CS 475/675 Spring 2025: Crowdmark Assignment 5

Due July 18 at 11:59 pm Eastern.
Submit all components of your solutions (written/analytical work, code/scripts, figures, plots, output, etc.) to CrowdMark in PDF form in the section for each question.

You must also separately submit a single zip file containing any and all code/scripts you write to the A05 DropBox on LEARN, in runnable format (that is, .m).

For full marks, be sure to show all your work!

1. In this question you will prove a Theorem which is stated, but not proved, in the Lecture Notes for Lecture 17. Let $n \geq 1$ be an arbitrary positive integer. Let $\hat{L}$ be an arbitrary normalized graph Laplacian matrix, of size $n \times n$. The rest of the notation in this question is as it is in the Lecture Notes.

[6] (a) For any vector $x = \begin{bmatrix} x_1 \cdots x_n \end{bmatrix}^T \in \mathbb{R}^n$, prove that

$$x^T \hat{L} x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2 .$$

[4] (b) Prove that $\hat{L}$ is symmetric and positive **semi**-definite.

[4] (c) Prove that $\hat{L}$ has n non-negative eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$.

[4] (d) Prove that the smallest eigenvalue of $\hat{L}$ is 0 and the corresponding eigenvector is $D^{\frac{1}{2}}\mathbf{1}$.

2. This question is about computing eigenvalues and eigenvectors.

   Consider the $n \times n$ tridiagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

   Let $v^{(k)}$ be the k-th eigenvector of A and $\lambda^{(k)}$ the corresponding eigenvalue. Then they are given by:

$$v_j^{(k)} = \sin\left(\frac{kj\pi}{n+1}\right) \qquad \text{and} \qquad \lambda^{(k)} = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right),$$

   where $v_j^{(k)}$ is the j-th component of $v^{(k)}$. Note: $\lambda^{(1)} < \lambda^{(2)} < \cdots < \lambda^{(n)}$.

[15] (a) Implement the following numerical methods: power iteration, Rayleigh quotient iteration, and QR iteration (without shifts). Create the following Matlab functions:

```
[v,lambda,iter] = PowerIteration(A,v0,maxiter,tol)
[v,lambda,iter] = RayleighQuotient(A,v0,maxiter,tol)
[V,Lambda,iter] = QRIteration(A,maxiter,tol)
```

   The first two Matlab functions take as inputs the matrix A, the initial vector $v0$, the maximum number of iterations *maxiter* and the tolerance *tol*, and compute the approximate eigenvector v, approximate eigenvalue *lambda*, and the number of iterations to convergence, *iter*. The third Matlab function computes all the eigenvectors and eigenvalues of A, using **simultaneous iteration**. The approximate eigenvectors are stored in matrix V and eigenvalues are stored in vector *Lambda*.

   For RayleighQuotient, you can use Matlab backslash '\' to solve linear systems. For QRIteration, use Matlab's built-in function **qr** to compute the required QR factorizations.

   For all these methods, the stopping criterion is:

$$\|A\tilde{v} - \tilde{\lambda}\tilde{v}\|_2 < tol,$$

   where $\tilde{v}$ and $\tilde{\lambda}$ are the approximate eigenvector and eigenvalue, respectively. For QRIteration, stop when this criterion is satisfied by **all** of the individual eigenvalue/eigenvector pairs. Submit all of your code.

[15] (b) Next, you will compute some eigenvectors and eigenvalues of A using the above methods for the case $n = 100$. Set the maximum iteration number, $maxiter = 10000$, and the tolerance, $tol = 10^{-4}$. Create a Matlab program, EigenMethods.m, which performs the following:

i. Use PowerIteration to compute the largest eigenvalue of A and its associated eigenvector. Use $v0 = [1, 0, \ldots, 0]^T$ for the initial vector.

ii. Use RayleighQuotient to compute an eigenvector and eigenvalue of A. The initial vector $v0 = [1, 1, \ldots, 1]^T$.

iii. Use QRIteration to compute all eigenvectors and eigenvalues of A.

For PowerIteration, make a plot of the computed eigenvector. Display the value of the computed eigenvalue and the number of iterations on the title of the plot. Do the same for RayleighQuotient. For QRIteration, use its output to make a plot of all eigenvalues, titled "Eigenvalues of A". Also, create separate plots of the column vectors $v^{(20)}$, $v^{(40)}$, $v^{(60)}$, $v^{(80)}$ of V. Title these plots with the corresponding eigenvalue, its index, and number of iterations QRIteration used.

Be careful: Our eigenvalue/vector numbering convention above has the eigenvalues in ascending order, so make sure your plots are consistent with that ordering. You may find the Matlab commands **fliplr**/**flipud** useful. Submit all of your code and plots.

3. This question is about spectral clustering image segmentation.

[15] (a) Create the Matlab function:

```
NL = CreateImageGraph(U)
```

where U is an $m \times n$ image and NL is the corresponding (sparse) normalized graph Laplacian matrix of size $mn \times mn$. Use $\sigma^2_{dist} = 150$ for the distance weighting, and $\sigma^2_{int} = 0.002$ for the intensity weighting. Consider the 8 surrounding neighbour pixels when constructing the weighted graph data. Submit your code.

[15] (b) The incomplete Matlab program CellSegment.m reads a block from a cell image file cellimage.tif, computes the normalized graph Laplacian matrix (using CreateImageGraph above), performs normalized spectral clustering image segmentation, and displays the final segmentation results. You should complete the missing part of the program as indicated in the code, i.e., implement normalized spectral clustering. This will involve extracting the appropriate eigenvectors from NL, normalizing the rows of the eigenvector matrix, and performing K-means to cluster them.

You may use the Matlab command eigs to extract the required (subset of) eigenvectors from the sparse matrix (careful: eigs default behavior returns eigenpairs for a few of the **largest** magnitude eigenvalues). Similarly, you can perform K-means clustering using the Matlab command kmeans, to determine the cluster indices for the output variable index found in the code. (For kmeans, use the additional parameter replicates, set to 20, to run K-means 20 times and return the best result.) Use $K = 9$ for the K-means process.

Submit all your code. Also submit a copy of Figure 1 generated by the program (i.e., containing the original image and the segmentation results).