Handed out: Wed., Oct. 20.

Due: Fri., Oct. 29 in lecture.

1. Develop an efficient algorithm for the following problem, and analyze its running time accurate to the leading terms. Given an $n \times n$ lower triangular matrix G that is the Cholesky factor of A (i.e., $A = GG^T$), and given a matrix $H \in \mathbf{R}^{n \times p}$, where $p < n$, find the Cholesky factor of $A + HH^T$.

   Suggested approach: Observe that $A + HH^T = [H, G][H^T; G^T]$. If we could compute a QR factorization of $[H^T; G^T]$ efficiently, then the problem would be solved (why? consider the R-factor here). To compute this factorization efficiently, consider Householder transformations that have extra sparsity to account for the nonzero pattern of $[H^T; G^T]$.

2. (a) Derive an algorithm to solve the following problem. Given $A \in \mathbf{R}^{m \times n}$ find $\mathbf{x} \in \mathbf{R}^m$ and $\mathbf{y} \in \mathbf{R}^n$ to maximize $\mathbf{x}^T A \mathbf{y}$, subject to the constraints that $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$. In fact, show also that the maximum value of $\mathbf{x}^T A \mathbf{y}$ under these constraints is equal to $\|A\|_2$.

   [Hint for the algorithm: use the SVD. Hint for showing that $\|A\|_2 = \sigma_1(A)$ is the maximum possible value: use the Cauchy-Schwarz inequality.]

   (b) The following problem arises in structural mechanics ("maximum shear stress"). Let $A \in \mathbf{R}^{n \times n}$ be symmetric. The problem is to find $\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$ that maximize $\mathbf{x}^T A \mathbf{y}$ subject to the constraints that $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$, and $\mathbf{x}^T \mathbf{y} = 0$. Develop an algorithm to solve this problem, and justify the steps of the algorithm.

   [Hints: (1) First solve the problem in the very special case that A is 2×2 of the form $A = \mathrm{diag}(a, -a)$. You can use part (a) of this question to argue that the solution you found is optimal. (2) Then generalize to the case of an $n \times n$ diagonal matrix whose maximum diagonal entry is a and whose minimum diagonal entry is $-a$. (3) Then generalize to an arbitrary $n \times n$ diagonal matrix by claiming that the optimal $\mathbf{x}, \mathbf{y}$ are unchanged if A is replaced by $A + \alpha I$. (4) Finally, use diagonalization and change of variables to solve for the general case.]

3. (Exercise 4.24 from Heath.) Let A be an $n \times n$ real rank-one matrix of the form $A = \mathbf{u}\mathbf{v}^T$ where $\mathbf{u}^T \mathbf{v} \neq 0$.

   (a) Show that $\mathbf{u}^T \mathbf{v}$ is an eigenvalue of A. What is its eigenvector?

   (b) What are the other eigenvalues of A? [Hint: normalize $\mathbf{v}$ and extend it to an orthonormal basis.]

(c) Suppose the power method is applied to A. How many iterations are required for it to converge exactly to the eigenvector corresponding to the dominant eigenvalue? What assumptions must be made about the initial vector?

4. On the class web page is a dataset containing information from the "help" files for matrix functions in Matlab (i.e., the files accessible starting from the menu given by `help matlab/matfun`). Specifically, the following data items are stored in ps4q4.mat, which is downloadable and should be loaded into matlab via the statement `load ps4q4.mat`.

   - `doctitles`: A cell array with the titles of the help pages (46 documents total in the corpus). To access the title of the ith document, you would say `doctitles{i}`. Note the curly braces. Cell arrays are Matlab arrays in which the entries can be arbitrary Matlab data (e.g., entire strings in this case) rather than just numbers.

   - `A`: The term-document matrix, which is 866 by 46. It is stored in sparse format so that the data file is not too large. Make it full with the statement `A=full(A)`.

   - `finaltermlist`: A cell array with the 866 terms that appear in this corpus.

   Write three functions for retrieving documents from this corpus as follows.

   - `function qvector = convert_query(query,finaltermlist)`: This function takes as input a natural language query, stored in a string called `query`, and converts it to a vector in term-space (which is $\mathbf{R}^{866}$ for this dataset). It does so by matching the terms in the query against the terms in `finaltermlist`. For this function, the Matlab string-operation primitives `lower`, `strtok` and `strmatch` might be helpful. To check if a token `w` is a word, use `all(isletter(w))`.

   - `function scores = scoredoc(qvector,U,S,V,k)`: This function returns a score for each document (i.e., a vector in $\mathbf{R}^{46}$ for this corpus) for the query using the LSI operations covered in class. Here, `U,S,V` are the outputs of the Matlab `svd` function applied to A; k is the number of singular values to keep (I experimented with 10); and `qvector` is a query converted to a vector by the previous function.

   - `function topthree(doctitles,scores)`: This function has no return values, but prints out the top three LSI matches in the corpus for the scores computed by `scoredoc`. The `disp(sprintf(...))` construction and `sort` functions may be helpful.

   Hand in listings of these functions and three test runs, include a test for the query "estimate condition number":

```
>> q=convert_query('estimate condition number',finaltermlist);
>> scores=scoredoc(q,U,S,V,10);
>> topthree(doctitles,scores)
                 condest     0.255371
                 normest     0.243799
                normest1     0.235147
```