Eigenvectors, Condition numbers, Normal equations, and Gram-Schmidt & QR Decomposition

to accompany and embellish Ch. 2 of **Data-Driven Modeling & Scientific Computation**

by J. Nathan Kutz

Reasons to embellish (a.k.a. Topics that deserve closer attention):

- Eigenvectors and eigenvalues (how to compute them)
- Condition number and solution reliability
- A condition number estimate based on eigenvalues
- Overdetermined systems, Least-squares solutions, and Normal equations
- Gram-Schmidt orthogonalization
- QR factorization/decomposition

Recall (or, better yet, forget) our brief discussion of eigenvectors and eigenvalues

Usual engineering descriptions:

- **Vector**: An entity with size/magnitude and direction__
- **Vector space**: a set of vectors so that any linear combination of vectors in the set is also included in the set

Multiplication by $n \times n$ matrix maps a vector $v \in \mathbb{R}^n$ to some other vector $w \in \mathbb{R}^n$

Eigenvectors of matrix A are the special vectors whose direction are not changed by A

If direction does not change, only magnitude can change $\implies$ matrix mult $\iff$ scalar mult: $Av = \lambda v$

Non-singular $n \times n$ matrix A has n non-zero eigenvalues

$Av = \lambda v \implies (A - \lambda I)v = 0$ which always has a solution $v = 0$

Non-trivial solutions require non-uniqueness $\implies (A - \lambda I)$ is singular

$|A - \lambda I| = 0 \implies$ deg. n characteristic polynomial with roots $\lambda_i, i \in 0, 1, \ldots, n-1$

All $\lambda_i \neq 0$ since $|A| = \prod \lambda_i \neq 0$

For distinct eigenvalues of a symmetric matrix, eigenvectors are orthogonal.

$$Av_i = \lambda_i v_i, Av_j = \lambda_j v_j$$

Multiply by v_j^T, v_i^T respectively:

$$v_j^T A v_i = \lambda_i v_j^T v_i, \quad v_i^T A v_j = \lambda_j v_i^T v_j$$

Transpose first equation with $A = A^T$:

$$v_i^T A v_j = \lambda_i v_j^T v_i$$

Subtract from second equation:

$$0 = (\lambda_j - \lambda_i) v_i^T v_j \implies \lambda_i \neq \lambda_j \implies v_i \perp v_j$$

For symmetric matrix $A = A^T$, eigenvectors with distinct eigenvalues are orthogonal.

All eigenvalues distinct $\implies$ n mutually orthogonal vectors form an orthogonal basis

Any vector in $\mathrm{Span}(A)$ can be written as linear combination of eigenvectors:

$$w = c_0 v_0 + c_1 v_1 + \ldots + c_{n-1} v_{n-1} = \sum_{i=0}^{n} c_i v_i$$

For non-symmetric, use other procedures to find orthogonal/orthonormal basis

We'll return to that, but first let's compute leading (largest magnitude) eigenvalue λ_0 and the associated eigenvector v_0 of a symmetric matrix A.

Symmetric matrix A has n orthogonal eigenvectors that span $\mathbb{R}^n$.

Can write any vector u as linear combination of the orthogonal basis: $w = \sum_{i=0}^{n-1} c_i v_i$

When we multiply Aw, each piece of the sum maintains its direction but has magnitude scaled by eigenvalue:

$$Aw = \sum_{i=0}^{n-1} \lambda_i c_i v_i$$

Repeat (matrix iteration) $\rightarrow A^m w = \sum_{i=0}^{n-1} \lambda_i^m c_i v_i$

With enough iterations (m large enough), λ_0^m dominates and gives approximation of v_0

How do we obtain λ_0 from there?

Multiply one more time and evaluate component ratios...

Important question: Should you ever compute A^m? Why or why not?

Important question: Should you ever compute A^m? Why or why not?

Avoid computing A^m because it takes a lot more computing effort.

Do matrix-vector multiplication instead:

$$A^m v = A^{m-1}\left[Av\right] = A^{m-2}\left[A(Av)\right] = \ldots = A\left[A(\ldots(Av))\right]$$

Note that $\left[\,\cdot\,\right]$ is always a vector... No matrix-matrix multiplication occurs.

Connect eigenvalues to **condition number** $\kappa(A)$:
Condition number quantifies expected reliability of the solution of a linear system.

Rule of thumb: For $\epsilon_M = 10^{-t}$ and condition number $\kappa(A) = 10^p$,
expect solution correct to $\approx t - p$ digits.

`float64` $\implies \epsilon_M \approx 10^{-16}$
For systems with condition number $\kappa(A) > 10^{12}$ solutions have, at best, only a few correct digits.

Systems with $\kappa(A) \gtrapprox 10^{16}$ may be considered "numerically singular".

So how do we compute the condition number?

There are condition number estimates based on eigenvalues: $\kappa(A) \approx |\lambda_{max}/\lambda_{min}|$

Knowledge of just the "extreme" eigenvalues is helpful.

Note that for more general matrices, eigenvalues give way to singular values (More later as time allows...)

Note regarding determinant vs. condition number: Both are related to eigenvalues, but...

How does $Det(A)$ relate to λ_i?

$Det(A) = \prod_{i=0}^{n-1} \lambda_i$

But size of $|A| = Det(A)$ does not directly determine size of $\kappa(A)$.

For $A = 10\, I_{10\times 10},\ Det(A) = 10^{10}$, but $\kappa(A) =$???

Think in terms of **spectrum** (set of eigenvalues)

$$\lambda_{max} = 1, \lambda_{min} = 1 \implies \kappa(10\, I_{10\times 10}) = 1$$

This is an example of a well-conditioned matrix with a large determinant, so...

