ALGORITHMS FOR EIGENVALUES AND EIGENVECTORS

David F. Gleich

November 25, 2019

These notes are a collection of Trefethen and Bau (Lecture 28) and my own (later) interpolations. The result on the convergence of the power method is due to Luca Trevisan (http://theory.stanford.edu/~trevisan/expander-online/lecture03.pdf) which was pointed out to me by Petros Drineas.

In these notes, we will frequently assume that $\boldsymbol{A}$ is symmetric. Some of the methods generalize to non-symmetric matrices (e.g. the power method), but there are often complexities involved. Please consult Golub and van Loan for more details on these cases.

1 USEFUL PROPERTIES OF EIGENVALUES.

LEMMA 1 *Let $\boldsymbol{A}$ be a symmetric matrix. If $p(\boldsymbol{A})$ is a polynomial of $\boldsymbol{A}$, then the eigenvectors of $\boldsymbol{A}$ and $p(\boldsymbol{A})$ are the same, but the eigenvalues change.*

Proof Note that $p(\boldsymbol{A}) = c_0\boldsymbol{I} + c_1\boldsymbol{A} + c_2\boldsymbol{A}^2 + \cdots + c_k\boldsymbol{A}^k$ for some coefficients $c_0, \ldots, c_k$. Then $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$ is the eigenvalue decomposition and $\boldsymbol{A}^j = \boldsymbol{V}\boldsymbol{D}^j\boldsymbol{V}^T$. Hence $p(\boldsymbol{A}) = \boldsymbol{V}p(\boldsymbol{D})\boldsymbol{V}^T$. ∎

(The same proof holds for non-symmetric matrices with the Jordan canonical decomposition instead.)

LEMMA 2 (Gerschgorin disks, Golub and van Loan Theorem 7.2.1) *Let $\boldsymbol{A}$ be any matrix. Suppose that $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{F}$ where $\boldsymbol{D}$ is diagonal and $\boldsymbol{F}$ has zero diagonal. (So $\boldsymbol{D}$ are the diagonal entries of $\boldsymbol{A}$ and $\boldsymbol{F}$ are all the other entries. Then the eigenvalues of $\boldsymbol{A}$ are contained within the set:*

$$\lambda(\boldsymbol{A}) \subseteq \bigcup_{i=1}^{n} G_i$$

where G_i is the ith Gerschgorin disk:

$$G_i = \{z \in \mathbb{C} : |z - D_{i,i}| \leq \sum_j |F_{i,j}|.$$

Proof Let $\lambda \in \lambda(\boldsymbol{A})$ be any eigenvalue that is not equal to $D_{i,i}$ for any i. Then $\boldsymbol{A}\mathbf{x} = \lambda\mathbf{x}$ yields:

$$(\boldsymbol{D} + \boldsymbol{F} - \lambda I)\mathbf{x} = 0 \Leftrightarrow \mathbf{x} = (\boldsymbol{D} - \lambda\boldsymbol{I})^{-1}\boldsymbol{F}\mathbf{x}.$$

From the latter fact, we have:

$$1 \leq \|(\boldsymbol{D} - \lambda\boldsymbol{I})^{-1}\boldsymbol{F}\|$$

for any sub-multiplicative norm. If this is the ∞-norm, then

$$\|(\boldsymbol{D} - \lambda\boldsymbol{I})^{-1}\boldsymbol{F}\|_\infty = \sum_j \frac{|F_{k,j}|}{|D_{k,k} - \lambda|}$$

for some value k. Or $|D_{k,k} - \lambda| \leq \sum_j |F_{k,j}|$ and so λ is in G_k.

Note that if $\lambda = D_{i,i}$ then this result holds immediately. ∎

EXAMPLE 3 *For the Laplacian matrix in Poisson's equation on a 2d mesh, all the Gerschgorin disks intersect and give a bound on the largest eigenvalue of $\boldsymbol{A}$ of 8.*

2 THE POWER METHOD

Recall that one way to find the largest eigenvalue and associated eigenvector of a matrix is to use the power method

```
Given a starting vector x with norm(x) = 1 and tolerance τ
Iterate
  y = Ax
  λ = x^T y
  x = y/ norm(y)
Until norm(Ax − λx) ≤ τ (your tolerance)
```

If the largest magnitude eigenvalue associated with $\boldsymbol{A}$ is unique, then this iteration converges to it.

