QR factorization

There is a different algorithm to solve least-squares problems based on a different matrix factorization, the QR factorization.

Not as powerful / revealing as SVD, but easier to compute. We shall see its computation in detail.

Idea Mix Gaussian elimination / LU factorization with orthogonal transformations.

First, we start with an easy case.

The case of a vector

Problem

Given $\mathbf{x} \in \mathbb{R}^n$, find an orthogonal matrix Q such that $Q\mathbf{x}$ is of the form $\begin{bmatrix} s \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = s\mathbf{e}_1$.

(We call $\mathbf{e}_j$ the jth column of I.)

Remark Since orthogonal matrices preserve norm, s can only be $\pm\|\mathbf{x}\|$.

Householder reflectors

Lemma

For every $\mathbf{v} \in \mathbb{R}^m$, the matrix $H = I - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T$ is orthogonal *and* symmetric .

Written also $I - \frac{2}{\|\mathbf{v}\|^2}\mathbf{v}\mathbf{v}^T$, or $I - 2\mathbf{u}\mathbf{u}^T$ where $\mathbf{u} = \frac{1}{\|\mathbf{v}\|}\mathbf{v}$ has norm 1.
Proof: verify directly $HH^T = I$ and $H = H^T$.

Geometric idea: these are reflections (mirroring) with respect to the plane perpendicular to $\mathbf{v}$. Check for instance the case

$$\mathbf{u} = \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Cost-saving trick Rearrange parentheses! For each $\mathbf{x} \in \mathbb{R}^{m \times m}$ we can compute $H\mathbf{x} = (I - 2\mathbf{u}\mathbf{u}^T)\mathbf{x} = \mathbf{x} - 2\mathbf{u}(\mathbf{u}^T\mathbf{x})$ in $\mathcal{O}(m)$, and HA for any $A \in \mathbb{R}^{m \times m}$ in $\mathcal{O}(m^2)$.

Where can we get by reflecting

Lemma

Let $\mathbf{x}, \mathbf{y}$ be two vectors such that $\|\mathbf{x}\| = \|\mathbf{y}\|$. If one chooses $\mathbf{v} = \mathbf{x} - \mathbf{y}$, then $H = I - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T$ is such that $H\mathbf{x} = \mathbf{y}$.

Proof: boring algebra: substitute $\mathbf{x} = \mathbf{y} + \mathbf{v}$, clear denominators and expand.

Geometric idea: reflecting through the plane perpendicular to $\mathbf{x} - \mathbf{y}$ sends $\mathbf{x}$ into $\mathbf{y}$.

In particular, we can take $\mathbf{y} = \|\mathbf{x}\|\mathbf{e}_1 = \begin{bmatrix} \|\mathbf{x}\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

Matlab implementation

```
function [u, s] = householder_vector(x)
s = norm(x);
v = x;
v(1) = v(1) - s;
u = v / norm(v);
```

Testing it:

```
>> x = randn(4,1);
>> [u, s] = householder_vector(x);
>> x - 2*u*(u'*x)
ans =
   2.2541e+00
            0
  -1.1102e-16
            0
>> s
s =
   2.2541e+00
```

An extreme example

```
>> format short e
>> x = [1e4; 1e-6; 1e-6; 1e-6]
x =
    1.0000e+04
    1.0000e-06
    1.0000e-06
    1.0000e-06
>> [u, s] = householder_vector(x);
>> x - 2*u*(u'*x)
ans =
    1.0000e+04
    -1.0000e-06
    -1.0000e-06
    -1.0000e-06
```

The transformed vector is still at relative distance $10^{-10} \gg \mathsf{u}$ from being a multiple of $\mathbf{e}_1$; we did not improve things.

Reason for instability

Problem: subtracting two almost-equal values $\rightarrow$ cancellation.

```
>> x(1), norm(x), x(1) - norm(x)
ans =
       10000
ans =
       10000
ans =
       0
```

Small relative errors in the computation of `norm(x)` (e.g., computing $\|x\|(1+\varepsilon)$ instead) cause huge relative errors on u_1.

To improve stability, we make a small modification: we choose

- $s = -\|\mathbf{x}\|$ whenever $x_1 \geq 0$.
- $s = \|\mathbf{x}\|$ whenever $x_1 < 0$.

In this way, $x_1 - s$ always sums two numbers with the same sign.

Solution

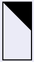
```
function [u, s] = householder_vector(x)
s = norm(x);
if x(1) >= 0, s = -s; end
v = x;
v(1) = v(1) - s;
u = v / norm(v);
```

Now that example works better:

```
>> x = [1e4; 1e-6; 1e-6; 1e-6];
>> [u, s] = householder_vector(x);
>> x - 2*u*(u'*x)
ans =
       -10000
            0
            0
            0
```

QR factorization

Theorem

For every $A \in \mathbb{R}^{m \times n}$, there exist $Q \in \mathbb{R}^{m \times m}$ orthogonal, R upper triangular (i.e., $i > j \implies R_{ij} = 0$, or ) such that $A = QR$.

Most interesting case for us: $m \geq n$ (square or tall-thin).

Note that we have already solved the case $n = 1$: $\mathbf{x} = H(s\mathbf{e}_1)$ is a QR factorization.

Idea: work like in Gaussian elimination / LU factorization: use orthogonal matrices to transform A into an upper triangular matrix, one column at a time.

