Numerical Methods for Computing Eigenvectors

Eric Zheng

21-241 Final Project

December 6, 2019

Abstract

In this document, I present some background on numerical methods for computing the eigenvectors and singular vectors of matrices. A Julia implementation of the power and QR methods is given, and the two algorithms are compared

Contents

1 Background: Eigenvectors and Eigenvalues

I assume that the reader is familiar with linear algebra at the college introductory level. In this section, I review a little bit of background to motivate the algorithms that will be developed in subsequent sections. We begin by recalling definition 1.1.

Definition 1.1: eigenvectors and eigenvalues

Consider an arbitrary $n \times n$ matrix A. For some $\mathbf{x} \in \mathbb{R}^n$ (with $\mathbf{x} \neq \mathbf{0}$), we say that $\mathbf{x}$ is an *eigenvector* of A iff, for some $\lambda \in \mathbb{R}$, $A\mathbf{x} = \lambda\mathbf{x}$. We denote λ as the corresponding *eigenvalue* of A.

From definition 1.1 follows immediately a somewhat naive way to compute the eigenvalues of a given matrix. Note that

$$A\mathbf{x} = \lambda\mathbf{x} \implies A\mathbf{x} - \lambda\mathbf{x} = \mathbf{0}$$

and $\lambda\mathbf{x} = \lambda I\mathbf{x}$, so we have

$$A\mathbf{x} - \lambda I\mathbf{x} = (A - \lambda I)\mathbf{x} = \mathbf{0}.$$

That is to say, $\lambda \in \mathbb{R}^n$ is an eigenvalue of A if and only if $A - \lambda I$ has a non-trivial null space. A matrix has a non-trivial null space if and only if it is singular, so we require that $\det(A - \lambda I) = 0$. This result is stated in theorem 1.1.

Theorem 1.1: computing eigenvalues as polynomial roots

Some $\lambda \in \mathbb{R}^n$ is an eigenvalue of the $n \times n$ matrix A if and only if $\det(A - \lambda I) = 0$. We call $\det(A - \lambda I)$ the *characteristic polynomial* for A. The problem then becomes identifying the roots of this polynomial.

Once the eigenvalues have been computed, we can find the corresponding eigenvectors by finding the null space of $A - \lambda I$, for example by using the reduced row echelon form. The key drawback of this method is that polynomial root-finding is sensitive to small numerical errors.

Example 1.1: sensitivity of polynomial root finding

Consider the matrix

However, the equivalence of eigenvalue-finding and polynomial root-finding gives some insight into how we could approach the problem with more advanced methods. It is known that polynomials of degree five and higher do not in general have a solution by radicals [cite], although we can use iterative methods to get arbitrarily good approximations of these roots. In the following sections, we will apply similar iterative methodologies to find the eigenvectors and eigenvalues of matrices.

2 The Power Method

2.1 Mathematical Formulation

Definition 2.1: dominant eigenvector

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of the matrix A, and let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be corresponding eigenvalues. We call $\mathbf{x}_i$ a *dominant eigenvector* if $|\lambda_i| > |\lambda_j|$ whenever $i \neq j$. The corresponding λ_i is called the *dominant eigenvalue*.

The power method for computing eigenvectors takes successive powers of the matrix A^k until some stopping criterion is reached. As k grows large, the columns of A will approach the dominant eigenvector of A. This is stated in theorem 2.1.

Theorem 2.1: the power method

If A is an $n \times n$ diagonalizable matrix with a dominant eigenvector $\mathbf{x}$, then the columns of A^k approach a multiple of $\mathbf{x}$ as k grows arbitrarily large.

