CHAPTER 0

Introduction

This book is targeted at an audience who has already had an intro to linear algebra course, but may not have the sense that they've mastered the topic. The intended audience may not even remember linear algebra well. For those readers, we have some review sections in the appendix. (The section on bra-ket notation is not usual covered in an intro course, but is also indispensable for the material.)

You'd normally spend a lot of time in an intro to linear algebra class doing calculations, including reduced row echelon form; inverses; eigenvalues and eigenvectors; and Cramer's rule. We skip these topics, even in the review section. This book covers a lot of applications of linear algebra, and we accept that many of the interested readers will be practicioners. So we don't need to pretend that these things can't be easily calculated with a computer program. I won't go so far as to say that there's no educational value in doing some of these calculations by hand, but given that you've successfully completed a linear algebra class, I give you permission to forget them.

This book attempts to put linear algebra concepts in a more intuitive context. As such it covers most topics that could be called linear algebra. But it doesn't cover everything. Some very rich and interesting advanced topics that are not covered here include:

- Abstract vector spaces, function spaces, infinite-dimensional spaces
- Tensors and multi-linear algebra
- Group representations

Finally, some good news for some of you. I've written this book to avoid talking about complex numbers. It's surprising, but I've pulled it off. With the approach I've taken, considering complex numbers only adds a bunch of pesky qualifiers to any statement I make. So throughout the book, we'll pretend that these don't even exist.

Chapters labeled [WIP] are not available currently.

CHAPTER 1

Singular Value Decomposition

1. Statement of SVD

We start the book with the most remarkable fact of linear algebra: For any matrix A, there exists a singular value decomposition (SVD). The SVD of a matrix A is three matrices, U, D, and V, where

- U and V are orthogonal,
- D is diagonal, and
- $A = UDV^T$.

(See Section **??** for a review on orthogonal matrices.)

If A is dimension $m \times n$, then U is dimension $m \times m$, V is dimension $n \times n$, and D is dimension $m \times n$. D is zero everywhere expect for potentially D_{ii} for $1 \leqslant i \leqslant \min(m, n)$. By convention, we sometimes call $\lambda_i := D_{ii}$. As well, we'll sometimes call $r := \min(m, n)$.

Example

Example

Let's whip up an example real quick to demonstrate. We start with an array chosen arbitrarily and show that there is an SVD.

$$\begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix} = UDV^T,$$

where

$$U = \begin{pmatrix} -0.2381 & -0.9712 \\ -0.9712 & 0.2381 \end{pmatrix}$$

$$D = \begin{pmatrix} 10.192 & 0 & 0 \\ 0 & 0.3399 & 0 \end{pmatrix}$$

$$V^T = \begin{pmatrix} -0.3092 & -0.4998 & -0.8090 \\ -0.7557 & 0.6456 & -0.1100 \\ -0.5774 & -0.5774 & 0.5774 \end{pmatrix}$$

This calculation was done with a computer; the numbers are rounded here. The reader should verify that:

(1) The matrices U, D, and V^T multiply to the original matrix.
(2) The two outside matrices are orthogonal. That is, the rows are orthonormal.

At this point, this is a neat party trick. It's not at all clear, but such a decomposition can be found for *any* matrix

The SVD is generally not unique for a given a matrix. This fact causes difficulties. Because the SVD is not unique, we usually say that a decomposition is *a* SVD and not *the* SVD. We show later in the chapter that SVDs are almost unique. Often we usually only need *any* SVD.

Example

Example
To give an example of non-uniqueness, let's look at the matrix:

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

This can be decomposed as:

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

Or as:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{1/2} & \sqrt{1/2} \\ 0 & \sqrt{1/2} & -\sqrt{1/2} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{1/2} & \sqrt{1/2} \\ 0 & \sqrt{1/2} & -\sqrt{1/2} \end{pmatrix}$$

If you follow the multiplication through, you can see that for the outside matrices, everything except the first row and column get ignored. We must choose orthonormal vectors for the other two rows and columns in order to satisfy the orthogonality condition of the SVD. But outside of that constraint, we're free to do whatever we want with these other rows and columns.

Low-rank matrices are one way that uniqueness can fail, but not the only way. See Section 3 for a more complete discussion.

Soon we'll show that the SVD exists. Said another way: We'll show there exists bases in the domain and range for which A is diagonal. (See Section 3 for a review of this concept.) On its face, this is a simple-yet-powerful theorem that doesn't have the usual qualifiers seen in linear algebra. It doesn't require complex numbers, and it's one of the few theorems that apply cleanly to rectangular matrices. But beyond these surface advantages, we'll see that the SVD is widely applicable. In fact, it's so applicable that we make this a cornerstone of the book. The SVD is critical to understanding linear algebra, because it expresses something fundamental about linear transformations.

After a brief detour to discuss bra-ket form, we'll spend the rest of this section exploring that fundamentality. We will view the SVD through three major lenses: Firstly by viewing matrixes as a collection of eigenvectors and eigenvectors. Then by viewing matrixes as the product of vectors and covectors. Finally by viewing matrices as bilinear forms.

1.1. Bra-ket Form. This section uses Bra-ket Notation heavily. If you're unfamiliar with this form, please read Section **??** before proceeding.

An often more-convenient way to write the SVD is:

$$A = \sum_{i=1}^{r} \lambda_i |u_i\rangle\langle v_i|$$

Here $|u_i\rangle$ are the columns of U and $\langle v_i|$ are the rows of V^T.

THEOREM 1.1.1. *Any matrix A can be written as:*

$$A = \sum_{i=1}^{r} \lambda_i |u_i\rangle\langle v_i|$$

PROOF. Say A is $m \times n$ dimensional with $r = \min(m, n)$ as usual. We write the matrices as though $m < n$, but other cases work out the same way. Write the SVD of A as:

$$A = UDV^T = \begin{bmatrix} | & | & & | \\ |u_1\rangle & |u_2\rangle & \cdots & |u_m\rangle \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \lambda_r & \cdots & 0 \end{bmatrix} \begin{bmatrix} — & \langle v_1| & — \\ — & \langle v_2| & — \\ & \vdots & \\ — & \langle v_n| & — \end{bmatrix}$$

We can write the diagonal matrix as:

$$D = \sum_{i=1}^{r} \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_i & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

Each summand is the diagonal matrix with all but the i-th entry zeroed out. We distribute the U on the left and V^T on the right, to end up with:

$$\begin{aligned} A &= \sum_{i=1}^{r} \begin{bmatrix} | & | & & | \\ |u_1\rangle & |u_2\rangle & \cdots & |u_m\rangle \\ | & | & & | \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_i & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} — & \langle v_1| & — \\ — & \langle v_2| & — \\ & \vdots & \\ — & \langle v_n| & — \end{bmatrix} \\ &= \sum_{i=1}^{r} \begin{bmatrix} | & | & & | \\ |u_1\rangle & |u_2\rangle & \cdots & |u_m\rangle \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & 0 & — \\ & \vdots & \\ — & \lambda_i\langle v_i| & — \\ & \vdots & \\ — & 0 & — \end{bmatrix} \\ &= \sum_{i=1}^{r} \begin{bmatrix} \lambda_i |u_i\rangle_1\langle v_i|_1 & \lambda_i |u_i\rangle_1\langle v_i|_2 & \cdots \\ \lambda_i |u_i\rangle_2\langle v_i|_1 & \lambda_i |u_i\rangle_2\langle v_i|_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{aligned}$$

When you compare this final form to the definition of the outer product, then you see that it equals $\sum_{i=1}^{r} \lambda_1 |u_i\rangle\langle v_i|$. □

The reader should pause to appreciate that although the matrix is $m \times n$, only $r = \min(m, n)$ summands are needed. What happened to the remaining $n - m$ rows of V^T?

Because of the orthogonality of U and V, the set of vectors $\{\langle v_i|\}$ form a basis of a subspace of the domain, and the covectors $\{|u_i\rangle\}$ form a basis of a subspace of the codomain. (See Exercises 1.1.6 and 1.1.7 for a more precise statement.) The orthogonality of these vectors is very important; we use that fact constantly throughout the book.

To see why this form is convenient, we'll show a quick proof that will be used later.

LEMMA 1.1.2. *Given an SVD, $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, then*

$$A^T A = \sum_i \lambda_i^2 |v_i\rangle\langle v_i| \tag{1}$$

PROOF. Transposition distributes over sums. As well $(AB)^T = B^T A^T$, so we have that each summand $(|u_i\rangle\langle v_i|)^T = |v_i\rangle\langle u_i|$. So we can start the $A^T A$ equation as:

$$\begin{aligned} A^T A &= \left(\textstyle\sum_i \lambda_i |v_i\rangle\langle u_i|\right) \left(\textstyle\sum_j \lambda_j |u_j\rangle\langle v_j|\right) \\ &= \textstyle\sum_i \sum_j \lambda_i \lambda_j |v_i\rangle\langle u_i|u_j\rangle\langle v_j| \end{aligned}$$

Now we look closer at the middle term $\langle u_i|u_j\rangle$. Because the vectors $|u_j\rangle$ are orthonormal, this product equals 0 when $i \neq j$ and 1 when $i = j$. So the double index which loops over all possible values of i and j ends up being non-zero for only those terms where $i = j$.

$$A^T A = \sum_i \lambda_i \lambda_i |v_i\rangle\langle u_i|u_i\rangle\langle v_i| = \sum_i \lambda_i^2 |v_i\rangle\langle v_i|$$

□

We will call the decomposition $\sum_i \lambda_i |u_i\rangle\langle v_i|$ the *bra-ket form* (or *summation form* of the SVD; the UDV^T form is called the *matrix form* of the SVD. The steps from above can be done in reverse to go from bra-ket form to matrix form.

Example

Example
Given an SVD in bra-ket form $A = 5|u_1\rangle\langle v_1| + 2|u_2\rangle\langle v_2|$, where

$$\begin{aligned} \vec{u}_1 &= (1,0) \\ \vec{v}_1 &= (\sqrt{1/2}, -\sqrt{1/2}) \\ \vec{u}_2 &= (0,-1) \\ \vec{v}_2 &= (\sqrt{1/2}, \sqrt{1/2}) \end{aligned}$$

Then $A = UDV^T$ where

$$U = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

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

$$V = \begin{pmatrix} \sqrt{1/2} & -\sqrt{1/2} \\ \sqrt{1/2} & \sqrt{1/2} \end{pmatrix}$$

Notice that we can just slot the vectors and scalars into their corresponding spots. We end up with the matrix form of the SVD. In Exercise 1.1.2, you'll be asked to confirm these calculations.

There is some caveat about rank here.

Example

Example
Let's continue the above example.

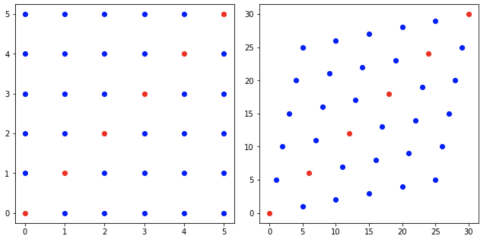
$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

We can write this using the bra-ket form as:

$$A = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} = |e_1\rangle\langle e_1|$$

The right-most expression uses the convention that $\vec{e}_i$ is a vector with a 1 in the i-th position and 0 elsewhere.

In most cases, the bra-ket form of a 3×3 matrix will have three summands, but that didn't happen for this example. We can think of this as having two extra terms with coefficient $\lambda_2 = \lambda_3 = 0$. In that case, you can get creative with what you choose $|u_2\rangle\langle v_2|$ and $|u_3\rangle\langle v_3|$ to be. As long as the vectors are orthonormal, this is a valid SVD. Filling in random vectors like this is necessary to convert back to the matrix form, as we did above. (See Exercise 1.1.4.)



1.2. Eigenvalues and Eigenvectors. A basic fact about linear maps is that they're often invariant for certain axes. That is, $A\vec{v} = \lambda\vec{v}$ for some $\vec{v}$ and λ. This is a somewhat remarkable fact. For example, a map, A, sends $(1, 0)$ to $(5, 1)$ and $(0, 1)$ to $(1, 5)$. Imagine picking up the x-y plane at these $(0, 1)$ and $(1, 0)$ and dragging them to their respective targets. As you drag these points, you stretch some parts of the plane and squeeze other parts. Yet the squeezing cancels out along the line $y = x$, and this axis only gets stretched out radially. The second and fourth quadrants get stretched apart, causing a similar cancelation effect along the line $y = -x$. This example is easy to visualize. Yet even when you do fairly complicated linear transformations with many dimensions, usually you'll get some axes somewhere where all the pulling and pushing cancel out.

The fact that eigenvalues usually exist isn't obvious, and it's a fact that people can't shut up about. If you remember one thing from linear algebra, it's probably eigenvalues. Eigenvalues and eigenvectors are important. But talking about eigenvalues is difficult and requires a lot of caveats. Eigenvalues only exist for square matrices and they are sometimes complex. Defective matrices don't have a "full set" of eigenvalues. (Defective matrices are covered much later in the book.) The SVD doesn't have any of these qualifiers. We'll see throughout this book that for many applications of eigenvalues, the SVD will work instead. Look to Section 5 for a couple of immediate examples.

It's not clear yet why the SVD is analogous to eigenvectors. To demonstrate: Say a matrix A has a set of eigenvectors, $|v_i\rangle$ that span the domain with corresponding eigenvalues $\{\lambda_i\}$. Define a linear map P by the actions $P|e_i\rangle = |v_i\rangle$, and define D as a diagonal matrix with λ_i as the i-th entry on the diagonal. Then $A = PDP^{-1}$. This must be true because it's true for every eigenvector, and ever vector is a linear combination of eigenvectors (because the span):

$$PDP^{-1}|v_i\rangle = PD|e_i\rangle = \lambda_i P|e_i\rangle = \lambda_i|v_i\rangle = A|v_i\rangle$$

This equality makes sense. P^{-1} maps the eigenvalues of A to the standard basis; D stretches invariantly along each of the standard basis; and P maps the standard basis back. The net effect is that the invariant stretching from D gets moved to the eigenvectors. An SVD, UDV^{-1}, on the other hand rotates, stretches along standard basis, then rotates to (potentially) new rotation. (In the last sentence, we call $V^T = V^{-1}$, which is true for all orthogonal matrices; we also called the orthogonal matrices "rotations" which is a good way to think of orthogonal matrices.) We can see that PDP^{-1} and UDV^{-1} are just two decompositions of matrices. The former, called a diagonalization, requires the outside matrices to be inverses, while the SVD requires the outside matrices to be orthogonal. Diagonalizations exist sometimes, whereas SVD exist all the time. We'll see in the Section 4 that these two concepts

are intimately connected. See Chapter **??** for a discussion of how the SVD is also a diagonalization.

1.3. Vectors and Covectors. One view of matrices is that they're the product of vectors and covectors. The columns of A tell us how to act on standard basis vectors. Call the i-th column of A $|A_{*i}\rangle$. Then,

$$A = \begin{bmatrix} | & | & & | \\ |A_{*1}\rangle & |A_{*2}\rangle & \cdots & |A_{*n}\rangle \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ A|e_1\rangle & A|e_2\rangle & \cdots & A|e_n\rangle \\ | & | & & | \end{bmatrix}$$

Any vector $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ can be written as $|v\rangle = \sum_i v_i|e_i\rangle$. So to figure out the effect of A on $|v\rangle$, we only need to find the i-th component of $|v\rangle$ and multiple by $|A_{*1}\rangle$. That is,

$$\begin{aligned} A|v\rangle &= A\textstyle\sum_i v_i|e_i\rangle \\ &= \textstyle\sum_i v_i A|e_i\rangle \\ &= \textstyle\sum_i |A_{*i}\rangle v_i \end{aligned}$$

In the last line, we've written the coefficient after the covector to hint at the next step. The next step, we observe that $\langle e_i|v\rangle = \sum_j \langle e_i|v_j|e_j\rangle = v_i$, where the last equality follows from the orthonormality of the standard basis. Now we can write the above equality as

$$A|v\rangle = \textstyle\sum_i |A_{*i}\rangle\langle e_i|v\rangle$$

Comparing both sides, we can now say that $A = \sum_i |A_{*i}\rangle\langle e_i|$. This rewriting recognizes that A is a sum of vector/covector products. The covectors $\{\langle e_i|\}$ project onto the standard axes to get a scalar. Then the vectors $\{|A_{*i}\rangle\}$ give direction to each of these scalars. This is an important way to understand matrices.

But by studying linear algebra, you learn that there's nothing special about the standard basis $\{|e_i\rangle\}$. For any basis $\{|v_i\rangle\}$, we can write the matrix A with this basis on the domain. (See Section **??** for more detailed discussion on bases.) Call the columns of this new matrix $\{|u_i\rangle\}$, and it follows that $A = \sum_i |u_i\rangle\langle v_i|$. Although $\{\langle v_i|\}$ are orthonormal, it's not usually true of the resulting $\{|u_i\rangle\}$. The existence of an SVD is the existence of a domain basis $\{\langle v_i|\}$ for which the corresponding $\{|u_i\rangle\}$ are also orthogonal, but not normal. We can write $|u_i\rangle = \lambda_i|u_i'\rangle$, so that $\{u_i'\}$ are orthonormal. The basis $\{\langle v_i|\}$ that produces orthogonal columns on A is a very special basis. There's no reason to prefer to represent A in the standard basis or any other basis. So we might as well choose $\{\langle v_i|\}$, giving $A = \sum_i \lambda_i|u_i'\rangle\langle v_i|$, the SVD. We'll see throughout the course of this book that it simplifies many things.

I want to revisit the equivalence of the two forms of the SVD with this new language. Given a UDV^T representation of a matrix, A, we can write $V^T = \sum_i |e_i\rangle\langle v_i|$, where the $|v_i\rangle$ are the rows of V^T, and we can write $U = \sum_i |u_i\rangle\langle e_i|$ where the $|u_i\rangle$ are the columns of U. And of course $D = \sum_i \lambda_i|e_i\rangle\langle e_i|$, where λ_i are the diagonal components of D. Now we have

$$A = UDV^T = \left(\sum_i |u_i\rangle\langle e_i|\right)\left(\sum_j \lambda_j |e_j\rangle\langle e_j|\right)\left(\sum_k |e_k\rangle\langle v_k|\right) = \sum_i \lambda_i |u_i\rangle\langle v_i|$$

where the last equality follows from the orthonomality of $|e_i\rangle$.

1.4. Coordinates. In this section we continue to build intuition.

Example: Coordinate Systems

Coordinate Systems Consider a basis, $\mathcal{B}$, of $\mathbb{R}^3$ given by $\vec{v}_1 = \langle 1/\sqrt{2}, 1/\sqrt{2}, 0\rangle$, $\vec{v}_2 = \langle 1/\sqrt{2}, -1/\sqrt{2}, 0\rangle$, $\vec{v}_3 = \langle 0, 0, -1\rangle$. Because $\mathcal{B}$ is a basis, we know that any vector in $\mathbb{R}^3$ can be written as a linear combination of of these three vectors. For example, $\langle 1, 2, 3\rangle = 3/\sqrt{2}\vec{v}_1 - 1/\sqrt{2}\vec{v}_2 - 3\vec{v}_3$. Occasionally, this linear combination is written in its own vector notation, with a special $\mathcal{B}$ subscript. $\langle 1, 2, 3\rangle = \langle 3/\sqrt{2}, -1/sqrt2, -3\rangle_{\mathcal{B}}$.

The purpose of this short section is to make the connection that both $\langle 1, 2, 3\rangle$ and $\langle 3/\sqrt{2}, -1/sqrt2, -3\rangle_{\mathcal{B}}$ are equally legitimate representations of the same vector. This is a conceptual decoupling between a vector and the coordinates we use to write it. The observation is akin to the realization that the number 14 can be written in binary as 1110; the number itself is not either of these things, but is merely represented by them.

And if any choice of coordinates are equally legitimate, we may as well choose the bases given by the SVD, $M = \sum_i |u_i\rangle\langle v_i|$. That is, if $\mathcal{B}_u = (|u_1\rangle, |u_2\rangle, \cdots, |u_n\rangle)$ and $\mathcal{B}_v = (|v_1\rangle, |v_2\rangle, \cdots, |v_n\rangle)$ then $M\langle x_1, x_2, \cdots x_n\rangle_{\mathcal{B}_u} = \langle \lambda_1 x_1, \lambda_2 x_2, \cdots, \lambda_n x_n\rangle_{\mathcal{B}_v}$. So the existance of an SVD says that any matrix is merely a dillation under the correct coordinate systems. Quite a surprising fact!

We can take this notation a bit further even.

Example: Coordinate Systems

Functions on Coordinate Systems You and two friends, Leeta and Odo, are doing some classical physics problem, and have to draw your own axes to solve the problem. You correctly label the canonical x and y axes, $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$. But Leeta is looking at the problem from an angle, so she ends up with axes $\langle 1/\sqrt{2}, 1/\sqrt{2}\rangle$ and $\langle -1/\sqrt{2}, 1/\sqrt{2}\rangle$. And Odo is left-handed, so he ends up with axes $\langle 0, 1\rangle$ and $\langle 1, 0\rangle$. As an intermediate step, the problem requires you to write down a rotation matrix. Because the axes are made up by the solver, they shouldn't affect the ultimate solution. So you can write the rotation in any basis. When you use the canonical basis $\mathcal{B}$, you get the usual rotation matrix $R_{\mathcal{B}\mathcal{B}} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. When Leeta calculates the same matrix in her basis, $\mathcal{B}_L$, she too ends up with $[R]_{\mathcal{B}_L\mathcal{B}_L} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. Note the notation on the matrix, which means that both the domain and range are written in Leeta's coordinates, $\mathcal{B}_L$. Now Odo's a little odd, so in his coordinates, $\mathcal{B}_O$, counterclockwise rotation rotates his y-axis towards his x-axis, and he gets $[R]_{\mathcal{O}\mathcal{O}} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$.
As you may guess, it's also possible to write down the rotation matrix where the input is in one coordinate system and the output is in another. For example $[R]_{\mathcal{B}_L\mathcal{B}_O} = \begin{pmatrix} \cos(\theta - \pi/4) & \sin(\theta - \pi/4) \\ -\sin(\theta - \pi/4) & \cos(\theta - \pi/4) \end{pmatrix}$.
If we have a change-of-basis matrix, C that maps our coordinates to Leeta's then we could write, for any matrix, M, $[M]_{\mathcal{B}\mathcal{B}} = C^{-1}[M]_{\mathcal{B}_L\mathcal{B}_L}C$. The change-of-basis matrix itself can be written as $C = [I]_{\mathcal{B}_L\mathcal{B}}$.

With this notation for matrices in different basis, the SVD of a matrix M is simply bases $\mathcal{B}$ and $\mathcal{B}'$ for which $[M]_{\mathcal{B}\mathcal{B}'} = I$.

1.5. Bilinear Forms. Linear forms are defined for matrices generally. But we'll define a linear form as a function $f : \mathbb{R}^n \to \mathbb{R}$ that is "linear" meaning that it satifies the two properties:

(1) $f(\vec{x} + \vec{y}) = f(\vec{x}) + f(\vec{y})$ for all $\vec{x}, \vec{y} \in \mathbb{R}^n$.
(2) $f(\lambda\vec{x}) = \lambda f(\vec{x})$ for all $\lambda \in \mathbb{R}$ and $\vec{x} \in \mathbb{R}^n$.

