HW1 Least square approximation & QR iterative algorithm

VLSI DSP HW1

Shun-Linag Yeh, NCHU Lab612

3/16 2023

INDEX

Least square approximation

Problem

1. Lest square optimization problem

   For an over constrained linear system $\mathbf{Ax}=\mathbf{b}$, find the least square solution of $\mathbf{x}$ using

   a) Pseudo inverse $\hat{\mathbf{x}} = \mathbf{A}^{+} \cdot \mathbf{b}$

   b) QR decomposition,

   c) Compare if a) and b) yield the same result?

Derivation steps

1. For pseudoinverse uses the pinv matlab function to find the pseudoInverse of A, then approximation x through the equation.
2. For QR decomposition approximation, generate QR using matlab function , using the normal equation (A^t) Ax = (A^t)b, since A = QR, replace A with QR in the normal equation yields , plug the QR into the derived equation to get the solution.

Code

```matlab
A = [15 -13 20 -8;
     -5 -15 -4 -4;
     -17 16 -2 9;
     10 -19 -14 -15;
     -7 8 -7 15;
     14 10 -8 -17;
     -5 -3 16 -2;
     13 -5 -10 -19];

b = [13 10 -15 9 3 18 3 20];
b = b';
A_dagger = pinv(A);
% [Q, R] = qr(A);
% disp(Q);
% disp(R);

x_hat_dagger = A_dagger * b;
x_hat_qr     = ((R')*R)^(-1)*(R')*(Q')*b;

disp("Solution with pseuodoInverse");
disp(x_hat_dagger);
disp("Solution with QR");
disp(x_hat_qr);
```

Result

```
Solution with pseuodoInverse
    0.4638
   -0.1005
   -0.0716
   -0.4137

Solution with QR
    0.4638
   -0.1005
   -0.0716
   -0.4137
```

QR iterative algorithm

Problem

2. Eigen decomposition
For a symmetric matrix $\mathbf{M}$ shown below, find its Eigen value decomposition using a QR decomposition based iterative algorithm.

$$
\mathrm{M} = \begin{bmatrix}
-2 & 16 & -6 & -16 & 3 & 15 & -6 & -19 \\
16 & -17 & 10 & -2 & 7 & 8 & 3 & 5 \\
-6 & 10 & 15 & -1 & -15 & -18 & 9 & -8 \\
-16 & -2 & -1 & 9 & 0 & 0 & 0 & 18 \\
3 & 7 & -15 & 0 & 14 & 19 & -12 & 11 \\
15 & 8 & -18 & 0 & 19 & 10 & -8 & -17 \\
-6 & 3 & 9 & 0 & -12 & -8 & 15 & 20 \\
-19 & 5 & -8 & 18 & 11 & -17 & 20 & 20
\end{bmatrix}
$$

Eigen decomposition means to find a matrix decomposition of the format shown below

$$
\mathbf{M} = \mathbf{Q} \cdot \mathbf{\Lambda} \cdot \mathbf{Q}^t \quad \text{where} \quad \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & 0 & \\ & 0 & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}
$$

is a diagonal matrix consisting of real Eigen values. $\mathbf{Q}$ is orthogonal with the property $\mathbf{Q} \cdot \mathbf{Q}^t = \mathbf{I}$ and all its column vectors as Eigen vectors.

Givens rotation

$$
\begin{bmatrix}
1 & & & & & & & & \\
& \ddots & & & & & & & \\
& & 1 & & & & & & \\
& & & c & & s & & & \\
& & & & 1 & & & & \\
& & & & & \ddots & & & \\
& & & & & & 1 & & \\
& & & -s & & c & & & \\
& & & & & & & 1 & \\
& & & & & & & & \ddots \\
& & & & & & & & & 1
\end{bmatrix}
\begin{bmatrix} \times \\ \vdots \\ \times \\ a \\ \times \\ \vdots \\ \times \\ b \\ \times \\ \vdots \\ \times \end{bmatrix}
=
\begin{bmatrix} \times \\ \vdots \\ \times \\ r \\ \times \\ \vdots \\ \times \\ 0 \\ \times \\ \vdots \\ \times \end{bmatrix}.
$$

- So, to transform A into an upper triangular matrix R, we can find a product of rotations Q such that $Q^T A = R$.

- It is important to note that the straightforward approach to computing the entries c and s of the Givens rotation,
$$c = \frac{a}{\sqrt{a^2+b^2}}, \quad s = \frac{b}{\sqrt{a^2+b^2}},$$
is not always advisable, because in floating-point arithmetic, the computation of $\sqrt{a^2+b^2}$ could overflow.
- To get around this problem, suppose that $|b| \geq |a|$. Then, we can instead compute
$$t = \frac{a}{b}, \quad s = \frac{\operatorname{sgn}(b)}{\sqrt{1+t^2}}, \quad c = st, \tag{1}$$
which is guaranteed not to overflow since the only number that is squared is at most one in magnitude.
- Similarly, if $|a| \geq |b|$, then we compute
$$t = \frac{b}{a}, \quad c = \frac{\operatorname{sgn}(a)}{\sqrt{1+t^2}}, \quad s = ct. \tag{2}$$

Tridiagnalization

1. After computing M' = Q'M using givens rotation through the linear transformation of rotation matrices where Q' is products of a series of Rotation matrices G, we yields an upper triangular matrix that is close to tridiagonal form.
2. Replace the upper symmetrical part of M' with the same result as the lower entries where i<j.
3. We can get triangularized matrix M'',where M'' = Q Q' M

QR Iterative Algorithm

Algorithm. (QR Factorization via Givens rotations) Let $m \geq n$ and let $A \in \mathbb{R}^{m \times n}$ have full column rank. The following algorithm uses Givens rotations to compute the QR Factorization $A = QR$, where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular.