Proof. Since A is diagonalizable, let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a basis of eigenvectors for $\mathbb{R}^n$, where we order the eigenvectors so that $\mathbf{x}_1$ is a dominant eigenvector. Now since the $\mathbf{x}_i$'s form a basis, any $\mathbf{v} \in \mathbb{R}^n$ can be expressed as the linear combination

$$\mathbf{v} = c_1\mathbf{x}_1 + \cdots + c_n\mathbf{x}_n.$$

Then by linearity, we have

$$\begin{aligned} A^k\mathbf{v} &= A^k(c_1\mathbf{x}_1 + \cdots + c_n\mathbf{x}_n) \\ &= A^k c_1\mathbf{x}_1 + \cdots + A^k c_n\mathbf{x}_n \\ &= \lambda_1^k c_1\mathbf{x}_1 + \cdots + \lambda_n^k c_n\mathbf{x}_n. \end{aligned}$$

But since $|\lambda_1| > |\lambda_i|$ for all $i \geq 2$, we see that the first term $\lambda_1^k c_1\mathbf{x}_1$ dominates as k grows very large (as long as $c_1 \neq 0$). Hence for large k, $A^k\mathbf{v} \approx \lambda_1^k c_1\mathbf{x}_1$, if $c_1 \neq 0$. Taking $\mathbf{v}$ to be the standard basis vectors then produces the desired result, since at least one of the $\mathbf{e}_i$'s must have a component along $\mathbf{x}_1$. □

One issue with the power method is that it assumes that A is both diagonalizable and has a dominant eigenvector. These flaws are highlighted in examples 2.1 and 2.2. Another example of the power method's failure to converge is a rotation matrix [1], which does not typically have real eigenvalues.

Example 2.1: power method on a non-diagonalizable matrix

The power method can fail when a matrix is not diagonalizable. Consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

A has eigenvalue $\lambda = 0$ with eigenvector $(1, 0)$. But A is nilpotent, so the columns of A^k will never approach a multiple of $(1, 0)$.
ACTUALLY...rotation matrix might be a better example, since this can be thought of more as a failure around zero (i.e. $(0, 0)$ is a legitimate multiple of $(1, 0)$)

Example 2.2: power method without a dominant eigenvalue

The power method can fail when a matrix does not have a dominant eigenvalue, even if it is diagonalizable. Consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Now A is clearly diagonalizable (it's already diagonal), but its eigenvalues are $\lambda = \pm 1$, so A does not have a dominant eigenvalue. If we attempt the power method, we find that $A^2 = I$, so the successive powers A^k will oscillate between A (when k is odd) and I (when k is even). Neither of these contain the eigenvectors of A, which are $(1, 1)$ and $(1, -1)$.

Note that diagonalizability is a sufficient condition for the power method to converge (when combined with a dominant eigenvalue), but it is not necessary. Example 2.3 gives a matrix that is not diagonalizable yet converges under the power method.

Example 2.3: power method despite non-diagonalizability

Consider the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

This matrix has only one eigenvalue $\lambda = 1$ with eigenvectors $(0, 1)$ and $(0, -1)$, which are not independent. But an easy induction reveals that

$$A^k = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$$

whose columns indeed converge to the eigenvector $(0, 1)$ as k grows large.

2.2 Convergence Analysis

For all its faults, the power method is a very elegant way to compute the dominant eigenvector of a matrix. However, we are typically concerned not just with how elegant a method is, but also how efficient it is.

2.3 Rayleigh Quotients

Up until now, we have developed a method for finding the eigenvectors of a matrix, but what about the eigenvalues? Based on definition 1.1, it is tempting to just take an eigenvector $\mathbf{x} \in \mathbb{R}^n$, compute $A\mathbf{x} = \lambda\mathbf{x}$, and then see how much the components of $\mathbf{x}$ were scaled to find λ.

The issue is that our algorithm only generates an approximation to $\mathbf{x}$, not $\mathbf{x}$ itself [1]. Hence each of the components will not be exactly scaled by λ: some might be a little larger than they should be, and some might be a little smaller. The next thing that comes to mind is to let μ_i be the amount by which each component of $\mathbf{x}$ was scaled and then just average the μ_i's. Unfortunately, this approach is sensitive to small errors, particularly around 0 [2], as demonstrated in example 2.4.

