CS168: The Modern Algorithmic Toolbox
Lecture #8: How PCA Works

Tim Roughgarden & Gregory Valiant*

April 21, 2021

1 Introduction

Last lecture introduced the idea of principal components analysis (PCA). The definition of the method is, for a given data set and parameter k, to compute the k-dimensional subspace (through the origin) that minimizes the average squared distance between the points and the subspace, or equivalently that maximizes the variance of the projections of the data points onto the subspace. We talked about a couple of common use cases. The first is data visualization, where one uses the top few principal components as a new coordinate axis and plots the projections of all of the data points in this new coordinate system. We saw a remarkable example using European genomic data, where such a plot showed that geographic information is embedded in the participants' DNA. The second use case discussed was data compression, with the "Eigenfaces" project being a canonical example, where taking k in the range of 100 or 150 is large enough to closely approximate a bunch of images (65,000-dimensional data). Finally, we mentioned some of the primary weaknesses of PCA: it can get tricked by high-variance noise, it fails to discover nonlinear structure, and the orthogonality constraints on the principal components mean that principal components after the first few can be difficult to interpret.

Today's lecture is about how PCA actually works — that is, how to actually compute the top k principal components of a data set. Along the way, we'll develop your internal mapping between the linear algebra used to describe the method and the simple geometry that explains what's really going on. Ideally, after understanding this lecture, PCA should seem almost obvious in hindsight.

2 Characterizing Principal Components

2.1 The Setup

Recall that the input to the PCA method is n d-dimensional data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and a parameter $k \in \{1, 2, \ldots, d\}$. We assume that the data is centered, meaning that $\sum_{i=1}^n \mathbf{x}_i$ is the all-zero vector. (This can be enforced by subtracting out the sample mean $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ in a preprocessing step.) The output of the method is defined as k orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ — the "top k principal components" — that maximize the objective function

$$\frac{1}{n}\sum_{i=1}^{n} \underbrace{\sum_{j=1}^{k} \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2}_{\text{squared projection length}}. \tag{1}$$

But how would one actually solve this optimization problem and compute the $\mathbf{v}_j$'s? Even with $k = 1$, there are an infinite number of unit vectors to try. A big reason for the popularity of PCA is that this optimization problem is easily solved using sophomore-level linear algebra. After we review the necessary preliminaries and build up your geometric intuition, the solution should seem straightforward in hindsight.

2.2 Rewriting the Optimization Problem

To develop the solution, we first consider only the $k = 1$ case. We'll see that the case of general k reduces to this case (Section 2.6). With $k = 1$, the objective function is

$$\operatorname*{argmax}_{\mathbf{v}\,:\,\|\mathbf{v}\|=1} \frac{1}{n}\sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2. \tag{2}$$

Let's see how to rewrite variance-maximization (2) using linear algebra. First, we take the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ — remember these are in d-dimensional space — and write them as the rows of an $n \times d$ matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_n & — \end{bmatrix}.$$

Thus, for a unit vector $\mathbf{v} \in \mathbb{R}^d$, we have

$$\mathbf{X}\mathbf{v} = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{v} \rangle \\ \langle \mathbf{x}_2, \mathbf{v} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{v} \rangle \end{bmatrix},$$

so $\mathbf{Xv}$ is just a column vector populated with all the projection lengths of the $\mathbf{x}_i$'s onto the line spanned by $\mathbf{v}$. We care about the sum of the squares of these (recall (2)), which motivates taking the inner product of $\mathbf{Xv}$ with itself:

$$\mathbf{v}^\top \mathbf{X}^\top \mathbf{X}\mathbf{v} = (\mathbf{Xv})^\top (\mathbf{Xv}) = \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2.$$

Summarizing, our variance-maximization problem can be rephrased as

$$\operatorname*{argmax}_{\mathbf{v}\,:\,\|\mathbf{v}\|=1} \mathbf{v}^\top \mathbf{A}\mathbf{v}, \tag{3}$$

where $\mathbf{A}$ is a $d \times d$ matrix of the form $\mathbf{X}^\top \mathbf{X}$.[1] This problem is called "maximizing a quadratic form."