$Q = I$
$R = A$
for $j = 1, 2, \ldots, n$ **do**
  for $i = m, m-1, \ldots, j+1$ **do**
    $[c, s] = \mathbf{givens}(r_{i-1,j}, r_{ij})$
    $R = G(i-1, i, c, s)^T R$
    $Q = QG(i-1, i, c, s)$
  end for
end for

Code

Main driver

```
m = [2 16 -6 -16 3 15 -6 -19;
     16 -17 10 -2 7 8 3 5;
     -6 10 15 -1 -15 -18 9 -8;
     -16 -2 -1 9 0 0 0 18;
     3 7 -15 0 14 19 -12 11;
     15 8 -18 0 19 10 -8 -17;
     -6 3 9 0 -12 -8 15 20;
     -19 5 -8 18 11 -17 20 20];

max_iters = 20000;

[M_tilda, Q, R] = tridiagonal_and_QR(m);
disp("Tridiagonalized matrix");
disp(M_tilda);

M = QR_eigen_decompose(m, max_iters);
disp("M after QR_eigen_decompose");
disp(M);

[V, D] = eig(m);
disp("M after eigen_decomposition through matlab function");
disp(D);

function M = QR_eigen_decompose(A, n)
    M = A;

    for i = 1:n
        %[Q,R] = qr(M);
        [M_tilda, Q, R] = tridiagonal_and_QR(M);
        M = R * Q;
    end

end
```

Tridiagonalization and QR iterative

```
% qrgivens.m
function [M_tilda, Q, R] = tridiagonal_and_QR(A)
    [m, m] = size(A);
    Q = eye(m);
    R_close = A;
    R = A;

    % QR
    for j = 1:m

        for i = m:-1:(j + 1)
            G = eye(m);
            [c, s] = givensrotation(R(i - 1, j), R(i, j));
            G([i - 1, i], [i - 1, i]) = [c -s; s c];
            %Near Upper triangular matrix
            if i >= j + 2
                R_close = G' * R_close;
            end

            R = G' * R;
            %disp(G');
            %The orthogonal matrix s.t. M_tilda = QM
            Q = G' * Q;
            %disp(Q)
        end

    end

    for i = 1:m

        for j = 1:m

            if j > i
                R_close(i, j) = R_close(j, i);
            end

        end

    end

    M_tilda = R_close;

    % Since inverse is Q transposed!
    Q = Q';
end
```

Givens Rotation

```
% Givens rotation
function [c, s] = givensrotation(a, b)
    %Input two entries a,b outputs the coefficient of givens rotation matrix.
    if b == 0
        c = 1;
        s = 0;
    else

        if abs(b) > abs(a)
            r = a / b;
            s = 1 / sqrt(1 + r ^ 2);
            c = s * r;
        else
            r = b / a;
            c = 1 / sqrt(1 + r ^ 2);
            s = c * r;
        end

    end

end
```

Result

Tridiagonalized matrix

```
Tridiagonalized matrix
    2.0000  -34.3366    0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0000
  -34.3366    7.9216  -21.8460   -0.0000    0.0000    0.0000   -0.0000    0.0000
    0.0000  -21.8460   14.2004   28.3512   -0.0000   -0.0000   -0.0000    0.0000
   -0.0000   -0.0000   28.3512   -9.1448  -10.1490   -0.0000   -0.0000   -0.0000
    0.0000    0.0000   -0.0000  -10.1490  -11.3477   -4.7140    0.0000   -0.0000
   -0.0000    0.0000   -0.0000   -0.0000   -4.7140   -9.0758    9.4462   -0.0000
    0.0000   -0.0000   -0.0000   -0.0000    0.0000    9.4462   -1.4133  -10.2268
    0.0000    0.0000    0.0000   -0.0000   -0.0000   -0.0000  -10.2268  -18.2736
```

Eigenvalue matrix after eigen-decomposition

```
M after QR_eigen_decompose
   67.8045   -0.0000   -0.0000    0.0000    0.0000   -0.0000   -0.0000   -0.0000
    0.0000   46.9072   -0.0000    0.0000   -0.0000    0.0000    0.0000   -0.0000
         0   -0.0000  -36.1316    0.0000   -0.0000   -0.0000    0.0000    0.0000
         0         0   -0.0000  -25.4939    0.0000   -0.0000    0.0000    0.0000
         0         0         0    0.0000   17.0615    0.0000   -0.0000    0.0000
         0         0         0         0   -0.0000   -9.6425    0.0000   -0.0000
         0         0         0         0         0   -0.0000    6.0585   -0.0000
         0         0         0         0         0         0    0.0000    1.4364
```

Eigenvalue decompose using matlab

```
M after eigen_decomposition through matlab function
  -36.1316         0         0         0         0         0         0         0
         0  -25.4939         0         0         0         0         0         0
         0         0   -9.6425         0         0         0         0         0
         0         0         0    1.4364         0         0         0         0
         0         0         0         0    6.0585         0         0         0
         0         0         0         0         0   17.0615         0         0
         0         0         0         0         0         0   46.9072         0
         0         0         0         0         0         0         0   67.8045
```

Note

1. The result of the Eigen-decomposition differs due to the ordering of the orthnormal basis and the ordering of eigenvectors when calculating the QR matrcies.
2. The eigenvalues are all the same but in different orderings.

Reference

[1] University of South California, Section 4.2.1: Givens Rotations, Math610 Jim Lambers