CHAPTER 6

Eigenvectors

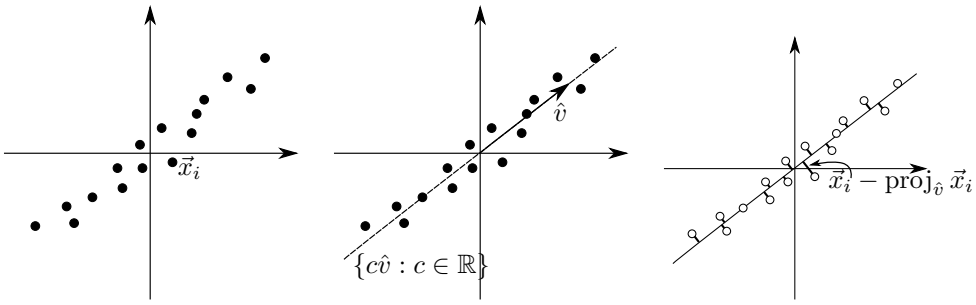
CONTENTS

WE turn our attention now to a *nonlinear* problem about matrices: Finding their eigenvalues and eigenvectors. Eigenvectors $\vec{x}$ and corresponding eigenvalues λ of a square matrix A are determined by the equation $A\vec{x} = \lambda\vec{x}$. There are many ways to see that the eigenvalue problem is nonlinear. For instance, there is a *product* of unknowns λ and $\vec{x}$. Furthermore, to avoid the trivial solution $\vec{x} = \vec{0}$, we constrain $\|\vec{x}\|_2 = 1$; this constraint keeps $\vec{x}$ on the unit sphere, which is not a vector space. Thanks to this structure, algorithms for finding eigenspaces will be considerably different from techniques for solving and analyzing linear systems of equations.

6.1 MOTIVATION

Despite the arbitrary-looking form of the equation $A\vec{x} = \lambda\vec{x}$, the problem of finding eigenvectors and eigenvalues arises naturally in many circumstances. To illustrate this point, before presenting algorithms for finding eigenvectors and eigenvalues we motivate our discussion with a few examples.

It is worth reminding ourselves of one source of eigenvalue problems already considered in Chapter 1. As explained in Example 1.27, the following fact will guide many of our examples:



(a) Input data (b) Principal axis (c) Projection error

Figure 6.1 (a) A dataset with correlation between the horizontal and vertical axes; (b) we seek the unit vector $\hat{v}$ such that all data points are well-approximated by some point along $\text{span}\{\hat{v}\}$; (c) to find $\hat{v}$, we can minimize the sum of squared residual norms $\sum_i \|\vec{x}_i - \text{proj}_{\hat{v}}\, \vec{x}_i\|_2^2$ with the constraint that $\|\hat{v}\|_2 = 1$.

When A is symmetric, the eigenvectors of A are the critical points of $\vec{x}^\top A\vec{x}$ under the constraint $\|\vec{x}\|_2 = 1$.

Many eigenvalue problems are constructed using this fact as a starting point.

6.1.1 Statistics

Suppose we have machinery for collecting statistical observations about a collection of items. For instance, in a medical study we may collect the age, weight, blood pressure, and heart rate of every patient in a hospital. Each patient i can be represented by a point $\vec{x}_i \in \mathbb{R}^4$ storing these four values.

These statistics may exhibit strong correlations between the different dimensions, as in Figure 6.1(a). For instance, patients with higher blood pressures may be likely to have higher weights or heart rates. For this reason, although we collected our data in $\mathbb{R}^4$, in reality it may—to some approximate degree—live in a lower-dimensional space capturing the relationships between the different dimensions.

For now, suppose that there exists a *one*-dimensional space approximating our dataset, illustrated in Figure 6.1(b). Then, we expect that there exists some vector $\vec{v}$ such that each data point $\vec{x}_i$ can be written as $\vec{x}_i \approx c_i\vec{v}$ for a different $c_i \in \mathbb{R}$. From before, we know that the best approximation of $\vec{x}_i$ parallel to $\vec{v}$ is $\text{proj}_{\vec{v}}\, \vec{x}_i$. Defining $\hat{v} \equiv \vec{v}/\|\vec{v}\|_2$, we can write

$$\begin{aligned}\text{proj}_{\vec{v}}\, \vec{x}_i &= \frac{\vec{x}_i \cdot \vec{v}}{\vec{v} \cdot \vec{v}}\vec{v} \text{ by definition} \\ &= (\vec{x}_i \cdot \hat{v})\hat{v} \text{ since } \vec{v} \cdot \vec{v} = \|\vec{v}\|_2^2.\end{aligned}$$

The magnitude of $\vec{v}$ does not matter for the problem at hand, since the projection of $\vec{x}_i$ onto any nonzero multiple of $\hat{v}$ is the same, so it is reasonable to restrict our search to the space of *unit* vectors $\hat{v}$.

Following the pattern of least-squares, we have a new optimization problem:

$$\begin{aligned}&\text{minimize}_{\hat{v}} \sum_i \|\vec{x}_i - \text{proj}_{\hat{v}}\, \vec{x}_i\|_2^2 \\ &\text{subject to } \|\hat{v}\|_2 = 1.\end{aligned}$$

This problem minimizes the sum of squared differences between the data points $\vec{x}_i$ and their best approximation as a multiple of $\hat{v}$, as in Figure 6.1(c). We can simplify our optimization objective using the observations we already have made and some linear algebra:

$$\begin{aligned}
\sum_i \|\vec{x}_i - \text{proj}_{\hat{v}}\, \vec{x}_i\|_2^2 &= \sum_i \|\vec{x}_i - (\vec{x}_i \cdot \hat{v})\hat{v}\|_2^2 \text{ as explained above} \\
&= \sum_i \left(\|\vec{x}_i\|_2^2 - 2(\vec{x}_i \cdot \hat{v})(\vec{x}_i \cdot \hat{v}) + (\vec{x}_i \cdot \hat{v})^2 \|\hat{v}\|_2^2 \right) \text{ since } \|\vec{w}\|_2^2 = \vec{w} \cdot \vec{w} \\
&= \sum_i \left(\|\vec{x}_i\|_2^2 - (\vec{x}_i \cdot \hat{v})^2 \right) \text{ since } \|\hat{v}\|_2 = 1 \\
&= \text{const.} - \sum_i (\vec{x}_i \cdot \hat{v})^2 \text{ since the unknown here is } \hat{v} \\
&= \text{const.} - \|X^\top \hat{v}\|_2^2, \text{ where the columns of } X \text{ are the vectors } \vec{x}_i.
\end{aligned}$$

After removing the negative sign, this derivation shows that we can solve an equivalent maximization problem:

$$\begin{aligned}
&\text{maximize } \|X^\top \hat{v}\|_2^2 \\
&\text{subject to } \|\hat{v}\|_2^2 = 1.
\end{aligned}$$

Statisticians may recognize this equivalence as maximizing variance rather than minimizing approximation error.

We know $\|X^\top \hat{v}\|_2^2 = \hat{v}^\top X X^\top \hat{v}$, so by Example 1.27, $\hat{v}$ is the eigenvector of $XX^\top$ with the highest eigenvalue. The vector $\hat{v}$ is known as the first *principal component* of the dataset.

6.1.2 Differential Equations

Many physical forces can be written as functions of position. For instance, the force exerted by a spring connecting two particles at positions $\vec{x}, \vec{y} \in \mathbb{R}^3$ is $k(\vec{x} - \vec{y})$ by Hooke's Law; such spring forces are used to approximate forces holding cloth together in many simulation systems for computer graphics. Even when forces are not *linear* in position, we often approximate them in a linear fashion. In particular, in a physical system with n particles, we can encode the positions of all the particles simultaneously in a vector $\vec{X} \in \mathbb{R}^{3n}$. Then, the forces in the system might be approximated as $\vec{F} \approx A\vec{X}$ for some matrix $A \in \mathbb{R}^{3n \times 3n}$.

Newton's second law of motion states $F = ma$, or force equals mass times acceleration. In our context, we can write a diagonal *mass matrix* $M \in \mathbb{R}^{3n \times 3n}$ containing the mass of each particle in the system. Then, the second law can be written as $\vec{F} = M\vec{X}''$, where prime denotes differentiation in time. By definition, $\vec{X}'' = (\vec{X}')'$, so after defining $\vec{V} \equiv \vec{X}'$ we have a *first*-order system of equations:

$$\frac{d}{dt} \begin{pmatrix} \vec{X} \\ \vec{V} \end{pmatrix} = \begin{pmatrix} 0 & I_{3n \times 3n} \\ M^{-1}A & 0 \end{pmatrix} \begin{pmatrix} \vec{X} \\ \vec{V} \end{pmatrix}.$$

Here, we simultaneously compute both positions in $\vec{X} \in \mathbb{R}^{3n}$ and velocities $\vec{V} \in \mathbb{R}^{3n}$ of all n particles as functions of time; we will explore this reduction in more detail in Chapter 15.

Beyond this reduction, differential equations of the form $\vec{y}' = B\vec{y}$ for an unknown function $\vec{y}(t)$ and fixed matrix B appear in simulation of cloth, springs, heat, waves, and other

(a) Database of photos (b) Spectral embedding

Figure 6.2 Suppose we are given (a) an unsorted database of photographs with some matrix W measuring the similarity between image i and image j. (b) The one-dimensional spectral embedding assigns each photograph i a value x_i so that if images i and j are similar, then x_i will be close to x_j (figure generated by D. Hyde).

phenomena. Suppose we know eigenvectors $\vec{y}_1, \ldots, \vec{y}_k$ of B satisfying $B\vec{y}_i = \lambda_i \vec{y}_i$. If we write the initial condition of the differential equation in terms of the eigenvectors as

$$\vec{y}(0) = c_1\vec{y}_1 + \cdots + c_k\vec{y}_k,$$

then the solution of the differential equation can be written in closed form:

$$\vec{y}(t) = c_1 e^{\lambda_1 t}\vec{y}_1 + \cdots + c_k e^{\lambda_k t}\vec{y}_k.$$

That is, if we expand the initial conditions of this differential equation in the eigenvector basis, then we know the solution for all times $t \geq 0$ for free; you will check this formula in Exercise 6.1. This formula is not the end of the story for simulation: Finding the complete set of eigenvectors of B is expensive, and B may evolve over time.

