CS 621: Final Exam

Monday, December 12, 2005

Class Averages

Problem 1	15 points	10.8
Problem 2	20 points	11.6
Problem 3	20 points	16.8
Problem 4	15 points	7.6
Problem 5	15 points	6.8
Problem 6	15 points	12.3
	100 points	65.9

Rough grade Equivalents: A = 70-100, B = 55-69, C = 40-54

1. (a) (7 points) Complete the following function so that it performs as specified:

```
  function [W Y] = Update(B,C,u,v)
% B and C are n-by-k matrices
% u and v are n-by-1 vectors
% W and Y are n-by-(k+1) matrices so that (I + W*Y') = (I + B*C')*(I + u*v')
```

How many flops does your implementation require assuming that $n >> k$?

See Problem 1 in Assignment 1. This is about block manipulation, working with low rank matrices.

If $P = I + BC^T$ then

$$(I + BC^T)(I + uv^T) = I + BC^T + (Pu)v^T = I + [\ B\ \ Pu\ ][\ C\ \ v\ ]^T \equiv I + WY^T$$

and so

```
x = u + W*(Y'*u);   % 4 point    -3 if  x = u + W*Y'*u  as this is O(n^2 k)
W = [B x];          % 1 point
Y = [C v];          % 1 point
% 4nk flops         % 1 point
```

Alternate solution: If $P = I + uv^T$ then

$$(I + BC^T)(I + uv^T) = P + BC^TP = I + uv^T + B(P^TC)^T = I + [\ u\ B\ ][\ v\ P^TC\ ]^T \equiv I + WY^T$$

and so

```
C1 = C + v*(u'*C)*;   % 4 points    -3 if  C1 = C + v*u'*C  as this is O(n^3)
W = [u B];            % 1 point
Y = [v C1];           % 1 point
% 4nk flops          % 1 point
```

(b) (8 points) Suppose $A \in \mathbb{R}^{m\times n}$ has rank n, $b \in \mathbb{R}^m$, and $\lambda \in \mathbb{R}$. Show how to use the SVD of A to compute $\| x(\lambda) \|_2^2$ where $x(\lambda) \in \mathbb{R}^n$ solves the problem

$$\min_{x\ \in\ \mathbb{R}^n} \left\| \begin{bmatrix} A \\ \lambda I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2$$

This is about substituting the SVD into a matrix equation and simplifying. Similar problems to this one came up when we talked about constrained least squares.

The easiest way to proceed is to substitute $A = U\Sigma V^T$ into the normal equation characterization of the solution:

$$(A^TA + \lambda^2 I)x(\lambda) = A^Tb \qquad \Rightarrow \qquad (\Sigma^T\Sigma + \lambda^2 I)y(\lambda) = \Sigma^TU^Tb$$

where $y(\lambda) = V^Tx(\lambda)$. It follows that

$$x(\lambda) = \sum_{j=1}^{n} \frac{\sigma_j U(:,j)^Tb}{\sigma_j^2 + \lambda^2} V(:,j) \qquad \Rightarrow \qquad \| x(\lambda) \|_2^2 \ = \ \sum_{j=1}^{n} \left(\frac{\sigma_j U(:,j)^Tb}{\sigma_j^2 + \lambda^2} \right)^2$$

If you work with the original form you get this transformed problem

$$\left\| \begin{bmatrix} A \\ \lambda I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} \Sigma \\ \lambda V \end{bmatrix} V^T x - \begin{bmatrix} U^T b \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} \Sigma \\ \lambda I_n \end{bmatrix} V^T x - \begin{bmatrix} U^T b \\ 0 \end{bmatrix} \right\|_2$$

where we multiplied the second block row by V^T to get the final form. 5 points for this. You then have to say how to get $x(\lambda)$. Here again, normal equations makes this easy. Or you can talk about an $O(n)$ QR factorization of a block matrix that is diagonal-over-diagonal

2. (20 points) Recall that if $C \in \mathbb{R}^{n\times n}$ then `[L,U,P] = lu(C)` returns the factorization $PC = LU$ where $L \in \mathbb{R}^{n\times n}$ is unit lower triangular, $U \in \mathbb{R}^{n\times n}$ is upper triangular and $P \in \mathbb{R}^{n\times n}$ is a permutation matrix.

It is known that μ is a non-repeated eigenvalue of $A \in \mathbb{R}^{n\times n}$ and that there is a unique $x \in \mathbb{R}^n$ so that (i) $Ax = \mu x$, (ii) $x_i > 0, i = 1{:}n$, and (iii) $\| x \|_2 = 1$ and a unique $y \in \mathbb{R}^n$ so that (i) $A^T y = \mu y$, $y_i > 0, i = 1{:}n$, and (iii) $\| y \|_2 = 1$ Write a MATLAB function `[x,y] = TwoVecs(A,mu)` that computes x and y. Make effective use of `lu`. Ignore round-off. and be efficient.

This problem is about working with the LU factorization $PC = LU$. This factorization always exists when we do partial pivoting–even if C is singular.

