Math 471 (Numerical methods)

Chapter 4 . Eigenvalues and Eigenvectors

Overlap §4.1–4.3 of Bradie

§4.0 Review on eigenvalues and eigenvectors

Definition. A nonzero vector $\vec{v}$ is called an eigenvector of A if there exists a scalar number λ – called an eigenvalue – such that

$$A\vec{v} = \lambda\vec{v}.$$

For simplicity, we always assume there exist n eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and assume

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > ...$$

but nevertheless one should be aware of repeated eigenvalues in practice (which is indeed directly related to repeated roots of a polynomial).

Why eigenvalues and eigenvectors? There can be tens of answers. Roughly speaking, eigenvalues provide a way of characterizing a matrix using scalars, and also eigenvectors help complete the information. Given an n-by-n matrix A, if we can find n eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and associated n eigenvectors $\vec{v}_1, \vec{v}_2, ..., \vec{v}_n$ that are *linearly independent*, then

$$A = V\Lambda V^{-1} \text{ for } \Lambda := \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \text{ and } V = \begin{pmatrix} \vec{v}_1, \vec{v}_2, ..., \vec{v}_n \end{pmatrix} \tag{1}$$

This is the diagonalization of A. Note that not all square matrices are diagonalizable but we will leave such exceptions to higher level courses.

- Relation to growth rate of ODE solutions

$$\frac{d\vec{x}}{dt} = A\vec{x} \implies \vec{x}(t) = Ve^{\Lambda t}V^{-1}\vec{x}(0) \tag{2}$$

where $e^{\Lambda t} = \begin{pmatrix} e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_n t} \end{pmatrix}$.

- It can be used for the so-called principle components, i.e. the eigenvectors $\vec{v}_1, \vec{v}_2, ...$ associated with the largest eigenvalues in absolute value. Given any n-by-1 vector $\vec{x}$, by (1),

$$A\vec{x} = (V\Lambda)(V^{-1}\vec{x}). \tag{3}$$

  Thus, mult. A with $\vec{x}$ yields a combination of columns of $V\Lambda$ for which the weights are given in the vector $V^{-1}\vec{x}$. Among all columns of $V\Lambda$, we see that the principle component $\vec{v}_1$ is the most amplified since the diagonal matrix Λ rescales the magnitude of columns in V.

- The principle compenent, in a population model for example, indicates the most prominent population distribution that majorizes all other eigenvectors. In statistics, the principle component of the correlation matrix indicates the most correlated way of combining variables into one index.

- In theoretical PDE and numerical PDE, the eigenvalue structure tells e.g. the characteristic frequencies of an instrument, the dissipation rate of heat, the direction along which the strongest strain incurred when a building experiences external force.

- etc.

§ 4.1 The power method.

Direct computation of $\det(\lambda I - A) = 0$ is highly expensive and unstable. Iterative method is the preferred way.

Proposition. Let matrix A have the diagonalization (1). Assume $\lambda_1 > 0$ has the largest absolute value – otherwise, simply replace A with $-A$. Let $\vec{x} \neq 0$. Then, as k approaches infinity,

$$\lim_{k\to\infty} \frac{A^k\vec{x}}{\lambda_1^k} = V \begin{pmatrix} 1 & & \\ & 0 & \\ & & \ddots \\ & & & 0 \end{pmatrix} V^{-1}\vec{x}$$

Proof. Formula (1) implies that $A^k = V\Lambda V^{-1}V\Lambda V^{-1}...V\Lambda V^{-1} = V\Lambda^k V^{-1}$. Therefore,

$$\frac{A^k\vec{x}}{\lambda_1^k} = V\frac{\Lambda^k}{\lambda_1^k}V^{-1}\vec{x} = V\begin{pmatrix} 1 & & & \\ & (\lambda_2/\lambda_1)^k & & \\ & & \ddots & \\ & & & (\lambda_n/\lambda_1)^k \end{pmatrix}V^{-1}\vec{x}. \tag{4}$$

When $k \to \infty$, all but the first diagonal terms $(\lambda_i/\lambda_1)^k$ have base less than 1 in ters of absolute value. Only the first diagonal term will survive the limiting process as $k \to \infty$.

DONE.

This example shows that, as $t \to \infty$, the largest eigenvalue dominates the growth of $\vec{x}(t)$. It actually hints a numerical method to compute the largest eigenvalue!