The matrix $\mathbf{X}^\top \mathbf{X}$ has a natural interpretation.[2] The (i, j) entry of this matrix is the inner product of the ith row of $\mathbf{X}^\top$ and the jth column of $\mathbf{X}$ — i.e., of the ith and jth columns of $\mathbf{X}$. So $\mathbf{X}^\top \mathbf{X}$ just collects the inner products of columns of $\mathbf{X}$, and is a symmetric matrix. For example, suppose the $\mathbf{x}_i$'s represent documents, with dimensions (i.e., columns of $\mathbf{X}$) corresponding to words. Then the inner product of two columns of $\mathbf{X}$ measures how frequently the corresponding pair of words co-occur in a document.[3] The matrix $\mathbf{X}^\top \mathbf{X}$ is called the *covariance* or *correlation matrix* of the $\mathbf{x}_i$'s, depending on whether or not each of the coordinates was normalized so as to have unit variance in a preprocessing step (as discussed in Lecture #7).

2.3 Solving (3): The Diagonal Case

To gain some understanding for the optimization problem (3) that PCA solves, let's begin with a very special case: where $\mathbf{A}$ is a diagonal matrix

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & & \lambda_d \end{pmatrix} \tag{4}$$

with sorted nonnegative entries $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ on the diagonal.

Any $d \times d$, $\mathbf{A}$, matrix can be thought of as a function that maps points in $\mathbb{R}^d$ back to points in $\mathbb{R}^d$, defined by the function that maps a d-dimensional vector v to the vector $\mathbf{A}v$.

[1] We are ignoring the $\frac{1}{n}$ scaling factor in (2), because the optimal solution $\mathbf{v}$ is the same with or without it.

[2] You might recall that this matrix also appeared in the closed-form solution to the ordinary least squares problem.

[3] So after centering such an $\mathbf{X}$, frequently co-occurring pairs of words correspond to positive entries of $\mathbf{X}^\top \mathbf{X}$ and pairs of words that almost never appear together are negative entries.

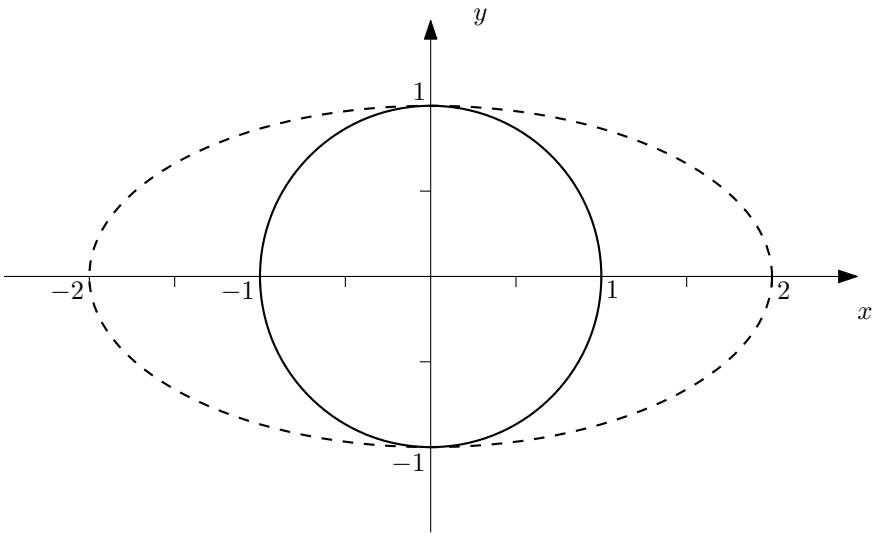

Figure 1: The point (x, y) on the unit circle is mapped to $(2x, y)$.

In that sense, any matrix can be thought of as this map that "moves $\mathbb{R}^d$ around," in effect. There is a simple geometric way to think about diagonal matrices—for example, the matrix

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

moves each point (x, y) of the plane to the point $(2x, y)$ with double the x-coordinate and the same y-coordinate. Similarly, the points of the circle $\{(x, y) : x^2 + y^2 = 1\}$ are mapped to the points $\{\frac{x^2}{2^2} + y^2 = 1\}$ of the ellipse shown in Figure 1. More generally, a diagonal matrix of the form (4) can be thought of as "stretching" $\mathbb{R}^d$, with the ith axis getting stretched by the factor λ_i, and the unit circle being mapped to the corresponding "ellipsoid" (i.e., the analog of an ellipse in more than 2 dimensions).