We state without proving that all linear forms can be written as $\lambda\langle u|$ for some unit vector $\vec{u}$, meaning that $f(|x\rangle) = \lambda\langle u|x\rangle$.

By the definition of linear forms, the sum of any two linear forms is itself a linear form. It's interesting, though not surprising that the sum of two covectors $\lambda\langle u|$ must therefore result in a new covector, so that it can also be written this way.

By analogy a bilinear form is a function $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ that satisfies linearity for both inputs:

(1) $f(\vec{x} + \vec{y}, \vec{z}) = f(\vec{x}, \vec{z}) + f(\vec{y}, \vec{z})$ for all $\vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$.
(2) $f(\vec{z}, \vec{x} + \vec{y}) = f(\vec{z}, \vec{x}) + f(\vec{z}, \vec{y})$ for all $\vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$.
(3) $f(\lambda\vec{x}, \vec{y}) = \lambda f(\vec{x}, \vec{y}) = f(\vec{x}, \lambda\vec{y})$ for all $\lambda \in \mathbb{R}$ and $\vec{x}, \vec{y} \in \mathbb{R}^n$.

We also state without proof that all bilinear forms can be written as a matrix, A, so that $f(\vec{x}, \vec{y}) = \langle x|A|y\rangle$. Conversely all matrices yield a bilinear form in this way. So maybe a good way to think about matrices is exactly as bilinear forms. In analogy to linear forms, say that a particular matrix A can be written as $\lambda|u\rangle\langle v|$. Then it's action on a pair of vectors $\vec{x}, \vec{y}$ is $\lambda\langle x|u\rangle\langle v|y\rangle$, which is a product of two linear forms. This seems like a useful analogy. But unlike linear forms, we can't add together a bunch of matrices of the form $\lambda|u\rangle\langle v|$ and still expect to get a matrix of the form $\lambda|u\rangle\langle v|$. Instead you get a sum of such matrices, $\sum_i \lambda_i|u_i\rangle\langle v_i|$. The surprising thing here is that if you added very many such matrices, you could always reduce to $r := \min(m, n)$ summands. Either way, this representation seems like a sensible way to represent a matrix.

1.6. Conventions. The bra-ket form of the SVD yields a set of scalars $\{\lambda_i\}$ and vectors $\{\vec{u}_i, \vec{v}_i\}$. We can and do require that all of these be real. We also require that all the scalars be positive $\lambda_i > 0$. We can require this because:

(1) If an SVD has a negative summand, $\lambda_i|u\rangle\langle v|$, then we get a positive-only SVD by replacing that summand with $(-\lambda_i)|-u\rangle\langle v|$.
(2) If an SVD has a summand with $\lambda_i = 0$, then we omit this and don't count it as part of the SVD.

Finally we adopt the convention that these are ordered: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \lambda_r$.

We prove in Section 3 that with our conventions, then multiset[1] of scalars is unique to the matrix A; that is, all SVDs of A yield the same multiset. We call these values the *singular values* of A.

1.7. Outro. The goal of this book is to build intuition and demonstrate applications. With the intuition, the applications will seem natural. (You should agree when you read Chapter 2.) Hopefully by the end of the book, you'll be able to recall or modify the techniques more easily. At this point, you should have some intuition for what an SVD is, but how somebody comes up with such a representation is still fairly abstract. The rest of this chapter proves the existence and almost-uniqueness of the SVD. Outside of these sections, this book generally is not heavy on the proofs. But the techniques used in these proofs should help you get a concrete sense for what the SVD is.

Exercises

1.1.1 Continue the example from the beginning of the chapter, write $\begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix}$ in bra-ket form. (Note: You should not have to recalculate anything.)

1.1.2 Confirm the calculation in Example X by multiplying and adding the bra-ket form of A and by multiplying the matrix form of A. These should be the same.

1.1.3 Given a matrix $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and a bra-ket form representation with a single term $\lambda|u\rangle\langle v|$. Prove that $\lambda = \pm 1$, $|u\rangle = \pm|e_1\rangle$, and $\langle v| = \pm\langle e_1|$. (This is a specific case of the almost-uniqueness that we prove in a later section.)

[1] By *multiset* we mean a collection where we care about the number of times each element appears, but we don't care about the order.

1.1.4 Given a matrix with SVD in bra-ket form:

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

Write A in matrix-form SVD two different ways.

1.1.5 The orthonormality of U and V will often be useful. Given an SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, verify the following.
(a) For $|v\rangle = \sum_i a_i |v_i\rangle$, $A|v\rangle = \sum_i a_i \lambda_i |u_i\rangle$.
(b) $\langle u_i|A|v_i\rangle = \lambda_i$.

1.1.6 Given a matrix with SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, prove that $A\vec{v} = 0$ if and only if $\vec{v}$ is perpendicular to each $\vec{v}_i$.

1.1.7 Given a matrix with SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, prove that a vector $\vec{u}$ equals $A\vec{v}$ for some $\vec{v}$ if and only if $\vec{u} \in \text{span}\,(\vec{u}_i)$.

1.1.8 An invertible square matrix has an SVD $A = UDV^T$, with $D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_r \end{pmatrix}$. Show that all $\lambda_i \neq 0$. Next argue that $D^{-1} = \begin{pmatrix} \frac{1}{\lambda_1} & & & \\ & \frac{1}{\lambda_2} & & \\ & & \ddots & \\ & & & \frac{1}{\lambda_r} \end{pmatrix}$. Finally prove that $A^{-1} = VD^{-1}U^T$.

2. Existence

In this section, we prove the existence of an SVD. This proof is constructive, in the sense that it gives an algorithm for building a bra-ket form of an SVD.

Algorithm 1: SVD

Algorithm 1: SVD

Set $A_0 \leftarrow A$ While $A_i \neq 0$

- Find the unit vectors $\vec{u}_i$ and $\vec{v}_i$ that maximize[a] $\langle u_i|A_{i-1}|v_i\rangle$.
- Set $A_i = A_{i-1} - |u_i\rangle\langle u_i|A|v_i\rangle\langle v_i|$

[a] The keen-eyed mathematician may object that maximums don't always exist. But because we've restricted to unit vectors, this is compact. Therefore maximums exist.

When the algorithm finishes, we have $A_r = 0$. So that $0 = A_r = A - \sum_{i=1}^{r} |u_i\rangle\langle u_i|A_{i-1}|v_i\rangle\langle v_i|$. We recognize that $\lambda_i := \langle u_i|A_{i-1}|v_i\rangle$ is just a scalar, and add these to both sides so that we get $A = \sum_{i=1}^{r} \lambda_i |u_i\rangle\langle v_i|$. The resulting vectors are normal, by choice. But for this to be an SVD as described in the first section, we need the vectors $\{u_i\}_i$ to be a perpendicular set, and for $\{v_i\}_i$ to be a perpendicular set. We prove that this is so in Theorem 1.2.5.

An objection to this algorithm may be that it only works if the while loop eventually terminates. For the algorithm to work, it must be true that $A_r = 0$ for some r. In Theorem 1.2.6, we show that for A with dimension $m \times n$ this must happen within $r := \min(m, n)$ steps.

Before proving these two theorems, let's build some intuition. We see that $|u_i\rangle\langle u_i|A_{i-1}|v_i\rangle\langle v_i| = \lambda_i|u_i\rangle\langle v_i|$ is just a matrix; and it's the same shape as A_{i-1}. But it's not just any matrix. It's the matrix that captures all the signal that A_{i-1} has in the axes $\vec{u}_i$ and $\vec{v}_i$. To illustrate what this means, let's first look at a lower-dimensional analogy.

Example: Covector Analogy

Covector Analogy

Consider the axis $\vec{v} = \begin{pmatrix} 3 & 3 & 2 \end{pmatrix}$. If we wanted to remove all the signal along the axis $\vec{e}_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$, then we would subtract $3\vec{e}_1 = \begin{pmatrix} 3 & 0 & 0 \end{pmatrix}$ from $\vec{v}$ to get $\begin{pmatrix} 0 & 3 & 2 \end{pmatrix}$.

Saying that this "removes signal" along e_1 means that now any vector along $\vec{e}_1$, $\lambda\vec{e}_1$, gets zeroed out by $\vec{v} - 3\vec{e}_1$. That is, $\langle v - 3e_1|e_1\rangle = 0$. We arrived at the coefficient $\lambda = 3$ by looking at the first element of $\vec{v}$. This can be formalized as $3 = \langle v|e_1\rangle$.

Now we could repeat this with the second dimension $3 = \langle v|e_2\rangle$. So now $v - 3\langle e_1| - 3\langle e_2| = \begin{pmatrix} 0 & 0 & 2 \end{pmatrix}$. This covector equals zero when it pairs with *any* linear combination of $|e_1\rangle$ and $|e_2\rangle$. That is $\left\langle \begin{pmatrix} 0 & 0 & 2 \end{pmatrix} \middle| \begin{pmatrix} \lambda & \mu & 0 \end{pmatrix} \right\rangle = 0$. We can do an SVD-like algorithm to assign v's signal to it's components: $v = \sum_{i=1}^{3}\langle v|e_i\rangle\langle e_i|$.

Moreover there's nothing special about the standard vectors. It's true that for any basis $\{\vec{v}_i\}_{i=1}^{3}$, $\langle v| = \sum_{i=1}^{3}\langle v|v_i\rangle\langle v_i|$. It's further true that if we stop short, the covector zeros out encountered vectors, like $(v - \langle v|v_1\rangle\langle v_1| - \langle v|v_2\rangle\langle v_2|)(\lambda|v_1\rangle + \mu|v_2\rangle) = 0$. We say that $\langle v|v_i\rangle\langle v_i|$ "captures all the info" from $\langle v|$ in the direction of $\langle v_i|$, because when subtract off that bit, there's nothing left in that direction.

This process is known as the Gram-Schmidt process applied to orthonormal vectors. The SVD can be thought of as a generalization of this to matrices. We say that $|u\rangle\langle u|A|v\rangle\langle v|$ captures all the info from A in the direction of $\vec{u}$ and $\vec{v}$, because $\langle u|(A - |u\rangle\langle u|A|v\rangle\langle v|)|v\rangle = 0$. (Check that this is true.) This isn't a proof yet, but this is why we think to try this.

By choosing the maximizing pair of vectors for remaining signal with each step, we guarantee that we're getting perpendicular vectors from earlier vectors. The reasoning is that if we chose some vector that was pointing in a similar direction as an earlier vector, then we'd be pointing partly in a zeroing direction. The zeroing part of the new matrix, $|u\rangle\langle u|A|v\rangle\langle v|$, starts by projecting onto the zeroed axis; if this projection is non-zero, then some part of the vector is going to waste.

2.1. Proof of the Correctness of Algorithm 2.

LEMMA 1.2.1. *For any matrix A, if $\langle u|A|v\rangle = \max_{\|u_i\|=\|v_i\|=1}\langle u_i|A|v_i\rangle$, then $A|v\rangle = \lambda|u\rangle$, for some scalar λ.*

PROOF. Fix the maximizing $|v\rangle$. $A|v\rangle$ maps to some vector. We call this vector $\lambda|t\rangle$, where $|t\rangle$ is a unit vector. (The magnitude has been factored out into the scalar λ. So $\langle u|A|v\rangle = \lambda\langle u|t\rangle$. Given that $\langle u|$ and $|t\rangle$ are unit vectors, the inner product is given by the cosine of the angle between them (Theorem 9.5.3), which is maximized when the vectors are the same. □

This is an important fact that helps with the next lemma. It also advances our intuition. We think of a matrix A as acting on a vector by projecting that vector onto a covector ($\vec{v}$) and using the resulting coefficient λ to tell us how far to go in the direction of another vector u. The above lemma says that when we've chosen the maximizing unit vector (u) and covector (v), as prescribed by the SVD algorithm, we've chosen a matching pair. If we wanted to remove the effect of the projecting onto v, then we need to subtract from the output some multiple of u.

Note that though Lemma 1.2.1 doesn't make a claim about the value of λ, it's easy to see that $\lambda = \langle u|A|v\rangle$, because $\langle u|A|v\rangle = \lambda\langle u|u\rangle = \lambda$.

Next we use this fact to see:

LEMMA 1.2.2. *Given that $\langle u_1|$ and $|v_1\rangle$ maximize $\langle u|A|v\rangle$. Define $A_1 = A - \lambda_1|u_1\rangle\langle v_1|$, with $\lambda_1 = \langle u_1|A|v_1\rangle$. Then $A_1|v_1\rangle\langle v_1| = 0$.*

PROOF.

$$\begin{aligned} A_1|v_1\rangle\langle v_1| &= \left(A - \lambda_1|u_1\rangle\langle v_1|\right)|v_1\rangle\langle v_1| \\ &= A|v_1\rangle\langle v_1| - \lambda_1|u_1\rangle\langle v_1|v_1\rangle\langle v_1| && \text{Distribute} \\ &= \left(A|v_1\rangle\right)\langle v_1| - \lambda_1|u_1\rangle\langle v_1| \\ &= \left(\lambda_1|u_1\rangle\right)\langle v_1| - \lambda_1|u_1\rangle\langle v_1| && \text{By above lemma} \\ &= 0 \end{aligned}$$

□

We can immediately conclude from Lemma 1.2.2 that $A_1 = A_1\left(I - |v_1\rangle\langle v_1|\right)$. This says that applying A_1 to a vector $|v\rangle$ is the same as first applying $(I - |v_1\rangle\langle v_1|)$ to $|v\rangle$, then applying A_1 to the resulting vector. $(I - |v_1\rangle\langle v_1|)$ is a special matrix that acts on any vector by projecting onto the subspace of all vectors perpendicular to $|v_1\rangle$. This will be justified reach the section on projection matrices. For now it's sufficient to understand:

(1) $\left(I - |v_1\rangle\langle v_1|\right)|v\rangle$ is exactly $|v\rangle$ when $\langle v_1|v\rangle = 0$. I.e. v is perpendicular to v_1, and
(2) $\|\left(I - |v_1\rangle\langle v_1|\right)|v\rangle\| < 1$, otherwise.

The first statement is obvious. The second statement should be fairly intuitive because we're subtracting off a component of a vector. This is analogous to zeroing out the first term of a unit vector, which will leave it smaller. A more formal proof of this point is left as Exercise 1.2.1.

We state without justification that this also holds for more dimensions. This fact follows in much the same way as single case we've discussed, but the proof is tedious.

THEOREM 1.2.3. *For A_i as defined above,*

(1) $A_i|v\rangle = A_i\left(I - \sum_{k=1}^{i-1}|v_k\rangle\langle v_k|\right)|v\rangle$
(2) $\left\|\left(I - \sum_{k=1}^{i}|v_k\rangle\langle v_k|\right)|v\rangle\right\| < 1$ *unless $|v\rangle$ is perpendicular to each of the earlier $|v_k\rangle$*

Finally because the same logic can be done on the other side of the matrix, we have $A_i = \left(I - \sum_{k=1}^{i}|u_k\rangle\langle u_k|\right)A_i$. So we make a more general statement.

COROLLARY 1.2.4. *For A_i as defined above,*

$$A_i = \left(I - \sum_{k=1}^{i} |u_k\rangle\langle u_k|\right) A_i \left(I - \sum_{k=1}^{i} |v_k\rangle\langle v_k|\right)$$

We're ready to prove the theorems mentioned at the opening of this chapter. First we'll prove that the vectors $\{\langle u_i|\}$ are perpendicular, and that $\{|v_i\rangle\}$ are perpendicular. Specifically we argue that each new vector generated by the algorithm is perpendicular to all previous ones.

THEOREM 1.2.5. *Let A_i be as in Algorithm 2 with $A_i \neq 0$. If u_i, v_i maximize $\langle u|A_i|v\rangle$, then u_i is perpendicular to u_j and v_i is perpendicular to v_j for all $j \leqslant i$.*

PROOF. Suppose to the contrary that one or both of the vectors is not perpendicular, and that the maximum is M.

M must be positive. To see this: $\langle u|A_i|v\rangle$ must be non-zero for at least one pair of vectors, for otherwise $A_i = 0$. Consider any non-zero value, $N = \langle u|A_i|v\rangle$. If N is positive, then the maximimum $M > N > 0$. Otherwise $-N = \langle -u|A|v\rangle$ must be positive, which also implies $M > -N > 0$.

Using Corollary 1.2.4, we can write M as:

$$M = \langle u|A_i|v\rangle = \langle u| \left(I - \sum_{k=1}^{i} |u_k\rangle\langle u_k|\right) A_i \left(I - \sum_{k=1}^{i} |v_k\rangle\langle v_k|\right) |v\rangle = \langle u'|A_i|v'\rangle$$

where $\langle u'| = \langle u| \left(I - \sum_{k=1}^{i} |u_k\rangle\langle u_k|\right)$ and $|v'\rangle = \left(I - \sum_{k=1}^{i} |v_k\rangle\langle v_k|\right) |v\rangle$.

Because we assumed that either $\vec{u}$ or $\vec{v}$ are not perpendicular to all previous vectors, one of vectors u' or v' have magnitude less than 1, by Theorem 1.2.3, point (b).

We can now reach a contradiction, because the unit vectors $\frac{\langle u'|}{\|\langle u'|\|}$ and $\frac{|v'\rangle}{\||v'\rangle\|}$ do a better job of maximizing $\langle u|A|v\rangle$ than $\langle u_i|$ and $|v_i\rangle$. This is because:

$$\frac{\langle u'|}{\|\langle u'|\|} A_i \frac{|v'\rangle}{\||v'\rangle\|} = \frac{\langle u'|A_i|v'\rangle}{\|\langle u'|\| \, \||v'\rangle\|} = \frac{M}{\|\langle u'|\| \, \||v'\rangle\|} > M > 0$$

□

Finally we prove that Algorithm 2 terminates.

THEOREM 1.2.6. *If A is dimension $m \times n$, call $r := \min(m, n)$, then $A_r = 0$.*

Note: This doesn't preclude $A_{r'} = 0$ for some $r' < r$. If that happens, then the algorithm will terminate early.

PROOF. Suppose we've gone through r steps, then it must be that the set of perpendicular covectors $\{\langle u_i|\}$ span the codomain *or* the set of perpendicular vectors $\{|v_i\rangle\}$ span the domain, depending on whether $r = m$ or $r = n$. Without loss of generality, say that the domain is spanned. Then

$$\begin{aligned} A_r &= A_r \left(I - \textstyle\sum_{i=1}^{r'} |v_i\rangle\langle v_i|\right) \\ &= A0 \\ &= 0 \end{aligned}$$

□

The first equality follows from Theorem 1.2.3 and the second from Exercise 1.2.2.

2.2. Change of bases for outer product sums. Before moving on, we'll state a fact that we'll need later. This is a little out of place here, except that it shows a nice application of the concepts cover in this section.

We will see in Exercise 1.2.2 that for any basis $\{|v_i\rangle\}$, we have $I = \sum_i |v_i\rangle\langle v_i|$. And a natural extension of this is that if you had two sets of orthonormal vectors, $\{|v_i\rangle\}$ and $\{|v_i'\rangle\}$, spanning the same space, then $\sum_i |v_i\rangle\langle v_i| = \sum_i |v_i'\rangle\langle v_i'|$. This is true, but we'll an even more general statement. We will encounter in the next section sums of outer products, but they won't be necessarily symmetric. So we need that $\sum_i |u_i\rangle\langle v_i| = \sum_i |u_i'\rangle\langle v_i'|$. This only happens under special circumstances, but these happen to be the circumstances that arise when picking apart the SVD.

LEMMA 1.2.7. *For orthonomal sets of vectors* $\{\langle u_i|\}_{i=1}^n$, $\{|v_i\rangle\}_{i=1}^n$, $\{\langle u_i'|\}_{i=1}^n$, *and* $\{|v_i'\rangle\}_{i=1}^n$, *we have*

$$\sum_i |u_i\rangle\langle v_i| = \sum_i |u_i'\rangle\langle v_i'|$$

If the following conditions hold:

(1) $\operatorname{span}\{\langle u_i|\} = \operatorname{span}\{\langle u_i'|\}$ *and* $\operatorname{span}\{|v_i\rangle\} = \operatorname{span}\{|v_i'\rangle\}$.
(2) There exists a fixed linear map B *such that* $B|v_i\rangle = |u_i\rangle$ *and* $B|v_i'\rangle = |u_i'\rangle$.

For the second point, there will always exist some mapping from one orthonormal set to another (of the same cardinality). The key assumption here is that the B is the same between the pairs of sets.

PROOF. Call $M = \sum_i |u_i\rangle\langle v_i|$ and $M' = \sum_i |u_i'\rangle\langle v'|$. To prove these matrices are equal, we show that they act the same on every vector. We chose some arbitrary vector $|v\rangle$, and show that $M|v\rangle = M'|v\rangle$.

By Theorem 9.5.2, we can write $|v\rangle = |v_\parallel\rangle + |v_\perp\rangle$ with W in the Theorem set as $\operatorname{span}\{|v_i\rangle\} = \operatorname{span}\{|v_i'\rangle\}$. With this construction $\langle v_i|v_\perp\rangle = 0$ for each $\langle v_i|$, so $M|v_\perp\rangle = M'|v_\perp\rangle = 0$.

Because $|v_\parallel\rangle \in \operatorname{span}\{|v_i\rangle\}$, $|v_\parallel\rangle = \sum_i a_i|v_i\rangle$.

$$\begin{aligned} M|v\rangle &= M|v_\parallel\rangle + M|v_\perp\rangle \\ &= M|v_\parallel\rangle \\ &= \left(\textstyle\sum_i |u_i\rangle\langle v_i|\right)\left(\sum_j a_j|v_j\rangle\right) \\ &= \textstyle\sum_i |u_i\rangle a_i \\ &= \textstyle\sum_i a_i B|v_i\rangle \\ &= B\left(\sum_i a_i|v_i\rangle\right) \\ &= B|v_\parallel\rangle \end{aligned}$$

The same calculation with primes on all the M, u, and v variables gives that $M'|v\rangle$ also equals $B|v_\parallel\rangle$. □

EXERCISES

1.2.1 Given unit vectors $\vec{u}$ and $\vec{v}$. Prove that if $\langle u|v\rangle \neq 0$, then $\|(I - |u\rangle\langle u|)\,|v\rangle\| < 1$. (Hint: For any vector $\vec{v}$, $\|\vec{v}\| < 1$ if and only $\langle v|v\rangle < 1$.)

1.2.2 Given a basis $\{\vec{v}_i\}_{i=1}^n$ of $\mathbb{R}^n$, prove that $I = \sum_{i=1}^n |v_i\rangle\langle v_i|$. That is, show for any $\vec{v}$, $\left(\sum_i |v_i\rangle\langle v_i|\right)|v\rangle = |v\rangle$.

1.2.3 Given a matrix with a SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, let $\{A_i\}$ be the matrices given by Algorithm 2. Lemma 1.2.1 implies that $A_{i-1}|v_i\rangle = \lambda_i |u_i\rangle$. Show that $A|v_i\rangle = \lambda_i |u_i\rangle$.

1.2.4 Given a matrix with a SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, show that $A = \sum_i A|v_i\rangle\langle v_i|$. (For fun, prove this two different ways.)