• **Power method.**

The idea behind the power method is extremely simple: start with an initial vector $\vec{x}_0$ and iteratively multiply it with A: $\vec{x}_0, A\vec{x}_0, A^2\vec{x}_0, ..., A^k\vec{x}_0....$ When k get large enough, the principle component becomes the most amplified (for a factor of λ_1^k) and therefore outgrows all other eigenvetors (amplified for factors of λ_i^k which are less than λ_1^k in absolute values). A proper rescaling procedure then brings $A^k\vec{x}_0$ to a moderate size, which serves as an approximation for the principle component $\vec{v}_1$.

The mathematical formulation of the above explanation has been done in (4). For convenience, we rewrite it in terms of the columns of V, i.e. the eigenvectors of A, (also let $\vec{y} = V^{-1}\vec{x}$)

$$\begin{aligned} &\frac{A^k\vec{x}}{\lambda_1^k} = y_1\vec{v}_1 + y_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\vec{v}_2 + y_3\left(\frac{\lambda_3}{\lambda_1}\right)^k\vec{v}_3 + ... + y_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\vec{v}_n \\ &\text{which converges to} \quad y_1\vec{v}_1 \quad \text{at rate} \qquad O(|\lambda_2/\lambda_1|^k). \end{aligned} \tag{5}$$

But the problem is we don't know λ_1 *a priori* so (5) is not practical at all. Instead, we rescale $A^k\vec{x}$ by its norm.

Algorithm(Power method)

```
% pick initial guess x
for k=1:N
   x=A*x; % increase the power of A
   x=x/max(abs(x)); % rescale x by its infinity norm (or any other norm you like)
end
```

```
x % the approximate eigenvector
(x'*A*x)/(x'*x) % the approximate eigenvalue
```

Note. The last line uses the *Rayleigh quotient* to compute the approximate eigenvalue. Given any nonzero vector $\vec{x}$, the associated Rayleigh quotient is defined as

$$\frac{\vec{x}^T A\vec{x}}{\vec{x}^T\vec{x}}.$$

If $\vec{x}$ is an exact eigenvalue, then its Rayleigh quotient is exactly the associated eigenvalue. (Why? Exercise for the reader.)

What about convergence rate? Equation (5) suggests that the non-principle terms of $A^k\vec{x}_0/\lambda_1^k$ decay to zero at a rate $\sim O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ so the convergence rate should be linear. In the above code, however, we essentially compute $\dfrac{A^k\vec{x}}{\|A^k\vec{x}\|}$ — this is sometimes called normalization.

Proof of linear convergence for the Power method. Let λ_1 be the leading eigenvalue with the largest absolute value. (Note. If $-\lambda_1$ is also an eigenvalue, then slightly more steps are needed.) Let λ_2 be the next largest eigenvalue in absolute value. Set $r = |\lambda_2|/|\lambda_1|$.

By equation (5), we find

$$\begin{aligned} A^k\vec{x} &= \lambda_1^k\left(y_1\vec{v}_1 + y_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\vec{v}_2 + y_3\left(\frac{\lambda_3}{\lambda_1}\right)^k\vec{v}_3 + \ldots + y_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\vec{v}_n\right) \\ &= \lambda_1^k(y_1\vec{v}_1 + O(|\lambda_2/\lambda_1|^k)) = \lambda_1^k(y_1\vec{v}_1 + O(r^k)) \end{aligned}$$

So, after rescaling,

$$\begin{aligned} \frac{A^k\vec{x}}{\|A^k\vec{x}\|} &= \frac{\lambda_1^k(y_1\vec{v}_1 + O(r^k))}{\|\lambda_1^k(y_1\vec{v}_1 + O(r^k))\|} \\ &= \frac{\lambda_1^k}{|\lambda_1^k|}\frac{y_1\vec{v}_1 + O(r^k)}{\|y_1\vec{v}_1 + O(r^k)\|} \\ &= \frac{\lambda_1^k}{|\lambda_1^k|}\left(\frac{y_1}{|y_1|}\frac{\vec{v}_1}{\|\vec{v}_1\|} + O(r^k)\right) \end{aligned}$$

Since $\dfrac{\lambda_1^k}{|\lambda_1^k|}$ and $\dfrac{y_1}{|y_1|}$ are scalars with absolute value 1, we conclude that

$$\frac{A^k\vec{x}}{\|A^k\vec{x}\|} = C_k\vec{v}_1 + O(r^k) \quad \text{for} \quad r = |\lambda_2/\lambda_1|$$

where $|C_k| = 1$. If $\lambda_1 > 0$, then all C_k have the same sign; otherwise, signs of C_k are alternating.