A natural guess for the direction $\mathbf{v}$ that maximizes $\mathbf{v}^\top \mathbf{A}\mathbf{v}$ with $\mathbf{A}$ diagonal is the "direction of maximum stretch," namely $\mathbf{v} = \mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, \ldots, 0)$ denotes the first standard basis vector. (Recall $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d$.) To verify the guess, let $\mathbf{v}$ be an arbitrary unit vector, and write

$$\mathbf{v}^\top(\mathbf{A}\mathbf{v}) = \begin{pmatrix} v_1 & v_2 & \cdots & v_d \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 v_1 \\ \lambda_2 v_2 \\ \vdots \\ \lambda_d v_d \end{pmatrix} = \sum_{i=1}^{d} v_i^2 \lambda_i. \tag{5}$$

Since $\mathbf{v}$ is a unit vector, the v_i^2's sum to 1. Thus $\mathbf{v}^\top \mathbf{A}\mathbf{v}$ is always an average of the λ_i's, with the averaging weights given by the v_i^2's. Since λ_1 is the biggest λ_i, the way to make this average as large as possible is to set $v_1 = 1$ and $v_i = 0$ for $i > 1$. That is, $\mathbf{v} = \mathbf{e}_1$ maximizes $\mathbf{v}^\top \mathbf{A}\mathbf{v}$, as per our guess.

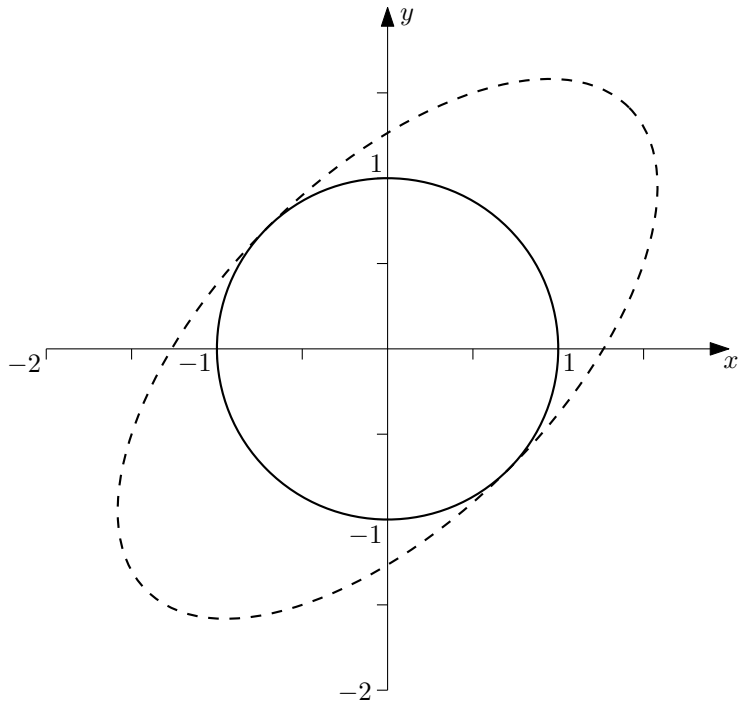

Figure 2: The same scaling as Figure 1, but now rotated 45 degrees.

2.4 Diagonals in Disguise

Let's generalize our solution in Section 2.3 by considering matrices $\mathbf{A}$ that, while not diagonal, are really just "diagonals in disguise." Geometrically, what we mean is that $\mathbf{A}$ still does nothing other than stretch out different orthogonal axes, possibly with these axes being a "rotated version" of the original ones. See Figure 2 for a rotated version of the previous example, which corresponds to the matrix

$$\begin{pmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\text{rotate back } 45^\circ} \cdot \underbrace{\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}}_{\text{stretch}} \cdot \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\text{rotate clockwise } 45^\circ}. \tag{6}$$