— TODO – Insert picture of the eigenvalue convergence.

2.1 A USEFUL UPPER BOUND ON THE SPECTRAL RADIUS OF A SYMMETRIC POSITIVE DEFINITE MATRIX.

In many cases, we do not actually need the largest eigenvalue of a matrix itself, but rather, a bound on the spectral radius. That is, we want to compute a value θ such that $\rho(\boldsymbol{A}) \leq \theta$. This is easy to do via Gerschgorin disks. However, we also usually want θ to be close to $\rho(\boldsymbol{A})$. Gerschgorin disks can be fairly far away.

EXAMPLE 4 *Consider the matrix* $\boldsymbol{A} = \begin{bmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & 0 \end{bmatrix}$ *where* $\mathbf{e}$ *has length n. Then the Gerschgorin bound on the largest eigenvalue if n. However, the largest eigenvalue is actually* $\sqrt{n}$.

For this reason, we might think of using the power method to approximate the spectral radius. For a symmetric matrix, however, we will always have $|\lambda| \leq |\rho(\boldsymbol{A})|$ because the spectral radius is the most extreme point. The following theorem gives an upper bound.

THEOREM 5 (Trevisan 9.6, Lecture Notes on Graph Partitioning, Expanders and Spectral Methods, 2006) *Let* $\boldsymbol{M}$ *be a symmetric positive semi-definite matrix, then running the power method for k steps from a vector* $\mathbf{x}^{(0)}$ *with* $x_i^{(0)} = \pm 1$ *at random, produces a vector* $\mathbf{x}^{(k)}$ *with Rayleigh quotient*

$$\mathbf{x}^{(k)^T} \boldsymbol{M} \mathbf{x}^{(k)} / \mathbf{x}^{(k)^T} \mathbf{x}^{(k)} \geq \rho(\boldsymbol{A})(1-\varepsilon) \frac{1}{1 + 4n(1-\varepsilon)^{2k}}.$$

with probability at least 3/16.

This can be then translated into a useful upper-bound on $\rho(\boldsymbol{A})$ and you can use the tightest value of ε to make the result you want.

2.2 THE SMALLEST EIGENVALUE OF A SYMMETRIC POSITIVE SEMI-DEFINITE MATRIX.

The above result gives us an immediate algorithm to find the smallest eigenvalue of a symmetric positive semi-definite matrix. We get an upperbound $\theta \geq \rho(\boldsymbol{A})$ and run the power method on $\theta \boldsymbol{I} - \boldsymbol{A}$. In this case, we have to adjust the eigenvalue estimate $\lambda = \mathbf{x}^T \mathbf{y}$ to $\lambda = \theta - \mathbf{x}^T \mathbf{y}$ to adjust for the difference.

2.3 THE INVERSE POWER METHOD.

There are a variety of ways to use the power method to get other eigenvalues besides the largest. The first is the inverse power method, where we run the power method on $\boldsymbol{A}^{-1}$ itself, which corresponds to the changing one line of the iteration:

$$\mathbf{y} = \boldsymbol{A}\mathbf{x} \quad \rightarrow \quad \boldsymbol{A}\mathbf{y} = \mathbf{x}$$

so that we solve a linear system at each step. With this change, the power method will converge to the *smallest magnitude* eigenvalue of $\boldsymbol{A}$ instead. (That is, the one closest to o.)

— TODO – Insert picture of the eigenvalue transformation $1/\lambda$

2.4 TARGETING A SPECIFIC EIGENVALUE.

If we wish to find an eigenvalue close to a value α, then we can use the iteration:

$$(\boldsymbol{A} - \alpha \boldsymbol{I})\mathbf{y} = \mathbf{x}.$$

The eigenvalues of $\boldsymbol{A}$ that are near α will be close to zero in the matrix $(\boldsymbol{A} - \alpha \boldsymbol{I})$, and so when we use inverse iteration on $(\boldsymbol{A} - \alpha \boldsymbol{I})$ then we will find those as the solutions.