Since $P(A - \mu I) = LU$ is singular, U must have a zero on its diagonal. From

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & 0 & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix} \begin{bmatrix} -U_{11}^{-1}U_{12} \\ 1 \\ 0 \end{bmatrix} = 0$$

and the fact that $Ux = 0 \Rightarrow Ax = 0$, it follows from (ii) that U must have a single zero on its diagonal and it must be in the (n, n) position. (Otherwise, we would have an eigenvector with a zero bottom oomponent. So how do you find $z \in \mathbb{R}^{n-1}$ so that

$$\begin{bmatrix} \tilde{U} & u \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ 1 \end{bmatrix} = 0.$$

Answer: $z = -\tilde{U}^{-1}u$. So here is how to compute x:

```
function [x,y] = TwoVecs(A,mu)
[n,n] = size(A);
[L,U,P] = lu(A -mu*eye(n,n));
z = -U(1:n-1,1:n-1)\U(1:n-1,n);
x = [z;1]; x = x/norm(x).
```

Note that $(A-\mu I)^T P^T = U^T L^T$ and that e_n is a null vector for U^T. So if $L^T w = e_n$, then $A-\mu I)^T P^T w = 0$. So here is how to get y:

```
w = [L(1:n-1,1:n-1)'\L(n,1:n-1)';1];
y = P'*w;
y = y/norm(y);
```

Roughly 5 points for LU of $A - \mu I$, 5 points for reasoning about the zero of U's diagonal, 5 points for getting x, and 5 points for using the transposed factorization to get y. If you worked with $PA = LU$ you could get up

to half credit if you showed how to work with the transposed factorization and had other insights.

3. (a) (10 points) Assume that

$$A = \begin{bmatrix} d_1 & f^T \\ f & \text{diag}(d_2, \ldots, d_n) \end{bmatrix} \qquad f \in \mathbb{R}^{(n-1)}$$

is positive definite. Show how to determine

$$G = \begin{bmatrix} \delta_1 & 0 \\ g & \Delta \end{bmatrix} \qquad g \in \mathbb{R}^{(n-1)}$$

with $\Delta = \text{diag}(\delta_2, \ldots, \delta_n)$ so that $A = GG^T$.

This Problem is about equating blocks in $A = GG^T$.

- $\delta_1^2 = d_1$ so $\delta_1 = \sqrt{d_1}$.
- $g_i \delta_1 = f_i$ so $g = f/\delta_1$.
- $g_i^2 + \delta_i^2 = d_i$ so $\delta(2{:}n) = \sqrt{d(2{:}n) - g. * g}$.

This was worth 15 points. (Revised the 10-10 point split in this problem to 15-5.) Side note: There was a typo in the problem; Δ is not diagonal, it is the Cholesky factor of $D - gg^T$. An earlier draft of this problem had you compute the Cholesky fatorization of PAP^T where P is a permutation. Setting $P =$ exchange matrix then gives the structured G of the problem. Full credit was given for working with the block comparisons assuming that Δ was diagonal. The typo didn't seem to affect anyone's score. Indeed, for many students this was the highest scoring problem.

(b) (10 points) Given a unit 2-norm vector $v \in \mathbb{R}^{(n-1)}$, show how the SVD can be used to determine c and s with $c^2 + s^2 = 1$ so that the solution to

$$\begin{bmatrix} \delta_1 & 0 \\ g & \text{diag}(\delta_2, \ldots, \delta_n) \end{bmatrix} \begin{bmatrix} \mu \\ z \end{bmatrix} = \begin{bmatrix} c \\ sv \end{bmatrix}$$

has maximum 2-norm.

If

$$\begin{bmatrix} \delta_1 & 0 \\ g & \Delta \end{bmatrix} \begin{bmatrix} \mu \\ z \end{bmatrix} = \begin{bmatrix} c \\ sv \end{bmatrix}$$

then

$$\begin{bmatrix} \mu \\ z \end{bmatrix} = \begin{bmatrix} c/\delta_1 \\ \Delta^{-1}(sv - cg/\delta_1) \end{bmatrix} = M \begin{bmatrix} c \\ s \end{bmatrix}$$

where

$$M = \begin{bmatrix} 1/\delta_1 & 0 \\ -\Delta^{-1} g/\delta_1 & \Delta^{-1} v \end{bmatrix}$$

Thus, if we compute the SVD of $M = U \Sigma V^T$, then

$$V(:,1) = \begin{bmatrix} c \\ s \end{bmatrix}$$

4. (15 points) In each of the following **u** designates the unit roundoff. Brief "order-of-magnitud" answers are expected.

(a) Assume that A is symmetric and positive definite. How in the worst case might the solution to $Ax = b$ change if the entries in A are perturbed by $O(\mathbf{u}\| A \|)$?

Relative error in perturbed solution is about $\mathbf{u}\kappa(A)$. Positive definite systems *can be* ill conditioned.

(b) How in the worst case might the solution to the full rank least squares problem $\min\| Ax - b \|_2$ change if the entries in A are perturbed by $O(\mathbf{u}\| A \|)$?

Relative error in perturbed solution is about $\mathbf{u}(\kappa(A) + \rho_{LS}\kappa(A)^2)$. Where ρ_{LS} is the norm of the minimum residual of the original problem.