So what's a "rotated diagonal" in higher dimensions? The appropriate generalization of a rotation is an *orthogonal matrix*.[4] Recall that an orthogonal matrix $\mathbf{Q}$ is a square matrix where the columns are a set of orthonormal vectors — that is, each column has unit length, and the inner product of two different columns is 0. A key property of orthogonal matrices is that they preserve length — that is, $\|\mathbf{Q}\mathbf{v}\| = \|\mathbf{v}\|$ for every vector $\mathbf{v}$. We review briefly why this is: since the columns of $\mathbf{Q}$ are orthonormal vectors, we have

$$\mathbf{Q}^\top \mathbf{Q} = \mathbf{I},$$

[4] In addition to rotations, orthogonal matrices capture operations like reflections and permutations of the coordinates.

since the (i, j) entry of $\mathbf{Q}^\top\mathbf{Q}$ is just the inner product of the ith and jth columns of $\mathbf{Q}$. (It also follows that $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}$. This shows that $\mathbf{Q}^\top$ is also an orthogonal matrix, or equivalently that the rows of an orthogonal matrix are orthonormal vectors.) This means that the inverse of an orthogonal matrix is simply its transpose. (For example, see the two inverse rotation matrices in (6).) Then, we have

$$\|\mathbf{Q}\mathbf{v}\|^2 = (\mathbf{Q}\mathbf{v})^\top(\mathbf{Q}\mathbf{v}) = \mathbf{v}^\top\mathbf{Q}^\top\mathbf{Q}\mathbf{v} = \mathbf{v}^\top\mathbf{v} = \|\mathbf{v}\|^2,$$

showing that $\mathbf{Q}\mathbf{v}$ and $\mathbf{v}$ have same norm.

Now consider a matrix $\mathbf{A}$ that can be written $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ for an orthogonal matrix $\mathbf{Q}$ and diagonal matrix $\mathbf{D}$ as in (4) — this is what we mean by a "diagonal in disguise." Such a matrix $\mathbf{A}$ has a "direction of maximum stretch" — the (rotated) axis that gets stretched the most (i.e., by λ_1). Since the direction of maximum stretch under $\mathbf{D}$ is $\mathbf{e}_1$, the direction of maximum stretch under $\mathbf{A}$ is the direction that gets mapped to $\mathbf{e}_1$ under $\mathbf{Q}^\top$ — which is $(\mathbf{Q}^{-1})^\top\mathbf{e}_1$ or, equivalently, $\mathbf{Q}\mathbf{e}_1$. Notice that $\mathbf{Q}\mathbf{e}_1$ is simply the first column of $\mathbf{Q}$ — the first row of $\mathbf{Q}^\top$.

This direction of maximum stretch is again the solution to the variance-maximization problem (2). To see this, first plug in this choice $\mathbf{v}_1 = \mathbf{Q}\mathbf{e}_1$ to obtain

$$\mathbf{v}_1^\top\mathbf{A}\mathbf{v}_1 = \mathbf{v}_1^\top\mathbf{Q}\mathbf{D}\mathbf{Q}^\top\mathbf{v}_1 = \mathbf{e}_1^\top\mathbf{Q}^\top\mathbf{Q}\mathbf{D}\mathbf{Q}^\top\mathbf{Q}\mathbf{e}_1 = \mathbf{e}_1^\top\mathbf{D}\mathbf{e}_1 = \lambda_1.$$

Second, for every unit vector $\mathbf{v}$, $\mathbf{Q}^\top\mathbf{v}$ is also a unit vector (since $\mathbf{Q}$ is orthogonal), so $\mathbf{v}^\top\mathbf{Q}\mathbf{D}\mathbf{Q}^\top\mathbf{v}$ is an average of the λ_i's, just as in (5) (with averaging weights given by the squared coordinates of $\mathbf{Q}^\top\mathbf{v}$, rather than those of $\mathbf{v}$). Thus $\mathbf{v}^\top\mathbf{A}\mathbf{v} \leq \lambda_1$ for every unit vector $\mathbf{v}$, implying that $\mathbf{v}_1 = \mathbf{Q}\mathbf{e}_1$ maximizes $\mathbf{v}^\top\mathbf{A}\mathbf{v}$.