— TODO – Insert picture of the eigenvalue transformation $1/(\lambda - \alpha)$

2.5 USING A FOLDED SPECTRUM TO TARGET AN EIGENVALUE.

If $\boldsymbol{A}$ is large and sparse, then we may not have a good way to solve linear systems with $\boldsymbol{A}$. This is often the case for problems that arise based on data where we do not have well-known and effective preconditioner techniques. In this case, if we are interested in finding the smallest eigenvalue, we can employ an idea called the *folded spectrum*. The idea is that $\boldsymbol{A}^2$ is always a symmetric positive semi-definite matrix whose small eigenvalues are close to 0. If we have an upper-bound on the spectral radius of $\theta^2 \leq \boldsymbol{A}^2$, then we can use the same idea for the smallest eigenvalue of a symmetric positive semi-definite matrix.

This can further be adapted to target an eigenvalue near α by using $(\boldsymbol{A} - \alpha \boldsymbol{I})^2$ to make eigenvalues nearby α close to zero. This approach is called the *folded spectrum* method.

2.6 FINDING OTHER EIGENVALUES VIA DEFLATION

We can actually use the power method itself to compute multiple eigenvectors via a procedure called deflation. Let $\mathbf{x}, \lambda$ be an eigenpair of $\boldsymbol{A}$. Then create a householder matrix $\boldsymbol{H}$ with $\mathbf{x}$ as associated vector such that $\boldsymbol{H}\mathbf{x} = \mathbf{e}_1$. The matrix $\boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^T$ has the following structure:

$$\begin{bmatrix} \lambda & 0 \\ 0 & \boldsymbol{B} \end{bmatrix}$$

This is actually what we did way back when we created the SVD!

EXAMPLE 6 *Here is some example code to demonstrate this.*

```
A = randn(5,5)
A = A+A'
## Example of the deflation method
λs,X = eig(A)
eval = rand(1:size(A,2))
x = X[:,eval] # just an
xe1 = zeros(size(A,1))
xe1[1] = norm(x)
v = xe1-x
H = eye(size(A)...) - 2*v*v'/(v'*v)
display(H*A*H')
display(λs[eval])
```

3 SUBSPACE ITERATION AND THE QR ALGORITHM

We can generalize the power method to compute the largest few eigenvalues and vectors.

```
Given a starting orthogonal matrix X and tolerance τ
Iterate
  Y = AX
  Λ = diagonal elements of X^T AX arranged as a diagonal matrix
  [X, R] = qr(Y)
Until norm(AX − XΛ) ≤ τ (your tolerance)
```

Again, if the matrix is symmetric, and these large eigenvalues are unique, we can show that this converges. This method is also called the block power method or the orthogonal iteration.

3.1 SUBSPACE ITERATION

Except that when this is usually done, we want to do it for all eigenvalues and vectors, and start with the identity matrix.

```
Given tolerance τ
Set X = I
Iterate
  Y = AX
  Λ = diagonal elements of X^T AX arranged as a diagonal matrix
  [X, R] = qr(Y)
Until norm(AX − XΛ) ≤ τ (your tolerance)
```

EXAMPLE 7 *Here's one sample of the subspace method converging*

```
A = randn(5,5)
A = A+A'

X = eye(size(A,1),size(A,2))
D = diagm(sort(diag(X'*A*X)))
for i = 1:200
  Y = A*X
  D = diagm(sort(diag(X'*Y)))
  X,R = qr(Y)
end
@show [diag(D) eigvals(A)]
```

3.2 THE QR ITERATION

Here's another way you'll read about to compute eigenvalues and eigenvectors and it's called the QR iteration.

```
Given tolerance τ
X = I
Iterate
  Q, R = qr(A)
  A = RQ.
  Λ = diagonal elements of A arranged as a diagonal matrix
  X = XQ
Until norm(AX − XΛ) ≤ τ (your tolerance)
```

When I saw this algorihtm, I found it truly strange! Why would you take the product of a QR factorization in the reverse order? The following important note gives some intuition for what is going on:

Important note. $\boldsymbol{A} = \boldsymbol{QR}$ implies that $\boldsymbol{R} = \boldsymbol{Q}^T\boldsymbol{A}$. So $\boldsymbol{A}_{\text{next}} = \boldsymbol{Q}^T\boldsymbol{AQ}$. Essentially, we are multiplying by an orthogonal matrix on the left and the right.

EXAMPLE 8 *Here's one sample of the QR iteration converging*

```
X = eye(size(A,1),size(A,2))
A = copy(Ainit)
for i = 1:100
  Q,R = qr(A)
  A = R*Q
  D = diagm(sort(diag(A)))
  X = X*Q
end
@show [diag(D) eigvals(Ainit)]
```

3.3 QR AND SUBSPACE ITERATION ARE EQUIVALENT

Here, we'll consider the following two methods and show that they are equivalent.