6.1.3 Spectral Embedding

Suppose we have a collection of n items in a dataset and a measure $w_{ij} \geq 0$ of how similar elements i and j are; we will assume $w_{ij} = w_{ji}$. For instance, maybe we are given a collection of photographs as in Figure 6.2(a) and take w_{ij} to be a measure of the amount of overlap between the distributions of colors in photo i and in photo j.

Given the matrix W of w_{ij} values, we might wish to sort the photographs based on their similarity to simplify viewing and exploring the collection. That is, we could lay them out on a line so that the pair of photos i and j is close when w_{ij} is large, as in Figure 6.2(b). The measurements in w_{ij} may be noisy or inconsistent, however, so it may not be obvious how to sort the n photos directly using the n^2 values in W.

One way to order the collection would be to assign a number x_i to each item i such that similar objects are assigned similar numbers; we can then sort the collection based on the values in $\vec{x}$. We can measure how well an assignment of values in $\vec{x}$ groups similar objects by using the energy function

$$E(\vec{x}) \equiv \sum_{ij} w_{ij}(x_i - x_j)^2.$$

The difference $(x_i - x_j)^2$ is small when x_i and x_j are assigned similar values. Given the weighting w_{ij} next to $(x_i - x_j)^2$, minimizing $E(\vec{x})$ asks that items i and j with high similarity scores w_{ij} get mapped the closest.

Minimizing $E(\vec{x})$ with no constraints gives a minimum $\vec{x}$ with $E(\vec{x}) = 0$: $x_i = \text{const.}$ for all i. Furthermore, adding a constraint $\|\vec{x}\|_2 = 1$ does *not* remove this constant solution: Taking $x_i = 1/\sqrt{n}$ for all i gives $\|\vec{x}\|_2 = 1$ and $E(\vec{x}) = 0$. To obtain a nontrivial output, we must remove this case as well:

$$\begin{aligned} &\text{minimize } E(\vec{x}) \\ &\text{subject to } \|\vec{x}\|_2^2 = 1 \\ &\qquad \vec{1} \cdot \vec{x} = 0. \end{aligned}$$

Our second constraint requires that the sum of elements in $\vec{x}$ is zero, preventing the choice $x_1 = x_2 = \cdots = x_n$ when combined with the $\|\vec{x}\|_2 = 1$ constraint.

We can simplify the energy in a few steps:

$$\begin{aligned} E(\vec{x}) &= \sum_{ij} w_{ij}(x_i - x_j)^2 \text{ by definition} \\ &= \sum_{ij} w_{ij}(x_i^2 - 2x_i x_j + x_j^2) \\ &= \sum_i a_i x_i^2 - 2\sum_{ij} w_{ij} x_i x_j + \sum_j a_j x_j^2 \text{ where } \vec{a} \equiv W\vec{1}, \text{ since } W^\top = W \\ &= 2\vec{x}^\top (A - W)\vec{x} \text{ where } A \equiv \text{diag}(\vec{a}). \end{aligned}$$

We can check that $\vec{1}$ is an eigenvector of $A - W$ with eigenvalue 0:

$$(A - W)\vec{1} = A\vec{1} - W\vec{1} = \vec{a} - \vec{a} = \vec{0}.$$

More interestingly, the eigenvector corresponding to the *second*-smallest eigenvalue is the minimizer for our constrained problem above! One way to see this fact is to write the Lagrange multiplier function corresponding to this optimization:

$$\Lambda \equiv 2\vec{x}^\top (A - W)\vec{x} - \lambda(1 - \|\vec{x}\|_2^2) - \mu(\vec{1} \cdot \vec{x}).$$

Applying Theorem 1.1, at the optimal point we must have:

$$\begin{aligned} 0 &= \nabla_{\vec{x}} \Lambda = 4(A - W)\vec{x} + 2\lambda\vec{x} - \mu\vec{1} \\ 1 &= \|\vec{x}\|_2^2 \\ 0 &= \vec{1} \cdot \vec{x}. \end{aligned}$$

If we take the dot product of both sides of the first expression with $\vec{1}$ shows

$$\begin{aligned} 0 &= \vec{1} \cdot [4(A - W)\vec{x} + 2\lambda\vec{x} - \mu\vec{1}] \\ &= 4\vec{1}^\top (A - W)\vec{x} - \mu n \text{ since } \vec{1} \cdot \vec{x} = 0 \\ &= -\mu n \text{ since } A\vec{1} = W\vec{1} = \vec{a} \\ \implies \mu &= 0. \end{aligned}$$

Substituting this new observation into the Lagrange multiplier condition, we find:

$$2(W - A)\vec{x} = \lambda\vec{x}.$$

We explicitly ignore the eigenvalue $\lambda = 0$ of $W - A$ corresponding to the eigenvector $\vec{1}$, so $\vec{x}$ must be the eigenvector with the *second*-smallest eigenvalue. The resulting $\vec{x}$ is the "spectral embedding" of W onto one dimension, referring to the fact that we call the set of eigenvalues of a matrix its spectrum. Taking more eigenvectors of $A - W$ provides embeddings into higher dimensions.

6.2 PROPERTIES OF EIGENVECTORS

We have established a variety of applications in need of eigenspace computation. Before we can explore algorithms for this purpose, however, we will more closely examine the structure of the eigenvalue problem.

We can begin with a few definitions that likely are evident at this point:

Definition 6.1 (Eigenvalue and eigenvector)**.** An *eigenvector* $\vec{x} \neq \vec{0}$ of a matrix $A \in \mathbb{R}^{n\times n}$ is any vector satisfying $A\vec{x} = \lambda\vec{x}$ for some $\lambda \in \mathbb{R}$; the corresponding λ is known as an *eigenvalue*. *Complex* eigenvalues and eigenvectors satisfy the same relationships with $\lambda \in \mathbb{C}$ and $\vec{x} \in \mathbb{C}^n$.

Definition 6.2 (Spectrum and spectral radius)**.** The *spectrum* of A is the set of eigenvalues of A. The *spectral radius* $\rho(A)$ is the maximum value $|\lambda|$ over all eigenvalues λ of A.

The scale of an eigenvector is not important. In particular, $A(c\vec{x}) = cA\vec{x} = c\lambda\vec{x} = \lambda(c\vec{x})$, so $c\vec{x}$ is an eigenvector with the same eigenvalue. For this reason, we can restrict our search to those eigenvectors $\vec{x}$ with $\|\vec{x}\|_2 = 1$ without losing any nontrivial structure. Adding this constraint does not completely relieve ambiguity, since $\pm\vec{x}$ are both eigenvectors with the same eigenvalue, but this case is easier to detect.

The algebraic properties of eigenvectors and eigenvalues are the subject of many mathematical studies in themselves. Some basic properties will suffice for the discussion at hand, and hence we will mention just a few theorems affecting the design of numerical algorithms. The proofs here parallel the development in [4].

First, we should check that every matrix has at least one eigenvector, so that our search for eigenvectors is not in vain. Our strategy for this and other related problems is to notice that λ is an eigenvalue such that $A\vec{x} = \lambda\vec{x}$ if and only if $(A - \lambda I_{n\times n})\vec{x} = \vec{0}$. That is, λ is an eigenvalue of A exactly when the matrix $A - \lambda I_{n\times n}$ is not full-rank.

Proposition 6.1 ([4], Theorem 2.1)**.** Every matrix $A \in \mathbb{R}^{n\times n}$ has at least one (potentially complex) eigenvector.

Proof. Take any vector $\vec{x} \in \mathbb{R}^n\backslash\{\vec{0}\}$, and assume $A \neq 0$ since this matrix trivially has eigenvalue 0. The set $\{\vec{x}, A\vec{x}, A^2\vec{x}, \cdots, A^n\vec{x}\}$ must be linearly dependent because it contains $n+1$ vectors in n dimensions. So, there exist constants $c_0, \ldots, c_n \in \mathbb{R}$ not all zero such that $\vec{0} = c_0\vec{x} + c_1A\vec{x} + \cdots + c_nA^n\vec{x}$. Define a polynomial

$$f(z) \equiv c_0 + c_1z + \cdots + c_nz^n.$$

By the Fundamental Theorem of Algebra, there exist $m \geq 1$ roots $z_i \in \mathbb{C}$ and $c \neq 0$ such that

$$f(z) = c(z - z_1)(z - z_2)\cdots(z - z_m).$$

Applying this factorization,

$$\begin{aligned}\vec{0} &= c_0\vec{x} + c_1A\vec{x} + \cdots + c_nA^n\vec{x}\\ &= (c_0I_{n\times n} + c_1A + \cdots + c_nA^n)\vec{x}\\ &= c(A - z_1I_{n\times n})\cdots(A - z_mI_{n\times n})\vec{x}.\end{aligned}$$

In this form, at least one $A - z_iI_{n\times n}$ has a null space, since otherwise each term would be invertible, forcing $\vec{x} = \vec{0}$. If we take $\vec{v}$ to be a nonzero vector in the null space of $A - z_iI_{n\times n}$, then by construction $A\vec{v} = z_i\vec{v}$, as needed. □

There is one additional fact worth checking to motivate our discussion of eigenvector computation. While it can be the case that a single eigenvalue admits more than one corresponding eigenvector, when two eigenvectors have different eigenvalues they cannot be related in the following sense:

Proposition 6.2 ([4], Proposition 2.2)**.** Eigenvectors corresponding to different eigenvalues must be linearly independent.

Proof. Suppose this is not the case. Then there exist eigenvectors $\vec{x}_1, \ldots, \vec{x}_k$ with distinct eigenvalues $\lambda_1, \ldots, \lambda_k$ that are linearly dependent. This implies that there are coefficients $c_1, \ldots, c_k$ not all zero with $\vec{0} = c_1\vec{x}_1 + \cdots + c_k\vec{x}_k$.