(c) Assume that λ is a nonrepeated eigenvalue of a symmetric matrix A. How in the worst case might λ change if the entries in A are perturbed by $O(\mathbf{u}\| A \|)$?

If we perturb A by $O(\mathbf{u}\| A \|)$ then eigenvalues move no more than $O(\mathbf{u}\| A \|)$ The eigenvalues of a symmetric matrix are perfectly well conditioned.

(d) Assume that λ is a nonrepeated eigenvalue of an unsymmetric matrix A. How in the worst casemight λ change if the entries in A are perturbed by $O(\mathbf{u}\| A \|)$?

If we perturb A by $O(\mathbf{u}\| A \|)$ then eigenvalues can move $O(\mathbf{u}\| A \|/s$ where s is the cosine of the angle between the left and right eigenvector–a possibly very small number.

(e) Assume that A has a unique minimum singular value. How in the worst case might the corresponding right singular vector change if the entries in A are perturbed by $O(\mathbf{u}\| A \|)$?

The direction that $V(:, n)$ points can change a lot if there isnt a sufficient gap between σ_{n-1} and σ_n. (Recall the test data for the TLS problem that was assigned.)

5. (15 points) Complete the following function so that it performs as specified.

```
  function alpha = ConstrainedMax(A,u)
% A is an n-by-n symmetric matrix
% u is a unit 2-norm n-vector
% Let R be the Rayleigh quotient R(x) = (x'*A*x)/(x'*x)
% alpha is the maximum value of R(x) subject to the constraint that x is orthogonal to u.
```

You may use `QR` and `schur`. Do not worry about efficiency in your implementation. But write a few sentences indicating how you would implement `ConstrainedMax` if A was large and sparse and efficiency *was* important.

Suppose $Q \in \mathbb{R}^{n\times(n-1)}$ has orthonormal columns and $\text{ran}(Q) = \text{span}\{u\}^{\perp}$. If $x \in \text{span}\{u\}^{\perp}$ then $x = Qy$ for some $y \in \mathbb{R}^{(n-1)}$. It follows that

$$R(x) = R(Qy) = (Qy)^T A(Qy)/(Qy)^T(Qy) = y^T(Q^T AQ)y/y^T y.$$

Thus, the optimum y is the largest eigenvector of $Q^T AQ \in \mathbb{R}^{(n-1)\times(n-1)}$. The easiest way to compute Q is to note that if P is a Householder matrix so $Pu = e_1$, then $Q = P(:, 2{:}n)$. But you are allowed to "say" this by using `qr`

```
[V,R] = qr(u)
P = V(:,2:n)                    % 5 ppoints for getting a basis for the subspace
A0 = P'*A*P;
[U,D] = schur(A0)               % 5 points for solving the "reduced" eigen problem
alpha = max(diag(D));           % 3 points for knowing how to maximize a rayleigh quotient
```

For Lanczos, if the initial vector is u, then all subsequent q-vectors are orthogonal. If T is the tridiagonal after k steps, then the largest eigenvalue of $T(2{:}k, 2{:}k)$ approximates the largest eigenvalue of the matrix $A0$ above. (2 points)

6. (a)(5 points) Suppose $Q^T AQ = T$ is the Schur decomposition of $A \in \mathbb{R}^{n\times n}$ and that all of A's eigenvalues are real. Explain how the columns of Q define a nested sequence of invariant subspaces.

Comparing columns in AQ = QT we see that

$$Aq_k = \sum_{i=1}^{k} t_{ik} q_i$$

Thus, Aq_1 is a multiple of q_1. A times any vector in $\text{span}\{q_1, q_2\}$ remains in that space. A times any vector in $\text{span}\{q_1, q_2, q_3\}$ remains in that space. etc. Thus $\text{span}\{q_1, \ldots, q_k\}$ is an invariant subspace for A, $k = 1{:}n$.

(b)(10 points) Complete the following function so that it performs as specified

```
  function [c,s] = ReOrder(T)
% T is a 2-by-2 upper triangular matrix.
% c^2 + s^2 = 1 with the property that if
%           S = [c  s; -s  c]'*T*[c  s; -s  c]
%
% then S is upper triangular with S(1,1) = T(2,2) and S(2,2) = T(1,1)
```

Your implementation should not make use of `schur`, `eig`, `qr` or `svd`. It must handle all possible T in a numerically reliable fashion.

```
% Get eigenvector associated with T(2,2)
x = [T(1,2) ; T(2,2)-T(1,1)]
% Determine c and s so [c s;-s c]'*x  has a zero second component
%  s*T(1,2) + c*(T(2,2)-T(1,1))...
if abs(T(1,2)) < abs(T(2,2)-T(1,1))
    mu = T(1,2)/(T(1,1)-T(2,2)); c = 1/sqrt(1 + mu^2); s = c*mu
else
    if abs(T(1,2))==0
       s = 1; c = 0;
    else
       mu = (T(1,1)-T(2,2))/T(1,2); s = 1/sqrt(1 + mu^2); c = mu*s;
    end
end
```