Subspace iteration

$\boldsymbol{X}_0 = \boldsymbol{I}$
for i = 1, ...
$\boldsymbol{X}_i, \boldsymbol{R}_i = \text{qr}(\boldsymbol{AX}_{i-1})$

QR iteration

$\boldsymbol{A}_1 = \boldsymbol{A}$
for i = 1, ...
$\boldsymbol{Q}_i, \boldsymbol{R}_i = \text{qr}(\boldsymbol{A}_i)$
$\boldsymbol{A}_{i+1} = \boldsymbol{R}_i\boldsymbol{Q}_i$

LEMMA 9 $\boldsymbol{X}_i = \boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_i$

Proof We'll prove this by induction. The key insight is that $\boldsymbol{R}_i$ is actually the same matrix. (Technical note, we need to assume that the signs of the diagonal elements of $\boldsymbol{R}_i$ are taken to be positive to get a unique $\boldsymbol{Q}$ factor in the QR decomposition.)

Base case. $\boldsymbol{X}_1 = \text{qr}_Q(\boldsymbol{AX}_{i-1}) = \text{qr}(\boldsymbol{A})$ and also $\boldsymbol{Q}_1 = \text{qr}_Q(\boldsymbol{A})$ as well. Because $\boldsymbol{R}_i = \boldsymbol{Q}_1^T\boldsymbol{A} = \boldsymbol{X}_1^T\boldsymbol{A}$, we get the same $\boldsymbol{R}_i$ factor here.

Inductive hypothesis. We assume that $\boldsymbol{X}_i = \boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_i$.

Important note. We saw this before, but $\boldsymbol{A}_i = \boldsymbol{Q}_i\boldsymbol{R}_i$ implies that $\boldsymbol{R}_i = \boldsymbol{Q}_i^T\boldsymbol{A}_i$. So $\boldsymbol{A}_{i+1} = \boldsymbol{Q}_i^T\boldsymbol{A}_i\boldsymbol{Q}_i$. If we iterate this, then note that

$$\boldsymbol{A}_{i+1} = \boldsymbol{Q}_i^T\boldsymbol{A}_i\boldsymbol{Q}_i = \boldsymbol{Q}_i^T\cdots\boldsymbol{Q}_1^T\boldsymbol{A}\boldsymbol{Q}_1\cdots\boldsymbol{Q}_i. \qquad \blacksquare$$

By our induction hypotheis $\boldsymbol{A}_{i+1} = \boldsymbol{X}_i^T\boldsymbol{A}\boldsymbol{X}_i$.

Note that $\boldsymbol{X}_{i+1}\boldsymbol{R}_{i+1} = \mathrm{qr}(\boldsymbol{A}\boldsymbol{X}_i) = \mathrm{qr}(\boldsymbol{A}\boldsymbol{Q}_1\cdots\boldsymbol{Q}_i)$.

So $\boldsymbol{X}_{i+1}^T\boldsymbol{A}\boldsymbol{Q}_1\cdots\boldsymbol{Q}_i = \boldsymbol{R}_{i+1}$.

3.4 THE ISSUE WITH QR

The problem with the QR method is that it is very expensive. Each iteration is $O(n^3)$.

In order to make this more efficient, we want to translate the matrix $\boldsymbol{A}$ into one that has a structure that makes the RQ-iteration efficient. The properties we need are:

1. we can translate a general symmetric matrix $\boldsymbol{A}$ to a matrix $\boldsymbol{B}$ with this property via orthogonal transformations $\boldsymbol{B} = \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q}$ or similarity transformations $\boldsymbol{B} = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}$ (so we preserve eigenvalues)
2. computing a QR factorization of $\boldsymbol{B}$ is efficient
3. computing the product $\boldsymbol{R}\boldsymbol{Q}$ is efficient
4. the matrix $\boldsymbol{R}\boldsymbol{Q}$ has the same property as $\boldsymbol{B}$.

There are some modestly esoteric classes of matrices that enable these (such as hierarchical semi-seperable), but the most straightforward case is a tridiagonal matrix.

Here, we show that if $\boldsymbol{B}$ is tridiagonal, then $\boldsymbol{R}\boldsymbol{Q}$ is also tridiagonal. This gives us most of the properties since we always saw how tridiagonal matrices enable a linear-time QR factorization. (In the section on GMRES.)