We see from above that the convergence rate of the power method highly depends on the ratio $r = |\lambda_2/\lambda_1|$, which can be very slow for $r \approx 1$. Nevertheless, the power method can only find the largest eigenvalue. What about the rest of it?

§ 4.2 Inverse power method

Observe that, if A is equipped with n eigenpairs $(\lambda_1, \vec{v}_1), (\lambda_2, \vec{v}_2), ..., (\lambda_n, \vec{v}_n)$ and A is invertible, then its inverse A^{-1} is equipped with eigenpairs $(\lambda_1^{-1}, \vec{v}_1), (\lambda_2^{-1}, \vec{v}_2), ..., (\lambda_n^{-1}, \vec{v}_n)$. Why?

$$A\vec{v} = \lambda\vec{v} \implies \vec{v} = A^{-1}(\lambda\vec{v}) \implies \lambda^{-1}\vec{v} = A^{-1}\vec{v}$$

Thus, λ_n^{-1} becomes the largest eigenvalue of A^{-1} and applying the power method to A^{-1} leads to convergence to $(\lambda_n^{-1}, \vec{v}_n)$. This way, we can find the *smallest* eigenvalue of the original A.

By a very similar argument, $A - \mu I$ is equipped with $(\lambda_1 - \mu, \vec{v}_1), (\lambda_2 - \mu, \vec{v}_2), ..., (\lambda_n - \mu, \vec{v}_n)$. Here, we call μ a shift in the spectrum. One step further, $(A - \mu I)^{-1}$ is equipped with $((\lambda_1 - \mu)^{-1}, \vec{v}_1), ((\lambda_2 - \mu)^{-1}, \vec{v}_2), ..., ((\lambda_n - \mu)^{-1}, \vec{v}_n)$. What is the significance of such transformation? The answer is, if the shift μ is chosen to be very close to one particular eigenvalue, say λ_m, then performing the power method on $(A - \mu I)^{-1}$ will converge to $((\lambda_m - \mu)^{-1}, \vec{v}_m)$ at a linear rate $O(r^k)$ with

$$r = \frac{\max_{j \neq m}\{|\lambda_j - \mu|^{-1}\}}{|\lambda_m - \mu|^{-1}}$$

Here, the numerator is the second largest eigenvalue of $(A - \mu I)^{-1}$, only smaller than $(\lambda_m - \mu)^{-1}$.

• Example. Let A be a 3-by-3 matrix with exact eigenvalues $\lambda_1 = 5, \lambda_2 = 2, \lambda_3 = -1$. Then the inverse power method converges to $\lambda_3, \vec{v}_3$ at rate $O(r^k)$ for

$$r = \left|\frac{\lambda_2^{-1}}{\lambda_3^{-1}}\right| = 0.5$$

The shifted inverse power method with $\mu = 1.4$ converges to $\lambda_2, \vec{v}_2$ with

$$r = \frac{|\lambda_3 - \mu|^{-1}}{|\lambda_2 - \mu|^{-1}} = \frac{1}{6}.$$

Note that $|\lambda_1 - \mu|^{-1}$ is the smallest eigenvalue of $A - \mu I$ and therefore is not used in the computation of r.

Algorithm(Shifted Inverse Power Method)

```
% pick initial guess x and a shift u.
% Set u=0 to find the smallest eigenvalue

uI=u*eye(size(A,1)); % create a shift matrix with the same size of A

for k=1:N
   x=inv(A-uI)*x; % Here, a lot of improvement can be made if we replace
                  % inv(A-uI) with fast algorithms
   x=x/norm(x,2); % rescale x by its 2 norm (or any other norm you like)
end
x % the approximate eigenvector
(x'*A*x)/(x'*x) % the approximate eigenvalue. Directly compute
                % eigenvalue for A, not A-uI
```

Please note that the `inv(A-uI)` part could be computationally expensive, e.g. $O(n^2)$ dense matrix. In practice, always consider replacing it with faster methods. For instance, we can perform the LU factorization on $A - \mu I$ once at the beginning of the code and solve $(A - \mu I)x^{(k+1)} = x^{(k)}$ with $O(m)$ operations in each iteration step. Here m is the number of nonzero entries in the LU factorization of $A - \mu I$.

