the dth article where $d \in [|\mathcal{D}|]$. Note that $\sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t, d)$ is the number of words (the length) of the dth article. We refer to $f_{\text{term}}(t, d)$ as the *term frequency* (really "term count").

For the first few parts of this problem you will be using a pre-processed $\mathcal{W}$ set and pre-computed $f_{\text{term}}(t, d)$ values. The pre-processed data appears in the files `wordVecWords.txt` and `wordVecV.mat`. The first file represents the set $\mathcal{W}$ where elements of $\mathcal{W}$ are listed line by line, for 1651 lines, i.e., $|\mathcal{W}| = 1651$. You can load the content in the second file into MATLAB by using command `load 'wordVecV.mat'`. After loading, you will see a matrix `V` of dimensions 1651×10. The value in the tth row and dth column of this matrix is $f_{\text{term}}(t, d)$. Use the provided data in `V` to answer parts (a) to (d) of this problem.

(a) Let the $|\mathcal{W}|$-dimensional vectors v_d, $d \in [|\mathcal{D}|]$ be defined as $v_d = (f_{\text{term}}(1, d), f_{\text{term}}(2, d), \ldots, f_{\text{term}}(|\mathcal{W}|, d))$. Using v_d to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)? Which two are closest in angle distance (smallest angle)? Are they the same pair, if not, what could be a reason for them being different? You may find the `pdist` command in MATLAB useful for computing pairwise Euclidean and angle distances of vectors.

(b) In this part let the $|\mathcal{W}|$-dimensional *normalized* vectors $\tilde{v}_d$, $d \in [|\mathcal{D}|]$ be defined as $\tilde{v}_d = v_d / \sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t, d)$, where the v_d are defined as in the previous part. Using $\tilde{v}_d$ to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)? Which two are closest in angle distance (smallest angle)? Are your answers the same as in the previous part? What would be a reason for using this normalization?

Now, let $f_{\text{doc}}(t) = \sum_{d=1}^{|\mathcal{D}|} \mathbb{I}[f_{\text{term}}(t, d) > 0]$ where $\mathbb{I}(\cdot)$ is the indicator function taking value one if the clause is true and zero else. The function $f_{\text{doc}}(t)$ counts in how many documents the tth word appears. We refer to $f_{\text{doc}}(t)$ as the *document frequency*.

We combine the term and document frequency definitions into what is called the *term frequency-inverse document frequency score* (TF-IDF), defined as

$$w(t,d) = \frac{f_{\text{term}}(t,d)}{\sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t,d)} \sqrt{\log\left(\frac{|\mathcal{D}|}{f_{\text{doc}}(t)}\right)}.$$

Note, the denominator of the log is never zero since, by definition, each term appears in at least one document.

(c) Now let the $|\mathcal{W}|$-dimensional vectors w_d, $d \in [|\mathcal{D}|]$ be defined as $w_d = (w(1,d), w(2,d), \ldots, w(|\mathcal{W}|,d))$. Using w_d to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)?

(d) What might be a reason for using the "inverse document frequency" adjustment? What is the adjustment doing geometrically?

OPTIONAL PART: The following part (e) is a *optional* part.

(e) In the previous parts of the problem you used the pre-processed $\mathcal{W}$ set and pre-computed $f_{\text{term}}(t,d)$ values provided in `wordVecV.mat` and term indexes in `wordVecWords.txt`. In this part of the problem you will reproduce your results from (a) to (d) without using this pre-computed data. Specifically, start from the raw text files `wordVecArticles.txt` and `wordVecTitles.txt`, and write code to obtain $\mathcal{W}$ and $f_{\text{term}}(t,d)$. As always, you are welcome to use whichever software language you wish to solve this problem. We note that Python is particularly well suited to text processing. For example, you may find the snippet of Python code following the problem statement useful. This snippet loads in data from the given text file and stores it in the variable `articles`. It also counts the number of word occurrences in the first article and stores the resulting (word, count) pairs in the variable `wordcounts`. You could use this as a starting point in the generation of the vectors we provide in `wordVecV.mat` and then calculate angles and distances in MATLAB. Alternately, you could show how to complete the entire processing pipeline in Python. Be sure to include a printout of your code.

```
from collections import Counter
articles = [line.rstrip('\n') for line in open('wordVecArticles.txt')]
wordcounts = Counter(articles[0].split())
```

Problem 1.8 (Approximating a square wave: The discrete-time Fourier series)
In this problem you will approximate a given discretized square wave signal using a *sinusoidal basis*. The following MATLAB function produces the discrete approximation of two periods of a period-10 square wave signal $x(t)$ where $t \in [0, 20]$, discretized in increments of 0.0001 seconds. In addition, the script also produces 30 orthogonal basis vectors that we will later use to approximate the given signal. The function returns two vectors `time_pos` and `sq_wave`, and a matrix `B_unnorm`.

The vector `sq_wave` consists of the signal output for each time position specified in `time_pos` vector. The matrix `B_unnorm` includes 30 unnormalized row vectors that are of the same length as `sq_wave`.

```
function [time_pos, sq_wave, B_unnorm] = generate_data
    n_comps = 30; period = 10; fundFreq = 1/period;
    time_pos = 0:0.0001:2*period; harmonics = 2*(1:n_comps)-1;
    sq_wave = floor(0.9*sin(2*pi*fundFreq*time_pos))+.5;   %% generate the signal
    B_unnorm = sin(2*pi*fundFreq*(harmonics'*time_pos))/2; %% generate the basis
end
```

(a) Plot the square wave signal `sq_wave` against `time_pos`.

(b) How would you numerically test for the orthogonality of the basis vectors (rows of `B_unnorm`)? Plot the first 6 and 30th basis vectors against `time_pos` in separate sub-plots that share the same time axis.