3. Almost Uniqueness

An annoying fact about the SVD is that it's not unique. But it is *almost* unique. In this section, we see that there are limitations.

Let's review briefly. Given a matrix A, we can find a SVD, $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$. The λ_i do not need to be distinct. For the sake of this section, we write this as

$$A = \sum_i \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|, \tag{2}$$

where $\lambda_1 > \lambda_2 > \cdots$. By indexing the vectors/covectors with a second index j, we can make sure that all our λ_i are unique.

Example

Example

Suppose we had an SVD:

$$A = 2|u_1\rangle\langle v_1| + 2|u_2\rangle\langle v_2| + 2|u_3\rangle\langle v_3| + |u_4\rangle\langle v_4| + |u_5\rangle\langle v_5|$$

To rewrite this in the language of Equation (2), we would write:

$$A = 2 \cdot (|u_{11}\rangle\langle v_{11}| + |u_{12}\rangle\langle v_{12}| + |u_{13}\rangle\langle v_{13}|) + 1 \cdot (|u_{21}\rangle\langle v_{21}| + |u_{22}\rangle\langle v_{22}|)$$

So that $\lambda_1 = 2$ and $\lambda_2 = 1$. $|u_1\rangle$ gets renamed to $|u_{11}\rangle$, $\langle v_4|$ gets remapped to $\langle v_{21}|$, etc.

If the SVD was built with our algorithm from the previous section, then $\langle u_{11}|A|v_{11}\rangle = \lambda_1$ maximizes $\langle u|A|v\rangle$. Of course because $\langle u_{1j}|A|v_{1j}\rangle = \lambda_1$ for any $1 \leqslant j \leqslant n_i$, the algorithm may just have chosen $v_{1,2}$ or $v_{1,3}$ first, so any reordering of these could have been produced. In fact we don't need to be restricted to these exact vectors; for *any* unit vector $|v\rangle \in \operatorname{span}\{|v_{i*}\rangle\}$, we can find a $\langle u|$ (given by Lemma 1.2.1) for which $\langle u|A|v\rangle = \lambda_1$. It also turns out that any $|v\rangle \notin \operatorname{span}\{|v_{i*}\rangle\}$, there is *not* a $\langle u|$ for which $\langle u|A|v\rangle = \lambda_1$. These two statements are proven in Corollary 1.3.2.

Together these show that if you follow the algorithm from the previous section of repeatedly finding the maximizing vectors, the only variation you can get is a change of basis in $|v_{i*}\rangle$. (If $n_i = 1$, then this means that we can only vary $\lambda_1|u\rangle\langle v| = \lambda_1|-u\rangle\langle -v|$.) Actually we've only shown this for for the largest $\lambda = \lambda_1$, but by subtracting off these components and applying the same argument to the resulting difference with a new largest $\lambda = \lambda_2$, we can apply the same argument; this is a trick we'll use repeatedly.

So running the algorithm won't give you the exact same answer each time you do it. It may vary, but only in choosing different a different basis for each $\{|v_{i*}\rangle\}$. But is it possible to use some other algorithm to come up with a SVD? And maybe if you did, you'd get a much different SVD? We show in this section that no matter

how you come up with an SVD it must be the same as one of the SVDs created by the algorithm. That is, it can only vary by a change of basis for each spanning space span $\{\langle u_{i*}|\}$ and span $\{|v_{i*}\rangle\}$.

THEOREM 1.3.1. *Given an SVD of a matrix, $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, then for any $\vec{u}$, $\vec{v}$, $\langle u|A|v\rangle \leqslant \lambda_1$.*

PROOF. Using Lemma 1.2.1, we get that

$$\max_{\|u\|=\|v\|=1} \langle u|A|v\rangle = \max_{\|v\|=1} \frac{\langle v|A^T}{\lambda} A|v\rangle = \max_{\|v\|=1} \frac{\langle v|A^T A|v\rangle}{\|A|v\rangle\|}$$

For any vector $\vec{w}$, $\|\vec{w}\| = \sqrt{\vec{w}^T\vec{w}}$. So,

$$\max_{\|v\|=1} \frac{\langle v|A^T A|v\rangle}{\|A|v\rangle\|} = \max_{\|v\|=1} \frac{\langle v|A^T A|v\rangle}{\sqrt{\langle v|A^T A|v\rangle}} = \max_{\|v\|=1} \sqrt{\langle v|A^T A|v\rangle}$$

A common trick is to notice that the square root of a value is maximal exactly when the value itself is maximal. So that the same $|v\rangle$ that maximizes $\langle u|A|v\rangle$ also maximizes $\langle v|A^T A|v\rangle$.

So to maximize $\langle v|A^T A|v\rangle$, we use the fact that $A^T A = \sum_i \lambda_i^2 |v_i\rangle\langle v_i|$ (Lemma 1.1.2). If $\{\vec{v}_i\}$ do not span the domain (with dimension = n), then we pad it with additional basis vectors, and $\lambda_i = 0$. $A^T A = \sum_{i=1}^n \lambda_i^2 |v_i\rangle\langle v_i|$. Now $\{\vec{v}_i\}_{i=1}^n$ is a basis so that any unit vector $|v\rangle = \sum_{i=1}^n a_i |v_i\rangle$, with $\sum_{i=1}^n a_i^2 = 1$.

$$\begin{aligned} \langle v|A^T A|v\rangle &= \langle v| \left(\textstyle\sum_i \lambda_i^2 |v_i\rangle\langle v_i|\right) |v\rangle \\ &= \left(\textstyle\sum_i a_i \langle v_i|\right) \left(\textstyle\sum_i \lambda_i^2 |v_i\rangle\langle v_i|\right) \left(\textstyle\sum_i a_i |v_i\rangle\right) \\ &= \textstyle\sum_i a_i^2 \lambda_i^2 \leqslant \sum_i a_i^2 \lambda_1^2 = \lambda_1^2 \end{aligned} \tag{3}$$

The inequality on the last line follows because λ_1 is larger than all the other λ_i.

Putting it all together, we have:

$$\langle u|A|v\rangle \leqslant \max_{\|u\|=\|v\|=1} \langle u|A|v\rangle = \max_{\|v\|=1} \sqrt{\langle v|A^T A|v\rangle} \leqslant \sqrt{\lambda_1^2} = \lambda_1$$

□

Notice, that by eliminating the $\langle u|$, the problem of maximizing became easier. After finding $|v\rangle$, we could use Lemma 1.2.1 to find $\langle u|$. But by symmetry, we could have instead found the $\langle u|$ that maximizes $\langle u|AA^T|u\rangle$.

While the theorem gives an upper bound of λ_1, note that this bound is achievable, namely with $\vec{u}_1$ and $\vec{v}_1$. But we can be a little more specific:

COROLLARY 1.3.2. *Given a matrix with SVD, $A = \sum_i \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$ (with $\lambda_1 > \lambda_2 > \cdots$) and a unit vector $|v\rangle$, $\max_{\|u\|=1}\langle u|A|v\rangle = \lambda_1$ if and only if $|v\rangle \in$ span $\{|v_{i*}\rangle\}$.*

We skip some of the details to this proof. It amounts to observing that the equality at the end of Eq. (3) is strict unless all the non-zero terms have λ_i coefficients equal to λ_1.

PROOF. Redo the work of Lemma 1.1.2, but this time with the SVD written as $A = \sum_i \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$ to get

$$A^T A = \sum_i \lambda_i^2 \sum_{j=1}^{n_i} |v_{ij}\rangle\langle v_{ij}|$$

Write $|v\rangle = \sum a_{ij}|v_{ij}\rangle$. Because $|v\rangle$ is a unit vector, $\sum a_{ij}^2 = 1$. By the same steps logic as Eq. (3), we get that

$$\langle v|A^T A|v\rangle = \sum_i a_i^2 \lambda_i^2$$

where $a_i^2 = \sum_j a_{ij}^2$. Again we have that the coefficients must satisfy $\sum_i a_i^2 = 1$, but this time the coefficients are distinct, so it's clear that this equals λ_1 if and only if $a_1 = 1$ and all others are 0. But this happens if and only if $|v\rangle \in \text{span}\{|v_{1*}\rangle\}$. □

Some words should be said about Theorem 1.3.1 and Corollary 1.3.2. We'll see in a bit that this is used to prove the almost-uniqueness of the SVD. In many ways, this theorem is the keystone of the entire chapter; it connects all the concepts. We needed to first prove the existence of an SVD to prove it. But we now see a motivation for Algorithm 2. Until now, the singular values seemed like an artifact of the specific SVD representation. But now we see that the first singular value λ_1 is something fundamental about the matrix - it's an upper bound that doesn't depend on the SVD representation. (By repeated subtraction, all singular values are representation-independent.) So it make sense that we should try to find and use these singular values. Looking ahead, the next chapter shows how to calculate the vectors $|v_i\rangle$. That calculation relies entirely on the trick (first encountered in Theorem 1.3.1) of canceling the $\langle u_i|$ by taking $A^T A$.

Theorem 1.3.1 and its corollary are also perhaps the most surprising. It seems sort of like magic. Maybe you're along for the ride when I said that we'll take $A^T A$ to cancel out the $|u_i\rangle$, but it seems like luck when the you end up with $\sum_i a_i^2$, which just happened to be 1, by nature of looking at unit vectors. Maybe there's some magic here. But at the same time, matrices are sometimes thought of as quadratic operators. For a vector $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ and an $n \times n$ matrix M, $\vec{x}^T M \vec{x}$ has every power-2 term of x with coefficients derived from M. (Check this.) Given this quadratic nature, maybe it's less surprising that it'd take vector components to their magnitude.

Imagine that your matrix A is a diagonal matrix. This is already an SVD with $D = A$ and $U = V = I$. Look at how obvious the logic here is now. With some $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, $\langle x|D|x\rangle = \sum_i \lambda_i x_i^2$. If you wanted the first singular value, λ_1, the largest value, then you need to do something to get A_{11}. Of course $A_{11} = \langle e_1|A|e_1\rangle$ would do the trick. When U and V aren't the identity then you should do $\langle e_1|U^T$ and $V|e_1\rangle$ to cancel. The existence of the SVD just says that aside from these little twists, U and V^T, all matrices are just these canonical quadratic forms.

With that digression behind us, we can now actually prove almost-uniqueness. All the hard work has been done already. This theorem brings together lots of ideas, so take your time with it.

THEOREM 1.3.3. *Given two SVDs of a matrix* $A = \sum_i \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}| = \sum_i \lambda'_i \sum_{j=1}^{n'_i} |u'_{ij}\rangle\langle v'_{ij}|$*, then for each* i*:*

(1) $\lambda_i = \lambda'_i$
(2) $n_i = n'_i$
(3) $\operatorname{span}\{\vec{u}_{i*}\} = \operatorname{span}\{\vec{u}'_{i*}\}$
(4) $\operatorname{span}\{\vec{v}_{i*}\} = \operatorname{span}\{\vec{v}'_{i*}\}$

PROOF. We first prove the four statements for $i = 1$.

Theorem 1.3.1 tells us that the largest value of $\langle u|A|v\rangle$ is the largest singular value for any SVD. Because these SVDs represent the same matrix, A, it follows that these two singular values are equal, $\lambda_1 = \lambda'_1$. So (1) is true.

Because $\langle u_{1j}|A|v_{1j}\rangle = \lambda_1 = \lambda'_1$ for each j, Corollary 1.3.2 tells us that $\langle u_{1j}| \in \operatorname{span}\{\langle u'_{1*}|\}$ and $|v_{1j}\rangle \in \operatorname{span}\{|v'_{1*}\rangle\}$. Since spans are closed under linear combinations, it follows also that $\operatorname{span}\{\langle u_{1*}|\} \subset \operatorname{span}\{\langle u'_{1*}|\}$ and $\operatorname{span}\{|v_{1*}\rangle\} \subset \operatorname{span}\{|v'_{1*}\rangle\}$. Applying the same argument in the other direction, we get the subset going the other direction, so that by double inclusion, we can conclude that $\operatorname{span}\{\langle u_{1*}|\} = \operatorname{span}\{\langle u'_{1*}|\}$ and $\operatorname{span}\{|v_{1*}\rangle\} = \operatorname{span}\{|v'_{1i*}\rangle\}$. This proves (3) and (4).

Because $\{|v_{1*}\rangle\}$ are orthonormal, the dimension of their span is the number of elements in the set, n_1. The span of $\{|v'_{1*}\rangle\}$ is the same space, so the same dimension. But this time the dimension is the number of elements in this orthonormal set, n'_1. It follows that $n_1 = n'_1$, which proves (2).

We've only proven this for the first singular value. This is because Theorem 1.3.1 and Corollary 1.3.2 are only proven for the maximum singular value. We want to use the usual trick of subtracting that signal out and repeating the argument on the resulting matrix, repeatedly. Usually we gloss over this detail, but here requires a little more care. We subtract $A - \lambda_1 \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$. On the representation $\sum_{i=1}^{n} \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$, this just removes the first summand.

$$A - \lambda_1 \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}| = \sum_{i=2}^{n} \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$$

However for the other representation, we have something messier.

$$A - \lambda_1 \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}| = \lambda_1 \left(\sum_{j=1}^{n_j} |u'_{ij}\rangle\langle v'_{ij}| - \sum_{j=1}^{n_j} |u_{ij}\rangle\langle v_{ij}| \right) + \sum_{i=2}^{n} \lambda'_i \sum_{j=1}^{n'_i} |u'_{ij}\rangle\langle v'_{ij}|$$

In order to repeat the above argument, we need to reduce this to an SVD again. We use Lemma 1.2.7 with $B = \frac{1}{\lambda_1} A$ to conclude that $\sum_{j=1}^{n_j} |u'_{ij}\rangle\langle v'_{ij}| = \sum_{j=1}^{n_j} |u_{ij}\rangle\langle v_{ij}|$, which cancels out the λ_1 terms from the above, resulting in:

$$A - \lambda_1 \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}| = \sum_{i=2}^{n} \lambda'_i \sum_{j=1}^{n'_i} |u'_{ij}\rangle\langle v'_{ij}|$$

Now we have the two SVDs of $A - \lambda_1 \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$, $\sum_{i=2}^{n} \lambda'_i \sum_{j=1}^{n'_i} |u'_{ij}\rangle\langle v'_{ij}|$ and $\sum_{i=2}^{n} \lambda_i \sum_{j=1}^{n_i} |u_{ij}\rangle\langle v_{ij}|$. So we can repeat the above arguments on this. Subtract, and repeat again, and so on.

□

This tells you is that no matter how you obtain the SVD, it will be in basically the same form. It will have the same singular values with the same multiplicity, and the only difference is that there may be a different basis chosen for the vectors.

This is a lot of work to show almost-uniqueness, but by now we should start to have some intuition for what the SVD is. Firstly, the singular values are something fundamental about the matrix, and not something that depends on the SVD representation. Secondly, our algorithm from the previous section is a rational thing to try. The largest singular value defines a space where these values can occur, but then after you subtract out that dimension, you get a big jump down to the next singular value.

Exercises

1.3.1 With an SVD given by Eq. (2), show that if $|v\rangle \in \text{span}\{|v_{i*}\rangle\}$, then $A|v\rangle \in \text{span}\{|u_{i*}\rangle\}$.

1.3.2 Given a matrix with SVD $A = \sum_i \lambda_i |v_i\rangle\langle v_i|$, show that for any $n \in \mathbb{N}$, $A^n = \sum_i \lambda_i^n |v_i\rangle\langle v_i|$.

1.3.3 Given a matrix with a SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, show that $A = \left(\sum_i |u_i\rangle\langle u_i|\right) A \left(\sum_j |v_j\rangle\langle v_j|\right)$ by distributing the sums and arguing that $\langle u_i|A|v_j\rangle$ equals λ_i if $i = j$ and 0 otherwise. How could you show this equality without distributing?

1.3.4 Given an invertible square matrix A with minimum singular λ_n, prove that for any unit vectors $\vec{u}$ and $\vec{v}$, $\langle u|A|v\rangle \geqslant \lambda_n$.

1.3.5 Given a matrix A, prove that for any value t between the smallest and largest singular value $\lambda_n \leqslant t \leqslant \lambda_1$, there are some unit vectors $\langle u|$ and $|v\rangle$ for which $\langle u|A|v\rangle = t$.

4. How to Calculate

So far we haven't said how to actually calculate an SVD. Algorithm 2 says to repeatedly find maximal vectors, but we haven't said how to do this. In this section we show that this calculation amounts to calculating certain eigenvalues and eigenvectors. In practice, you could calculate the SVD using a computer program. (See Section 1.) Yet knowing how to calculate manually will advance our understanding of the SVD.

Before going on, we need a small fact about calculus. I'll state a lemma here without proof. Calculus is outside the scope of this book, but I provide some discussion in Appendix X.

Lemma 1.4.1. *For any square matrix B,*

$$\frac{\partial}{\partial|v\rangle}\langle v|B|v\rangle = 2B|v\rangle$$

The left-hand side of this equation is a vector whose entries equal $\frac{\partial\langle v|B|v\rangle}{\partial v_i}$. This lemma seems simple. (It's analogous to $d(bx^2) = 2x$.). Indeed, it could be demonstrated with basic calculus, but the calculation is slightly more difficult than you may guess.

We saw in the last section that we could find $|v_1\rangle$ in $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$ as the unit vector $|v\rangle$ that maximizes $\langle v|A^TA|v\rangle$. To calculate this max, we can do some calculus, but we have to be careful to restrict to unit vectors ($\langle v|v\rangle = 1$). To enforce this, we use Lagrange multipliers. (We call the Lagrange multiplier ε instead of the usual λ to avoid confusion with singular values.)

$$\langle v|A^TA|v\rangle - \lambda\langle v|v\rangle = \langle v|\left(A^TA - \varepsilon I\right)|v\rangle$$

By taking the derivative using Lemma 1.4.1 and setting to zero, we get

$$\left(A^TA - \varepsilon I\right)|v_1\rangle = 0$$

That is, $|v\rangle_1$ is an eigenvector of A^TA. Though this doesn't tell us which eigenvector, we can check all eigenvectors until we get a maximum. Similarly $|u_1\rangle$ is an eigenvector of AA^T.

If the eigenvalue corresponding to $|v_1\rangle$ is ε_1, then $\langle v_1|A^TA|v_1\rangle = \varepsilon_1$. From the discussion in the proof of Theorem 1.3.1,

$$\lambda_1 = \langle u_1|A|v_1\rangle = \sqrt{\langle v_1|A^TA|v_1\rangle} = \sqrt{\varepsilon_1}$$

It's not surprising that these vectors are eigenvectors of A^TA, because recall that

$$A^TA = \sum_i \lambda_i^2 |v_i\rangle\langle v_i|.$$

It's clear to see that $A^TA|v_i\rangle = \lambda_i^2|v_i\rangle$, which makes it an eigenvector.

Yet another way to see this to write $A = UDV^T$, so that $A^TA = VD^2V^T$. Because V^T rotates the domain. You can see this as a rotation, a stretch or squeeze along the axes, then a rotation back (V). V^T maps each of the vectors $\vec{v}_i$ to the axes, and V maps the axes back to the vectors $\vec{v}_i$. It clear to see then that VD^2V^T maps each $\vec{v}_i$ to some $\varepsilon_i\vec{v}_i$.

Exercises

1.4.1 Using the eigenvector technique presented in this section, calculate the SVD of $\begin{pmatrix} 2 & 6 \\ 1 & 1 \end{pmatrix}$.

1.4.2 Prove that the eigenvalues of A^TA form the same multiset as the eigenvalues of AA^T.

1.4.3 Given a matrix A, let $\{A_i\}$ be the matrices found in Algorithm 2. Show that for any $j > i$ an eigenvector corresponding to a non-zero eigenvalue of A_j is also an eigenvector of A_i.

1.4.4 Given a symmetric matrix A with distinct singular values, prove that every SVD has the form $\sum_i \lambda_i |v_i\rangle\langle v_i|$.

5. Quick Conclusions

Though it took a lot of work to build some understanding of the SVD, we will use it throughout the book. Next chapter, we can discuss a number of applications of the SVD. For now, I will whet your appetite with some immediate conclusions.

5.1. Rank-Nullity Theorem.

THEOREM 1.5.1. *If A is an $m \times n$-dimensional matrix, then* $\text{Dim}\left(\text{Range}(A)\right) + \text{Dim}\left(\text{NullSpace}(A)\right) = n$.

PROOF. Choose some SVD of $A = \sum_{i=1}^{r} |u_i\rangle\langle v_i|$. Expand $\{|v_i\rangle\}_{i=1}^{r}$ to a basis of the domain by adding vectors $|v_{r+1}\rangle, |v_{r+2}\rangle, ..., |v_n\rangle$. $\text{Range}(A) = \text{Span}\left\{\langle u_i|\right\}_{i=1}^{r}$. And $\text{NullSpace}(A) = \text{Span}\left\{|v_i\rangle\right\}_{i=r+1}^{n}$. (See Exercises **??** and **??**.) So the equality follows. □

5.2. Determinant.

THEOREM 1.5.2. *Let A be an $m \times m$-dimensional matrix with m (not-necessarily-distinct) singular values, $\lambda_1, ..., \lambda_m$. Then $|A| = \prod_{i=1}^{n} \lambda_i$.*

PROOF. Choose an SVD, $A = UDV^T$. Because determinant distributes over matrix multiplication, $|A| = |U| \cdot |D| \cdot |V^T|$. Because they're orthogonal, $|U| = |V^T| = 1$. So $|A| = |D|$. D is a diagonal matrix with entries $\lambda_1, ..., \lambda_m$. The usual formula for determinants on a diagonal matrix yields a product on the diagonal terms. □

There's another way to understand this. The orthogonal matrices U and V^T are isometries. This means that they preserve distances, and importantly, volumes. The determinant can be understood as the volume of the image of a unit cube. With a UDV^T transformation, we see the cube gets rotated (V^T), stretched or squashed (D) in a way that maps unit cubes to cubes of volume $|D|$, then rotates again (U). The rotations don't affect the volume, so the unit cube gets mapped to some cube of volume $|D|$.

5.3. Misc.

THEOREM 1.5.3. *Given a matrix with SVD, $X = \sum_i \lambda_i |u_i\rangle\langle v_i|$, the span of the columns of X is the span of $\{|u_i\rangle\}_i$, and the span of the rows of X is the span of $\{|v_i\rangle\}_i$.*

PROOF. TODO □

EXERCISES

1.5.1 In some contexts, a matrix norm is defined as

$$\|A\| = \max_{|v\rangle \neq 0} \frac{\|A|v\rangle\|}{\||v\rangle\|}.$$

Given an SVD, $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, show that $\|A\| = \lambda_1$.

1.5.2 Call $\sigma_1(A)$ the largest singular value of A. Prove that $\sigma_1(A + B) \leqslant \sigma_1(A) + \sigma_1(B)$.

1.5.3 Let $\mathcal{F}$ be the vector space spanned by

$$\{|\cos(x)\rangle, |\cos(2x)\rangle, |\cos(3x)\rangle, |\sin(x)\rangle, |\sin(2x)\rangle, |\sin(3x)\rangle\}.$$

Show that the determinant D is linear on this space. Define the inner product on this space as $\langle u(x)|v(x)\rangle = \frac{1}{\pi}\int_{x=-\pi}^{\pi} u(x)v(x)\,\mathrm{d}\,x$. Show that $|D| = 36$.