2.5 General Covariance Matrices

We've seen that when the matrix $\mathbf{A}$ can be written as $\mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ for an orthogonal matrix $\mathbf{Q}$ and diagonal matrix $\mathbf{D}$, it's easy to understand how to maximize the variance (2): the optimal solution is to set $\mathbf{v}$ equal to the first row of $\mathbf{Q}^\top$, and geometrically this is just the direction of maximum stretch when we view $\mathbf{A}$ as a map from $\mathbb{R}^d$ to itself. But we don't want to solve the problem (2) only for diagonals in disguise — we want to solve it for an arbitrary covariance matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$. Happily, we've already done this: recall from linear algebra that *every matrix* $\mathbf{A}$ *of the form* $\mathbf{X}^\top\mathbf{X}$ *can be written as* $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ *for an orthogonal matrix* $\mathbf{Q}$ *and diagonal matrix* $\mathbf{D}$ *as in* (4).[5]

Summarizing, we've shown the following:

[5]If you look back at your notes from your linear algebra class, the most likely relevant statement is that every symmetric matrix can be diagonalized by orthogonal matrices. (The proof is not obvious, but it is covered in standard linear algebra courses.) For some symmetric matrices $\mathbf{A}$, the corresponding diagonal matrix $\mathbf{D}$ will have some negative entries. For symmetric matrices of the form $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$, however, all of the diagonal entries are nonnegative. Such matrices are called "positive semindefinite." To see this fact, note that (i) if $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$, then $\mathbf{v}^\top\mathbf{A}\mathbf{v} = (\mathbf{X}\mathbf{v})^\top\mathbf{X}\mathbf{v} \geq 0$ for every $\mathbf{v}$; and (ii) if $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$ with the ith diagonal entry of $\mathbf{D}$ negative, then taking $\mathbf{v} = \mathbf{Q}\mathbf{e}_i$ provides a vector with $\mathbf{v}^\top\mathbf{A}\mathbf{v} < 0$ (why?).

When $k = 1$, the solution to (2) is the first row of $\mathbf{Q}^\top$, where $\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ with $\mathbf{Q}$ orthonormal and $\mathbf{D}$ diagonal with sorted diagonal entries.

2.6 Larger Values of k

The solution to the variance-maximization problem (1) is analogous, namely to pick the k orthogonal axes that are stretched the most by $\mathbf{A}$. Extending the derivation above gives:

> For general k, the solution to (1) is the first k rows of $\mathbf{Q}^\top$, where $\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ with $\mathbf{Q}$ orthonormal and $\mathbf{D}$ diagonal with sorted diagonal entries.

Note that since $\mathbf{Q}$ is orthonormal, the first k rows of $\mathbf{Q}^\top$ are indeed a set of k orthonormal vectors, as required in (1).[6] For a matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$, these are called the *top k principal components of* $\mathbf{X}$.

2.7 Eigenvectors and Eigenvalues

Now that we've characterized the top k principal components as the first k rows of $\mathbf{Q}^\top$ in the decomposition $\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$, how can we actually compute them? The answer follows from a reinterpretation of these vectors as eigenvectors of the covariance matrix $\mathbf{X}^\top\mathbf{X}$.

Recall that an *eigenvector* of a matrix $\mathbf{A}$ is a vector $\mathbf{v}$ that is stretched in the same direction by $\mathbf{A}$, meaning $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ for some $\lambda \in \mathbb{R}$. The value λ is the corresponding *eigenvalue*. Eigenvectors are just the "axes of stretch" of the geometric discussions above, with eigenvalues giving the stretch factors.[7]

When we write $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ as $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$, we're actually writing the matrix in terms of its eigenvectors and eigenvalues — the rows of $\mathbf{Q}^\top$ are the eigenvectors of $\mathbf{A}$, and the diagonal entries in $\mathbf{D}$ are the corresponding eigenvalues. To see this, consider the ith row of $\mathbf{Q}^\top$, which can be written $\mathbf{Q}\mathbf{e}_i$. Since $\mathbf{Q}^\top\mathbf{Q} = \mathbf{Q}\mathbf{Q}^\top = \mathbf{I}$ we have, $\mathbf{A}\mathbf{Q}\mathbf{e}_i = \mathbf{Q}\mathbf{D}\mathbf{e}_i = \lambda_i\mathbf{Q}\mathbf{e}_i$. Hence the ith row of $\mathbf{Q}^\top$ is the eigenvector of $\mathbf{A}$ with eigenvalue λ_i. Thus:

> *PCA boils down to computing the k eigenvectors of the covariance matrix $\mathbf{X}^\top\mathbf{X}$ that have the largest eigenvalues.*

You will often see the sentence above given as the definition of the PCA method; after this lecture there should be no mystery as to where the definition comes from.