For any two indices i and j, since $A\vec{x}_j = \lambda_j\vec{x}_j$, we can simplify the product

$$(A - \lambda_i I_{n\times n})\vec{x}_j = A\vec{x}_j - \lambda_i\vec{x}_j = \lambda_j\vec{x}_j - \lambda_i\vec{x}_j = (\lambda_j - \lambda_i)\vec{x}_j.$$

Pre-multiplying the relationship $\vec{0} = c_1\vec{x}_1 + \cdots + c_k\vec{x}_k$ by the matrix $(A - \lambda_2 I_{n\times n}) \cdots (A - \lambda_k I_{n\times n})$ shows:

$$\begin{aligned}\vec{0} &= (A - \lambda_2 I_{n\times n}) \cdots (A - \lambda_k I_{n\times n})(c_1\vec{x}_1 + \cdots + c_k\vec{x}_k) \\ &= c_1(\lambda_1 - \lambda_2) \cdots (\lambda_1 - \lambda_k)\vec{x}_1.\end{aligned}$$

Since all the λ_i's are distinct, this shows $c_1 = 0$. The same argument shows that the rest of the c_i's have to be zero, contradicting linear dependence. □

This proposition shows that an $n\times n$ matrix can have at most n distinct eigenvalues, since a set of n eigenvalues yields n linearly independent vectors. The maximum number of linearly independent eigenvectors corresponding to an eigenvalue λ is the *geometric multiplicity* of λ. It is not true, however, that a matrix has to have *exactly* n linearly independent eigenvectors. This is the case for many matrices, which we will call *nondefective*:

Definition 6.3 (Nondefective)**.** A matrix $A \in \mathbb{R}^{n\times n}$ is *nondefective* or *diagonalizable* if its eigenvectors span $\mathbb{R}^n$.

Example 6.1 (Defective matrix)**.** The matrix

$$\begin{pmatrix} 5 & 2 \\ 0 & 5 \end{pmatrix}$$

has only one linearly independent eigenvector $(1, 0)$.

We call nondefective matrices *diagonalizable* for the following reason: If a matrix is nondefective, then it has n eigenvectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$ with corresponding (possibly nonunique) eigenvalues $\lambda_1, \ldots, \lambda_n$. Take the columns of X to be the vectors $\vec{x}_i$, and define D to be the diagonal matrix with $\lambda_1, \ldots, \lambda_n$ along the diagonal. Then, we have $AX = XD$; this relationship is a "stacked" version of $A\vec{x}_i = \lambda_i\vec{x}_i$. Applying X^{-1} to both sides, $D = X^{-1}AX$, meaning A is diagonalized by a *similarity transformation* $A \mapsto X^{-1}AX$:

Definition 6.4 (Similar matrices)**.** Two matrices A and B are *similar* if there exists T with $B = T^{-1}AT$.

Similar matrices have the same eigenvalues, since if $B\vec{x} = \lambda x$, by substituting $B = T^{-1}AT$ we know $T^{-1}AT\vec{x} = \lambda\vec{x}$. Hence, $A(T\vec{x}) = \lambda(T\vec{x})$, showing $T\vec{x}$ is an eigenvector of A with eigenvalue λ. In other words:

We can apply all the similarity transformations we want to a matrix without modifying its set of eigenvalues.

This observation is the foundation of many eigenvector computation methods, which start with a general matrix A and reduce it to a matrix whose eigenvalues are more obvious by applying similarity transformations. This procedure is analogous to applying row operations to reduce a matrix to triangular form for use in solving linear systems of equations.

6.2.1 Symmetric and Positive Definite Matrices

Unsurprisingly given our special consideration of Gram matrices $A^\top A$ in previous chapters, symmetric and/or positive definite matrices enjoy special eigenvector structure. If we can verify *a priori* that a matrix is symmetric or positive definite, specialized algorithms can be used to extract its eigenvectors more quickly.

Our original definition of eigenvalues allows them to be complex values in $\mathbb{C}$ even if A is a real matrix. We can prove, however, that in the symmetric case we do not need complex arithmetic. To do so, we will generalize symmetric matrices to matrices in $\mathbb{C}^{n\times n}$ by introducing the set of *Hermitian* matrices:

Definition 6.5 (Complex conjugate)**.** The *complex conjugate* of a number $z = a+bi \in \mathbb{C}$, where $a, b \in \mathbb{R}$, is $\bar{z} \equiv a - bi$. The complex conjugate $\bar{A}$ of a matrix $A \in \mathbb{C}^{m\times n}$ is the matrix with elements $\bar{a}_{ij}$.

Definition 6.6 (Conjugate transpose)**.** The *conjugate transpose* of $A \in \mathbb{C}^{m\times n}$ is $A^H \equiv \bar{A}^\top$.

Definition 6.7 (Hermitian matrix)**.** A matrix $A \in \mathbb{C}^{n\times n}$ is *Hermitian* if $A = A^H$.

A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is automatically Hermitian because it has no complex part.

We also can generalize the notion of a dot product to complex vectors by defining an *inner product* as follows:

$$\langle \vec{x}, \vec{y} \rangle \equiv \sum_i x_i \bar{y}_i,$$

where $\vec{x}, \vec{y} \in \mathbb{C}^n$. Once again, this definition coincides with $\vec{x} \cdot \vec{y}$ when $\vec{x}, \vec{y} \in \mathbb{R}^n$; in the complex case, however, dot product symmetry is replaced by the condition $\langle \vec{v}, \vec{w} \rangle = \overline{\langle \vec{w}, \vec{v} \rangle}$.

We now can prove that it is not necessary to search for complex eigenvalues of symmetric or Hermitian matrices:

Proposition 6.3. All eigenvalues of Hermitian matrices are real.

Proof. Suppose $A \in \mathbb{C}^{n\times n}$ is Hermitian with $A\vec{x} = \lambda\vec{x}$. By scaling, we can assume $\|\vec{x}\|_2^2 = \langle \vec{x}, \vec{x} \rangle = 1$. Then:

$$\begin{aligned}
\lambda &= \lambda\langle \vec{x}, \vec{x} \rangle \text{ since } \vec{x} \text{ has norm 1} \\
&= \langle \lambda\vec{x}, \vec{x} \rangle \text{ by linearity of } \langle \cdot, \cdot \rangle \\
&= \langle A\vec{x}, \vec{x} \rangle \text{ since } A\vec{x} = \lambda\vec{x} \\
&= (A\vec{x})^\top \bar{\vec{x}} \text{ by definition of } \langle \cdot, \cdot \rangle \\
&= \vec{x}^\top \overline{(\bar{A}^\top \vec{x})} \text{ by expanding the product and applying the identity } \overline{ab} = \bar{a}\bar{b} \\
&= \langle \vec{x}, A^H\vec{x} \rangle \text{ by definition of } A^H \text{ and } \langle \cdot, \cdot \rangle
\end{aligned}$$

$$
\begin{aligned}
&= \langle \vec{x}, A\vec{x} \rangle \text{ since } A = A^H \\
&= \bar{\lambda} \langle \vec{x}, \vec{x} \rangle \text{ since } A\vec{x} = \lambda \vec{x} \\
&= \bar{\lambda} \text{ since } \vec{x} \text{ has norm 1.}
\end{aligned}
$$

Thus $\lambda = \bar{\lambda}$, which can happen only if $\lambda \in \mathbb{R}$, as needed. □

Not only are the eigenvalues of Hermitian (and symmetric) matrices real, but also their eigenvectors must be orthogonal:

Proposition 6.4. Eigenvectors corresponding to distinct eigenvalues of Hermitian matrices must be orthogonal.

Proof. Suppose $A \in \mathbb{C}^{n \times n}$ is Hermitian, and suppose $\lambda \neq \mu$ with $A\vec{x} = \lambda \vec{x}$ and $A\vec{y} = \mu \vec{y}$. By the previous proposition we know $\lambda, \mu \in \mathbb{R}$. Then, $\langle A\vec{x}, \vec{y} \rangle = \lambda \langle \vec{x}, \vec{y} \rangle$. But since A is Hermitian we can also write $\langle A\vec{x}, \vec{y} \rangle = \langle \vec{x}, A^H \vec{y} \rangle = \langle \vec{x}, A\vec{y} \rangle = \mu \langle \vec{x}, \vec{y} \rangle$. Thus, $\lambda \langle \vec{x}, \vec{y} \rangle = \mu \langle \vec{x}, \vec{y} \rangle$. Since $\lambda \neq \mu$, we must have $\langle \vec{x}, \vec{y} \rangle = 0$. □

Finally, we state (without proof) a crowning result of linear algebra, the Spectral Theorem. This theorem states that all symmetric or Hermitian matrices are non-defective and therefore must have exactly n orthogonal eigenvectors.

Theorem 6.1 (Spectral Theorem)**.** Suppose $A \in \mathbb{C}^{n \times n}$ is Hermitian (if $A \in \mathbb{R}^{n \times n}$, suppose it is symmetric). Then, A has exactly n orthonormal eigenvectors $\vec{x}_1, \cdots, \vec{x}_n$ with (possibly repeated) eigenvalues $\lambda_1, \ldots, \lambda_n$. In other words, there exists an orthogonal matrix X of eigenvectors and diagonal matrix D of eigenvalues such that $D = X^\top A X$.

This theorem implies that any $\vec{y} \in \mathbb{R}^n$ can be decomposed into a linear combination of the eigenvectors of a Hermitian A. Many calculations are easier in this basis, as shown below:

Example 6.2 (Computation using eigenvectors)**.** Take $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$ to be the unit-length eigenvectors of a symmetric invertible matrix $A \in \mathbb{R}^{n \times n}$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$. Suppose we wish to solve $A\vec{y} = \vec{b}$. By the Spectral Theorem, we can decompose $\vec{b} = c_1 \vec{x}_1 + \cdots + c_n \vec{x}_n$, where $c_i = \vec{b} \cdot \vec{x}_i$ by orthonormality. Then,

$$\vec{y} = \frac{c_1}{\lambda_1} \vec{x}_1 + \cdots + \frac{c_n}{\lambda_n} \vec{x}_n.$$

The fastest way to check this formula is to multiply $\vec{y}$ by A and make sure we recover $\vec{b}$:

$$
\begin{aligned}
A\vec{y} &= A \left(\frac{c_1}{\lambda_1} \vec{x}_1 + \cdots + \frac{c_n}{\lambda_n} \vec{x}_n \right) \\
&= \frac{c_1}{\lambda_1} A\vec{x}_1 + \cdots + \frac{c_n}{\lambda_n} A\vec{x}_n \\
&= c_1 \vec{x}_1 + \cdots + c_n \vec{x}_n \text{ since } A\vec{x}_k = \lambda_k \vec{x}_k \text{ for all } k \\
&= \vec{b}, \text{ as desired.}
\end{aligned}
$$

The calculation above has both positive and negative implications. It shows that given the eigenvectors and eigenvalues of symmetric matrix A, operations like inversion become straightforward. On the flip side, this means that finding the full set of eigenvectors of a symmetric matrix A is "at least" as difficult as solving $A\vec{x} = \vec{b}$.