CHAPTER 2

Direct Applications of SVD

1. Computer Programming

For the applications we discuss, the work will be done with computers. Though we won't focus on programming in this book, there will be exercises with computer work.

If you have a language of choice, feel free to use that. If not, I recommend Python. For a quick start, I recommend using Google Colab to make small Python notebooks.

To do an SVD in Python is easy:

```
import numpy as np

A = ...  # Make matrix as a numpy array or a list of lists
U, d, VT = np.linalg.svd(A)
D = np.diag(d)
```

Some exercises will involve a deeper dive on the programming side. For these problems, you may need to look up and learn about libraries not covered in this book. I label these problems with a $\diamondsuit$.

Exercises

For these exercises, we use

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 3 & 5 \\ 3 & 6 & 9 \end{pmatrix}$$

2.1.1 Use the computer to calculate the SVD of A.

2.1.2 Use the computer to multiply U, D, and V^T. Verify that it's the same as A.

2.1.3 Verify that U and V^T are orthogonal.

2.1.4 Write this in bra-ket notation.

2.1.5 Use the computer to multiply out the summands in the bra-ket notation. Notice that these are not rational number despite the sum being whole.

2.1.6 Use the bra-ket notation to find the null space.

2. Collaborative Filtering

Famously Netflix created a prize several years ago, which paid anyone who could beat their recommendation algorithm by more than ten percent. The winner of the prize used collaborative filtering.

Netflix streams movies to a large number of users. Most of those streams originate from Netflix's recommendations. One way to think of recommendation is

a map from users to movies. We might say that this is a linear map, so that this is just a giant matrix.[1] But before we're ready to talk about a recommendation matrix, let's instead start with a "watch matrix" W. $W_{ij} = 1$ if user i watched movie j, and 0 otherwise.

This isn't very useful yet, but let's try to understand this better. When trying to understand a matrix, we could try taking the SVD.

$$W = \sum_i \lambda_i |m_i\rangle\langle u_i|$$

This decomposition has millions of summands, and not all of them are equally important. The vectors are always unit vectors, but the λ_i will typically vary quite a bit. Some will be very large, while others will be relatively small.

An observation is that if you had a very tiny λ_n and you omitted the summand from the SVD (call it A'), you'd get a matrix that is very similar. This is similar in the sense that for all $|v\rangle$, $A|v\rangle \approx A'|v\rangle$.

But who cares? Well, this gives a good way to smooth out a linear map. This is sometimes called a *low-rank approximation*. Similar to fitting a polynomial to a scatterplot, a low-rank approximation can help capture the signal through the noise. And experience has shown that this works quite well.

But why choose this particular low-rank matrix? Well we'd want $\|A|v\rangle - A'|v\rangle\|$ to be small as often as possible. Let's again use the SVD $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, and for the sake of this example. What's the best rank-1 matrix A' to approximate this? It turns out that $\|A|v\rangle - A'|v\rangle\|$ is minimal for an "average" $|v\rangle$ precisely when A' is the first summand of the SVD of A.

THEOREM 2.2.1. *Given a matrix with SVD, $A = \sum_i \lambda_i |u_i\rangle\langle v_i|$, then*

$$\min_{A' \text{ rank-1}} E_{|v\rangle}\left(\|A|v\rangle - A'|v\rangle\|\right)$$

is achieved when $A' = \lambda_1 |u_1\rangle\langle v_1|$.

Proving Theorem 2.2.1 would need a better definition of "average" and would be too much of a diversion to prove here, but we do leave a simplified version as Exercise 2.2.1. Applying this theorem repeatedly we see that the best low-rank approximation we can get for a matrix is to just truncate the SVD after a number of the largest summands. Very cool!

Let's talk about Netflix again. W is a large $m \times n$, where m is the number of users and n is the number of movies. The rank is probably close to n, which is probably over ten thousand. But what if we low-rank approximated W with a 100-dimensional:

$$W' = \sum_{i=1}^{100} \lambda_i |m_i\rangle\langle u_i|$$

Maybe this captures signal through the noise, but what practically does it give us? You'll remember from Exercise 2.1.5 that the summands of the SVD usually aren't rational, even if their sum is. W mapped users to movies, but it wasn't very useful, because it only told us what movies a user already watched. W' maps users to movies, but it gives an irrational value for every movie, which gives us a

[1] The shear size of the matrix presents technical challenges that are beyond the scope of this book.

total ranking of movies. Now we can tell you not just what a user watched, but what they're likely to have watched. By recommending movies that the user hasn't watched, but score high for, they're likely to *want* to watch.

This low-rank approximation is called **collaborative filtering**. The j-th row in the resulting matrix represents an interest profile for each user, with one entry for every movie. This translates to recommendations by simply looking for those movies with the highest score. There's a lot of conversation in this section, but the algorithm is really nothing more than a truncated SVD.

Some form of Collaborative Filter won the Netflix prize. Though conceptually simple when stood up against some of today's complex models, it performed quite well. I should point out that they trained this not on W, but on S. Where S_{ij} represents the star rating that user j gave to movie i. (That is, users manually rated movies from 1 star to 5 stars.) If you were working at Netflix today and tried to build this, the choice between W and S would be a modeling decision you'd have to make.

2.1. Latent Dimensions and Embeddings. As it happens, SVD is very good at getting meaningful signal. The summands of the SVD end up being meaningful genres. You might get a dimension that represents action movies, which scores high for action movies and low for non-action movies. Or you might get a Kurt Russell dimension which is near 1 for all Kurt Russell movies and near 0 for all others. The values $\langle m|m_i\rangle$ end up being a value which describes how well the the movie $\langle m|$ fits into the i-th genre, and the values $\langle u_i|u\rangle$ describe how much the user $|u\rangle$ is interested in the i-th genre.

Continuing our example, say that our low-rank matrix has two genres action and Kurt Russell (KR), and say that both $\lambda = 1$. So that

$$W' = |m_{action}\rangle\langle u_{action}| + |m_{KR}\rangle\langle u_{KR}|$$

Let's say our movie library includes Overboard ($\mathcal{O}$), Escape From New York ($\mathcal{E}$), Frankenstein ($\mathcal{F}$), and Rambo ($\mathcal{R}$). (Sorry to anyone that hates movies.) Let's also assume the following genre assignments:

$$\begin{array}{ll} \langle m_{\mathcal{O}}|m_{action}\rangle = 0 & \langle m_{\mathcal{O}}|m_{KR}\rangle = 1 \\ \langle m_{\mathcal{E}}|m_{action}\rangle = 1 & \langle m_{\mathcal{E}}|m_{KR}\rangle = 1 \\ \langle m_{\mathcal{F}}|m_{action}\rangle = 0 & \langle m_{\mathcal{F}}|m_{KR}\rangle = 0 \\ \langle m_{\mathcal{R}}|m_{action}\rangle = 1 & \langle m_{\mathcal{R}}|m_{KR}\rangle = 0 \end{array}$$

Now assume you're a user who is three times more interested in action movies than in Kurt Russell. That is:

$$\langle u_{action}|u\rangle = 3 \qquad \langle u_{KR}|u\rangle = 1$$

Then we see that something like Rambo may score pretty high for this user:

$$\langle m_{\mathcal{R}}|W'|u\rangle = \langle m_{\mathcal{R}}|m_{action}\rangle\langle u_{action}|u\rangle + \langle m_{\mathcal{R}}|m_{KR}\rangle\langle u_{KR}|u\rangle = 1\cdot 3 + 0\cdot 1 = 3$$

And though we'll see that the user has no interest in Frankenstein, $\langle m_{\mathcal{F}}|W'|u\rangle = 0$, they'll have some mild interest in Overboard, $\langle m_{\mathcal{O}}|W'|u\rangle = 1$. Yet, because these features are additive, this user will have the most interest in Escape From New York, a movie that is both an action movie and stars Kurt Russell, $\langle m_{\mathcal{E}}|W'|u\rangle = 4$.

Collaborative Filtering is also smart enough to figure out how two weak signal should be considered more highly than a single strong signal. For the sake of argument say that Death Proof ($\mathcal{D}$) is rated lower for Kurt Russell, because he's not the lead, $\langle m_{\mathcal{D}}|m_{KR}\rangle = 0.5$, and is rated lower for action, because it's a bit of a genre-bender, $\langle m_{\mathcal{D}}|m_{action}\rangle = 0.7$. We can do the calculation to find that $\langle m_{\mathcal{D}}|W'|u\rangle = 0.7 \cdot 3 + 0.5 \cdot 1 = 2.6$. So it should be ranked above Overboard but below Rambo, closer to Rambo.

We should pause to mention the importance of orthogonality here. You may object that Kurt Russell has done mostly action movies. Isn't a Kurt Russell fan just a specific type of action fan? Though this example isn't very realistic, in actuality the dimensions will be orthogonal. If a user watches Escape From New York, The Thing, and Big Trouble in Little China, Collaborative Filtering can figure out how much of this behavior is attributable to the user being a run-of-the-mill action fan, and how much should be attributed to a Kurt Russell-affinity. They may end up being 0.8 on the action axis, and 0.3 on the Kurt Russell fan axis. But that 0.3 won't be in a vacuum. The model can figure out that for that user Kurt Russell enhances an action movie by 0.3, so that an action movie with Kurt Russell will end up totaling 1.1. This 0.3 will apply to all Kurt Russell movies, so that if some other signal gives them a rom-com score of 0.35, then Overboard (a rom-com with Kurt Russell) will score 0.65.

The dimensions "action" and "KR" aren't decided ahead of time in a board room, but rather come out as the result of Collaborative Filtering. These are called **latent dimensions**, where "latent" means something like hidden or underlying. Collaborative Filtering basically "finds" these hidden dimensions by finding what genres should be projected too to find the most prominent dimensions. They may turn out to even surprise the Netflix data scientists by surfacing genres that they didn't know exist, like "90s irreverent comedies."

Our example isn't realistic, because in practice the numbers won't be rational, so 1 and 0 are unlikely. Rather what you'd see is that there may be a gradation. For the action dimension, maybe Escape From New York scores a 0.99, but Big Trouble in Little China (an action movie with fewer action scenes) may score a 0.82. Hateful Eight and Miracle might score a middling 0.67 and 0.40 respectively; these speak more to how likely they are to appeal to action-movie fans then anything else. And Overboard might score a low 0.11. (These numbers are all made up, of course.). For the Kurt Russell dimension, movies may be a little more black-and-white, but there will still be some gray. Movies where Kurt Russell is on the poster may score higher than when he has a smaller role. Some non-Kurt Russell movies might score high based on how much they resemble a Kurt Russell movie. Rather those movies will score based on how much they would appeal to a Kurt Russell fan. For example, Ladyhawke originally cast Kurt Russell as the lead, and would make sense in his career; so it might score 0.4 for KR despite not having Kurt Russell anywhere in it. The scores may even surprise the data scientists doing the calculation. They may see Overload score really high on the Kurt Russell dimension, only to realize that it stars Kurt Russell's son. Or they may see The Christmas Chronicles score lower than they expected; although this movie features Kurt Russell prominently, it's a break from the movies he's most well-known for.

When you browse Netflix, you'll see Netflix group movies by genre or category, like say "Irreverent 90s Comedies." Although we don't know how Netflix makes

these categories, it's likely they do something like: List movies that score high for a given latent dimension, then have a human curate these by adding or subtracting movies.

2.2. Embeddings. Embeddings is a term that's used a lot in ML. There's a precise definition for it, and it's rich with connotation. But for us, we'll just use this to mean a dimensionality reducer. For example if you take a user, their full set of data would be their row in W, what movies they watched and didn't watch. If there are m movies, then this is an m-dimensional vector. But as soon as we've decided that we're going to do Collaborative Filtering, we don't need that much data from a user. All we actual care about is the values $\langle u_i|u\rangle$ for each i representing a latent dimension. (In our above example there were 100 of these.)

The map

$$\vec{u} \mapsto (\langle u_1|u\rangle, \langle u_2|u\rangle, ..,)$$

is a dimensionality reducer, and so it fits our definition of an embedding.

It also happens to be a linear map, and it's exactly equal to U^T, where $W' = MDU^T$ is the matrix form of the SVD of W'. Another embedding is $\sqrt{D}U^T$, where $\sqrt{D}$ is a diagonal matrix with the terms the square roots of the terms on D.

We should be a little more careful with dimensions. Recall that U^T is an $n \times n$ matrix where n is the number of users. This isn't actually a dimensionality reducer, but we think of it like it is. This is because DU^T will zero out all but the first 100 entries. $\sqrt{D}U^T$ is an $m \times n$ matrix, but the all but the first 100 column will be zero. So actually if we replaced $\sqrt{D}$ with a diagonal matrix with the same entries but shaped $100 \times n$, we'd have a true dimension reduction. Let's go ahead and do that, so that all this is just a pesky caveat.

If we did the same thing for movies, the linear map $\sqrt{D}M^T$ takes movies vectors (columns in W') to a 100-dimensional vector space. We can think of this space as the same space as the image of $\sqrt{D}U^T$. If we take the embedding of a movie $|m\rangle$, $\sqrt{D}M^T|m\rangle$, and the embedding of a user $|u\rangle$, $\sqrt{D}U^T|u\rangle$, and take the inner product, we get $\langle m|M\sqrt{D}\sqrt{D}U^T|u\rangle = \langle m|W'|u\rangle$. So from these two embeddings, we can reconstruct the entries in W', by doing dot products. Recall that the dot product is the product of the norms times the cosine of the angle between them. In practice the embeddings will be comparable in norm, so the closeness of the movie/user vectors as measure by angle corresponds to how similar the the movie is to the user. In fact, similar movies will be close to each other as well, owing to the fact that users who like one movie will like the other. Similar users will also be close.

This is just a rearrangement of the same data, but the interpretation is helpful. We could leverage approximate nearest neighbor algorithms to produce good recommendations on extremely large datasets. As well the movie-to-movie similarity that this interpretation gives you, let's you make categories like, "Because you watched The Christmas Chronicles."

2.3. Averaging. There's a cool feature about Collaborative Filter that I want to mention. Call $B = \sqrt{D}U^T$, the embedding of users, and $A = \sqrt{D}M^T$, the embedding of movies, so that $W \approx W' = A^T B$. Let's say that you know the embeddings of movies, A, the singular values D, and the watch matrix W. From these can you get user embeddings B? Yes, some simple algebra yields:

$$\begin{aligned} W \approx A^T B \implies B &\approx \left(A^T\right)^{-1} W \\ &= \left(MD^{1/2}\right)^{-1} W \\ &= \left(D^{-1/2}M^T\right) W \\ &= D^{-1/2}D^{-1/2}D^{1/2}M^T W \\ &= D^{-1}AW \end{aligned}$$

Call $D^{-1}A$ the "modified embedding" (a non-standard term). This amounts to taking the embedding of each movie, and dividing the i-th element by λ_i. Now the above equation says that the embedding for a user is approximately the sum of the modified embeddings of movies that the user watched! This is hugely helpful when a user starts on the platform. After they watch a few movies, we could determine what their embedding should be, without have to refactor an enormous matrix.

There's some theoretical value here too. Let's say that all the top 100 singular values are close in value, call it λ. (The idea here is that because we're only keeping the largest singular values, they'll be near to each other.) Then we can say that $D \approx \lambda I$. (It's this author's experience that this assumption won't be close to true; however making the assumption will still make a useful model that makes good recommendations.). Under this assumption, we can drop the "modified." Then the user embedding is the sum of the movie embeddings across all the movies they watched, *AND* the movie embedding is the sum of the user embeddings across all the users who watched it. This fact gives a good intuition for what Collaborative Filtering actually does.

Quick note on notation here before we wrap: Breaking UDV^T into two bits $U\sqrt{D}\sqrt{D}V^T$, the bra-ket notation is a little trickier. To keep it being a sum, we need to put the i-th term of $\sqrt{D}V^T$ into the i-th coordinate by attaching the $|e_i\rangle$. So we write $\sqrt{D}V^T = \sum_i \sqrt{\lambda_i}|e_i\rangle\langle v_i|$ and $U\sqrt{D} = \sum_i \sqrt{\lambda_i}|u_i\rangle\langle e_i|$. Verify that these multiply to equal $\sum_i \lambda_i|u_i\rangle\langle v_i|$

Exercises

2.2.1 In this exercise we're gonna be proving a simplified version of Theorem 2.2.1; instead of proving this for an average vector, we prove it for an average basis vector. Given an SVD $A = \sum_i \lambda_i|u_i\rangle\langle v_i|$, with distinct singular values and the vectors $\{\langle v_i|\}$ spanning the domain, prove that $\frac{1}{n}\sum_j \|(A - \lambda|u\rangle\langle v|)\,|v_j\rangle\|$ ($\|\vec{u}\| = \|\vec{v}\| = 1$) is minimal when $\lambda = \lambda_1$ and $|u\rangle = |u_1\rangle$ and $\langle v| = \langle v_1|$. (Hint: Use $\|\vec{x}\| = \langle x|x\rangle$. Then use $\sum_i \langle v_i|v\rangle^2 = 1$ along with some other identities to reduce. Then use Corollary 1.3.2 along with some calculus to minimize.)

2.2.2 Suppose you are running a small movie-streaming service, and you have the following watch matrix:

	Squid Game	Tiger King	Nomadland	The Father	Mank	Hamilton	Borat	Palm
Abed	1	0	1	1	0	0	0	
Bayani	1	1	1	1	0	0	0	
Carol	0	1	1	1	1	0	0	
Diego	1	1	1	0	0	0	1	
Ellie	1	1	0	1	0	1	1	
Frank	1	1	0	0	0	1	0	

Using a computer, perform Collaborative Filtering using 1 latent dimension, 2 dimensions, 3, 4, and 5 dimension. For each of these dimensions, what movie is recommended to each user? (Do not recommend a

movie that the user has already watched.) Discuss these results. How do the recommendations change as you increase dimension? Do you agree with the recommendations? How many dimensions would you recommend using?

2.2.3 Prove that $\langle u_i|W^TW|u_j\rangle$ equals the number of movies that users i and j both watched.

2.2.4 Users sometimes provide star ratings (1-5 stars) for movies. An alternative approach would be to use these ratings instead of a watch matrix. What are some pros and cons of this alternative approach?

2.2.5 A user not watching a movie doesn't mean that they don't like the movie. How can you use user traffic to determine when a user isn't interested in a movie? How could you use this information in a Collaborative Filtering approach?

2.2.6 Suppose that a matrix A has all values $0 \leqslant A_{ij} \leqslant 1$. And let A' be a low-dimensional approximation obtained by taking a partial sum of the SVD. Prove that A' also has all values $0 \leqslant A'_{ij} \leqslant 1$.

2.2.7 Imagine that you're building a Netflix-like recommendation engine, and you've discovered that your engine doesn't work well for black users. To measure this effect, you modify the loss equation from Theorem 2.2.1:

$$\min_{A' \text{ rank-}r} E_{|v\rangle, \text{ black users}} \left(\| A|v\rangle - A'|v\rangle \| \right)$$

Explain how you might modify Collaborative Filtering to reduce this loss. Consider algorithm runtime, and how you'll score new users.

2.2.8 ♢ Download the MovieLens dataset from https://grouplens.org/datasets/movielens/. (Choose which size dataset you want to work with. For a greater challenge, do a larger set.) Perform Collaborative Filtering on the data. Test your model by filling in 5-stars for some of your favorite movies; what movies get recommended? (If you don't like movies, then try entering Escape from New York, Overboard, and Deathproof.) Bonus: Make software (or Jupyter notebook) that let's the user enter movies, and updates the top-10 recommendation list.

3. Principle Component Analysis

Let's say you're trying to build a credit profile on a set of users. We know that everyone gets a one-number credit score, but if you're a bank or an insurance company, you may find this insufficient. These same credit companies will sell you a packet of hundreds (or thousands) of variables for each of your customers. You can use these to predict loan defaults[2] or car accidents. You could even build an insurance-specific credit score for each of your customers. (Such a score would omit medical debt, by law.) This will be our motivating example for this section. Although the use of credit scores can be controversial, we'll focus only on the modeling challenge it presents.

A difficulty of working with hundreds of variables, is that modeling can be difficult. Experienced practicioners may agree with this immediately. Neural network architects may laugh at less than a thousand variables. And newer modelers may not see why this would be difficult; can't I just hit run on my regression? So I feel this claim needs some justification. I'll give one example of a problem: Let's take

[2]"Default" means failure to make payments owed.

for example a regression on a hundred variables. Likely some of these are correlated with each other. Linear regressions struggle with correlation, sometimes it will give a large coefficient to one of variables, other times to a correlated variable. The inclusion or exclusion of a few points can change which variable gets the coefficient. If you're not careful, a regular refresh can change your model a lot. Even if you only added a few points, you could greatly impact the credit score you give to some users. With low-variable models, it is easier to monitor and account for this.

So if you're a credit modeler, you may want to perform a dimensionality reduction, something we know about. The most canonical dimensionality reducer is the **Principal Component Analysis (PCA)**. We'll describe this soon, but first let's imagine that you're the first person to come up with a dimensionality reducer - never been done before. You have 1000-dimensions, but you're targeting 10. Your first approach might be to just take the first 10 variables on your list. Nice job! You've just built a dimensionality reducer! But it's not very good. There's not actually an *order* to variable, so why these ten and not some other ten? Well, okay, which variables would be the best candidates?

An example of a bad variable to use is number of bankruptcies in the last three years, because the vast majority of users have not had any bankruptcies in the last three years. And if you only have ten variables, you don't want to waste one on a variable that doesn't give you much information. On the other hand, a variable like total credit card debt will be a better variable because it gives you a big range of experiences that varies from customer to customer. There are lots of things that make variables "good" or "bad;" I'm choosing to focus on this specific criteria. I propose that if a variable has lots of variance, then it's a good variable to try. Without domain knowledge about what these variables are and how they will be used, this criteria seems as good as any.

Recall the defintion of variance: It's a measure of the spread of data points, with more points further away from the center being better. It can be calculated as:

$$\mathrm{Var}(X) = \mathrm{E}(X^2) - \mathrm{E}(X)^2 \tag{4}$$

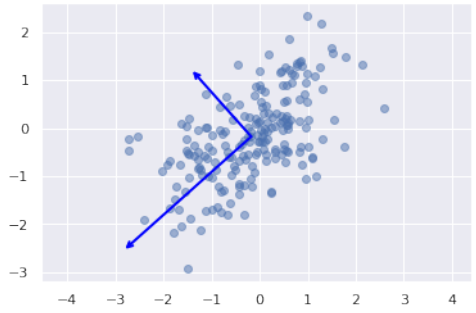
The E means expected value, or average. We'll ignore the second term; this move is justified later in the Normalization section. So we get:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i^2$$

We can ignore the $1/n$ also, because n is constant. If you have a matrix A where A_{ij} is the value of the i-th variable for the j-th user, then the values of the i-th variable is the vector $\langle e_i|A$. And the sum of the squares is $\langle e_i|AA^T|e_i\rangle$. Pretty cool; it's starting to look familiar.