Are the top k principal components of $\mathbf{X}$ — the k eigenvectors of $\mathbf{X}^\top\mathbf{X}$ that have the largest eigenvalues — uniquely defined? More or less, but there is some fine print. First, the set of diagonal entries in the matrix $\mathbf{D}$ — the set of eigenvalues of $\mathbf{X}^\top\mathbf{X}$ — is uniquely defined. Recall that we're ordering the coordinates so that these entries are in sorted order.

[6] An alternative way to arrive at the same k vectors is: choose $\mathbf{v}_1$ as the variance-maximizing direction (as in (2)); choose $\mathbf{v}_2$ as the variance-maximizing direction orthogonal to $\mathbf{v}_1$; choose $\mathbf{v}_3$ as the variance-maximizing direction orthogonal to both $\mathbf{v}_1$ and $\mathbf{v}_2$; and so on.

[7] Eigenvectors and eigenvalues will reappear in Lectures #11–12 on spectral graph theory, where they are unreasonably effective at illuminating network structure. They also play an important role in the analysis of Markov chains (Lecture #14).

If these eigenvalues are all distinct, then the matrix $\mathbf{Q}$ is also unique (up to a sign flip in each column).[8] If an eigenvalue occurs with multiplicity d, then there is a d-dimensional subspace of corresponding eigenvectors. Any orthonormal basis of this subspace can be chosen as the corresponding principal components.

3 The Power Iteration Method

3.1 Methods for Computing Principal Components

How does one compute the top k principal components? One method uses the "singular value decomposition (SVD);" we'll talk about this in detail next lecture. (The SVD can be computed in roughly cubic time, which is fine for medium-size but not for large-scale problems.) We conclude this lecture with a description of a second method, *power iteration*. This method is popular in practice, especially in settings where one only wants the top few eigenvectors (as is often the case in PCA applications). In Matlab and SciPythere are optimized and ready-to-use implementations of both the SVD and power iteration methods.

3.2 The Algorithm

We first describe how to use the power iteration method to compute the first eigenvector (the one with largest eigenvalue), and then explain how to use it to find the rest of them. To understand the geometric intuition behind the method, recall that if one views the matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ as a function that maps the unit sphere to an ellipsoid, then the longest axis of the ellipsoid corresponds to the top eigenvector of $\mathbf{A}$ (Figures 1 and 2). Given that the top eigenvector corresponds to the direction in which multiplication by $\mathbf{A}$ stretches the vector the most, it is natural to hope that if we start with a random vector $\mathbf{v}$, and keep applying $\mathbf{A}$ over and over, then we will end up having stretched $\mathbf{v}$ so much in the direction of $\mathbf{A}$'s top eigenvector that the image of $\mathbf{v}$ will lie almost entirely in this same direction. For example, in Figure 2, applying $\mathbf{A}$ twice (rotate/stretch/rotate-back/rotate/stretch/rotate-back) will stretch the ellipsoid twice as far along the southwest-northeast direction (i.e., along the first eigenvector). Further applications of $\mathbf{A}$ will make this axis of stretch even more pronounced. Eventually, almost all of the points on the original unit circle get mapped to points that are very close to this axis.

Here is the formal statement of the power iteration method:

[8] If $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$ and $\hat{\mathbf{Q}}$ is $\mathbf{Q}$ with all signs flipped in some column, then $\mathbf{A} = \hat{\mathbf{Q}}\mathbf{D}\hat{\mathbf{Q}}^\top$ as well.

Algorithm 1
POWER ITERATION

Given matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$:

- Select random unit vector $\mathbf{u}_0$
- For $i = 1, 2, \ldots$, set $\mathbf{u}_i = \mathbf{A}^i \mathbf{u}_0$. If $\mathbf{u}_i / ||\mathbf{u}_i|| \approx \mathbf{u}_{i-1} / ||\mathbf{u}_{i-1}||$, then return $\mathbf{u}_i / ||\mathbf{u}_i||$.