Returning from our foray into the complex numbers, we revisit the real numbers to prove one final useful fact about positive definite matrices:

Proposition 6.5. All eigenvalues of positive definite matrices are positive.

Proof. Take $A \in \mathbb{R}^{n\times n}$ positive definite, and suppose $A\vec{x} = \lambda\vec{x}$ with $\|\vec{x}\|_2 = 1$. By positive definiteness, we know $\vec{x}^\top A\vec{x} > 0$. But, $\vec{x}^\top A\vec{x} = \vec{x}^\top(\lambda\vec{x}) = \lambda\|\vec{x}\|_2^2 = \lambda$, as needed. □

This property is not nearly as remarkable as those associated with symmetric or Hermitian matrices, but it helps *order* the eigenvalues of A. Positive definite matrices enjoy the property that the eigenvalue with smallest absolute value is also the eigenvalue closest to zero, and the eigenvalue with largest absolute value is the one farthest from zero. This property influences methods that seek only a subset of the eigenvalues of a matrix, usually at one of the two ends of its spectrum.

6.2.2 Specialized Properties

We mention some specialized properties of eigenvectors and eigenvalues that influence more advanced methods for their computation. They largely will not figure into our subsequent discussion, so this section can be skipped if readers lack sufficient background.

6.2.2.1 Characteristic Polynomial

The determinant of a matrix $\det A$ satisfies $\det A \neq 0$ if and only if A is invertible. Thus, one way to find eigenvalues of a matrix is to find roots of the *characteristic polynomial*

$$p_A(\lambda) = \det(A - \lambda I_{n\times n}).$$

We have chosen to avoid determinants in our discussion of linear algebra, but simplifying p_A reveals that it is an n-th degree polynomial in λ.

From this construction, we can define the *algebraic multiplicity* of an eigenvalue λ as its multiplicity as a root of p_A. The algebraic multiplicity of any eigenvalue is at least as large as its geometric multiplicity. If the algebraic multiplicity is 1, the root is called *simple*, because it corresponds to a single eigenvector that is linearly independent from any others. Eigenvalues for which the algebraic and geometric multiplicities are not equal are called *defective*, since the corresponding matrix must also be defective in the sense of Definition 6.3.

In numerical analysis, it is common to avoid using the determinant of a matrix. While it is a convenient theoretical construction, its practical applicability is limited. Determinants are difficult to compute. In fact, most eigenvalue algorithms do not attempt to find roots of p_A directly, since doing so would require evaluation of a determinant. Furthermore, the determinant $\det A$ has *nothing* to do with the conditioning of A, so a near-but-not-exactly zero determinant of $\det(A - \lambda I_{n\times n})$ might not show that λ is nearly an eigenvalue of A.

6.2.2.2 Jordan Normal Form

We can only diagonalize a matrix when it has a full eigenspace. All matrices, however, are similar to a matrix in Jordan normal form, a general layout satisfying the following criteria:

- Nonzero values are on the diagonal entries a_{ii} and on the "superdiagonal" $a_{i(i+1)}$.
- Diagonal values are eigenvalues repeated as many times as their algebraic multiplicity; the matrix is block diagonal about these clusters.
- Off-diagonal values are 1 or 0.

Thus, the shape looks something like the following:

$$\begin{pmatrix} \lambda_1 & 1 & & & & & & \\ & \lambda_1 & 1 & & & & & \\ & & \lambda_1 & & & & & \\ & & & \lambda_2 & 1 & & & \\ & & & & \lambda_2 & & & \\ & & & & & \lambda_3 & & \\ & & & & & & \ddots & \end{pmatrix}.$$

Jordan normal form is attractive theoretically because it always exists, but the $1/0$ structure is discrete and unstable under numerical perturbation.

6.3 COMPUTING A SINGLE EIGENVALUE

Computing the eigenvalues of a matrix is a well-studied problem with many potential algorithmic approaches. Each is tuned for a different situation, and achieving near-optimal conditioning or speed requires experimentation with several techniques. Here, we cover a few popular algorithms for the eigenvalue problem encountered in practice.

6.3.1 Power Iteration

Assume that $A \in \mathbb{R}^{n\times n}$ is non-defective and nonzero with all real eigenvalues, e.g., A is symmetric. By definition, A has a full set of eigenvectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$; we sort them such that their corresponding eigenvalues satisfy $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Take an arbitrary vector $\vec{v} \in \mathbb{R}^n$. Since the eigenvectors of A span $\mathbb{R}^n$, we can write $\vec{v}$ in the $\vec{x}_i$ basis as $\vec{v} = c_1\vec{x}_1 + \cdots + c_n\vec{x}_n$. Applying A to both sides,

$$\begin{aligned} A\vec{v} &= c_1A\vec{x}_1 + \cdots + c_nA\vec{x}_n \\ &= c_1\lambda_1\vec{x}_1 + \cdots + c_n\lambda_n\vec{x}_n \text{ since } A\vec{x}_i = \lambda_i\vec{x}_i \\ &= \lambda_1\left(c_1\vec{x}_1 + \frac{\lambda_2}{\lambda_1}c_2\vec{x}_2 + \cdots + \frac{\lambda_n}{\lambda_1}c_n\vec{x}_n\right) \\ A^2\vec{v} &= \lambda_1^2\left(c_1\vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^2 c_2\vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^2 c_n\vec{x}_n\right) \\ &\vdots \\ A^k\vec{v} &= \lambda_1^k\left(c_1\vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k c_2\vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k c_n\vec{x}_n\right). \end{aligned}$$

As $k \to \infty$, the ratio $(\lambda_i/\lambda_1)^k \to 0$ unless $\lambda_i = \pm\lambda_1$, since λ_1 has the largest magnitude of any eigenvalue by construction. If $\vec{x}$ is the projection of $\vec{v}$ onto the space of eigenvectors with eigenvalues λ_1, then—at least when the absolute values $|\lambda_i|$ are unique—as $k \to \infty$ the following approximation begins to dominate: $A^k\vec{v} \approx \lambda_1^k\vec{x}$.

This argument leads to an exceedingly simple algorithm for computing a single eigenvector $\vec{x}_1$ of A corresponding to its largest-magnitude eigenvalue λ_1:

1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
2. Iterate until convergence for increasing k: $\vec{v}_k = A\vec{v}_{k-1}$

```
function POWER-ITERATION(A)
    v ← ARBITRARY(n)
    for k ← 1, 2, 3, ...
        v ← Av
    return v
```

(a)

```
function NORMALIZED-ITERATION(A)
    v ← ARBITRARY(n)
    for k ← 1, 2, 3, ...
        w ← Av
        v ← w/‖w‖
    return v
```

(b)

Figure 6.3 Power iteration (a) without and (b) with normalization for finding the largest eigenvalue of a matrix.

```
function INVERSE-ITERATION(A)
    v ← ARBITRARY(n)
    for k ← 1, 2, 3, ...
        w ← A^{-1}v
        v ← w/‖w‖
    return v
```

(a)

```
function INVERSE-ITERATION-LU(A)
    v ← ARBITRARY(n)
    L, U ← LU-FACTORIZE(A)
    for k ← 1, 2, 3, ...
        y ← FORWARD-SUBSTITUTE(L, v)
        w ← BACK-SUBSTITUTE(U, y)
        v ← w/‖w‖
    return v
```

(b)

Figure 6.4 Inverse iteration (a) without and (b) with LU factorization.

This algorithm, known as *power iteration* and detailed in Figure 6.3(a), produces vectors $\vec{v}_k$ more and more parallel to the desired $\vec{x}_1$ as $k \to \infty$. Although we have not considered the defective case here, it is still guaranteed to converge; see [98] for a more advanced discussion.

One time that this technique may fail is if we accidentally choose $\vec{v}_1$ such that $c_1 = 0$, but the odds of this peculiarity occurring are vanishingly small. Such a failure mode only occurs when the initial guess has no component parallel to $\vec{x}_1$. Also, while power iteration can succeed in the presence of repeated eigenvalues, it can fail if λ and $-\lambda$ are both eigenvalues of A with the largest magnitude. In the absence of these degeneracies, the rate of convergence for power iteration depends on the decay rate of terms 2 to n in the sum above for $A^k\vec{v}$ and hence is determined by the ratio of the second-largest-magnitude eigenvalue of A to the largest.

If $|\lambda_1| > 1$, however, then $\|\vec{v}_k\| \to \infty$ as $k \to \infty$, an undesirable property for floating-point arithmetic. We only care about the *direction* of the eigenvector rather than its magnitude, so scaling has no effect on the quality of our solution. To avoid dealing with large-magnitude vectors, we can normalize $\vec{v}_k$ at each step, producing the *normalized power iteration* algorithm in Figure 6.3(b). In the algorithm listing, we purposely do not decorate the norm $\|\cdot\|$ with a particular subscript. Mathematically, *any* norm will suffice for preventing $\vec{v}_k$ from going to infinity, since we have shown that all norms on $\mathbb{R}^n$ are equivalent. In practice, we often use the *infinity* norm $\|\cdot\|_\infty$; this choice has the convenient property that during iteration $\|A\vec{v}_k\|_\infty \to |\lambda_1|$.

6.3.2 Inverse Iteration

We now have an iterative algorithm for approximating the *largest*-magnitude eigenvalue λ_1 of a matrix A. Suppose A is invertible, so that we can evaluate $\vec{y} = A^{-1}\vec{v}$ by solving $A\vec{y} = \vec{v}$

```
function RAYLEIGH-QUOTIENT-ITERATION(A, σ)
  v ← ARBITRARY(n)
  for k ← 1, 2, 3, ...
    w ← (A − σI_{n×n})^{-1} v
    v ← w/‖w‖
    σ ← (v^T A v)/‖v‖_2^2
  return v
```

Figure 6.5 Rayleigh quotient iteration for finding an eigenvalue close to an initial guess σ.

using techniques covered in previous chapters. If $A\vec{x} = \lambda\vec{x}$, then $\vec{x} = \lambda A^{-1}\vec{x}$, or equivalently $A^{-1}\vec{x} = \frac{1}{\lambda}\vec{x}$. Thus, $1/\lambda$ is an eigenvalue of A^{-1} with eigenvector $\vec{x}$.