§ 4.2 (cont...) Rayleigh quotient iteration

The idea of Rayleigh quotient iteration is based on the shifted inverse power method. Instead of using a fixed shift μ, this method updates the shift at each iteration using the Rayleigh quotient on the latest version of the approximate eigenvector $\vec{x}$,

$$\mu \leftarrow \vec{x}^T A\vec{x}/\vec{x}^T\vec{x}$$

Remember that the Rayleigh quotient gives an approximate eigenvalue and therefore should be close to the actual one. So we make μ closer and closer to λ and therefore make $(\lambda - \mu)^{-1}$ larger and larger, well separated from the second largest eigenvalue. This will improve convergence rate, i.e. reduce r.

A more refined analysis reveals that the convergence rate of the Rayleigh quotient iteration is cubic

$$|\lambda^{(k+1)} - \lambda| \leq C|\lambda^{(k)} - \lambda|^3$$

A drawback of this method is that now we have to solve a brand new linear system in every iteration $(A - \mu^{(k)}I)x^{(k+1)} = x^{(k)}$ since the shift $\mu^{(k)}$ is updated in each iteration. On the other hand, the inverse power method always solve a system with the same LHS $(A - \mu I)x^{(k+1)} = x^{(k)}$ and thus can reuse the same LU factorization obtained at the beginning of the code. In other words, the inverse power method uses less operations *per iteration* than the Rayleigh quotient iteration. The trade-off is that the inverse power method only converges at a linear rate.

Note. If A is tridiagonal, then solving $(A-\mu^{(k)}I)x^{(k+1)} = x^{(k)}$ takes $O(n)$ operations which is comparable to the inverse power method. So, at least, the Rayleigh quotient method outperforms the inverse power method for tridiagonal system.

§4.3 **Deflation: finding intermediate eigenvalues w/o shift**

The efficiency of the shifted inverse power method and Rayleigh quotient iteration highly relies on how close the "guessed" value of μ is to the actual eigenvalue sought after. If we have no information *a priori* about the intermediate eigenvalues $\lambda_2, \lambda_3, ...$, these methods become unpractical and sometimes misleading. A remedy is to find $\lambda_1, \vec{v}_1$ first and apply the (inverse) power method again with an additional surgery to remove the $\vec{v}_1$ part from the computation.

For simplicity, let's use a symmetric matrix A with real enties. It is a theorem that a real symmetric matrix has n real eigenvalues and moreover, the associated eigenvectors form an orthogonal basis of R^n. That is, for *any* vector $\vec{x}$, we can decompose it

$$\vec{x} = y_1\vec{v}_1 + y_2\vec{v}_2 + ... \tag{6}$$

where ther coefficients are given by

$$y_i = \frac{\vec{x}^T\vec{v}_i}{\|\vec{x}\|_2}$$

and if all the eigenvectors are rescaled to unit norm, then the coeffiecients are simply $y_i = \vec{x}^T\vec{v}_i$.

Note. Make sure the right formula is used to compute y_i, depending on whether or not $\|\vec{v}_i\|_2 = 1$.

Now that, by (6), the $\vec{v}_1$ part of any vector can be calculated, we can remove this part from our calculation in the power method. Suppose at the k-the step, we arrive at $\vec{x}^{(k)} = y_1\vec{v}_1 + y_2\vec{v}_2 + ...$ Do a simple deduction

$$x^{(k)} \leftarrow x^{(k)} - y_1\vec{v}_1$$

and proceed to the next iteration with a new $x^{(k)} = y_2\vec{v}_2 + ...$

Indeed, if this removal procedure is done initially on $\vec{x}_0$ so that $\vec{x}_0 = y_2\vec{v}_2 + ...$, then *theoretically* $A^k\vec{x}_0 = y_2\lambda_2^k\vec{v}_2 + ...$ are all free of $\vec{v}_1$. So why do we have to remove $\vec{v}_1$ during every iteration? The answer is, in practice, some small $\vec{v}_1$ part always emerges due to round-off error. Without the removal procedure, such part will eventually outgrow the rest and the computation converges to $\lambda_1, \vec{v}_1$ again.

Generalization to find $\lambda_3, \lambda_4, ...$ Once the first two eigenpairs are calculated, we may go on to find $(\lambda_3, \vec{v}_3)$. This time, remove both $\vec{v}_1$ and $\vec{v}_2$ parts from the computation, which looks like this

```
% Suppose the 2-norm of v1 and v2 are rescaled to unit

for i=i:N
.............
y1=x'*v1; y2=x'*v2;
x=x-y1*v1-y2*v2;
.........
end
```