Lecture 26: Eigenvalue Computation

- Given (symmetric) matrix A, find the top eigenpair (evalue/evector)? find the top k? Find all of them?

- Recall: even if $A \in \mathbb{Q}^{n \times n}$ the results may be irrational, so get approximations that converge to right answer $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

- Focus on symmetric matrices (so have $A = Q \Lambda Q^T$ where
  - Q is an orthogonal matrix $(QQ^T = Q^TQ = I)$ and cols are eigenvectors
  - $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2 \ldots \lambda_n)$ are the eigenvalues, and these are real-valued.

Schur decomposition for this case

— × —

The First Algorithm: Power Iteration

$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$

$v^0 \leftarrow$ random unit vector say

repeat: $v^{t+1} \leftarrow A v^t$ (renormalize if you like, to control the length)

Suppose the eigenbasis is $q_1 q_2 \ldots q_n$

then $v^0 = \sum_{i=1}^{n} c_i q_i$ (suppose)

$$\Rightarrow v^t = A^t\left(\sum c_i q_i\right) = \sum c_i \lambda_i^{2t} q_i$$

$$= \lambda_1^{2t}\left(c_1 q_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^{2t} \cdot c_2 q_2 + \ldots\ldots + \left(\frac{\lambda_n}{\lambda_1}\right)^{2t} c_n q_n\right)$$

Note: if $|\lambda_2| < |\lambda_1|$ then the terms except the first fade down (relatively) whereas the first term $c_1 q_1$ remain fixed (up to this scaling by λ_1^{2t}) $\Rightarrow$ converge to q_1.

(also $c_1 \neq 0$ which happens w.p.)

What about the smallest eigenvalue?

Inverse iteration

Sps A has full rank then A^{-1} has evals $1/\lambda_1, 1/\lambda_2 \cdots 1/\lambda_n$.

So if λ_n was smallest in magnitude (say $|\lambda_1| \geq |\lambda_2| \cdots > |\lambda_n|$) strict

then now $1/\lambda_n$ is largest in magnitude, and power iteration is good for it

So at each step want

$v^{t+1} \leftarrow (A^{-1}) v^t$, or equivalently $A v^{t+1} = v^t$

linear system solve

Gaussian Elim, e.g.

e.g. if $A = LU$ (via Gaussian elim)

lower triangular, upper triangular.

then solve $Lz = v^t$ and then $Uv^{t+1} = z$.

Now if $|\lambda_n| \ll |\lambda_{n-1}|$ then get good convergence!

Shifting:

- Another idea is that evals of $A - \mu I$ are $\{\lambda_i - \mu\}_{i=1}^{n}$.
- So if we have a good guess μ for some λ_i (but μ is not an eigenvalue itself)

  then solve for evals of $(A - \mu I)^{-1}$.

  Since $|\lambda_i - \mu|$ is small, the gap between this and $|\lambda_j - \mu|$ is large,

  and the power method will converge rapidly.

Widely used, apparently numerically stable as well. (Trefethen-Bau, Golub-van Loan)

How do you get an estimate of an eigenvalue?

Use the inverse iteration itself, and the Rayleigh quotient.

Recall: $R(x) := \dfrac{x^T A x}{x^T x}$

as a function of λ

Fact: given any x, the minimizer of $\|(A-\lambda I)x\|_2$ is $\lambda = R(x)$.

So if x were an eigenvector, $R(x)$ would be the eigenvalue and this expression (the norm) would be zero.

But if x is not an eigenvector, the norm may be non-zero.

Still can think of $R(x)$ as an extension of evalue to all $x \in \mathbb{R}^n$.

Anyhow: $R(x)$ gives a way to get a scalar from x.

Rayleigh Quotient Iteration:

$v^0 \leftarrow$ random unit vector. $\lambda^{(0)} = R(v^0)$

repeat
- $(A - \lambda^{(k-1)} I)x = v^{(k-1)}$ // Inverse power iteration
- $v^{(k)} = x/\|x\|$ // normalize
- $\lambda^{(k)} = R(v^{(k)})$.

According to the texts, RQI "almost always converges"
Also its convergence is cubic (error drops from $\varepsilon \to \varepsilon^3$ every timestep !!)

Here's a heuristic argument.

by calculations: $\nabla R(x) = \frac{2}{x^T x}(Ax - R(x)x)$

(1) $R(x)$ is a smooth function on S^{n-1}. Sps we are at x, and q^* is an eigenvector

then $\nabla R(q^*) = 0 \Rightarrow R(x) - R(q^*) = O(\|x - q^*\|^2)$

(2) Sps λ^* is corresponding eigenvalue (and it is simple, no repeated eigenvalues here)

$\Rightarrow |\lambda^{(k)} - \lambda^*| = O(\varepsilon^2)$ if $\|v^{(k)} - q^*\| \le \varepsilon$

(3) then $\|v^{(k+1)} - q^*\| \le O(|\lambda^{(k)} - \lambda^*| \times \|v^k - q^*\|)$

$\le O(\varepsilon^2 \cdot \varepsilon) = O(\varepsilon^3)$.

b/c we're really scaling most mass up by $\frac{1}{\lambda^{(k)} - \lambda}$ when we mult by $\frac{1}{A - \lambda I}$, so rescaling gives $O(\lambda^{(k)} - \lambda^*)$

— X —

All these were computing one eigenpair, but what about the entire decomposition?