If $|a| \geq |b|$ then $|b|^{-1} \geq |a|^{-1}$, so the smallest-magnitude eigenvalue of A is the *largest*-magnitude eigenvector of A^{-1}. This construction yields an algorithm for finding λ_n rather than λ_1 called *inverse power iteration*, in Figure 6.4(a). This iterative scheme is nothing more than the power method from §6.3.1 applied to A^{-1}.

We repeatedly are solving systems of equations using the same matrix A but different right-hand sides, a perfect application of factorization techniques from previous chapters. For instance, if we write $A = LU$, then we could formulate an equivalent but considerably more efficient version of inverse power iteration illustrated in Figure 6.4(b). With this simplification, each solve for $A^{-1}\vec{v}$ is carried out in two steps, first by solving $L\vec{y} = \vec{v}$ and then by solving $U\vec{w} = \vec{y}$ as suggested in §3.5.1.

6.3.3 Shifting

Suppose λ_2 is the eigenvalue of A with second-largest magnitude. Power iteration converges fastest when $|\lambda_2/\lambda_1|$ is small, since in this case the power $(\lambda_2/\lambda_1)^k$ decays quickly. If this ratio is nearly 1, it may take many iterations before a single eigenvector is isolated.

If the eigenvalues of A are $\lambda_1, \ldots, \lambda_n$ with corresponding eigenvectors $\vec{x}_1, \ldots, \vec{x}_n$, then the eigenvalues of $A - \sigma I_{n\times n}$ are $\lambda_1 - \sigma, \ldots, \lambda_n - \sigma$, since:

$$(A - \sigma I_{n\times n})\vec{x}_i = A\vec{x}_i - \sigma\vec{x}_i = \lambda_i\vec{x}_i - \sigma\vec{x}_i = (\lambda_i - \sigma)\vec{x}_i.$$

With this idea in mind, one way to make power iteration converge quickly is to choose σ such that:

$$\left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|.$$

That is, we find eigenvectors of $A - \sigma I_{n\times n}$ rather than A itself, choosing σ to widen the gap between the first and second eigenvalue to improve convergence rates. Guessing a good σ, however, can be an art, since we do not know the eigenvalues of A *a priori*.

More generally, if we think that σ is near an eigenvalue of A, then $A - \sigma I_{n\times n}$ has an eigenvalue close to 0 that we can reveal by inverse iteration. In other words, to use power iteration to target a particular eigenvalue of A rather than its largest or smallest eigenvalue as in previous sections, we shift A so that the eigenvalue we want is close to zero and then can apply inverse iteration to the result.

If our initial guess of σ is inaccurate, we could try to update it from iteration to iteration of the power method. For example, if we have a fixed guess of an eigenvector $\vec{x}$ of A, then

by the normal equations the least-squares approximation of the corresponding eigenvalue σ is given by

$$\sigma \approx \frac{\vec{x}^\top A\vec{x}}{\|\vec{x}\|_2^2}.$$

This fraction is known as a Rayleigh quotient. Thus, we can attempt to increase convergence by using *Rayleigh quotient iteration*, in Figure 6.5, which uses this approximation for σ to update the shift in each step.

Rayleigh quotient iteration usually takes fewer steps to converge than power iteration given a good starting guess σ, but the matrix $A - \sigma_k I_{n\times n}$ is different each iteration and cannot be prefactored as in Figure 6.4(b). In other words, fewer iterations are necessary but each iteration takes more time. This trade-off makes the Rayleigh method more or less preferable to power iteration with a fixed shift depending on the particular choice and size of A. As an additional caveat, if σ_k is *too good* an estimate of an eigenvalue, the matrix $A - \sigma_k I_{n\times n}$ can become near-singular, causing conditioning issues during inverse iteration; that said, depending on the linear solver, this ill-conditioning may not be a concern because it occurs in the direction of the eigenvector being computed. In the opposite case, it can be difficult to control which eigenvalue is isolated by Rayleigh quotient iteration, especially if the initial guess is inaccurate.

6.4 FINDING MULTIPLE EIGENVALUES

So far, we have described techniques for finding a single eigenvalue/eigenvector pair: power iteration to find the largest eigenvalue, inverse iteration to find the smallest, and shifting to target values in between. For many applications, however, a single eigenvalue will not suffice. Thankfully, we can modify these techniques to handle this case as well.

6.4.1 Deflation

Recall the high-level structure of power iteration: Choose an arbitrary $\vec{v}_1$, and iteratively multiply it by A until only the largest eigenvalue λ_1 survives. Take $\vec{x}_1$ to be the corresponding eigenvector.

We were quick to dismiss an unlikely failure mode of this algorithm when $\vec{v}_1 \cdot \vec{x}_1 = 0$, that is, when the initial eigenvector guess has no component parallel to $\vec{x}_1$. In this case, no matter how many times we apply A, the result will never have a component parallel to $\vec{x}_1$. The probability of choosing such a $\vec{v}_1$ randomly is vanishingly small, so in all but the most pernicious of cases power iteration is a stable technique.

We can turn this drawback on its head to formulate a method for finding more than one eigenvalue of a symmetric matrix A. Suppose we find $\vec{x}_1$ and λ_1 via power iteration as before. After convergence, we can restart power iteration after projecting $\vec{x}_1$ out of the initial guess $\vec{v}_1$. Since the eigenvectors of A are orthogonal, by the argument in §6.3.1, power iteration after this projection will recover its *second*-largest eigenvalue!

Due to finite-precision arithmetic, applying A to a vector may inadvertently introduce a small component parallel to $\vec{x}_1$. We can avoid this effect by projecting in each iteration. This change yields the algorithm in Figure 6.6 for computing the eigenvalues in order of descending magnitude.

The inner loop of projected iteration is equivalent to power iteration on the matrix AP, where P projects out $\vec{v}_1, \ldots, \vec{v}_{\ell-1}$:

$$P\vec{x} = \vec{x} - \text{proj}_{\text{span}\,\{\vec{v}_1,\ldots,\vec{v}_{\ell-1}\}}\,\vec{x}.$$

```
function PROJECTED-ITERATION(symmetric A,k)
  for ℓ ← 1, 2, ..., k
    v_ℓ ← ARBITRARY(n)
    for p ← 1, 2, 3, ...
      u ← v_ℓ − proj_{span{v_1,...,v_{ℓ−1}}} v_ℓ
      w ← Au
      v_ℓ ← w/‖w‖
  return v_1, ..., v_k
```

Figure 6.6 Projection for finding k eigenvectors of a symmetric matrix A with the largest eigenvalues. If $\vec{u} = \vec{0}$ at any point, the remaining eigenvalues of A are all zero.

AP has the same eigenvectors as A with eigenvalues $0, \ldots, 0, \lambda_\ell, \ldots, \lambda_n$. More generally, the method of *deflation* involves modifying the matrix A so that power iteration reveals an eigenvector that has not already been computed. For instance, AP is a modification of A so that the large eigenvalues we already have computed are zeroed out.

Projection can fail if A is asymmetric. Other deflation formulas, however, can work in its place with similar efficiency. For instance, suppose $A\vec{x}_1 = \lambda_1\vec{x}_1$ with $\|\vec{x}_1\|_2 = 1$. Take H to be the Householder matrix (see §5.5) such that $H\vec{x}_1 = \vec{e}_1$, the first standard basis vector. From our discussion in §6.2, similarity transforms do not affect the set of eigenvalues, so we safely can conjugate A by H without changing A's eigenvalues. Consider what happens when we multiply $HAH^\top$ by $\vec{e}_1$:

$$\begin{aligned} HAH^\top\vec{e}_1 &= HAH\vec{e}_1 \text{ since } H \text{ is symmetric} \\ &= HA\vec{x}_1 \text{ since } H\vec{x}_1 = \vec{e}_1 \text{ and } H^2 = I_{n\times n} \\ &= \lambda_1 H\vec{x}_1 \text{ since } A\vec{x}_1 = \lambda_1\vec{x}_1 \\ &= \lambda_1\vec{e}_1 \text{ by definition of } H. \end{aligned}$$

From this chain of equalities, the first column of $HAH^\top$ is $\lambda_1\vec{e}_1$, showing that $HAH^\top$ has the following structure [58]:

$$HAH^\top = \begin{pmatrix} \lambda_1 & \vec{b}^\top \\ \vec{0} & B \end{pmatrix}.$$

The matrix $B \in \mathbb{R}^{(n-1)\times(n-1)}$ has eigenvalues $\lambda_2, \ldots, \lambda_n$. Recursively applying this observation, another algorithm for deflation successively generates smaller and smaller B matrices, with each eigenvalue computed using power iteration.

6.4.2 QR Iteration

Deflation has the drawback that each eigenvector must be computed separately, which can be slow and can accumulate error if individual eigenvalues are not accurate. Our remaining algorithms attempt to find more than one eigenvector simultaneously.

Recall that similar matrices A and $B = T^{-1}AT$ have the same eigenvalues for any invertible T. An algorithm seeking the eigenvalues of A can apply similarity transformations to A with abandon in the same way that Gaussian elimination premultiplies by row operations. Applying T^{-1} may be difficult, however, since it would require inverting T, so to make such a strategy practical we seek T's whose inverses are known.

```
function QR-Iteration(A ∈ ℝ^{n×n})
  for k ← 1, 2, 3, ...
    Q, R ← QR-Factorize(A)
    A ← RQ
  return diag(R)
```

Figure 6.7 QR iteration for finding all the eigenvalues of A in the non-repeated eigenvalue case.

One of our motivators for the QR factorization in Chapter 5 was that the matrix Q is *orthogonal*, satisfying $Q^{-1} = Q^\top$. Because of this formula, Q and Q^{-1} are equally straightforward to apply, making orthogonal matrices strong choices for similarity transformations. We already applied this observation in §6.4.1 when we deflated using Householder matrices. Conjugating by orthogonal matrices also does not affect the conditioning of the eigenvalue problem.

But if we do not know *any* eigenvectors of A, which orthogonal matrix Q should we choose? Ideally, Q should involve the structure of A while being straightforward to compute. It is less clear how to apply Householder matrices strategically to reveal multiple eigenvalues in parallel (some advanced techniques do exactly this!), but we do know how to generate one orthogonal Q from A by factoring $A = QR$. Then, experimentally we might conjugate A by Q to find:

$$Q^{-1}AQ = Q^\top AQ = Q^\top(QR)Q = (Q^\top Q)RQ = RQ.$$

Amazingly, conjugating $A = QR$ by the orthogonal matrix Q is identical to writing the product RQ!