Example 2.4: sensitivity of averaging for finding λ

Consider the matrix

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

which has $\mathbf{x}_1 = (1, 0)$ and $\mathbf{x}_2 = (0, 1)$ with corresponding $\lambda_1 = 2$ and $\lambda_2 = 1$. Now suppose we have an eigenvector approximation $\hat{\mathbf{x}}_1 = (1, 0.00001)$. This is very close to the true dominant eigenvector:

$$\|\hat{\mathbf{x}}_1 - \mathbf{x}_1\| = 0.00001,$$

yet we have:

$$A\hat{\mathbf{x}}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0.00001 \end{bmatrix} = \begin{bmatrix} 2 \\ 0.00001 \end{bmatrix}.$$

If we try to approximate λ_1 by diving component-wise and then averaging, we get:

$$\mu_1 = 2/1 = 2$$
$$\mu_2 = 0.00001/0.00001 = 1$$

which would give $\lambda \approx 3/2$, which is significantly off from the true value $\lambda = 2$, despite the very good eigenvector approximation.

[1] In fact, even if we gave the algorithm infinite time to run, it is mathematically impossible for a computer with a finite alphabet to represent arbitrary reals, since $\mathbb{R}$ is uncountable.

So how should we compute eigenvalues given a corresponding eigenvector? A common way to do this is to use the Rayleigh quotient, given in definition 2.2

Definition 2.2: Rayleigh quotient CITE

If we have an approximation $\mathbf{x} \in \mathbb{R}^n$ for an eigenvector of A, then the *Rayleigh quotient* approximation for the corresponding eigenvalue λ is given by

$$\lambda \approx \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}.$$

This can be thought of as the average of the μ_i's weighted by the square of each component [2], so errors around small components affect the overall eigenvalue approximation very little. For example, using the same numbers as example 2.4, we get

$$\lambda \approx \frac{\hat{\mathbf{x}}_1^T A \hat{\mathbf{x}}_1}{\hat{\mathbf{x}}^T \hat{\mathbf{x}}} = \frac{2.0000000001}{1.0000000001} \approx 1.999999999$$

which is much closer to the true value of $\lambda = 2$.

2.4 Deflation

One limitation is that the power method, as presented in theorem 2.1, only finds the dominant eigenvector of a matrix. For many physical systems, the behavior is determined mostly by the dominant eigenvector, so this is sufficient. However, in the special case where A is symmetric, we can actually do better. By the spectral theorem (restated in theorem 4.1), we can take the spectral decomposition of symmetric A into orthogonal X and diagonal Λ, as in

$$A = X \Lambda X^T = \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T + \cdots + \lambda_n \mathbf{x}_n \mathbf{x}_n^T$$

where we sort the eigenvalues in order of magnitude. Now applying the power method will yield $\mathbf{x}_1$ (which in turn can be used to find λ_1). But observe that

$$B_1 = A - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T = \lambda_2 \mathbf{x}_2 \mathbf{x}_2^T + \cdots + \lambda_n \mathbf{x}_n \mathbf{x}_n^T$$

is a symmetric matrix that has eigenvalues $\lambda_2, \ldots, \lambda_n$ and corresponding $\mathbf{x}_2, \ldots, \mathbf{x}_n$! Hence we can apply the power method to B_1 to find $\mathbf{x}_2$ and λ_2. We can keep on going with

$$B_2 = B_1 - \lambda_2 \mathbf{x}_2 \mathbf{x}_2^T = \lambda_3 \mathbf{x}_3 \mathbf{x}_3^T + \cdots + \lambda_n \mathbf{x}_n \mathbf{x}_n^T$$

to which we can apply the power method to pick off $\mathbf{x}_3$ and λ_3. We keep on going until we have found all n eigenvectors, a technique known as *deflation*.

In fact, even if A is not symmetric, we can apply a similar deflation technique for finding all the eigenvalues of A, although we don't get any additional eigenvectors beyond the dominant one. The reader is referred to [3] for a full

explanation, but in general the procedure is given in theorem 2.2. In the case of a symmetric matrix, this simplifies to the previously described deflation procedure.

Theorem 2.2: Hotelling deflation

Suppose A is a matrix with eigenvalues $\lambda_1, \ldots, \lambda_k$ and corresponding eigenvalues $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Then the matrix

$$B = A - \frac{\lambda_1}{\mathbf{x}_1^T \mathbf{x}} \mathbf{x}_1 \mathbf{x}_1^T$$

has eigenvalues $\lambda_2, \ldots, \lambda_k$, although it does not necessarily have the same eigenvectors $\mathbf{x}_2, \ldots, \mathbf{x}_k$.