With this proposal, you're trying to find the standard axes $\vec{e_i}$ that maximize $\langle e_i|AA^T|e_i\rangle$. But we know now that the standard axes are just one choice of bases; there's nothing really special about them. To be more general, let's consider any axis. That is find the unit vector that maximizes $\langle u|AA^T|u\rangle$. We know how to do this. We can perform the same algorithm that we use to find the $\{|u_i\rangle\}$ in the SVD $A = \sum_i \lambda_i |u_i\rangle\langle u_i|$. If we only want 10 axes, then we apply the algorithm for just

FIGURE 1. First and second principle components.



10 steps, which will be the $|u_i\rangle$ that correspond to the 10 largest singular values. These chosen axes are called the **principle components**.

What's nice about the SVD is that it builds orthogonality right in. This means that, like with Kurt Russell and action movies, the new axes will be independent. To visualize this, we think briefly about scatter plots. If we have n users and m features about them, we can view this as n points on an m-dimensional scatter plot. If two variables are highly correlated, then they'll fall mostly along a line. For example, number of credit cards and amount of credit card debt will be correlated, so they will live mostly along a line. The first principle component will pick up this axis. Any other principle component will have to be perpendicular to this. Maybe it'll represent something like, the amount of credit card debt above what's typical for the number of cards they have.

After you get the Principle Components, we project a user's data onto these axes. This projection map can be called $\sum_i |e_i\rangle\langle u_i|$. The $|e_i\rangle$ terms just say that we want the the i-th projection to go in the i-th coordinate.

3.1. Normalization. It's impossible to talk about PCA without discussing normalization. Let's say you have two variables, age and annual income. These things are likely correlated in your data, and you want to capture the best axis to capture these. But age only varies between users by a couple of decades, whereas income varies by tens of thousands of dollars. PCA will say that most of the variance lives with the variance on income. But this doesn't actually make sense because income is measured in dollars and age is measured in years. Our answer shouldn't change if we decide to measure income in Euros and age in minutes. We need something unitless that captures some general sense of spread. As well, the range of incomes will be different than the range of ages. Why should a model capture that people don't get credit cards until the age of 18? We need something that also centers.

There are a few different ways to recenter and scale. More generally such techniques are called regularization. We'll take about a specific technique, call *normalization*.[3] This is done before we apply any algorithm. For each variable X, with points x_i, we make new variables:

[3]Some readers will recognize the equation as the z-score used for normal distribution, from which normalization gets its name.

$$x'_i := \frac{x_i - \mathrm{E}(X)}{\sqrt{\mathrm{Var}(X)}}$$

Say that the x'_i are points of the new variable X'. Well mention some nice effects of normalization. Subtracting the $\mathrm{E}(X)$ guarantees that $\mathrm{E}(X') = 0$. This is what allows us to ignore the $\mathrm{E}(X)^2$ in Eq. (4) above. We put a square root in the denominator (vs. using variance) to ensure that the unit matches the numerator. Also notice that $\langle x'_i|(A')^T A'|x'_i\rangle = \frac{\langle x_i|A^T A|x_i\rangle}{\lambda_i^2} = \frac{\lambda_i^2}{\lambda_i^2} = 1$. Because the $\{|x_i\rangle\}$ form a basis, every vector in the span can be written as $|x\rangle = \sum_i a_i|x_i\rangle$, where $\sum_i a_i^2 = 1$; it follows that $\langle x|(A')^T A'|x\rangle = 1$ for all $|x\rangle$.

3.2. User embeddings. Start with a matrix A that maps users to their normalized features with SVD, $A = \sum_i |u_i\rangle\langle v_i|$. (Notice: The $\lambda_i = 1$ because we normalized.) We've been talking about principle components as the vectors $\{\vec{u}_i\}$. But if we want to know the values for an actual user e_j, then we can first map the user to their raw variables, $A|e_j\rangle$. Then project the raw values (number of bankruptcies, number of credit cards, etc.) onto the principle components by applying the map $\sum_{i=1}^{10} |u_i\rangle\langle u_i|$. Then this equals:

$$\left(\sum_{i=1}^{10} |e_i\rangle\langle u_i|\right)\left(\sum_{i=1}^{1000} |u_i\rangle\langle v_i|\right)|e_j\rangle = \sum_{i=1}^{10} |e_i\rangle\langle v_i|e_j\rangle$$

Compare to the user embeddings from the last section, we get that the embedding of the j-th user with 10 latent dimensions is $\sum_i |e_i\rangle\langle v_i|e_j\rangle$. They're the same. We might think of PCA as Collaborative Filtering, where we just don't use the embeddings of the features. It's an imperfect analogy, because really these are used in different cases. Not only do we not want the "embeddings" of the features, but the vectors $\{\vec{u}_i\}$ are not even meaningful in the same way that the movie vectors were; the rows of $U\sqrt{D}$ would be the embedding of a credit profile that has a single normalized feature set to 1 and the rest set to 0, which doesn't represent anything.

Exercises

2.3.1 Load the Iris dataset. In Python this can be done with:

```
import sklearn.datasets
iris = sklearn.datasets.load_iris()
```

This is available in other languages too. The dataset has 150 datapoints with 4 numerical features and a single categorical target. In Python, you can get these as numpy arrays:

```
features, target = iris["data"], iris["target"]
```

Use PCA to reduce the feature space to a 3-dimensional space. Report the principle components. These should be three vectors, each 4-dimensions. You can either manually compute this using the SVD functionality introduced in Section 1, or you can look up a PCA function (like in sklearn).

2.3.2 Given a matrix A, and an SVD $AA^T = UDV^T$, consider the embedding of the i-th row of A given by $\sqrt{D}V^T A|e_i\rangle$. Show that the dot product of the i-th embedding and the j-th embedding equals $\langle e_i|A^T DA|e_j\rangle$.

2.3.3 One might argue that the rows of the watch matrix from the previous section should be normalized. What are some pros and cons of this?

2.3.4 $C = A^T A$ is sometimes called the covariance matrix, because every term, C_{ij} is the covariance between the i-th and j-th variable. Suppose A' normalizes the rows of A. Prove that $C' = (A')^T A'$ is a correlation matrix, in the sense that C'_{ij} is the Pearson's correlation coefficient between the i-th and j-th variable of A.

2.3.5 For both parts of this problem, assume you have some matrix

$$A = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$$

(a) Suppose that for all i, $y_i = mx_i$ for some fixed m. Prove that $A = \lambda|u\rangle\langle v|$, and that $|v\rangle$ spans all the data points (x_i, y_i). (Hint: Guess and verify an SVD, then observe that this is unique up to $\pm|v\rangle$.)

(b) Suppose that for all i, $y_i = (m + \varepsilon_i)x_i$, where each ε_i is taken independently from a normal distribution $\mathrm{N}(0, \sigma)$, for some small fixed σ. Given an SVD, $A = \lambda_1|u_1\rangle\langle v_1| + \lambda_2|u_2\rangle\langle v_2|$, prove that $\mathrm{E}\left(|v_1\rangle\right) = |v\rangle$, where $|v\rangle$ is from part (a).

2.3.6 Critique the following reasoning: "A model is trained on a PCA based on 3 years worth of data. After one month the PCA is updated with the extra month of data, but still using the first 3 years. I shouldn't have to retrain the model since the principle components will be approximately the same as the previous month."

4. Partial Least Squares

In the previous section, we may have rejected number of bankruptcies too quickly. Though it's uncommon, it may be very predictive of a default. If so, we should include it. Instead of looking at variance of the feature space by itself, **Partial Least Squares (PLS)**[4] tries to find those axes that most correlate with the target.

Continuing the example, say that the target is loan default (1 for default and 0 for no default). We have historical data which says whether users with a given profile defaulted. With n historical data points, we put these targets in a $n \times 1$ matrix X. We also have features for these data points, an $n \times m$ matrix Y. $X^T Y$ is a $1 \times m$ matrix, where each entry $(X^T Y)_{1j}$ is correlation coefficient of the j-th dimension of Y with the target variable, provide that X and Y are normalized. $\langle u|X^T Y|v\rangle$ is the correlation between the axes $X|u\rangle$ and $Y|v\rangle$. To maximize correlation, we find the maximizing $\vec{u}$ and $\vec{v}$, which we can again do with the SVD. $\vec{u}$ is a 1-dimensional unit vector, so it must be $\pm(1)$. The maximizing vector $|v\rangle$ is more interesting; it represents the axis in the feature space that most correlates with the target. Because $X^T V$ will have rank 1, we can only get a single PLS dimension out. However, this dimension is likely to be a good dimension for predicting a default.

If there are multiple target dimensions, you could still perform an SVD on $X^T Y$, and get as many PLS axes as there are target dimensions. (See Exercise

[4]Don't try to make sense of the name.

2.4.1 for an example of this.) In that case, the vectors $\{\vec{u}_i\}$ will be non-trivial. However, even in that case, only the dimensions in the feature space, $\{\vec{v}_i\}$ will be of interest.

Collaborative Filtering and PCA are considered **unsupervised models**, because they don't take into account target. PLS is a **supervised model** because it does. Unsupervised models are common for building embeddings, because we think of them as trying to capture some intrinsic about the data. A single embedding will often be used in multiple applications, and tying too closely to a single target may not be desirable.

PLS is less commonly used than the other algorithms presented here. One reason may be that it can only produce as many dimensions as you have targets. (However, see Exercise 2.4.4 for a way to supplement PLS and PCA.) Another reason may be that PCA is an unsupervised algorithm. This means that you don't need target data. But also, dimensionality reduction is usually done upstream of modeling; the output may be used for exploration or sent to many different applications. A third reason that PLS is less common than PCA may just simply be that it is less well-known. PLS might be thought of as a variant of PCA. PCA is valued for its simplicity, and modelers probably don't spend much time researching variants. In my experience, I've found that PLS can be very useful, especially if you have a single application in mind. It's my opinion that PLS should be a tool in any modelers tool belt.

Exercises

2.4.1 Load the Iris dataset. (See Exercise 2.3.1 for more details.) Make three target variables, 1 or 0 indicating whether the datapoint is a Setosa, Versicolous, or Virginica iris. Use PLS to reduce features to a 3-dimensional space.

2.4.2 Critique the following model proposal: "Given loan data, we use PLS to get the single dimension that is most correlated with default. The applicants who score lower on this dimension must be less likely to default. Approve all loans below a certain threshold."

2.4.3 You and a coworker are building an insurance model on credit data. The model will be a logistic regression to predict the probability of an insurance claim. You have historical claim data. You think you should use only PLS, and your coworker thinks you should only use PCA. How can you decide with data which model is better?

2.4.4 Consider a scheme where you use both PLS and PCA. If you start with 1 PLS dimension $|x\rangle$, how can you choose the dimension that is orthogonal to this dimension and that captures the most variance? That is, how do you find

$$\max_{\||u_1\rangle\|=1, \langle x|u_1\rangle=0} \langle u_1|A^T A|u_1\rangle$$

2.4.5 Imagine you fit a logistic regression on a PLS dimension. Because the PLS dimension and the regression are both fit on the same data, you have concerns that you are overfitting the data. How can you confirm that your model is not overfit?

2.4.6 Given a feature matrix, $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, and a target variable, $\begin{pmatrix} x \\ y \end{pmatrix}$, suppose you compute a PLS dimension as $\begin{pmatrix} u \\ v \end{pmatrix}$. Compute $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial v}{\partial x}$, and $\frac{\partial v}{\partial y}$.

2.4.7 $\diamondsuit$ Start with embeddings from problem 2.2.8 or found online, and filter this list to 100 movies that people are likely to know. (Look at the IMDB top 250 if you aren't sure.) Make a game, called Hot-or-Cold: This game chooses a random (secret) game from the 100 movies. The user guesses which movie it is, and the program tells them how close they are to the target, by telling them the cosine similarity between the guessed movie and the secret movie. Show the player the top 100 list, or allow the user to enter any movie from the larger dataset.

5. Canonical Correlation Analysis

For this last section, we imagine that we're making recommendations in the iPhone app store. A partner team uses a model that takes as input features of apps that users have downloaded, and outputs a recommendation for a new app. We don't know exactly how that model works, but that team counts on us to build features that well represents apps.

We'll consider a couple of good approaches to featurizing apps. One approach is to use Collaborative Filtering to get co-download embeddings for the apps. This means that if two apps have lots of users in common, then the apps' embedding vectors will be near each other; apps with less users in common will be far apart. Another approach is to use the app description. These descriptions are entered by the developer and are presented to the user in the app store. Somebody from the natural language team has made embeddings based on these text descriptions.

Although these two embeddings are capturing different signal, these signals are likely to be correlated. For example, apps like Microsoft Teams, Microsoft Word, and Microsoft OneDrive are likely to be downloaded by the same people, but these are also likely to all say "Microsoft" in the description. Here we think correlated signals may amplify each other. The word "bird" might show up in bird watching apps, but also in bird-themed cell phone games (Angry Birds, Flappy Bird, and some knock-offs). It's not until a signal shows up in both embeddings that we think we've caught a family of related apps.

To do this, we look for axes that maximize covariance. This is exactly the same problem as PLS. Specifically, we start with two matrices of features X and Y of dimension $m_1 \times n$ and $m_2 \times n$, with n the number of users, and m_1 and m_2 are the number of dimensions in the two feature sets. The matrix XY^T has entries m_{ij} equal to the correlation between the i-th feature in X and the j-th feature in Y. We look for the axes that maximize $\langle u|XY^T|v\rangle$. Again this is solved with an SVD. We call the search for these axes a **Canonical Correlation Analysis (CCA)**.

Many of the topics covered in this chapter are closely related, but none more than PLS and CCA. Again the use cases for the two are much different. For PLS, we use the target variable to determine the best axis among a single set of features. But for CCA, we use two sets of features to find the best axis.

Because of this there are two sets of features, we end up having two signals to which we can attach for any user. We can embed a user $|e_i\rangle$ by first mapping them to their X features, then embedding this like we did in the Collaborative Filtering

section, $\sqrt{D}U^T X|e_i\rangle$. *Or* we could map first into Y and embed, $\sqrt{D}VY^T|e_i\rangle$. These are *almost* the same signal (see Exercise 2.5.2), and in practice we could use just one of these signals.

Exercises

2.5.1 For each of the following scenarios choose one of the models we covered in this chapter, and describe how it could be used to help the situation.

(a) Twitter is trying to recommend posts to users. You have a lot of data about historical post views for each, and data about posts, including co-views and text embeddings.

(b) You're trying to build a spam filter. You have a large labeled set, and features for each of the emails, including text embeddings, domain-specific flags, and author information.

(c) YouTube is trying to build a user embedding based on a very large number of features, including geographic, platform, demographics, watch history, and others. The embeddings will be used for several applications.

(d) Pinterest is trying to build a user embedding based only on what posts the user favorited. The embeddings will be used for several applications.

(e) An insurance company is trying to build a fraud detector based only on the text content of police reports. You have a large number of labeled data points.

2.5.2 You do CCA on two matrices X and Y to get that $XY^T = UDV^T$. Imagine that we used the axes $\{\vec{u}_i\}$ to perform a linear regression. A linear regression treats the vectors $\{\vec{u}_i\}$ as the data points, along with targets y_i, and it tries to find the coefficients that minimize the square error:

$$\min_{\{c_j\}} \sum_i \left(y_i - \sum_j c_j u_{ij} \right)^2 = \min_{\langle c|} \sum_i \left(y_i - \langle c|u_i\rangle \right)^2$$

Show that when $D = I$, that:

$$\min_{\langle c|} \sum_i \left(y_i - \langle c|u_i\rangle \right)^2 = \min_{\langle c|} \sum_i \left(y_i - \langle c|v_i\rangle \right)^2$$

2.5.3 Imagine you have two embeddings representing the same data. To get a 20-dimensional representation of the data, you suggest using CCA on the two embeddings. Your coworker argues that to ensure orthogonality, you should instead do PCA on both embeddings individually to get 10 dimensions from each. How would you argue that your approach is better?

2.5.4 Suppose that X and Y have the same latent dimensions with different singular values. That is, $X = \sum_i \lambda_i |u_i\rangle\langle v_i|$ and $Y = \sum_i \mu_i |u_i\rangle\langle v_i|$. Show that the first axis that CCA produces is $\langle v_1|$.

2.5.5 Suppose that X and Y have SVDs $X = \sum_i \lambda_i^X |u_i^X\rangle\langle v_i^X|$ and $Y = \sum_i \lambda_i^Y |u_i^Y\rangle\langle v_i^Y|$, with $\langle u_i^X | v_j^Y \rangle = 0$ and $\langle u_i^Y | v_j^X \rangle = 0$ for all i and j. (The superscripts are indices, and not exponents.) Prove that the resulting CCA axis are the principle components from X and Y. (Hint: Consider the PCA of $(X - Y)$.)

CHAPTER 3

[WIP] Regression

1. Projection matrices

We'll talk now about an important family of matrices, called projection matrices. A *Projection Matrix* is a square matrix with an SVD given by $P = \sum_i |v_i\rangle\langle v_i|$ for some set of orthonormal vectors. The important properties are that each vector is paired with itself and that all coefficients are 1. Note that projection matrices under this definition are necessarily symmetric. At its surface level, this is a simple definition - a projection matrix just has a special SVD. But this has a number of equivalent properties, that are sometimes given as definition.

THEOREM 3.1.1. *For a matrix P, with $\mathcal{P} := \mathrm{Im}(P)$, the following properties are equivalent:*

- *a Every SVD representation of P has the form $P = \sum_i |v_i\rangle\langle v_i|$, where the set of $\{|v_i\rangle\}_i$ span the $\mathcal{P}$.*
- *b $P|v\rangle = |v\rangle$ for all $|v\rangle \in \mathcal{P}$, and $P|w\rangle = |0\rangle$ for all $|w\rangle \in \mathcal{P}^\perp$.*
- *c $P^2 = P$ and symmetric.*

Property b is the projection property that gives these matrices their name, and it's probably the most important of the three. This property says that the matrix projects vectors onto it's image, $\mathcal{P}$. We know from Theorem 9.5.2 that we can write any vector $|v\rangle$ as $|v_\parallel\rangle + |v_\perp\rangle$, where $|v_\parallel\rangle \in \mathcal{P}$ and $|v_\perp\rangle \in \mathcal{P}^\perp$. Property b then tells us that $P|v\rangle = |v_\parallel\rangle$; this is the orthogonal projection of $|v\rangle$ to the subspace $\mathcal{P}$. This is called orthogonal, because the difference, $|v_\perp\rangle$ is orthogonal to $\mathcal{P}$. $|v_\parallel\rangle$ happens to be the vector in $\mathcal{P}$ that's closest to $|v\rangle$. When we say "projection" colloquially, usually we mean orthogonal projection; for example the projection of all the points on a sphere to a planar subspace is the disc that gets projected (like a shadow cast) from the sphere onto the plane. $\mathcal{P}$ is can be called the *Invariant Subspace* of P.

Property c contains the property that $P^2 = P$, which is called *idempotence*, meaning simply that if you apply the matrix repeatedly, it does the same as if you apply it only once. See Exercise 3.1.9 for an example of the type of arithmetic you can do with these matrices. Property c also mentions that the matrix must be symmetric. The reader may be familiar with this property that means $P^T = P$. If you're unfamiliar, read the beginning of Section 1, search the internet, or skip this property for now.

PROOF. $a \implies b$ Let $|v\rangle$ be in $\mathcal{P} = \mathrm{Im}(P)$, then $|v\rangle = \sum_i a_i|v_i\rangle$. Then $P|v\rangle = \sum_j \sum_i a_i|v_j\rangle\langle v_j|v_i\rangle = \sum_j a_j|v_j\rangle = |v\rangle$. If $|w\rangle \in \mathcal{P}^\perp$, then $P|w\rangle = |0\rangle$ is immediate.

$b \implies c$ $P^2 = P$ if $P^2|v\rangle = P|v\rangle$ for all $|v\rangle$. Given the above argument $P|v\rangle = |v_\parallel\rangle$, where $|v_\parallel\rangle \in \mathcal{P}$. So we have $P^2|v\rangle = P|v_\parallel\rangle = |v_\parallel\rangle = P|v\rangle$.

To prove that P must be symmetric, we show that $\langle u|P|v\rangle = \langle u|P^T|v\rangle$ for all $\langle u|$, $|v\rangle$. (Why is this sufficient?) We have that

$$\begin{aligned}\langle u|P|v\rangle &= \langle u|v_\parallel\rangle \\ &= \langle u_\parallel|v_\parallel\rangle + \langle u_\perp|v_\parallel\rangle \\ &= \langle u_\parallel|v_\parallel\rangle\end{aligned}$$

The last equality follows from the fact that $v_\parallel \in \mathcal{P}$, and is therefore perpendicular to $u_\perp$. A similar argument says that $\langle u|P^T|v\rangle = (\langle u|P^T)|v\rangle = \langle u_\parallel|v\rangle = \langle u_\parallel|v_\parallel\rangle$. So these calculations are the same.

$c \implies a$ We look ahead to a theorem from Section 1 to say that any SVD for a symmetric matrix has the form $P = \sum_i \lambda_i |v_i\rangle\langle v_i|$. Now when we square this we get

$$\begin{aligned}P^2 &= \left(\sum_i \lambda_i |v_i\rangle\langle v_i|\right)\left(\sum_j \lambda_j |v_j\rangle\langle v_j|\right) \\ &= \sum_i \sum_j \lambda_i\lambda_j |v_i\rangle\langle v_i|v_j\rangle\langle v_i| \\ &= \sum_i \lambda_i^2 |v_i\rangle\langle v_i|\end{aligned}$$

Where the last equality from the usual trick of taking advantage of orthonormality to cancel any term with $\langle v_i|v_j\rangle$ where $i \neq j$.

Now $P^2 = P \implies \sum_i \lambda_i^2 |v_i\rangle\langle v_i| = \sum_i \lambda_i |v_i\rangle\langle v_i|$. We can apply $|v_i\rangle$ to both side of either matrix to get $\lambda_i^2 = \lambda_i$, implying that λ_i is 0 or 1, completing the proof. □

Corollary 3.1.2. *For a projection matrix P with range $\mathcal{P}$, $|P|v\rangle - |v\rangle| = \min_{|w\rangle\in\mathbb{P}} \||w\rangle - |v\rangle\|$.*

Proof. TODO □

Recall that we used projection matrices already in Theorem 1.2.3. There we argued that a matrix like $I - |v\rangle\langle v|$ is a projection. With Theorem 3.1.1, we can see that this must be a projection matrix, because it's obviously symmetric, and the reader can check that it's idempotent. But what is it's SVD? By Exercise 1.2.2, we can choose any basis $\{|v_i\rangle\}_i$, where $|v_1\rangle = |v\rangle$, then $I = \sum_i |v_i\rangle\langle v_i|$, and $I - |v\rangle\langle v| = \sum_{i\neq 1} |v_i\rangle\langle v_i|$ is a valid SVD. This begins to establish the one-to-one connection between a projection matrix's SVD and its invariant subspace, made explicit by the next theorem.