This matrix $A_2 \equiv RQ$ is *not* equal to $A = QR$, but it has the same eigenvalues. Hence, we can factor $A_2 = Q_2R_2$ to get a new orthogonal matrix Q_2 and once again conjugate to define $A_3 \equiv R_2Q_2$. Repeating this process indefinitely generates a sequence of similar matrices $A, A_2, A_3, \ldots$ with the same eigenvalues. Curiously, for many choices of A, as $k \to \infty$, one can check numerically that while iterating QR factorization in this manner, R_k becomes an upper-triangular matrix containing the eigenvalues of A along its diagonal.

Based on this elegant observation, in the 1950s multiple groups of European mathematicians studied the same iterative algorithm for finding the eigenvalues of a matrix A, shown in Figure 6.7:

Repeatedly factorize $A = QR$ and replace A with RQ.

Take A_k to be A after the k-th iteration of this method; that is $A_1 = A = Q_1R_1$, $A_2 = R_1Q_1 = Q_2R_2$, $A_3 = R_2Q_2 = Q_3R_3$, and so on. Since they are related via conjugation by a sequence of Q matrices, the matrices A_k all have the same eigenvalues as A. So, our analysis must show (1) when we expect this technique to converge and (2) if and how the limit point reveals eigenvalues of A. We will answer these questions in reverse order, for the case when A is symmetric and invertible with no repeated eigenvalues up to sign; so, if $\lambda \neq 0$ is an eigenvalue of A, then $-\lambda$ is *not* an eigenvalue of A. More advanced analysis and application to asymmetric or defective matrices can be found in [50] and elsewhere.

We begin by proving a proposition that will help characterize limit behavior of the QR iteration algorithm:*

*The conditions of this proposition can be relaxed but are sufficient for the discussion at hand.

Proposition 6.6. Take $A, B \in \mathbb{R}^{n\times n}$. Suppose that the eigenvectors of A span $\mathbb{R}^n$ and have distinct eigenvalues. Then, $AB = BA$ if and only if A and B have the same set of eigenvectors (with possibly different eigenvalues).

Proof. Suppose A and B have eigenvectors $\vec{x}_1, \ldots, \vec{x}_n$ with eigenvalues $\lambda_1^A, \ldots, \lambda_n^A$ for A and eigenvalues $\lambda_1^B, \ldots, \lambda_n^B$ for B. Any $\vec{y} \in \mathbb{R}^n$ can be decomposed as $\vec{y} = \sum_i a_i \vec{x}_i$, showing:

$$BA\vec{y} = BA\sum_i a_i\vec{x}_i = B\sum_i \lambda_i^A\vec{x}_i = \sum_i \lambda_i^A\lambda_i^B\vec{x}_i$$
$$AB\vec{y} = AB\sum_i a_i\vec{x}_i = A\sum_i \lambda_i^B\vec{x}_i = \sum_i \lambda_i^A\lambda_i^B\vec{x}_i.$$

So, $AB\vec{y} = BA\vec{y}$ for all $\vec{y} \in \mathbb{R}^n$, or equivalently $AB = BA$.

Now, suppose $AB = BA$, and take $\vec{x}$ to be any eigenvector of A with $A\vec{x} = \lambda\vec{x}$. Then, $A(B\vec{x}) = (AB)\vec{x} = (BA)\vec{x} = B(A\vec{x}) = \lambda(B\vec{x})$. We have two cases:

- If $B\vec{x} \neq \vec{0}$, then $B\vec{x}$ is an eigenvector of A with eigenvalue λ. Since A has no repeated eigenvalues and $\vec{x}$ is also an eigenvector of A with eigenvalue λ, we must have $B\vec{x} = c\vec{x}$ for some $c \neq 0$. In other words, $\vec{x}$ is also an eigenvector of B with eigenvalue c.
- If $B\vec{x} = \vec{0}$, then $\vec{x}$ is an eigenvector of B with eigenvalue 0.

Hence, all of the eigenvectors of A are eigenvectors of B. Since the eigenvectors of A span $\mathbb{R}^n$, A and B have exactly the same set of eigenvectors. □

Returning to QR iteration, suppose $A_k \to A_\infty$ as $k \to \infty$. If we factor $A_\infty = Q_\infty R_\infty$, then since QR iteration converged, $A_\infty = Q_\infty R_\infty = R_\infty Q_\infty$. By the conjugation property, $Q_\infty^\top A_\infty Q_\infty = R_\infty Q_\infty = A_\infty$, or equivalently $A_\infty Q_\infty = Q_\infty A_\infty$. Since A_∞ has a full set of distinct eigenvalues, by Proposition 6.6, Q_∞ has the same eigenvectors as A_∞. The eigenvalues of Q_∞ are ± 1 by orthogonality. Suppose $A_\infty\vec{x} = \lambda\vec{x}$. In this case,

$$\lambda\vec{x} = A_\infty\vec{x} = Q_\infty R_\infty\vec{x} = R_\infty Q_\infty\vec{x} = \pm R_\infty\vec{x},$$

so $R_\infty\vec{x} = \pm\lambda\vec{x}$. Since R_∞ is upper triangular, we now know (Exercise 6.3):

The eigenvalues of A_∞—and hence the eigenvalues of A—equal the diagonal elements of R_∞ up to sign.

We can remove the sign caveat by computing QR using rotations rather than reflections.

The derivation above assumes that there exists A_∞ with $A_k \to A_\infty$ as $k \to \infty$. Although we have not shown it yet, QR iteration is a stable method guaranteed to converge in many situations, and even when it does not converge, the relevant eigenstructure of A often can be computed from R_k as $k \to \infty$ regardless. We will not derive exact convergence conditions here but will provide some intuition for why we might expect this method to converge, at least given our restrictions on A.

To help motivate when we expect QR iteration to converge and yield eigenvalues along the diagonal of R_∞, suppose the columns of A are given by $\vec{a}_1, \ldots, \vec{a}_n$, and consider the matrix A^k for large k. We can write:

$$A^k = A^{k-1}\cdot A = \begin{pmatrix} | & | & & | \\ A^{k-1}\vec{a}_1 & A^{k-1}\vec{a}_2 & \cdots & A^{k-1}\vec{a}_n \\ | & | & & | \end{pmatrix}.$$

By our derivation of power iteration, in the absence of degeneracies, the first column of A^k will become more and more parallel to the eigenvector $\vec{x}_1$ of A with largest magnitude $|\lambda_1|$ as $k \to \infty$, since we took a vector $\vec{a}_1$ and multiplied it by A many times.

Applying intuition from deflation, suppose we project $\vec{x}_1$, which is approximately parallel to the first column of A^k, out of the second column of A^k. By orthogonality of the eigenvectors of A, we equivalently could have projected $\vec{x}_1$ out of $\vec{a}_2$ *initially* and then applied A^{k-1}. For this reason, as in §6.4.1, thanks to the removal of $\vec{x}_1$ the result of either process must be nearly parallel to $\vec{x}_2$, the vector with the *second*-most dominant eigenvalue! Proceeding inductively, when A is symmetric and thus has a full set of orthogonal eigenvectors, factoring $A^k = QR$ yields a set of near-eigenvectors of A in the columns of Q, in order of decreasing eigenvalue magnitude, with the corresponding eigenvalues along the diagonal of R.

Multiplying to find A^k for large k approximately takes the condition number of A to the k-th power, so computing the QR decomposition of A^k explicitly is likely to lead to numerical problems. Since decomposing A^k would reveal the eigenvector structure of A, however, we use this fact to our advantage without paying numerically. To do so, we make the following observation about QR iteration:

$$\begin{aligned}
A &= Q_1 R_1 \text{ by definition of QR iteration} \\
A^2 &= (Q_1 R_1)(Q_1 R_1) \\
&= Q_1 (R_1 Q_1) R_1 \text{ by regrouping} \\
&= Q_1 Q_2 R_2 R_1 \text{ since } A_2 = R_1 Q_1 = Q_2 R_2 \\
&\vdots \\
A^k &= Q_1 Q_2 \cdots Q_k R_k R_{k-1} \cdots R_1 \text{ by induction.}
\end{aligned}$$

Grouping the Q_i variables and the R_i variables separately provides a QR factorization of A^k. In other words, we can use the Q_k's and R_k's constructed during each step of QR iteration to construct a factorization of A^k, and thus we expect the columns of the product $Q_1 \cdots Q_k$ to converge to the eigenvectors of A.

By a similar argument, we show a related fact about the iterates $A_1, A_2, \ldots$ from QR iteration. Since $A_k = Q_k R_k$, we substitute $R_k = Q_k^\top A_k$ inductively to show:

$$\begin{aligned}
A_1 &= A \\
A_2 &= R_1 Q_1 \text{ by our construction of QR iteration} \\
&= Q_1^\top A Q_1 \text{ since } R_1 = Q_1^\top A_1 \\
A_3 &= R_2 Q_2 \\
&= Q_2^\top A_2 Q_2 \\
&= Q_2^\top Q_1^\top A Q_1 Q_2 \text{ from the previous step} \\
&\vdots \\
A_{k+1} &= Q_k^\top \cdots Q_1^\top A Q_1 \cdots Q_k \text{ inductively} \\
&= (Q_1 \cdots Q_k)^\top A (Q_1 \cdots Q_k),
\end{aligned}$$

where A_k is the k-th matrix from QR iteration. Thus, A_{k+1} is the matrix A conjugated by the product $\bar{Q}_k \equiv Q_1 \cdots Q_k$. We argued earlier that the columns of $\bar{Q}_k$ converge to the eigenvectors of A. Since conjugating by the matrix of eigenvectors yields a diagonal matrix of eigenvalues, we know $A_{k+1} = \bar{Q}_k^\top A \bar{Q}$ will have approximate eigenvalues of A along its diagonal as $k \to \infty$, at least when eigenvalues are not repeated.

In the case of symmetric matrices without repeated eigenvalues, we have shown that both A_k and R_k will converge unconditionally to diagonal matrices containing the eigenvalues of

A, while the product of the Q_k's will converge to a matrix of the corresponding eigenvectors. This case is but one example of the power of QR iteration, which is applied to many problems in which more than a few eigenvectors are needed of a given matrix A.