(c) Normalize the given basis vectors to obtain an orthonormal basis. Project the square wave signal onto the normalized basis vectors using ℓ_2 projection and compute the projection coefficients. Arrange the basis vectors in the decreasing order of the magnitude of the projection coefficients. Approximate the square wave using the first $1, 2, 3, 4, 5, 6, 30$ basis vectors, and plot the approximations in separate sub-plots that share the same time axis. Plot the original signal on top of each approximation and compare.

OPTIONAL PROBLEMS

Problem 1.9 (Inner products and projection in function spaces)
(Note: This problem can be solved analytically in closed-form or numerically. The former is significantly more work. Both methods should yield the same final insight. Some tips about numerical methods are given following the problem statement.)

While the focus of this course is on finite (and mostly real) vector spaces, notions of inner products spaces and the approximation of a vector by its projection into a subspace, also hold for infinite-dimensional vector spaces where each vector is a function $f : \mathbb{R} \to \mathbb{R}$. In this problem let $C[-\pi, \pi]$ denote the set of continuous real-valued functions on $[-\pi, \pi]$. Observe that one can add and scale functions pointwise, thereby defining $C[-\pi, \pi]$ to be a vector space. We pick

$$\langle f, g \rangle = \int_{-\pi}^{\pi} f(x)g(x)dx \tag{1}$$

to be the inner product of two functions $f, g \in C[-\pi, \pi]$. One can verify that all properties of an inner product are satisfied by (1). The norm induced by this inner product is

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\int_{-\pi}^{\pi} (f(x))^2 dx}.$$

In this problem you consider the the function $g : [-\pi, \pi] \to \mathbb{R}$ defined pointwise as $g(x) = \sin(x)$. Your task is to find the best approximation to g in the subspace of $C[-\pi, \pi]$ that consists of polynomials with real coefficients and degree at most five. We denote this subspace as $\mathcal{U} = \{u | u(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \alpha_4 x^4 + \alpha_5 x^5, \alpha_i \in \mathbb{R} \ \forall \ i \in \{0, 1, \dots 5\}\}$. By best we mean that the optimizing u^* is

$$u^* = \arg\min_{u \in \mathcal{U}} \|g - u\|^2.$$

where, to be explicit,

$$\|g - u\|^2 = \langle g - u, g - u \rangle = \int_{-\pi}^{\pi} (g(x) - u(x))^2 dx = \int_{-\pi}^{\pi} (\sin(x) - u(x))^2 dx.$$

In the following you are asked to apply the Gram-Schmidt procedure to produce an orthogonal basis. You are welcome to do this either numerically or analytically. By "numerically" we mean you are welcome to use a numerical integration technique to compute the integrations involved in the inner products. The integrations can be computed analytically to produce closed-form solutions, but it is a non-trivial amount of work. However you chose to do it, please do the following.

(a) First apply the Gram-Schmidt procedure using the inner production defined in (1) to the basis $\{v^{(0)}, v^{(1)}, v^{(2)}, v^{(3)}, v^{(4)}, v^{(5)}\}$ of $\mathcal{U}$ where, defined pointwise, $v^{(i)}(x) = x^i$, $i \in \{0, 1, 2, 3, 4, 5\}$ to produce an orthonormal basis $\{e^{(0)}, e^{(1)}, e^{(2)}, e^{(3)}, e^{(4)}, e^{(5)}\}$ where each $e^{(i)} \in C[-\pi, \pi]$.

(b) Next, using (1) and the formulas for ℓ_2 projection we developed in class (which apply again here), compute $\alpha_0, \dots, \alpha_5$. (To check correctness note that, up to numerical precision, you should find that $\alpha_1 = 0.987862$.)

(c) You should see a sensical pattern to the even coefficients $\alpha_0, \alpha_2, \alpha_4$. What is the pattern you observe and why is it intuitively correct?

(d) Another often used polynomial approximation to the $\sin x$ function the Taylor approximation

$$\sin x \simeq x - \frac{x^3}{3!} + \frac{x^5}{5!}.$$

Plot your approximation from part (b) against the Taylor approximation. You should observe that your approximation looks much better over the entire interval $[-\pi, \pi]$. Where is the Taylor approximation accurate and where is it not accurate? What was it about the formulation of the ℓ_2 projection problem that makes your approximation better in the regions where the Taylor approximation is not good?

While you are welcome to use any method to perform the integration in 1, we describe below two numerical methods that you may use. The following MATLAB code snippet defines a few *anonymous functions* that help us compute the inner product of two functions. We use anonymous functions since they do not have to be defined in separate `.m` files. Please read MATLAB documentation for further information about this type of functions. Executing the following piece of code will print to console $\langle \sin, v^{(3)} \rangle$ and $\| \sin \|$. The purpose of each line of code is explained in the comments.

```
%% divide the interval [-pi, pi] into 1000 small intervals
eps = 1000;
xi = -pi:pi/eps:pi;
dx = pi/eps;

%% utility functions
intgrt = @(f,g) dot(f(xi),g(xi))*dx; % discrete approx. of the continous integral
inprod = @(f,g) intgrt(f,g);    % compute inner product between f and g
normf = @(f) sqrt(inprod(f,f)); % compute the norm of f

%% test
sx = @(x) sin(x);
v3 = @(x) x.^3;

inprod(sx,v3)                   % prints the inner product <sin(x), x^3>
normf(sx)                       % prints the norm of sin(x)
```

Alternatively, you may use MATLAB symbolic functions to implement $g, v^{(0)}, \ldots, v^{(5)}$, and use the built-in `int` command to perform the integration.