Theorem 3.1.3. *A projection matrix has an invariant subspace spanned by $\{|v_i\rangle\}_i$ if and only if $\sum_i |v_i\rangle\langle v_i|$ is an SVD for the matrix.*

Proof. Start by assuming that $\{|v_i\rangle\}_i$ spans the invariant subspace, $\mathcal{P}$, of P. Then we argue as above that for any $\langle u|$, $|v\rangle$, that $\langle u|P|v\rangle = \langle u_\parallel|v_\parallel\rangle$. Now for any $|v\rangle$, we can write $|v\rangle = |v_\parallel\rangle + |v_\perp\rangle$. Since $|v_\parallel\rangle \in \mathcal{P}$, the span of $\{|v_i\rangle\}_i$ we can write $|v_\parallel\rangle = \sum_i a_i|v_i\rangle$, for some $\{a_i\}_i$. So that $\left(\sum_i \langle v_i|\right)|v\rangle = \left(\sum_i \langle v_i|\right)\left(\sum_j a_j|v_j\rangle\right) = \sum_i a_i|v_i\rangle$. So that

$$\begin{aligned}\left(\sum_i \langle v_i|\right)|v\rangle &= \left(\sum_i \langle v_i|\right)\left(|v_\parallel\rangle + |v_\perp\rangle\right) \\ &= \left(\sum_i \langle v_i|\right)|v_\parallel\rangle + \left(\sum_i \langle v_i|\right)|v_\perp\rangle \\ &= |v_\parallel\rangle + 0\end{aligned}$$

Applying the same to the other side, we get that $\langle u|\left(\sum_i |v_i\rangle\langle v_i|\right)|v\rangle = \langle u_\parallel|v_\parallel\rangle$. So that $P = \sum_i |v_i\rangle\langle v_i|$, and this is therefore an SVD of P.

To prove the other direction, assume that $P = \sum_{i=1}^k |v_i\rangle\langle v_i|$, we take some basis of the domain $\{|w_i\rangle\}_i$ with $|w_i\rangle = |v_i\rangle$ for $i = 1..k$. That is, this basis extends the $|v_i\rangle$ to the entire domain, as in Theorem 9.5.1. Now any vector $|v\rangle = \sum_{i=1}^n a_i|w_i\rangle$,

and $\left(\sum_{i=1}^{k} |v_i\rangle\langle v_i|\right) \sum_{j=1}^{n} a_j|w_j\rangle = \sum_{i=1}^{k} a_i|w_i\rangle$. This equals $|v\rangle$ if and only if $a_i = 0$ for all $i > k$; that is if $|v\rangle \in (\,span)\,\{|v_i\rangle\}$. □

Notice that this theorem doesn't rely on the fact that the invariant subspace is the image of the projection operator, but from the SVD, we can see that the image is precisely spanned by the $\{|v_i\rangle\}$.

Before moving on, we want to highlight a special projection matrix, $|v\rangle\langle v|$. For this to be a projection matrix, $|v\rangle$ must be a unit vector. (Why?) This takes a vector and projects onto $|v\rangle$, so that $|w\rangle \to |v\rangle\langle v|w\rangle$. Recall that the $\langle v|w\rangle$ is the inner product, or dot product; this projection is sometimes written as $\text{proj}_{\vec{v}}\,\vec{w} = \left(\vec{v}\dot{\vec{w}}\right)\vec{v}$. Because $|v\rangle$ is a unit vector, it gives the direction of $|v\rangle$ and $\langle v|w\rangle$ gives the magnitude of the resulting projection. Doing some trig, we get that the magnitude must also equal $\||w\rangle\| \cos\theta$, where θ is the angle between $|v\rangle$ and $|w\rangle$. More generally if $|v\rangle$ isn't unital, then $\langle v|w\rangle = \||v\rangle\| \, \||w\rangle\| \cos\theta$. This formula gives a concrete interpretation of the inner product.

Exercises

3.1.1 To make the shadow example explicit, argue that the unit sphere given by all points $x^2 + y^2 + z^2 = 1$ maps to a unit disc under the projection matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. Graph these. Generalize by arguing that the image of the unit sphere under any projection is a unit disc (potentially 0 or 1 dimensional).

3.1.2 Prove that the image of any matrix forms a subspace.

3.1.3 Prove that a projection matrix is either singular or the identity matrix.

3.1.4 Matrices do not have a unique SVD representation. However, prove that if a matrix has any SVD of the form $P = \sum_i |v_i\rangle\langle v_i|$, then all SVDs must be of the form $P = \sum_j |v_j\rangle\langle v_j|$.

3.1.5 Planes in higher dimensions are linear spaces with codimension of 1, meaning that there's a unique (up to magnitude) vector that's perpendicular to the plane, called the normal vector. Let $|v\rangle$ be a normal vector to the plane, and argue that $\langle v|x\rangle = 0$ is a formula for the plane. Why might it be valuable to choose a normal vector?

3.1.6 Prove the *Cauchy Schwartz inequality* that $\|\langle u, v\rangle\|^2 \leqslant \langle u, u\rangle\langle v, v\rangle$.

3.1.7 Intuitively we expect the composition of two projection matrices, P and Q, to be a projection matrix, PQ, with the image being the intersection of $\text{Im}(A)$ and $\text{Im}(B)$. Prove that this is the case.

3.1.8 Let P and Q be projection matrices with images $\mathcal{P}$ and $\mathcal{Q}$ that are orthogonal - that is $\langle p|q\rangle = 0$ for all $|p\rangle \in \mathcal{P}$, $|q\rangle \in \mathcal{Q}$. Prove that $P + Q$ is a projection matrix also.

3.1.9 Given the definition of $\exp(M) = \sum_n \frac{M^n}{n!}$, for any matrix, M, show that $\exp P = e + (e-1)P$ for all idempotent matrices, P. We generalize this in Section 4.

2. Inverses and pseudo-inverses

An inverse of a matrix, M, is a new matrix, M^{-1} that "undoes" M. That is, whenever $M|v\rangle = |w\rangle$, then $M^{-1}|w\rangle = |v\rangle$. Said another way, $MM^{-1} = M^{-1}M = I$.

Example

Rotation Matrix A rotation matrix in two dimensions is given by

$$R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

This acts on any vector $|v\rangle$ by rotating counter clockwise by θ. It follows then that $R(\theta)^{-1} = R(-\theta)$.

We note that not all matrices have inverses; more on that in a bit.

Notice that if A, B, and AB are invertible, then $(AB)^{-1} = B^{-1}A^{-1}$. That is, the order is reversed, and because matrices do not commute, this matters. This is because if $AB|v\rangle = |w\rangle$, then $B|v\rangle = A^{-1}|w\rangle$, and finally $|v\rangle = B^{-1}A^{-1}|w\rangle$.

Next we notice that if a matrix is a diagonal with non-zero entries $D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$, then the inverse is also a diagonal matrix, $D^{-1} = \begin{pmatrix} 1/\lambda_1 & 0 & \cdots & 0 \\ 0 & 1/\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\lambda_n \end{pmatrix}$. The reader can easily verify that these are inverses.

Putting these facts together with the fact that $M^{-1} = M^T$ for orthogonal matrices (see Section 1), we can calculate the inverse of a matrix, M, with the given SVD (in matrix form) of $M = UDV^T$. That is, $M^{-1} = (V^T)^{-1}D^{-1}U^{-1} = VD^{-1}U^T$. This form is already given as an SVD, since the multiplicands are still orthogonal and diagonal. Since every square matrix has an SVD UDV^T, this approach gives an inverse whenever D has all non-zero diagonals; that is whenever D is full-rank. Conversely if D does have a zero-entry on the diagonal, then it's not invertible, or *singular*. To see this, suppose there's a zero entry in the $(1,1)$ entry of D, then nothing maps to the first row of U. Restated in summation form if $M = \sum_i \lambda_i |u_i\rangle\langle v_i|$ where $\lambda_1 = 0$, then no $|v\rangle$ maps to $M|v\rangle = |u_1\rangle$; the best we could do is some combination of $|u_i\rangle$ with $i > 1$, which are all perpendicular to $|u_1\rangle$. Given Theorem 1.5.2, we can conclude that a matrix is singular if and only if the determinant is 0.

What inversion does to an SVD is very natural. Suppose a square matrix $M = \sum_i \lambda_i |u_i\rangle\langle v_i|$ is full-rank, that is $\lambda_i \neq 0$ for $i = 1..n$, with n equal to the dimension of domain. Then $M^{-1} = \sum_i \frac{1}{\lambda_i} |v_i\rangle\langle u_i|$. Because M is full-rank, both $\{|u_i\rangle\}_i$ and $\{|v_i\rangle\}_i$ form bases, sometimes called coordinates. The SVD of M basically says that to calculate $M|v\rangle$, you must first write $|v\rangle$ in the $\{|v_i\rangle\}_i$ coordinate system, then scale each i-th coordinate by λ_i, and the result is $M|v\rangle$ written in the $\{|u_i\rangle\}_i$ coordinate system. (See Subsection 1.4 for more details.) The inverse then writes a vector in the $\{|u_i\rangle\}_i$ coordinate system, unscales the i-th coordinate by $1/\lambda_i$, and ends up with a vector written in the $\{|v_i\rangle\}_i$ coordinate system.

We now look at an example of a singular matrix.

Example

Singular Rotation Matrix The following matrix rotates the first two dimensions, but zeroes out the third dimension.

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$R\langle x, y, z\rangle$ first zeroes out the third coordinate, then rotates $\langle x, y\rangle$ by θ in the $x-y$ plane, landing on $\langle x', y', 0\rangle$. Of course, this is singular because we nothing maps to $\langle x', y', z'\rangle$ if $z' \neq 0$. Any vector $\langle x', y', 0\rangle$ does have a vector that maps to it, $\langle x, y, 0\rangle$. However, this isn't unique, because $\langle x, y, z\rangle$ also maps to $\langle x', y', 0\rangle$ for all $z \in \mathbb{R}$. Because of the non-uniqueness, we don't call these inverses.

Although R has no inverse, we could do an almost-inverse. Given a vector, $|w\rangle$, we could project $|w\rangle$ to the first two dimensions, resulting in $P|w\rangle$, then apply $R(-\theta)$ to those two dimensions. This would give us a new vector, $|v\rangle$, which has the property that $R|v\rangle$ is as close to $|w\rangle$ as possible. This "close as possible" claim is justified by the fact that $R|v\rangle = RR(-\theta)P|w\rangle = P|w\rangle$ is the projection of $|w\rangle$ onto the image of R. By Theorem 9.5.2, the projection is the closest vector in the first two dimensions (the image of R) to $|w\rangle$. Writing such an "almost-inverse" down would look like.

$$S = \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 0 \end{pmatrix} (|e_1\rangle\langle e_1| + |e_2\rangle\langle e_2|)$$

The first matrix is the inverse on the image of R and the second matrix is projection onto that image. If we knew an SVD for $R = \lambda_1|u_1\rangle\langle v_1| + \lambda_2|u_2\rangle\langle v_2|$, then we'd know that the image $\text{Span}\{|u_1\rangle, |u_2\rangle\} = \text{Span}\{|e_1\rangle, |e_2\rangle\}$. By Theorem 3.1.3, we can rewrite $|e_1\rangle\langle e_1| + |e_2\rangle\langle e_2| = |u_1\rangle\langle v_1| + |u_2\rangle\langle v_2|$. And so we can rewrite,

$$S = \left(\frac{1}{\lambda_1}|v_1\rangle\langle u_1| + \frac{1}{\lambda_2}|v_2\rangle\langle u_2|\right)(|u_1\rangle\langle v_1| + |u_2\rangle\langle v_2|) = \frac{1}{\lambda_1}|v_1\rangle\langle u_1| + \frac{1}{\lambda_2}|v_2\rangle\langle u_2|$$

The first matrix is the inverse of the SVD of R, which exists if we restrict to the first two dimenisions. This tells us that the almost inverse is given by doing the inverse to the non-zero terms of the SVD. It almost doesn't feel worth mentioning projection, but when you consider that there's an implicit zero term in the SVD of R, we see that $R = R = \lambda_1|u_1\rangle\langle v_1| + \lambda_2|u_2\rangle\langle v_2| + 0|e_3\rangle\langle e_3|$. It hardly feels justifiable to invert the non-zero terms and leave the zero term untouched. But when we consider that it's the ortogonal projection onto R's image, then it seems justifiable by closeness arguments.

Generalizing the example, we define the *pseudo-inverse* of a matrix $M = \sum_i \lambda_i|u_i\rangle\langle v_i|$ is $M^+ = \sum_i \frac{1}{\lambda_i}|v_i\rangle\langle u_i|$. The pseudo-inverse of a matrix always exists, even for non-square matrices. In general, there's no good way to write this in matrix form, because we're unable to ignore the zero entries in the diagonal matrix.

Also in general, there is no such relationship M^+, M, and I. We'll discuss some special cases now.

Given a matrix $M = \sum_i \lambda_i |u_i\rangle\langle v_i|$, with non-zero terms, if $\{|v_i\rangle\}_i$ span the domain, then $M^T M = \sum_i \lambda_i^2 |v_i\rangle\langle v_i|$ is square and full-rank, and therefore invertible. It's inverse using arguments above must be $\left(M^T M\right)^{-1} = \sum_i \frac{1}{\lambda_i^2} |v_i\rangle\langle v_i|$. This looks like projection, with some scaling, so we try composing it with M to see if we can get the pseudo-inverse. And indeed, we see that $\left(M^T M\right)^{-1} M^T = \sum_i \frac{1}{\lambda_i} |v_i\rangle\langle u_i| = M^+$. In the case that $\{|v_i\rangle\}_i$ span the domain, and $M^+ = \left(M^T M\right)^{-1} M^T$, we have that $M^+ M = \left(M^T M\right)^{-1} M^T M = I$. We say then that M^+ is a *Left Inverse* of M.

On the other hand, given a matrix $M = \sum_i \lambda_i |u_i\rangle\langle v_i|$, with non-zero terms, if $\{|u_i\rangle\}_i$ span the codomain, then $MM^T = \sum_i \lambda_i^2 |u_i\rangle\langle u_i|$ is full-rank, and therefore invertible. In this case $M^+ = M^T \left(MM^T\right)^{-1}$, and $MM^+ = I$, making M^+ a *Right Inverse* of M.

The definition of a pseudo-inverse is very natural when you have the bra-ket form of the SVD. But most linear algebra books do not center the SVD. So the pseudo-inverse is given as an abstract matrix satisfying certain properties. In the two cases outlined above, an explicit formula can be given for the pseudo-inverse. As such some books decide to only define the pseudo-inverse when it can be written explicitly. But this gets the implication backwards: It doesn't make sense to define the pseudo-matrix in terms of the inverse, because the inverse is a special case of the pseudo-inverse. These books prefer this only because they've taken time to teach matrix inversion in detail, even though almost all matrices are not invertible.

The condition for a left inverse is that $\{|v_i\rangle\}_i$ span the domain. But this is equivalent to saying that M is injective. That is, for any two distinct vectors $|v\rangle$ and $|w\rangle$ in the domain, $M|v\rangle \neq M|w\rangle$. It's outside the scope of this book, but any injective function (even non-linear functions) have left inverses. The condition for a right inverse is that $\{|u_i\rangle\}_i$ span the codomain. This is equivalent to the matrix being surjective. That is, for any vector $|w\rangle$ in the codomain, there's a vector $|v\rangle$ in the domain with $M|v\rangle = |w\rangle$.

Exercise 1.2.2 shows that for any basis of $\mathbb{R}^n$, $\{|v_i\rangle\}$, $\sum_i |v_i\rangle\langle v_i| = I$. This section puts that exercise in a new light, because we can recognize $\sum_i |v_i\rangle\langle v_i|$ as a projection matrix. As such, its invariant subspace is it's image, but because the image is the basis, it spans $\mathbb{R}^n$; that is, $\left(\sum_i |v_i\rangle\langle v_i|\right)|v\rangle = |v\rangle$ for all $|v\rangle \in \mathbb{R}^n$. So it's the identity.

THEOREM 3.2.1. *If a matrix Q has orthonormal rows, then $Q^T Q = I$ If it has orthonomal columns then $QQ^T = I$. Such matrices are called* semi-orthogonal.

PROOF. We'll prove the first part. Any matrix Q with rows given by $\{|v_i\rangle\}$ can be written as $Q = \sum_i |e_i\rangle\langle v_i|$. (See Section 1.3.) This implies that $Q^T Q = \sum_i |v_i\rangle\langle v_i|$. Though this is always true, when the $|v_i\rangle$ are orthonormal, $\sum_i |v_i\rangle\langle v_i|$ form a valid SVD. In this case, $Q^T Q$ must be a projection matrix, invariant on the span of $\{|v_i\rangle\}$. If Q has n rows, then $Q^T Q$ is a an $n \times n$ matrix, so $\{|v_i\rangle\}$ spans the domain and range, and $Q^T Q$ must fix every element. □

Note that by Theorem **??**, square matrices have orthonormal rows precisely when they orthonormal columns. By the above theorem, such matrices must have $Q^T Q = QQ^T = I$. So Theorem 9.1.2 that $Q^{-1} = Q^T$ for orthogonal matrices follows immediately.

Exercises

3.2.1 Prove that if $AB = I$, then $BA = I$.

3.2.2 Show that $R(\theta)R(-\theta) = I$ by multiplying matrices, and applying trig identities.

3.2.3 Prove that A and B are both invertible if and only if AB is invertible.

3.2.4 Prove the above claim that for a matrix $M = \sum_i \lambda_i |u_i\rangle\langle v_i|$, $\{|v_i\rangle\}_i$ spans the domain if and only M is injective.

3.2.5 Given that the SVD of an $n \times n$ square matrix can be computed in $O(n^3)$-time, argue that we can compute the inverse in $O(n^3)$-time. Note: This is not the most efficient way to calculate an inverse.

3.2.6 Show that when an inverse exists, there is a unique inverse. That is, for an invertible matrix, A, there is only one matrix, B, for which $AB = I$. Pseudo-inverses are also unique.

3.2.7 Prove the following facts about the pseudo-inverse:

a $AA^+A = A$

b $A^+AA^+ = A^+$

c $(AA^+)^T = AA^+$ (We can continue to assume that A and A^+ are all real.)

d $(A^+A)^T = A^+A$

3.2.8 Show that $(AB)^+ = B^+A^+$.

3.2.9 Why is it true that a square matrix is injective if and only if it's surjective?

3.2.10 A 1×1 matrix (m) maps 1-scalars (x) to 1-scalars (mx). When is this matrix invertible? What is the inverse when it's invertible? What is the pseudo-inverse when it's singular?

3. Least Squares Regression

3.1. Solving System of Equations. Let's look at how inverses can be used to solve problems. We'll present a word problem like you might see in a high school class.

Example

Word Problem You're selling tickets to a 2-day show. You sell two types of tickets, regular and VIP. On the first day, you sell 100 regular tickets and 3 VIP tickets, and make 11,500 dollars. On the second day, you sell 90 regular tickets and 6 VIP tickets and make 12,000 dollars. How much does a VIP ticket cost versus a regular ticket?
Say that a regular ticket costs β_1 dollars and a VIP ticket costs β_2 dollars. We then have the linear relationship:

$$\begin{pmatrix} 11500 \\ 12000 \end{pmatrix} = \begin{pmatrix} 100 & 3 \\ 90 & 6 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

Of course you can solve this system of equations with some logic or using algebra techniques, but linear algebra saves you from having to do anything difficult and scales nicely to larger systems (at least from a human effort perspective). The matrix in this equation is invertible, so we can solve this system, by saying:

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 100 & 3 \\ 90 & 6 \end{pmatrix}^{-1} \begin{pmatrix} 11500 \\ 12000 \end{pmatrix} = \begin{pmatrix} 100 \\ 500 \end{pmatrix}$$

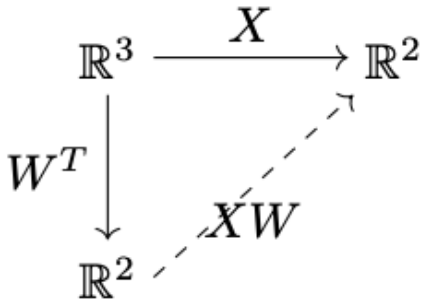
This is bread-and-butter applied linear algebra, and will probably be familiar to the reader. It's not surprising that solving system of linear equations is the primary motivation for studying inverses; this is the matrix-equivalent of solving $y = ax$ for a.

3.2. Under-determined System of Equation. But what if we modify the example slightly, so that we add a third class of tickets, called "VIP Plus?" Now your system looks like:

$$\begin{pmatrix} 12500 \\ 13000 \end{pmatrix} = \begin{pmatrix} 100 & 3 & 2 \\ 90 & 6 & 1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

You may recall from algebra that you generally cannot (and never for linear equations) solve for three unknowns with only two equations. Such a system is called *underdetermined*. But with the help of pseudo-inverses we can make a best guess. For now let's call the system $|y\rangle = X|\beta\rangle$, but we'll continue to assume that X is 2×3. We can almost solve this equation as $X^+|y\rangle = |\beta\rangle$.

It's probably still a little unclear what the pseudo-inverse means, or why this is a good approximate solution. So let's try a different approach: We know that as a 2×3-dimensional matrix, X cannot possibly be invertible. But suppose it's rank-2, as in our example. (A 2×3 matrix cannot have more than rank 2.) Then we want to reduce the dimensionality of the domain by 1 to end up with a 2×2 matrix, that we can then invert. We know from Section 3, that the PCA is the way to reduce dimensions with minimal information loss. For a rank-2 matrix, there will be two PCA axes that we can choose from. Let $X = UDV^T$ be an SVD. Now V^T will be a 3×3 orthogonality matrix, where the bottom row has no impact on X, because of a zero entry in D. The two PCA axes will be the first two rows of V^T, $|v_1\rangle$ and $|v_2\rangle$. To transform data points to the two SVD axes, we can just project onto each one. That is we let the first coordinate be the magnitude of the projection



onto $|v_1\rangle$, and the second coordinate will be projection onto $|v_2\rangle$. To transform any point to these two axes, we can just apply the matrix $W^T := |e_1\rangle\langle v_1| + |e_2\rangle\langle v_2|$.

We project down to $\mathbb{R}^2$ via W^T, then extend the map X via W to get XW, which is the composition of X and $(W^T)^{-1} = W$.

This gives us a map from the PCA space to the image. The inverse $(XW)^{-1}$ can be composed with $|y\rangle$ gives coefficients of the PCA axes. If we wanted to recover coefficients in the original $\mathbb{R}^3$, then we need to compose this result with W. So that the resulting coefficients are $W(XW)^{-1}|y\rangle$. Notice that if X were rectangular, then W would be full rank and invertible, so that this would reduce to $WW^{-1}X^{-1}|y\rangle = X^{-1}|y\rangle$ - This is a good sanity check.