Here's a generalization of the idea.

But first, let's recall another idea: the QR factorization ← considered one of the top 10 algos of the 20th century by Engineering articles.

write matrix $A = QR$ ← upper triangular matrix

↑ orthonormal columns

$$\begin{pmatrix} | & & | \\ A_1 & \cdots & A_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & & q_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ 0 & & & r_{nn} \end{pmatrix}$$

So A_i is written as the combination of the first i cols of Q.

- Well known factorization: Gram Schmidt orthogonalization

where $q_i = A_i - \sum_{j<i} \langle A_i, q_j \rangle . q_j$, renormalized to be unit vector.

↑ now q_i are even orthonormal.

people don't use it, numerical issues.

use modified versions, or Householder triangularization (see books).

— X —

One other fact, very useful.

Given an invertible matrix S, A and $S^{-1}AS$ have same eigenvalues.

indeed

$Ax = \lambda x \iff (S^{-1}AS)(S^{-1}x) = (S^{-1}x)\lambda$.

A and $S^{-1}AS$ are called $\boxed{\text{similar.}}$ $\Rightarrow$ OK to replace A by some similar matrix.

OK, back to computing multiple eigenvalues/eigenvectors at once. (say r at once)

$Q^{(0)} \leftarrow$ orthogonal columns matrix $\in \mathbb{R}^{n\times r}$.

$$\underline{\text{repeat}}_{k=1,2..} : \begin{cases} Z = A\,Q^{(k-1)} \\ Q^{k} \cdot R^{k} = Z \end{cases} \qquad // \text{ QR factorization of } Z$$

Orthogonal Iteration:

Note: if $r=1$, this is power iteration.

In fact consider $Q^{(0)}e_1$, $Q^{(1)}e_1$, ... $Q^{(k)}e_1$ (even when $r>1$).

this is the run of the power method on $Q^{(0)}e_1$.

Thm: performing this operation, and assuming that $|\lambda_{j+1}| < |\lambda_j|$ means this process converges to the top r eigenvalues/vectors of A.

$\begin{cases} q_2^{(k)} \text{ is the vector where we're removing } q_1^{(k)} \text{ each time} \ldots \\ q_3^{(k)} \text{ - - - - - - } q_1^{(k)} \,\&\, q_2^{(k)} \ldots\ldots \text{ etc.} \end{cases}$

(subtracting the other higher evectors out prevents them from all converging to q_1, top evector)

— x —

So sps we take $r=n$.

$Q^{(0)}$ = orthogonal cols matrix $\in \mathbb{R}^{n\times n}$.

$$\text{repeat } \forall k\geq 1 : \begin{cases} Z = A\,Q^{(k-1)} \\ Q^{(k)}R^{(k)} = Z \end{cases} \qquad // \text{ QR decomp.}$$

Can be rewritten as: the QR algorithm (as opposed to the QR decomposition)

$A^{(0)} = A$

For $k \geq 1$

$Q^{(k)}R^{(k)} = A^{(k-1)}$ // QR decomp of $A^{(k-1)}$.

$A^{(k)} = R^{(k)}Q^{(k)}$

So take the QR decomposition of $A^{(k-1)}$, flip the two, and multiply them!!?!

Two obvious questions:

① is the matrix $A^{(k)}$ even similar to A?

YES.

$$A^{(k)} = R^{(k)} Q^{(k)}$$
$$= (Q^{(k)^{-1}} Q^{(k)})\, R^{(k)} Q^{(k)}$$
$$= Q^{-1}(QR)Q \;=\; Q^{-1} A^{(k-1)} \cdot Q$$

so similar to $A^{(0)}$ by induction $= A$

Good. But why does it compute the eigenvalues? why will $Q^{(k)} \to$ the eigenvectors?

② Equivalence of Orthogonal Iteration & QR decomposition

$\underline{Q}^{(0)} = I$ $\qquad A^{(0)} = A$

$Z = A\,\underline{Q}^{(k-1)}$

$\underline{Q}^{(k)} R^{(k)} = Z$

$A^{(k)} = (\underline{Q}^{(k)})^T A\, \underline{Q}^{(k)}$

not part of algo

$A^{(0)} = A$ $\qquad \underline{Q}^{(0)} = I$

$Q^{(k)} R^{(k)} = A^{(k-1)}$

$A^{(k)} = R^{(k)} Q^{(k)}$.

$\underline{Q}^{(k)} = \underline{Q}^{(k-1)} \cdot Q^{(k)}$.

not part of algo

Inductively: show that the two are the same.

— x —

QR algorithm (John Francis, 1961) is called by [Trefethen-Bau] as one of the crown jewels of numerical analysis, and nominated as one of the top 10 algos of 20th century.

However: to get good guarantees, need other ideas. E.g. shifting etc; avoid Gram-Schmidt, use Householder, ...

But still, get good results in many cases.

And does not get simpler than this to state! ☺

(slow otherwise or may not converge. E.g. for $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$)

Lots more in books, see details in [TB] or [GvL].

Python notebooks also off webpage.

—— x ——

- Did not give many proofs of convergence
  - almost no discussion of stability here
  - bit precision issues.

Lots of interesting / deep qns!

All these for another day, another class.