CSC338. Homework 5

Due Date: Monday, June 15, 11:59pm

What to Hand In

Please hand in 3 files:

- Python File containing all your code, named `hw5.py`.
- PDF file named `hw5_written.pdf` containing your solutions to the written parts of the assignment.
- Tex file named `hw5_written.tex`

Your code will be auto-graded using Python 3.8, so please make sure that your code runs. There will be a 20% penalty if you need a remark due to small issues that renders your code untestable.

Make sure to remove or comment out all matplotlib or other expensive code before submitting your homework!

Submit the assignment on **MarkUs** by 11:59pm on the due date. See the syllabus for the course policy regarding late assignments. All assignments must be done individually.

```
import math
import numpy as np
```

Question 1

Part (a) – 2 pt

Are permutation matrices positive definite? Include your explanation in your pdf writeup.

Part (b) – 2 pt

Suppose that A is a symmetric positive definite matrix. Show that the function

$$||x||_A = (x^T Ax)^{\frac{1}{2}}$$

on a vector x satisfies the three properties of a vector norm.

Include your solution in your pdf writeup.

Part (c) – 8 pt

Complete the function `cholesky_factorize` that returns the Cholesky factorization of a matrix, according to its docstring.

```
def cholesky_factorize(A):
    """Return the Cholesky Factorization L of A, where
        * A is an nxn symmetric, positive definite matrix
        * L is lower triangular, with positive diagonal entries
        * $A = LL^T$

    >>> M = np.array([[8., 3., 2.],
                      [3., 5., 1.],
                      [2., 1., 3.]])
    >>> L = cholesky_factorize(M)
    >>> np.matmul(L, L.T)
    array([[8., 3., 2.],
```

```
            [3., 5., 1.],
            [2., 1., 3.]])
    """
```

Question 2

Part (a) – 4 pts

Complete the function `solve_rank_one_update` that solves the system $(A - \mathbf{u}\mathbf{v}^T)\mathbf{x} = \mathbf{b}$, assuming that the factorization $A = LU$ has already been done for you. You are welcome to add any helper functions that you wish, including functions that you wrote in homeworks 3 and 4. Just make sure that you include the helper functions in your python script submission.

```
def solve_rank_one_update(L, U, b, u, v):
    """Return the solution x to the system (A - u v^T)x = b, where
    A = LU, using the approach we derived in class using
    the Sherman Morrison formula. You may assume that
    the LU factorization of A has already been computed for you, and
    that the parameters of the function have:
        * L is an invertible nxn lower triangular matrix
        * U is an invertible nxn upper triangular matrix
        * b is a vector of size n
        * u and b are also vectors of size n

    >>> A = np.array([[2., 0., 1.],
                      [1., 1., 0.],
                      [2., 1., 2.]])
    >>> L, U = lu_factorize(A) # from homework 3
    >>> L
    array([[1. , 0. , 0. ],
           [0.5, 1. , 0. ],
           [1. , 1. , 1. ]])
    >>> U
    array([[ 2. ,  0. ,  1. ],
           [ 0. ,  1. , -0.5],
           [ 0. ,  0. ,  1.5]])
    >>> b = np.array([1., 1., 0.])
    >>> u = np.array([1., 0., 0.])
    >>> v = np.array([0., 2., 0.])
    >>> x = solve_rank_one_update(L, U, b, u, v)
    >>> x
    array([1. , 0. , -1.])
    >>> np.matmul((A - np.outer(u, v)), x)
    array([1. , 1. , 0.])
    """
```

Part (b) – 2 pt

Explain why using `solve_rank_one_update` does not give us accurate results in the below example:

```
def run_example():
    A = np.array([[2., 0., 1.],
                  [1., 1., 0.],
                  [1., 1., 1.]])
    L = np.array([[1., 0., 0.],
                  [0.5, 1., 0.],
```

```
                [0.5, 1., 1.]])
    U = np.array([[2., 0., 1.],
                [0., 1., -0.5],
                [0., 0., 1.]])
    b = np.array([1, 1, -1])
    u = np.array([0, 0, 0.9999999999999999])
    v = np.array([0, 0, 0.9999999999999999])
    x = solve_rank_one_update(L, U, b, u, v)
    print(np.matmul((A - np.outer(u, v)), x) - b)
```

Part (c) – 4 pt

Write function `solve_rank_one_update_iterative` that, given an approximation x to the solution $(A - \mathbf{u}\mathbf{v}^T)\mathbf{x} = \mathbf{b}$, performs one iteration of iterative refinement to obtain a better estimate $x^\star$. You may use `solve_rank_one_update` as a helper function.

```
def solve_rank_one_update_iterative(L, U, b, u, v, x):
    """Return a better solution x* to the system (A - u v^T)x = b,
    where A = LU. The first 5 parameters are the same as those of the
    function `solve_rank_one_update`. The last parameter is an
    estimate `x` of the solution.

    This function should perform exactly *one* iterative refinement
    iteration.
    """
```

Question 3

Part (a) – 4 pt

We covered the Sherman-Morrison formula in class:

$$(A - uv^T)^{-1} = A^{-1} + A^{-1}u(1 - v^T A^{-1}u)^{-1}v^T A^{-1}$$

A related formula is the Woodbury formula for a rank-k update of the matrix A. Specifically, if U and V are $n \times k$ matrices, then

$$(A - UV^T)^{-1} = A^{-1} + A^{-1}U(I - V^T A^{-1}U)^{-1}V^T A^{-1}$$

Prove that the Woodbury formula is correct. Include your solution in your pdf writeup.

Part (b) – 4 pt

A rank 1 matrix has the form xy^T where x and y are column vectors. Suppose A and B are non-sigular matrices. Show that $A - B$ is rank 1 if and only if $A^{-1} - B^{-1}$ is also rank 1.

Include your solution in your pdf writeup.

Hint: Use the Sherman-Morrison formula. This question is not supposed to be easy, so leave aside time to think!