2.5 Optimization for Sparse Matrices

As we have developed it thus far, the power method involves computing large powers A^k. While we can make this somewhat efficient via tricks like exponentiation by squaring, matrix multiplication is an expensive thing to do; the best known algorithms run in roughly $O(n^{2.373})$ [4] for an $n \times n$ matrix. Can we do better?

A faster approach is to reframe the problem as a bunch of matrix-vector multiplications, which can be computed naively in $O(n^2)$. In fact, if A is a sparse matrix (i.e. most entires are zero), matrix-vector multiplication can become *extremely* efficient [1]. We can tweak theorem 2.1 slightly into theorem 2.3, whose proof is much the same as theorem 2.1.

Theorem 2.3: more efficient power method

Suppose A is an $n \times n$ diagonalizable matrix with a dominant eigenvalues λ_1 and corresponding eigenvector $\mathbf{x}_1$. Then there exists some $\mathbf{v}_0 \in \mathbb{R}^n$ such that

$$\mathbf{v}_k = A^k \mathbf{v}_0 = A\mathbf{v}_{k-1}$$

approaches a multiple of $\mathbf{x}_1$ as k grows arbitrarily large.

Now if A is diagonalizable, then any $\mathbf{v}_0 \in \mathbb{R}^n$ can be expressed in the basis of eigenvectors as

$$\mathbf{v}_0 = c_1 \mathbf{x}_1 + \cdots + c_n \mathbf{x}_n.$$

As long as $\mathbf{v}_0$ has some component $c_1 \neq 0$ along $\mathbf{x}_1$, then $A^k \mathbf{v}_0$ will converge to $\mathbf{x}_1$ as k grows large. This is great, since $A^k \mathbf{v}_0$ can be recursively computed as $A(A^{k-1}\mathbf{v}_0)$, which is k matrix-vector multiplications instead of k matrix-matrix multiplications.

2.6 Implementation

The diagonalizability problem can be avoided in most cases because the matrices we are typically interested in are diagonalizable.

[then the faster power method should be presented]

[oh, and also finding eigenvalues from eigenvectors]

3 The QR Method

3.1 Mathematical Formulation

Another iterative method to compute the eigenvectors of a matrix is known as the QR method. The key insight behind this method is that similar matrices have the same eigenvalues.

[now in order to compute the QR decomposition of A, we require that A have independent columns; for a square matrix, this means that A must be invertible?]

Example 3.1: QR method on matrix without full rank

The QR method can fail when the given matrix does not have full rank. [check this] Consider the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

whose eigenvalues are $\lambda_1 = 1$, $\lambda_2 = 0$. But A's QR decomposition is

$$Q = -\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad R = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}$$

so we have

$$A' = RQ = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

but the QR decomposition of A' is

$$Q' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R' = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

so we have

$$A'' = R'Q' =$$

WAIT...I may have misunderstood the algorithm; I think it converges to a triangular matrix, which has eigenvalues on the diagonal???

The advantage of the QR method is that it does not suffer from the rounding errors that deflation-based methods do. However, it is also not as efficient, especially in many practical scenarios where we have large systems and only

care about the most dominant eigenvectors. [cite last slide in that PPT, unless I get a better source]

3.2 Shifting for Better Efficiency

Fun stuff

4 Connection to Singular Vectors

Up until now, we have been almost exclusively concerned with finding the eigenvectors and eigenvalues of a matrix. But what about finding the singular vectors of a matrix?

Note that the singular vectors of any $m \times n$ matrix A are just the eigenvectors of $A^T A$, so by finding the eigenvectors we already have a procedure for finding the singular vectors. To make the connection even better, note that $A^T A$ is symmetric, so by theorem 4.1 it is in fact diagonalizable. We can thus use the power method to find the eigenvectors and eigenvalues of $A^T A$, which give us the singular vectors and values of A.

Theorem 4.1: spectral theorem for real symmetric matrices

If A is a real symmetric $n \times n$ matrix, then A has n orthonormal eigenvectors. The reader is referred to [cite] for a full proof of this theorem.