In practice, a few simplifying steps are usually applied before commencing QR iteration. QR factorization of a full matrix is relatively expensive computationally, so each iteration of the algorithm as we have described it is costly for large matrices. One way to avoid this cost for symmetric A is first to *tridiagonalize* A, systematically conjugating it by orthogonal matrices until entries not on or immediately adjacent to the diagonal are zero; tridiagonalization can be carried out using Householder matrices in $O(n^3)$ time for $A \in \mathbb{R}^{n \times n}$ [22]. QR factorization of symmetric tridiagonal matrices is much more efficient than the general case [92].

Example 6.3 (QR iteration)**.** To illustrate typical behavior of QR iteration, we apply the algorithm to the matrix

$$A = \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix}.$$

The first few iterations, computed numerically, are shown below:

$$A_1 = \begin{pmatrix} 2.000 & 3.000 \\ 3.000 & 2.000 \end{pmatrix} = \underbrace{\begin{pmatrix} -0.555 & 0.832 \\ -0.832 & -0.555 \end{pmatrix}}_{Q_1} \underbrace{\begin{pmatrix} -3.606 & -3.328 \\ 0.000 & 1.387 \end{pmatrix}}_{R_1}$$

$$\implies A_2 = R_1 Q_1 = \begin{pmatrix} 4.769 & -1.154 \\ -1.154 & -0.769 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 4.769 & -1.154 \\ -1.154 & -0.769 \end{pmatrix} = \underbrace{\begin{pmatrix} -0.972 & -0.235 \\ 0.235 & -0.972 \end{pmatrix}}_{Q_2} \underbrace{\begin{pmatrix} -4.907 & 0.941 \\ 0.000 & 1.019 \end{pmatrix}}_{R_2}$$

$$\implies A_3 = R_2 Q_2 = \begin{pmatrix} 4.990 & 0.240 \\ 0.240 & -0.990 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 4.990 & 0.240 \\ 0.240 & -0.990 \end{pmatrix} = \underbrace{\begin{pmatrix} -0.999 & 0.048 \\ -0.048 & -0.999 \end{pmatrix}}_{Q_3} \underbrace{\begin{pmatrix} -4.996 & -0.192 \\ 0.000 & 1.001 \end{pmatrix}}_{R_3}$$

$$\implies A_4 = R_3 Q_3 = \begin{pmatrix} 5.000 & -0.048 \\ -0.048 & -1.000 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} 5.000 & -0.048 \\ -0.048 & -1.000 \end{pmatrix} = \underbrace{\begin{pmatrix} -1.000 & -0.010 \\ 0.010 & -1.000 \end{pmatrix}}_{Q_4} \underbrace{\begin{pmatrix} -5.000 & 0.038 \\ 0.000 & 1.000 \end{pmatrix}}_{R_4}$$

$$\implies A_5 = R_4 Q_4 = \begin{pmatrix} 5.000 & 0.010 \\ 0.010 & -1.000 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} 5.000 & 0.010 \\ 0.010 & -1.000 \end{pmatrix} = \underbrace{\begin{pmatrix} -1.000 & 0.002 \\ -0.002 & -1.000 \end{pmatrix}}_{Q_5} \underbrace{\begin{pmatrix} -5.000 & -0.008 \\ 0.000 & 1.000 \end{pmatrix}}_{R_5}$$

$$\implies A_6 = R_5 Q_5 = \begin{pmatrix} 5.000 & -0.002 \\ -0.002 & -1.000 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} 5.000 & -0.002 \\ -0.002 & -1.000 \end{pmatrix} = \underbrace{\begin{pmatrix} -1.000 & -0.000 \\ 0.000 & -1.000 \end{pmatrix}}_{Q_6} \underbrace{\begin{pmatrix} -5.000 & 0.002 \\ 0.000 & 1.000 \end{pmatrix}}_{R_6}$$

$$\implies A_7 = R_6 Q_6 = \begin{pmatrix} 5.000 & 0.000 \\ 0.000 & -1.000 \end{pmatrix}$$

The diagonal elements of A_k converge to the eigenvalues 5 and -1 of A, as expected.

6.4.3 Krylov Subspace Methods

Our justification for QR iteration involved analyzing the columns of A^k as $k \to \infty$, applying observations we already made about power iteration in §6.3.1. More generally, for a vector $\vec{b} \in \mathbb{R}^n$, we can examine the so-called *Krylov matrix*

$$K_k \equiv \begin{pmatrix} | & | & | & & | \\ \vec{b} & A\vec{b} & A^2\vec{b} & \cdots & A^{k-1}\vec{b} \\ | & | & | & & | \end{pmatrix}.$$

Methods analyzing K_k to find eigenvectors and eigenvalues generally are classified as *Krylov subspace methods*. For instance, the *Arnoldi iteration* algorithm uses Gram-Schmidt orthogonalization to maintain an orthogonal basis $\{\vec{q}_1, \ldots, \vec{q}_k\}$ for the column space of K_k:

1. Begin by taking $\vec{q}_1$ to be an arbitrary unit-norm vector.
2. For $k = 2, 3, \ldots$
   (a) Take $\vec{a}_k = A\vec{q}_{k-1}$.
   (b) Project out the $\vec{q}$'s you already have computed:
   $$\vec{b}_k = \vec{a}_k - \text{proj}_{\text{span}\,\{\vec{q}_1,\ldots,\vec{q}_{k-1}\}}\vec{a}_k.$$
   (c) Renormalize to find the next $\vec{q}_k = \vec{b}_k / \|\vec{b}_k\|_2$.

The matrix Q_k whose columns are the vectors found above is an orthogonal matrix with the same column space as K_k, and eigenvalue estimates can be recovered from the structure of $Q_k^\top A Q_k$.

The use of Gram-Schmidt makes this technique unstable, and its timing gets progressively worse as k increases. So, extensions are needed to make it feasible. For instance, one approach involves running some iterations of the Arnoldi algorithm, using the output to generate a better guess for the initial $\vec{q}_1$, and restarting [80]. Methods in this class are suited for problems requiring multiple eigenvectors at the ends of the spectrum of A without computing the complete set. They also can be applied to designing iterative methods for solving linear systems of equations, as we will explore in Chapter 11.

6.5 SENSITIVITY AND CONDITIONING

We have only outlined a few eigenvalue techniques out of a rich and long-standing literature. Almost any algorithmic technique has been experimented with for finding spectra, from iterative methods to root-finding on the characteristic polynomial to methods that divide matrices into blocks for parallel processing.

As with linear solvers, we can evaluate the conditioning of an eigenvalue problem independently of the solution technique. This analysis can help understand whether a simplistic iterative algorithm will be successful at finding the eigenvectors of a given matrix or if more complex stabilized methods are necessary. To do so, we will derive a condition number for the problem of finding eigenvalues for a given matrix A. Before proceeding, we should highlight that the conditioning of an eigenvalue problem is *not* the same as the condition number of the matrix for solving linear systems.

Suppose a matrix A has an eigenvector $\vec{x}$ with eigenvalue λ. Analyzing the conditioning of the eigenvalue problem involves analyzing the stability of $\vec{x}$ and λ to perturbations in A. To this end, we might perturb A by a small matrix δA, thus changing the set of eigenvectors.

We can write eigenvectors of $A + \delta A$ as perturbations of eigenvectors of A by solving the problem

$$(A + \delta A)(\vec{x} + \delta\vec{x}) = (\lambda + \delta\lambda)(\vec{x} + \delta\vec{x}).$$

Expanding both sides yields:

$$A\vec{x} + A\delta\vec{x} + \delta A \cdot \vec{x} + \delta A \cdot \delta\vec{x} = \lambda\vec{x} + \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x} + \delta\lambda \cdot \delta\vec{x}.$$

Since δA is small, we will assume that $\delta\vec{x}$ and $\delta\lambda$ also are small (this assumption should be checked in a more rigorous treatment!). Products between these variables then are negligible, yielding the following approximation:

$$A\vec{x} + A\delta\vec{x} + \delta A \cdot \vec{x} \approx \lambda\vec{x} + \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x}.$$

Since $A\vec{x} = \lambda\vec{x}$, we can subtract this vector from both sides to find:

$$A\delta\vec{x} + \delta A \cdot \vec{x} \approx \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x}.$$

We now apply an analytical trick to complete our derivation. Since $A\vec{x} = \lambda\vec{x}$, we know $(A - \lambda I_{n\times n})\vec{x} = \vec{0}$, so $A - \lambda I_{n\times n}$ is not full rank. The transpose of a matrix is full-rank only if the matrix is full-rank, so we know $(A - \lambda I_{n\times n})^\top = A^\top - \lambda I_{n\times n}$ also has a null space vector $\vec{y}$, with $A^\top\vec{y} = \lambda\vec{y}$. We call $\vec{y}$ a *left* eigenvector corresponding to $\vec{x}$. Left-multiplying our perturbation estimate above by $\vec{y}^\top$ shows

$$\vec{y}^\top(A\delta\vec{x} + \delta A \cdot \vec{x}) \approx \vec{y}^\top(\lambda\delta\vec{x} + \delta\lambda \cdot \vec{x}).$$

Since $A^\top\vec{y} = \lambda\vec{y}$, we can simplify:

$$\vec{y}^\top\delta A \cdot \vec{x} \approx \delta\lambda\vec{y}^\top\vec{x}.$$

Rearranging yields:

$$\delta\lambda \approx \frac{\vec{y}^\top(\delta A)\vec{x}}{\vec{y}^\top\vec{x}}.$$

Finally, assume $\|\vec{x}\|_2 = 1$ and $\|\vec{y}\|_2 = 1$. Then, taking norms on both sides shows:

$$|\delta\lambda| \lessapprox \frac{\|\delta A\|_2}{|\vec{y} \cdot \vec{x}|}.$$

This expression shows that conditioning of the eigenvalue problem roughly depends directly on the size of the perturbation δA and inversely on the angle between the left and right eigenvectors $\vec{x}$ and $\vec{y}$.

Based on this derivation, we can use $1/|\vec{x}\cdot\vec{y}|$ as an approximate condition number for finding the eigenvalue λ corresponding to eigenvector $\vec{x}$ of A. Symmetric matrices have the same left and right eigenvectors, so $\vec{x} = \vec{y}$, yielding a condition number of 1. This strong conditioning reflects the fact that the eigenvectors of symmetric matrices are orthogonal and thus maximally separated.

6.6 EXERCISES

6.1 Verify the solution $\vec{y}(t)$ given in §6.1.2 to the ODE $\vec{y}' = B\vec{y}$.

6.2 Define

$$A \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Can power iteration find eigenvalues of this matrix? Why or why not?