QR factorization via Householder matrices

We start from $A \in \mathbb{R}^{m \times n}$.
Step 1: take $[\mathbf{u}_1, s_1] =$ householder_vector(A(:, 1)) to get

$$H_1 A = \begin{bmatrix} s_1 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} =: A_1.$$

How can we introduce more zeros without spoiling those already computed in the first column?

Idea Left-multiply by a matrix of the form $Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & H_2 \end{bmatrix}$. It leaves the first row unchanged and multiplies the others by $H_2 \in \mathbb{R}^{(m-1)\times(m-1)}$.
Step 2: take $[\mathbf{u}_2, s_2] =$ householder_vector(A1(:, 1)), and compute

$$A_2 = \begin{bmatrix} 1 & 0 \\ 0 & H_2 \end{bmatrix} \begin{bmatrix} B_2 & C_2 \\ 0 & D_2 \end{bmatrix} = \begin{bmatrix} B_2 & C_2 \\ 0 & H_2 D_2 \end{bmatrix} = \begin{bmatrix} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} = \begin{bmatrix} B_3 & C_3 \\ 0 & D_3 \end{bmatrix}.$$

Continue...

$$\begin{bmatrix} I_{2\times 2} & 0 \\ 0 & H_3 \end{bmatrix} \begin{bmatrix} B_3 & C_3 \\ 0 & D_3 \end{bmatrix} = \begin{bmatrix} B_3 & C_3 \\ 0 & H_3 D_3 \end{bmatrix} = \begin{bmatrix} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & s_3 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix},$$

$$\begin{bmatrix} I_{3\times 3} & 0 \\ 0 & H_4 \end{bmatrix} \begin{bmatrix} B_4 & C_4 \\ 0 & D_4 \end{bmatrix} = \begin{bmatrix} B_4 & C_4 \\ 0 & H_4 D_4 \end{bmatrix} = \begin{bmatrix} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & s_3 & * \\ 0 & 0 & 0 & s_4 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

After the nth step ($n =$ number of columns), we have a sequence of orthogonal matrices such that $Q_n \cdots Q_3 Q_2 Q_1 A = R$ is triangular.

$$A = \underbrace{(Q_1^T Q_2^T \cdots Q_n^T)}_{:=Q} R.$$

(Recall: products of orthogonal matrices is orthogonal.)

Matlab implementation

```
function [Q, R] = myqr(A)
[m, n] = size(A);
R = A;
Q = eye(m);
for k = 1:n
    % invariant: Q*R = A
    u = householder_vector(R(k:end, k));
    H = eye(length(u)) - 2*u*u';
    A(k:end,k:end) = H * R(k:end,k:end);
    Q(:, k:end) = Q(:, k:end) * H;
end
```

This is still not the final version of the algorithm!

Problem: as written here, it would have quartic cost ($\mathcal{O}(m^4)$ for a square matrix).

Optimizations

Huge optimization: don't form H: use $HA_k = A_k - 2u(u^T A_k)$.

This optimization brings down the cost from quartic to cubic.

Minor optimization: write s and zeros manually in `A(k:end, k)`.

Detail: if A is square, we can stop after step $n-1$; the matrix is already upper triangular.

Rectangular QR

If $m \gg n$, like for SVD, computing/storing Q is expensive.
Thin QR (like thin SVD): restrict to $Q_0 \in \mathbb{R}^{m \times n}, R_0 \in \mathbb{R}^{n \times n}$.

$$A = \begin{bmatrix} Q_0 & Q_c \end{bmatrix} \begin{bmatrix} R_0 \\ 0 \end{bmatrix} = Q_0 R_0.$$

There are two alternatives for handling Q_0 without forming the big matrix Q:

- Just return the $\mathbf{u}_i$'s: the implicit form $Q = Q_1 Q_2 \dots Q_n$, $Q_k = \text{blkdiag}(I_{k-1}, I - 2\mathbf{u}_k \mathbf{u}_k^T)$ is not an array full of numbers, but still you can perform operations such as matrix products at the same cost, or even cheaper.
- In particular, you can use the $\mathbf{u}_k$'s to compute $Q \left[\begin{smallmatrix} I_n \\ 0 \end{smallmatrix}\right] = Q_0$.

Cost

Computational cost of thin QR factorization via Householder reflectors (assuming $m \geq n$): $2mn^2 - \frac{2}{3}n^3 + \mathcal{O}(mn)$ flops.

More important than this exact formula is its behavior in two common regimes:

- $\frac{4}{3}n^3$ for square matrices ($m = n$).
- Scales like $2mn^2$ when $m \gg n$ (tall-thin A).

Book references: Trefethen-Bau, Lecture 10. Demmel, Sec. 3.4.1.

Exercises

1. Check the "boring algebra" in the proof that $H\mathbf{x} = \mathbf{y}$.
2. Is the QR factorization unique? (Hint: play with signs).
3. Can you identify (without computation) a QR of a matrix with zero structure $\begin{bmatrix} 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & * & * & * \\ 0 & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$? (Hint: swapping rows is an orthogonal transformation).
4. Show that if $A \in \mathbb{R}^{m \times m}$ is singular (non-invertible), then its QR factor R has a zero diagonal entry. (Hint: determinants!)
5. $\star$ Suppose that a matrix $A \in \mathbb{R}^{m \times m}$ is 'upper triangular plus one more diagonal', e.g., $\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$ (these are called *Hessenberg matrices*). Can you modify the algorithm so that it has cost only $O(m^2)$ for matrices with this structure?
6. For a square $A \in \mathbb{R}^{m \times m}$, what does the last step $k = m$ of QR factorization do?