So this tells us that we can almost invert by first projecting to the PCA, and this give an almost inverse of $W(XW)^{-1}$. We compare to the usual pseudo inverse of $X^T\left(XX^T\right)$.

$$\begin{aligned}
W(XW)^{-1} &= (|v_1\rangle\langle e_1| + |v_2\rangle\langle e_2|)\left[\left(\textstyle\sum_i \lambda_i |u_i\rangle\langle v_i|\right)(|v_1\rangle\langle e_1| + |v_2\rangle\langle e_2|)\right]^{-1} \\
&= (|v_1\rangle\langle e_1| + |v_2\rangle\langle e_2|)\left[\lambda_1|u_1\rangle\langle e_1| + \lambda_2|u_2\rangle\langle e_2|\right]^{-1} \\
&= (|v_1\rangle\langle e_1| + |v_2\rangle\langle e_2|)\left(\tfrac{1}{\lambda_1}|e_1\rangle\langle u_1| + \tfrac{1}{\lambda_2}|e_2\rangle\langle u_2|\right) \\
&= \tfrac{1}{\lambda_1}|v_1\rangle\langle u_1| + \tfrac{1}{\lambda_2}|v_2\rangle\langle u_2| \\
&= X^+
\end{aligned}$$

The last equality follows from the previous section. This calculation shows that taking pseudo-inverse is the same as projecting on PCA axes and inverting the resulting full-rank matrix. This adds some intuition, because PCA is a straightforward dimensionality reduction.

3.3. Over-determined System of Equations. The other side of under-determined systems of equations is, of course, over-determined systems of equations. These are scenarios where there are more equations than unknowns. Importantly, an over-determined system gives different coefficients depending on which subset of equations you use. Going back to our example, if you had VIP tickets at 500 dollars and regular tickets at 100 dollars, and data from multiple weeks, you could choose any given weekend, and solve the problem just as you did in the earlier section. In fact, in this case, if you did do an SVD, you'd get a rank-2 matrix, and the pseudo-inverse would still give the same answer.

Imagine instead a situation where customers spend additional money on merchandise and refreshments. We still know for each day the number of regular customers, x_1, and VIP customers, x_2. And we know for each day, how much money

we made, y. Then we let β_1 and β_2 be the typical total spend (ticket plus merch plus refreshments) of regular and VIP customers. Can we solve for β_1 and β_2? If we found the right values of β, we may not have that $y = \beta_1 x_1 + \beta_2 x_2$ holds for any day in our data set, but it will still provide the best estimate.

Example

Over-determined System Suppose, as above, we know for each day how many of each type of ticket we sold, and the total revenue after merch and refreshments. If we have sales like this.
TODO: Fix
We can formulate this as a matrix problem like:

$$\begin{pmatrix} 12.6 \\ 13.7 \\ 14.0 \\ 16.4 \\ 15.8 \\ 10.7 \end{pmatrix} = \begin{pmatrix} 100 & 3 \\ 90 & 6 \\ 94 & 5 \\ 163 & 0 \\ 119 & 5 \\ 78 & 4 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

Then we can almost solve for $\vec{\beta}$, by taking the pseudo-inverse. Call this matrix $|y\rangle = X|\beta\rangle$. We come up with a solution $\hat{\beta} = X^+|y\rangle = X^T \left(XX^T\right)^{-1}$.

This example shouldn't be surprising or even interesting at this point. But I do want to point out that this is a more interesting problem than under-determined or full-rank square matrices. This is the first example that resembles a real world problem. By solving it, we learn something interesting about the customers; we probably already knew how much tickets themselves cost. The over-determined matrix arises from real-world problems more often. A possible explanation is that we're unlikely to build a model with more variables than data points. There are situations where a family of inputs is quite large, but in those cases, a dimensionality-reduction, like PCA is done explicitly as a pre-processing step. An old-fashioned rule of thumb (from before the era of big data) states that with n data points, you shouldn't try to model more than $\sqrt[3]{n}$ variables. So if you did have a 216×1000 matrix, it'd be better probably to use PCA to project onto the 6 most priniple axes, then set up a linear system on a 216×6 matrix. This may do a better job smoothing out the noise; and you'd see better performance on out-of-sample data points.

The solution you get from $\hat{\beta} = X^+|y\rangle = X^T \left(XX^T\right)^{-1}$ is the "best" solution in some abstract sense; the argument is something, something, orthogonal projection. But for the over-determined scenario has a clearer inpretation. Specifically, $\hat{\beta}$ is what we get when we minimize $\mathcal{L} = \sum_i \left(\beta_1 (x_i)_1 + \beta_2 (x_i)_2 - y_i\right)^2$. Written in matrix form, this is

$$\begin{aligned} \mathcal{L} &= \left(X|\beta\rangle - |y\rangle\right)^T \left(X|\beta\rangle - |y\rangle\right) \\ &= \left(\langle\beta|X^T - \langle y|\right) \left(X|\beta\rangle - |y\rangle\right) \\ &= \langle\beta|X^T X|\beta\rangle - \langle\beta|X^T|y\rangle - \langle y|X|\beta\rangle + \langle y|y\rangle \\ &= \langle\beta|X^T X|\beta\rangle - 2\langle y|X|\beta\rangle + \langle y|y\rangle \end{aligned}$$

Where the last equality follows from the fact that $\langle\beta|X^T|y\rangle$ is a scalar, so it's its own inverse. (Or equivalently, the commutativity of the inner product.) We minimize this using Lemma 1.4.1.

$$0 = \frac{\partial \mathcal{L}}{\partial \vec{\beta}} = 2X^T X|\beta\rangle - 2X^T|y\rangle$$

Solving this for $|\beta\rangle$, we get $|\beta\rangle = \left(X^T X\right)^{-1} X^T|y\rangle$, precisely as before.

$\mathcal{L}$ is the sum of the squares of the *residuals*, the difference between predicted and actual. And as such, this type of model is called *Least Squares Regression*, or sometimes *Ordinary Least Squares (OLS)*. You may ask if $\mathcal{L}$ is the correct thing to minimize. Minimizing $\mathcal{L}$ achieves the maximum likelihood estimation when we assume that the residuals are all drawn from the same normal curve, and that observations are independent. This assumption allows us to write covariance of residuals as a matrix $\sigma^2 I$. The maximum likelihood estimation is outside the scope of this book, but we can think of it like a best guess. The key assumption though is that we assume that the residuals are independent samples belonging to a normal distribution, and that the variance doesn't change for different inputs. If that still doesn't make sense, it suffices for now to know that linear regression is a simple technique that works well very often; as such, it should always be your first attempt.

One notation that we find useful is the "hat" notation. After we solve for $|\beta\rangle$, we say that the solution is "beta-hat" or $|\hat{\beta}\rangle$. This just means the predicted value of β. With this notation we may write $|\hat{y}\rangle = X|\hat{\beta}\rangle$ to mean the predicted values of $|y\rangle$ that we get by plugging $|\hat{\beta}\rangle$ into our original equation. It helps to keep this distinguished from the original $|y\rangle$. For example, now we can write the vector of residuals as $|\hat{y}\rangle - |y\rangle$. It then follows that $\mathcal{L} = \||\hat{y}\rangle - |y\rangle\|^2$.

We have that $|\hat{y}\rangle = X|\hat{\beta}\rangle = X\left(X^T X\right)^{-1} X^T|y\rangle$. We sometimes call $H := X\left(X^T X\right)^{-1} X^T$ the "hat matrix," because it puts a hat on $|y\rangle$: $|\hat{y}\rangle = H|y\rangle$. The remarkable thing about the hat matrix is that it equals

$$\begin{aligned} H &= X\left(\left(X^T X\right)^{-1} X^T\right) \\ &= \left(\textstyle\sum_i \lambda_i |u_i\rangle\langle v_i|\right)\left(\sum_j \frac{1}{\lambda_j}|v_j\rangle\langle u_j|\right) \\ &= \textstyle\sum_i |u_i\rangle\langle u_i| \end{aligned}$$

which is a projection matrix! (Aren't you glad we just went over those?) If X is a $m \times n$ matrix, representing m data points with n variables, then H is a projection matrix in $\mathbb{R}^m$. This is the sample space, where each dimension represents a data point. It's strange to think of this as a space. A vector in this space, like $|y\rangle$ represents the values observed across the data points. Each column in X is a vector in the sample space, representing all the sampled values for that input variable. The subspace spanned by $\{|u_i\rangle\}_i$ is the same as the space spanned by the columns X, by Theorem 1.5.3. So you can think of this subspace as the space of all vectors that can be written as a linear combination of the input variables. If $|y\rangle$ is in this span, then $|\hat{y}\rangle = |y\rangle$, meaning that $|y\rangle$ can be written exactly as a linear combination of the input variables, and there's no "almost-ness" to the almost inverse we compute. Otherwise, $|\hat{y}\rangle$ is the closest vector (to $|y\rangle$) in the span of inputs, by Theorem 3.1.2, which is why $|\hat{y}\rangle$ minimizes $\mathcal{L} = \||\hat{y}\rangle - |y\rangle\|^2$.

Exercises

3.3.1 Calculate $\hat{\beta}$ for Example 3.3.

3.3.2 An $m \times n$ matrix with $m < n$ is often surjective. Find an example of such a matrix that is not surjective.

3.3.3 Prove that $\mathcal{L}$ is minimize precisely when $\sqrt{\mathcal{L}}$ is minimized.

For the next two exercises call $|y\rangle = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ v_m \end{pmatrix}$ and $X = \begin{pmatrix} X_{1,1} & X_{1,2} \\ X_{2,1} & X_{2,2} \\ \vdots & \vdots \\ X_{m,1} & X_{m,2} \end{pmatrix}$.

3.3.4 In the case that $y_m = X_{m,1} + 2 \cdot X_{m,2}$, prove that $|\hat{\beta}\rangle = \left(X^T X\right)^{-1} X^T |y\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

3.3.5 Call $\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \left(X^T X\right)^{-1} X^T |y\rangle$. Prove that $\bar{y} = \beta_1 \bar{X_1} + \beta_2 \bar{X_2}$, where $\bar{y} = \frac{1}{m} \sum_i y_i$ and $\bar{X_j} = \frac{1}{m} \sum_i X_{i,j}$.

4. Least Squares Regression, Part 2

4.1. Variance. We start this section by solving Example 3.3 with a computer program.

OLS Regression Results

Dep. Variable:	revenue	R-squared (uncentered):	0.997
Model:	OLS	Adj. R-squared (uncentered):	0.995
Method:	Least Squares	F-statistic:	570.9
Date:	Sat, 16 Dec 2023	Prob (F-statistic):	1.22e-05
Time:	19:10:10	Log-Likelihood:	-8.1232
No. Observations:	6	AIC:	20.25
Df Residuals:	4	BIC:	19.83
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
regular_tickets	0.1130	0.007	16.900	0.000	0.094	0.132
vip_tickets	0.9309	0.172	5.403	0.006	0.453	1.409

Omnibus:	nan	Durbin–Watson:	2.641
Prob(Omnibus):	nan	Jarque–Bera (JB):	1.799
Skew:	1.325	Prob(JB):	0.407
Kurtosis:	3.422	Cond. No.	40.8

We read this as the estimated coefficient for the regular_tickets is \$113.00 and the estimated coefficient for vip_tickets is \$930.90. (Recall that the y values are in thousands of dollars.) Which in the context of the problem, this means that we estimate each regular ticketholder will spend 113 dollars (including ticket cost), while each VIP ticketholder will spend around 931 dollars. This is a valuable insight. Say you know that you're selling regular tickets for \$100 and VIP tickets for \$500. Now you know that VIPs will spend almost their entire ticket price on merchandise and refreshments, whereas regular ticket holders spend a much less significant amount. This kinda insight can inform how you market to VIPs, where you place merchandise, or even how you set prices. And it can be achieved without having to track the movement of every VIP throughout the venue.

We should be careful however about how we state our conclusions. Say the venue has an East entrance and a West entrance, with separate ticket stands and separate revenue, but virtually identical. If you run an analysis on the East entrance and somebody else runs an analysis on the West entrance, you'd be embarassed to confidently assert that the average VIP spends \$931 if the analyst on the West side concludes that a VIP spends \$510, barely more than their ticket price. But this could happen. By modeling this as a linear regression, we're asserting that $y = \beta_1 x_1 + \beta_2 x_2 + \varepsilon$, where ε is a random normal variable. And although we're guessing β_1 and β_2 based on the data we've observed, our estimate itself is subject to the randomness. So it stands to reason that if we measured on different days or even a different half of the venue on the same days, we'd expect to get different estimates for β.

But we do have a sense of the range of error on our estimate. We need to dive into stats a little bit. Variance gives a sense of spread and it's calculated as $\text{Var}(T) = \sum_i (t_i - E(T))^2$, where $\{t_i\}_i$ is a large sample of T, and $E(T) = \frac{1}{\#(T)} \sum_i t_i$. (Statisticians may object to the use of "large sample" instead of "population," but close enough is fine for this book.) Standard deviation is the square root of variance, and for normal variables, this is all we need to comment on the relative likelihood of different scenarios. Any observation drawn from a normal distribution is 95% likely to be within about 2 standard deviations of the center. In Table 4.1, the coefficient of vip_tickets is \$930.90 with a standard deviation of \$172. The 95% confidence interval is given as \$453 and \$1,409. This means that there's a 95% chance that whatever the West entrance measures for this coefficient will between \$453 and \$1,409, with a 95% probability. This is a pretty wide range. (With a keen eye, you may notice that the confidence interval is more than 2 standard deviations. This is because statsmodels uses a more-accurate t-test rather than a z-test. Don't worry about this difference, at around 30 or more data points, the difference is negligible.)

The 95% confidence interval is all points that are within 2 standard deviations of the mean, $|\hat{\beta}\rangle$; said another way, it's all points for which the *z-score*, $z = \frac{|\beta\rangle - |\hat{\beta}\rangle}{\sigma}$ is less than 2.

There's a multidimensional version of variance. For a multivariate normal distribution, with center $|\mu\rangle$ and variance Σ given as a matrix, the "Mahalanobis distance" of a point $|x\rangle$ is given by $\sqrt{\langle x-\mu|\Sigma^{-1}|x-\mu\rangle}$. In the one dimensional case, this equals the absolute value of the z-score. And this generalizes the z-score in that 95% of data drawn from the distribution have Mahalanobis distance less than 2. As before $|\mu\rangle = \frac{1}{n}\sum_{i=1}^{n}|x_i\rangle$, the average of points. And $\Sigma = \frac{1}{n}\sum_{i=1}^{n}\left(|x_i\rangle - |\mu\rangle\rangle\right)\left(\langle x_i| - \langle\mu|\rangle\right)$. We have a convenient identity here:

$$\begin{aligned}
\Sigma &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(|x_i\rangle - |\mu\rangle\rangle\right)\left(\langle x_i| - \langle\mu|\rangle\right) \\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(|x_i\rangle\langle x_i| - |\mu\rangle\langle x_i| - |x_i\rangle\langle\mu| + |\mu\rangle\langle\mu|\right) \\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}|x_i\rangle\langle x_i| - \tfrac{1}{n}\sum_{i=1}^{n}|\mu\rangle\langle x_i| - \tfrac{1}{n}\sum_{i=1}^{n}|x_i\rangle\langle\mu| + \tfrac{1}{n}\sum_{i=1}^{n}|\mu\rangle\langle\mu| \\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}|x_i\rangle\langle x_i| - |\mu\rangle\langle\mu| - |\mu\rangle\langle\mu| + |\mu\rangle\langle\mu| \\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}|x_i\rangle\langle x_i| - |\mu\rangle\langle\mu| \\
&= \mathrm{E}\left(|x\rangle\langle x|\right) - \mathrm{E}\left(|x\rangle\right)\mathrm{E}\left(\langle x|\right)
\end{aligned}$$

This is a rather quick treatment of variance. See Section 1 for more on this topic.

THEOREM 3.4.1. *$|\beta\rangle$ is a normal random variable with center $\left(X^TX\right)^{-1}X^T\,\mathrm{E}\left(|y\rangle\right)$ and variance $\left(X^TX\right)^{-1}\sigma^2$, where $\sigma^2 I = \mathrm{E}\left(|\hat{y}\rangle - |y\rangle\right)$. (Recall that the assumptions of OLS allow us to write covariance of residuals as a multiple of I.)*

PROOF. We start with $|\hat{\beta}\rangle = \left(X^TX\right)^{-1}X^T|y\rangle$. On the right-hand side, only $|y\rangle$ is random (being a function of ε, so that $\mathrm{E}\left(|\hat{\beta}\rangle\right) = \left(X^TX\right)^{-1}X^T\,\mathrm{E}\left(|y\rangle\right)$. (See Exercise 3.4.2.)

Now using the identy above we do some serious math here:

$$\begin{aligned}
\Sigma &= \left[\mathrm{E}\left(|\hat{\beta}\rangle\langle\hat{\beta}|\right)\right] - \left[\mathrm{E}\left(|\hat{\beta}\rangle\right)\mathrm{E}\left(\langle\hat{\beta}|\right)\right] \\
&= \left[\mathrm{E}\left(\left(X^TX\right)^{-1}X^T|y\rangle\langle y|X\left(X^TX\right)^{-1}\right)\right] - \left[\mathrm{E}\left(\left(X^TX\right)^{-1}X^T|y\rangle\right)\mathrm{E}\left(\langle y|X\left(X^TX\right)^{-1}\right)\right] \\
&= \left[\left(X^TX\right)^{-1}X^T\,\mathrm{E}\left(|y\rangle\langle y|\right)X\left(X^TX\right)^{-1}\right] - \left[\left(X^TX\right)^{-1}X^T|\,\mathrm{E}\left(y\rangle\right)\mathrm{E}\left(\langle y|\right)X\left(X^TX\right)^{-1}\right] \\
&= \left(X^TX\right)^{-1}X^T\left[\mathrm{E}\left(|y\rangle\langle y|\right) - \mathrm{E}\left(|y\rangle\right)\mathrm{E}\left(\langle y|\right)\right]X\left(X^TX\right)^{-1} \\
&= \left(X^TX\right)^{-1}X^T\left[\mathrm{E}\left(|\hat{y}\rangle - |y\rangle\rangle\right]X\left(X^TX\right)^{-1} \\
&= \left(X^TX\right)^{-1}X^T\left[\sigma^2 I\right]X\left(X^TX\right)^{-1} \\
&= \left(X^TX\right)^{-1}\sigma^2
\end{aligned}$$

The last equality follows from that the fact that scalars and I commute with all matrices, so that the penultimate equation is $\sigma^2\left(X^TX\right)^{-1}X^TX\left(X^TX\right)^{-1}$, allowing for some cancellations. □

The standard deviations given in Table 4.1 are given by the diagonal elements $\left(\left(X^TX\right)^{-1}\right)_{i,i}$. The idea is that if you start with the coefficient estimate $|\hat{\beta}\rangle$ and vary one dimension, $|\beta'\rangle = |\hat{\beta}\rangle + \varepsilon|e_i\rangle$, then the Mahalanobis distance becomes

$$\begin{aligned}\sqrt{\langle \varepsilon e_i|\Sigma^{-1}|\varepsilon e_i\rangle} &= \varepsilon\sqrt{\left(\Sigma^{-1}\right)_{i,i}} \\ &\approx \varepsilon\sqrt{\left(\Sigma_{i,i}\right)^{-1}} \\ &= \frac{\varepsilon}{\sqrt{\Sigma_{i,i}}} \\ &= \frac{\left\| |\beta'\rangle - |\hat{\beta}\rangle \right|}{\sqrt{\left((X^TX)^{-1}\right)_{i,i}}}\end{aligned}$$

At the end, this looks like an ordinary z score in one dimension, which justifies the confidence intervals reported by the regression program. The approximation in this calculation is doing some heavy lifting. The approximation is exact when X^TX is diagonal (which is the case that the input variables are independent). In practice this is often close enough to correct, but there are real cases where this isn't true. Imagine the same problem setup, but there are some "mega-event" weekends that we sell ten thousand tickets, instead of the usual hundred or so. We'd expect mega-event weekends to have large numbers for both VIP and regular weekends, introducing a strong correlation. When variables are correlated, our prediction of coefficients is also correlated. That is if we lowered the predicted spend from VIP customers, we'd increase the predicted spend from regular customers to make up the difference. But in these cases, a single-dimension confidence interval is less meaningful: It may actually be likely that VIP spend will be predicted significantly higher, but only in tandem with a lower regular spend. We talk about this more in the next subsection.

So far we've discussed variance on the coefficients themselves. And this is useful. One thing you may want to do with linear regressions is report the coefficient. In our example, it gives the relative "worth" of each type of customer, which can help with some business decisions. But another use of linear regression is "inference" or making out-of-sample predictions. This, for example, would be answering a question like, "If we sell 100 regular tickets and 10 VIP tickets, how much revenue should we expect to make? Given the numbers above, we model this as $\hat{y} = 113(100) + 930.9(10)$, or $y = 113(100) + 930.9(10) + \varepsilon$ to express the randomness. Or written in matrix form $y = \begin{pmatrix} 100 & 10 \end{pmatrix}\begin{pmatrix} 113 \\ 930.9 \end{pmatrix} + \varepsilon = \langle x|\hat{\beta}\rangle + \varepsilon$. As a point estimate, we'd expect revenue of $\$20,609$, but what kind of range can we give on this estimate?

Given that $|y\rangle = \langle x|\hat{\beta}\rangle + \varepsilon$, and both $|\hat{\beta}\rangle$ and ε are random, we have

$$\begin{aligned}\text{Var}\left(|y\rangle\right) &= \text{Var}\left(\langle x|\hat{\beta}\rangle\right) + \text{Var}(\varepsilon) \\ &= \langle x|\Sigma^2|x\rangle + \sigma^2 \\ &= \langle x|\left(X^TX\right)^{-1}\sigma^2|x\rangle + \sigma^2 \\ &= \sigma^2\left(\langle x|\left(X^TX\right)^{-1}|x\rangle + 1\right)\end{aligned}$$

Where Σ^2 is taken from Theorem 3.4.1. Some of the details here are covered in Exercise 3.4.1, but this calculation can also be revisited after reading Section 1. The interesting thing about this calculation is the common σ^2, the variance of the residuals. In the first case factors into the uncertainity in the coefficients, $|\hat{\beta}\rangle$, but in the second case, it adds constant variance into this single new prediction. The range given on coefficients is sometimes called "confidence interval," while an

equivalent interval given on a new prediction using $\sigma^2\left(\langle x|\left(X^TX\right)^{-1}|x\rangle+1\right)$ is called "prediction interval."

We look at some measures of variance, given a training set. So far, we've talked about residual sum of squares (RSS) $=\sum_i\left(y_i-\hat{y}_i\right)^2$. This measures how far off our predictions are. But when we talk about variance of y, we usually mean $\sum_i\left(y_i-\bar{y}\right)^2$, where $\bar{y}=\frac{1}{n}\sum_i y_i$, the average. We call this variance, the total sum of squares (TSS). Finally, we have a explained sum of squares (ESS) $=\sum_i\left(\hat{y}_i-\bar{y}\right)^2$. This name is justified by the next theorem.