Often, rather than computing $\mathbf{A}\mathbf{u}_0, \mathbf{A}^2\mathbf{u}_0, \mathbf{A}^3\mathbf{u}_0, \mathbf{A}^4\mathbf{u}_0, \mathbf{A}^5\mathbf{u}_0, \ldots$ one instead uses repeated squaring and computes $\mathbf{A}\mathbf{u}_0, \mathbf{A}^2\mathbf{u}_0, \mathbf{A}^4\mathbf{u}_0, \mathbf{A}^8\mathbf{u}_0, \ldots$. (This replaces a larger number of matrix-vector multiplications with a smaller number of matrix-matrix multiplications.)

3.3 The Analysis

To show that the power iteration algorithm works, we first establish that if $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$, then $\mathbf{A}^i = \mathbf{Q}\mathbf{D}^i\mathbf{Q}^\top$ — that is, $\mathbf{A}^i$ has the same orientation (i.e. eigenvectors) as $\mathbf{A}$, but all of the eigenvalues are raised to the ith power (and are hence exaggerated—e.g. if $\lambda_1 > 2\lambda_2$, then $\lambda_1^{10} > 1000\lambda_2^{10}$).

Claim 3.1 *If* $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$, *then* $\mathbf{A}^i = \mathbf{Q}\mathbf{D}^i\mathbf{Q}^\top$.

Proof: We prove this via induction on i. The base case, $i = 1$ is immediate. Assuming that the statement holds for some specific $i \geq 1$, consider the following:

$$\mathbf{A}^{i+1} = \mathbf{A}^i \cdot \mathbf{A} = \mathbf{Q}\mathbf{D}^i \underbrace{\mathbf{Q}^\top\mathbf{Q}}_{=\mathbf{I}} \mathbf{D}\mathbf{Q}^\top = \mathbf{Q}\mathbf{D}^i\mathbf{D}\mathbf{Q}^\top = \mathbf{Q}\mathbf{D}^{i+1}\mathbf{Q}^\top,$$

where we used our induction hypothesis to get the second equality, and the third equality follows from the orthogonality of $\mathbf{Q}$. ■

We can now quantify the performance of the power iteration algorithm:

Theorem 3.2 *For any* $\epsilon, \delta > 0$, *letting* $\mathbf{v}_1$ *denote the top eigenvector of* $\mathbf{A}$, *with probability at least* $1 - \delta$ *over the choice of* $\mathbf{u}_0$,

$$|\langle \frac{\mathbf{A}^t\mathbf{u}_0}{\|\mathbf{A}^t\mathbf{u}_0\|}, \mathbf{v}_1 \rangle| \geq 1 - \epsilon,$$

provided

$$t > O\left(\frac{\log(d) + \log\frac{1}{\epsilon} + \log\frac{1}{\delta}}{\log\frac{\lambda_1}{\lambda_2}}\right),$$

where λ_1, *and* λ_2 *denote the largest and second-largest eigenvalues of* $\mathbf{A}$, *and* d *is the dimension of* A.

This result shows that the number of iterations required scales as $\frac{\log d}{\log(\lambda_1/\lambda_2)}$. The ratio λ_1/λ_2 is an important parameter called the *spectral gap* of the matrix $\mathbf{A}$. The bigger the spectral gap, the more pronounced is the direction of maximum stretch (compared to other axes of stretch). If the spectral gap is large, then we are in excellent shape. If $\lambda_1 \approx \lambda_2$, then the algorithm might take a long time (or might never) find $\mathbf{v}_1$.[9]