Consider a symmetric tridiagonal matrix

$$\boldsymbol{B} = \begin{bmatrix} \bullet & \bullet & & & \\ \bullet & \bullet & \bullet & & \\ & \bullet & \bullet & \bullet & \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \end{bmatrix}$$

Then in the first step of QR, we are going to "zero" out the $2,1$ entry via a Givens transform. This gives us:

$$\boldsymbol{Q}_1 = \begin{bmatrix} \bullet & \bullet & & & \\ \bullet & \bullet & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \qquad \boldsymbol{Q}_1^T\boldsymbol{B} = \begin{bmatrix} \times & \times & \times & & \\ 0 & \times & \times & & \\ & \bullet & \bullet & \bullet & \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \end{bmatrix}$$

$$\boldsymbol{Q}_2 = \begin{bmatrix} 1 & & & & \\ & \bullet & \bullet & & \\ & \bullet & \bullet & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \qquad \boldsymbol{Q}_2^T\boldsymbol{Q}_1^T\boldsymbol{B} = \begin{bmatrix} \bullet & \bullet & \bullet & & \\ 0 & \times & \times & \times & \\ & 0 & \times & \times & \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \end{bmatrix}$$

Hence, at the end, we'll have

$$\boldsymbol{Q}_4^T\boldsymbol{Q}_3^T\boldsymbol{Q}_2^T\boldsymbol{Q}_1^T\boldsymbol{B} = \boldsymbol{R} = \begin{bmatrix} \bullet & \bullet & \bullet & & \\ 0 & \bullet & \bullet & \bullet & \\ & 0 & \bullet & \bullet & \bullet \\ & & 0 & \bullet & \bullet \\ & & & 0 & \bullet \end{bmatrix}$$

$$Q = Q_1 Q_2 Q_3 Q_4 = \begin{bmatrix} \bullet & \bullet & & & \\ \bullet & \bullet & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & \bullet & \bullet & & \\ & \bullet & \bullet & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \cdots$$

So

$$RQ = \underbrace{\begin{bmatrix} \bullet & \bullet & \bullet & & \\ 0 & \bullet & \bullet & \bullet & \\ & 0 & \bullet & \bullet & \bullet \\ & & 0 & \bullet & \bullet \\ & & & 0 & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & & & \\ \bullet & \bullet & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}}_{\text{linear combination of columns 1,2}} Q_2 Q_3 Q_4 = \begin{bmatrix} \times & \times & \bullet & & \\ \times & \times & \bullet & \bullet & \\ & 0 & \bullet & \bullet & \bullet \\ & & 0 & \bullet & \bullet \\ & & & 0 & \bullet \end{bmatrix} Q_2 Q_3 Q_4$$

$$= \begin{bmatrix} \bullet & \times & \times & & \\ \bullet & \times & \times & \bullet & \\ & \times & \times & \bullet & \bullet \\ & & 0 & \bullet & \bullet \\ & & & 0 & \bullet \end{bmatrix} Q_3 Q_4$$

$$= \begin{bmatrix} \bullet & \bullet & \times & \times & \\ \bullet & \bullet & \times & \times & \\ & \bullet & \times & \times & \bullet \\ & & \times & \times & \bullet \\ & & & 0 & \bullet \end{bmatrix} Q_4$$

$$= \begin{bmatrix} \bullet & \bullet & \bullet & \times & \times \\ \bullet & \bullet & \bullet & \times & \times \\ & \bullet & \bullet & \times & \times \\ & & \bullet & \times & \times \\ & & & \times & \times \end{bmatrix}$$

We can keep going with this and we'll find that RQ is upper-Hessenberg.
But, $R = Q^T A$ so $RQ = Q^T QQ$ is symmetric, upper-Hessenberg, i.e. tridiagonal!

4 REDUCTION TO TRIDIAGONAL FORM

EXAMPLE 10 *A = randn(6)*

5 QR WITH SHIFTS

6 EIGENVALUES AND EIGENVECTORS OF SPARSE MATRICES

Run the power method! Then we'll get

6.1 DEFLATION
6.2 SUBSPACE ITERATION
6.3 LANCZOS
6.4 ARPACK

EXAMPLE 11 *This example shows how to use the eigs function in Matlab*

```
A = sprandsym(10000,50/10000);
[V,D] = eigs(A);
```