EECS 127/227AT Optimization Models in Engineering
Spring 2019 Homework 2

Release date: 9/12/19
Due date: 9/19/19, 23:00 (11 pm)

Please LaTeX or handwrite your homework solution and submit an electronic version.

Submission Format
Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

1. **Vectors and Orthogonality**

   Let $\{v_1, ..., v_n\}$ be an orthonormal basis in $\mathbb{R}^n$. Prove for all $x \in \mathbb{R}^n$ that

$$\sum_{i=1}^{n} |\langle x, v_i \rangle|^2 = \langle x, x \rangle.$$

2. Image Representation

Please refer to the corresponding Jupyter notebook: “image_representation.ipynb”

3. Representation of a graph as a matrix: the adjacency matrix

In this exercise, we are interpreting a graph as a matrix. Then, we show that finding a path between two nodes can be done with a matrix multiplication. This can be useful to quickly compute several shortest paths, or to re-compute shortest paths when the weights on the links of the graph only change slightly.

We are given a graph as a set of vertices $V = \{1, ..., n\}$, with edges $E \subseteq V \times V$. We assume that the graph is undirected (without arrows), meaning that $(i, j) \in E$ implies $(j, i) \in E$.

We define the adjacency matrix of the graph A by:

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

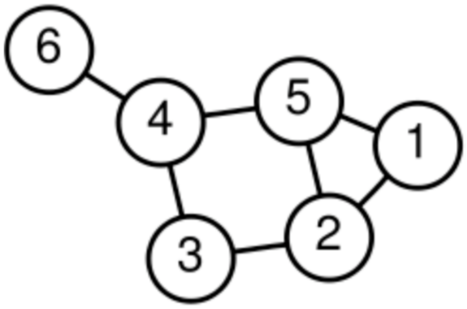

Figure 1: Example of an undirected graph.

(a) Form the adjacency matrix for the graph shown in Figure 1.

(b) Turning to a generic graph, show that the adjacency matrix A is symmetric.

Now let's show that if there is a path between $i \in V$ and $j \in V$, then there exists $n \in \mathbb{N}$ such that $e_i^\top A^n e_j \neq 0$. We define e_i as the vector of length n where the ith entry is 1 and all other entries are 0.

Let's proceed by induction. Define the following property:

$$P(n): \text{ there is a path of length } n \text{ between } i \text{ and } j \implies e_i^\top A^n e_j \neq 0 \tag{1}$$

And show that $P(1)$ is true and that $\forall n > 1,\ P(n-1) \implies P(n)$. This will show that $\forall n \geq 1$, $P(n)$ is true.

To get some intuition about the exercise that you will now solve, it is strongly recommended to run the iPython notebook "Adjacency matrix.ipynb". You do **NOT** need to submit the iPython notebook to gradescope.

Let assume that there is a path – denoted p – between i and j. We denote the length of the path p as n.

(c) Show that if $n = 1$ – meaning that there is a path of length 1 between i and j – then $e_i^\top A^n e_j \neq 0$. State what is the value of $e_i^\top A^n e_j$.

Now let's assume that the property is true for $n - 1$, with $n > 1$. Let show that it is true for n.

(d) Show that $e_i^\top A^n e_j = \sum_k (e_i^\top A e_k)(e_k^\top A^{n-1} e_j)$. This equation means that any path of length n between i and j is the combination of a link between i and a vertices k and a path of length $n-1$ from k to j.

(e) Show that $\forall k, (e_i^\top A e_k)(e_k^\top A^{n-1} e_j) \geq 0$.

Let's denote l the first node achieved in the path p from i.

(f) Explain why $e_l^\top A^{n-1} e_j \neq 0$.

(g) State the value of $e_i^\top A e_l$.

(h) Conclude that $e_i^\top A^n e_j \neq 0$.

(i) Have you shown that there is a path between i and j if and only if $e_i^\top A^n e_j \neq 0$?

If you enjoyed this exercise, feel free to read:

- Algebraic Graph Theory, C.Godsil, G.Royle
- Algebraic Graph Algorithms, P.Sankowski

4. Gram-Schmidt

Any set of n linearly independent vectors in $\mathbb{R}^n$ could be used as a basis for $\mathbb{R}^n$. However, certain bases could be more suitable for certain operations than others. For example, an orthonormal basis could facilitate solving linear equations.

(a) Given a matrix $A \in \mathbb{R}^{n \times n}$, it could be represented as a multiplication of two matrices

$$A = QR,$$

where Q is a unitary matrix (its columns form an orthonormal basis for $\mathbb{R}^n$) and R is an upper-triangular matrix. For the matrix A, describe how Gram-Schmidt process could be used to find the Q and R matrices, and apply this to

$$A = \begin{bmatrix} 3 & -3 & 1 \\ 4 & -4 & -7 \\ 0 & 3 & 3 \end{bmatrix}$$

to find a unitary matrix Q and an upper-triangular matrix R.

(b) Given an invertible matrix $A \in \mathbb{R}^{n \times n}$ and an observation vector $b \in \mathbb{R}^n$, the solution to the equality

$$Ax = b$$

is given as $x = A^{-1}b$. For the matrix $A = QR$ from part (a), assume that we want to solve

$$Ax = \begin{bmatrix} 8 \\ -6 \\ 3 \end{bmatrix}.$$

By using the fact that Q is a unitary matrix, find $\bar{b}$ such that

$$Rx = \bar{b}.$$

Then, given the upper-triangular matrix R and $\bar{b}$ in part (c), find the elements of x sequentially.

(c) Describe how your solution in the previous problem is akin to Gaussian elimination in solving a system of linear equations.

(d) Given an invertible matrix $B \in \mathbb{R}^{n \times n}$ and an observation vector $c \in \mathbb{R}^n$, find the computational cost of finding the solution z to the equation $Bz = c$ by using the QR decomposition of B. Assume that Q and R matrices are available, and adding, multiplying, and dividing scalars take one unit of "computation".

As an example, computing the inner product $a^\top b$ is said to be $O(n)$, since we have n scalar multiplications total – one for each $a_i b_i$. Similarly, matrix vector multiplication is $O(n^2)$, since matrix vector multiplication can be viewed as computing n inner products. The computational cost for inverting a matrix in $\mathbb{R}^n$ is $O(n^3)$, and consequently, the cost grows rapidly as the set of equations grows in size. This is why the expression $A^{-1}b$ is usually not computed by directly inverting the matrix A. Instead, the QR decomposition of A is exploited to decrease the computational cost.