5 Algorithm Comparison

Maybe make a table comparing power and QR, and add in some pretty graphs

6 Stuff I forgot about

6.1 Singular Vectors and Values

We have seen that eigenvectors and eigenvalues are useful for understanding the behavior of square matrices, but what about the more general case of $m \times n$ matrices? To handle this, we define the notion of singular vectors and values in definition 6.1.

Definition 6.1: singular vectors and values

Given an $m \times n$ matrix A, we denote the eigenvectors of $A^T A$ as the *singular vectors* of A. The corresponding eigenvalues of $A^T A$ are called the *singular values* of A.

Theorem 6.1: existence of singular vectors and values

Any $m \times n$ matrix A admits a decomposition $A = U\Sigma V^T$, where

- U is an $m \times m$ orthogonal matrix
- Σ is an $m \times n$ matrix that is "diagonal" in that $\Sigma_{ij} = 0$ if $i \neq j$
- V is an $n \times n$ orthogonal matrix.

7 Numerical Methods

In general, we have seen that it is possible to compute eigenvectors of an $n \times n$ matrix as the roots of an n-degree polynomial. Unfortunately, it is a well-known result that polynomials of degree 5 and higher to not in general admit a solution by radicals [cite]...

The most obvious way to compute the eigenvectors of A is to first find the eigenvalues by LU decomposition and then use elimination to find the null space of $A - \lambda I$.

7.1 The Power Algorithm

In this section, we present the *power method*, a simple method for computing the eigenvectors and eigenvalues of A. As the name suggests, this method involves taking matrix powers A^k for increasingly large values of k. To use the power method, we first choose a random starting vector $\mathbf{x}_0 \in \mathbb{R}^n$.

Theorem 7.1: convergence of the power method

Consider an $n \times n$ matrix A. If $|\lambda_1| > |\lambda_i|$ whenever $i \neq 1$, then there exists some vector $\mathbf{x}_0 \in \mathbb{R}^n$ such that $A^k\mathbf{x}_0$ converges to a scalar multiple of $\mathbf{x}_1$ as k grows arbitrarily large.

Proof. Suppose A has $e > 1$ eigenvalues. Now let $\mathbf{x}_0 =$ □

This suggests the method behind algorithm 7.1.

There remain a few important questions to be addressed:

- How quickly does the power method converge?
- What about the case when $|\lambda_1| = |\lambda_2|$?
- What about zero eigenvalues?
- How do we choose a suitable initial $\mathbf{x}_0$?

- What if A does not have a full basis of eigenvectors? (i.e. our $\mathbf{x}_0$ might have a component orthogonal to the span of the eigenvectors—then convergence may not be guaranteed?)
- What about degenerate eigenvalues?

Present algorithm Explain why it works Possibly bounds on accuracy?

Conditions on convergence: if there are multiple eigenvalues with the same magnitude, this will not converge. (Since any symmetric matrix has real eigenvalues, this only occurs when λ and $-\lambda$ are both eigenvalues of A.) [Is there some other condition on null spaces and stuff?] Also, the rate at which this converges depends greatly on the ratio between the two eigenvalues!

Also, does this handle degenerate eigenvalues?

We first present algorithm 7.2 for symmetric matrices only because it is highly illustrative. Additionally, [all we need for singular vectors]

We square and then normalize by the first nonzero column of A^n. If such a column does not exist (i.e. A^n is the zero matrix), then A must have been a nilpotent matrix. In this case, the only eigenvalue can be zero, as proved in theorem 7.2. In this case, the problem reduces to finding the null space of A.

Now a matrix can have at most n independent eigenvectors, so we only have to iterate up to n.

Theorem 7.2: eigenvalues of a nilpotent matrix

If A is a nilpotent matrix (i.e. A^n is the zero matrix for some $n \in \mathbb{N}^+$), then the only eigenvalue of A is $\lambda = 0$.