Proof of Theorem 3.2: Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ denote the eigenvectors of $\mathbf{A}$, with associated eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots$. These vectors form an orthonormal basis for $\mathbb{R}^d$. Suppose we write the random initial vector $\mathbf{u}_0 = \sum_{j=1}^d c_j \mathbf{v}_j$ in terms of this basis. From Claim 3.1, $\mathbf{A}^t\mathbf{u}_0 = \sum_{j=1}^d c_j \lambda_j^t \mathbf{v}_j$, and hence we have the following:

$$\begin{aligned}
|\langle \frac{\mathbf{A}^t\mathbf{u}_0}{\|\mathbf{A}^t\mathbf{u}_0\|}, \mathbf{v}_1 \rangle| &= \frac{c_1\lambda_1^t}{\sqrt{\sum_{i=1}^d (\lambda_i^t c_i)^2}} \\
&\geq \frac{c_1\lambda_1^t}{\sqrt{c_1^2\lambda_1^{2t} + \lambda_2^{2t}\sum_{i\geq 2} c_i^2}} \geq \frac{c_1\lambda_1^t}{\sqrt{c_1^2\lambda_1^{2t} + \lambda_2^{2t}}} \\
&\geq \frac{c_1\lambda_1^t}{c_1\lambda_1^t + \lambda_2^t} \geq 1 - \frac{1}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t,
\end{aligned}$$

where the second-to-last inequality follows from the fact that for any $a, b \geq 0$, $\sqrt{a^2+b^2} \leq a + b$, and the last inequality follows from dividing the top and bottom by the numerator, $c_1\lambda_1^t$, and noting that for $x > 0$ it holds that $1/(1+x) \geq 1 - x$.

So, to have this quantity be close to 1, we just need to ensure that $\frac{1}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t$ is close to zero. Intuitively, c_1 will likely not be too much smaller than $1/\sqrt{d}$, in which case we just need t to be sufficiently large so that $(\lambda_1/\lambda_2)^t \gg \sqrt{d}$, which holds for $t > \log(d)/\log(\lambda_1/\lambda_2)$.

To make this precise, recall that $\mathbf{u}_0$ was selected to be a random unit vector: we claim that for any direction, $\mathbf{v}$, the magnitude of the projection of a random unit vector along $\mathbf{v}$ is at least $\frac{\delta}{2\sqrt{d}}$ with probability $1 - \delta$. (This can be proved from the fact that a random unit vector can be generated by independently selecting each coordinate from the standard Gaussian, then scaling so as to have unit norm.) So, if we want $\frac{1}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t < \epsilon$ with probability at least $1 - \delta$, we simply plug in $\frac{\delta}{2\sqrt{d}}$ for c_1 and solve for t, yielding the claimed theorem. ■

3.4 Computing Further Principal Components

Applying the power iteration algorithm to the covariance matrix $\mathbf{X}^\top\mathbf{X}$ of a data matrix $\mathbf{X}$ finds (a close approximation to) the top principal component of $\mathbf{X}$. We can reuse the same method to compute subsequent principal components one-by-one, up to the desired number k. Specifically, to find the top k principal components:

[9] If $\lambda_1 = \lambda_2$, then $\mathbf{v}_1$ and $\mathbf{v}_2$ are not uniquely defined — there is a two-dimensional subspace of eigenvectors with this eigenvalue. In this case, the power iteration algorithm will simply return a vector that lies in this subspace, which is the correct thing to do.

1. Find the top component, $\mathbf{v}_1$, using power iteration.

2. Project the data matrix orthogonally to $\mathbf{v}_1$:

$$\begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_n & — \end{bmatrix} \mapsto \begin{bmatrix} — & (\mathbf{x}_1 - \langle \mathbf{x}_1, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \\ — & (\mathbf{x}_2 - \langle \mathbf{x}_2, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \\ & \vdots & \\ — & (\mathbf{x}_n - \langle \mathbf{x}_n, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \end{bmatrix}.$$

   This corresponds to subtracting out the variance of the data that is already explained by the first principal component $\mathbf{v}_1$.

3. Recurse by finding the top $k-1$ principal components of the new data matrix.

The correctness of this greedy algorithm follows from the fact that the k-dimensional subspace that maximizes the norms of the projections of a data matrix X contains the $(k-1)$-dimensional subspace that maximizes the norms of the projections.

3.5 How to Choose k?

How do you know how many principal components are "enough"? For data visualization, often you just want the first few. In other applications, like compression, the simple answer is that you don't. In general, it is worth computing a lot of them and plotting their eigenvalues. Often the eigenvalues become small after a certain point — e.g., your data might have 200 dimensions, but after the first 50 eigenvalues, the rest are all tiny. Looking at this plot might give you some heuristic sense of how to choose the number of components so as to maximize the signal of your data, while preserving the low-dimensionality.