6.3 Show that the eigenvalues of upper-triangular matrices $U \in \mathbb{R}^{n \times n}$ are exactly their diagonal elements.

6.4 Extending Exercise 6.3, if we assume that the eigenvectors of U are $\vec{v}_k$ satisfying $U\vec{v}_k = u_{kk}\vec{v}_k$, characterize $\text{span}\{\vec{v}_1, \ldots, \vec{v}_k\}$ for $1 \leq k \leq n$ when the diagonal values u_{kk} of U are distinct.

6.5 We showed that the Rayleigh quotient iteration method can converge more quickly than power iteration. Why, however, might it still be more efficient to use the power method in some cases?

6.6 (Suggested by J. Yeo) Suppose $\vec{u}$ and $\vec{v}$ are vectors in $\mathbb{R}^n$ such that $\vec{u}^\top \vec{v} = 1$, and define $A \equiv \vec{u}\vec{v}^\top$.

(a) What are the eigenvalues of A?

(b) How many iterations does power iteration take to converge to the dominant eigenvalue of A?

6.7 (Suggested by J. Yeo) Suppose $B \in \mathbb{R}^{n \times n}$ is diagonalizable with eigenvalues λ_i satisfying $0 < \lambda_1 = \lambda_2 < \lambda_3 < \cdots < \lambda_n$. Let $\vec{v}_i$ be the eigenvector corresponding to λ_i. Show that the inverse power method applied to B converges to a linear combination of $\vec{v}_1$ and $\vec{v}_2$.

6.8 ("Mini-Riesz Representation Theorem") We will say $\langle \cdot, \cdot \rangle$ is an *inner product* on $\mathbb{R}^n$ if it satisfies:

1. $\langle \vec{x}, \vec{y} \rangle = \langle \vec{y}, \vec{x} \rangle \; \forall \vec{x}, \vec{y} \in \mathbb{R}^n$,
2. $\langle \alpha\vec{x}, \vec{y} \rangle = \alpha \langle \vec{x}, \vec{y} \rangle \; \forall \vec{x}, \vec{y} \in \mathbb{R}^n, \alpha \in \mathbb{R}$,
3. $\langle \vec{x} + \vec{y}, \vec{z} \rangle = \langle \vec{x}, \vec{z} \rangle + \langle \vec{y}, \vec{z} \rangle \; \forall \vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$, and
4. $\langle \vec{x}, \vec{x} \rangle \geq 0$ with equality if and only if $\vec{x} = \vec{0}$.

(a) Given an inner product $\langle \cdot, \cdot \rangle$, show that there exists a matrix $A \in \mathbb{R}^{n \times n}$ (depending on $\langle \cdot, \cdot \rangle$) such that $\langle \vec{x}, \vec{y} \rangle = \vec{x}^\top A\vec{y}$ for all $\vec{x}, \vec{y} \in \mathbb{R}^n$. Also, show that there exists a matrix $M \in \mathbb{R}^{n \times n}$ such that $\langle \vec{x}, \vec{y} \rangle = (M\vec{x}) \cdot (M\vec{y})$ for all $\vec{x}, \vec{y} \in \mathbb{R}^n$. [This shows that all inner products are dot products after suitable rotation, stretching, and shearing of $\mathbb{R}^n$!]

(b) A *Mahalanobis metric* on $\mathbb{R}^n$ is a distance function of the form $d(\vec{x}, \vec{y}) = \sqrt{\langle \vec{x} - \vec{y}, \vec{x} - \vec{y} \rangle}$ for a fixed inner product $\langle \cdot, \cdot \rangle$. Use Exercise 6.8a to write Mahalanobis metrics in terms of matrices M, and show that $d(\vec{x}, \vec{y})^2$ is a quadratic function in $\vec{x}$ and $\vec{y}$ jointly.

(c) Suppose we are given several pairs $(\vec{x}_i, \vec{y}_i) \in \mathbb{R}^n \times \mathbb{R}^n$. A typical "metric learning" problem involves finding a nontrivial Mahalanobis metric such that each $\vec{x}_i$ is close to each $\vec{y}_i$ with respect to that metric. Propose an optimization problem for this task that can be solved using eigenvector computation.
Note: Make sure that your optimal Mahalanobis distance is not identically zero, but it is acceptable if your optimization allows *pseudometrics*; that is, there can exist some $\vec{x} \neq \vec{y}$ with $d(\vec{x}, \vec{y}) = 0$.

6.9 ("Shifted QR iteration") A widely used generalization of the QR iteration algorithm for finding eigenvectors and eigenvalues of $A \in \mathbb{R}^{n\times n}$ uses a shift in each iteration:

$$\begin{aligned} A_0 &= A \\ A_k - \sigma_k I_{n\times n} &= Q_k R_k \\ A_{k+1} &= R_k Q_k + \sigma_k I_{n\times n}. \end{aligned}$$

Uniformly choosing $\sigma_k \equiv 0$ recovers basic QR iteration. Different variants of this method propose heuristics for choosing $\sigma_k \neq 0$ to encourage convergence or numerical stability.

(a) Show that A_k is similar to A for all $k \geq 0$.

(b) Propose a heuristic for choosing σ_k based on the construction of Rayleigh quotient iteration. Explain when you expect your method to converge faster than basic QR iteration.

6.10 Suppose $A, B \in \mathbb{R}^{n\times n}$ are symmetric and positive definite.

(a) Define a matrix $\sqrt{A} \in \mathbb{R}^{n\times n}$ and show that $(\sqrt{A})^2 = A$. Generally speaking, $\sqrt{A}$ is not the same as L in the Cholesky factorization $A = LL^\top$.

(b) Do most matrices have unique square roots? Why or why not?

(c) We can define the *exponential* of A as $e^A \equiv \sum_{k=0}^{\infty} \frac{1}{k!} A^k$; this sum is unconditionally convergent (you do not have to prove this!). Write an alternative expression for e^A in terms of the eigenvectors and eigenvalues of A.

(d) If $AB = BA$, show $e^{A+B} = e^A e^B$.

(e) Show that the ordinary differential equation $\vec{y}'(t) = -A\vec{y}$ with $\vec{y}(0) = \vec{y}_0$ for some $\vec{y}_0 \in \mathbb{R}^n$ is solved by $\vec{y}(t) = e^{-At}\vec{y}_0$. What happens as $t \to \infty$?

6.11 ("Epidemiology") Suppose $\vec{x}_0 \in \mathbb{R}^n$ contains sizes of different populations carrying a particular infection in year 0; for example, when tracking malaria we might take x_{01} to be the number of humans with malaria and x_{02} to be the number of mosquitoes carrying the disease. By writing relationships like "The average mosquito infects two humans," we can write a matrix M such that $\vec{x}_1 \equiv M\vec{x}_0$ predicts populations in year 1, $\vec{x}_2 \equiv M^2\vec{x}_0$ predicts populations in year 2, and so on.

(a) The spectral radius $\rho(M)$ is given by $\max_i |\lambda_i|$, where the eigenvalues of M are $\lambda_1, \ldots, \lambda_k$. Epidemiologists call this number the "reproduction number" $\mathcal{R}_0$ of M. Explain the difference between the cases $\mathcal{R}_0 < 1$ and $\mathcal{R}_0 > 1$ in terms of the spread of disease. Which case is more dangerous?

(b) Suppose we only care about proportions. For instance, we might use $M \in \mathbb{R}^{50\times 50}$ to model transmission of diseases between residents in each of the 50 states of the USA, and we only care about the fraction of the total people with a disease who live in each state. If $\vec{y}_0$ holds these proportions in year 0, give an iterative scheme to predict proportions in future years. Characterize behavior as time goes to infinity.

Note: Those readers concerned about computer graphics applications of this material should know that the reproduction number $\mathcal{R}_0$ is referenced in the 2011 thriller *Contagion*.

6.12 ("Normalized cuts," [110]) Similar to spectral embedding (§6.1.3), suppose we have a collection of n objects and a symmetric matrix $W \in (\mathbb{R}^+)^{n\times n}$ whose entries w_{ij} measure the similarity between object i and object j. Rather than computing an embedding, however, now we would like to *cluster* the objects into two groups. This machinery is used to mark photos as day or night and to classify pixels in an image as foreground or background.

(a) Suppose we cluster $\{1,\ldots,n\}$ into two disjoint sets A and B; this clustering defines a *cut* of the collection. We define the *cut score* of (A,B) as follows:

$$C(A,B) \equiv \sum_{\substack{i\in A\\ j\in B}} w_{ij}.$$

This score is large if objects in A and B are similar. Efficiency aside, why is it inadvisable to minimize $C(A,B)$ with respect to A and B?

(b) Define the *volume* of a set A as $V(A) \equiv \sum_{i\in A}\sum_{j=1}^{n} w_{ij}$. To alleviate issues with minimizing the cut score directly, instead we will attempt minimize the *normalized cut* score $N(A,B) \equiv C(A,B)(V(A)^{-1} + V(B)^{-1})$. What does this score measure?

(c) For a fixed choice of A and B, define $\vec{x} \in \mathbb{R}^n$ as follows:

$$x_i \equiv \begin{cases} V(A)^{-1} & \text{if } i \in A \\ -V(B)^{-1} & \text{if } i \in B. \end{cases}$$

Explain how to construct matrices L and D from W such that

$$\vec{x}^\top L\vec{x} = \sum_{\substack{i\in A\\ j\in B}} w_{ij}\left(V(A)^{-1} + V(B)^{-1}\right)^2$$
$$\vec{x}^\top D\vec{x} = V(A)^{-1} + V(B)^{-1}.$$

Conclude that $N(A,B) = \frac{\vec{x}^\top L\vec{x}}{\vec{x}^\top D\vec{x}}$.

(d) Show that $\vec{x}^\top D\vec{1} = 0$.

(e) The *normalized cuts algorithm* computes A and B by optimizing for $\vec{x}$. Argue that the result of the following optimization lower-bounds the minimum normalized cut score of any partition (A,B) :

$$\begin{array}{rl} \min_{\vec{x}} & \frac{\vec{x}^\top L\vec{x}}{\vec{x}^\top D\vec{x}} \\ \text{subject to} & \vec{x}^\top D\vec{1} = 0. \end{array}$$

Assuming D is invertible, show that this relaxed $\vec{x}$ can be computed using an eigenvalue problem.