Proof. Suppose $\mathbf{x} \neq \mathbf{0}$ were an eigenvector of A with eigenvalue $\lambda \neq 0$. Then $A^n\mathbf{x} = \lambda^n\mathbf{x}$, but if $\lambda \neq 0$, then this is some nonzero vector. However, $A^n = 0$, so we also have $A^n\mathbf{x} = \mathbf{0}$, a contradiction. □

Algorithm 7.1: `DominantEigen`

input : real diagonalizable $n \times n$ matrix A
parameter: tolerance $\varepsilon > 0$, max iterations $N > 0$
output : dominant eigenpair $(\mathbf{x}, \lambda)$

$A' \leftarrow A$;
$i \leftarrow 0$; `// number of iterations so far`
$\mathbf{x} \leftarrow$ `FirstNonzeroCol(`A'`)`;
$A' \leftarrow$ `SquareAndNormalize(`A'`)`;
while $\|\mathbf{x} -$ `FirstNonzeroCol(`A'`)`$\| > \varepsilon$ and $i < N$ **do**
  $\mathbf{x} \leftarrow$ `FirstNonzeroCol(`A'`)`;
  $A' \leftarrow$ `SquareAndNormalize(`A'`)`;
  $i \leftarrow i + 1$;
$\mathbf{x} \leftarrow$ `FirstNonzeroCol(`A'`)`;
$\lambda \leftarrow (A\mathbf{x} \cdot \mathbf{x})/(\mathbf{x} \cdot \mathbf{x})$;

Algorithm 7.2: `EigenPowerSymmetric`

input : real symmetric $n \times n$ matrix S
parameter: tolerance $\varepsilon > 0$, max iterations $N > 0$
output : list xs of eigenvectors and λs of eigenvalues

$xs \leftarrow$ empty list;
$\lambda s \leftarrow$ empty list;
$Z \leftarrow n \times n$ zero matrix;
for $i \in [n]$ **do**
  $\mathbf{x}$, $\lambda \leftarrow$ `DominantEigen` $(S - Z)$;
  $Z \leftarrow Z + \lambda \mathbf{x}\mathbf{x}^T$;
  append $\mathbf{x}$ to xs;
  append λ to λs;

Algorithm 7.2 also gives us an easy way to compute the singular values and singular vectors of any real matrix A: simply apply `EigenPowerSymmetric`$(A^T A)$.

Finding the eigenvectors of a general square matrix (not necessarily symmetric) is somewhat trickier. We will

[probably put a disclaimer at the top that we will only concern ourselves with real matrices, although we will sometimes emphasize this point in our algorithms. should we permit complex eigenvalues? nah, that moves us from Rn to Cn]

Example 7.1: failure of the power method

The power method can fail when the original matrix does not have a dominant eigenvalue. For example, consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

which has eigenvalues $\lambda = \pm 1$.

Example 7.2: another failure of the power method

The power method can also fail when the original matrix has complex eigenvalues. For example, consider the matrix

$$A = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which is intuitively a rotation around the z-axis. In this case, there is only one real eigenvalue, $\lambda = 1$. If we choose a good initial $\mathbf{x}_0 = (0, 0, 1)$, the algorithm does converge to the correct solution.

7.2 The QR Algorithm

Algorithm 7.3: `EigenQR`

input : real $m \times n$ matrix A with independent columns
output : eigenvectors $\mathbf{x}_i$ and corresponding eigenvalues λ_i

Algorithm 7.4: `Singular`

input : real $m \times n$ matrix A
parameter: tolerance $\varepsilon > 0$, max iterations $N > 0$
output : list of singular vectors vs and singular values σs

```
/* here we use the power method, but we could also use
   the QR method                                        */
```
xs, $\lambda s \leftarrow$ `EigenPowerSymmetric(`$A^T A$`)`;
$\sigma s \leftarrow$ filter positive λs;
$l \leftarrow$ length of σs;
$vs \leftarrow$ first l of xs;
return vs, σs

8 Results

Compare the two methods

References

[1] https://www.cs.huji.ac.il/~csip/tirgul2.pdf

[2] https://web.mit.edu/18.06/www/Spring17/Power-Method.pdf

[3] http://www.macs.citadel.edu/chenm/344.dir/08.dir/lect4_2.pdf

[4] https://www.cs.toronto.edu/~yuvalf/Limitations.pdf