THEOREM 3.4.2. *With ESS, RSS, and TSS as defined above,*

$$ESS+RSS=TSS.$$

PROOF. Throughout, it will be useful to write any sum of squares, $\sum_i x_i^2$, as an inner product $\langle x|x\rangle$. When needed, $|\bar{y}\rangle$ will be a vector with every entry equal to $\bar{y}$. Other vectors will match names but without the subscripts.

$$\begin{aligned}
\text{TSS} &= \langle y-\bar{y}|y-\bar{y}\rangle \\
&= \langle y-\hat{y}+\hat{y}-\bar{y}|y-\hat{y}+\hat{y}-\bar{y}\rangle \\
&= \langle y-\hat{y}|y-\hat{y}\rangle+\langle\hat{y}-\bar{y}|\hat{y}-\bar{y}\rangle+2\left(\langle y|\hat{y}\rangle-\langle\hat{y}|\hat{y}\rangle\right)-2\left(\langle y-\hat{y}|\bar{y}\rangle\right) \\
&= \text{RSS}+\text{ESS}+(0)+\left(\bar{y}\textstyle\sum_i\left(y_i-\bar{y}_i\right)\right) \\
&= \text{RSS}+\text{ESS}+(0)+(0)
\end{aligned}$$

The zeroing of the first parenthetical follows from Exercise 3.4.3, and the second follows from the fact that $\langle y-\hat{y}|$, the residuals, average to zero, and therefore sum to zero. □

Notice that both ESS and RSS must be positive. In the case that $y_i=\hat{y}_i$ for all i, this is a perfect prediction; here ESS=TSS and RSS=0. In the case that $y_i=\bar{y}$ for all i, then the model is totally informationless; here ESS=0 and RSS=TSS. We see that as the prediction gets better TSS gradually falls more into the ESS bucket, rather than the RSS bucket. ESS/TSS is a number between 0 and 1 that measures goodness of fit; this is called the *coefficient of determination*, or more commonly just R^2. ($\sqrt{R^2}$ also has a meaning, which justifies the exponent, but we don't discuss that here.) This is also reported in Figure 4.1.

4.2. Multicolinearity. If you ask a computer to do a linear regression, the computer will set off on calculating $|\hat{y}\rangle=X|\hat{\beta}\rangle=X\left(X^TX\right)^{-1}X^T|y\rangle$. But this exists only if X^TX is invertible. As mentioned earlier, this is invertible if and only if the columns of X span the domain. The columns of X fail to span the domain if there's some linear relationship among the columns (by Theorem 9.2.1). For example, let's say that you're pricing insurance policies. You have a matrix X where each row is a data point (customer) and the first column is a constant 1, added to many models; the second columns is an indicator for male, 1 if the customer is a man, and 0 if the customer is a woman; and the third column is an indicator for female. Call these columns "const", "is_male", and "is_female". Imagine that the dataset has every customer recorded as either male or female (due to software limitations or some policy), then we'd have the relationship const = is_male + is_female. In this case X^TX is not invertible. This is called *aliasing* or *multicollinearity*.

There's a plain English problem here. If you removed the const column and run a regression against the insurance cost of the customers, then you'd find that the coefficient for is_male would be the average cost for men, and is_female's coefficient would be the average cost for women. Let's say for this example, it's \$600 and \$400 respectively. If instead we dropped the is_male column and kept the other two. Then we'd see that const has coefficient 600 and is_female -200, so that men still get a prediction of 600 and women still get 400. But what happens if we include all three variables? We may see that we get const 0, is_male 600, and is_female 400. But we could just as validly get const 1000, is_male -400, and is_female -600. Or we could get const 99999, is_male -99399, and is_female -99599. Any of these are equally valid, so the linear regression doesn't know which to pick. There is still a pseudoinverse even for low-rank X, but without the formula $X\left(X^TX\right)^{-1}X^T$. However, we don't want to jump to using that in this case, because in this case there's not a perfect answer. It's true that we may get a decent overall model, but imagine a model reporting that being male saves you $99,399$ dollars from your insurance; it doesn't make sense! It's better to force the modeler to resolve the alias, usually by dropping an unneeded column.

Perhaps a more pernicious issue is the issue of near-aliasing. Imagine the same aliasing issue as above, but this time there's a single data point with both is_male and is_female equal to 0. This is no longer technically aliasing, but the problem persists. Imagine that the single non-binary data point had a cost of $10,000$ dollars, then our coefficients will be const 10000, is_male -9400, and is_female -9600. Again this presents a problem with explanability. As well there's a problem of variance; all the confidence intervals will be wide. There's a problem of over-leveraged points; the single non-binary point has too much influence on predictions. There's a stability problem; that is, if we run the same model next year, predictions could be widely different. Instability and lack of explanability are unaaceptable for insurance, but also many other applications.

Example

Near Aliasing Example Let's think about what's happening mathematically. Let X be an $(2n+1) \times 2$ matrix with every entry equal to 0 or 1, we'll assume that for the first $2n$ rows, the two columns match, and that exactly half of these will 1. The last row will differ. So for example this may look like this:

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

Assume that n is large. This matrix is almost-aliased because there's nearly a linear relationship between the two columns: Specifically the first column minus the second column is nearly 0.
Say that X has the SVD $\sum_i \lambda_i |u_i\rangle\langle v_i|$. Then $X^T X = \sum_i \lambda_i^2 |v_i\rangle\langle v_i|$. Meanwhile $X^T X$ is

$$\begin{pmatrix} n+1 & n \\ n & n \end{pmatrix}$$

Using the arguments from Section 4, we know that we can calculate eigenvalues and eigenvectors from here. We don't cover calculations much in this book, because when n isn't a variable, this can be done with a computer. But the reader may recall that it looks like:

$$0 = \begin{Vmatrix} n+1-\lambda & n \\ n & n-\lambda \end{Vmatrix} = (n+1-\lambda)(n-\lambda) - n^2 = \lambda^2 - (2n+1)\lambda + n$$

Solving this for λ using the quadratic equation, we get $\lambda = \frac{(2n+1) \pm \sqrt{4n^2+1}}{2}$. $\sqrt{4n^2+1}$ is between $2n = \sqrt{4n^2}$ and $2n+1 = \sqrt{4n^2+4n+1}$. So the two eigenvalues are a and $2n+b$, with both a and b between 0 and 1. Associated with a is a singular matrix

$$\begin{pmatrix} n+1-a & n \\ n & n-a \end{pmatrix}$$

with an eigenvector pretty close to $\langle 1/sqrt2, -1/sqrt2 \rangle$. Associated with $2n+b$ is a singular matrix

$$\begin{pmatrix} -n+1-b & n \\ n & -n-b \end{pmatrix}$$

with an eigenvector close to $\langle 1/sqrt2, 1/sqrt2 \rangle$. These two eigenvectors are $|v_1\rangle$ and $|v_2\rangle$. To get $\langle u_1|$ out of the equation $X = \lambda_1 |u_1\rangle\langle v_1| + \lambda_2 |u_2\rangle\langle v_2|$, we can compute $\frac{1}{\lambda_1} X |v_1\rangle$. If $\vec{v}_1 = \langle 1/sqrt2, -1/sqrt2 \rangle$, then $\frac{1}{\lambda_1} X |v_1\rangle = |e_{2n+1}\rangle$. Rather than worrying too much about constants, we know that $\vec{u}_1$ must be a unit vector in the direction of $\vec{e}_{2n+1}$ (because all other vector entries cancel. Because $\langle 1/sqrt2, -1/sqrt2 \rangle$ is only approximately $\vec{v}_1$, $|e_{2n+1}\rangle$ is only approximately $|u_1\rangle$. Similarly, $|u_2\rangle$ is approximately a unit vector point in the direction of the average of the two columns; the average of the two columns is 1/2 for the last entry, and the value from either column for the rest of the entries. We call this vector $|\bar{u}\rangle$. Let's also call $\vec{v}_{\text{sum}} = \langle 1/sqrt2, 1/sqrt2 \rangle$ and $\vec{v}_{\text{diff}} = \langle 1/sqrt2, -1/sqrt2 \rangle$
Recalling that the eigenvalues we calculated earlier are λ_i^2, we have that

OLS works best when the input variables are independent, and fails completely when there's a linaer relationship (perfect correlation). But there's a spectrum of correlation. Small categories, like non-binary, are usually grouped with larger buckets to avoid near-aliasing. But near-aliasing can also be the consequence of over-zealous feature engineering. Imagine you find that historical insurance cost is a good predictor of future cost. As such you make two variables, number of accidents, and size category of last accident (1-3). A coworker says that you should instead do average size category of all accidents. Another coworker says that you should total all losses and bucket (1-9). Another coworker prefers a lighter touch, and creates a variable called "is loss" which is 1 or 0 based on history. If you include all of these variables, you won't get a perfect alias, but you will get a ton of correlation. For example "is loss" will be correlated with number accidents and total losses, since most customers have no accidents. Average size category will equal last size category most of the time, since most people with accidents had only one accident. You may even find that total losses (1-9) is usually close to 3 x (number of accidents - 1) + (last accident size). Long story short here is that lots of correlation produces weaker models.

The lack of stability here is a big deal. Unlike the non-binary example above, it's not clear where the instability occurs. What data points will get a different prediction if we update the model? What happens to our model if we remove a few data points? You don't want to be on-the-job and change your model dramatically when you remove a few data points. You don't want to give a much different price to an important client than you did a week ago. It would be better if you used fewer variables. A priori, I don't know which ones are the best, but you could learn with a forward regression, for example. There's also an art to building features that are likely to be close to independent. For example, instead of including both average accident size and last accident size; you could do last accident size and "trend", which equals last accident size minus second-to-last accident size (or NULL if only one accident). This would just as correlated, but it moves all the instability to the trend variable, which can be more clearly analyzed. Today non-linear models (random forest or neural network) are common, and throwing in many highly-correlated variables is a standard operating here. The "include everything" strategy is less artful, but scales better to many models, updating frequently.

4.3. VIF and Cook's D. This subsection will be light on proofs; the curious reader is encouraged to search. We frame this subsection around understanding how variance depends on data points (rows of X) and input variables (columns of X).

To begin with, we present a formula without proof.

$$\operatorname{Var}\left(\hat{\beta}_j\right) = \frac{\sigma^2}{n \operatorname{Var}(X_j)} \cdot \frac{1}{1 - R_j^2}$$

The left-hand side is just the variance of the estimate for β_j. On the right-hand side, σ^2 as usual is the variance of the residuals, n is the number of data points, $\operatorname{Var}(X_j)$ is the variance of the j-th column of X (viewed as n observations of a random scalar variable), and R_j^2 is the coefficient of determination when we fit X_j as a linear regression on the other columns of X. (Note that we've again been careless about degrees of freedom, and this equation is only approximate.)

This formula tells us that variance of a coefficient estimate varies directly with the variance of the residuals, and inversely with the number of observations and the variance of the input variable itself. We already knew about the variance of residuals. The n in the denominator says that coefficients will become less variable with more data, which is expected. (Notice that the residual part of the prediction interval does not shrink with n though.) The variance of the column in the denominator makes some sense: Imagine, for example, I had an input variable representing length in feet, and I had a coefficient confidence interval of $(9, 15)$; if I replaced this with the same variable in yards, I should have a coefficient confidence interval of $(3, 5)$. The only difference is the scale, and variance measures this scale.

The second multiplicand, $\frac{1}{1-R_j^2}$, is the most surprising part of this formula. It is called the *Variance Inflation Factor (VIF)*. This inflation factor is minimized when $R_j^2 = 0$. This is the case that the X_j is independent from the other columns of X; that is, there's no relationship. On the other end of the spectrum, if there was aliasing, then X_j could be prefectly predicted by the other columns of X, and the variance would be something like infinite. The VIF tells you how much the variance of the the j-th coefficient is inflated due to the correlation between the variables in X. If you think you have multicollinearity problems, you can use the VIF of each variable (computed for you by a computer), to determine which variables you should be most concerned about. Some practicianers will use rules of thumb, like "if VIF > 5, then the variable is highly correlated to others."

Sometimes high variance in $\vec{\beta}$ are due to individual points, as in our example with a single non-binary data point. Cook's Distance or *Cook's D* provides a very simple definition for how much a single point can change our prediction. Call $\hat{y}_j$ the predicted value for the j-th data point, and $\hat{y}_{j(i)}$ the predicted value for the j-th data point when the i-th data point is removed from the data set. The define

$$D_i := \frac{\sum_j \left(\hat{y}_j - \hat{y}_{j(i)}\right)^2}{r\sigma^2}$$

Where r is the number of input variables. This is very intuitive; it just measures how much the predictions change by including the i-th data point. The denominator is a sort of normalizing factor that gives comparable values of D for different types of models. When normalized, we can use rules of thumb, like "remove any data point with $D_i > 4$." Generally, we can look at data points with the highest D_i as the most influential, and consider removing them. Or we can just use this as an additional piece of information. Imagine you were building a model to predict outcomes of American football games. If the most influential data point was a large upset that occurred because the quarterback was injured, then this may tell you that injuries need to be a variable in your model.

The nice thing about Cook's D is that there's this identity that I won't prove:

$$D_i = \frac{\varepsilon_i^2}{r\sigma^2}\left(\frac{h_i i}{(1-h_i i)^2}\right)$$

Where $h_i i$ is the i-th diagonal entry of the hat matrix that we have to compute to solve for $\hat{\beta}$ anyway. This allows us to get all the values of Cook's D fairly inexpensively.

4.4. Closing remarks. One note is that the least squares regression is among many models. But whereas many complex or weak-assumption models, like neural networks, ARIMA, or MCMC, generally solve an optimization via some variation of gradient descent, least squares regression uses a matrix solution. Solving this is fast for computers, even with large datasets.

Finally, it should be mentioned that this book is meant to motivate and explore the mathematics that go into a regression. But this does not make you a modeling expert. There are many considerations that go into modeling, like model choice, data collection, data quality, outlier modeling, train/test splits, hyperparameter optimization, over-fitting, winner's curse, model stability, explanability, acceptability, and model matainence. These things are outside the scope of this book. And although many of these topics are less mathematically complex than the formula itself, these topics can take years to master. For the practicianer, this book should be supplemented with modeling books and practice.

Exercises

3.4.1 Let $|u\rangle$ be a random vector, the expected value of $|u\rangle$ is itself a vector, $|\mathrm{E}(u)\rangle$, obtained by taking the expected value of each component, and the variance is a matrix, call Σ^2. (See section 1 for details.) Let $|v\rangle$ be a non-random (constant) vector. Then $\langle u|v\rangle$ is a scalar, and so expected value and variance of this are also scalars. Show the following:
- a $\mathrm{E}\left(|u\rangle + |v\rangle\right) = |\mathrm{E}(u)\rangle + |v\rangle$.
- b $\mathrm{E}\left(\langle u|v\rangle\right) = \langle \mathrm{E}(u)|v\rangle$.
- c $\mathrm{Var}\left(\langle u|v\rangle\right) = \langle v|\Sigma^2|v\rangle$.
- d $\mathrm{Var}\left(|u\rangle + |u'\rangle\right) = \mathrm{Var}(|u\rangle) + \mathrm{Var}(|u'\rangle)$. And in particular if $|u'\rangle$ is constant (non-random), then $\mathrm{Var}\left(|u\rangle + |u'\rangle\right) = \mathrm{Var}(|u\rangle)$.

3.4.2 For a non-random matrix X, prove that $\mathrm{E}\left(X|v\rangle\right) = X\,\mathrm{E}\left(|v\rangle\right)$.

3.4.3 Prove that $\langle y|\hat{y}\rangle = \langle \hat{y}|\hat{y}\rangle$, using the definition $|\hat{y}\rangle = X|\hat{\beta}\rangle = X\left(X^TX\right)^{-1}X^T|y\rangle$.

3.4.4 We will explore inverses for almost diagonal matrices in a very special case: Given an $n \times n$ matrix, M with $|M_{i,j}| = \begin{cases} 0 & ; \quad i < j \\ 1 & ; \quad i = j \\ \leqslant \varepsilon/m & ; \quad i > j \end{cases}$ Prove that $\left|\left(M^{-1}\right)_{i,j}\right| = \begin{cases} 0 & ; \quad i < j \\ 1 & ; \quad i = j \\ \leqslant \varepsilon/\left(1-\varepsilon\right) & ; \quad i > j \end{cases}$. (Hint: Write $M = I + E$, then $M^{-1} = I + E + E^2 + \cdots$. Argue that $E^m = 0$.)

3.4.5 Figure 4.1 reports AIC, which takes into account parsimony of the models. Look this up. How is this defined, and how those that relate to R^2 for ordinary least squares?

3.4.6 Return to the example of VIP tickets. Imagine that the venue has some days where they sell tens of tickets and other days where they sell thousands of tickets. As a result the number of VIP tickets they sell is highly correlated with the number of regular tickets they sell. How can you model this to account for the correlation?

5. Gram-Schmidt and QR factorizations

5.1. Gram-Schmidt Algorithm.

5.2. QR factorization.

5.3. Least Squares Regression as Projection.

6. Computer Programming, Regression

This section contains some questions to run with a computer. Some of the problems will require the reader to lookup commands. Examples will be given in Python, but feel free to use the language of your choice.

Python has two common packages for linear regression, sklearn (sklearn.linear_model.LinearRegression) and statsmodels. sklearn is far more widely used, and better supports more modern modeling best practices. But this author prefers statsmodels for simple use cases, primarily due to its friendly readouts.

```
import pandas as pd
import statsmodels.api as sm

df = pd.DataFrame(
    data = {
        "revenue": [14.1, 14.8, 15.1, 18.0, 20.0, 11.8],
        "regular_tickets": [100, 90, 94, 163, 119, 78],
        "vip_tickets": [3, 6, 5, 0, 5, 4],
    }
)

model = sm.OLS(df["revenue"], df[["regular_tickets", "vip_tickets"]])
results = model.fit()

print(results.summary())
```

This chapter's contents are discussed entirely through exercises. These exercises use the diabetes dataset, which can be loaded with sklearn.

```
from sklearn import datasets
X, y = datasets.load_diabetes(return_X_y = True , as_frame = True)
```

If you're using another language: This is a small dataset so you can copy as a CSV or literals, after finding it online or executing the above code in a Python environment.

Exercises

3.6.1 Using the diabetes data, fit a logistic regression. What is the 95 percent confidence interval for each coefficient? What is R^2?

3.6.2 Build a *forward regression*: Using the diabetes dataset, first build 10 single-variable regressions, one for each variable. Determine which of these models has the best R^2. Keep the variable used for that model, call it X. Then build 9 two-variable regressions, with inputs of X and each of the other variables. Determine which of these pairs of variables produces the best R^2. Then repeat to find the best three variable model that includes these two variables. Continue until you've included all 10 variables. Graph the best R^2 for each of 10 stages.

Forward regression is a greedy algorithm for deciding which variables are the most predictive. Here we use R^2, but we could also have used some

other metric, like AIC. For the diabetes dataset, the best R^2 is achieved when all variables are included. But this usually isn't the case in practice. Forward regression tries to find the optimal subset of variables to include. However, what it finds isn't optimal, just close. I've found that usually one forward regression followed by one backwards regression on the result does a good job.

3.6.3 Perform *bootstrapping* by choosing a random 70% of diabetes data, and running a regression. Do this $10,000$ times and record the coefficients. For each coefficient give a confidence interval by reporting the range that the coefficient falls into 90% of the time. Compare these to the confidence intervals given by the regression. Note: You can use a similar technique to get a range on R^2.

3.6.4 Using the diabetes dataset, model a regression as $y = \beta_{age} X_{age}$ to get $\hat{y}_1$. Then model $y - \hat{y}_1 = \beta_{bmi} X_{bmi}$, setting $\hat{y}_2 = \hat{y}_1 + \hat{\beta}_{bmi} x_{bmi}$. Then model $y - \hat{y}_2 = \beta_{age} X_{age}$, setting $\hat{y}_3 = \hat{y}_2 + \hat{\beta}_{age} X_{age}$. Then repeat modeling the residuals on BMI to get $\hat{y}_4$. Then get $\hat{y}_4$ using age, $\hat{y}_5$ on BMI, etc. Do this until you get convergence. Compare the final values of β_{age} and β_{bmi} to the coefficients you get when you regress $y = \beta_{age} X_{age} + \beta_{bmi} X_{bmi}$.

3.6.5 Fit β_{age} on the diabetes dataset using Gram-Schmidt, as described in the previous section. Verify that this is the same as what you get when you perform a regression.

3.6.6 Make a table with covariances between all the variables. Comment on the results. (Seaborn is my preferred package for this.)

3.6.7 Search Kaggle for the diabetes dataset to see how others are using this.

7. Legendre polynomials

CHAPTER 4

[WIP] Generalized Regression

1. **Polynomial Regression**
2. **Logistic**
3. **GLM**
4. **LOESS**

CHAPTER 5

[WIP] Definite Matrices

1. **Definite matrices**
2. **Quadratic programming**
3. **Partial second derivative test**
4. **Ridge Regression**

CHAPTER 6

[WIP] Symmetric Matrices

1. Symmetric Matrices

2. Inner products

3. Spectral Theorem

4. Taylor series and smooth functions

5. Functions on Non-symmetric matrices

CHAPTER 7

[WIP] Stats

1. Covariance matrix

2. Multi-dimensional normal

3. Moment generating function

4. Markov Chain

5. PageRank

6. Linear discriminant analysis

CHAPTER 8

[WIP] SVM

CHAPTER 9

Review

1. Orthogonal Matrices

This section is under construction. In the meantime, `https://www.nagwa.com/en/explainers/476190725258/` is a good source to review this topic.

THEOREM 9.1.1. *A square matrix has orthonormal rows if and only if it has orthonormal columns. We call such matrices* orthogonal.

THEOREM 9.1.2. *The inverse of a orthogonal matrix is its transpose.* $Q^{-1} = Q^T$.

2. Spans and Bases

This section is under construction. In the meantime, `https://math.libretexts.org/Bookshelves/Linear_Algebra/Book%3A_Linear_Algebra_(Schilling_Nachtergaele_and_Lankham)/05%3A_Span_and_Bases` is a good source.

THEOREM 9.2.1. *If n vectors,* $\{|v_i\rangle\}_i$ *in* $\mathbb{R}^n$ *do not span* $\mathbb{R}^n$*, then there is some linear relationship among the vectors. That is* $\sum_i \lambda_i |v_i\rangle = 0$*, where not all* $\lambda_i = 0$*. Similarly, if m vectors belong to a m-dimensional subspace, then there's a linear relationship.*

3. Change of Basis

This section is under construction. In the meantime, `https://eli.thegreenplace.net/2015/change-of-basis-in-linear-algebra/` is a good source.

4. Bra-Ket Notation

This section is under construction. In the meantime, `https://www.mathpages.com/home/kmath638/kmath638.htm` is a good source.

5. Misc

This section is under construction.

THEOREM 9.5.1. *For any set of orthonormal vectors,* $\{|v_i\rangle\}_{i=1}^k$ *in* $\mathbb{R}^n$*, there exists a basis,* $\{|w_i\rangle\}_{i=1}^n$ *such that* $|v_i\rangle = |w_i\rangle$ *for* $i = 1..k$*. We call this* Extending the Basis.

THEOREM 9.5.2. *For any subspace W and any vector* $|v\rangle$*,* $|v\rangle$ *can be written uniquely as* $|v\rangle = |v_\parallel\rangle + |v_\perp\rangle$*, where* $|v_\parallel\rangle \in W$ *and* $\langle w|v_\perp\rangle = 0$ *for all* $w \in W$*. Moreover,* $|v_parallel\rangle$ *is the unique vector in W that minimizes* $\||v\rangle - |v_\parallel\rangle\|$ *with minimum value* $\|v_\perp\|$.

THEOREM 9.5.3. *Given two vectors, $\vec{u}$, $\vec{v}$, with angle between them θ, the inner product can be expressed as:*

$$\langle u|v\rangle = \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos\theta$$