Det can be helpful sometimes but really need to look at condition number κ.

Back to **overdetermined systems and least squares solutions**:

Overdetermined system: $Ax = b$ where A is $M \times n$ and x, b are n-vectors

(big M and little n to indicate a lot of equations in fewer variables)

Best fit straight line involves as many data points as you like, but only 2 variables.

It is possible to just multiply by A^T to get the normal equations: $A^T Ax = A^T b$

Normal equations have nice symmetric matrix $A^T A$, so this is all good news, right!?

Stop to think about how the condition number $\kappa(A^T A)$ compares with $\kappa(A)$...

Get a big hint from the square symmetric case ($M = n$, $A^T = A$):

- A^T has the same eigenvalues and eigenvevctors as A (Think about the characteristic polynomial...)
- Because eigenvectors coincide, eigenvalues of the product matrix $A^T A$ are products of the eigenvalues of A
- A^T contributes to κ just the same as A
- Condition number gets squared! (even in the "friendliest" case)

"Normal" problem $A^T A x = A^T b$ is numerically **much** more sensitive, so an alternative approach is taken: **QR Factorization**

QR factorization

Factor a matrix into the product of:

- Orthonormal matrix Q with $Q^T = Q^{-1}$ so $Q^TQ = QQ^T = I$
  (columns of Q are mutually orthogonal unit vectors)
- Upper triangular matrix R

R is to remind us that orignal matrix and this factor can both be rectangular
(and we need to handle the non-square case for overdetermined systems)

How do we obtain a QR factorization? One approach uses a method you may have already seen... **Gram-Schmidt Orthogonalization**

Gram-Schmidt Orthgonalization

Start with any set of vectors $A = \{a_0, a_1, \ldots, a_{n-1}\}$ from a given vector space

Produce:

- New set $U = \{u_0, u_1, \ldots, u_{n-1}\}$ that is mutually orthogonal
- Orthonormal set $E = \{e_0, e_1, \ldots . e_{n-1}\}$; $e_i = \frac{u_i}{|u_i|}$

both spanning the same subspace: $Span(A) = Span(U) = Span(E)$

$u_i \cdot u_j = 0$ for $i \neq j$

$$e_i \cdot e_j = \delta_{i,j}; \qquad \delta_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Idea of **Gram-Schmidt** is fairly straightforward:

0. Initialize A with the given set of vectors and U and E as empty.
1. Remove the first vector from A; insert it as the first vector in U so $u_0 = v_0$.
   Insert normalized version as first element in E; i.e. $e_0 = u_0 / \|u_0\|$
2. Take the next vector a_i from A; subtract off the components in the direction of $u_0, \ldots, u_{i-1}$; insert what is left as u_i and normalize to get e_i.
3. Continue until V is empty.

Remaining issue:
How to compute the component of v_i in the direction of u_j so that it can be removed.

Projection of v onto u

- Component of v along direction of u
- The size of a vector's component in a direction is computed as inner product of the vector with a unit vector in the component direction

Component of v in direction of u has:

- directed length $v \cdot \hat{u} = v \cdot u/|u| = v \cdot e$
- direction u (or $e = u/|u|$)

$$\begin{aligned} \text{Component of } v_i \text{ along } u_j &= (v_i \cdot e_j)e_j \\ &= \frac{v_i \cdot u_j}{u_j \cdot u_j} u_j \end{aligned}$$

Gram-Schmidt Formulas

$$
\begin{aligned}
u_0 &= a_0, & e_0 &= u_0/|u_0| \\
u_1 &= a_1 - (a_1 \cdot e_0)e_0, & e_1 &= u_1/|u_1| \\
u_2 &= a_2 - (a_2 \cdot e_0)e_0 - (a_2 \cdot e_1)e_1, & e_2 &= u_2/|u_2| \\
u_k &= a_k - (a_k \cdot e_0)e_0 - (a_k \cdot e_1)e_1 - \ldots - (a_k \cdot e_{k-1})e_{k-1}, & e_k &= u_k/|u_k| \\
u_{n-1} &= a_{n-1} - \sum_{j=0}^{n-2}(a_{n-1} \cdot e_j)e_j, & e_j &= u_{n-1}/|u_{n-1}|
\end{aligned}
$$

In matrix language, **Gram Schmidt** becomes a QR **Factorization** where we reconstruct the a_i by summing components along the e_i directions:

$$A = [a_0|a_1|\cdots|a_{n-1}] = \underbrace{[e_0|e_1|\cdots|e_{n-1}]}_{\text{orthogonal } Q} \underbrace{\begin{bmatrix} a_0 \cdot e_0 & a_1 \cdot e_0 & \cdots & a_{n-1} \cdot e_0 \\ 0 & a_1 \cdot e_1 & \cdots & a_{n-1} \cdot e_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1} \cdot e_{n-1} \end{bmatrix}}_{\text{rectangular } R} = QR$$

QR factorization produces:

- Matrix Q that is easy to invert by transpose (and conjugate for complex values)
- Matrix R that is "triangular" (possibly with additional rows of zero at the bottom).

Factorization of A into QR provides way to solve $Ax = b$ that applies to non-square A

$$
\begin{aligned}
Ax &= b \\
QRx &= b \\
Q^{-1}QRx &= Q^{-1}b \quad (\text{ Note:} Q^{-1}Q = I \text{ without matrix mult.}) \\
Rx &= Q^T b
\end{aligned}
$$

which is amenable to a triangular (backsubstitution) solver.

We should run into QR factorizations again later in the quarter...

And feel free to check out Kutz Section 2.5 on using eigenanalysis for facial recognition.