CS/MA 321-001: FINAL PROJECT

There are two topics below. Please select *one* of these topics for your final project. You are welcome to try both but only one is required.

Topic 1: orthogonal matching pursuit

Orthogonal Matching Pursuit (OMP) is a greedy algorithm for finding a sparse solution (if one exists) to underdetermined systems (i.e., systems with more columns than rows). Consider the following linear system:

$$\boldsymbol{Ax} = \boldsymbol{b} \qquad\qquad \boldsymbol{A} \in \mathbb{R}^{m\times n},\ \boldsymbol{x} \in \mathbb{R}^{n},\ \boldsymbol{b} \in \mathbb{R}^{m},$$

where $m \ll n$. Typically, this system has infinitely many solutions, as there are more variables than constraints. Of particular interest in the field of sparse recovery and compressed sensing is a solution that contains as many zeros as possible. To this end, one can apply OMP which operates as follows:

Task: Approximate the solution of (P_0): $\min_{\mathbf{x}} \|\mathbf{x}\|_0$ subject to $\mathbf{Ax} = \mathbf{b}$.
Parameters: We are given the matrix $\mathbf{A}$, the vector $\mathbf{b}$, and the error threshold ϵ_0.
Initialization: Initialize $k = 0$, and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

Main Iteration: Increment k by 1 and perform the following steps:

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find a minimizer, j_0 of $\epsilon(j)$: $\forall\ j \notin \mathcal{S}^{k-1},\ \epsilon(j_0) \le \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{Ax} - \mathbf{b}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.
- **Update Residual:** Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$.
- **Stopping Rule:** If $\|\mathbf{r}^k\|_2 < \epsilon_0$, stop. Otherwise, apply another iteration.

Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

Here $\|\cdot\|_0$ counts the number of nonzeros components in a vector and Support denotes the indices of these nonzero components.

Tasks:

- Describe how this algorithm works as well as its connection to least squares.
- Implement this algorithm either on a simulated dataset or some real dataset of your choice (e.g., signal/image processing).

Topic 2: QR decomposition and eigenvalue problems

The QR decomposition is one of the most important matrix decomposition techniques in numerical analysis and scientific computing. In this problem, you are guided to complete the following tasks:

- There are two main types of algorithms for implementing QR decomposition: Gram–Schmidt orthogonalization and the Householder reflector (see Chapter 4.3 in the textbook). Please write your own code to implement one of these methods. (In practice, the Householder reflector is preferred for stability reasons.)
- Now you know how to implement QR decomposition, you can use it to solve various interesting problems. For example, it can be applied to find eigenvalues of matrices. Consider a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Linear algebra tells us that $\boldsymbol{A}$ has n real eigenvalues $\lambda_1, \ldots, \lambda_n$ counting multiplicity. (λ is called an eigenvalue of $\boldsymbol{A}$ with eigenvector $\boldsymbol{u} \neq \boldsymbol{0}$ if $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}$). The following algorithm, based on successive QR decompositions, can be used to find the eigenvalues and eigenvectors:

Algorithm 1: Iterative Eigenvalue Solver Using QR Decomposition

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$
Initialize: $\boldsymbol{A}^{(0)} \leftarrow \boldsymbol{A}, k = 0, K > 0$
while $k \leq K$ **do**
  (QR Decomposition) $\boldsymbol{Q}^{(k)}\boldsymbol{R}^{(k)} = \boldsymbol{A}^{(k)}$
  (Switching) $\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{R}^{(k)}\boldsymbol{Q}^{(k)}$
  $k \leftarrow k + 1$
end while
Output: $\boldsymbol{Q}_K = \boldsymbol{Q}^{(1)} \cdots \boldsymbol{Q}^{(K)}$ (matrix product), $\boldsymbol{A}^{(K)}$

Please implement the above algorithm and discuss your findings on the connection between $\boldsymbol{Q}_K/\boldsymbol{A}^{(K)}$ and the eigenvectors/eigenvalues of $\boldsymbol{A}$. (Hint: Check the columns of $\boldsymbol{Q}_K$ to see if they are the eigenvectors of $\boldsymbol{A}$. You may also want to take a look at the diagonals of $\boldsymbol{A}^{(K)}$ to see if they are eigenvalues.)